



SUSE Linux Enterprise High Availability Extension 15 SP3

管理ガイド

管理ガイド

SUSE Linux Enterprise High Availability Extension 15 SP3

著者: Tanja Roth、Thomas Schraitle

このガイドは、SUSE® Linux Enterprise High Availability Extensionを使用してクラスターを設定、構成、および管理する必要がある管理者を対象にしています。構成と管理をすばやく効率的に行うため、この製品にはグラフィカルユーザインタフェースとコマンドラインインタフェース(CLI)の両方が備わっています。重要なタスクを実行するために、このガイドには両方のアプローチが説明されています。これにより、ニーズを満たす適切なツールを選択できるようになります。

発行日: 2025 年 3 月 20 日

<https://documentation.suse.com> 

Copyright © 2006–2025 SUSE LLC and contributors. All rights reserved.

この文書は、GNUフリー文書ライセンスのバージョン1.2または(オプションとして)バージョン1.3の条項に従って、複製、配布、および/または改変が許可されています。ただし、この著作権表示およびライセンスは変更せずに記載すること。ライセンスバージョン1.2のコピーは、「GNUフリー文書ライセンス」セクションに含まれています。

SUSEの商標については、<http://www.suse.com/company/legal/> を参照してください。その他の第三者のすべての商標は、各社の所有に帰属します。商標記号(®、™など)は、SUSEおよび関連会社の商標を示します。アスタリスク(*)は、第三者の商標を示します。

本書のすべての情報は、細心の注意を払って編集されています。しかし、このことは絶対に正確であることを保証するものではありません。SUSE LLC、その関係者、著者、翻訳者のいずれも誤りまたはその結果に対して一切責任を負いかねます。

目次

序文 xvi

- 1 利用可能なマニュアル xvi
- 2 ドキュメントの改善 xvi
- 3 マニュアルの表記規則 xvii
- 4 サポート xix
SUSE Linux Enterprise High Availability Extensionのサポートステートメント xix • 技術プレビュー xx

I インストール、セットアップ、およびアップグレード 1

1 製品の概要 2

- 1.1 拡張としての提供 2
- 1.2 主な特長 2
広範なクラスタリングシナリオ 3 • 柔軟性 3 • ストレージとデータレプリケーション 3 • 仮想化環境のサポート 4 • ローカル、メトロ、およびGeoクラスタのサポート 4 • リソースエージェント 5 • ユーザフレンドリな管理ツール 5
- 1.3 利点 6
- 1.4 クラスタ設定: ストレージ 9
- 1.5 アーキテクチャ 11
アーキテクチャ層 11 • プロセスフロー 14

2 システム要件と推奨事項 15

- 2.1 ハードウェア要件 15
- 2.2 ソフトウェアの必要条件 16
- 2.3 ストレージ要件 17
- 2.4 その他の要件と推奨事項 18

3 High Availability Extensionのインストール 20

- 3.1 手動インストール 20
- 3.2 AutoYaSTによる大量インストールと展開 20

4 YaSTクラスタモジュールの使用 23

- 4.1 用語の定義 23
- 4.2 YaSTクラスタモジュール 25
- 4.3 通信チャンネルの定義 27
- 4.4 認証設定の定義 32
- 4.5 すべてのノードへの設定の転送 33
 - YaSTによるCsync2の設定 34 • Csync2を使用した変更内容の同期 35
- 4.6 クラスタノード間の接続ステータスの同期 37
- 4.7 サービスの設定 38
- 4.8 ノードごとにクラスタをオンラインにする 39

5 クラスタアップグレードとソフトウェアパッケージの更新 41

- 5.1 用語集 41
- 5.2 最新の製品バージョンへのクラスタアップグレード 42
 - SLE HAおよびSLE HA Geoでサポートされるアップグレードパス 43 • アップグレード前に必要な準備 48 • クラスタオフラインアップグレード 48 • クラスタローリングアップグレード 51
- 5.3 クラスタノード上のソフトウェアパッケージの更新 55
- 5.4 詳細の参照先 56

II 設定および管理 57

6 設定および管理の基本事項 58

- 6.1 ユースケースのシナリオ 58

- 6.2 クォーラムの判断 59
 - グローバルクラスタオプション 60 • グローバルオプションno-quorum-policy 60 • グローバルオプションstonith-enabled 61 • 2ノードクラスタのCorosync設定 62 • NノードクラスタのCorosync設定 62
- 6.3 クラスタリソース 63
 - リソースの管理 63 • サポートされるリソースエージェントクラス 64 • リソースのタイプ 66 • リソーステンプレート 67 • 高度なリソースタイプ 67 • リソースオプション(メタ属性) 70 • インスタンス属性(パラメータ) 73 • リソース操作 75 • タイムアウト値 77
- 6.4 リソース監視 78
- 6.5 リソースの制約 79
 - 制約のタイプ 80 • スコアと無限大 83 • リソーステンプレートと制約 83 • フェールオーバーノード 84 • フェールバックノード 85 • 負荷インパクトに基づくリソースの配置 86 • タグの使用によるリソースのグループ化 89
- 6.6 リモートホストでのサービスの管理 89
 - 監視プラグインを使用したリモートホストでのサービスの監視 90 • pacemaker_remoteを使用したリモートノードでのサービスの管理 91
- 6.7 システムヘルスの監視 92
- 6.8 詳細の参照先 94

7 Hawk2を使用したクラスタリソースの設定と管理 95

- 7.1 Hawk2の要件 95
- 7.2 ログイン 96
- 7.3 Hawk2の概要: 主な構成要素 97
 - 左のナビゲーションバー 98 • 最上位の行 99
- 7.4 グローバルクラスタオプションの設定 99
- 7.5 クラスタリソースの設定 101
 - 現在のクラスタ設定の表示(CIB) 102 • ウィザードを使用したリソースの追加 103 • 単純なリソースの追加 104 • リソーステンプレートの追

	加 105 • リソースの変更 106 • STONITHリソースの追加 108 • クラスタリソースグループの追加 109 • クローンリソースの追加 111 • マルチステートリソースの追加 112 • タグの使用によるリソースのグループ化 114 • リソース監視の設定 115
7.6	制約の設定 117
	場所制約の追加 117 • コロケーション制約の追加 119 • 順序制約の追加 121 • 制約のためにリソースセットを使用する 123 • 詳細の参照先 124 • リソースフェールオーバーノードの指定 125 • リソースフェールバックノードの指定(リソースの固着性) 126 • 負荷インパクトに基づくリソース配置の設定 127
7.7	クラスタリソースの管理 130
	リソースとグループの編集 130 • リソースの開始 130 • リソースのクリーンアップ 131 • クラスタリソースの削除 132 • クラスタリソースの移行 133
7.8	クラスタの監視 134
	単一クラスタの監視 134 • 複数のクラスタの監視 135
7.9	バッチモードの使用 138
7.10	クラスタ履歴の表示 142
	ノードまたリソースの最近のイベントの表示 142 • クラスタレポートのための履歴エクスプローラーの使用 143 • 履歴エクスプローラーの遷移詳細の表示 146
7.11	クラスタヘルスの確認 147
8	クラスタリソースの設定と管理(コマンドライン) 149
8.1	crmsh - 概要 149
	ヘルプの表示 150 • crmshのサブコマンドの実行 151 • OCFリソースエージェントに関する情報の表示 153 • crmshのシェルスクリプトの使用 154 • crmshのクラスタスクリプトの使用 155 • 設定テンプレートの使用 158 • シャドーイング設定のテスト 160 • 設定の変更のデバッグ 161 • クラスタダイアグラム 161
8.2	Corosync設定の管理 161

- 8.3 クラスタリソースの設定 162
 - ファイルからのクラスタリソースのロード 163 • クラスタリソースの作成 163 • リソーステンプレートの作成 164 • STONITHリソースの作成 165 • リソース制約の設定 166 • リソースフェールオーバーノードの指定 169 • リソースフェールバックノードの指定(リソースの固着性) 169 • 負荷インパクトに基づくリソース配置の設定 170 • リソース監視の設定 172 • クラスタリソースグループの設定 173 • クローンリソースの設定 173
- 8.4 クラスタリソースの管理 175
 - クラスタリソースの表示 175 • 新しいクラスタリソースの開始 176 • クラスタリソースの停止 176 • リソースのクリーンアップ 177 • クラスタリソースの削除 177 • クラスタリソースの移行 178 • リソースのグループ化/タグ付け 178 • ヘルスステータスの取得 179
- 8.5 cib.xmlから独立したパスワードの設定 179
- 8.6 履歴情報の取得 180
- 8.7 詳細の参照先 181
- 9 リソースエージェントの追加または変更 182**
 - 9.1 STONITHエージェント 182
 - 9.2 OCFリソースエージェントの作成 182
 - 9.3 OCF戻りコードと障害回復 183
- 10 フェンシングとSTONITH 186**
 - 10.1 フェンシングのクラス 186
 - 10.2 ノードレベルのフェンシング 187
 - STONITHデバイス 187 • STONITHの実装 188
 - 10.3 STONITHのリソースと環境設定 189
 - STONITHリソースの設定例 189
 - 10.4 フェンシングデバイスの監視 192
 - 10.5 特殊なフェンシングデバイス 193
 - 10.6 基本的な推奨事項 195

10.7	詳細の参照先	196
11	ストレージ保護とSBD	197
11.1	概念の概要	197
11.2	SBDの手動設定の概要	199
11.3	要件	199
11.4	SBDデバイスの数	200
11.5	タイムアウトの計算	201
11.6	ウォッチドッグのセットアップ	202
	ハードウェアウォッチドッグの使用	203
	ソフトウェアウォッチドッグ (softdog)の使用	204
11.7	デバイスでのSBDの設定	205
11.8	ディスクレスSBDの設定	211
11.9	SBDとフェンシングのテスト	213
11.10	ストレージ保護のための追加メカニズム	214
	sg_persistリソースの設定	214
	sfexを使用した排他的なストレージアクティビティの保証	216
11.11	詳細の参照先	217
12	QDeviceとQNetd	218
12.1	概念の概要	218
12.2	要件と前提条件	220
12.3	QNetdサーバのセットアップ	220
12.4	QDeviceクライアントをQNetdサーバに接続する	221
12.5	ヒューリスティックスを使用したQDeviceの設定	221
12.6	クォーラムステータスの確認と表示	222
12.7	詳細の参照先	224

13 アクセス制御リスト 225

- 13.1 要件と前提条件 225
- 13.2 概念の概要 226
- 13.3 クラスタでのACLの使用の有効化 226
- 13.4 読み込み専用monitor役割の作成 227
Hawk2による読み込み専用monitor役割の作成 227 • crmshによる読み込み専用monitor役割の作成 229
- 13.5 ユーザの削除 230
Hawk2によるユーザの削除 230 • crmshによるユーザの削除 231
- 13.6 既存の役割の削除 231
Hawk2による既存の役割の削除 231 • crmshによる既存の役割の削除 232
- 13.7 XPath式によるACLルールの設定 232
- 13.8 短縮によるACLルールの設定 233

14 ネットワークデバイスボンディング 235

- 14.1 YaSTによるボンディングデバイスの設定 235
- 14.2 ボンディングスレーブのホットプラグ 238
- 14.3 詳細の参照先 240

15 負荷分散 241

- 15.1 概念の概要 241
- 15.2 Linux仮想サーバによる負荷分散の設定 243
Director 243 • ユーザスペースのコントローラとデーモン 243 • パケット転送 244 • スケジューリングアルゴリズム 244 • YaSTによるIP負荷分散の設定 245 • 追加設定 250
- 15.3 HAProxyによる負荷分散の設定 250
- 15.4 詳細の参照先 253

16 Geoクラスタ(マルチサイトクラスタ) 255

17 保守タスクの実行 256

- 17.1 クラスタノードを切断する意味 256
- 17.2 保守タスクのためのさまざまなオプション 258
- 17.3 保守作業の準備と終了 259
- 17.4 クラスタを保守モードにする 259
- 17.5 ノードを保守モードにする 260
- 17.6 ノードをスタンバイモードにする 260
- 17.7 リソースを保守モードにする 261
- 17.8 リソースを非管理対象モードにする 262
- 17.9 保守モード中のクラスタノードの再起動 263

III ストレージとデータレプリケーション 264

18 分散ロックマネージャ(DLM:Distributed Lock Manager) 265

- 18.1 DLM通信のプロトコル 265
- 18.2 DLMクラスタリソースの設定 265

19 OCFS2 268

- 19.1 特長と利点 268
- 19.2 OCFS2のパッケージと管理ユーティリティ 269
- 19.3 OCFS2サービスとSTONITHリソースの設定 270
- 19.4 OCFS2ボリュームの作成 271
- 19.5 OCFS2ボリュームのマウント 274
- 19.6 Hawk2でのOCFS2リソースの設定 275
- 19.7 OCFS2ファイルシステム上でクォータを使用する 276

19.8	詳細の参照先	277
20	GFS2	278
20.1	GFS2パッケージおよび管理ユーティリティ	278
20.2	GFS2サービスとSTONITHリソースの設定	279
20.3	GFS2ボリュームの作成	280
20.4	GFS2ボリュームのマウント	281
21	DRBD	283
21.1	概念の概要	283
21.2	DRBDサービスのインストール	284
21.3	DRBDサービスの設定	285
	手動によるDRBDの設定 • YaSTによるDRBDの設定 • DRBDリ ソースの初期化とフォーマット	286 • 288 • 291
21.4	DRBD 8から DRBD 9への移行	292
21.5	スタックされたDRBDデバイスの作成	293
21.6	リソースレベルのフェンシングの使用	295
21.7	DRBDサービスのテスト	295
21.8	DRBDデバイスの監視	297
21.9	DRBDのチューニング	298
21.10	DRBDのトラブルシュート	298
	設定 • ホスト名 • TCPポート7788 • DRBDデバイスが再 起動後に破損した	298 • 299 • 299 • 300
21.11	詳細の参照先	300
22	Cluster Logical Volume Manager (Cluster LVM)	301
22.1	概念の概要	301

- 22.2 クラスタLVMの設定 302
クラスタリソースの作成 302 • シナリオ: SAN上でiSCSIを使用するCluster LVM 304 • シナリオ: DRBDを使用したCluster LVM 308

- 22.3 有効なLVM2デバイスの明示的な設定 310

- 22.4 Mirror LVからCluster MDへのオンラインマイグレーション 311
マイグレーション前のサンプルセットアップ 311 • Mirror LVをCluster MDにマイグレートする 313 • マイグレーション後のサンプルセットアップ 314

- 22.5 詳細の参照先 315

23 クラスタマルチデバイス(Cluster MD) 316

- 23.1 概念の概要 316
- 23.2 クラスタ化されたMD RAIDデバイスの作成 316
- 23.3 リソースエージェントの設定 318
- 23.4 デバイスの追加 318
- 23.5 一時的に障害が発生したデバイスの再追加 319
- 23.6 デバイスの削除 319
- 23.7 障害復旧サイトで通常のRAIDとしてCluster MDをアセンブルする障害復旧サイトで通常のRAIDとしてCluster MDのアセンブルする 320

24 Sambaクラスタリング 321

- 24.1 概念の概要 321
- 24.2 基本的な設定 323
- 24.3 Active Directoryドメインへの追加 326
- 24.4 クラスタ対応Sambaのデバッグとテスト 327
- 24.5 詳細の参照先 329

25 Rear (Relax-and-Recover)による障害復旧 330

- 25.1 概念の概要 330
障害復旧プランの作成 330 • 障害復旧とは 331 • Rearによる障害復旧 331 • Rearの要件 331 • Rearバージョンの更新 331 • Btrfsに伴う制限事項 332 • シナリオとバックアップのツール 333 • 基本手順 334
- 25.2 Rearおよびバックアップソリューションのセットアップ 334
- 25.3 復旧インストールシステムの作成 336
- 25.4 復旧プロセスのテスト 337
- 25.5 障害からの復旧 337
- 25.6 詳細の参照先 338

IV 付録 339

A トラブルシューティング 340

- A.1 インストールと最初のステップ 340
- A.2 ログ記録 341
- A.3 リソース 342
- A.4 STONITHとフェンシング 344
- A.5 履歴 345
- A.6 Hawk2 346
- A.7 その他 346
- A.8 詳細の参照先 349

B 命名規則 350

C クラスタ管理ツール(コマンドライン) 351

D rootアクセスなしでのクラスタレポートの実行 353

- D.1 ローカルユーザアカウントの作成 353
- D.2 パスワード不要のSSHアカウントの設定 354

D.3 **sudo**の設定 356

D.4 クラスタレポートの生成 358

用語集 360

E **GNU licenses** 367

序文

1 利用可能なマニュアル

オンラインマニュアル

本製品のオンラインマニュアルは、<https://documentation.suse.com/#sle-ha> で入手できます。様々な形式のマニュアルをブラウズまたはダウンロードできます。

他の製品のオンラインマニュアルは、<https://documentation.suse.com/> で検索してください。



注記: 最新のアップデート

最新のマニュアルアップデートは、通常、英語版マニュアルで入手できます。

リリースノート

リリースノートは<https://www.suse.com/releasesnotes/> を参照してください。

ご使用のシステムで

オフラインで利用するには、システムの `/usr/share/doc` にインストールされたマニュアルを確認してください。「マニュアルページ」には、多くのコマンドについても詳しく説明されています。説明を表示するには、`man` コマンドに確認したいコマンドの名前を付加して実行してください。システムに `man` コマンドがインストールされていない場合は、`sudo zypper install man` コマンドでインストールします。

2 ドキュメントの改善

このドキュメントに対するフィードバックや貢献を歓迎します! 次のチャンネルがあります。

サービス要求およびサポート

ご使用の製品に利用できるサービスとサポートのオプションについては、<http://www.suse.com/support/> を参照してください。

サービス要求を開くには、SUSE Customer Centerでの購読が必要です。<https://scc.suse.com/support/requests> からログインして新規作成をクリックしてください。

バグレポート

<https://bugzilla.suse.com/> から入手できるドキュメントを使用して、問題を報告してください。このプロセスを簡略化するために、このドキュメントのHTMLバージョンの見出しの横にあるReport Documentation Bug (ドキュメントバグの報告)リンクを使用できます。リンクを使用すると、Bugzillaで適切な製品とカテゴリが事前に選択され、現在のセクションへのリンクが追加されます。バグレポートの入力を直ちに開始できます。Bugzillaアカウントが必要です。

ドキュメントの編集に貢献

このドキュメントに貢献するには、このドキュメントのHTMLバージョンの見出しの横にあるEdit Source (ソースの編集)リンクを使用してください。GitHubのソースコードに移動し、そこからプル要求を提出できます。GitHubアカウントが必要です。

このドキュメントに使用されるドキュメント環境に関する詳細については、[リポジトリのREADME \(https://github.com/SUSE/doc-sleha/blob/master/README.adoc\)](https://github.com/SUSE/doc-sleha/blob/master/README.adoc) を参照してください。

メール

ドキュメントに関するエラーの報告やフィードバックはdoc-team@suse.com宛に送信してください。ドキュメントのタイトル、製品のバージョン、およびドキュメントの発行日を明記してください。関連するセクション番号とタイトル(またはURLを含めて)、問題の簡潔な説明を記載してください。

3 マニュアルの表記規則

このマニュアルでは、次の通知と表記規則が使用されています。

- `tux > command`

`root`ユーザを含む、任意のユーザが実行可能なコマンド。

- `root # command`

`root`特権で実行する必要のあるコマンド。多くの場合、これらのコマンドの頭に`sudo`コマンドを置いて実行することもできます。

- `crm(live)#`

対話型crmシェルで実行されるコマンド。詳細については、[第8章「クラスタリソースの設定と管理\(コマンドライン\)」](#)を参照してください。

- `/etc/passwd`:ディレクトリ名とファイル名

- PLACEHOLDER:PLACEHOLDERは、実際の値で置き換えられます
- PATH:環境変数PATH
- ls、--help:コマンド、オプション、およびパラメータ
- user:ユーザまたはグループ
- packagename:パッケージの名前
- Alt , Alt - F1 :使用するキーまたはキーの組み合わせ、キーはキーボード上と同様、大文字で表示される
- ファイル、ファイル > 名前を付けて保存: メニュー項目、ボタン
- amd64, em64t, ipf > この説明は、amd64、em64t、およびipfの各アーキテクチャにのみ当てはまります。矢印は、テキストブロックの先頭と終わりを示します。 < >
- 「Dancing Penguins」 (「Penguins」の章、↑他のマニュアル):他のマニュアルの章への参照です。
- 通知



警告

続行する前に知っておくべき、無視できない情報。セキュリティ上の問題、データ損失の可能性、ハードウェアの損傷、または物理的な危険について警告します。



重要

続行する前に知っておくべき重要な情報です。



注記

追加情報。たとえば、ソフトウェアバージョンの違いに関する情報です。



ヒント

ガイドラインや実地的なアドバイスなどの役に立つ情報です。


クラスタノードと名前、リソース、およびに制約に関する命名規則の概要については、[付録B 命名規則](#)を参照してください。

4 サポート

SUSE Linux Enterprise High Availability Extensionのサポートステートメントと、技術プレビューに関する概要を以下に示します。製品ライフサイクルの詳細については、SUSE Linux Enterprise Server 15 SP3の『アップグレードガイド (<https://documentation.suse.com/sles-15/html/SLES-all/cha-upgrade-background.html>) 』を参照してください。

サポートを受ける資格がある場合は、SUSE Linux Enterprise Server 15 SP3の『管理ガイド (<https://documentation.suse.com/sles-15/html/SLES-all/cha-adm-support.html>) 』でサポートチケットの情報を収集する方法の詳細を確認してください。

4.1 SUSE Linux Enterprise High Availability Extensionのサポートステートメント

サポートを受けるには、サービス内容に応じたサブスクリプションをSUSEから購入する必要があります。利用可能なサポートサービスを具体的に確認するには、<https://www.suse.com/support/>  にアクセスして製品を選択してください。

サポートレベルは次のように定義されます。

L1

問題の判別。互換性情報、使用サポート、継続的な保守、情報収集、および利用可能なドキュメントを使用した基本的なトラブルシューティングを提供するように設計されたテクニカルサポートを意味します。

L2

問題の切り分け。データの分析、お客様の問題の再現、問題領域の特定、レベル1で解決できない問題の解決、またはレベル3の準備を行うように設計されたテクニカルサポートを意味します。

L3

問題解決。技術者の関与の下で問題を解決するように設計されたテクニカルサポートを意味し、レベル2サポートで特定された製品の不具合を解決します。

契約されているお客様およびパートナーの場合、SUSE Linux Enterprise High Availability Extensionでは、次のものを除くすべてのパッケージに対してL3サポートを提供します。

- 技術プレビュー。
- サウンド、グラフィック、フォント、およびアートワーク。
- 追加の顧客契約が必要なパッケージ。

- パッケージの中には、「Workstation Extension」モジュールの一部として出荷されるものがあります。これについてはL2サポートまでしか提供されません。
- 名前が `-devel` で終わるパッケージ(ヘッダファイルなどの開発用リソースが含まれるパッケージ)のサポートを受けるには、そのメインパッケージが必要です。

SUSEは、オリジナルのパッケージだけをサポートします。すなわち、変更や再コンパイルを行ったパッケージはサポート対象外です。

4.2 技術プレビュー

技術プレビューとは、今後のイノベーションを垣間見ていただくための、SUSEによって提供されるパッケージ、スタック、または機能を意味します。プレビューは、使用中の環境内で新しいテクノロジーをテストする際の利便性のために用意されています。私たちはフィードバックを歓迎しています。技術プレビューをテストする場合は、SUSEの担当者に連絡して、経験や使用例をお知らせください。お客様からの情報を、今後の開発に役立てさせていただきます。

ただし、技術プレビューには、次の制限事項があります。

- 技術プレビューはまだ開発中です。したがって、機能が不完全であったり、不安定であったり、何らかの理由で運用環境での使用には適していなかったりする場合があります。
- 技術プレビューにはサポートが提供されません。
- 技術プレビューは、特定のハードウェアアーキテクチャでしか利用できないことがあります。
- 技術プレビューの詳細および機能は、変更される場合があります。そのため、今後リリースされる技術プレビューへのアップグレードができない場合や、再インストールが必要となる場合があります。
- 技術プレビューは、任意の時点で終了する可能性があります。たとえば、SUSEでプレビューがお客様または市場のニーズを満たしていない、またはエンタープライズ基準に準拠していないことが判明した場合などです。SUSEでは、このようなテクノロジーのサポートされるバージョンを将来的に提供できない場合があります。

ご使用の製品に付属している技術プレビューの概要については、<https://www.suse.com/releasesnotes/>にあるリリースノートを参照してください。

I インストール、セットアップ、およびアップグレード

- 1 製品の概要 2
- 2 システム要件と推奨事項 15
- 3 High Availability Extensionのインストール 20
- 4 YaSTクラスタモジュールの使用 23
- 5 クラスタアップグレードとソフトウェアパッケージの更新 41

1 製品の概要

SUSE® Linux Enterprise High Availability Extensionは、オープンソースクラスタリング技術の統合スイートです。これにより、高可用性の物理および仮想Linuxクラスタを実装し、シングルポイント障害を排除することができます。データ、アプリケーション、サービスなどの重要なネットワークリソースの高度な可用性と管理のしやすさを実現します。その結果、ミッションクリティカルなLinuxワークロードに対してビジネスの継続性維持、データ整合性の保護、予期せぬダウンタイムの削減を行います。

基本的な監視、メッセージング、およびクラスタリソース管理の機能を標準装備し、個々の管理対象クラスタリソースのフェールオーバー、フェールバック、およびマイグレーション(負荷分散)をサポートします。

この章では、High Availability Extensionの主な製品機能と利点を紹介します。ここには、いくつかのクラスタ例が記載されており、クラスタを設定するコンポーネントについて学ぶことができます。最後のセクションでは、アーキテクチャの概要を示し、クラスタ内の個々のアーキテクチャ層とプロセスについて説明します。

High Availabilityクラスタのコンテキストでよく使用される用語については、[用語集](#)を参照してください。

1.1 拡張としての提供

High Availability Extensionは、SUSE Linux Enterprise Server 15 SP3の拡張として入手できます。Geo Clustering for SUSE Linux Enterprise High Availability Extensionを使用することで、無制限の距離にまたがってHigh Availabilityクラスタを使用できるようになります。

1.2 主な特長

SUSE® Linux Enterprise High Availability Extensionでは、ネットワークリソースの可用性を確保し、管理することができます。以降のセクションでは、いくつかの主要機能に焦点を合わせて説明します。

1.2.1 広範なクラスタリングシナリオ

High Availability Extensionは次のシナリオをサポートしています。

- アクティブ/アクティブ設定
- アクティブ/パッシブ設定: N+1、N+M、Nから1、NからM
- ハイブリッド物理仮想クラスタ。仮想サーバを物理サーバとともにクラスタ化できます。これによって、サービスの可用性とリソースの使用状況が向上します。
- ローカルクラスタ
- メトロクラスタ(「ストレッチされた」ローカルクラスタ)
- Geoクラスタ(地理的に離れたクラスタ)

クラスタには、最大32のLinuxサーバを含めることができます。pacemaker_remoteを使用すると、この制限を超えて追加のLinuxサーバを含めるようにクラスタを拡張できます。クラスタ内のどのサーバも、クラスタ内の障害が発生したサーバのリソース(アプリケーション、サービス、IPアドレス、およびファイルシステム)を再起動することができます。

1.2.2 柔軟性

High Availability Extensionには、Corosyncメッセージングおよびメンバーシップ層のほか、Pacemakerクラスタリソースマネージャが標準装備されています。管理者は、Pacemakerを使用して、リソースのヘルスと状態を継続的に監視し、依存関係を管理することができます。高度に設定可能なルールとポリシーに基づいて、サービスを自動的に停止および開始することができます。High Availability Extensionでは、ユーザの組織に適した特定のアプリケーションおよびハードウェアインフラストラクチャに合わせて、クラスタのカスタマイズが可能です。時間依存設定を使用して、サービスを特定の時刻に修復済みのノードに自動的にフェールバック(マイグレート)させることができます。

1.2.3 ストレージとデータレプリケーション

High Availability Extensionでは必要に応じてサーバストレージを自動的に割り当て、再割り当てすることができます。ファイバチャネルまたはiSCSIストレージエリアネットワーク(SAN)をサポートしています。共有ディスクもサポートされていますが、必要要件ではありません。SUSE Linux Enterprise High Availability Extensionには、クラスタ対応のファイルシステム(OCFS2)とCluster Logical Volume Manager (Cluster LVM2)も含まれています。データをレプリケーションする場合は、DRBD*を使用して、High Availabilityサービスのデータ

をクラスタのアクティブノードからスタンバイノードへミラーリングします。さらに、SUSE Linux Enterprise High Availability Extensionでは、Sambaクラスタリング技術であるCTDB (Cluster Trivial Database)もサポートしています。

1.2.4 仮想化環境のサポート

SUSE Linux Enterprise High Availability Extensionは、物理Linuxサーバと仮想Linuxサーバの混合クラスタリングをサポートしています。SUSE Linux Enterprise Server 15 SP3には、オープンソースの仮想化ハイパーバイザであるXenと、KVM (カーネルベースの仮想マシン)が付属しています。KVMは、ハードウェア仮想化の拡張機能に基づいた、Linux用の仮想化ソフトウェアです。High Availability Extension内のクラスタリソースマネージャは、仮想サーバで実行中のサービスと物理サーバで実行中のサービスを認識、監視、および管理できます。ゲストシステムは、クラスタにサービスとして管理されます。

1.2.5 ローカル、メトロ、およびGeoクラスタのサポート

SUSE Linux Enterprise High Availability Extensionは、様々な地理的なシナリオをサポートするように拡張されています。SUSE Linux Enterprise High Availability Extension用のGeo Clusteringでは、地理的に分散したクラスタ(Geoクラスタ)のサポートが利用可能です。

ローカルクラスタ

1つのロケーション内の単一のクラスタ(たとえば、すべてのノードが1つのデータセンターにある)。クラスタはノード間の通信にマルチキャストまたはユニキャストを使用し、フェールオーバーを内部で管理します。ネットワークの遅延時間は無視できます。ストレージは通常、すべてのノードに同時にアクセスされます。

メトロクラスタ

すべてのサイトがファイバチャネルで接続された、複数の建物またはデータセンターにわたってストレッチできる単一のクラスタ。クラスタはノード間の通信にマルチキャストまたはユニキャストを使用し、フェールオーバーを内部で管理します。ネットワークの遅延時間は通常は短くなります(約20マイルの距離で5ms未満)。ネットワークのレイテンシは通常は短くなります(約20マイルの距離で<5ms)。ストレージは頻繁にレプリケートされます(ミラーリングまたは同期レプリケーション)

Geoクラスタ(マルチサイトクラスタ)

それぞれにローカルクラスタを持つ、複数の地理的に離れたサイト。サイトはIPによって交信します。サイト全体のフェールオーバーはより高いレベルのエンティティによって調整されます。Geoクラスタは限られたネットワーク帯域幅および高レイテンシに対応する必要があります。ストレージは同期的にレプリケートされます。

個々のクラスターノード間の地理的距離が大きいほど、クラスターが提供するサービスの高可用性を妨げる可能性のある要因が多くなります。ネットワークの遅延時間、限られた帯域幅およびストレージへのアクセスが長距離クラスターの課題として残ります。

1.2.6 リソースエージェント

SUSE Linux Enterprise High Availability Extensionには、Apache、IPv4、IPv6、その他多数のリソースを管理するための膨大な数のリソースエージェントが含まれています。またIBM WebSphere Application Serverなどの一般的なサードパーティアプリケーション用のリソースエージェントも含まれています。ご利用の製品に含まれているOpen Cluster Framework (OCF)リソースエージェントの概要は、[8.1.3項「OCFリソースエージェントに関する情報の表示」](#)で説明される`crm ra`コマンドを使用してください。

1.2.7 ユーザフレンドリな管理ツール

High Availability Extensionには、一連の強力なツールが付属しています。クラスターの基本的なインストールとセットアップ、および効果的な設定と管理のためにこれらのツールを使用してください。

YaST

一般的なシステムインストールおよび管理用グラフィカルユーザインタフェース。『インストールおよびセットアップクイックスタート』で説明されているように、YaSTを使用して、High Availability ExtensionをSUSE Linux Enterprise Server上にインストールします。YaSTでは、クラスターまたは個々のコンポーネントの設定に役立つように、High Availabilityカテゴリ内の次のモジュールも提供しています。

- クラスター: 基本的なクラスターセットアップ。詳細については、[第4章「YaSTクラスターモジュールの使用」](#)を参照してください。
- DRBD: Distributed Replicated Block Deviceの設定。
- IP負荷分散: Linux仮想サーバまたはHAProxyによる負荷分散の設定。詳細については、[第15章「負荷分散」](#)を参照してください。

Hawk2

High AvailabilityクラスターをLinuxまたは非Linuxマシンから同様に監視および管理することができる、ユーザフレンドリなWebベースのインタフェース。Hawk2には、(グラフィカルな)Webブラウザを使用して、クラスターの内部または外部の任意のマシンからア

クセスできます。したがって、使用しているシステムが最小限のグラフィカルユーザインタフェースしか提供していない場合でも、理想的なソリューションとなります。詳細については、[第7章「Hawk2を使用したクラスタリソースの設定と管理」](#)を参照してください。

crmシェル

リソースを設定し、すべての監視または管理作業を実行する、統合されたパワフルなコマンドラインインタフェースです。詳細については、[第8章「クラスタリソースの設定と管理\(コマンドライン\)」](#)を参照してください。

1.3 利点

High Availability Extensionを使用すると、最大32台のLinuxサーバを高可用性クラスタ(HAクラスタ)に設定できます。リソースを動的に切り替えたり、クラスタ内の任意のノードに移動することができます。ノード障害発生時のリソースの自動マイグレーションの設定ができます。また、ハードウェアのトラブルシューティングやワークロードのバランスをとるために、リソースを手動で移動することもできます。

High Availability Extensionは、コモディティコンポーネントによる高可用性を提供しています。アプリケーションと操作をクラスタに統合することによって、運用コストを削減できます。High Availability Extensionを使用すると、クラスタ全体を一元的に管理することもできます。変化するワークロード要件に合わせてリソースを調整する(つまり、手動でクラスタを「負荷分散」する)ことができます。3ノード以上でクラスタを設定すると、複数のノードが「ホットスペア」を共用できて無駄がありません。

その他にも重要な利点として、予測できないサービス停止を削減したり、ソフトウェアおよびハードウェアの保守やアップグレードのための計画的なサービス停止を削減できる点が挙げられます。

次に、クラスタによるメリットについて説明します。

- 可用性の向上
- パフォーマンスの改善
- 運用コストの低減
- 拡張性
- 障害回復
- データの保護

- サーバの集約
- ストレージの集約

共有ディスクサブシステムにRAIDを導入することによって、共有ディスクの耐障害性を強化できます。

次のシナリオは、High Availability Extensionの利点を紹介するものです。

クラスタシナリオ例

ノード3台でクラスタが設定され、それぞれのノードにWebサーバをインストールしたと仮定します。クラスタ内の各ノードが、2つのWebサイトをホストしています。各Webサイトのすべてのデータ、グラフィックス、Webページコンテンツは、クラスタ内の各ノードに接続された、共有ディスクサブシステムに保存されています。次の図は、このクラスタのセットアップを示しています。

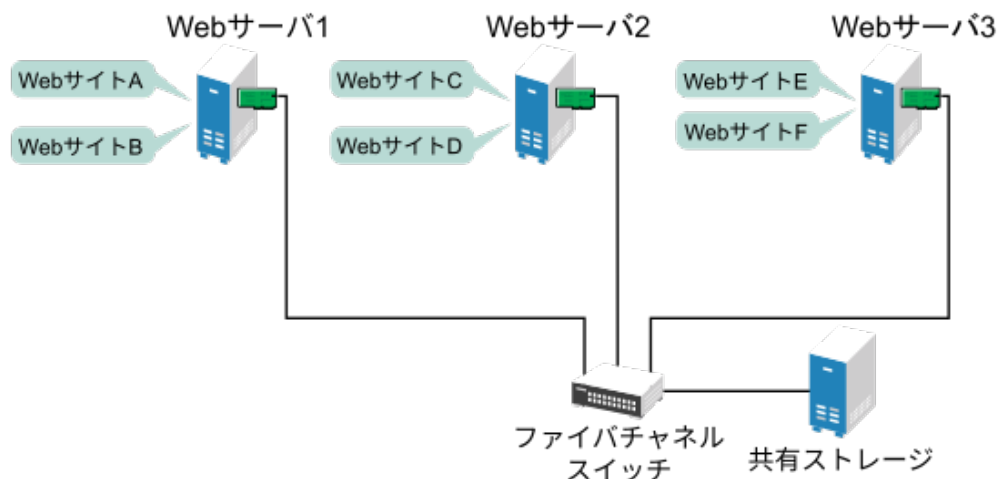


図 1.1: 3サーバクラスタ

通常のクラスタ操作では、クラスタ内の各ノードが他のノードと常に交信し、すべての登録済みリソースを定期的にポーリングして、障害を検出します。

Webサーバ1でハードウェアまたはソフトウェアの障害が発生したため、このサーバを利用してインターネットアクセス、電子メール、および情報収集を行っているユーザの接続が切断されたとします。次の図は、Webサーバ1で障害が発生した場合のリソースの移動を表したものです。

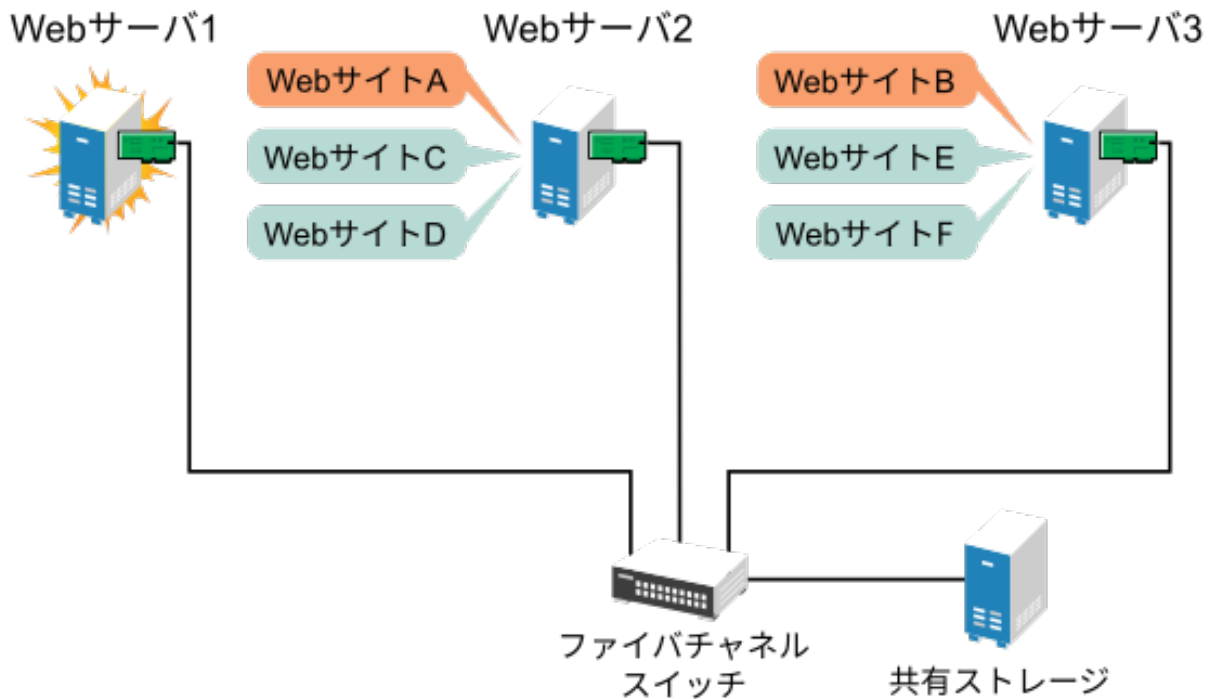


図 1.2: 1台のサーバに障害が発生した後の3サーバクラスタ

WebサイトAがWebサーバ2に、WebサイトBがWebサーバ3に移動します。IPアドレスと証明書もWebサーバ2とWebサーバ3に移動します。

クラスタを設定するときに、それぞれのWebサーバがホストしているWebサイトについて、障害発生時の移動先を指定します。先に説明した例では、WebサイトAの移動先としてWebサーバ2が、WebサイトBの移動先としてWebサーバ3が指定されています。このようにして、Webサーバ1によって処理されていたワークロードが、残りのクラスタメンバーに均等に分散され、可用性を維持できます。

Webサーバ1で障害が発生すると、High Availability Extensionソフトウェアは次の処理を実行します。

- 障害を検出し、Webサーバ1が本当に機能しなくなっていることをSTONITHを使用して検証。STONITHは「Shoot The Other Node In The Head」の略です。これは、動作異常のノードを停止することでクラスタに問題を発生させないようにする手段です。
- Webサーバ1にマウントされていた共有データディレクトリを、Webサーバ2およびWebサーバ3に再マウント。
- Webサーバ1で動作していたアプリケーションを、Webサーバ2およびWebサーバ3で再起動。
- IPアドレスをWebサーバ2およびWebサーバ3に移動。

この例では、フェールオーバープロセスが迅速に実行され、ユーザはWebサイトの情報へのアクセスを数秒程度で回復できます。通常、再度ログインする必要はありません。

ここで、Webサーバ1で発生した問題が解決し、通常に操作できる状態に戻ったと仮定します。WebサイトAおよびWebサイトBは、Webサーバ1に自動的にフェールバック(復帰)することも、そのままの状態を維持することもできます。これは、リソースの設定方法によって決まります。サービスをWebサーバ1に戻すと、ある程度のダウンタイムが生じます。このため、High Availability Extensionでは、サービスの中断がほとんどまたはまったく発生しなくなるまで、マイグレーションを延期することもできます。いずれの場合でも利点と欠点があります。

High Availability Extensionは、リソースマイグレーション機能も提供します。アプリケーション、Webサイトなどをシステム管理の必要性に応じて、クラスタ内の他のサーバに移動することができます。

たとえば、WebサイトAまたはWebサイトBをWebサーバ1からクラスタ内の他のサーバに手動で移動することができます。これは、Webサーバ1のアップグレードや定期メンテナンスを実施する場合、また、Webサイトのパフォーマンスやアクセスを向上させる場合に有効な機能です。

1.4 クラスタ設定: ストレージ

High Availability Extensionでのクラスタ構成には、共有ディスクサブシステムが含まれる場合と含まれない場合があります。共有ディスクサブシステムの接続には、高速ファイバチャネルカード、ケーブル、およびスイッチを使用でき、また設定にはiSCSIを使用することができます。ノードの障害時には、クラスタ内の別の指定されたノードが、障害の発生したノードにマウントされていた共有ディスクディレクトリを自動的にマウントします。この機能によって、ネットワークユーザは、共有ディスクサブシステム上のディレクトリに対するアクセスを中断することなく実行できます。

！ 重要: LVM2を伴う共有ディスクサブシステム

共有ディスクサブシステムをLVM2と使用する場合、クラスタ内の、アクセスが必要なすべてのサーバにそのサブシステムを接続する必要があります。

一般的なりソースの例としては、データ、アプリケーション、およびサービスなどがあります。次の図は、一般的なファイバチャネルクラスタの設定を表したものです。緑色の線は、Ethernet電源スイッチへの接続を示しています。このようなデバイスは、ネットワークを介して制御することが可能であり、ping要求が失敗したときにノードを再起動することができます。

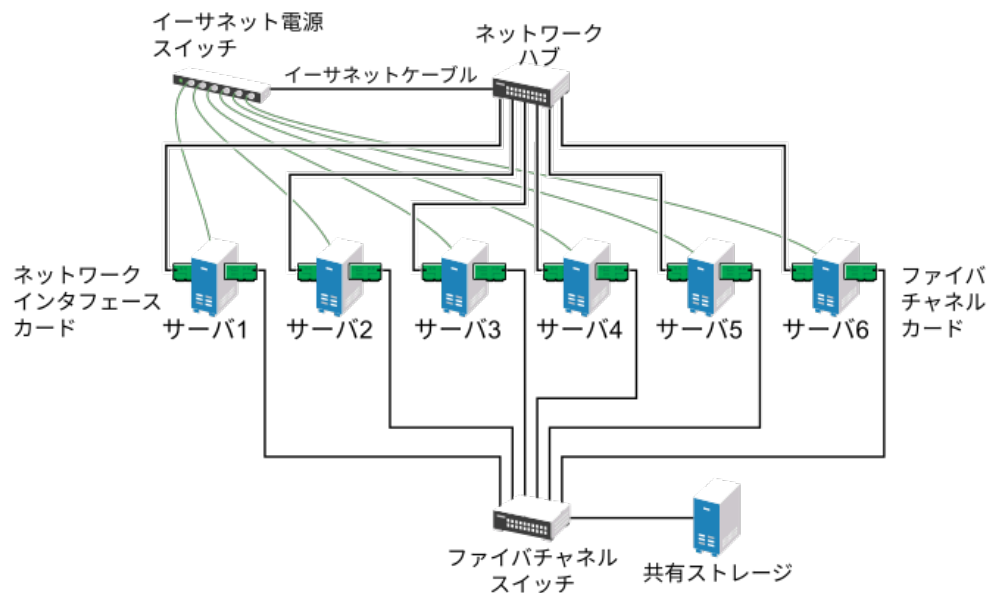


図 1.3: 一般的なファイバチャネルクラスタの設定

ファイバチャネルは最も高いパフォーマンスを提供しますが、iSCSIを利用するようにクラスタを設定することもできます。iSCSIは低コストなストレージエリアネットワーク(SAN)を作成するための方法として、ファイバチャネルの代わりに使用できます。次の図は、一般的なiSCSIクラスタの設定を表したものです。

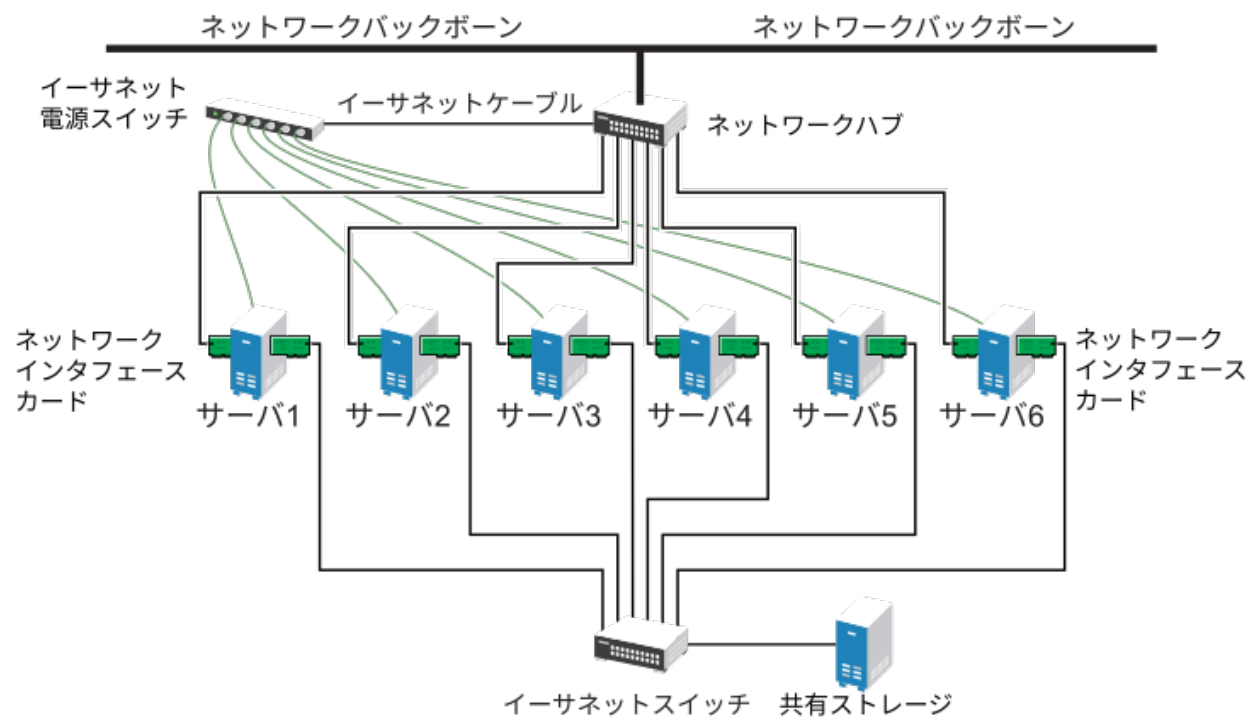


図 1.4: 一般的なiSCSIクラスタの設定

ほとんどのクラスタには共有ディスクサブシステムが含まれていますが、共有ディスクサブシステムなしのクラスタを作成することもできます。次の図は、共有ディスクサブシステムなしのクラスタを表したものです。

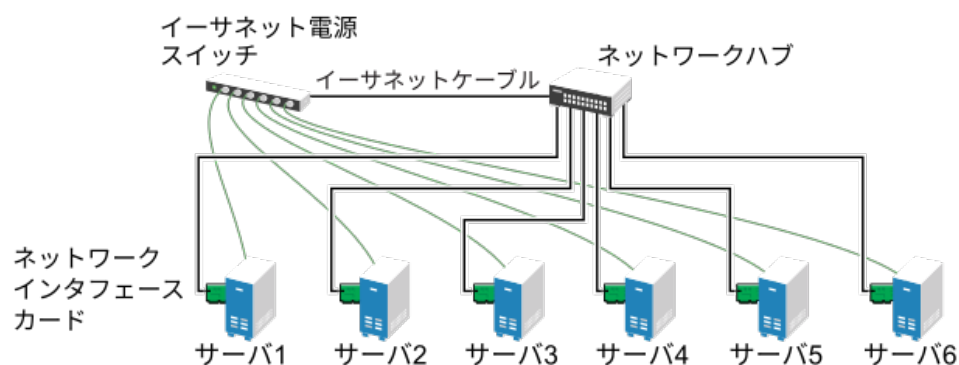


図 1.5: 共有ストレージなしの一般的なクラスタ設定

1.5 アーキテクチャ

このセクションでは、High Availability Extensionアーキテクチャの概要を説明します。アーキテクチャコンポーネントと、その相互運用方法について説明します。

1.5.1 アーキテクチャ層

High Availability Extensionのアーキテクチャは層化されています。図1.6「アーキテクチャ」に異なる層と関連するコンポーネントを示します。

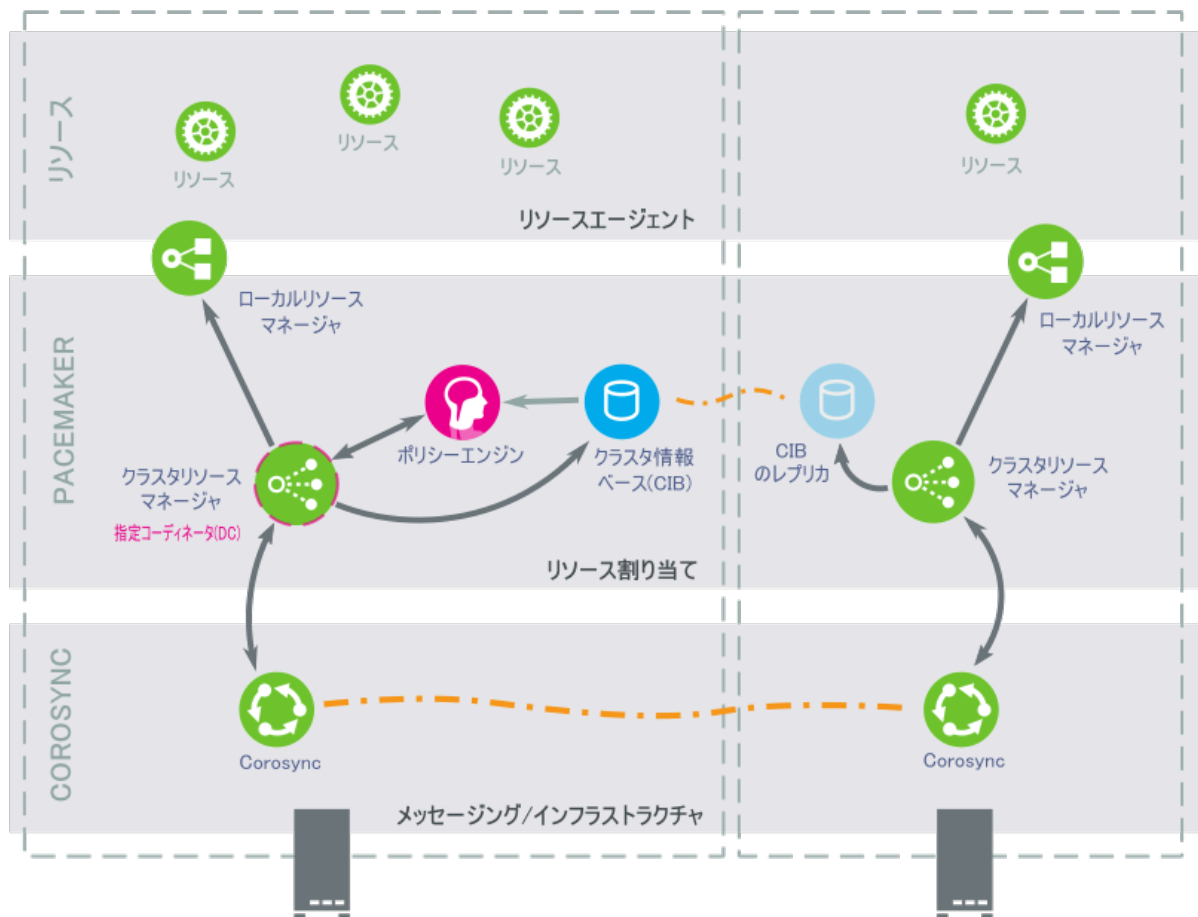


図 1.6: アーキテクチャ

1.5.1.1 メンバーシップとメッセージング層(Corosync)

このコンポーネントは、クラスタのメッセージング、メンバーシップ、クォーラムに関する信頼性の高い情報を提供します。具体的には、グループ通信システムであるCorosyncクラスタエンジンがその処理を担っています。

1.5.1.2 クラスタリソースマネージャ(Pacemaker)

クラスタリソースマネージャであるPacemakerは、クラスタ内で発生したイベントへの対応を司る「頭脳」です。これは、あらゆるアクションを統括するpacemaker-controldクラスタコントローラとして実装されます。イベントとは、クラスタにおけるノードの加入と離脱、リソースでの障害発生、保守などの計画的なアクティビティのことを指します。

ローカルリソースマネージャ

ローカルリソースマネージャは、各ノードのPacemaker層とリソース層の間に存在し、`pacemaker-execd`デーモンとして実装されます。このデーモンにより、Pacemakerでのリソースの起動、停止、監視が可能になります。

CIB (クラスタ情報データベース)

Pacemakerは、ノードごとにCIBを保持しています。CIBとは、クラスタ設定のXML表現のことで、クラスタの各オプション、ノード、リソース、制約、個々の要素間の関係性などが記述されています。CIBには、現在のクラスタのステータスも反映されます。各クラスタノードにはCIBレプリカが配置され、クラスタ全体との同期がとられます。クラスタの設定とステータスの読み書きは、`pacemaker-based`デーモンが行います。

DC (指定コーディネータ)

DCは、クラスタ内にあるすべてのノードの中から選択されます。この操作は、DCがまだ指定されていない場合や、現在のDCがなんらかの理由でクラスタを離脱した場合に行われます。DCは、ノードのフェンシングやリソースの移動など、クラスタ全体におよぶ変更が必要かどうかを判断できる、クラスタ内で唯一のエンティティです。その他すべてのノードは、現在のDCから設定とリソース割り当て情報を取得します。

ポリシーエンジン

ポリシーエンジンはすべてのノードで実行できますが、DC上にあるものだけがアクティブになります。このエンジンは`pacemaker-schedulerd`デーモンとして実装されます。クラスタ遷移が必要になると、`pacemaker-schedulerd`はクラスタの現在の状態と設定を基に、次のクラスタの状態を計算します。また、次の状態を達成するためにどんなアクションを行う必要があるかも決定します。

1.5.1.3 リソースおよびリソースエージェント

高可用性を備えたクラスタでは、高い可用性を維持し続ける必要があるサービスのことを「リソース」と呼びます。RA (リソースエージェント)とは、これらのクラスタリソースの起動、停止、監視に使用されるスクリプトのことです。

1.5.2 プロセスフロー

pacemakerdデーモンは、その他のあらゆる関連デーモンの起動と監視を行います。pacemaker-controldデーモンはすべてのアクションを統括し、自身のインスタンスを各クラスタノードに配置します。Pacemakerは、マスタとして動作するインスタンスを1つ選択することにより、クラスタのすべての意思決定を一元化します。選択したpacemaker-controldデーモンで障害が発生する場合は、新たなインスタンスがマスタになります。

クラスタ内で実行するアクションの多くは、クラスタ全体におよぶ変更を伴います。これらのアクションにはクラスタリソースの追加や削除、リソース制約の変更などがあります。このようなアクションを実行する場合は、クラスタ内でどのような変化が発生するのかを理解することが重要です。

たとえば、クラスタIPアドレスリソースを追加するとします。そのためには、crmシェルかWebインタフェースを使用してCIBを変更できます。DCでアクションを実行する必要はありません。クラスタの任意のノードでいずれかのツールを使用して、DCにリレーされます。そして、DCがすべてのクラスタノードにCIBの変更を複製します。

このときは、CIBの情報に基づいて、pacemaker-schedulerdがクラスタの理想的な状態とその達成方法を計算し、命令のリストをDCにフィードします。DCはメッセージング/インフラストラクチャ層を介してコマンドを送信し、他のノードのpacemaker-controldピアがこれらのコマンドを受信します。それぞれのピアは、ローカルでリソースエージェントeg pacemaker-execdを使用してリソースに変更を加えます。pacemaker-execdはクラスタに対応しておらず、リソースエージェントと直接通信します。

すべてのピアノードは操作結果をDCに返送します。DCが、すべての必要な操作がクラスタ内で成功したことを確認すると、クラスタはアイドル状態に戻り、次のイベントを待機します。予定通り実行されなかった操作があれば、CIBに記録された新しい情報を基に、pacemaker-schedulerdを再度呼び出します。

場合によっては、共有データの保護や完全なリソース復旧のためにノードの電源を切らなければならないことがあります。Pacemakerクラスタにおけるノードレベルフェンシングの実装は、STONITHです。このPacemakerにはフェンシングサブシステムpacemaker-fencedが内蔵されています。STONITHデバイスは、(特定のフェンシングエージェントを使用する)クラスタリソースとして設定する必要があります。これにより、初めてフェンシングデバイスの監視が可能になるからです。クライアントは障害を検出すると、pacemaker-fencedへ要求を送信します。このデーモンはフェンシングエージェントを実行することにより、ノードを停止します。

2 システム要件と推奨事項

次のセクションでは、SUSE® Linux Enterprise High Availability Extensionのシステム要件と前提条件について説明します。また、クラスタセットアップの推奨事項についても説明します。

2.1 ハードウェア要件

次のリストは、SUSE® Linux Enterprise High Availability Extensionに基づくクラスタのハードウェア要件を指定しています。これらの要件は、最低のハードウェア設定を表しています。クラスタの使用方法によっては、ハードウェアを追加しなければならないこともあります。

サーバ

2.2項「ソフトウェアの必要条件」に指定されたソフトウェアを搭載した1～32台のLinuxサーバ。

サーバはベアメタルでも仮想マシンでも構いません。各サーバが同一のハードウェア設定(メモリ、ディスクスペースなど)になっている必要はありませんが、アーキテクチャは同じである必要があります。クロスプラットフォームのクラスタはサポートされていません。

`pacemaker_remote`を使用すると、32ノード制限を超えて追加のLinuxサーバを含めるようにクラスタを拡張できます。

通信チャンネル

クラスタノードあたり、少なくとも2つのTCP/IP通信メディア。ネットワーク機器は、クラスタ通信に使用する通信手段(マルチキャストまたはユニキャスト)をサポートする必要があります。通信メディアは100Mbit/s以上のデータレートをサポートする必要があります。サポートされるクラスタセットアップでは、2つ以上の冗長通信パスが必要です。これは次のように実行できます。

- ネットワークデバイスボンディング(推奨)。
- Corosync内の2つ目の通信チャンネル。

詳細については、第14章「ネットワークデバイスボンディング」と手順4.3「冗長通信チャンネルの定義」をそれぞれ参照してください。

ノードフェンシング/STONITH

「スプリットブレイン」シナリオを回避するため、クラスタにはノードフェンシングメカニズムが必要です。スプリットブレインシナリオでは、クラスタノードは、お互いを認識していない2つ以上のグループに分割されます(ハードウェアまたはソフトウェアの障害か、ネットワーク接続の切断による)。フェンシングメカニズムにより、問題のあるノードを分離します(通常はノードをリセットするか、ノードの電源をオフにすることによって分離します)。これをSTONITH (「Shoot the other node in the head」)と呼びます。ノードフェンシングメカニズムは、物理デバイス(電源スイッチ)でも、SBD (ディスクによるSTONITH)のようなメカニズムとウォッチドッグを組み合わせたものでも構いません。SBDを使用するには共有ストレージが必要です。

SBDが使用される場合を除き、High Availabilityクラスタの各ノードには少なくとも1つのSTONITHデバイスが必要です。ノードごとに複数のSTONITHデバイスを使用することを強くお勧めします。

❗ 重要: STONITHがない場合はサポートなし

- クラスタにはノードフェンシングメカニズムが必要です。
- グローバルクラスタオプション`stonith-enabled`および`startup-fencing`を`true`に設定する必要があります。これらを変更するとサポートされなくなります。

2.2 ソフトウェアの必要条件

クラスタの一部になるすべてのノードには、少なくとも次のモジュールと拡張機能が含まれている必要があります。

- Basesystem Module 15 SP3
- Server Applications Module 15 SP3
- SUSE Linux Enterprise High Availability Extension 15 SP3

インストール中に選択したシステム役割に応じて、デフォルトで次のソフトウェアパターンがインストールされます。

表 2.1: システムの役割とインストールされるパターン

システムの役割	ソフトウェアパターン(YaST/Zypper)
HAノード	<ul style="list-style-type: none"> • High Availability (sles_ha) • 強化された基本システム (enhanced_base)
HA GEOノード	<ul style="list-style-type: none"> • Geo Clustering for High Availability (ha_geo) • 強化された基本システム (enhanced_base)



注記: 最小インストール

これらのシステムの役割を介してインストールすると、最小限のインストールにのみ抑えられます。必要に応じて、パッケージを手動で追加しなければならない場合があります。

最初別のシステムの役割が割り当てられていたマシンの場合、sles_haまたはha_geoパターンとその他の必要なパッケージを手動でインストールする必要があります。

2.3 ストレージ要件

一部のサービスでは、共有ストレージが必要です。外部NFS共有を使用する場合は、冗長通信パスを介してすべてのクラスタノードから確実にアクセスできる必要があります。

クラスタでデータの可用性を高めたい場合は、共有ディスクシステム(SAN: Storage Area Network)の利用をお勧めします。共有ディスクシステムを使用する場合は、次の要件を満たしていることを確認してください。

- メーカーの指示のに従い、共有ディスクシステムが適切に設定され、正しく動作していることを確認します。
- 共有ディスクシステム中のディスクを、ミラーリングまたはRAIDを使用して耐障害性が高められるように設定してください。

- 共有ディスクシステムのアクセスにiSCSIを使用している場合、iSCSIイニシエータとターゲットを正しく設定していることを確認します。
- 2台のマシンにデータを配分するミラーリングRAIDシステムを実装するためにDRBD*を使用する際、DRBDに提供されるデバイスにのみアクセスし、決してバックアップデバイスにはアクセスしないようにします。冗長性を確保するために、クラスタの残りの部分と同一のNICを利用できます。

SBDをSTONITHメカニズムとして使用する場合は、共有ストレージに対して追加の要件が適用されます。詳細については、[11.3項「要件」](#)を参照してください。

2.4 その他の要件と推奨事項

サポートされていて、役に立つHigh Availabilityセットアップについては、次の推奨事項を検討してください。

クラスタノード数

3つ以上のノードを持つクラスタに対して、奇数のクラスタノードを使用してクォーラムを持つようにすることを強くお勧めします。クォーラムの詳細については、[6.2項「クォーラムの判断」](#)を参照してください。通常のクラスタには最大32ノードを含めることができます。`pacemaker_remote`サービスを使用すると、高可用性クラスタを拡張して、この制限を超えて追加のノードを含めることができます。詳細については、『Pacemaker Remote Quick Start』を参照してください。

時刻同期


クラスタノードはクラスタ外のNTPサーバに同期する必要があります。SUSE Linux Enterprise High Availability Extension 15以降、Chronyは、NTPでデフォルトで実装されるようになりました。詳細については、SUSE Linux Enterprise Server 15 SP3の『[管理ガイド \(https://documentation.suse.com/sles-15/html/SLES-all/cha-ntp.html\)](https://documentation.suse.com/sles-15/html/SLES-all/cha-ntp.html)』を参照してください。

ノードが同期されていない場合、クラスタが正常に動作しないことがあります。また、同期が行われていないと、ログファイルとクラスタレポートの分析が非常に困難になります。ブートストラップスクリプトを使用するときにNTPがまだ設定されていない場合、警告が表示されます。

ネットワークインタフェースカード(NIC)名

すべてのノード上で同一である必要があります。

ホスト名およびIPアドレス

- 静的IPアドレスを使用します。
- `/etc/hosts`ファイルにあるすべてのクラスタノードを、完全修飾ホスト名およびショートホスト名で一覧表示します。クラスタのメンバーが名前で見つけられることが重要です。名前を使用できない場合、内部クラスタ通信は失敗します。Pacemakerがノード名を取得する方法の詳細については、http://clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/s-node-name.html  も参照してください。

SSH

すべてのクラスタノードはSSHによって互いにアクセスできる必要があります。`crm report` (トラブルシューティング用)などのツールおよびHawk2の履歴エクスプローラは、ノード間でパスワード不要のSSHアクセスを必要とします。それがない場合、現在のノードからしかデータを収集できません。



注記: 規定要件

パスワード不要のSSHアクセスが規定要件に適合しない場合は、[付録D rootアクセスなしでのクラスタレポートの実行](#)で説明されている次善策を使用して`crm report`を実行できます。

履歴エクスプローラについては、現在のところ、パスワード不要のログインに代わる方法はありません。

3 High Availability Extensionのインストール

初めてSUSE® Linux Enterprise High Availability Extensionを使用してHigh Availabilityクラスタを設定する場合、最も簡単な方法は、基本的な2ノードクラスタで開始することです。2ノードクラスタを使用して、一部のテストを実行することもできます。後で、AutoYaSTを使用して既存のクラスタノードのクローンを作成することにより、さらにノードを追加できます。クローンを作成したノードには、元のノードと同じパッケージがインストールされ、クローンノードは同じシステム設定を持つことになります。

前のバージョンのSUSE Linux Enterprise High Availability Extensionを実行する既存のクラスタをアップグレードする場合は、[第5章「クラスタアップグレードとソフトウェアパッケージの更新」](#)を参照してください。


3.1 手動インストール

High Availability Extension用のパッケージの手動インストールについては、項目「インストールおよびセットアップクイックスタート」を参照してください。このマニュアルでは、基本的な2ノードクラスタをセットアップする手順を説明しています。

3.2 AutoYaSTによる大量インストールと展開

2ノードクラスタをインストールしてセットアップした後で、AutoYaSTを使用して既存のノードのクローンを作成し、クラスタにそのクローンを追加することによりクラスタを拡張できます。

AutoYaSTでは、インストールおよび設定データを含むプロファイルを使用します。このプロファイルによって、インストールする対象と、インストールしたシステムが最終的に使用準備が整ったシステムになるように設定する方法がAutoYaSTに指示されます。そこでこのプロファイルはさまざまな方法による大量配備に使用できます(たとえば、既存のクラスタノードのクローンなど)。

さまざまなシナリオでAutoYaSTを使用する方法の詳細な手順については、SUSE Linux Enterprise Server 15 SP3の『AutoYaSTガイド (<https://documentation.suse.com/sles-15/html/SLES-all/book-autoyast.html>) 』を参照してください。

！ 重要: 同一のハードウェアを使用している環境

手順3.1「AutoYaSTによるクラスタノードのクローン作成」では、同じハードウェア構成を持つ一群のマシンにSUSE Linux Enterprise High Availability Extension 15 SP3を展開していることを前提としています。

同じではないハードウェア上にクラスタノードを展開する必要がある場合は、『SUSE Linux Enterprise 15 SP3導入ガイド』、「Automated Installation」の章の「Rule-Based Autoinstallation」セクションを参照してください。

手順 3.1: AUTOYASTによるクラスタノードのクローン作成

1. クローンを作成するノードが正しくインストールされ、設定されていることを確認します。詳細については、SUSE Linux Enterprise High Availability Extension用の『インストールおよびセットアップクイックスタート』または第4章「YaSTクラスタモジュールの使用」を参照してください。
2. 単純な大量インストールについては、『SUSE Linux Enterprise 15 SP3 導入ガイド』の説明に従ってください。これには、次の基本ステップがあります。
 - a. AutoYaSTプロファイルの作成AutoYaST GUIを使用して、既存のシステム設定をもとにプロファイルを作成し、変更します。AutoYaSTでは、高可用性モジュールを選択し、クローンボタンをクリックします。必要な場合は、他のモジュールの設定を調整し、その結果のコントロールファイルをXMLとして保存します。
DRBDを設定した場合、AutoYaST GUIでこのモジュールを選択してクローンを作成することもできます。
 - b. AutoYaSTプロファイルのソースと、他のノードのインストールルーチンに渡すパラメータを決定します。
 - c. SUSE Linux Enterprise ServerのソースとSUSE Linux Enterprise High Availability Extensionインストールデータを決定します。
 - d. 自動インストールのブートシナリオを決定し、設定します。
 - e. パラメータを手動で追加するか、またはinfoファイルを作成することにより、インストールルーチンにコマンド行を渡します。
 - f. 自動インストールプロセスを開始および監視します。

クローンのインストールに成功したら、次の手順を実行して、クローンノードをクラスタに加えます。

手順 3.2: クローンノードをオンラインにする

1. 4.5項 「すべてのノードへの設定の転送」の説明に従って、Csync2を使用して、設定済みのノードからクローンノードへ重要な設定ファイルを転送します。
2. ノードをオンラインにするには、4.8項 「ノードごとにクラスタをオンラインにする」の説明のとおり、クローンノード上でPacemakerサービスを開始します。

これで、`/etc/corosync/corosync.conf`ファイルがCsync2を介してクローンノードに適用されたので、クローンノードがクラスタに加わります。CIBは、クラスタノード間で自動的に同期されます。

4 YaSTクラスタモジュールの使用

YaSTクラスタモジュールでは、クラスタを手動で(最初から)設定するか、既存のクラスタのオプションを変更することができます。

ただし、クラスタの設定に自動化された方法を選ぶ場合は、項目「インストールおよびセットアップクイックスタート」を参照してください。このマニュアルでは、必要なパッケージのインストール方法と、`ha-cluster-bootstrap`スクリプトを使用して基本的な2ノードクラスタを設定する手順を説明しています。

たとえば、1つのノードをYaSTクラスタで設定してから、ブートストラップスクリプトの1つを使用して他のノードを統合させる(またはその逆も可能)など、両方のセットアップ方法を組み合わせることもできます。

4.1 用語の定義

YaSTクラスタモジュールおよびこの章で使用されているいくつかの主要な用語を以下に定義します。

バインドネットワークアドレス(`bindnetaddr`)

Corosyncエグゼクティブのバインド先のネットワークアドレス。クラスタ間の設定ファイルの共有を簡素化するため、Corosyncはネットワークインタフェースネットマスクを使用して、ネットワークのルーティングに使用されるアドレスビットのみをマスクします。たとえば、ローカルインタフェースが192.168.5.92でネットマスクが255.255.255.0の場合、`bindnetaddr`は192.168.5.0に設定します。ローカルインタフェースが192.168.5.92でネットマスクが255.255.255.192の場合は、`bindnetaddr`を192.168.5.64に設定します。


`ringX_addr`を含む`nodelist`が`/etc/corosync/corosync.conf`で明示的に設定されている場合、`bindnetaddr`は厳密には必要ありません。



注記: すべてのノードのネットワークアドレス

すべてのノード上で同じCorosync設定が使用されるため、ネットワークアドレスは、特定のネットワークインタフェースのアドレスではなく、`bindnetaddr`として使用します。

conntrackツール

カーネル内の接続トラッキングシステムとやり取りできるようにして、iptablesでの「ステートフルな」パケット検査を可能にします。High Availability Extensionによって、クラスタノード間の接続ステータスを同期化するために使用されます。詳細については、<http://conntrack-tools.netfilter.org/>  を参照してください。

Csync2

クラスタ内のすべてのノード、およびGeoクラスタ全体に設定ファイルを複製するために使用できる同期ツールです。Csync2は、同期グループ別にソートされた任意の数のホストを操作できます。各同期グループは、メンバーホストの独自のリストとその包含/除外パターン(同期グループ内でどのファイルを同期するか定義するパターン)を持っています。グループ、各グループに属するホスト名、および各グループの包含/除外ルールは、Csync2設定ファイル/etc/csync2/csync2.cfgで指定されます。

Csync2は、認証には、同期グループ内でIPアドレスと事前共有キーを使用します。管理者は、同期グループごとに1つのキーファイルを生成し、そのファイルをすべてのグループメンバーにコピーする必要があります。

Csync2の詳細については、<http://oss.linbit.com/csync2/paper.pdf>  を参照してください。

既存のクラスタ

「既存のクラスタ」という用語は、1つ以上のノードで構成されるクラスタを指すものとして使用されます。既存のクラスタは、通信チャネルを定義する基本的なCorosync設定を持ちますが、必ずしもリソース設定を持つとは限りません。

マルチキャスト

ネットワーク内で一対多数の通信に使用される技術で、クラスタ通信に使用できます。Corosyncはマルチキャストとユニキャストの両方をサポートしています。



注記: スイッチとマルチキャスト

クラスタ通信にマルチキャストを使用するには、ご使用のスイッチがマルチキャストをサポートしていることを確認します。

マルチキャストアドレス(mcastaddr)

Corosyncエグゼクティブによるマルチキャストに使用されるIPアドレス。このIPアドレスはIPv4またはIPv6のいずれかに設定できます。IPv6ネットワークを使用する場合は、ノードのIDを指定する必要があります。プライベートネットワークでは、どのようなマルチキャストアドレスでも使用できます。

マルチキャストポート(mcastport)

クラスタ通信に使用されるポート。Corosyncでは、マルチキャストの受信用に指定するmcastportと、マルチキャストの送信用のmcastport -1の、2つのポートを使用します。

冗長リングプロトコル(RRP)

ネットワーク障害の一部または全体に対する災害耐性のために、複数の冗長ローカルエリアネットワークが使用できるようになります。この方法では、ひとつのネットワークが作動中である限り、クラスタ通信を維持できます。Corosyncはトータム冗長リングプロトコルをサポートします。信頼できるソートされた方式でメッセージを配信するために、論理トークンパスリングがすべての参加ノードに課せられます。ノードがメッセージをブロードキャストできるのは、トークンを保持している場合のみです。Corosyncに定義済みの冗長通信チャンネルを持つ場合、RRPを使用してこれらのインタフェースの使用方法をクラスタに伝えます。RRPでは次の3つのモードを使用できます(rrp_mode)。

- activeに設定した場合、Corosyncは両方のインタフェースをアクティブに使用します。ただし、このモードは非推奨の機能です。
- passiveに設定した場合、Corosyncは代わりに使用可能なネットワークを介してメッセージを送信します。
- noneに設定した場合、RRPは無効になります。

ユニキャスト

ひとつのあて先ネットワークにメッセージを送信する技術Corosyncはマルチキャストとユニキャストの両方をサポートしています。Corosyncでは、ユニキャストはUDP-unicast (UDPU)として実装されます。

4.2 YaSTクラスタモジュール

YaSTを起動して、高可用性 > クラスタを選択します。または、コマンドラインでモジュールを開始します。

```
sudo yast2 cluster
```

次のリストは、YaSTクラスタモジュールで使用可能な画面の概要を示しています。この画面には、クラスタセットアップの成功に必要なパラメータが含まれているかどうか、またはそのパラメータがオプションであるかどうか説明されています。

通信チャンネル(必須)

クラスタノード間の通信に1つまたは2つの通信チャンネルを定義できます。転送プロトコルとして、マルチキャスト(UDP)またはユニキャスト(UDPU)のいずれかを使用します。詳細については、[4.3項「通信チャンネルの定義」](#)を参照してください。

！ 重要: 冗長通信パス

サポートされるクラスタセットアップでは、2つ以上の冗長通信パスが必要です。推奨される方法は、[第14章「ネットワークデバイスボンディング」](#)で説明されるように、ネットワークデバイスボンディングを使用することです。

使用できない場合は、Corosync内に2つ目の通信チャンネルを定義する必要があります。

セキュリティ(オプションだが推奨)

クラスタの認証設定を定義できます。共有シークレットが必要なHMAC/SHA1認証を使用して、メッセージを保護し、認証することができます。詳細については、[4.4項「認証設定の定義」](#)を参照してください。

Csyc2の設定(オプションだが推奨)

Csyc2では、設定変更を追跡して、クラスタノード間でファイルの同期を取ることができます。詳細については、[4.5項「すべてのノードへの設定の転送」](#)を参照してください。

conntrackdの設定(オプション)

ユーザスペースconntrackdを設定できます。iptablesでの「ステートフルな」パケット検査のためにconntrackツールを使用します。詳細については、[4.6項「クラスタノード間の接続ステータスの同期」](#)を参照してください。

サービス(必須)

クラスタノードをオンラインにするためにサービスを設定できます。ブート時にPacemakerサービスを開始するかどうか、およびノード間の通信に必要なポートをファイアウォールで開くかどうかを定義します。詳細については、[4.7項「サービスの設定」](#)を参照してください。

初めてクラスタモジュールを起動した場合は、モジュールが、ウィザードのように、基本設定に必要なすべてのステップをガイドします。そうでない場合は、左パネルのカテゴリをクリックして、ステップごとに設定オプションにアクセスします。



注記: YaSTクラスタモジュールの設定

YaSTクラスタモジュール内のいくつかの設定は、現在のノードにのみ適用されます。他の設定はCsync2を使用してすべてのノードに自動的に転送できます。これについての詳しい情報は次のセクションを参照してください。

4.3 通信チャネルの定義

クラスタノード間で正常な通信を行うには、少なくとも1つの通信チャネルを定義します。[手順 4.1](#)または[手順 4.2](#)のそれぞれで説明されるように、転送プロトコルとしてマルチキャスト(UDP)またはユニキャスト(UDPU)のいずれかを使用します。2番目の冗長チャネル([手順 4.3](#))を定義する場合は、両方の通信チャネルで「同じ」プロトコルを使用する必要があります。



注記: パブリッククラウド: ユニキャストを使用する

SUSE Linux Enterprise High Availability Extensionをパブリッククラウドプラットフォームに展開する場合は、ユニキャストで通信してください。クラウドプラットフォームでは、一般的にマルチキャストでの通信がサポートされていません。

YaST通信チャネル画面で定義されるすべての設定は、`/etc/corosync/corosync.conf`に書き込まれます。マルチキャストおよびユニキャストセットアップのサンプルファイルは、`/usr/share/doc/packages/corosync`にあります。

IPv4アドレスを使用する場合、ノードIDはオプションです。IPv6アドレスを使用する場合、ノードIDは必須です。各ノードにIDを手動で指定する代わりに、YaSTクラスタモジュールには、クラスタノードごとに固有のIDを自動的に生成するオプションが含まれています。

手順 4.1: 最初の通信チャネルの定義(マルチキャスト)

マルチキャストを使用する場合、すべてのクラスタノードに対して同じ`bindnetaddr`、`mcastaddr`、`mcastport`が使用されます。クラスタ内のすべてのノードは同じマルチキャストアドレスを使用することで互いを認識します。別のクラスタは、別のマルチキャストアドレスを使用します。

1. YaSTクラスタモジュールを起動して、通信チャネルカテゴリに切り替えます。
2. 転送プロトコルをMulticastに設定します。
3. バインドネットワークアドレスを定義します。クラスタマルチキャストに使用するサブネットに値を設定します。

4. マルチキャストアドレスを定義します。
5. ポートを定義します。
6. クラスタノードごとに一意のIDを自動的に生成するには、ノードIDの自動生成を有効にしたままにします。
7. クラスタ名を定義します。
8. 期待する得票数の数を入力します。これは、パーティションされたクラスタでCorosyncがクォーラムを計算する場合に重要です。デフォルトでは、各ノードには1票が割り当てられています。期待する得票数の数は、クラスタ内のノード数と一致する必要があります。
9. 変更内容を確認します。
10. 必要な場合は、手順4.3「冗長通信チャネルの定義」で説明するように、Corosyncで冗長な通信チャネルを定義します。

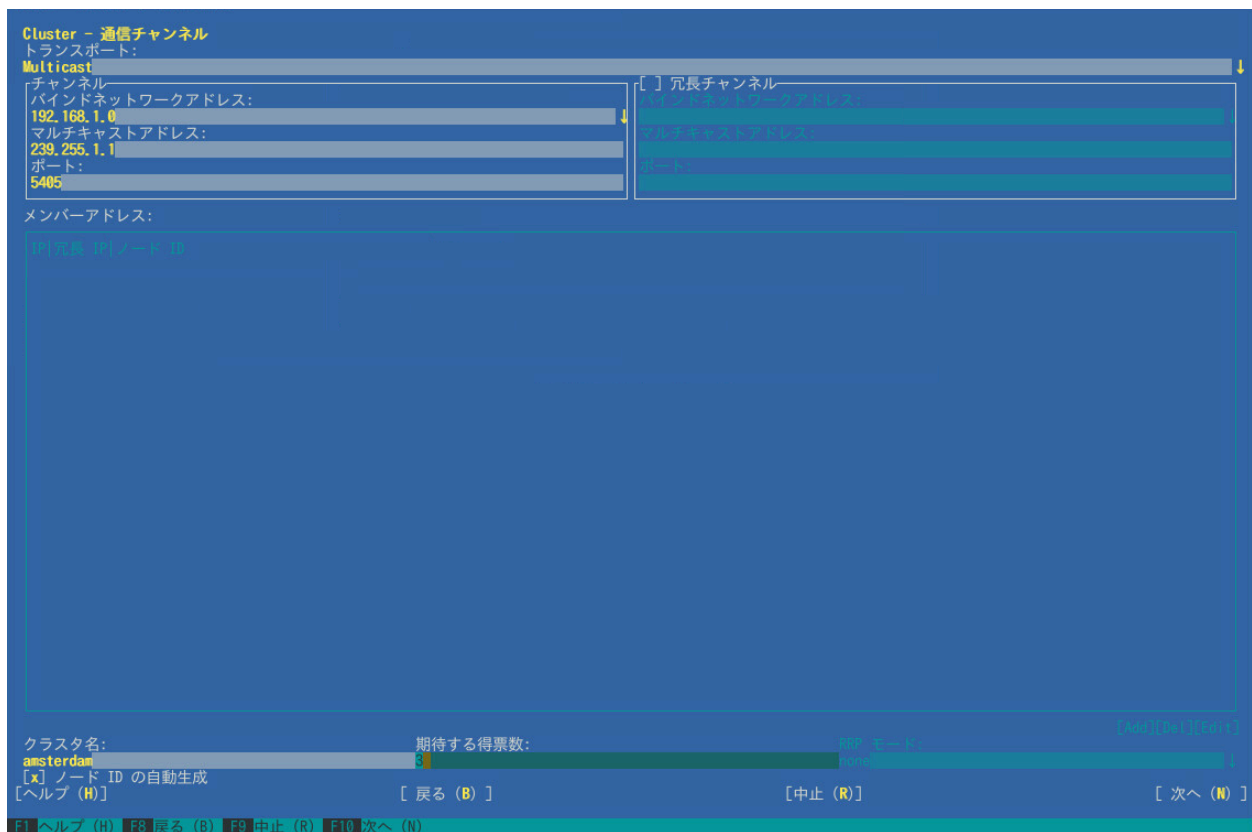


図 4.1: YASTクラスタ - マルチキャスト設定

クラスタ通信にマルチキャストではなくユニキャストを使用する場合は、次の手順に従います。

手順 4.2: 最初の通信チャネルの定義(ユニキャスト)

1. YaSTクラスタモジュールを起動して、通信チャネルカテゴリに切り替えます。
2. 転送プロトコルをUnicastに設定します。
3. ポートを定義します。
4. ユニキャスト通信では、Corosyncはクラスタ内のすべてのノードのIPアドレスを認識する必要があります。クラスタの一部になる各ノードで、追加をクリックし、次の詳細を入力します。
 - [IPアドレス]
 - [冗長IPアドレス] (Corosyncで2つ目の通信チャネルを使用する場合にのみ必要)
 - [ノードID] (ノードIDの自動生成オプションが無効になっている場合にのみ必要)

クラスタメンバーのアドレスを変更または削除するには、編集または削除ボタンを使用します。
5. クラスタノードごとに一意のIDを自動的に生成するには、ノードIDの自動生成を有効にしたままにします。
6. クラスタ名を定義します。
7. 期待する得票数の数を入力します。これは、パーティションされたクラスタでCorosyncがクォーラムを計算する場合に重要です。デフォルトでは、各ノードには1票が割り当てられています。期待する得票数の数は、クラスタ内のノード数と一致する必要があります。
8. 変更内容を確認します。
9. 必要な場合は、[手順4.3「冗長通信チャネルの定義」](#)で説明するように、Corosyncで冗長な通信チャネルを定義します。

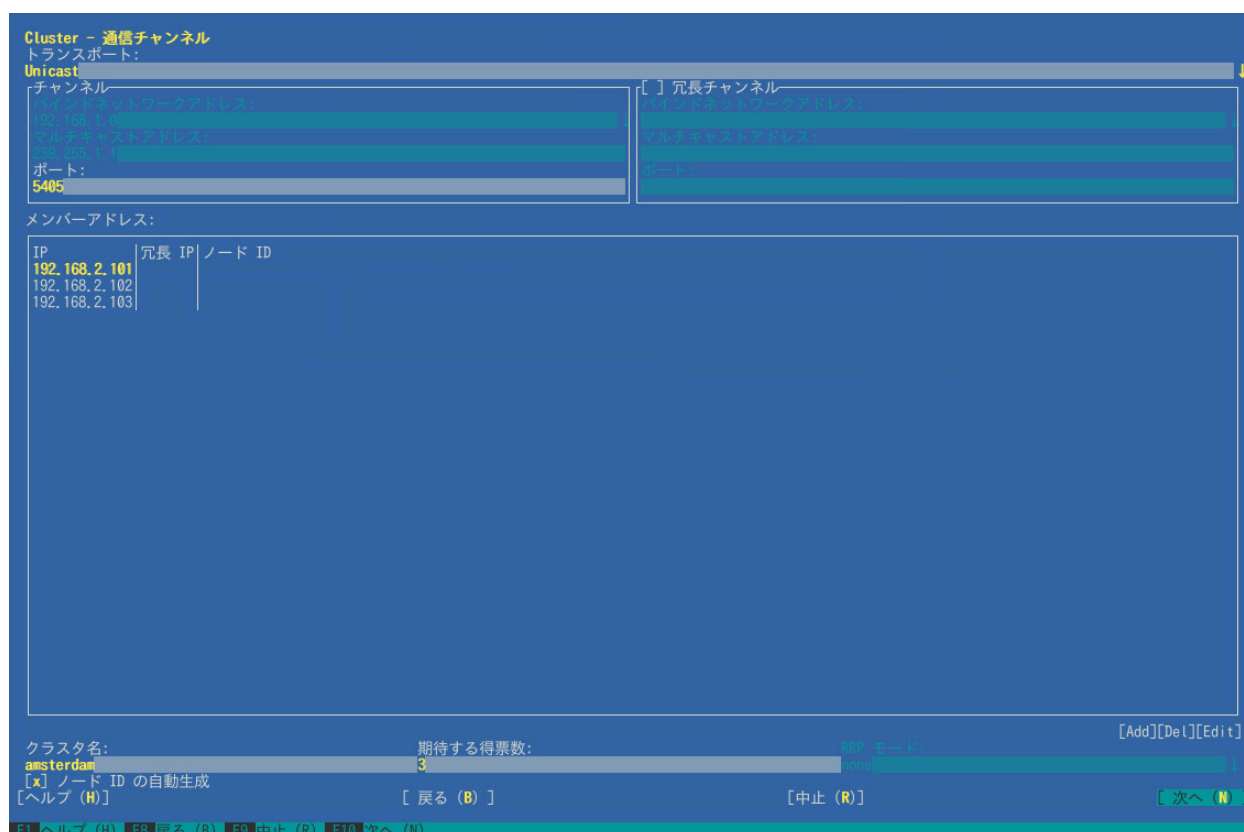


図 4.2: YAST クラスタ - ユニキャスト設定

ネットワークデバイスボンディングが何らかの理由で使用できない場合、第2の選択は、Corosyncに冗長通信チャンネル(2つ目のリング)を定義することです。この方法では、2つの物理的に分かれたネットワークが通信に使用できます。1つのネットワークが失敗しても、クラスタノードは、もう一方のネットワークを介して通信できます。

Corosync内の追加の通信チャンネルは2つ目のトークンパスリングを形成します。/etc/corosync/corosync.confでは、設定した最初のチャンネルはプライマリリングで、リング番号0を取得します。2つ目のリング(冗長チャンネル)はリング番号1を取得します。

Corosyncに定義済みの冗長通信チャンネルを持つ場合、RRPを使用してこれらのインターフェースの使用方法をクラスタに伝えます。RRPでは、2つの物理的に別個のネットワークが通信に使用されます。1つのネットワークが失敗しても、クラスタノードは、もう一方のネットワークを介して通信できます。

RRPでは次の3つのモードを使用できます。

- activeに設定した場合、Corosyncは両方のインタフェースをアクティブに使用します。ただし、このモードは非推奨の機能です。
- passiveに設定した場合、Corosyncは代わりに使用可能なネットワークを介してメッセージを送信します。
- noneに設定した場合、RRPは無効になります。

手順 4.3: 冗長通信チャネルの定義

！ 重要: 冗長リングおよび/etc/hosts

Corosync内で複数のリングが設定されている場合、各ノードが複数のIPアドレスを持つことができます。これはすべてのノードの/etc/hostsに反映する必要があります。

1. YaSTクラスタモジュールを起動して、通信チャネルカテゴリに切り替えます。
2. 冗長チャネルを有効にします。冗長チャネルは、定義した最初の通信チャネルと同じプロトコルを使用する必要があります。
3. マルチキャストを使用する場合は冗長チャネル用に次のパラメータを入力します:。使用するバインドネットワークアドレス、マルチキャストアドレス、およびポート。ユニキャストを使用する場合は次のパラメータを定義します:。使用するバインドネットワークアドレス、およびポート。クラスタに参加するすべてのノードのIPアドレスを入力します。
4. Corosyncに、異なるチャネルを使用する方法とタイミングを伝えるには、使用するrrp_modeを選択します。
 - 通信チャネルが1つだけ定義されている場合、rrp-modeが自動的に無効化されます(値なし)。
 - activeに設定した場合、Corosyncは両方のインタフェースをアクティブに使用します。ただし、このモードは非推奨の機能です。
 - passiveに設定した場合、Corosyncは代わりに使用可能なネットワークを介してメッセージを送信します。

RRPの使用時に、High Availability Extensionは現在のリングの状態を監視し、障害発生後に冗長リングを自動的に再度有効化します。

または、corosync-cfgtoolを使用してリングの状態を手動で確認します。使用可能なオプションは-hで参照できます。

5. 変更内容を確認します。

4.4 認証設定の定義

クラスタの認証設定を定義するには、HMAC/SHA1認証を使用できます。共有シークレットが必要なHMAC/SHA認証を使用して、メッセージを保護し、認証する必要があります。指定した認証キー(パスワード)が、クラスタ中のすべてのノードで使用されます。

手順 4.4: 安全な認証を有効にする

1. YaSTクラスタモジュールを起動し、セキュリティカテゴリに切り替えます。
2. 安全認証の有効化をオンにします。
3. 新しく作成したクラスタの場合は、認証キーファイルの生成をクリックします。認証キーが作成され、/etc/corosync/authkeyに書き込まれます。
ご使用のマシンを既存のクラスタに参加させたい場合、新しいキーファイルは生成しないでください。代わりに、いずれかのノードから/etc/corosync/authkeyを(手動またはCsync2によって)ご使用のマシンにコピーします。
4. 変更内容を確認します。YaSTが設定を/etc/corosync/corosync.confに書き込みます。

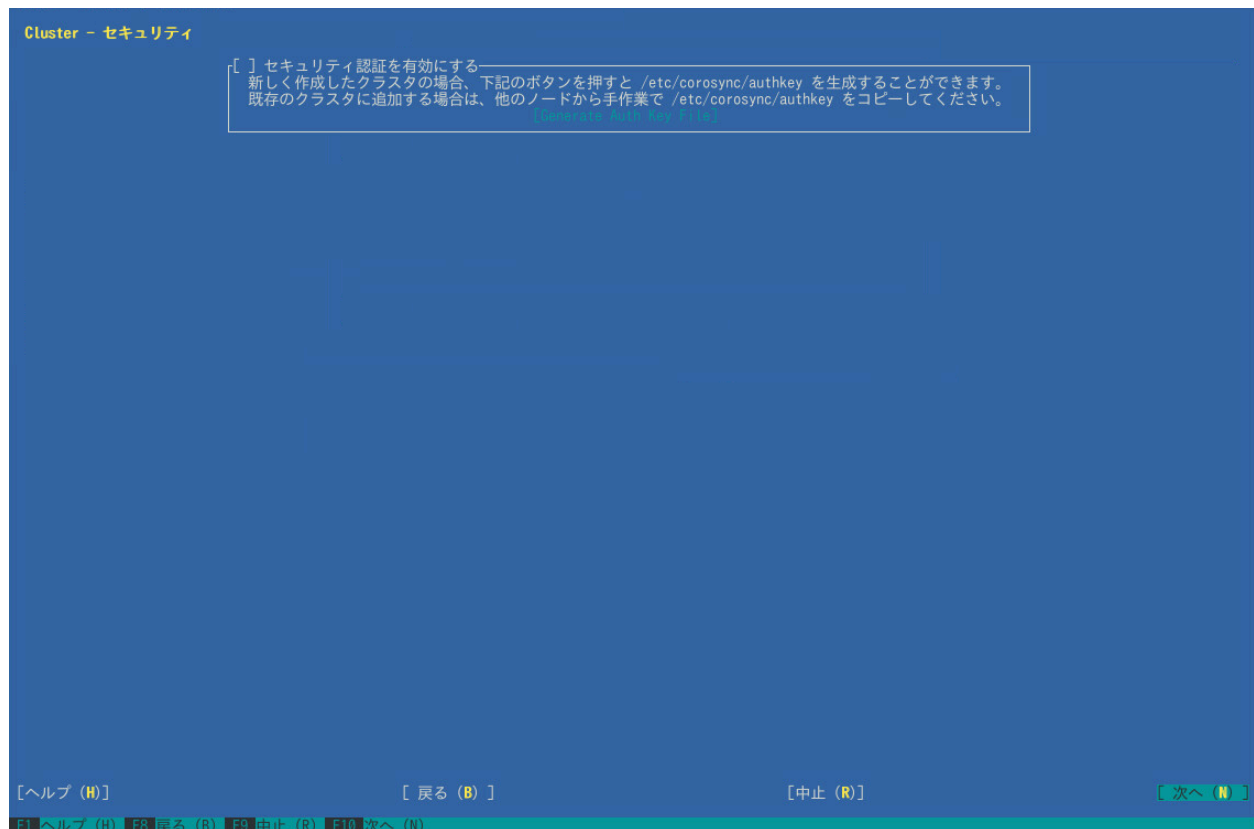


図 4.3: YASTクラスタ -セキュリティ

4.5 すべてのノードへの設定の転送

結果として生成された設定ファイルをすべてのノードに手動でコピーする代わりに、**csync2**ツールを使用して、クラスタ内のすべてのノードにレプリケートします。

これには、次の基本手順を必要とします。

1. YaSTによるCsync2の設定。
2. Csync2による設定ファイルの同期。

Csync2では、設定変更を追跡して、クラスタノード間でファイルの同期を取ることができます。

- 操作に対して重要なファイルのリストを定義できます。
- (他のクラスタノードに対して)これらのファイルの変更を表示できます。

- 1つのコマンドで複数の設定済みファイルの同期を取ることができます。
- `~/.bash_logout`の単純なシェルスクリプトを使用して、システムからログアウトする前に、同期化されていない変更について通知できます。

Csync2の詳細については、<http://oss.linbit.com/csync2/>と<http://oss.linbit.com/csync2/paper.pdf>にアクセスしてください。

4.5.1 YaSTによるCsync2の設定

1. YaSTクラスタモジュールを起動して、Csync2カテゴリに切り替えます。
2. 同期グループを指定するには、同期ホストグループで追加をクリックし、クラスタ内のすべてのノードのローカルホスト名を入力します。ノードごとに、`hostname`コマンドから返された文字列を正確に使用する必要があります。



ヒント: ホスト名解決

ホスト名解決がネットワークで正しく機能しない場合は、各クラスタノードのホスト名とIPアドレスの組み合わせを指定することもできます。この指定には、`HOSTNAME@IP`文字列(たとえば、`alice@192.168.2.100`)を使用します。Csync2は、接続時にIPアドレスを使用します。

3. 事前共有キーの生成をクリックして、同期グループのキーファイルを生成します。キーファイルは、`/etc/csync2/key_hagroup`に書き込まれます。このファイルは、作成後に、クラスタのすべてのメンバーに手動でコピーする必要があります。
4. すべてのノード間で、通常、同期される必要のあるファイルを同期ファイルリストに入れるには、推奨ファイルの追加をクリックします。
5. 同期するファイルのリストからファイルを編集、追加、または削除するには、該当する各ボタンを使用します。ファイルごとに絶対パス名を入力する必要があります。
6. Csync2をオンにするをクリックして、Csync2をアクティブにします。これによって次のコマンドが実行され、ブート時にCsync2が自動的に起動します。

```
root # systemctl enable csync2.socket
```

7. 変更内容を確認します。YaSTがCsync2の設定内容を`/etc/csync2/csync2.cfg`に書き込みます。

8. ここで同期プロセスを開始するには、4.5.2項「Csync2を使用した変更内容の同期」で続行します。

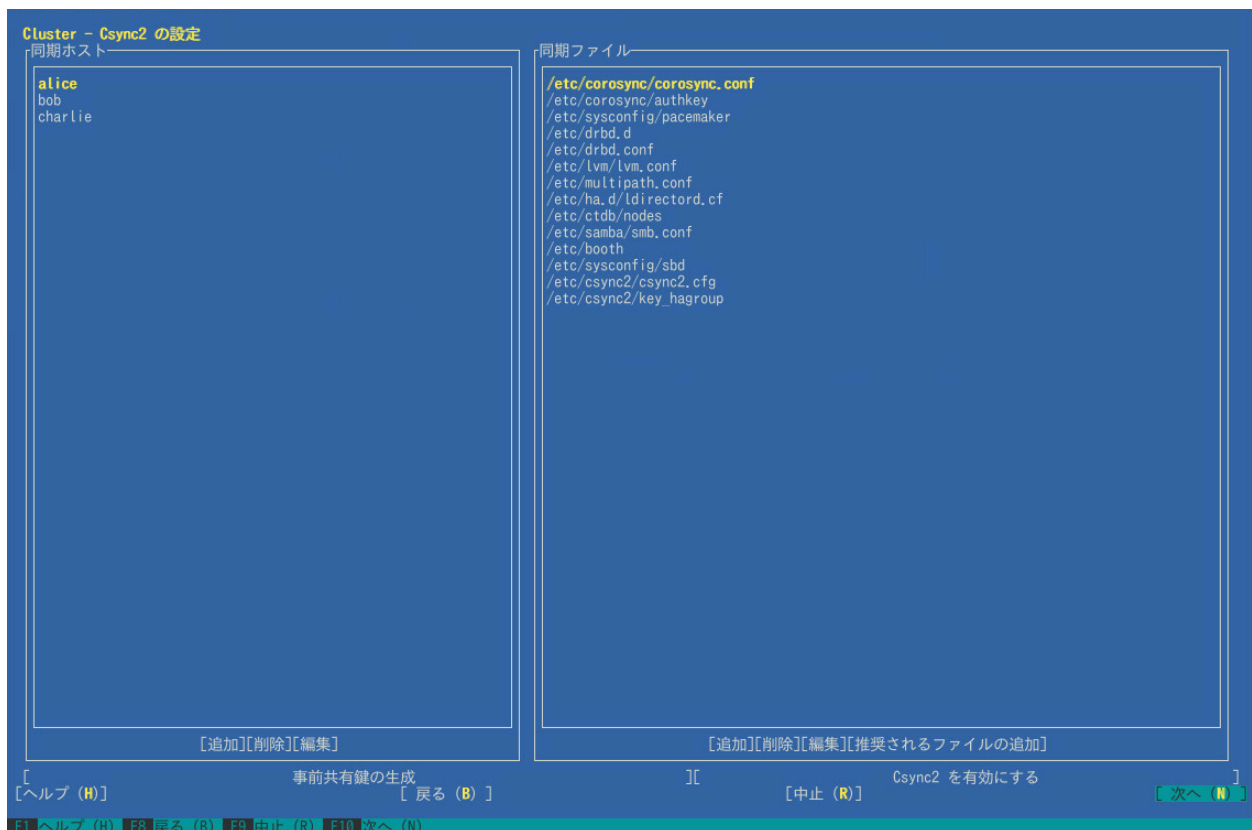


図 4.4: YAST クラスタ - CSYNC2

4.5.2 Csync2を使用した変更内容の同期

Csync2を使用してファイルを正常に同期するには、以下の前提条件が満たされている必要があります。

- 同じCsync2設定をすべてのクラスタノードで使用する必要があります。
- 同じCsync2認証キーをすべてのクラスタノードで使用する必要があります。
- Csync2はすべてのクラスタノード上で実行されている必要があります。

したがって、Csync2を初めて実行する前に、以下の準備を行う必要があります。

手順 4.5: CSYNC2による初期同期の準備

1. ファイル `/etc/csync2/csync2.cfg` を、4.5.1項「YaSTによるCsync2の設定」で説明されるとおりに設定した後、すべてのノードに手動でコピーします。

2. 4.5.1項のステップ3の1つのノードで作成した/etc/csync2/key_hagroupファイルを、クラスタ内のすべてのノードにコピーしてください。このファイルは、Csync2による認証で必要になります。ただし、すべてのノードで同じファイルでなければならないので、他のノードではファイルを再生成しないでください。
3. すべてのノード上で次のコマンドを実行して、Csync2サービスを今すぐ開始します。

```
root # systemctl start csync2.socket
```

手順 4.6: CSYNC2による設定ファイルの同期

1. 最初にすべてのファイルを一度同期させるには、設定の「コピー元」であるマシン上で次のコマンドを実行します。

```
root # csync2 -xv
```

これによって、すべてのファイルをその他のノードにプッシュすることで、一度に同期を行います。すべてのファイルが正常に同期されると、Csync2がエラーなしで終了します。

同期対象の1つ以上のファイルが(現在のノードだけでなく)他のノード上で変更されている場合は、Csync2から衝突が報告されます。次の出力とよく似た出力が表示されます。

```
While syncing file /etc/corosync/corosync.conf:  
ERROR from peer hex-14: File is also marked dirty here!  
Finished with 1 errors.
```

2. 現在のノードのファイルバージョンが「最良」だと確信する場合は、そのファイルを強制して再同期を行い、競合を解決できます。

```
root # csync2 -f /etc/corosync/corosync.conf  
root # csync2 -x
```

Csync2オプションの詳細については、次のコマンドを実行してください

```
csync2 -help
```



注記: 変更後の同期のプッシュ

Csync2は変更のみをプッシュします。Csync2はマシン間でファイルを絶えず同期しているわけではありません。

同期が必要なファイルを更新する際はいつも、変更を加えたマシン上で `csync2 -xv` を実行することで、変更をその他のマシンにプッシュする必要があります。変更されていないファイルが配置された他のマシン上でこのコマンドを実行しても、何も起こりません。

4.6 クラスタノード間の接続ステータスの同期

iptablesに対して「ステートフルな」パケット検査ができるようにするには、conntrackツールを設定して使用します。これには、次の基本手順を必要とします。

手順 4.7: YASTによるconntrackdの設定

YaSTクラスタモジュールを使用して、ユーザスペースconntrackdを設定します(図 4.5 「YaSTクラスタ - conntrackd」を参照してください)。これには、その他の通信チャネルに使用されていない専用のネットワークインタフェースが必要です。デーモンは後でリソースエージェントによって起動できます。

1. YaSTクラスタモジュールを起動して、conntrackdの設定カテゴリに切り替えます。
2. 接続ステータスの同期に使用するマルチキャストアドレスを定義します。
3. グループ番号で、接続ステータスを同期させるグループのID番号を定義します。
4. `/etc/conntrackd/conntrackd.conf` の生成をクリックして、conntrackdの設定ファイルを作成します。
5. 既存のクラスタでオプションを変更した場合、変更を確認して、クラスタモジュールを終了します。
6. クラスタ設定を先に進めるには、次へをクリックして、4.7項「サービスの設定」で続行します。
7. 専用インタフェースを選択して、接続ステータスを同期します。選択したインタフェースのIPv4アドレスが自動的に検出され、YaSTに表示されます。これはすでに設定済みで、マルチキャストをサポートしている必要があります。



図 4.5: YAST クラスタ - conntrackd

conntrack ツールを設定したら、これを Linux Virtual Server で使用できます([負荷分散](#)を参照してください)。

4.7 サービスの設定

YaST クラスタモジュールは、ブート時にノード上で一定のサービスを開始するかどうか定義します。サービスを手動で開始または停止するためにモジュールを使用することもできます。クラスタノードをオンラインにし、クラスタリソースマネージャを起動するには、Pacemaker をサービスとして実行する必要があります。

手順 4.8: PACEMAKER の有効化

1. YaST クラスタモジュール内で、サービスカテゴリに切り替えます。
2. このクラスタノードがブートするたびに Pacemaker を起動するには、**起動中グループ**で該当するオプションを選択します。起動中グループで、オフを選択する場合は、このノードがブートするたびに手動で Pacemaker を起動する必要があります。Pacemaker を手動で起動するには、次のコマンドを使用します。

```
root # crm cluster start
```

3. Pacemakerをただちに起動または停止するには、それぞれのボタンをクリックします。
4. 現在のマシン上でのクラスタ通信に必要なポートをファイアウォールで開くには、ファイアウォールでポートを開くをアクティブにします。
5. 変更内容を確認します。この設定は、すべてのクラスタノードではなく、ご使用のマシンにのみ適用されることにご注意ください。

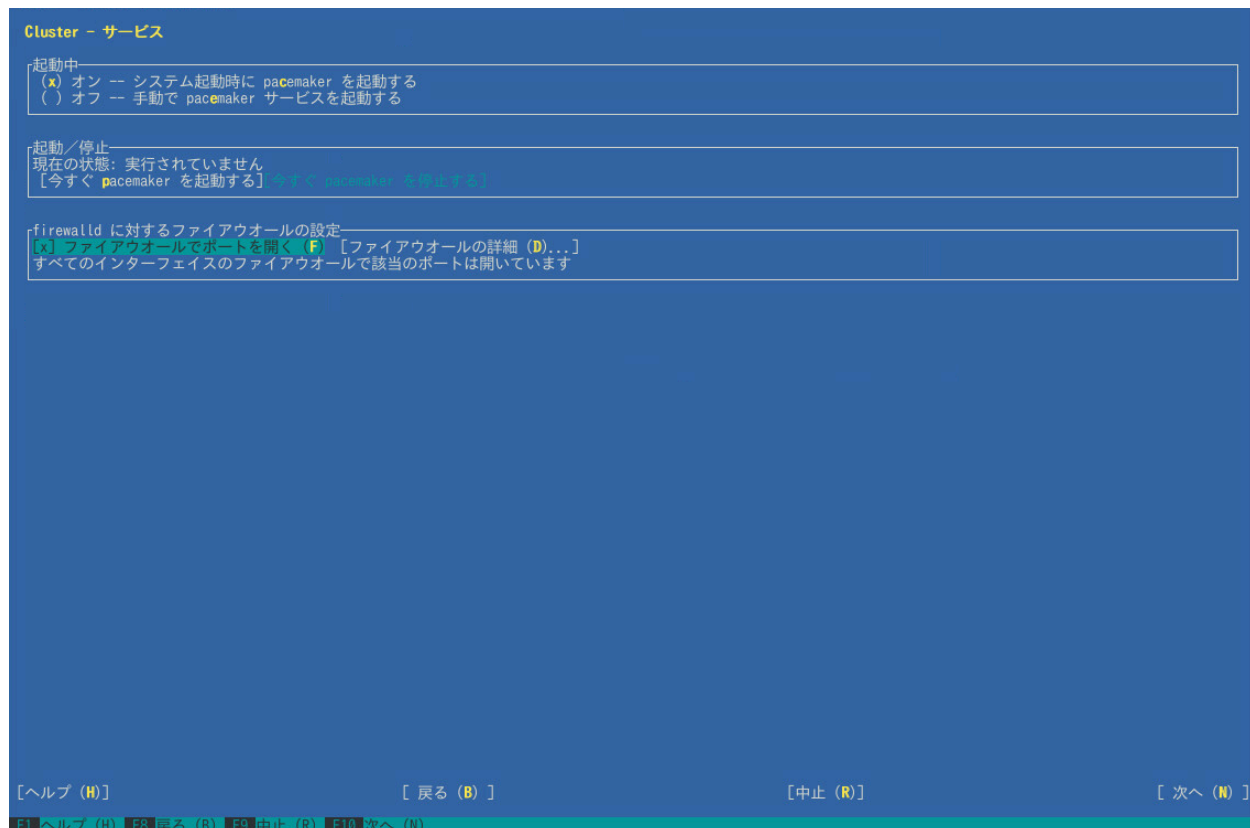


図 4.6: YASTクラスタ - サービス

4.8 ノードごとにクラスタをオンラインにする

最初のクラスタ設定が完了した後、各クラスタノード上でクラスタサービスを開始し、スタックをオンラインにします。

手順 4.9: クラスタサービスの開始とステータスの確認

1. 既存のノードにログインします。

2. サービスがすでに実行していることを確認します。

```
root # crm cluster status
```

実行していない場合は、クラスタサービスを今すぐ開始します。

```
root # crm cluster start
```

3. それぞれのクラスタノードに対してこの手順を繰り返します。
4. ノードの1つで、**crm status**コマンドを使用してクラスタの状態を確認します。すべてのノードがオンラインの場合、出力は次のようになります。

```
root # crm status
Cluster Summary:
* Stack: corosync
* Current DC: alice (version ...) - partition with quorum
* Last updated: ...
* Last change: ... by hacluster via crmd on bob
* 2 nodes configured
* 1 resource instance configured

Node List:
* Online: [ alice bob ]
...
```

この出力は、クラスタリソースマネージャが起動し、リソースを管理できる状態にあることを示しています。

基本設定を完了し、ノードがオンラインになったら、クラスタリソースの設定を開始できます。crmシェル(crmsh)やHawk2などのクラスタ管理ツールのいずれかを使用します。詳細については、[第8章「クラスタリソースの設定と管理\(コマンドライン\)」](#)または[第7章「Hawk2を使用したクラスタリソースの設定と管理」](#)を参照してください。

5 クラスタアップグレードとソフトウェアパッケージの更新

この章では、次の2つの異なるシナリオ(SUSE Linux Enterprise High Availability Extensionの別バージョン(メジャーリリースまたはサービスパック)へのクラスタアップグレード、およびクラスタノード上の各パッケージの更新)について説明します。5.2項「最新の製品バージョンへのクラスタアップグレード」および5.3項「クラスタノード上のソフトウェアパッケージの更新」を参照してください。

クラスタをアップグレードする場合、アップグレードを開始する前に、5.2.1項「SLE HAおよびSLE HA Geoでサポートされるアップグレードパス」および5.2.2項「アップグレード前に必要な準備」を確認してください。

5.1 用語集

次に、この章で使用される最も重要な用語の定義を示します。

メジャーリリース、 一般出荷(GA)バージョン

メジャーリリースとは、新しい機能やツールを導入し、非推奨になっていたコンポーネントを削除した、新しい製品バージョンです。これには、後方互換性のない変更が存在します。

クラスタオフラインアップグレード

新しい製品バージョンに後方互換性のない大きな変更が含まれている場合は、クラスタオフラインアップグレードによってクラスタをアップグレードする必要があります。すべてのノードをオンラインに戻す前に、すべてのノードをオフラインにして、クラスタ全体をアップグレードする必要があります。

クラスタローリングアップグレード

クラスタローリングアップグレードでは、一度に1つのクラスタノードがアップグレードされ、残りのクラスタはまだ実行されています。最初のノードをオフラインにし、アップグレードして、オンラインに戻してから、クラスタに参加させます。その後、すべてのクラスタノードがメジャーバージョンにアップグレードされるまで、1つずつアップグレードを続けます。

サービスパック(SP)

複数のパッチを組み合わせて、インストールまたは展開しやすい形式にします。サービスパックには番号が付けられ、通常、プログラムのセキュリティ修正、更新、アップグレード、または拡張機能が含まれます。

この状態から

新しい「マイナー」バージョンのパッケージのインストールです。通常、セキュリティの修正やその他の重要な修正が含まれています。

アップグレード

パッケージまたは配布の新しい「主要」バージョンのインストール。これにより「新機能」がもたらされます。[クラスタオフラインアップグレード](#)および[クラスタローリングアップグレード](#)も参照してください。

5.2 最新の製品バージョンへのクラスタアップグレード

サポートされるアップグレードパスおよびアップグレードの実行方法は、現在の製品バージョンおよび移行したいターゲットバージョンによって異なります。

High Availability Extensionには、基盤となる基本システムと同じサポートされているアップグレードパスがあります。完全な概要については、『SUSE Linux Enterprise Serverアップグレードガイド』の「SUSE Linux Enterprise Server 15 SP3へのサポートされているアップグレードパス (<https://documentation.suse.com/sles-15/html/SLES-all/cha-upgrade-paths.html#sec-upgrade-paths-supported>) [🔗](#)」を参照してください。

また、高可用性クラスタスタックはクラスタをアップグレードするための2つの方法を提供するため、次のルールが適用されます。

- **クラスタローリングアップグレード**. クラスタローリングアップグレードは、同じメジャーリリース内(あるサービスパックから次のサービスパックへ、または製品のGAバージョンからSP1へ)でのみサポートされます。
- **クラスタオフラインアップグレード**. あるメジャーリリースから次のメジャーリリース(たとえば、SLE HA 12からSLE HA 15など)、または特定のメジャーリリース内のサービスパックから次のメジャーリリース(たとえば、SLE HA 12 SP3からSLE HA 15)へアップグレードするには、クラスタオフラインアップグレードが必要です。

5.2.1項では、あるバージョンから次のバージョンへ移行するためにSLE HA (Geo)でサポートされているアップグレードパスと方法を一覧表示しています。列詳細情報の参照先には、参照する必要がある特定のアップグレードドキュメント(基本システムとGeo Clustering for SUSE Linux Enterprise High Availability Extensionも含まれます)が一覧表示されています。このマニュアルは以下から入手できます。

- <https://documentation.suse.com/sles-15>
- <https://documentation.suse.com/sle-ha-15>

！ 重要: 混合クラスタおよびアップグレード後に元に戻す操作はサポートされない

- SUSE Linux Enterprise High Availability Extension 12/SUSE Linux Enterprise High Availability Extension 15で実行する混合クラスタはサポートされていません。
- 製品バージョン15へのアップグレードプロセス後に、製品バージョン12に戻す処理は、サポートされていません。

5.2.1 SLE HAおよびSLE HA Geoでサポートされるアップグレードパス

アップグレード元とアップグレード先	アップグレードパス	詳細情報の参照先
SLE HA 11 SP3から SLE HA (Geo) 12	クラスタオフラインアップグレード	<ul style="list-style-type: none">• 基本システム: SLES 12用の『導入ガイド』の「Updating and Upgrading SUSE Linux Enterprise」のパート• SLE HA: 製品バージョン11から12へのアップグレード: クラスタオフラインアップグレード• SLE HA Geo: SLE HA 12用の『Geo Clustering クイックスタート』の「Upgrading from SLE HA (Geo) 11 SP 3 to SLE HA Geo 12」の項

アップグレード元とアップグレード先	アップグレードパス	詳細情報の参照先
SLE HA (Geo) 11 SP4からSLE HA (Geo) 12 SP1	クラスタオフラインアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 12 SP1用の『導入ガイド』の「Updating and Upgrading SUSE Linux Enterprise」のパート SLE HA: 製品バージョン11から12へのアップグレード: クラスタオフラインアップグレード SLE HA Geo: SLE HA 12 SP1用の『Geo Clustering クイックスタート』の「Upgrading to the Latest Product Version」の項
SLE HA (Geo) 12からSLE HA (Geo) 12 SP1	クラスタローリングアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 12 SP1用の『導入ガイド』の「Updating and Upgrading SUSE Linux Enterprise」のパート SLE HA: クラスタローリングアップグレードの実行 SLE HA Geo: SLE HA 12 SP1用の『Geo Clustering クイックスタート』の「Upgrading to the Latest Product Version」の項
SLE HA (Geo) 12 SP1からSLE HA (Geo) 12 SP2	クラスタローリングアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 12 SP2用の『導入ガイド』の「Updating and Upgrading SUSE Linux Enterprise」のパート SLE HA: クラスタローリングアップグレードの実行 SLE HA Geo: SLE HA 12 SP2用の『Geo Clustering クイックスタート』の「Upgrading to the Latest Product Version」の項 DRBD 8から DRBD 9への移行DRBD 8から DRBD 9への移行

アップグレード元とアップグレード先	アップグレードパス	詳細情報の参照先
SLE HA (Geo) 12 SP2からSLE HA (Geo) 12 SP3	クラスタローリングアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 12 SP3用の『導入ガイド』の「Updating and Upgrading SUSE Linux Enterprise」のパート SLE HA: クラスタローリングアップグレードの実行 SLE HA Geo: SLE HA 12 SP3用の『Geo Clusteringガイド』の「Upgrading to the Latest Product Version」の項
SLE HA (Geo) 12 SP3からSLE HA (Geo) 12 SP4	クラスタローリングアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 12 SP4用の『導入ガイド』の「Updating and Upgrading SUSE Linux Enterprise」のパート SLE HA: クラスタローリングアップグレードの実行 SLE HA Geo: SLE HA 12 SP4用の『Geo Clusteringガイド』の「Upgrading to the Latest Product Version」の項
SLE HA (Geo) 12 SP3からSLE HA (Geo) 15	クラスタオフラインアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 15 用の『アップグレードガイド』 SLE HA: 製品バージョン12から15へのアップグレード: クラスタオフラインアップグレード SLE HA Geo: 『Geo Clustering Guide』、第10章「Upgrading to the latest product version」 Cluster LVM: Mirror LVからCluster MDへのオンラインマイグレーション

アップグレード元とアップグレード先	アップグレードパス	詳細情報の参照先
SLE HA (Geo) 12 SP4からSLE HA (Geo) 12 SP5	クラスタローリングアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 12 SP5用の『導入ガイド』の「Updating and Upgrading SUSE Linux Enterprise」のパート SLE HA: クラスタローリングアップグレードの実行 SLE HA Geo: 『Geo Clustering Guide』、第10章「Upgrading to the latest product version」
SLE HA (Geo) 12 SP4からSLE HA (Geo) 15 SP1	クラスタオフラインアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 15 SP1用の『アップグレードガイド』 SLE HA: 製品バージョン12から15へのアップグレード: クラスタオフラインアップグレード SLE HA Geo: 『Geo Clustering Guide』、第10章「Upgrading to the latest product version」 Cluster LVM: Mirror LVからCluster MDへのオンラインマイグレーション
SLE HA (Geo) 12 SP5からSLE HA (Geo) 15 SP2	クラスタオフラインアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 15 SP2用の『アップグレードガイド』 SLE HA: 製品バージョン12から15へのアップグレード: クラスタオフラインアップグレード SLE HA Geo: 『Geo Clustering Guide』、第10章「Upgrading to the latest product version」 Cluster LVM: Mirror LVからCluster MDへのオンラインマイグレーション

アップグレード元とアップグレード先	アップグレードパス	詳細情報の参照先
SLE HA (Geo) 15から SLE HA (Geo) 15 SP1	クラスタローリングアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 15 SP1用の『アップグレードガイド』 SLE HA: クラスタローリングアップグレードの実行 SLE HA Geo: 『Geo Clustering Guide』、第10章「Upgrading to the latest product version」
SLE HA (Geo) 15 SP1から SLE HA (Geo) 15 SP2	クラスタローリングアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 15 SP2用の『アップグレードガイド』 SLE HA: クラスタローリングアップグレードの実行 SLE HA Geo: 『Geo Clustering Guide』、第10章「Upgrading to the latest product version」
SLE HA (Geo) 15 SP2から SLE HA (Geo) 15 SP3	クラスタローリングアップグレード	<ul style="list-style-type: none"> 基本システム: SLES 15 SP3用の『アップグレードガイド』 SLE HA: クラスタローリングアップグレードの実行 SLE HA Geo: 『Geo Clustering Guide』、第10章「Upgrading to the latest product version」



注記: サービスパックのスキップ

最も単純なアップグレードパスは、すべてのサービスパックを連続してインストールすることです。SUSE Linux Enterprise 15製品ライン(GAとそれ以降のサービスパック)については、アップグレード時に1つのサービスパックをスキップすることもサポートされています。たとえば、SLE HA 15 GAからSLE HA 15 SP2へのアップグレードや、SLE HA 15 SP1からSLE HA 15 SP3へのアップグレードがサポートされます。

5.2.2 アップグレード前に必要な準備

バックアップ

システムバックアップが最新で、復元可能かどうかを確認します。

テスト

運用環境で実行する前に、まず、クラスタセットアップのステージングインスタンスでアップグレード手順をテストします。これにより、メンテナンス期間に要するタイムフレームを予測できます。発生する可能性のある予期しない問題を検出し、解決するのにも役立ちます。

5.2.3 クラスタオフラインアップグレード

この項は次のシナリオに適用されます。

- SLE HA 11 SP3からSLE HA 12へのアップグレード - 詳細については、[手順5.1「製品バージョン11から12へのアップグレード: クラスタオフラインアップグレード」](#)を参照してください。
- SLE HA 11 SP4からSLE HA 12 SP1へのアップグレード - 詳細については、[手順5.1「製品バージョン11から12へのアップグレード: クラスタオフラインアップグレード」](#)を参照してください。
- SLE HA 12 SP3からSLE HA 15へのアップグレード - 詳細については、[手順5.2「製品バージョン12から15へのアップグレード: クラスタオフラインアップグレード」](#)を参照してください。
- SLE HA 12 SP4からSLE HA 15 SP1へのアップグレード - 詳細については、[手順5.2「製品バージョン12から15へのアップグレード: クラスタオフラインアップグレード」](#)を参照してください。
- SLE HA 12 SP5からSLE HA 15 SP2へのアップグレード - 詳細については、[手順5.2「製品バージョン12から15へのアップグレード: クラスタオフラインアップグレード」](#)を参照してください。

クラスタがまだこれらより古い製品バージョンに基づいている場合は、まず、必要なターゲットバージョンにアップグレードするためのソースとして使用できるバージョンのSLESおよびSLE HAにクラスタをアップグレードします。

手順 5.1: 製品バージョン11から12へのアップグレード: クラスタオフラインアップグレード

High Availability Extension 12クラスタスタックでは、さまざまなコンポーネントに大幅な変更が導入されています(たとえば、`/etc/corosync/corosync.conf`、OCFS2のディスクフォーマットなど)。したがって、SUSE Linux Enterprise High Availability Extension 11バージョンからのクラスタローリングアップグレードはサポートされません。代わりに、次に示すように、すべてのクラスタノードをオフラインにしてから、クラスタ全体をアップグレードする必要があります。

1. 各クラスタノードにログインし、次のコマンドを使用してクラスタスタックを停止します。

```
root # rcopenais stop
```

2. クラスタノードごとに、目的のターゲットバージョンのSUSE Linux Enterprise ServerおよびSUSE Linux Enterprise High Availability Extensionへのアップグレードを実行します。5.2.1項「SLE HAおよびSLE HA Geoでサポートされるアップグレードパス」を参照してください。
3. アップグレードプロセスが完了した後で、SUSE Linux Enterprise ServerおよびSUSE Linux Enterprise High Availability Extensionのアップグレード済みバージョンがインストールされた各ノードを再起動します。
4. クラスタセットアップでOCFS2を使用する場合は、次のコマンドを実行して、オンデバイス構造をアップデートします。

```
root # o2cluster --update PATH_TO_DEVICE
```

追加のパラメータがディスクに追加されます。これらのパラメータは、SUSE Linux Enterprise High Availability Extension 12および12 SPxに付属しているアップデートされたOCFS2バージョンで必要になります。

5. Corosyncバージョン2の`/etc/corosync/corosync.conf`をアップデートするには:
 - a. 1つのノードにログインして、YaSTクラスタモジュールを起動します。
 - b. 通信チャネルカテゴリに切り替えて、新しいパラメータ(クラスタ名および期待する得票数)の値を入力します。詳細については、手順4.1「最初の通信チャネルの定義(マルチキャスト)」または手順4.2「最初の通信チャネルの定義(ユニキャスト)」をそれぞれ参照してください。
Corosyncバージョン2で無効になっていたり、見つからない他のオプションをYaSTが検出する場合、変更を求めるプロンプトが表示されます。

- c. YaSTで変更内容を確認します。YaSTが変更を /etc/corosync/corosync.conf に書き込みます。
- d. クラスタに対してCsync2が設定されている場合は、次のコマンドを使用して、アップデートされたCorosync設定を他のクラスタノードにプッシュします。

```
root # csync2 -xv
```

Csync2の詳細については、4.5項「すべてのノードへの設定の転送」を参照してください。

または、/etc/corosync/corosync.confをすべてのクラスタノードに手動でコピーして、更新されたCorosync設定の同期を取ります。

6. 各ノードにログインして、次のコマンドを使用してクラスタスタックを起動します。

```
root # crm cluster start
```

7. **crm status**またはHawk2を使用してクラスタの状態を確認します。

8. ブート時に起動するように以下のサービスを設定します。

```
root # systemctl enable pacemaker
root # systemctl enable hawk
root # systemctl enable sbd
```



注記: CIB構文バージョンのアップグレード

新しい機能は、最新のCIB構文バージョンでのみ使用可能な場合もあります。新しい製品バージョンにアップグレードした場合でも、デフォルトではCIB構文バージョンはアップグレードされません。

1. バージョンをチェックします。

```
cibadmin -Q | grep validate-with
```

2. 最新のCIB構文バージョンにアップグレードします。

```
root # cibadmin --upgrade --force
```

！ 重要: 最初からのインストール

クラスタノードを(アップグレードするのではなく)最初からインストールすることにした場合、2.2項「ソフトウェアの必要条件」SUSE Linux Enterprise High Availability Extension 15 SP3に必要なモジュールのリストについて、を参照してください。モジュール、拡張機能、および関連製品の詳細については、SUSE Linux Enterprise Server 15のリリースノートを確認してください。これらは、<https://www.suse.com/releasesnotes/>から入手できます。

1. SUSE Linux Enterprise High Availability Extension 15へのオフラインアップグレードを開始する前に、注記: CIB構文バージョンのアップグレードの説明に従って現在のクラスタでCIB構文を手動でアップグレードしてください。
2. 各クラスタノードにログインし、次のコマンドを使用してクラスタスタックを停止します。

```
root # crm cluster stop
```

3. クラスタノードごとに、目的のターゲットバージョンのSUSE Linux Enterprise ServerおよびSUSE Linux Enterprise High Availability Extensionへのアップグレードを実行します。5.2.1項「SLE HAおよびSLE HA Geoでサポートされるアップグレードパス」を参照してください。
4. アップグレードプロセスが完了した後で、SUSE Linux Enterprise ServerおよびSUSE Linux Enterprise High Availability Extensionのアップグレード済みバージョンがインストールされた各ノードにログインして起動します。
5. Cluster LVMを使用する場合は、clvmdからlvmlockdに移行する必要があります。lvmlockdのマニュアルページの「changing a clvm VG to a lockd VG」および22.4項「Mirror LVからCluster MDへのオンラインマイグレーション」を参照してください。
6. 次のコマンドを実行してこのクラスタスタックを開始します。

```
root # crm cluster start
```

7. `crm status`またはHawk2を使用してクラスタの状態を確認します。

5.2.4 クラスタローリングアップグレード

この項は次のシナリオに適用されます。

- SLE HA 12からSLE HA 12 SP1へのアップグレード
- SLE HA 12 SP1からSLE HA 12 SP2へのアップグレード
- SLE HA 12 SP2からSLE HA 12 SP3へのアップグレード
- SLE HA 12 SP3からSLE HA 12 SP4へのアップグレード
- SLE HA 12 SP4からSLE HA 12 SP5へのアップグレード
- SLE HA 15からSLE HA 15 SP1へのアップグレード
- SLE HA 15 SP1からSLE HA 15 SP2へのアップグレード
- SLE HA 15 SP2からSLE HA 15 SP3へのアップグレード

シナリオに合わせて次の手順のいずれかを使用します。

- 一般的なローリングアップグレードの詳細については、[手順 5.3](#)を参照してください。
- 特定のローリングアップグレードについては、[手順 5.4](#)を参照してください。



警告: アクティブなクラスタスタック

ノードのアップグレードを開始する前に、「そのノード上の」クラスタスタックを「停止」します。

ソフトウェアの更新中にノード上のクラスタリソースマネージャがアクティブな場合、アクティブノードのフェンシングのような結果を招く場合があります。



重要: クラスタローリングアップグレードの時間制限

最新の製品バージョンに装備されている新機能は、「すべての」クラスタノードが最新の製品バージョンにアップグレードされた後でないと使用できません。混合バージョンのクラスタは、クラスタローリングアップグレード中の短いタイムフレームでのみサポートされます。クラスタローリングアップグレードは1週間以内に完了してください。

すべてのオンラインノードでアップグレードされたバージョンが実行される場合、古いバージョンの他のノードはアップグレードせずに(再)参加できなくなります。

手順 5.3: クラスタローリングアップグレードの実行

1. アップグレードするノードでrootとしてログインし、クラスタスタックを停止します。


```
root # crm cluster stop
```

2. 目的のターゲットバージョンのSUSE Linux Enterprise ServerおよびSUSE Linux Enterprise High Availability Extensionへのアップグレードを実行します。個々のアップグレードプロセスの詳細を確認するには、5.2.1項「SLE HAおよびSLE HA Geoでサポートされるアップグレードパス」を参照してください。
3. アップグレードしたノードでクラスタスタックを起動して、ノードをクラスタに再加入させます。

```
root # crm cluster start
```

4. 次のノードをオフラインにし、そのノードに関して手順を繰り返します。
5. **crm status**またはHawk2を使用してクラスタの状態を確認します。
クラスタノードに異なるCRMバージョンが検出される場合、Hawk2の状態画面に警告も表示されます。

！ 重要: ローリングアップグレードの時間制限

最新の製品バージョンに装備されている新機能は、「すべての」クラスタノードが最新の製品バージョンにアップグレードされた後でないと使用できません。混合バージョンのクラスタは、ローリングアップグレード中の短いタイムフレームでのみサポートされます。ローリングアップグレードは1週間以内に完了してください。

クラスタノードに異なるCRMバージョンが検出される場合、Hawk2の状態画面に警告も表示されます。

多くのお客様は、インプレースアップグレードのほかに、次のサービスパックに移行する場合でも新規インストールを希望しています。次の手順では、ノードaliceとbobを持つ2ノードクラスタを次のサービスパック(SP)にアップグレードする場合のシナリオを示しています。

手順 5.4: 新しいサービスパックのクラスタ全体での新規インストールの実行

1. クラスタ設定のバックアップを作成します。次のリストに、最小ファイルセットを示します。

```
/etc/corosync/corosync.conf  
/etc/corosync/authkey  
/etc/sysconfig/sbd  
/etc/modules-load.d/watchdog.conf  
/etc/hosts
```

```
/etc/ntp.conf
```

リソースに応じて、次のファイルが必要になる場合もあります。

```
/etc/services  
/etc/passwd  
/etc/shadow  
/etc/groups  
/etc/drbd/*  
/etc/lvm/lvm.conf  
/etc/mdadm.conf  
/etc/mdadm.SID.conf
```

2. ノードaliceから開始します。

- a. ノードをスタンバイモードにします。これにより、リソースをノードから移動できます。

```
root # crm --wait node standby alice reboot
```

オプション`--wait`を使用すると、クラスタが遷移を終えて、アイドル状態になったときにのみコマンドが返されます。`reboot`オプションには、ノードがもう一度オンラインになったときにはスタンバイモードではなくなるという効果があります。その名前にかかわらず、`reboot`オプションは、ノードがオフラインおよびオンラインになっている限り機能します。

- b. ノードaliceでクラスタサービスを停止します。

```
root # crm cluster stop
```

- c. この時点で、aliceには実行中のリソースはありません。ノードaliceをアップグレードして後で再起動します。クラスタサービスは起動時には開始されないことが想定されます。

- d. バックアップファイルをステップ1から元の場所にコピーします。

- e. ノードaliceをクラスタに戻します。

```
root # crm cluster start
```

- f. リソースに問題がないことを確認します。

3. ノードbobに対してステップ2を繰り返します。

5.3 クラスタノード上のソフトウェアパッケージの更新



警告: アクティブなクラスタスタック

ノードのパッケージ更新を開始する前に、クラスタスタックが影響を受けるか否かによって、「そのノード上の」クラスタスタックを「停止」するか、「ノードを保守モード」にします。詳細については[ステップ 1](#)を参照してください。

ソフトウェアの更新中にノード上のクラスタリソースマネージャがアクティブな場合、アクティブノードのフェンシングのような結果を招く場合があります。

1. ノード上にパッケージ更新をインストールする前に、次の内容を確認してください。

- その更新は、SUSE Linux Enterprise High Availability ExtensionまたはGeo Clustering for SUSE Linux Enterprise High Availability Extensionに属するパッケージに影響しますか。影響する場合は、ソフトウェアの更新を開始する前に、ノード上でクラスタスタックを停止します。

```
root # crm cluster stop
```

- パッケージ更新には再起動が必要ですか。必要な場合は、ソフトウェアの更新を開始する前に、ノード上でクラスタスタックを停止します。

```
root # crm cluster stop
```

- これらの状況のいずれにも該当しない場合は、クラスタスタックを停止する必要はありません。その場合は、ソフトウェアの更新を開始する前に、ノードを保守モードにします。

```
root # crm node maintenance NODE_NAME
```

保守モードの詳細については、[17.2項「保守タスクのためのさまざまなオプション」](#)を参照してください。

2. YaSTまたはZypperを使用してパッケージ更新をインストールします。

3. 更新が正常にインストールされたら、次のいずれかを行います。

- それぞれのノードでクラスタスタックを起動します([ステップ 1](#)で停止した場合)。


```
root # crm cluster start
```

- または、ノードの保守フラグを削除して、ノードを通常モードに戻します。

```
root # crm node ready NODE_NAME
```

4. **crm status**またはHawk2を使用してクラスタの状態を確認します。

5.4 詳細の参照先

アップグレード先の製品の変更点と新機能の詳細については、それぞれのリリースノートを参照してください。リリースノートは、<https://www.suse.com/releasesnotes/> で入手できます。

II 設定および管理

- 6 設定および管理の基本事項 58
- 7 Hawk2を使用したクラスタリソースの設定と管理 95
- 8 クラスタリソースの設定と管理(コマンドライン) 149
- 9 リソースエージェントの追加または変更 182
- 10 フェンシングとSTONITH 186
- 11 ストレージ保護とSBD 197
- 12 QDeviceとQNetd 218
- 13 アクセス制御リスト 225
- 14 ネットワークデバイスボンディング 235
- 15 負荷分散 241
- 16 Geoクラスタ(マルチサイトクラスタ) 255
- 17 保守タスクの実行 256

6 設定および管理の基本事項

HAクラスタの主な目的はユーザサービスの管理です。ユーザサービスの典型的な例は、Apache Webサーバまたはデータベースです。サービスとは、ユーザの観点からすると、指示に基づいて特別な何かを行うことを意味していますが、クラスタにとっては開始や停止できるリソースにすぎません。サービスの性質はクラスタには無関係なのです。

この章では、リソースを設定しクラスタを管理する場合に知っておく必要のある基本概念を紹介します。後続の章では、High Availability Extensionが提供する各管理ツールを使用して、主要な設定および管理タスクを行う方法を説明します。

6.1 ユースケースのシナリオ

一般的に、クラスタは次の2つのカテゴリのいずれかに分類されます。

- 2ノードクラスタ
- 2ノードより多いクラスタ。これは通常、奇数のノード数を意味します。

異なるトポロジを追加して、異なるユースケースを生成することもできます。次のユースケースは最も一般的です。

1つの場所の2ノードクラスタ

設定: FC SANまたは同様の共有ストレージ、レイヤ2ネットワーク。

使用シナリオ: サービスの高可用性、およびデータレプリケーションのデータ冗長性なしに焦点を当てた埋め込みクラスタ。このようなセットアップは無線ステーションや組立てラインコントローラなどに使用されます。

2つの場所の2ノードクラスタ(最も広く使用されている)

設定: 対称的なストレッチクラスタ、FC SAN、およびレイヤ2ネットワークのすべてが2つの場所に及ぶ。

使用シナリオ: サービスの高可用性、およびローカルデータの冗長性に焦点を当てた従来のストレッチクラスタ。データベースおよびエンタープライズリソース計画に適しており、ここ数年間で最も人気のあるセットアップの1つです。

3つの場所の奇数のノード数

設定: $2 \times N + 1$ ノード、FC SANが2つの主な場所に及ぶ。FC SANを使用しない補助的な3番目のサイト、過半数メーカーとして機能する。レイヤ2ネットワーク、少なくとも2つの主な場所に及ぶ。

使用シナリオ: サービスの高可用性、およびデータの冗長性に焦点を当てた従来のストレッチクラスタ。たとえば、データベース、エンタープライズリソースプランニング。

6.2 クォーラムの判断

1つ以上のノードとその他のクラスタ間で通信が失敗した場合は、常にクラスタパーティションが発生します。ノードは同じパーティション内の他のノードのみと通信可能で、切り離されたノードは認識しません。クラスタパーティションは、ノード(投票)の過半数を保有する場合、クォーラムを持つ(「定足数に達している」と定義されます。これを実現する方法は「クォーラム計算」によって実行されます。クォーラムはフェンシングの要件です。

クォーラム計算はSUSE Linux Enterprise High Availability Extension 11とSUSE Linux Enterprise High Availability Extension 15の間で変更されました。SUSE Linux Enterprise High Availability Extension 11では、クォーラムはPacemakerによって計算されました。SUSE Linux Enterprise High Availability Extension 12以降では、CorosyncがPacemakerの設定を変更せずに直接2ノードクラスタのクォーラムを処理できます。

クォーラムの計算方法は、次のような要因によって影響されます。

クラスタノード数

実行中のサービスを継続させるため、2ノードを超えるクラスタはクラスタパーティションの解決においてクォーラム(過半数)に依存します。次の数式に基づき、クラスタが機能するために必要な動作ノードの最少数を計算できます。

$$N \geq C/2 + 1$$

N = minimum number of operational nodes
 C = number of cluster nodes

たとえば、5ノードクラスタでは、最低3つの動作ノード(または障害が発生する可能性のある2ノード)が必要です。

2ノードクラスタまたは奇数のクラスタノードのいずれかを使用することを強くお勧めします。2ノードクラスタは、2サイト間のストレッチセットアップで重要です。奇数のノード数を持つクラスタは、1つのシングルサイトで構築するか、または3つのサイト間で分散させることができます。

Corosyncの設定

Corosyncはメッセージングおよびメンバーシップ層です。6.2.4項「2ノードクラスタのCorosync設定」および6.2.5項「NノードクラスタのCorosync設定」を参照してください。

6.2.1 グローバルクラスタオプション

グローバルクラスタオプションは、一定の状況下でのクラスタの動作を制御します。それらは、セットにグループ化され、Hawk2や`crm`シェルなどのクラスタ管理ツールで表示したり、変更することができます。

事前に定義されている値は、通常は、そのまま保持できます。ただし、クラスタの主要機能を正しく機能させるには、クラスタの基本的なセットアップ後に、次のパラメータを調整する必要があります。

- グローバルオプションno-quorum-policy
- グローバルオプションstonith-enabled

6.2.2 グローバルオプションno-quorum-policy

このグローバルオプションは、クラスタパーティションにクォーラムがない(ノードの過半数がパーティションに含まれない)場合どうするかを定義します。

許容値は、次のとおりです。

ignore

no-quorum-policyをignoreに設定するとクラスタがクォーラムを持つように動作します。リソース管理は続行されます。

SLES 11では、この値が2ノードのクラスタ用の推奨設定でした。SLES 12以降、このオプションは廃止されました。設定と条件に基づいて、Corosyncはクラスタノードまたは単一ノードに「クォーラム」を与えます。または与えません。

2ノードのクラスタの場合、クォーラムが失われた場合の唯一の意味のある動作は、常に反応することです。最初のステップとして、クォーラムを失ったノードのフェンシングを試行してください。

freeze

クォーラムが失われた場合は、クラスタパーティションがフリーズします。リソース管理は続行されます。実行中のリソースは停止されません(ただし、イベントの監視に対応して再起動される可能性があります)。ただし、影響を受けたパーティション内では、以後のリソースが開始されません。

一定のリソースが他のノードとの通信に依存しているクラスタの場合(たとえば、OCFS2マウントなど)は、この設定が推奨されます。この場合、デフォルト設定`no-quorum-policy=stop`は、次のようなシナリオになるので有効ではありません。つまり、ピアノードが到達不能な間はそれらのリソースを停止できなくなります。その代わりに、停止の試行は最終的にタイムアウトし、`stop failure`になり、エスカレートされた復元とフェンシングを引き起こします。

stop (デフォルト値)

クォーラムが失われると、影響を受けるクラスタパーティション内のすべてのリソースが整然と停止します。

suicide

クォーラムが失われると、影響を受けるクラスタパーティション内のすべてのノードがフェンシングされます。このオプションは、SBDと組み合わせる場合にのみ機能します。第11章「ストレージ保護とSBD」を参照してください。

6.2.3 グローバルオプションstonith-enabled

このグローバルオプションは、フェンシングを適用して、STONITHデバイスによる、障害ノードや停止できないリソースを持つノードのダウンを許可するかどうか定義します。通常のクラスタ操作には、STONITHデバイスの使用が必要なので、このグローバルオプションは、デフォルトでtrueに設定されています。デフォルト値では、クラスタは、STONITHリソースが定義されていない場合にはリソースの開始を拒否します。

何らかの理由でフェンシングを無効にする必要がある場合は、`stonith-enabled`をfalseに設定しますが、これはご使用の製品のサポートステータスに影響を及ぼすことに注意してください。また、`stonith-enabled=false`を指定すると、Distributed Lock Manager (DLM)のようなリソースやDLMによるすべてのサービス(lvmlockd、GFS2、OCFS2など)は開始できません。



重要: STONITHがない場合はサポートなし

STONITHがないクラスタはサポートされません。

6.2.4 2ノードクラスタのCorosync設定

ブートストラップスクリプトを使用する場合、Corosync設定には次のオプションを持つ`quorum`セクションがあります。

例 6.1: 2ノードクラスタのCOROSYNC設定の例

```
quorum {
    # Enable and configure quorum subsystem (default: off)
    # see also corosync.conf.5 and votequorum.5
    provider: corosync_votequorum
    expected_votes: 2
    two_node: 1
}
```

SUSE Linux Enterprise 11とは反対に、SUSE Linux Enterprise 12の`votequorum`サブシステムは、Corosyncバージョン2.xで機能します。つまり、`no-quorum-policy=ignore`オプションは使用してはならないことを意味します。

デフォルトで、`two_node: 1`が設定されている場合、`wait_for_all`オプションが自動的に有効になります。`wait_for_all`が有効でない場合、クラスタは両方のノードで平行に開始される必要があります。または、最初のノードが、見つからない2番目のノードで起動フェンシングを実行します。

6.2.5 NノードクラスタのCorosync設定

2ノードクラスタを使用しない場合、Nノードクラスタに奇数のノードを使用することを強くお勧めします。クォーラム設定に関して、次のオプションがあります。

- `ha-cluster-join`コマンドを使用したノードの追加、または
- Corosync設定の手動調整。

`/etc/corosync/corosync.conf`を手動で調整する場合、次の設定を使用します。

例 6.2: NノードクラスタのCOROSYNC設定の例

```
quorum {
    provider: corosync_votequorum ❶
    expected_votes: N ❷
    wait_for_all: 1 ❸
}
```

- ❶ Corosyncからのクォーラムサービスの使用
- ❷ 予想される投票数。このパラメータは`quorum`セクション内で提供されるか、または`nodelist`セクションが利用できる場合に自動的に計算されます。

- ③ wait for all (WFA)機能を有効にします。WFAが有効な場合、クラスタはすべてのノードが認識可能になった後でのみ定足数に達します。一部の起動時の競合状態を回避するために、`wait_for_all`を1に設定すると役立つ場合があります。たとえば、5ノードクラスタでは、すべてのノードに1つの投票が割り当てられているため、`expected_votes`を5に設定します。3つ以上のノードが互いに認識できるようになると、クラスタパーティションが定足数に達し、動作を開始できます。

6.3 クラスタリソース

クラスタの管理者は、クラスタ内のサーバ上の各リソースや、サーバ上で実行する各アプリケーションに対してクラスタリソースを作成する必要があります。クラスタリソースには、Webサイト、電子メールサーバ、データベース、ファイルシステム、仮想マシン、およびユーザが常時使用できるようにする他のサーバベースのアプリケーションまたはサービスなどが含まれます。

6.3.1 リソースの管理

リソースは、クラスタで使用する前にセットアップする必要があります。たとえば、Apacheサーバをクラスタリソースとして使用するには、まず、Apacheサーバをセットアップし、Apacheの環境設定を完了してから、クラスタで個々のリソースを起動します。

リソースに特定の環境要件がある場合は、それらの要件がすべてのクラスタノードに存在し、同一であることを確認してください。この種の設定は、High Availability Extensionでは管理されません。これは、管理者自身が行う必要があります。



注記: クラスタによって管理されるサービスには介入しないでください。クラスタによって管理されるサービスは操作しないでください。

High Availability Extensionでリソースを管理しているときに、同じリソースを他の方法(クラスタ外で、たとえば、手動、ブート、再起動など)で開始したり、停止してはなりません。High Availability Extensionソフトウェアが、すべてのサービスの開始または停止アクションを実行します。

サービスがクラスタ制御下ですでに実行された後にテストまたは保守タスクを実行する必要がある場合は、リソース、ノード、またはクラスタ全体を保守モードに設定してから、これらのいずれかに手動でタッチしてください。詳細については、[17.2項「保守タスクのためのさまざまなオプション」](#)を参照してください。

クラスタ内でリソースを設定したら、クラスタ管理ツールを使用して、すべてのリソースを手動で起動、停止、クリーンアップ、削除、または移行します。これらの操作の詳細については、使用しているクラスタ管理ツールに応じて次のいずれかを参照してください。

- Hawk2: 第7章 「Hawk2を使用したクラスタリソースの設定と管理」
- crmsh: 第8章 「クラスタリソースの設定と管理(コマンドライン)」

❗ 重要: リソースIDとノード名

クラスタリソースとクラスタノードは異なる名前にする必要があります。そうでない場合、Hawk2は失敗します。

6.3.2 サポートされるリソースエージェントクラス

追加するクラスタリソースごとに、リソースエージェントが準拠する基準を定義する必要があります。リソースエージェントは、提供するサービスを抽象化して正確なステータスをクラスタに渡すので、クラスタは管理するリソースについてコミットする必要がありません。クラスタは、リソースエージェントに依存して、start、stop、またはmonitorのコマンドの発行に適宜対応します。

通常、リソースエージェントはシェルスクリプトの形式で配布されます。High Availability Extensionは、次のクラスのリソースエージェントをサポートしています。

Open Cluster Framework (OCF)リソースエージェント

OCF RAエージェントは、High Availabilityでの使用に最適であり、特に、プロモータブルクローンリソースまたは特殊なモニタリング機能を必要とする場合に適しています。それらのエージェントは、通常、`/usr/lib/ocf/resource.d/provider`にあります。この機能はLSBスクリプトの機能と同様です。ただし、環境設定では、常に、パラメータの受け入れと処理を容易にする環境変数が使用されます。OCF仕様には、アクション終了コードの厳密な定義があります。9.3項「OCF戻りコードと障害回復」を参照してください。クラスタは、それらの仕様に正確に準拠します。

すべてのOCFリソースエージェントは少なくとも

もstart、stop、status、monitor、meta-dataのアクションを持つ必要があります。

meta-dataアクションは、エージェントの設定方法についての情報を取得します。


たとえば、プロバイダheartbeatでIPaddrエージェントの詳細を知るには、次のコマンドを使用します。

```
OCF_ROOT=/usr/lib/ocf /usr/lib/ocf/resource.d/heartbeat/IPaddr meta-data
```

出力は、XML形式の情報であり、いくつかのセクションを含みます(一般説明、利用可能なパラメータ、エージェント用の利用可能なアクション)。


または、`crmsh`を使用して、OCFリソースエージェントに関する情報を表示します。詳細については、[8.1.3項「OCFリソースエージェントに関する情報の表示」](#)を参照してください。

Linux Standards Base (LSB)スクリプト

LSBリソースエージェントは一般にオペレーティングシステム/配布パッケージによって提供され、`/etc/init.d`にあります。リソースエージェントをクラスターで使用するには、それらのエージェントがLSB iniスクリプトの仕様に準拠している必要があります。たとえば、リソースエージェントには、いくつかのアクションが実装されている必要があります。それらのアクションとして、少なくとも`start`、`stop`、`restart`、`reload`、`force-reload`、`status`があります。詳細については、http://refspecs.linuxbase.org/LSB_4.1.0/LSB-Core-generic/LSB-Core-generic/iniscriptact.html を参照してください。

これらのサービスの構成は標準化されていません。High AvailabilityでLSBスクリプトを使用する場合は、該当のスクリプトの設定方法を理解する必要があります。これに関する情報は、多くの場合、`/usr/share/doc/packages/PACKAGENAME`内の該当パッケージのマニュアルに記載されています。

Systemd

SUSE Linux Enterprise 12から、一般的なSystem V initデーモンがsystemdに置き代わりました。Pacemakerは、systemdサービスが存在する場合は、それを管理できます。initスクリプトの代わりに、systemdはユニットファイルを持ちます。一般的に、サービス(またはユニットファイル)は、オペレーティングシステムによって提供されます。既存のinitスクリプトを変換する場合は、<http://0pointer.de/blog/projects/systemd-for-admins-3.html> で詳細情報を検索してください。

サービス

現在、並列に存在する「通常」タイプのシステムサービスが多数あります: LSB (System V initに属する)、systemd、および(一部のディストリビューションでは) `upstart`。そのため、Pacemakerは、どれが指定のクラスターノードに適用されるのかをインテリジェントに理解する特殊なエイリアスをサポートします。これは、クラスターにsystemd、upstart、およびLSBサービスが混在する場合には特に役立ちます。Pacemakerは、次の順番で指定されたサービスを検索しようとします:。LSB (SYS-V) initスクリプト、systemdユニットファイル、またはUpstartジョブ。

Nagios

モニタリングプラグイン(かつてはNagiosプラグインと呼ばれていた)により、リモートホスト上のサービスを監視できます。Pacemakerは、モニタリングプラグインが存在する場合は、これを使用してリモートモニタリングを実行できます。詳細については、[6.6.1項「監視プラグインを使用したリモートホストでのサービスの監視」](#)を参照してください。

STONITH(フェンシング)リソースエージェント

このクラスは、フェンシング関係のリソース専用に使使されます。詳細については、[第10章「フェンシングとSTONITH」](#)を参照してください。

High Availability Extensionで提供されるエージェントは、OCF仕様に従って作成されています。

6.3.3 リソースのタイプ

次のリソースタイプを作成できます。

プリミティブ

プリミティブリソースは、リソースの中で最も基本的なタイプです。選択したクラスタ管理ツールでプリミティブリソースを作成する方法については、次を参照してください。

- Hawk2: [手順7.5「プリミティブリソースの追加」](#)
- crmsh: [8.3.2項「クラスタリソースの作成」](#)

グループ

グループには、一緒の場所で見つけ、連続して開始し、逆の順序で停止する必要があるリソースセットが含まれます。詳細については、[6.3.5.1項「グループ」](#)を参照してください。

クローン

クローンは、複数のホスト上でアクティブにできるリソースです。対応するリソースエージェントがサポートしていれば、どのようなリソースもクローン化できます。詳細については、[6.3.5.2項「クローン」](#)を参照してください。

プロモータブルクローン(以前はマスタ/スレーブまたはマルチステートリソースと呼ばれていました)は、昇格できる特別なタイプのクローンリソースです。

6.3.4 リソーステンプレート

類似した設定のリソースを多く作成する最も簡単な方法は、リソーステンプレートを定義することです。定義された後でテンプレートは、プリミティブ内で参照したり、6.5.3項「リソーステンプレートと制約」で説明するように、特定のタイプの制約内で参照することができます。

プリミティブ内でテンプレートを参照すると、そのテンプレートで定義されている操作、インスタンス属性(パラメータ)、メタ属性、使用属性がすべてプリミティブに継承されます。さらに、プリミティブに対して特定の操作または属性を定義することもできます。これらのいずれかがテンプレートとプリミティブの両方で定義されていた場合、プリミティブで定義した値の方が、テンプレートで定義された値よりも優先されます。

選択したクラスタ管理ツールでリソーステンプレートを定義する方法については、次を参照してください。

- Hawk2: 手順7.6「リソーステンプレートの追加」
- crmsh: 8.3.3項「リソーステンプレートの作成」

6.3.5 高度なリソースタイプ

プリミティブは、最も単純なタイプのリソースなので、設定が容易ですが、クラスタ設定には、より高度なリソースタイプ(グループ、クローン、プロモータブルクローンリソースなど)が必要になることがあります。

6.3.5.1 グループ

クラスタリソースの中には、他のコンポーネントやリソースに依存しているものもあります。それぞれのコンポーネントやリソースが決められた順序で開始され、依存しているリソースと同じサーバ上で同時に実行していなければならない場合があります。この設定を簡素化するには、クラスタリソースグループを使用できます。

例 6.3: WEBサーバのリソースグループ

リソースグループの1例として、IPアドレスとファイルシステムを必要とするWebサーバがあります。この場合、各コンポーネントは、個々のリソースであり、それらが組み合わされてクラスタリソースグループを構成します。リソースグループは、1つ以上のサーバで実行されます。ソフトウェアまたはハードウェアが機能しない場合には、個々のクラスタリソースと同様に、グループはクラスタ内の別のサーバにフェールオーバーします。

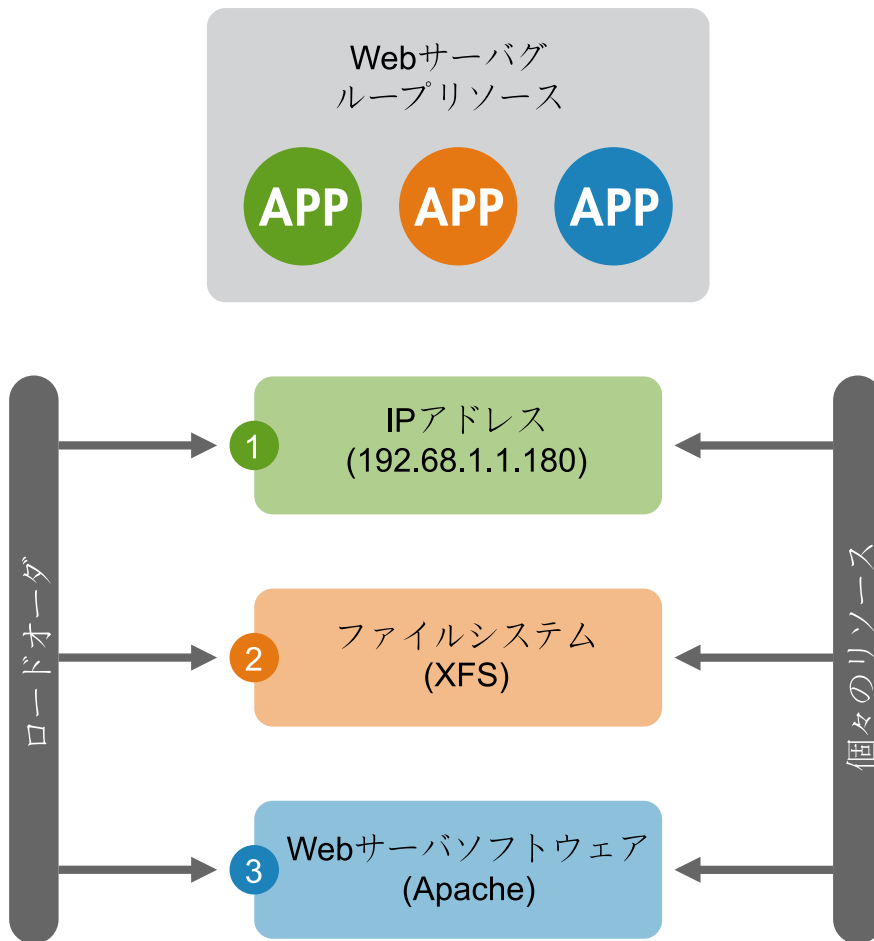


図 6.1: グループリソース

グループには次のプロパティがあります。

開始と停止

リソースは認識される順序で開始し、逆の順番で停止します。

依存関係

グループ内のリソースがどこかで開始できない場合は、グループ内のその後の全リソースは実行することができません。

コンテンツ

グループにはプリミティブクラスタリソースしか含むことができません。グループには1つ以上のリソースを含む必要があります。空の場合は設定は無効になります。グループリソースの子を参照するには、グループのIDではなく子のIDを使用します。

制約

制約でグループの子を参照することはできますが、通常はグループ名を使用することをお勧めします。

固着性

固着性はグループ内で統合可能なプロパティです。グループ内の「アクティブな」各メンバーは、グループの合計値に対して固着性を追加します。したがって、デフォルトの `resource-stickiness` が 100 で、グループに 7 つのメンバーがあり、そのうち 5 つがアクティブな場合は、グループが全体として、スコア 500 で、現在の場所を優先します。

リソース監視

グループのリソース監視を有効にするには、グループ内で監視の必要な各リソースに対して監視を設定する必要があります。

選択したクラスタ管理ツールでグループを作成する方法については、次を参照してください。

- Hawk2: [手順7.9「リソースグループの追加」](#)
- crmsh: [8.3.10項「クラスタリソースグループの設定」](#)

6.3.5.2 クローン

クラスタ内の複数のノードで特定のリソースを同時に実行することができます。このためには、リソースをクローンとして設定する必要があります。クローンとして設定するリソースの一例として、OCFS2などのクラスタファイルシステムが挙げられます。提供されているどのリソースも、クローンとして設定できます。これは、リソースのリソースエージェントによってサポートされます。クローンリソースは、ホスティングされているノードによって異なる設定をすることもできます。

リソースクローンには次の3つのタイプがあります。

匿名クローン

最も簡単なクローンタイプです。実行場所にかかわらず、同じ動作をします。このため、マシンごとにアクティブな匿名クローンのインスタンスは1つだけ存在できます。

グローバルに固有なクローン

このリソースは独自のエントリです。1つのノードで実行しているクローンのインスタンスは、別なノードの別なインスタンスとは異なり、同じノードの2つのインスタンスが同一になることもありません。

プロモータブルクローン(マルチステートリソース)

このリソースのアクティブインスタンスは、アクティブとパッシブという2つの状態に分けられます。プライマリとセカンダリ、またはマスタとスレーブと呼ばれることもあります。プロモータブルクローンが、匿名またはグローバルに固有の場合もあります。6.3.5.3項「プロモータブルクローン(マルチステートリソース)」も参照してください。

クローンは、グループまたは通常リソースを1つだけ含む必要があります。

リソースのモニタリングまたは制約を設定する場合、クローンには、単純なリソースとは異なる要件があります。詳細については、『Pacemaker Explained』(<http://www.clusterlabs.org/pacemaker/doc/>から入手可)を参照してください。特に、「Clones - Resources That Get Active on Multiple Hosts」のセクションを参照してください。

選択したクラスタ管理ツールでクローンを作成する方法については、次を参照してください。

- Hawk2: 手順7.10「クローンリソースの追加」
- crmsh: 8.3.11項「クローンリソースの設定」。

6.3.5.3 プロモータブルクローン(マルチステートリソース)

プロモータブルクローン(以前はマルチステートリソースと呼ばれていました)は、クローンが得意とするところです。これにより、インスタンスを2つの動作モード(masterまたはslaveと呼ばれているが、任意の名前を割り当てることができる)のいずれかに設定できます。プロモータブルクローンは、グループまたは通常リソースを1つだけ含む必要があります。

リソースのモニタリングまたは制約を設定する場合、プロモータブルクローンには、単純なリソースとは異なる要件があります。詳細については、『Pacemaker Explained』を参照してください。Pacemaker 1.1のバージョンは<http://www.clusterlabs.org/pacemaker/doc/>から入手できます。特に、「Multi-state - Resources That Have Multiple Modes」のセクションを参照してください。

6.3.6 リソースオプション(メタ属性)

追加した各リソースについて、オプションを定義できます。クラスタはオプションを使用して、リソースの動作方法を決定します。CRMに特定のリソースの処理方法を通知します。リソースオプションは、`crm_resource --meta`コマンドまたはHawk2を使用して設定できます(手順7.5「プリミティブリソースの追加」を参照)。

表 6.1: プリミティブリソースのオプション

オプション	説明	デフォルト
<u>優先度</u>	一部のリソースをアクティブにできない場合、クラスタは優先度の低いリソースを停止して、優先度の高いリソースをアクティブに維持します。	<u>0</u>
<u>target-role</u>	クラスタが維持しようとするこのリソースの状態。使用できる値: <u>stopped</u> 、 <u>started</u> 、 <u>master</u>	<u>開始日</u>
<u>is-managed</u>	クラスタがリソースを開始して停止できるかどうか。使用できる値: <u>true</u> 、 <u>false</u> 値が <u>false</u> に設定されていても、リソースの状態は引き続き監視され、障害が発生した場合は報告されます。これは、リソースを <u>maintenance="true"</u> に設定するのとは異なります。	<u>true</u>
<u>保守モード</u>	リソースは手動でタッチできるかどうか。使用できる値: <u>true</u> 、 <u>false</u> <u>true</u> に設定すると、すべてのリソースが非管理対象になり、クラスタによる監視が停止されるため、ステータスは追跡されなくなります。クラスタによってクラスタリソースの再起動が試行される代わりに、ユーザがクラスタリソースを停止または再起動できます。	<u>false</u>

オプション	説明	デフォルト
<u>resource-stickiness</u>	リソースが現在の状態をどの程度維持したいか。	計算済み
<u>migration-threshold</u>	ノードがこのリソースをホストできなくなるまで、このリソースについてノード上で発生する失敗の回数。	<u>INFINITY</u> (無効)
<u>multiple-active</u>	複数のノードでアクティブなリソースを検出した場合のクラスタの動作。使用できる値: <u>block</u> (リソースを管理されていないとマークする)、 <u>stop_only</u> 、 <u>stop_start</u>	<u>stop_start</u>
<u>failure-timeout</u>	失敗が発生していないように動作する(リソースを失敗したノードに戻す)前に、待機する秒数	<u>0</u> (無効)
<u>allow-migrate</u>	<u>migrate_to</u> または <u>migrate_from</u> のアクションをサポートするリソースにリソース移行を許可。	<u>false</u>
<u>remote-node</u>	このリソースが定義するリモートノードの名前。これにより、リモートノードのリソースが有効化されるだけでなく、リモートノードの識別に使用される固有の名前が定義されます。また、他のパラメータが設定されていない場合、この値は <u>remote-port</u> ポートで接続するホスト名と想定されます。	なし(無効)

オプション	説明	デフォルト
	 警告: 固有のIDの使用 この値は、既存のリソースやノードIDとは重複させないでください。	
<code>remote-port</code>	pacemaker_remoteへのゲスト接続用のカスタムポート。	<code>3121</code>
<code>remote-addr</code>	リモートノードの名前がゲストのホスト名ではない場合に接続するIPアドレスまたはホスト名。	<code>remote-node</code> (ホスト名として使用される値)
<code>remote-connect-timeout</code>	中断したゲスト接続がタイムアウトするまでの時間。	<code>60s</code>

6.3.7 インスタンス属性(パラメータ)

すべてのリソースクラスのスクリプトでは、動作方法および管理するサービスのインスタンスを指定するパラメータを指定できます。リソースエージェントがパラメータをサポートする場合、それらのパラメータを`crm_resource`コマンドまたはHawk2を使用して追加できます(手順7.5「プリミティブリソースの追加」を参照)。インスタンス属性は、`crm`コマンドラインユーティリティでは`params`、Hawk2では`Parameter`と呼ばれます。OCFスクリプトでサポートされているインスタンス属性のリストは、次のコマンドを`root`として実行すると参照できます。

```
root # crm ra info [class:[provider:]]resource_agent
```

または(オプション部分なし):

```
root # crm ra info resource_agent
```

出力には、サポートされているすべての属性、それらの目的、およびデフォルト値が一覧されます。

たとえば、次のコマンドを使用します。

```
root # crm ra info IPAddr
```

次の出力が返されます。

```
Manages virtual IPv4 addresses (portable version) (ocf:heartbeat:IPAddr)

This script manages IP alias IP addresses
It can add an IP alias, or remove one.

Parameters (* denotes required, [] the default):

ip* (string): IPv4 address
The IPv4 address to be configured in dotted quad notation, for example
"192.168.1.1".

nic (string, [eth0]): Network interface
The base network interface on which the IP address will be brought
online.

If left empty, the script will try and determine this from the
routing table.

Do NOT specify an alias interface in the form eth0:1 or anything here;
rather, specify the base interface only.

cidr_netmask (string): Netmask
The netmask for the interface in CIDR format. (ie, 24), or in
dotted quad notation 255.255.255.0).

If unspecified, the script will also try to determine this from the
routing table.

broadcast (string): Broadcast address
Broadcast address associated with the IP. If left empty, the script will
determine this from the netmask.

iflabel (string): Interface label
You can specify an additional label for your IP address here.

lvs_support (boolean, [false]): Enable support for LVS DR
Enable support for LVS Direct Routing configurations. In case a IP
address is stopped, only move it to the loopback device to allow the
local node to continue to service requests, but no longer advertise it
on the network.

local_stop_script (string):
Script called when the IP is released

local_start_script (string):
Script called when the IP is added
```

```
ARP_INTERVAL_MS (integer, [500]): milliseconds between gratuitous ARPs  
milliseconds between ARPs
```

```
ARP_REPEAT (integer, [10]): repeat count  
How many gratuitous ARPs to send out when bringing up a new address
```

```
ARP_BACKGROUND (boolean, [yes]): run in background  
run in background (no longer any reason to do this)
```

```
ARP_NETMASK (string, [ffffffffffff]): netmask for ARP  
netmask for ARP - in nonstandard hexadecimal format.
```

```
Operations' defaults (advisory minimum):
```

```
start          timeout=90  
stop           timeout=100  
monitor_0      interval=5s timeout=20s
```



注記: グループ、クローン、またはプロモータブルクローンのインスタンス属性

グループ、クローン、およびプロモータブルクローンリソースには、インスタンス属性がないので注意してください。ただし、インスタンス属性のセットは、グループ、クローン、またはプロモータブルクローンリソースの子によって継承されます。

6.3.8 リソース操作

デフォルトで、クラスタはリソースが良好な状態であることを保証しません。クラスタにこれを行わせるには、リソースの定義に監視操作を追加する必要があります。監視操作は、すべてのクラスまたはリソースエージェントに追加できます。詳細については、[6.4項「リソース監視」](#)を参照してください。

表 6.2: リソース操作のプロパティ

操作	説明
<u>id</u>	アクションに指定する名前。一意にする必要があります。(IDは表示されません)
<u>name</u>	実行するアクション。共通の値: <u>monitor</u> 、 <u>start</u> 、 <u>stop</u>
<u>interval</u>	操作を実行する頻度。単位: 秒

操作	説明
<u>timeout</u>	アクションが失敗したと宣言する前に待機する長さ。
<u>requires</u>	このアクションが発生する前に満たす必要のある条件。使用できる値: <u>nothing</u> 、 <u>quorum</u> 、 <u>fencing</u> デフォルトは、フェンシングが有効でリソースのクラスが <u>stonith</u> かどうかによります。STONITHリソースの場合、デフォルトは <u>nothing</u> です。
<u>on-fail</u>	このアクションが失敗した場合に実行するアクション。使用できる値: <ul style="list-style-type: none"> • <u>ignore</u>: リソースが失敗しなかったのよう動作します。 • <u>block</u>: リソースにこれ以上の操作を実行しません。 • <u>stop</u>: リソースを停止して、他の場所でも開始しません。 • <u>restart</u>: リソースを停止して再起動します(別のノード上で)。 • <u>fence</u>: リソースが失敗したノードを停止します(STONITH)。 • <u>standby</u>: リソースが失敗したノードから「すべて」のリソースを移動させます。
<u>enabled</u>	<u>false</u> の場合、操作は存在していない場合と同様に処理されます。使用できる値: <u>true</u> 、 <u>false</u>
<u>role</u>	リソースにこの役割がある場合のみ操作を実行します。

操作	説明
<code>record-pending</code>	グローバルに設定したり、個々のリソースに対して設定できます。リソース上の「in-flight」操作の状態をCIBに反映させます。
<code>description</code>	操作について説明します。

6.3.9 タイムアウト値

リソースのタイムアウト値は次の3つのパラメータの影響を受けることがあります。

- `op_defaults` (操作用のグローバルタイムアウト)
- リソーステンプレートに対して定義された特定のタイムアウト値
- リソースに対して定義された特定のタイムアウト値



注記: 値の優先度

リソースに対して「特定の」値が定義される場合、グローバルデフォルトより優先されます。また、リソースに対して定義された特定の値は、リソーステンプレートで定義された値より優先されます。

タイムアウト値を適切に設定することは非常に重要です。これらの値を短くしすぎると、次のような理由で、多数の(不必要な)フェンシング処理が発生します。

1. リソースでタイムアウトが発生すると、リソースは失敗し、クラスタはリソースを停止しようとします。
2. リソースの停止も失敗した場合(たとえば、停止用タイムアウトの設定が低すぎるため)、クラスタはノードをフェンシングします。これが制御不能になるノードを考慮します。

操作に対するグローバルデフォルトを調整し、`crmsh`およびHawk2の両方で特定のタイムアウト値を設定できます。タイムアウト値の決定および設定のベストプラクティスは次のとおりです。

手順 6.1: タイムアウト値の決定

1. 負荷の下でリソースが開始および停止するためにかかる時間を確認します。

2. 必要に応じて、`op_defaults` パラメータを追加して、それに応じて(デフォルトの)タイムアウト値を設定します。

- a. たとえば、`op_defaults` を 60 秒に設定します。

```
crm(live)configure# op_defaults timeout=60
```

- b. さらに長い時間を必要とするリソースについては、個別の値を定義します。

3. あるリソースに対して操作を設定する場合には、個別の `start` および `stop` 操作を追加します。Hawk2 を使用して設定する場合、これらの操作に適したタイムアウト値候補が表示されます。

6.4 リソース監視

リソースが実行中であるかどうか確認するには、そのリソースにリソースの監視を設定しておく必要があります。

リソースモニタが障害を検出すると、次の処理が行われます。

- `/etc/corosync/corosync.conf` の `logging` セクションで指定された設定に従って、ログファイルメッセージが生成されます。
- 障害がクラスタ管理ツール(Hawk2、`crm status`)と、CIB ステータスセクションに反映されます。
- クラスタが明瞭な復旧アクションを開始します。これらのアクションには、リソースを停止して障害状態を修復する、ローカルまたは別のノードでリソースを再起動するなどが含まれる場合があります。設定やクラスタの状態によっては、リソースが再起動されないこともあります。

リソースの監視を設定しなかった場合、開始成功後のリソース障害は通知されず、クラスタは常にリソース状態を良好として表示してしまいます。

停止されたリソースの監視

通常、リソースは動作している限り、クラスタのみによって監視されます。しかし、同時実行違反を検出するために、停止されるリソースの監視も設定する必要があります。

例:

```
primitive dummy1 ocf:heartbeat:Dummy \  
    op monitor interval="300s" role="Stopped" timeout="10s" \  
    op monitor interval="30s" timeout="10s"
```

この設定は、300秒ごとに、リソース dummy1 に対する監視操作をトリガします。これは、リソースが role="Stopped" に入ると有効になります。実行中には、リソースは 30秒 ごとに監視されます。

プローブ

CRMはすべてのノードの各リソースに対して、probeと呼ばれる初期監視を実行します。probeはリソースのクリーンアップ後にも実行されます。1つのリソースに対して複数の監視操作が定義されている場合、CRMは最も時間間隔の短い監視を1つ選択し、そのタイムアウト値をプローブのデフォルトタイムアウトとして使用します。監視操作が何も設定されていない場合は、クラスタ規模のデフォルトが適用されます。デフォルトは、20秒です(別途op_defaultsパラメータを設定して指定されていない場合)。自動計算やop_defaultsの値に依存したくない場合は、このリソースの「プローブ」に対して特定の監視操作を定義します。intervalを0に設定した監視操作を追加することで、この操作を行います。たとえば次のようになります。

```
crm(live)configure# primitive rsc1 ocf:pacemaker:Dummy \  
    op monitor interval="0" timeout="60"
```

rsc1のプローブは60sでタイムアウトになります。この値は、op_defaultsで定義されているグローバルなタイムアウト値や、その他の操作で設定されたタイムアウト値とは無関係です。それぞれのリソースのプローブを指定するためにinterval="0"を設定していない場合、CRMは、そのリソースに定義されている監視操作がほかにはないかどうかを自動的に確認し、上で説明されているようにプローブのタイムアウト値を計算します。

選択したクラスタ管理ツールでリソースに対して監視操作を追加する方法については、次を参照してください。

- Hawk2: [手順7.13「操作の追加または変更」](#)
- crmsh: [8.3.9項「リソース監視の設定」](#)

6.5 リソースの制約

すべてのリソースを設定する以外にも、多くの作業が必要です。クラスタが必要なすべてのリソースを認識しても、正しく処理できるとは限りません。リソースの制約を指定して、リソースを実行可能なクラスタノード、リソースのロード順序、特定のリソースが依存している他のリソースを指定することができます。

6.5.1 制約のタイプ

使用可能な制約には3種類あります。

リソースの場所

場所の制約はリソースを実行できるノード、できないノード、または実行に適したノードを定義するものです。

リソースのコロケーション

コロケーション制約は、ノード上で一緒に実行可能な、または一緒に実行することが禁止されているリソースをクラスタに伝えます。

リソースの順序

アクションの順序を定義する、順序の制約です。

！ 重要: 制約および特定のタイプのリソースに関する制限

- リソースグループの「メンバー」に対してコロケーション制約を作成しないでください。代わりに、リソースグループ全体を指すリソース制約を作成してください。その他のタイプの制約はすべて、リソースグループのメンバーに対して使用しても問題ありません。
- クローンリソースまたはプロモータブルクローンリソースが適用されているリソースで制約を使用しないでください。制約はクローンまたはプロモータブルクローンリソースに適用する必要があり、その子リソースに適用することはできません。

6.5.1.1 リソースセット

6.5.1.1.1 制約を定義するためにリソースセットを使用する

場所、コロケーション、または順序の制約を定義するための別のフォーマットとして、`resource sets`を使用することができます。リソースセットでは、プリミティブが1つのセットでグループ化されます。以前は、これはリソースグループを定義するか(デザインを正確に表現できない場合もあった)、個々の制約として各関係を定義することでこの操作が可能でした。個々の制約として定義した場合、多数のリソースとの組み合わせが増えるにつれて、制約が飛躍的に増加しました。リソースセットを介した設定で、冗長性が常に低減されるわけではありませんが、次の例が示すように、定義内容の把握と管理がより容易になります。

例 6.4: 場所制約のリソースセット

たとえば、crmshでリソースセット(loc-alice)の次の設定を使用して、2つの仮想IP (vip1とvip2)を同じノードaliceに配置できます。

```
crm(live)configure# primitive vip1 ocf:heartbeat:IPaddr2 params ip=192.168.1.5
crm(live)configure# primitive vip2 ocf:heartbeat:IPaddr2 params ip=192.168.1.6
crm(live)configure# location loc-alice { vip1 vip2 } inf: alice
```

リソースセットを使用してコロケーション制約の設定を置き換える場合は、次の2つの例を検討します。

例 6.5: コロケートされたリソースのチェーン

```
<constraints>
  <rsc_colocation id="coloc-1" rsc="B" with-rsc="A" score="INFINITY"/>
  <rsc_colocation id="coloc-2" rsc="C" with-rsc="B" score="INFINITY"/>
  <rsc_colocation id="coloc-3" rsc="D" with-rsc="C" score="INFINITY"/>
</constraints>
```

リソースセットで表される同一の設定:

```
<constraints>
  <rsc_colocation id="coloc-1" score="INFINITY" >
    <resource_set id="colocated-set-example" sequential="true">
      <resource_ref id="A"/>
      <resource_ref id="B"/>
      <resource_ref id="C"/>
      <resource_ref id="D"/>
    </resource_set>
  </rsc_colocation>
</constraints>
```

リソースセットを使用して順序の制約の設定を置き換える場合は、次の2つの例を検討します。

例 6.6: 順序付けされたリソースのチェーン

```
<constraints>
  <rsc_order id="order-1" first="A" then="B" />
  <rsc_order id="order-2" first="B" then="C" />
  <rsc_order id="order-3" first="C" then="D" />
</constraints>
```

順序付けされたリソースを持つリソースセットを使用して、同様な目的を達成できます。

例 6.7: リソースセットとして表される順序付けされたリソースのチェーン

```
<constraints>
  <rsc_order id="order-1">
    <resource_set id="ordered-set-example" sequential="true">
      <resource_ref id="A"/>
      <resource_ref id="B"/>
      <resource_ref id="C"/>
      <resource_ref id="D"/>
    </resource_set>
  </rsc_order>
</constraints>
```

これらのセットは、順序付けされている(`sequential=true`)場合もあれば、順序付けされていない場合(`sequential=false`)場合もあります。また、`require-all`属性を使用して、`AND`および`OR`ロジック間を切り替えることができます。

6.5.1.1.2 依存関係のないコロケーション制約のリソースセット

同じノード上にリソースのグループを配置する方が役立つ場合があります(コロケーション制約を定義)、リソース間に困難な依存関係を持つことはありません。たとえば、同じノード上に2つのリソースを配置したいが、それらの一方で障害が発生した場合に他方をクラスタで再起動したくない場合があります。これは、`weak bond`コマンドを使用して、`crm`シェルで実行できます。

選択したクラスタ管理ツールでこれらの「弱い結合」を設定する方法については、次を参照してください。

- `crmsh`: 8.3.5.3項「依存性なしのリソースセットのコロケーション」

6.5.1.2 詳細の参照先

様々な種類の制約を追加する方法については、選択したクラスタ管理ツールに応じて次のいずれかを参照してください。

- Hawk2: 7.6項「制約の設定」
- `crmsh`: 8.3.5項「リソース制約の設定」

制約の設定の詳細や、順序付けおよびコロケーションの基本的な概念についての詳しいバックグラウンド情報は次のドキュメントを参照してください。これらのドキュメントは、<http://www.clusterlabs.org/pacemaker/doc/>で入手できます。

- 『Pacemaker Explained』の「Resource Constraints」の章
- 『Colocation Explained』
- 『オーダーの概要』

6.5.2 スコアと無限大

制約を定義する際は、スコアも扱う必要があります。あらゆる種類のスコアはクラスタの動作方法と密接に関連しています。スコアの操作によって、リソースのマイグレーションから、速度が低下したクラスタで停止するリソースの決定まで、あらゆる作業を実行できます。スコアはリソースごとに計算され、リソースに対して負のスコアが付けられているノードは、そのリソースを実行できません。リソースのスコアを計算した後、クラスタはスコアが最も高いノードを選択します。

INFINITYは現在1,000,000と定義されています。この値の増減は、次の3つの基本ルールに従います。

- 任意の値+ INFINITY = INFINITY
- 任意の値- INFINITY = -INFINITY
- INFINITY - INFINITY = -INFINITY

リソース制約を定義する際は、各制約のスコアを指定します。スコアはこのリソース制約に割り当てる値を示します。スコアの低い制約は、それよりもスコアが高い制約より先に適用されます。1つのリソースに対して場所の制約を複数作成し、それぞれに異なるスコアを指定することで、リソースがフェールオーバーするノードの順序を指定できます。

6.5.3 リソーステンプレートと制約

リソーステンプレートを定義したら(6.3.4項「リソーステンプレート」を参照)、次のタイプの制約で参照できます。

- 順序の制約
- コロケーション制約
- rsc_ticket制約(Geoクラスタの場合)

ただし、コロケーション制約には、テンプレートへの参照を複数含めることはできません。リソースセットには、テンプレートへの参照を含めることはできません。

制約内で参照されたリソーステンプレートは、そのテンプレートから派生するすべてのプリミティブを表します。これは、そのリソーステンプレートを参照しているすべてのプリミティブリソースに、この制約が適用されることを意味します。制約内でリソーステンプレートを参照すれば、リソースセットの代替となり、クラスタ設定をかなりの程度単純化することができます。リソースセットの詳細については、[手順7.17「制約のためにリソースセットを使用する」](#)を参照してください。

6.5.4 フェールオーバーノード

リソースに障害が発生すると、自動的に再起動されます。現在のノードで再起動できない場合、または現在のノードでN回失敗した場合は、別のノードへのフェールオーバーが試行されます。リソースが失敗するたびに、その失敗回数が増加します。新しいノードへのマイグレーションを行う基準(`migration-threshold`)となるリソースの失敗をいくつか定義できます。クラスタ内に3つ以上ノードがある場合、特定のリソースのフェールオーバー先のノードはHigh Availabilityソフトウェアが選択します。

ただし、リソースに1つ以上の場所の制約と`migration-threshold`を設定することで、そのリソースのフェールオーバー先にするノードを指定できます。

選択したクラスタ管理ツールでフェールオーバーノードを指定する方法については、次を参照してください。

- Hawk2: [7.6.6項「リソースフェールオーバーノードの指定」](#)
- crmsh: [8.3.6項「リソースフェールオーバーノードの指定」](#)

例 6.8: マイグレーションしきい値 - プロセスフロー

たとえば、リソース「`rsc1`」に場所の制約を設定し、このリソースを「`alice`」で優先的に実行するように指定したと仮定します。そのノードで実行できなかった場合は、「`migration-threshold`」を確認して失敗回数と比較します。失敗回数 \geq マイグレーションしきい値の場合は、リソースは次の優先実行先として指定されているノードにマイグレートされます。

デフォルトでは、いったんしきい値に達すると、そのノードでは、リソースの失敗回数がリセットされるまで、失敗したリソースを実行できなくなります。これは、手動でクラスタ管理者が行うか、リソースに`failure-timeout`オプションを設定することで実行できます。

たとえば、`migration-threshold=2`と`failure-timeout=60s`を設定すると、リソースは、2回の失敗の後に新しいノードに移行します。そして、1分後に復帰できます(固着性と制約のスコアによる)。

移行しきい値の概念には2つの例外があり、これらの例外は、リソースの開始失敗か、停止失敗のどちらかで発生します。

- 起動の失敗では、失敗回数が`INFINITY`に設定されるので、常に、即時に移行が行われます。
- 停止時の失敗ではフェンシングが発生します(`[stonith-enabled]` がデフォルトである「`true`」に設定されている場合)。
STONITHリソースが定義されていない場合は(または`stonith-enabled`が`false`に設定されている場合)、リソースの移行は行われません。

選択したクラスタ管理ツールでマイグレーションしきい値を使用し、失敗回数をリセットする方法については、次を参照してください。

- Hawk2: 7.6.6項「リソースフェールオーバーノードの指定」
- crmsh: 8.3.6項「リソースフェールオーバーノードの指定」

6.5.5 フェールバックノード

ノードがオンライン状態に戻り、クラスタ内にある場合は、リソースが元のノードにフェールバックすることがあります。リソースを実行していたノードにリソースをフェールバックさせたくない場合や、リソースのフェールバック先として別のノードを指定する場合は、リソースの固着性の値を変更します。リソースの固着性は、リソースの作成時でも、その後も指定できます。

リソース固着性値の指定時には、次の予想される結果について考慮してください。

0の値:

デフォルトです。リソースはシステム内で最適な場所に配置されます。現在よりも「状態のよい」、または負荷の少ないノードが使用可能になると、移動することを意味しています。このオプションは自動フェールバックとほとんど同じですが、以前アクティブだったノード以外でもリソースをフェールバックできるという点が異なります。

0より大きい値:

リソースは現在の場所に留まることを望んでいます。状態がよいノードが使用可能になると移動される可能性があります。値が大きくなるほど、リソースが現在の場所に留まることを強く望んでいることを示します。

0より小さい値:

リソースは現在の場所から別な場所に移動することを望んでいます。絶対値が大きくなるほど、リソースが移動を強く望んでいることを示します。

INFINITYの値:

ノードがリソースの実行権利がなくなったために強制終了される場合(ノードのシャットダウン、ノードのスタンバイ、migration-thresholdに到達、または設定変更)以外は、リソースは常に現在の場所に留まります。このオプションは自動フェールバックを完全に無効にする場合とほとんど同じです。

-INFINITYの値:

リソースは現在の場所から常に移動されます。

6.5.6 負荷インパクトに基づくリソースの配置

すべてのリソースが同等ではありません。Xenゲストなどの一部のリソースでは、そのホストであるノードがリソースの容量要件を満たす必要があります。リソースの組み合わされたニーズが提供された容量より大きくなるようにリソースが配置されると、リソースのパフォーマンスが低下します(あるいは失敗することさえあります)。

これを考慮に入れて、High Availability Extensionでは、次のパラメータを指定できます。

1. 一定のノードが「提供する」容量
2. 一定のリソースが「要求する」容量
3. リソースの配置に関する全体的なストラテジ

選択したクラスタ管理ツールでこれらの設定を設定する方法については、次を参照してください。

- Hawk2: 7.6.8項 「負荷インパクトに基づくリソース配置の設定」
- crmsh: 8.3.8項 「負荷インパクトに基づくリソース配置の設定」

ノードは、リソースの要件を満たすだけの空き容量があれば、そのリソースに対して資格があるとみなされます。High Availability Extensionにとって、容量の性質は重要ではありません。High Availability Extensionは、リソースをノードに移動する前に、リソースのすべての容量要件が満たされているかどうかを確認するだけです。

リソースの要件とノードが提供する容量を手動で設定するには、使用属性を使用します。使用属性に任意の名前を付け、設定に必要なだけ名前/値のペアを定義します。ただし、属性値は、整数にする必要があります。

使用属性を持つ複数のリソースがグループ化されていたり、これらにコロケーション制約がある場合、High Availability Extensionではそのことを考慮に入れます。可能な場合、これらのリソースは、「すべての」容量要件を満たすことができるノードに配置されます。



注記: グループの使用属性

リソースグループに対して使用属性を直接設定することはできません。ただし、グループの設定を簡素化するために、グループ内のすべてのリソースに必要な合計容量を含む使用属性を追加することができます。

High Availability Extensionには、ノードの容量とリソースの要件を自動的に検出し、設定する手段も用意されています。

NodeUtilizationリソースエージェントは、ノードの容量をチェックします (CPUとRAMについて)。自動検出を設定するには、クラス、プロバイダ、タイプが`ocf:pacemaker:NodeUtilization`のクローンリソースを作成します。このクローンのインスタンスが各ノードに1つずつ実行している必要があります。インスタンスが開始すると、CIBでそのノードの設定に`utilization`セクションが追加されます。

リソースの最小要件の自動検出(RAMとCPU)に配慮し、Xenリソースエージェントが改良されました。Xenリソースは、開始時点でRAMとCPUの消費状況を反映します。リソース設定には、使用属性が自動的に追加されます。



注記: Xenとlibvirtに異なるリソースエージェントを適用

`ocf:heartbeat:Xen`リソースエージェントは、libvirtに使用するべきではありません。libvirtではマシン記述ファイルの変更が想定されているためです。

libvirtには、`ocf:heartbeat:VirtualDomain`リソースエージェントを使用します。

最小要件を検出することに加え、High Availability Extensionは、VirtualDomainリソースエージェントを通して現在の利用状況を監視することができ、仮想マシンでのCPUとRAMの使用状況を検出します。この機能を使用するには、クラス、プロバイダ、およびタイプが`ocf:heartbeat:VirtualDomain`のリソースを設定します。次のインスタンス属性を使用できます。`autoset_utilization_cpu`と`autoset_utilization_hv_memory`。両方ともデフォルトは`true`です。これにより、監視サイクルのたびにCIBで使用値が更新されます。

容量と要件を手動と自動のどちらで設定する場合でも、`placement-strategy`プロパティ(グローバルクラスタオプション内)で、配置ストラテジを指定する必要があります。次の値を使用できます。

default (デフォルト値)

使用値は考慮しません。リソースは、場所のスコアに従って割り当てられます。スコアが同じであれば、リソースはノード間で均等に分散されます。

utilization

リソースの要件を満たすだけの空き容量がノードにあるかどうか決定する際に、利用率を確認します。ただし、負荷分散は、まだ、ノードに割り当てられたリソースの数に基づいて行われます。

minimal

リソースの要件を満たすだけの空き容量がノードにあるかどうか決定する際に、利用率を確認します。できるだけ少ない数のノードにリソースを集中しようとします(残りのノードの電力節約のため)。

balanced

リソースの要件を満たすだけの空き容量がノードにあるかどうか決定する際に、利用率を確認します。リソースを均等に分散して、リソースのパフォーマンスを最適化しようとします。



注記: リソース優先度の設定

使用できる配置ストラテジは、最善策であり、まだ、複雑なヒューリスティックソルバで、常に最適な割り当て結果を得るには至っていません。リソースの優先度を正しく設定して、最重要なリソースが最初にスケジュールされるようにしてください。

例 6.9: 負荷分散型配置の設定例

次の例は、同等のノードから成る3ノードクラスと4つの仮想マシンを示しています。

```
node alice utilization memory="4000"
node bob utilization memory="4000"
node charlie utilization memory="4000"
primitive xenA ocf:heartbeat:Xen utilization hv_memory="3500" \
  params xmfile="/etc/xen/shared-vm/vm1"
  meta priority="10"
primitive xenB ocf:heartbeat:Xen utilization hv_memory="2000" \
  params xmfile="/etc/xen/shared-vm/vm2"
  meta priority="1"
primitive xenC ocf:heartbeat:Xen utilization hv_memory="2000" \
  params xmfile="/etc/xen/shared-vm/vm3"
  meta priority="1"
primitive xenD ocf:heartbeat:Xen utilization hv_memory="1000" \
  params xmfile="/etc/xen/shared-vm/vm4"
  meta priority="5"
property placement-strategy="minimal"
```

3ノードはすべてアクティブであり、まず、リソース xenA がノードに配置され、次に、xenD が配置されます。xenB と xenC は、一緒に割り当てられるか、またはどちらか1つが xenD とともに割り当てられます。

1つのノードに障害が発生した場合、残りのノード上で利用できるメモリ合計が少なすぎて、これらのリソースすべてはホストできません。xenAは確実に割り当てられ、xenDも同様です。ただし、残りのリソースxenBとxenCは、そのどちらかしか割り当てられません。xenBとxenCの優先度は同等なので、結果はまだ決められません。これを解決するためにも、どちらかに高い優先度を設定する必要があります。

6.5.7 タグの使用によるリソースのグループ化

タグは最近Pacemakerに追加された新機能です。タグは、コロケーションの作成や関係の順序付けを行わずに、複数のリソースをただちに参照する方法です。これは、概念的に関連するリソースをグループ化するのに役立つ場合があります。たとえば、データベースに関連するいくつかのリソースがある場合、databasesというタグを作成し、データベースに関連するすべてのリソースをこのタグに追加します。これにより、1つのコマンドでそれらすべてのリソースを停止または起動できます。

タグは制約でも使用できます。たとえば、次の場所制約loc-db-preferは、databasesでタグ付けしたリソースのセットに適用されます。

```
location loc-db-prefer databases 100: alice
```

選択したクラスタ管理ツールでタグを作成する方法については、次を参照してください。

- Hawk2: [手順7.12「タグの追加」](#)
- crmsh: [8.4.7項「リソースのグループ化/タグ付け」](#)

6.6 リモートホストでのサービスの管理

リモートホストでサービスを監視および管理できることが、ここ数年の間にますます重要になってきています。SUSE Linux Enterprise High Availability Extension 11 SP3では、監視プラグインを介したリモートホスト上のサービスの詳細な監視機能を提供してきました。SUSE Linux Enterprise High Availability Extension 15 SP3では、最近追加された pacemaker_remote サービスを使用すると、リモートマシンにクラスタスタックをインストールしていなくても、実際のクラスタノードと同様にリモートホスト上のリソースを全面的に管理および監視できます。

6.6.1 監視プラグインを使用したリモートホストでのサービスの監視

仮想マシンの監視はVMエージェント(ハイパーバイザにゲストが出現する場合のみチェックを行う)を使用して行うか、VirtualDomainまたはXenエージェントから呼び出される外部スクリプトによって行うことができます。これまでは、精度の高い監視を行うには、仮想マシン内にHigh Availabilityスタックを完全にセットアップするしか方法がありませんでした。

今回、High Availability Extensionでは、監視プラグイン(旧称はNagiosプラグイン)に対するサポートを提供することで、リモートホスト上のサービスを監視できるようになりました。ゲストイメージを変更することなく、ゲストの外部ステータスを収集できます。たとえば、VMゲストはWebサービスまたは単純なネットワークリソースを実行している可能性があります、これらはアクセス可能である必要があります。Nagiosリソースエージェントによって、ゲスト上のWebサービスまたはネットワークリソースを監視できるようになりました。これらのサービスにアクセスできなくなった場合は、High Availability Extensionがそれぞれのゲストの再起動またはマイグレーションをトリガします。

ゲストがサービス(そのゲストによって使用されるNFSサーバなど)に依存している場合、そのサービスは、クラスタによって管理される通常のリソースか、Nagiosリソースによって監視される外部サービスのどちらかにすることができます。

Nagiosリソースを設定するには、ホスト上に次のパッケージをインストールする必要があります：

- [monitoring-plugins](#)
- [monitoring-plugins-metadata](#)

必要に応じて、YaSTまたはZypperが、これ以上のパッケージに対する依存性を解決します。一般的な使用例としては、1つのリソースコンテナに属するリソースとして監視プラグインを設定します。このリソースコンテナは通常はVMです。いずれかのリソースに障害が発生したら、このコンテナが再起動されます。設定例については、[例6.10「監視プラグインのリソースの設定」](#)を参照してください。または、Nagiosリソースエージェントを使用してネットワーク経由でホストまたはサービスを監視する場合、このエージェントを通常のリソースとして設定することもできます。

例 6.10: 監視プラグインのリソースの設定

```
primitive vml ocf:heartbeat:VirtualDomain \  
    params hypervisor="qemu:///system" config="/etc/libvirt/qemu/vml.xml" \  
    op start interval="0" timeout="90" \  
    op stop interval="0" timeout="90" \  
    op monitor interval="10" timeout="30"
```



```
primitive vm1-sshd nagios:check_tcp \
  params hostname="vm1" port="22" \ ❶
  op start interval="0" timeout="120" \ ❷
  op monitor interval="10"
group g-vm1-and-services vm1 vm1-sshd \
  meta container="vm1" ❸
```

- ❶ サポートされるパラメータは、監視プラグインの長いオプションと同じです。プラグインは、パラメータ`hostname`によってサービスと接続します。したがって、この属性の値は解決可能なホスト名かIPアドレスである必要があります。
- ❷ ゲストオペレーティングシステムが起動してサービスが実行されるまでには少し時間がかかるので、監視リソースの起動タイムアウトは十分な長さに設定する必要があります。
- ❸ タイプが`ocf:heartbeat:Xen`、`ocf:heartbeat:VirtualDomain`、または`ocf:heartbeat:lxc`のクラスタリソースコンテナ。VMまたはLinuxコンテナのいずれかに設定できます。

上の例には、`check_tcp`プラグイン用の1つのリソースしか含まれていませんが、様々なプラグインタイプ(たとえば、`check_http`や`check_udp`など)用に複数のリソースを設定することもできます。

複数のサービスのホスト名が同じである場合、`hostname`パラメータを個別のプリミティブに追加するのではなく、グループに対して指定することもできます。例:

```
group g-vm1-and-services vm1 vm1-sshd vm1-httpd \
  meta container="vm1" \
  params hostname="vm1"
```

監視プラグインによって監視されているいずれかのサービスに、VM内で障害が発生した場合は、クラスタがこれを検出し、コンテナリソース(VM)を再起動します。この場合に実行される操作は、サービスの監視操作に関する`on-fail`属性を指定することで設定できます。デフォルトでは、`restart-container`に設定されています。

VMのマイグレーションしきい値を検討する場合は、サービスの障害発生回数が考慮されます。

6.6.2 `pacemaker_remote`を使用したリモートノードでのサービスの管理

`pacemaker_remote`サービスを使用すると、High Availabilityクラスタを仮想ノードまたはリモートベアメタルマシンに拡張することができます。クラスタスタックを実行して、クラスタのメンバーになる必要はありません。

High Availability Extensionでは現在、仮想環境(KVMおよびLXC)、およびこれらの仮想環境内に存在するリソースを起動できるようになりました(PacemakerまたはCorosyncの実行に仮想環境は必要としません)。

クラスタリソースとしての仮想マシンおよびVM内に存在するリソースの両方を管理する使用例では、次の設定を使用できるようになりました。

- 「通常」(ベアメタル)クラスタノードは、High Availability Extensionを実行します。
- 仮想マシンは、pacemaker_remoteサービスを実行します(VM側に必要な設定はほとんどありません)。
- 「通常」クラスタノード上のクラスタスタックはVMを起動し、VM上で実行されているpacemaker_remoteサービスに接続して、それらをリモートノードとしてクラスタに統合します。

リモートノードでクラスタスタックがインストールされていないときは、これには次の意味があります。

- リモートノードはクォーラムに参加しません。
- リモートノードはDCになることはできません。
- リモートノードは、スケーラビリティの制約に制限されません(Corosyncには32ノードのメンバー制限があります)。

remote_pacemakerサービスに関する詳細については(詳細な設定手順からなる複数の使用例を含む)、項目「Pacemaker Remote Quick Start」を参照してください。

6.7 システムヘルスの監視

ノードがディスク容量が使い尽くしたために、そこに割り当てられたリソースを管理できなくなることを避けるため、High Availability Extensionでは、ocf:pacemaker:SysInfoというリソースエージェントが提供されています。これを使用して、ディスクパーティションに関してノードのヘルスを監視します。SysInfo RAは、#health_diskという名前のノード属性を作成します。この属性は、監視対象のディスク空き容量が指定された制限を下回るとredに設定されます。

ノードのヘルスがクリティカルな状態に達した場合のCRMの対応方法を定義するには、グローバルなクラスタオプションであるnode-health-strategyを使用します。

手順 6.2: システムヘルスの監視設定

ノードがディスク容量を使い尽くした場合に、リソースを自動的にノードから移動させるには、次の手順に従います。

1. `ocf:pacemaker:SysInfo`リソースを設定します。

```
primitive sysinfo ocf:pacemaker:SysInfo \  
  params disks="/tmp /var" ① min_disk_free="100M" ② disk_unit="M" ③ \  
  op monitor interval="15s"
```

- ① 監視対象のディスクパーティション。たとえば、`/tmp`、`/usr`、`/var`、`/dev`など。複数のパーティションを属性値として指定するには、空白で区切ります。



注記: `/`のファイルシステムは常に監視されます。

`disks`でルートパーティション(`/`)を指定する必要はありません。これはデフォルトで常に監視されます。

- ② これらのパーティションの必要最小限の空きディスク容量。オプションで、計測に使用する単位を指定できます(上記の例では、メガバイトを表すMが使用されています)。指定しない場合、`min_disk_free`は`disk_unit`パラメータで定義されている単位にデフォルト設定されます。
- ③ ディスク容量をレポートする場合の単位。

2. リソース設定を完了するには、`ocf:pacemaker:SysInfo`のクローンを作成し、各クラスターノードでそれを起動します。

3. `node-health-strategy`を`migrate-on-red`に設定します。

```
property node-health-strategy="migrate-on-red"
```

`#health_disk`属性が`red`に設定されている場合、`pacemaker-schedulerd`によって、そのノードのリソースのスコアに`-INF`が追加されます。これにより、このノードからすべてのリソースが移動します。この処理は`STONITH`リソースのところで停止しますが、`STONITH`リソースが実行されていない場合でも、ノードをフェンスすることができます。フェンスでCIBに直接アクセスすることで、動作を続行できるからです。

ノードのヘルス状態が`red`になったら、原因となる問題を解決します。次に`red`ステータスをクリアして、ノードを再びリソースの実行に適した状態にします。クラスターノードにログインして、次のいずれかの方法を使用します。

- 次のコマンドを実行します。

```
root # crm node status-attr NODE delete #health_disk
```

- 該当するノードでPacemakerを再起動します。
- ノードを再起動します。



ノードがサービスに復帰し、再びリソースを実行できるようになります。

6.8 詳細の参照先

<http://crmsh.github.io/> 

crmシェル(crmsh)、High Availabilityクラスタ管理用の高度なコマンドラインインタフェースのホームページ。

<http://crmsh.github.io/documentation> 

crmshを使用した基本的なクラスタ設定の『Getting Started』チュートリアルとcrmシェルの包括的なマニュアルを含む、crmシェルに関するいくつかのドキュメント。マニュアルは<http://crmsh.github.io/man-2.0/> で入手できます。チュートリアルは<http://crmsh.github.io/start-guide/> に用意されています。

<http://clusterlabs.org/> 

High Availability Extensionに含まれているクラスタリソースマネージャであるPacemakerのホームページ。

<http://www.clusterlabs.org/pacemaker/doc/> 

いくつかの包括的なマニュアルと一般的な概念を説明するより簡潔なドキュメント。例:

- 『Pacemaker Explained』: 参考として包括的で詳細な情報が記載されています。
- 『Colocation Explained』
- 『オーダーの概要』

7 Hawk2を使用したクラスタリソースの設定と管理

クラスタリソースを設定および管理する場合、Hawk2、またはcrmシェル(crmsh) コマンドラインユーティリティのいずれかを使用します。Hawkがインストールされている前のバージョンのSUSE® Linux Enterprise High Availability Extensionからアップグレードする場合は、パッケージが現在のバージョン、Hawk2で置き換えられます。

Hawk2のユーザフレンドリなWebインタフェースを使用すると、High Availability クラスタをLinuxまたは非Linuxマシンから同様に監視および管理することができます。Hawk2には、(グラフィカルな)Webブラウザを使用して、クラスタの内部または外部の任意のマシンからアクセスできます。

7.1 Hawk2の要件

Hawk2にログインするには、次の要件を満たす必要があります。

hawk2 パッケージ

sapMDCsapVirtHostname hawk2 パッケージは、Hawk2と接続するすべてのクラスタ ノードにインストールする必要があります。

Webブラウザ

Hawk2を使用してクラスタノードにアクセスするマシンに必要なものは、JavaScriptとクッキーを有効にして接続を確立できるグラフィカルなWebブラウザです。

Hawk2サービス

Hawk2を使用するには、このWebインタフェースで接続するノード上で、それぞれのWebサービスが開始されている必要があります。[手順7.1「Hawk2サービスの開始」](#)を参照してください。

`ha-cluster-bootstrap` パッケージからスクリプトを使用してクラスタをセットアップした場合、Hawk2サービスはすでに有効になっています。

各クラスタノードのユーザ名、グループおよびパスワード

Hawk2ユーザは`haclient`グループのメンバーである必要があります。インストール時に`hacluster`という名前のLinuxユーザが作成されますが、このユーザが`haclient`グループに追加されます。

セットアップ用に `ha-cluster-init` スクリプトを使用している場合は、`hacluster` ユーザに対してデフォルトパスワードが設定されます。Hawk2を起動する前に、安全なパスワードに変更してください。`ha-cluster-init` スクリプトを使用しなかった場合は、最初に `hacluster` のパスワードを設定するか、または `haclient` グループのメンバーである新しいユーザを作成します。Hawk2を使用して接続する各ノードでこれを実行します。

ワイルドカード証明書の処理

ワイルドカード証明書は、複数のサブドメインに対して有効な公開鍵証明書です。たとえば、`*.example.com` のワイルドカード証明書は、ドメイン `www.example.com`、`login.example.com` などをセキュリティ保護します。

Hawk2はワイルドカード証明書と従来の証明書をサポートします。自己署名のデフォルトの公開鍵および証明書は `/srv/www/hawk/bin/generate-ssl-cert` で生成されます。独自の証明書(従来型またはワイルドカード)を使用するには、`/etc/ssl/certs/hawk.pem` で生成された証明書を独自の証明書に置き換えます。

手順 7.1: HAWK2サービスの開始

1. 接続先にするノードで、シェルを開き、`root` としてログインします。
2. 次のように入力して、サービスのステータスをチェックします。

```
root # systemctl status hawk
```

3. サービスが実行されていない場合は、次のコマンドでサービスを開始します。

```
root # systemctl start hawk
```

ブート時にHawk2を自動的に起動したい場合は、次のコマンドを実行します。

```
root # systemctl enable hawk
```

7.2 ログイン

Hawk2 Webインタフェースは、HTTPSプロトコルとポート7630を使用します。

Hawk2を使用して個々のクラスタノードにログインする代わりに、浮動、仮想IPアドレス (`IPaddr` または `IPaddr2`) をクラスタリソースとして設定できます。そのための特別な設定は不要です。サービスがどの物理ノードで実行されていても、クライアントはHawkサービスに接続できます。

クラスタを `ha-cluster-bootstrap` スクリプトを使用して設定する際には、クラスタ管理用に仮想IPを設定するかどうかを求められます。

手順 7.2: HAWK2 WEBインタフェースへのログイン

1. いずれかのマシンでWebブラウザを起動し、次のURLを入力します。

```
https://HAWKSERVER:7630/
```

HAWKSERVERの部分は、Hawk Webサービスを実行するクラスタノードのIPアドレスまたはホスト名で置き換えます。Hawk2でクラスタ管理用に仮想IPアドレスを設定した場合、その仮想IPアドレスでHAWKSERVERを置き換えます。



注記: 証明書の警告

初めてURLにアクセスしようとするときに証明書の警告が表示される場合は、自己署名証明書が使用されています。自己署名証明書は、デフォルトでは信頼されません。

証明書を検証するには、クラスタオペレータに証明書の詳細を求めます。

続行するには、ブラウザに例外を追加して警告をバイパスします。

自己署名証明書を公式認証局によって署名された証明書で置き換える方法の詳細については、[自己署名証明書の置き換え置換](#)を参照してください。

2. Hawk2ログイン画面で、haclusterユーザ(または、haclientグループのメンバーである他の任意のユーザ)のユーザ名とパスワードを入力します。
3. ログインをクリックします。

7.3 Hawk2の概要: 主な構成要素

Hawk2にログインすると、左側にはナビゲーションバー、右側には複数のリンクが含まれる最上位の行が表示されます。



注記: Hawk2で利用できる機能

デフォルトでは、rootまたはhaclusterとしてログインしたユーザは、すべてのクラスタ設定作業への、完全な読み込み/書き込みのアクセス権を持ちます。ただし、[アクセス制御リスト](#) (ACL)を使用すれば、より詳細なアクセス権限を定義することができます。

CRMでACLが有効になっている場合、Hawk2で利用できる機能は、ユーザ役割と割り当てられたアクセスパーミッションごとに異なります。Hawk2の履歴エクスプローラは、ユーザhaclusterのみが実行できます。

7.3.1 左のナビゲーションバー

監視機能

- 状態: クラスタの現在の状態の概要が表示されます(`crmsh`の`crm status`と同様です)。詳細については、7.8.1項「[単一クラスタの監視](#)」を参照してください。クラスタに`guest nodes` (`pacemaker_remote`デーモンが実行されているノード)が含まれる場合、それらのノードも表示されます。この画面はほぼリアルタイムで更新され、ノードまたはリソースの状態に変化があった場合、ほとんど瞬時に表示されます。
- ダッシュボード: 複数のクラスタを監視できます(Geoクラスタがセットアップされている場合は、別のサイトにあるクラスタも監視できます)。詳細については、7.8.2項「[複数のクラスタの監視](#)」を参照してください。クラスタに`guest nodes` (`pacemaker_remote`デーモンが実行されているノード)が含まれる場合、それらのノードも表示されます。この画面はほぼリアルタイムで更新され、ノードまたはリソースの状態に変化があった場合、ほとんど瞬時に表示されます。

トラブルシューティング

- 履歴: 履歴エクスプローラーを開いてクラスタレポートを生成できます。詳細については、7.10項「[クラスタ履歴の表示](#)」を参照してください。
- Command Log (コマンドログ): Hawk2によって最近実行された`crmsh`コマンドのリストを表示します。

設定

- リソースの追加: リソース設定画を開きます。詳細については、7.5項「[クラスタリソースの設定](#)」を参照してください。
- 制約の追加: 制約設定画面を開きます。詳細については、7.6項「[制約の設定](#)」を参照してください。
- ウィザード: さまざまなウィザードを選択できます。これにより、DRBDブロックデバイスなどの特定のワークロード用のリソースをウィザードに従って作成できます。詳細については、7.5.2項「[ウィザードを使用したリソースの追加](#)」を参照してください。
- 設定の編集: リソース、制約、ノード名と属性、タグ、アラート (http://crmsh.github.io/man/#cmdhelp_configure_alert)、フェンシングトポロジ (http://crmsh.github.io/man/#cmdhelp_configure_fencing_topology)などを編集できます。

- クラスタ設定: グローバルクラスタオプション、およびリソースと操作のデフォルトを変更できます。詳細については、[7.4項「グローバルクラスタオプションの設定」](#)を参照してください。
- アクセス制御 > Roles (役割): アクセス制御リスト(CIBへのアクセス権を記述した一連のルール)に対して役割を作成できる画面を開きます。詳細については、[手順13.2「Hawk2によるmonitor役割の追加」](#)を参照してください。
- アクセス制御 > Targets (ターゲット): アクセス制御リストのターゲット(システムユーザ)を作成して、そのターゲットに役割を割り当てることができる画面を開きます。詳細については、[手順13.3「Hawk2によるターゲットへの役割割り当て」](#)を参照してください。

7.3.2 最上位の行

Hawk2の最上位の行には、次のエントリが表示されます。

- バッチ: クリックすると、バッチモードに切り替わります。これにより、変更をシミュレートしてステージングし、それらの変更を単一のトランザクションとして適用できます。詳細については、[7.9項「バッチモードの使用」](#)を参照してください。
- ユーザ名: Hawk2用の設定ができます(たとえば、Webインタフェースの言語の設定やSTONITHを無効にした場合に警告を表示するかなどの設定)。
- Help (ヘルプ): SUSE Linux Enterprise High Availability Extensionドキュメントにアクセスしたり、リリースノートを参照したり、バグを報告したりします。
- ログアウト: クリックするとログアウトします。

7.4 グローバルクラスタオプションの設定

グローバルクラスタオプションは、一定の状況下でのクラスタの動作を制御します。これらは、セットにグループ化され、Hawk2、crmshなどのクラスタ管理ツールで表示し、変更することができます。事前に定義されている値は、通常は、そのまま保持できます。ただし、クラスタの主要機能を正しく機能させるには、クラスタの基本的なセットアップ後に、次のパラメータを調整する必要があります。

- グローバルオプションno-quorum-policy
- グローバルオプションstonith-enabled

手順 7.3: グローバルクラスタオプションの変更

1. Hawk2にログインします。

`https://HAWKSERVER:7630/`

2. 左のナビゲーションバーから、環境設定 > クラスタ設定を選択します。
クラスタ設定画面が開きます。グローバルクラスタオプションとその現在の値が表示されます。
画面の右側にパラメータの簡単な説明を表示するには、マウスポインタをパラメータに合わせます。



図 7.1: HAWK2 - クラスタの設定

3. no-quorum-policyおよびstonith-enabledの値を確認し、必要に応じて調整します。
 - a. no-quorum-policyを適切な値に設定します。詳しくは「6.2.2項「グローバルオプションno-quorum-policy」」を参照してください。
 - b. 何らかの理由でフェンシングを無効にする必要がある場合は、stonith-enabledをnoに設定します。通常のクラスタ操作にはSTONITHデバイスの使用が必要なため、デフォルトでは、trueに設定されています。デフォルト値では、クラスタは、STONITHリソースが設定されていない場合にはリソースの開始を拒否します。

❗ 重要: STONITHがない場合はサポートなし

- クラスタにはノードフェンシングメカニズムが必要です。
 - グローバルクラスタオプション `stonith-enabled` および `startup-fencing` を `true` に設定する必要があります。これらを変更するとサポートされなくなります。
- c. クラスタ設定からパラメータを削除するには、パラメータの横のマイナスアイコンをクリックします。パラメータを削除すると、クラスタはそのパラメータがデフォルト値に設定されている場合と同様に動作します。
- d. クラスタ設定に新たなパラメータを追加するには、ドロップダウンボックスから選択します。
4. リソースのデフォルトまたは操作のデフォルトを変更する必要がある場合は、次のような処理を実行します。
- a. 値を調整するには、ドロップダウンボックスから別の値を選択するか、値を直接編集します。
 - b. 新しいリソースのデフォルトまたは操作のデフォルトを追加するには、空のドロップダウンボックスから1つ選択し、値を入力します。デフォルト値が存在する場合は、Hawk2から自動的に提示されます。
 - c. パラメータを削除するには、その横のマイナスアイコンをクリックします。リソースのデフォルトと操作のデフォルトに値が指定されていない場合、クラスタは6.3.6項「リソースオプション(メタ属性)」および6.3.8項「リソース操作」にドキュメントされているデフォルト値を使用します。
5. 変更内容を確認します。

7.5 クラスタリソースの設定

クラスタの管理者は、クラスタ内のサーバ上の各リソースや、サーバ上で実行する各アプリケーションに対してクラスタリソースを作成する必要があります。クラスタリソースには、Webサイト、メールサーバ、データベース、ファイルシステム、仮想マシン、およびユーザが常時使用できるその他のサーバベースのアプリケーションまたはサービスなどが含まれます。

作成できるリソースタイプの概要については、[6.3.3項「リソースのタイプ」](#)を参照してください。リソースの基本情報(ID、クラス、プロバイダ、およびタイプ)を指定すると、Hawk2によって次のカテゴリが表示されます。

パラメータ(インスタンス属性)

リソースが制御するサービスのインスタンスを決定します。詳細については、[6.3.7項「インスタンス属性\(パラメータ\)」](#)を参照してください。

リソースを作成する際、Hawk2は必要なパラメータを自動的に表示します。これらを編集して、有効なリソースの設定を取得します。

操作

リソース監視に必要です。詳細については、[6.3.8項「リソース操作」](#)を参照してください。

リソースを作成する際、Hawk2は、重要なリソース操作を表示します([monitor](#)、[start](#)、および[stop](#))。

メタ属性

特定のリソースの処理方法をCRMに指示します。詳細については、[6.3.6項「リソースオプション\(メタ属性\)」](#)を参照してください。

リソースを作成する際、Hawk2はそのリソースの重要なメタ属性を自動的にリストにします(たとえばリソースの初期状態を定義する[target-role](#)属性です。デフォルトでは[Stopped](#)に設定されているため、リソースはすぐには始動しません)。

使用率

特定のリソースがノードから要求する容量をCRMに指示します。詳細については、[7.6.8項「負荷インパクトに基づくリソース配置の設定」](#)を参照してください。

これらのカテゴリのエントリと値は、リソースの作成中に調整することも、後から調整することもできます。

7.5.1 現在のクラスタ設定の表示(CIB)

クラスタ管理者はクラスタ設定を知る必要がある場合があります。Hawk2は、現在の設定をcrmシェル構文で、XMLとして、およびグラフとして表示できます。クラスタ設定をcrmシェル構文で表示するには、左ナビゲーションバーから、環境設定 > Edit Configuration (設定の編集)を選択し、表示をクリックします。代わりに設定をraw XMLで表示するには、XMLをクリックします。CIBで設定されたノードとリソースのグラフィカルな表現を示すには、グラフをクリックします。リソース間の関係も表示されます。

7.5.2 ウィザードを使用したリソースの追加

Hawk2ウィザードは、仮想IPアドレスやSDB STONITHリソースなどの単純なリソースを設定する場合に便利です。また、DRBDブロックデバイスやApache Webサーバのリソース設定などの、複数リソースを含む複雑な設定においても役立ちます。設定手順をウィザードに従って進めることができ、入力が必要なパラメータについては情報が提供されます。

手順 7.4: リソースウィザードの使用

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > ウィザードの順に選択します。
3. ウィザードの横にある下矢印アイコンをクリックして個々のカテゴリを展開し、目的のウィザードを選択します。
4. 画面の指示に従います。最後の設定手順が完了したら、Verify (検証)を選択して、入力した値を検証します。

Hawk2が実行するアクションと、設定の内容が表示されます。設定によっては、適用を選択して設定を適用する前に、rootパスワードの入力を求めるプロンプトが表示されることがあります。

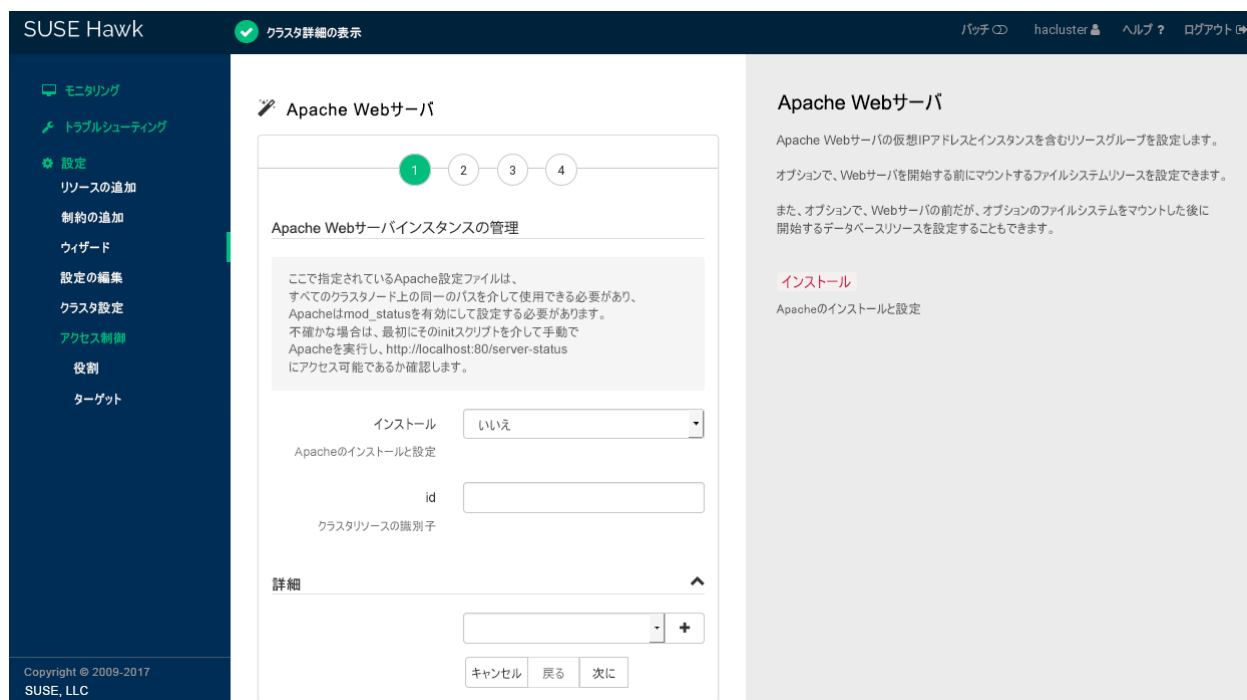


図 7.2: HAWK2 - APACHE WEBサーバ用のウィザード

7.5.3 単純なリソースの追加

最も基本的なタイプのリソースを作成するには、次の手順に従います。

手順 7.5: プリミティブリソースの追加

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > Add Resource (リソースの追加) > プリミティブ の順に選択します。
3. 固有のリソースIDを入力します。
4. リソース設定の基にするリソーステンプレートが存在する場合は、テンプレートで目的のテンプレートを選択します。テンプレートの設定の詳細については、[手順7.6「リソーステンプレートの追加」](#)を参照してください。
5. クラスで、使用するリソースエージェントのクラスを選択します。[lsb](#)、[ocf](#)、[service](#)、[stonith](#)、または[systemd](#)から選択できます。詳細については、[6.3.2項「サポートされるリソースエージェントクラス」](#)を参照してください。
6. [ocf](#)をクラスとして選択した場合、OCFリソースエージェントのプロバイダを指定します。OCFの指定によって、複数のベンダが同じリソースエージェントを提供できるようになります。
7. タイプリストから、使用するリソースエージェントを選択します(たとえばIPAddrまたはFilesystem)。このリソースエージェントの簡単な説明が表示されます。これで、リソースの基本情報が指定されました。



注記

タイプリストに表示される選択肢は、選択したクラス(OCFリソースの場合は、プロバイダも)によって異なります。



図 7.3: HAWK2 - プリミティブリソース

8. Hawk2によって提案されたパラメータ、操作、およびメタ属性を保持する場合は、作成をクリックして設定を終了します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。
- パラメータ、操作、またはメタ属性を調整するには、[7.5.5項「リソースの変更」](#)を参照してください。リソースの利用率属性を設定するには、[手順7.21「リソースが要求する容量の設定」](#)を参照してください。

7.5.4 リソーステンプレートの追加

類似した設定のリソースを多く作成する最も簡単な方法は、リソーステンプレートを定義することです。リソーステンプレートを定義した後は、プリミティブの中や、特定のタイプの制約で参照できるようになります。リソーステンプレートの機能と使用方法の詳細については、[6.5.3項「リソーステンプレートと制約」](#)を参照してください。

手順 7.6: リソーステンプレートの追加

リソーステンプレートは、プリミティブリソースと同様の方法で設定します。

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > Add Resource (リソースの追加) > テンプレートの順に選択します。
3. 固有のリソースIDを入力します。
4. 手順7.5「プリミティブリソースの追加」のステップ5以降の手順に従います。

7.5.5 リソースの変更

リソースの作成後、必要に応じてパラメータ、操作、またはメタ属性を調整することで、いつでもその設定を編集できます。

手順 7.7: リソースのパラメータ、操作、またはメタ属性の変更

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. Hawk2の状態画面で、Resources (リソース)リストに移動します。
3. 操作列で、変更したいリソースまたはグループの横にある下矢印アイコンをクリックして編集を選択します。
リソース設定画面が開きます。

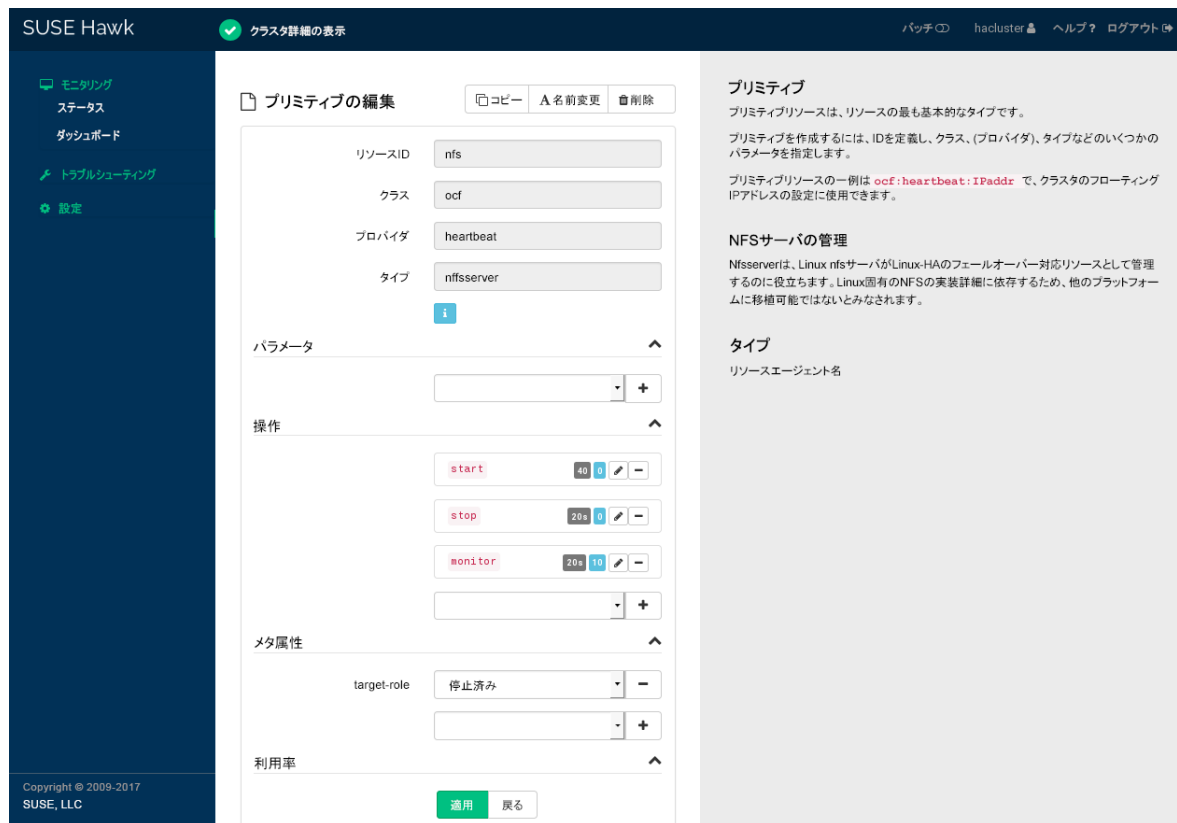


図 7.4: HAWK2 - プリミティブリソースの編集

4. 新たなパラメータ、操作、またはメタ属性を追加するには、空のドロップダウンボックスから項目を選択します。
5. 操作カテゴリの値を編集するには、それぞれのカテゴリの編集アイコンをクリックして操作の別の値を入力し、適用をクリックします。
6. 完了したら、リソース設定画面で適用ボタンをクリックして、パラメータ、操作、またはメタ属性の変更内容を確認します。
画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。

7.5.6 STONITHリソースの追加

！ 重要: STONITHがない場合はサポートなし

- クラスタにはノードフェンシングメカニズムが必要です。
- グローバルクラスタオプション`stonith-enabled`および`startup-fencing`を`true`に設定する必要があります。これらを変更するとサポートされなくなります。

デフォルトでは、グローバルクラスタオプションの`stonith-enabled`は`true`に設定されています。STONITHリソースが定義されていない場合、クラスタはどのリソースも開始することを拒否します。STONITHリソースが定義されていない場合、クラスタはどのリソースを開始することも拒否します。1つ以上のSTONITHリソースを設定して、STONITHのセットアップを完了します。SBD、libvirt (KVM/Xen)、またはvCenter/ESX ServerのSTONITHリソースを追加する場合、Hawk2ウィザードを使用するのが最も簡単な方法です(7.5.2項「ウィザードを使用したリソースの追加」を参照してください)。STONITHは他のリソースと同様に設定しますが、その動作はいくつかの点で異なります。詳細については、10.3項「STONITHのリソースと環境設定」を参照してください。

手順 7.8: STONITHリソースの追加

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > Add Resource (リソースの追加) > プリミティブ の順に選択します。
3. 固有のリソースIDを入力します。
4. クラスリストで、リソースエージェントクラスとして`stonith`を選択します。
5. タイプリストから、使用しているSTONITHデバイスを制御するためのSTONITHプラグインを選択します。このプラグインの簡単な説明が下に表示されます。
6. Hawk2は、自動的にそのリソースに必要なパラメータを表示します。それぞれのパラメータの値を入力します。
7. Hawk2は、重要なリソース操作を表示し、デフォルト値を提案します。ここで設定を変更しない場合、確定するとすぐに、Hawk2は提案した操作およびデフォルト値を追加します。

8. 変更理由がない場合は、デフォルトのメタ属性設定を保持します。

The screenshot shows the SUSE Hawk web interface for creating a primitive resource. The main form is titled 'プリミティブの作成' (Create Primitive). It contains several sections: 'リソースID' (Resource ID) with the value 'stonith-1', 'テンプレート' (Template) as a dropdown, 'クラス' (Class) as 'stonith', 'プロバイダ' (Provider) as 'apcmaster', and 'タイプ' (Type) as 'apcmaster'. Below these are 'パラメータ' (Parameters) including 'ipaddr' (10.161.15.43), 'ログイン' (login: stonithmaster), and 'パスワード' (password: STRONG_PASSWORD). There is a '操作' (Operations) section with buttons for 'start', 'stop', and 'monitor'. The 'メタ属性' (Meta-attributes) section shows 'target-role' set to '停止' (Stop). At the bottom, there is a '利用率' (Usage) section. On the right side of the interface, there is a sidebar with text explaining 'プリミティブ' (Primitive) and 'APC MasterSwitch'.

図 7.5: HAWK2 - STONITHリソース

9. 変更を確認して、STONITHリソースを作成します。

画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。

フェンシングを設定するには、制約を追加します。詳細については、[第10章「フェンシングとSTONITH」](#)を参照してください。

7.5.7 クラスタリソースグループの追加

クラスタリソースの中には、他のコンポーネントやリソースに依存しているものもあります。このような場合、各コンポーネントまたはリソースは特定の順番で起動し、同じサーバ上で動作する必要があります。この設定を簡単にするため、SUSE Linux Enterprise High Availability Extensionは、グループのコンセプトをサポートしています。

リソースグループには、一緒の場所で見つけ、連続して開始し、逆の順序で停止する必要のあるリソースセットが含まれます。リソースグループの例と、グループとそのプロパティの詳細について、[6.3.5.1項「グループ」](#)を参照してください。



注記: 空のグループ

グループには1つ以上のリソースを含む必要があります。空の場合は設定は無効になります。グループの作成中に、Hawk2ではさらにプリミティブを作成し、それらをグループに追加できます。詳細については、[7.7.1項「リソースとグループの編集」](#)を参照してください。

手順 7.9: リソースグループの追加

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > Add Resource (リソースの追加) > グループの順に選択します。
3. 固有のグループIDを入力します。
4. グループメンバーを定義するには、子リストで1つまたは複数のエントリを選択します。グループメンバーを再ソートするには、右側の「ハンドル」アイコンを使用して、メンバーを目的の順序にドラッグアンドドロップします。
5. 必要に応じて、メタ属性を変更または追加します。
6. 作成をクリックして、設定を完了します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。

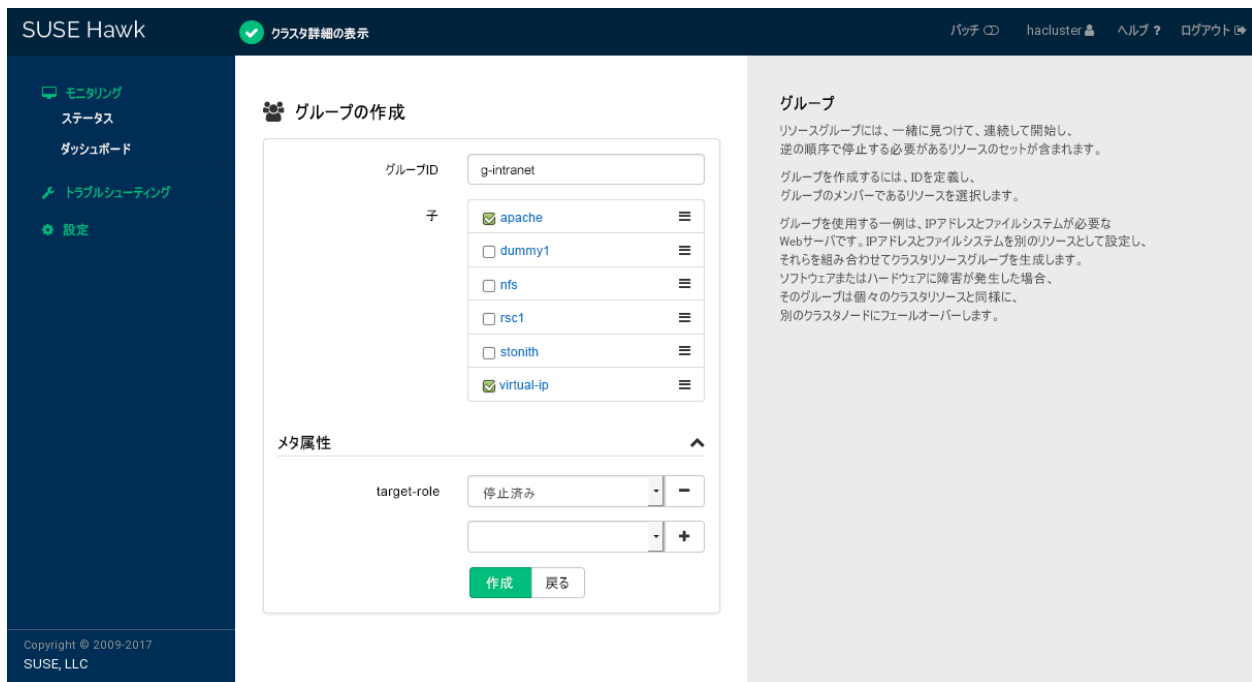


図 7.6: HAWK2 - リソースグループ

7.5.8 クローンリソースの追加

特定のリソースをクラスタ内の複数のノードで同時に実行することが必要な場合には、それらのリソースをクローンとして設定します。クローンとして設定できるリソースの一例は、OCFS2などのクラスタファイルシステムの`ocf:pacemaker:controld`です。標準のリソースまたはリソースグループであれば、どれでもクローンを作成できます。クローンリソースの各インスタンスは、すべて同じ動作にすることができます。ただし、ホストされているノードに応じて設定を変えることもできます。

利用可能なリソースクローンのタイプの概要は、[6.3.5.2項「クローン」](#)を参照してください。



注記: クローンの子リソース

クローンには、プリミティブまたはグループのいずれかを子リソースとして含めることができます。Hawk2では、クローンの作成中に子リソースを作成したり変更したりすることはできません。クローンを追加する前に、子リソースを作成し、必要に応じて設定しておいてください。詳細については、[7.5.3項「単純なリソースの追加」](#)または[7.5.7項「クラスタリソースグループの追加」](#)を参照してください。

手順 7.10: クローンリソースの追加

1. Hawk2にログインします。

`https://HAWKSERVER:7630/`

2. 左のナビゲーションバーから、環境設定 > Add Resource (リソースの追加) > クローンの順に選択します。
3. 固有のクローンIDを入力します。
4. 子リソースリストから、クローンのサブリソースとして使用するプリミティブまたはグループを選択します。
5. 必要に応じて、メタ属性を変更または追加します。
6. 作成をクリックして、設定を完了します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。

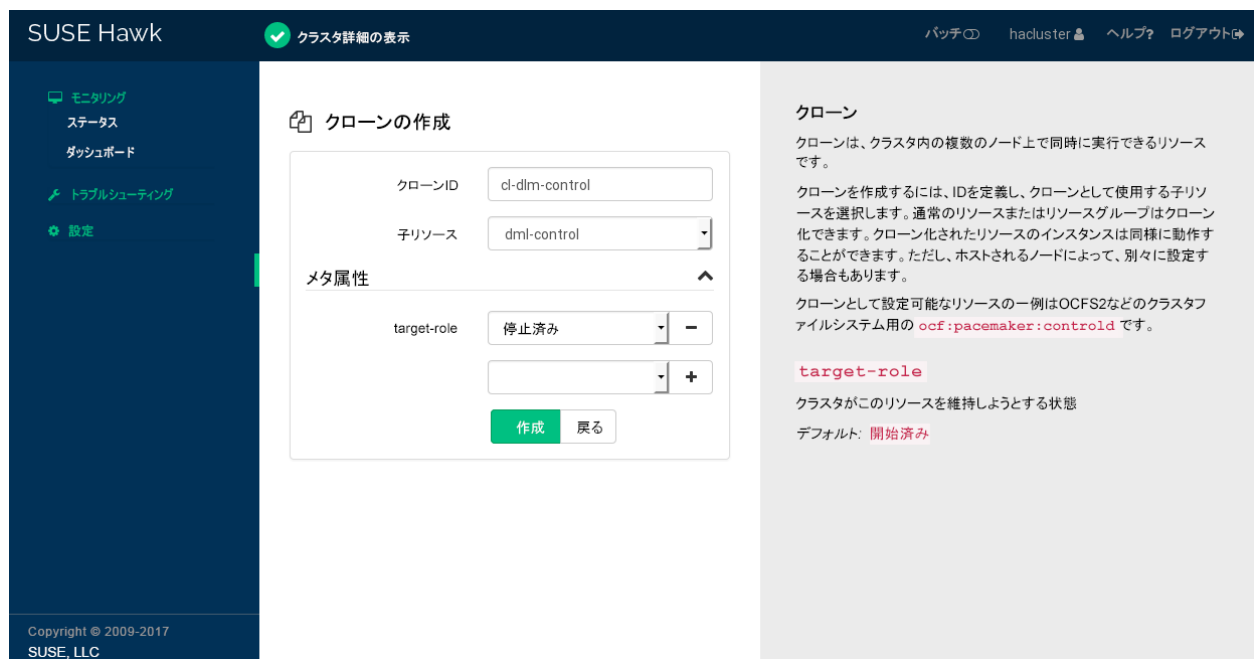


図 7.7: HAWK2 - クローンリソース

7.5.9 マルチステートリソースの追加

マルチステートリソースは、クローンが得意とするところです。これにより、インスタンスを2つの動作モード(active/passive、primary/secondary、またはmaster/slaveと呼ばれます)のいずれかに設定できます。マルチステートリソースは、グループまたは通常リソースを1つだけ含む必要があります。

リソースの監視または制約を設定する場合、マルチステートリソースには、単純なリソースとは異なる要件があります。詳細については、『Pacemaker Explained』(<http://www.clusterlabs.org/pacemaker/doc/>から入手可)を参照してください。特に、「Multi-state - Resources That Have Multiple Modes」のセクションを参照してください。



注記: マルチステートリソースの子リソース

マルチステートリソースには、プリミティブまたはグループのいずれかを子リソースとして含めることができます。Hawk2では、マルチステートリソースの作成中に子リソースを作成したり変更したりすることはできません。マルチステートリソースを追加する前に、子リソースを作成し、必要に応じて設定しておいてください。詳細については、7.5.3項「単純なリソースの追加」または7.5.7項「クラスタリソースグループの追加」を参照してください。

手順 7.11: マルチステートリソースの追加

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > Add Resource (リソースの追加) > マルチステートの順に選択します。
3. マルチステートIDに固有のIDを入力します。
4. 子リソースリストから、マルチステートリソースのサブリソースとして使用するプリミティブまたはグループを選択します。
5. 必要に応じて、メタ属性を変更または追加します。
6. 作成をクリックして、設定を完了します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。

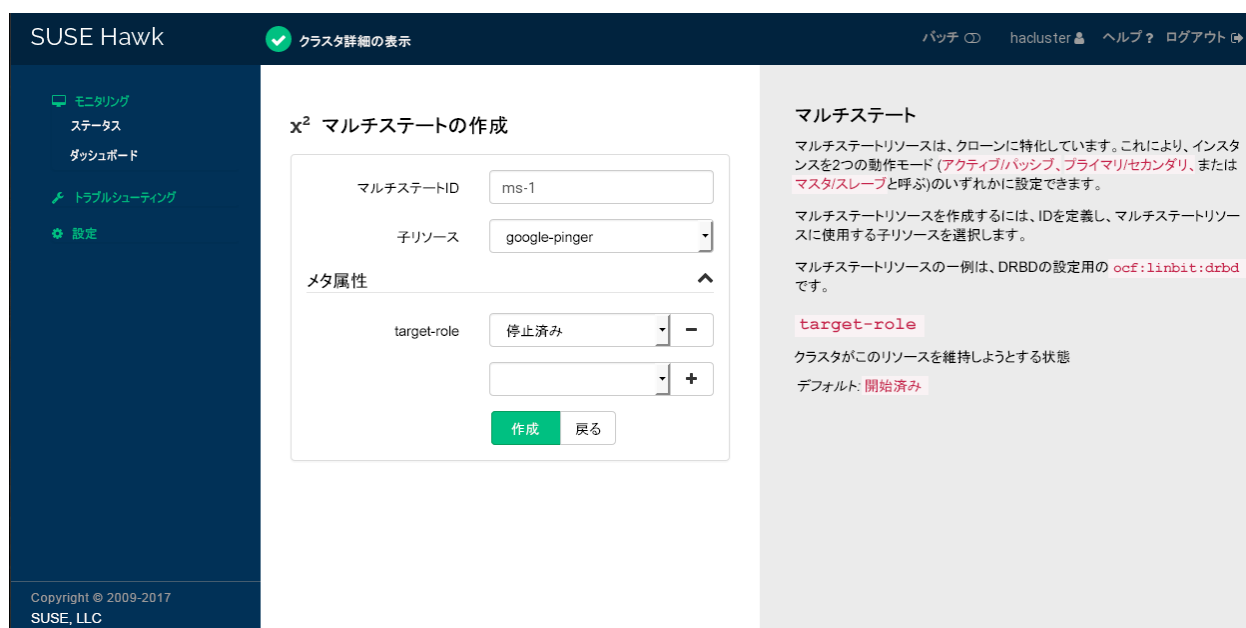


図 7.8: HAWK2 - マルチステートリソース

7.5.10 タグの使用によるリソースのグループ化

タグは、コロケーションの作成や関係の順序付けを行わずに、複数のリソースをただちに参照する方法です。タグを使用して、概念的に関連するリソースをグループ化できます。たとえば、データベースに関連する複数のリソースがある場合、関連するすべてのリソースを `database` という名前のタグに追加できます。

同じタグに属するすべてのリソースは、1つのコマンドで開始または停止できます。

手順 7.12: タグの追加

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > Add Resource (リソースの追加) > タグの順に選択します。
3. タグIDに固有のIDを入力します。
4. オブジェクトリストから、このタグで参照するリソースを選択します。
5. 作成をクリックして、設定を完了します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。



図 7.9: HAWK2 - タグ

7.5.11 リソース監視の設定

High Availability Extensionは、ノードの障害を検出するだけでなく、ノードの個々のリソースで障害が発生した場合、そのことも検出します。リソースが実行中であるかどうかを確認するには、そのリソースにリソースの監視を設定します。通常、リソースは動作中にのみ、クラスタによって監視されます。しかし、同時実行違反を検出するために、停止されるリソースの監視も設定する必要があります。リソースを監視するには、タイムアウト、開始遅延のいずれか、または両方と、間隔を指定します。間隔の指定によって、CRMにリソースステータスの確認頻度を指示します。startまたはstop操作に対するtimeoutなど、特定のパラメータも設定できます。

手順 7.13: 操作の追加または変更

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 手順7.5「プリミティブリソースの追加」の説明に従ってリソースを追加するか、既存のプリミティブを選択して編集します。
Hawk2は、重要な操作(start、stop、monitor)を自動的に表示し、デフォルト値を提案します。
提案された各値に属する属性を参照するには、マウスポインタをそれぞれの値に合わせます。

The image shows a user interface for configuring operations. It consists of three rows, each with a label ('start', 'stop', 'monitor') and a numeric input field. The 'start' and 'stop' rows have a value of 20. The 'monitor' row has a value of 20 and a 'timeout' of 10. Each row has a pencil icon for editing and a minus sign for decreasing the value. A tooltip labeled 'timeout' points to the '10' value in the 'monitor' section.

3. startまたはstop操作に対して提案されたtimeoutの値を変更するには、次の手順に従います。
 - a. 操作の隣のペンアイコンをクリックします。
 - b. 表示されるダイアログで、timeoutパラメータに別の値(例: 10)を入力し、変更内容を確認します。
4. monitor操作に対して提案されたintervalの値を変更するには、次の手順に従います。
 - a. 操作の隣のペンアイコンをクリックします。
 - b. 表示されるダイアログで、intervalに対して監視間隔の別の値を入力します。
 - c. リソースが停止されている場合にリソース監視を設定するには、次の手順に従います。
 - i. 下にある空のドロップダウンボックスから役割項目を選択します。
 - ii. [役割] ドロップダウンボックスから、[Stopped (停止済み)] を選択します。
 - iii. 適用をクリックし、変更内容を確認して操作のダイアログを閉じます。
5. リソース設定画面で変更内容を確認します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。

リソースモニタが障害を検出した場合の処理については、[6.4項「リソース監視」](#)を参照してください。

リソースの障害を表示するには、Hawk2で状態画面に切り替えて、関係するリソースを選択します。操作列で、下矢印アイコンをクリックして最近のイベントを選択します。表示されるダイアログに、リソースに対して実行された最近のアクションが一覧表示されます。障害は赤で表示されます。リソースの詳細を表示するには、操作列で虫眼鏡アイコンをクリックします。

Q

nfs

プリミティブ

エージェント

ocf:heartbeat:nfsserver

メタ属性

target-role

停止済み

操作

名前	タイムアウト	間隔
開始	40	0
停止	20s	0
モニタ	20s	10

制約

ID	タイプ	スコア	宛先
----	-----	-----	----

閉じる

図 7.10: HAWK2 - リソース詳細

7.6 制約の設定

すべてのリソースを設定したら、クラスタがそれらを扱う方法を指定します。リソース制約を使えば、リソースがどのクラスタノードで実行されるか、リソースをどの順番でロードするか、そして特定のリソース型のどのリソースに依存するかを指定することができます。

利用可能な制約のタイプの概要は、[6.5.1項「制約のタイプ」](#)を参照してください。制約を定義する際には、スコアも指定する必要があります。スコアおよびクラスタでのそれらの意味の詳細については、[6.5.2項「スコアと無限大」](#)を参照してください。

7.6.1 場所制約の追加

場所制約は、リソースを実行できるノード、実行に適したノード、または実行できないノードを決定します。たとえば、場所制約により、特定のデータベースに関連するすべてのリソースを同じノードに配置します。

手順 7.14: [場所制約の追加](#)

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > Add Constraint (制約の追加) > Location (場所) の順に選択します。

3. 固有の制約IDを入力します。

4. Resources (リソース)のリストから、制約を定義するリソースを1つまたは複数選択します。

5. スコアにスコアを入力します。スコアはこのリソース制約に割り当てる値を示します。正の値は、次のステップで指定するノードでリソースを実行できることを示します。負の値は、リソースをそのノードで実行すべきではないことを示します。スコアの高い制約は、それよりもスコアが低い制約より先に適用されます。使用頻度の高い次の値は、ドロップダウンボックスからも設定できます。

- ノードで強制的にリソースを実行するには、矢印アイコンをクリックして常にを選択します。これにより、スコアはINFINITYに設定されます。
- ノードでリソースを実行しない場合、矢印アイコンをクリックしてNever (実行しない)を選択します。これにより、スコアは-INFINITYに設定され、リソースはそのノードで実行してはならないことになります。
- スコアを0に設定するには、矢印アイコンをクリックしてAdvisory (推奨値)を選択します。これにより制約が無効になります。これは、リソース検出を設定してもリソースを制約したくない場合に便利です。

6. ノードを選択します。

7. 作成をクリックして、設定を完了します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。



図 7.11: HAWK2 - 場所制約

7.6.2 コロケーション制約の追加

コロケーション制約は、ノード上で一緒に実行可能な、または一緒に実行することが禁止されているリソースをクラスタに伝えます。コロケーション制約はリソース間の依存関係を定義するため、コロケーション制約を作成するには、少なくとも2つのリソースが必要です。

手順 7.15: コロケーション制約の追加

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > Add Constraint (制約の追加) > コロケーションの順に選択します。
3. 固有の制約IDを入力します。
4. スコアにスコアを入力します。スコアは複数のリソースの場所の関係を決定します。正の値は、リソースを同じノードで実行しなければならないことを示します。負の値は、リソースを同じノードで実行するべきではないことを示します。スコアと他の要因との組み合わせによって、ノードの配置先が決定します。
使用頻度の高い次の値は、ドロップダウンボックスからも設定できます。

- 同じノードで強制的にリソースを実行するには、矢印アイコンをクリックして常にを選択します。これにより、スコアはINFINITYに設定されます。
- リソースを同じノードで実行しない場合、矢印アイコンをクリックしてNever (実行しない)を選択します。これにより、スコアは-INFINITYに設定され、リソースは同じノードで実行してはならないことになります。

5. 制約のリソースを定義するには、次の手順に従います。

- a. リソースカテゴリのドロップダウンボックスから、リソース(またはテンプレート)を選択します。
リソースが追加され、下に新しい空のドロップダウンボックスが表示されます。
- b. この手順を繰り返してリソースを追加します。
最上位のリソースは次のリソースなど順に依存するため、クラスタはまず最後のリソースを置く場所を決め、次にその決定に基づいて依存するものを配置していきます。制約が満たされないと、クラスタは依存するリソースが実行しないようにすることがあります。
- c. コロケーション制約内のリソースの順序を入れ替えるには、リソースの横にある上矢印アイコンをクリックして、その上のエントリと入れ替えます。

6. 必要に応じて、各リソースの他のパラメータ(Started (開始)、Stopped (停止)、Master (マスタ)、Slave (スレーブ)、Promote (昇格)、Demote (降格)など)を指定します。リソースの横にある空のドロップダウンリストをクリックして、目的のエントリを選択します。

7. 作成をクリックして、設定を完了します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。

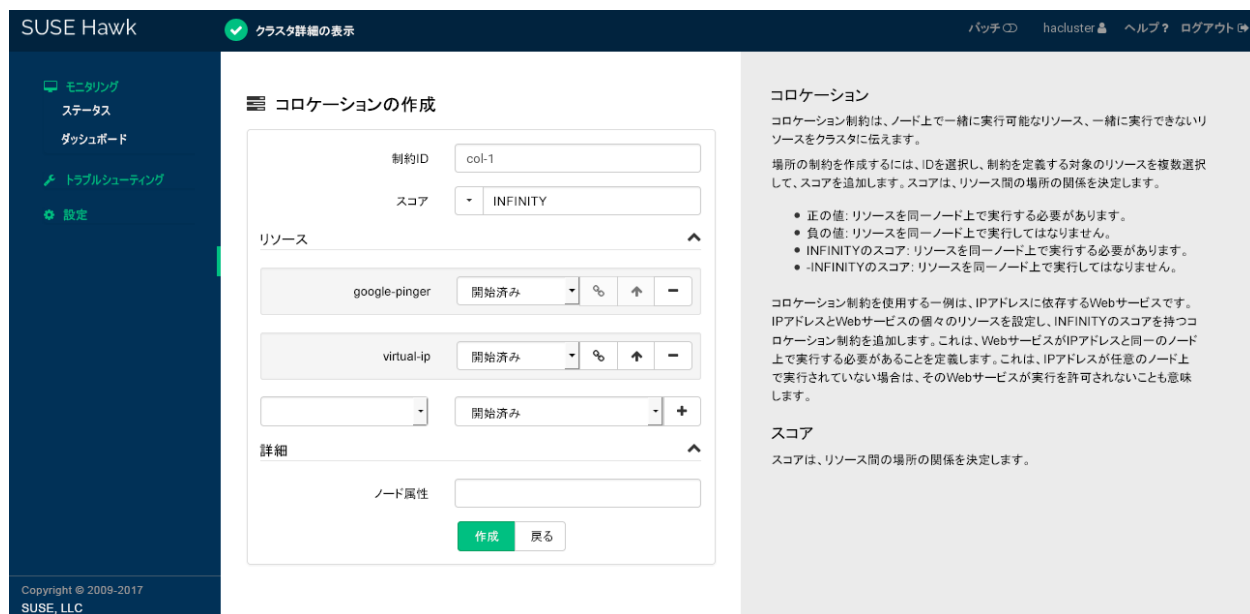


図 7.12: HAWK2 - コロケーション制約

7.6.3 順序制約の追加

順序制約は、リソースを開始および停止する順序を定義します。順序制約はリソース間の依存関係を定義するため、順序制約を作成するには、少なくとも2つのリソースを作成する必要があります。

手順 7.16: 順序制約の追加

1. Hawk2にログインします。

`https://HAWKSERVER:7630/`

2. 左のナビゲーションバーから、環境設定 > Add Constraint (制約の追加) > 順序 の順に選択します。
3. 固有の制約IDを入力します。
4. スコアにスコアを入力します。スコアがゼロより大きい場合、順序制約は必須になりますが、そうでない場合はオプションです。
使用頻度の高い次の値は、ドロップダウンボックスからも設定できます。

- 順序制約を必須にする場合、矢印アイコンをクリックして Mandatory (必須) を選択します。
 - 順序制約を提案にとどめる場合は、矢印アイコンをクリックして Optional (オプション) を選択します。
 - Serialize (順番に処理): リソースに対して2つの停止/開始アクションが同時に実行されないようにするには、矢印アイコンをクリックして Serialize (順番に処理) を選択します。これにより、1つのリソースの開始が完了しないと他のリソースを開始できなくなります。通常は、起動時にホストに高い負荷をかけるリソースに使用します。
5. 順序の制約の場合、オプションシンメトリックは常に有効にしてください。これは、リソースを停止するときには逆順で行うという指定です。
 6. 制約のリソースを定義するには、次の手順に従います。
 - a. リソースカテゴリのドロップダウンボックスから、リソース(またはテンプレート)を選択します。
リソースが追加され、下に新しい空のドロップダウンボックスが表示されます。
 - b. この手順を繰り返してリソースを追加します。
リソースは、最初に最上位のリソース、次に2番目のリソースという順序で開始されます。通常、リソースを停止するときには逆順で行われます。
 - c. 順序制約内のリソースの順序を入れ替えるには、リソースの横にある上矢印アイコンをクリックして、その上のエントリと入れ替えます。
 7. 必要に応じて、各リソースの他のパラメータ(Started (開始)、Stopped (停止)、Master (マスタ)、Slave (スレーブ)、Promote (昇格)、Demote (降格) など)を指定します。リソースの横にある空のドロップダウンリストをクリックして、目的のエントリを選択します。
 8. 変更を確認して、設定を完了します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。



図 7.13: HAWK2 - 順序制約

7.6.4 制約のためにリソースセットを使用する

制約を定義するための別のフォーマットとして、**リソースセット**を使用することができます。これらは、**グループ**と同じ意味論に従います。

手順 7.17: 制約のためにリソースセットを使用する

1. 場所制約内でリソースセットを使用するには:
 - a. 手順7.14「場所制約の追加」の説明に従って操作を進めます。ただし、**ステップ 4**は除きます。1つのリソースを選択する代わりに、**Ctrl** または **Shift** を押しながらマウスをクリックして、複数のリソースを選択します。これにより、場所制約内でリソースセットが作成されます。
 - b. 場所制約からリソースを削除するには、**Ctrl** を押しながらリソースを再度クリックして、選択解除します。
2. コロケーションまたは順序の制約内でリソースセットを使用するには:
 - a. 手順7.15「コロケーション制約の追加」または手順7.16「順序制約の追加」の説明に従います。ただし、制約に対してリソースを定義する手順(**ステップ 5.a**または**ステップ 6.a**)は除きます。
 - b. 複数のリソースを追加します。

- c. リソースセットを作成するため、リソースの横にあるチェーンアイコンをクリックして、そのリソースを上のリソースにリンクします。リソースセットは、セットに属しているリソースの周囲のフレームによって示されます。
- d. リソースセット内の複数のリソースを結合したり、複数のリソースセットを作成したりできます。

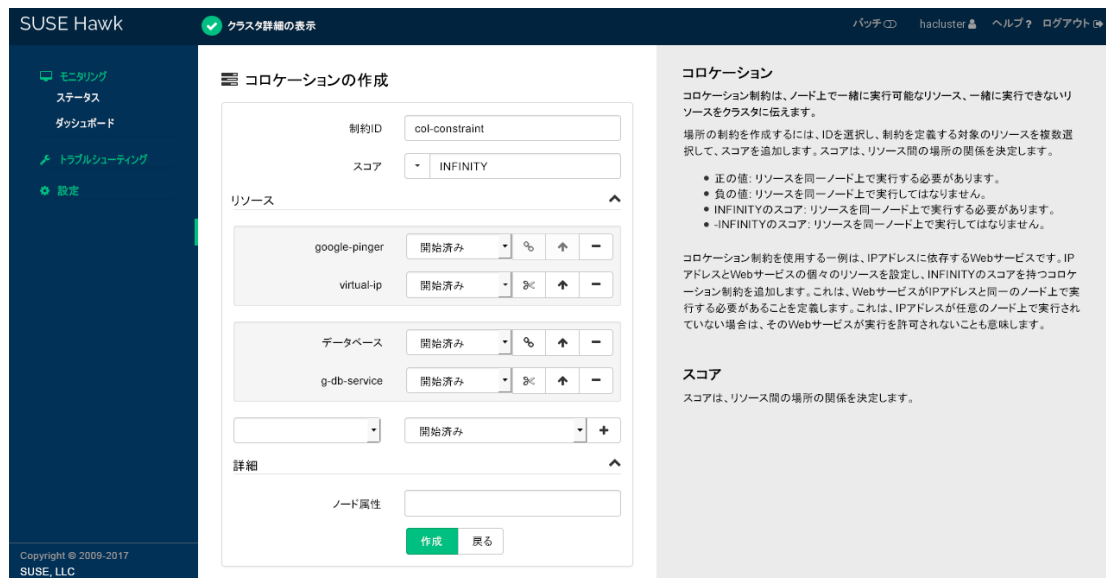


図 7.14: HAWK2 - コロケーション制約の2つのリソースセット

- e. 上のリソースからリソースをリンク解除するには、そのリソースの横にあるハサミアイコンをクリックします。

3. 変更を確認して、制約の設定を完了します。

7.6.5 詳細の参照先

制約の設定の詳細や、順序およびコロケーションの基本的な概念についての詳細なバックグラウンド情報は、<http://www.clusterlabs.org/pacemaker/doc/> で提供されているドキュメントを参照してください。

- 『Pacemaker Explained』の「Resource Constraints」の章
- 『Colocation Explained』
- 『オーダーの概要』

7.6.6 リソースフェールオーバーノードの指定

リソースに障害が発生すると、自動的に再起動されます。現在のノードで再起動できない場合、または現在のノードでN回失敗した場合は、別のノードへのフェールオーバーが試行されます。新しいノードへのマイグレートを行う基準(migration-threshold)となるリソースの失敗をいくつか定義できます。クラスタ内に3つ以上ノードがある場合、特定のリソースのフェールオーバー先のノードはHigh Availabilityソフトウェアにより選択されます。

リソースがフェールオーバーする特定のノードを前もって指定するには、次の手順に従います。

手順 7.18: フェールオーバーノードの指定

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 手順7.14「場所制約の追加」に記されている手順に従って、そのリソースの場所の制約を設定します。
3. 手順 7.7: リソースのパラメータ、操作、またはメタ属性の変更、ステップ 4に説明されている手順に従ってリソースにmigration-thresholdメタ属性を追加し、migration-thresholdの値を入力します。INFINITYではない正の値を指定する必要があります。
4. リソースの失敗回数を自動的に失効させる場合は、手順 7.5: プリミティブリソースの追加、ステップ 4に説明されている手順に従ってfailure-timeoutメタ属性をそのリソースに追加し、failure-timeoutの値を入力します。

SUSE Hawk クラスター詳細の表示

プリミティブの編集

リソースID: simple-testresource

クラス: ocf

プロバイダ: heartbeat

タイプ: ダミー

パラメータ

操作

メタ属性

migration-threshold: 1000

failure-timeout: 10

利用率

適用 元に戻す 戻る

プリミティブを作成するには、IDを定義し、クラス、(プロバイダ)、タイプなどのいくつかのパラメータを指定します。

プリミティブリソースの一例は `ocf:heartbeat:IPaddr2` で、クラスタのフローティングIPアドレスの設定に使用できます。

ステートレスリソースエージェントの例

これは、ダミーリソースエージェントです。実行中であるかどうかを追跡する以外は何も実行しません。その目的はテストを行うこととRAライターのテンプレートの役目を果たすることです。

NB: 下のアクションセクションで指定されたタイムアウトに注意してください。このタイムアウトは、エージェントが管理するリソースの種類にとって意味があるはずですが、最小タイムアウトであることが推奨されますが、すべての利用可能なリソースインスタンスをカバーできない場合があります。したがって、大きすぎる値にも、小さすぎる値にもせず、適切な値にしてください。最小タイムアウトは10秒を下回らないようにしてください。

タイプ

リソースエージェント名

5. リソースの初期設定として、追加のフェールオーバーノードを指定する場合は、追加の場所の制約を作成します。

マイグレーションしきい値と失敗カウントに関連したプロセスフローは、例6.8「マイグレーションしきい値 - プロセスフロー」に示されています。

リソースの失敗回数は、自動的に期限切れにする代わりに、いつでも、手動でクリーンアップすることもできます。詳細については、7.7.3項「リソースのクリーンアップ」を参照してください。

7.6.7 リソースフェールバックノードの指定(リソースの固着性)

ノードがオンライン状態に戻り、クラスタ内にある場合は、リソースが元のノードにフェールバックすることがあります。このことを防ぐ、またはリソースのフェールバック先として別のノードを指定するには、リソースの固着性の値を変更します。リソースの固着性は、リソースを作成するとき、または作成した後のどちらでも指定できます。

さまざまなリソース固着性値の意味については、6.5.5項「フェールバックノード」を参照してください。

手順 7.19: リソースの固着性の指定

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 手順 7.7: リソースのパラメータ、操作、またはメタ属性の変更、ステップ 4に従って、resource-stickinessメタ属性をリソースに追加します。
3. resource-stickinessとして、-INFINITYとINFINITYの間の値を指定します。



7.6.8 負荷インパクトに基づくリソース配置の設定

すべてのリソースが同等ではありません。Xenゲストなどの一部のリソースでは、そのホストであるノードがリソースの容量要件を満たす必要があります。配置したリソースの要件の合計が提供容量よりも大きくなった場合には、リソースのパフォーマンスが低下するか、または失敗します。

これを考慮に入れて、High Availability Extensionでは、次のパラメータを指定できます。

1. 一定のノードが「提供する」容量
2. 一定のリソースが「要求する」容量
3. リソースの配置に関する全体的なストラテジ

詳細と設定例については、6.5.6項「負荷インパクトに基づくリソースの配置」を参照してください。

使用属性は、リソースの要件と、ノードが提供する容量の両方を設定するために使用されます。リソースが要求する容量を設定するには、その前にノードの容量を設定する必要があります。

手順 7.20: ノードが提供する容量の設定

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、監視機能 > 状態 の順に選択します。
3. ノードタブで、容量を設定するノードを選択します。
4. 操作列で、下矢印アイコンをクリックして編集を選択します。
ノードの編集画面が開きます。
5. 使用率の下で、使用属性の名前を空のドロップダウンボックスに入力します。
名前は任意です(RAM_in_GBなど)。
6. 属性を追加するには、追加アイコンをクリックします。
7. 属性の隣の空のテキストボックスに、属性値を入力します。値は整数にする必要があります。



8. 必要なだけ使用属性を追加し、これらの属性すべての値を追加します。
9. 変更内容を確認します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。

手順 7.21: リソースが要求する容量の設定

プリミティブリソースを作成するときや、既存のプリミティブリソースを編集するとき、特定のリソースがノードに要求する容量を設定します。

リソースに使用属性を追加する前に、[手順 7.20](#)で説明するように、クラスタノードの使用属性を設定しておく必要があります。

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 既存のリソースに使用属性を追加する場合: 7.7.1項「リソースとグループの編集」に示すように、Manage (管理) > 状態の順に移動して、[リソース設定] ダイアログを開きます。
新しいリソースを作成する場合: 7.5.3項「単純なリソースの追加」に示すように、Configuration (設定) > Add Resource (リソースの追加)の順に移動して進みます。
3. [リソース設定] ダイアログで、使用率カテゴリに移動します。
4. 空のドロップダウンボックスから、手順 7.20でノードに対して設定した使用属性のいずれかを選択します。
5. 属性の隣の空のテキストボックスに、属性値を入力します。値は整数にする必要があります。
6. 必要なだけ使用属性を追加し、これらの属性すべての値を追加します。
7. 変更内容を確認します。画面上部に、アクションが成功したかどうかを示すメッセージが表示されます。

ノードが提供する容量とリソースが要求する容量を設定してから、配置ストラテジをグローバルクラスタオプションに設定します。そうしないと、容量設定は有効になりません。負荷のスケジュールに使用できるストラテジがいくつかあります。たとえば、負荷をできるだけ少ない数のノードに集中したり、使用可能なすべてのノードに均等に分散できます。詳細については、6.5.6項「負荷インパクトに基づくリソースの配置」を参照してください。

手順 7.22: 配置ストラテジの設定

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > クラスタ設定を選択し、各画面を開きます。グローバルクラスタオプション、およびリソースと操作のデフォルトが表示されます。
3. 画面上部にある空のドロップダウンボックスから、placement-strategyを選択します。
デフォルトでは、その値はデフォルトに設定され、使用属性と値が考慮されていないことを意味します。
4. 要件に応じて、配置ストラテジを適切な値に設定します。

- 5. 変更内容を確認します。

7.7 クラスタリソースの管理

Hawk2では、クラスタリソースを設定することに加えて、状態画面で既存のリソースを管理することができます。この画面の概要については、7.8.1項「単一クラスタの監視」を参照してください。

7.7.1 リソースとグループの編集

既存のリソースを編集する必要がある場合は、状態画面に移動します。操作列で、変更したいリソースまたはグループの横にある下矢印アイコンをクリックして編集を選択します。

編集画面が表示されます。プリミティブリソースを編集する場合は、次の操作が使用できます。

プリミティブの操作

- リソースのコピー。
- リソースの名前変更 (そのIDの変更)。
- リソースの削除。

グループを編集する場合は、次の操作が使用できます。

グループの操作

- このグループに追加される新しいプリミティブの作成。
- グループの名前変更 (そのIDの変更)。
- グループメンバーを再ソートするには、右側の「ハンドル」アイコンを使用して、メンバーを目的の順序にドラッグアンドドロップします。

7.7.2 リソースの開始

クラスタリソースは、起動する前に、正しく設定されているようにします。たとえば、Apacheサーバをクラスタリソースとして使用する場合は、まず、Apacheサーバを設定します。クラスタでそれぞれのリソースを開始する前に、Apache設定を完了します。



注記: クラスタによって管理されるサービスには介入しないでください。クラスタによって管理されるサービスは操作しないでください。

High Availability Extensionでリソースを管理しているときに、リソースを他の方法(クラスタ外で、たとえば、手動、ブート、再起動など)で開始したり、停止したりしてはなりません。High Availability Extensionソフトウェアが、すべてのサービスの開始または停止アクションを実行します。

ただし、サービスが適切に構成されているか確認したい場合は手動で開始しますが、High Availabilityが起動する前に停止してください。

現在クラスタで管理されているリソースへの介入については、まず、リソースを `maintenance mode` に設定します。詳細については、[手順17.5「リソースをHawk2を使用して保守モードにする」](#)を参照してください。

Hawk2でリソースを作成するときには、`target-role`メタ属性でその初期状態を設定することができます。その値を `stopped` に設定した場合、リソースは、作成後、自動的に開始することはありません。

手順 7.23: 新しいリソースの起動

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、監視機能 > 状態の順に選択します。Resources (リソース)のリストには状態も表示されます。
3. 起動するリソースを選択します。その操作列で、開始アイコンをクリックします。継続するには、表示されるメッセージに対して確認します。
リソースが開始すると、Hawk2はすぐにリソースの状態を緑に変え、それがどのノードで実行されているかを表示します。

7.7.3 リソースのクリーンアップ

リソースは、失敗した場合は自動的に再起動しますが、失敗のたびにリソースの失敗回数が増加します。

リソースに対して `migration-threshold` が設定されていた場合、失敗回数が移行しきい値に達すると、そのリソースはそのノードでは実行されなくなります。

リソースの失敗回数は、(リソースに`failure-timeout`オプションを設定することにより)自動的にリセットするか、または次に示すように手動でリセットできます。

手順 7.24: リソースのクリーンアップ

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、状態を選択します。Resources (リソース)のリストには状態も表示されます。
3. クリーンアップするリソースに移動します。操作列で、下矢印ボタンをクリックしてCleanup (クリーンアップ)を選択します。継続するには、表示されるメッセージに対して確認します。
これにより、コマンド`crm resource cleanup`が実行され、すべてのノードでリソースがクリーンアップされます。

7.7.4 クラスタリソースの削除

リソースをクラスタから削除する必要がある場合は、次の手順に従って、設定エラーが発生しないようにします。

手順 7.25: クラスタリソースの削除

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 手順7.24「リソースのクリーンアップ」の説明に従って、すべてのノードでリソースをクリーンアップします。
3. リソースを停止します。
 - a. 左のナビゲーションバーから、監視機能 > 状態の順に選択します。Resources (リソース)のリストには状態も表示されます。
 - b. 操作列で、リソースの横にあるStop (停止)ボタンをクリックします。
 - c. 継続するには、表示されるメッセージに対して確認します。
リソースが停止すると、状態列に変更が反映されます。
4. リソースを削除します。

- a. 左のナビゲーションバーから、環境設定 > 設定の編集の順に選択します。
- b. Resources (リソース)のリストで、それぞれのリソースに移動します。操作列で、リソースの横にあるDelete (削除)アイコンをクリックします。
- c. 継続するには、表示されるメッセージに対して確認します。

7.7.5 クラスタリソースの移行

7.6.6項「リソースフェールオーバーノードの指定」で説明したように、ソフトウェアまたはハードウェアの障害時には、クラスタは定義可能な特定のパラメータ(たとえばマイグレーションしきい値やリソースの固着性など)に従って、リソースを自動的にフェールオーバー(マイグレート)させます。クラスタ内の別のノードにリソースを手動で移行させることもできます。または、リソースを現在のノードから出すかはユーザが判断し、どこに移動するかはクラスタに判断させます。

手順 7.26: リソースを手動で移行する

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、監視機能 > 状態の順に選択します。Resources (リソース)のリストには状態も表示されます。

3. Resources (リソース)のリストで、それぞれのリソースを選択します。

4. 操作列で、下矢印ボタンをクリックしてMigrate (移行)を選択します。

5. 開くウィンドウには、次の選択肢があります。

- Away from current node (現在のノードから離れる): これによって現在のノードに対して -INFINITY スコアによる場所の制約が作成されます。
- または、別のノードにリソースを移動できます。これによって移動先ノードに対して INFINITY スコアの場所の制約が作成されます。

6. 選択内容を確認します。

リソースを再び元に戻すには、次の手順に従います。

手順 7.27: リソースの移行解除

1. Hawk2にログインします。

2. 左のナビゲーションバーから、監視機能 > 状態の順に選択します。Resources (リソース) のリストには状態も表示されます。
3. Resources (リソース) のリストで、それぞれのリソースに移動します。
4. 操作列で、下矢印ボタンをクリックして消去を選択します。継続するには、表示されるメッセージに対して確認します。
これによって、`crm_resource --clear` コマンドが使用されます。リソースは元の場所に戻ることができます。あるいは現在の場所に残ることもあります(リソースの固着性によって)。

詳細については、<http://www.clusterlabs.org/pacemaker/doc/> から入手できる『Pacemaker Explained』を参照してください。特に、「Resource Migration」のセクションを参照してください。

7.8 クラスタの監視

Hawk2には、単一のクラスタおよび複数のクラスタを監視するための異なる画面があります：状態画面とダッシュボード画面。

7.8.1 単一クラスタの監視

単一クラスタを監視するには、状態画面を使用します。Hawk2にログインした後で、状態画面がデフォルトで表示されます。右上隅のアイコンに、クラスタの状態の概要が表示されます。詳細情報を参照する場合は、次のカテゴリを確認します。

エラー

エラーが発生した場合、ページの上部に表示されます。

リソース

設定されているリソースが表示されます。その状態、名前(ID)、場所(リソースが実行されているノード)、リソースエージェントのタイプが含まれます。操作列で、リソースを開始または停止するか、いくつかのアクションをトリガーするか、詳を表示できます。トリガ可能なアクションには、保守モードへのリソースの設定(または保守モードの削除)、リソースの別のノードへの移行、リソースのクリーンアップ、リソースイベントの表示、またはリソースの編集が含まれます。

ノード

ログイン先のクラスタサイトに属するノードが表示されます。ノードの状態および名前が含まれます。Maintenance (保守)およびStandby (スタンバイ)列で、maintenanceまたはstandbyフラグを設定または解除できます。操作列で、ノードの最新イベントや詳細情報を参照できます。たとえば、それぞれのノードにutilization、standby、またはmaintenanceのどの属性が設定されているかを参照できます。

チケット

Geoクラスタリングでの使用向けにチケットを設定した場合にのみ表示されます。

ステータス	名前	場所	タイプ	操作
+	apache		ocf:heartbeat:Dummy	[icon] [icon] [icon]
+	データベース	kemter-3	ocf:heartbeat:Dummy	[icon] [icon] [icon]
+	dml-control	kemter-3	ocf:heartbeat:Dummy	[icon] [icon] [icon]
+	g-db-service		グループ(2)	[icon] [icon] [icon]
+	google-pinger		ocf:heartbeat:Dummy	[icon] [icon] [icon]
+	nfs		ocf:heartbeat:nfsserver	[icon] [icon] [icon]
+	stonith		stonith:external/ipmi	[icon] [icon] [icon]
+	virtual-ip		ocf:heartbeat:Dummy	[icon] [icon] [icon]

図 7.15: HAWK2 - クラスタの状態

7.8.2 複数のクラスタの監視

複数のクラスタを監視するには、Hawk2 Dashboard (ダッシュボード)を使用します。ダッシュボード画面に表示されるクラスタ情報は、サーバ側に保管されています。これらは、クラスタノード間で同期が取られています(クラスタノード間にパスワード不要のSSHアクセスが設定されている場合)。詳細については、[D.2項「パスワード不要のSSHアカウントの設定」](#)を参照してください。ただし、Hawk2を実行するマシンは、その目的のためにクラスタの一部である必要はなく、別個の無関係のシステムで構いません。

Hawk2で複数のクラスタを監視するには、一般的な[Hawk2の要件](#)に加え、次の前提条件も満たす必要があります。

前提条件

- Hawk3のダッシュボードで監視するすべてのクラスタでは、SUSE Linux Enterprise High Availability Extension 15 SP2を実行している必要があります。
- すべてのクラスタノードにあるHawk2の自己署名証明書を独自の証明書(または公式認証局によって署名された証明書)で置き換えていない場合は、「すべての」クラスタの「すべての」ノードで、少なくとも1回はHawk2にログインします。証明書を検証します(または、ブラウザで例外を追加して警告をスキップします)。そうしない場合、Hawk2はクラスタに接続できません。

手順 7.28: ダッシュボードを使用した複数クラスタの監視

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、監視機能 > ダッシュボード の順に選択します。
Hawk2は、現在のクラスタサイトのリソースおよびノードに関する概要を表示します。また、Geoクラスタでの使用向けに設定されたチケットを表示します。このビューで使用されているアイコンについての情報が必要な場合は、凡例をクリックします。リソースIDを検索するには、検索テキストボックスに名前(ID)を入力します。特定のノードのみを表示するには、フィルタアイコンをクリックしてフィルタリングオプションを選択します。

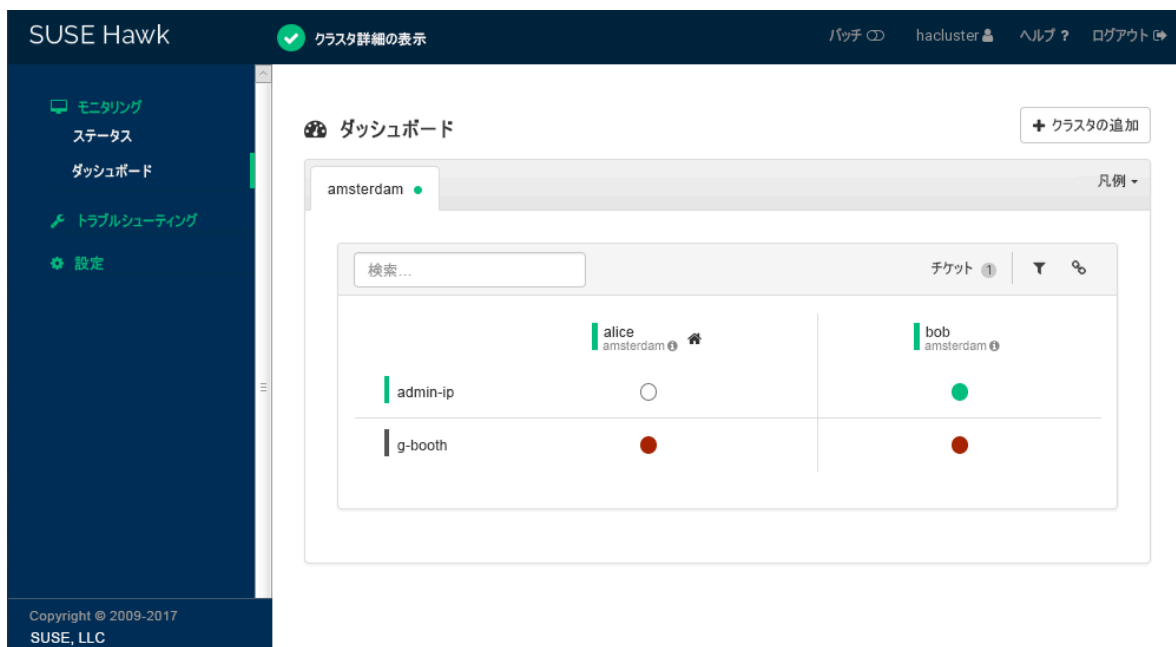


図 7.16: 最初のクラスタサイト1つのクラスタサイト(amsterdam)を表示するHAWK2ダッシュボード

3. 複数のクラスタにダッシュボードを追加するには、以下の操作を行います。

- a. クラスタの追加をクリックします。
- b. ダッシュボードでクラスタを識別するためのクラスタ名をクラスタ名に入力します。たとえば、berlinです。
- c. いずれかのノードの完全修飾ホスト名を、2つ目のクラスタに入力します。たとえば、charlieです。



- d. 追加をクリックします。新たに追加されたクラスタサイトに対して、Hawk2は2つ目のタブにノードやリソースの概要を表示します。



注記: 接続エラー

パスワードを入力してノードにログインすることを求めるプロンプトが表示された場合、このノードにはまだ接続したことがなく、自己署名証明書を置き換えていない状態です。そのような場合は、パスワードを入力しても、接続は次のメッセージを表示して失敗します。サーバへの接続エラーです。5秒ごとに再試行します...

続行するには、[自己署名証明書の置き換え置換](#)を参照してください。

4. クラスタサイトやその管理に関する詳細を参照するには、サイトのタブに切り替えてチェーンアイコンをクリックします。

Hawk2はこのサイトのステータスビューを新しいブラウザウィンドウかタブに表示します。この部分のGeoクラスタをそこから管理できます。

5. ダッシュボードからクラスタを削除するには、クラスタの詳細の右側にあるxアイコンをクリックします。

7.9 バッチモードの使用

Hawk2は、「クラスタシミュレータ」を含むバッチモードを提供します。これは次の操作に使用できます。

- 各変更を直ちに反映させるのではなく、クラスタに変更をステージングして、それらの変更を単一ランザクションとして適用する。
- たとえば、潜在的な障害シナリオを調べるため、変更やクラスタイベントをシミュレートする。

たとえば、互いに依存するリソースのグループを作成する場合にバッチモードを使用できます。バッチモードを使用すると、クラスタに中間的なまたは不完全な設定を適用することを回避できます。

バッチモードが有効な間は、リソースや制約を追加したり編集したり、クラスタ設定を変更できます。ノードのオンラインまたはオフライン化、リソース操作およびチケットの許可または取り消しなど、クラスタのイベントをシミュレートすることもできます。詳細については[手順7.30「ノード、リソース、またはチケットイベントの挿入」](#)を参照してください。

「クラスタシミュレータ」は、すべての変更後に自動的に実行され、ユーザインタフェースに予想される結果を表示します。たとえば、次のような場合もあります。バッチモードの最中にリソースを止めると、実際リソースはまだ実行中であるにも関わらず、ユーザインタフェースにはリソースが停止したと表示されます。

！ 重要: ウィザード、およびライブシステムへの変更

一部のウィザードには単なるクラスタ設定を超えるアクションが含まれます。これらのウィザードをバッチモードで使用する場合、クラスタ設定を超えるすべての変更がライブシステムに直ちに適用されます。

したがって、root許可が必要なウィザードはバッチモードでは実行できません。

手順 7.29: バッチモードの使用

1. Hawk2にログインします。

2. バッチモードを有効にするには、最上位の行からバッチを選択します。
最上位の行の下に追加のバーが表示されます。これは、バッチモードがアクティブであることを示し、バッチモードで実行可能なアクションへのリンクが含まれます。



図 7.17: HAWK2バッチモードが有効

3. バッチモードがアクティブなときに、リソースや制約の追加または編集、あるいはクラスタ設定の編集など、クラスタに対する変更を実行します。
変更はシミュレートされ、すべての画面に表示されます。
4. 行った変更の詳細を表示するには、バッチモードバーから表示を選択します。バッチモードウィンドウが開きます。
設定の変更について、ライブ状態とシミュレートされた変更間の相違がcrmsd構文で表示されます。_文字で開始される行は、現在の状態を表し、+で開始される行は、提案される状態を示します。
5. イベントを注入したり、さらに詳細を表示する場合は、手順 7.30を参照してください。
または、Close (閉じる)をクリックしてウィンドウを閉じます。
6. シミュレートした変更を破棄または適用するかのいずれかを選択し、選択内容を確認します。これによりバッチモードが無効になり、通常のモードに戻ります。

バッチモードで実行中に、Hawk2ではNode Events (ノードイベント)とResource Events (リソースイベント)を注入することもできます。

ノードイベント

ノードの状態を変更できます。使用可能な状態は、online (オンライン)、offline (オフライン)、およびunclean (アンクリーン)です。

リソースイベント

リソースの一部のプロパティを変更できます。たとえば、操作 (start、stop、monitorなど)、その操作を適用するノード、およびシミュレートされる予想される結果を設定できます。

チケットイベント

(Geoクラスタにおける)チケットの許可と取り消しの影響をテストできます。

手順 7.30: ノード、リソース、またはチケットイベントの挿入

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. バッチモードがまだアクティブでない場合、最上位の行でバッチをクリックして、バッチモードに切り替えます。
3. バッチモードバーで、表示をクリックして、バッチモードウィンドウを開きます。
4. ノードのステータスの変更をシミュレートするには
 - a. 注入 > ノードイベントを順にクリックします。
 - b. 操作するノードを選択し、そのターゲット状態を選択します。
 - c. 変更内容を確認します。イベントはバッチモードダイアログに一覧表示されるイベントのキューに追加されます。
5. リソースの操作をシミュレートするには
 - a. 注入 > リソースイベントを順にクリックします。
 - b. 操作するリソースを選択し、シミュレートする操作を選択します。
 - c. 必要に応じて、間隔を定義します。
 - d. 操作を実行するノードを選択し、ターゲットとする結果を選択します。イベントはバッチモードダイアログに一覧表示されるイベントのキューに追加されます。
 - e. 変更内容を確認します。

6. チケットアクションをシミュレートするには

- 挿入>チケットイベントを順にクリックします。
- 操作するチケットを選択し、シミュレートするアクションを選択します。
- 変更内容を確認します。イベントはバッチモードダイアログに一覧表示されるイベントのキューに追加されます。

7. バッチモードダイアログ(図 7.18)で、注入されたイベントごとに新しい行が表示されます。ここに一覧表示されるイベントは、直ちにシミュレートされ、状態画面に反映されます。

設定の変更も行った場合は、ライブ状態とシミュレートされた変更の間の相違が注入されたイベントの下に表示されます。



図 7.18: HAWK2のバッチモード - 注入されたイベントと設定の変更

- 注入されたイベントを削除するには、その隣の削除アイコンをクリックします。Hawk2では、それによって状態画面が更新されます。
- 実行されたシミュレーションに関する詳細を表示するには、シミュレータをクリックして、次のいずれかを選択します。

まとめ

詳細な概要を表示します。

CIB (in)/CIB (out)

CIB (in)では、初期のCIB状態を示します。CIB (out)では、遷移後のCIBの状態を示します。

遷移グラフ

遷移のグラフィカルな表現を示します。

遷移

遷移のXML表示を示します。

10. シミュレートされた変更を確認したら、バッチモードウィンドウを閉じます。
11. バッチモードを終了するには、シミュレートされた変更を適用するか、破棄するか of どれかを選択します。

7.10 クラスタ履歴の表示

Hawk2には、クラスタの過去のイベントを表示する、次のような機能があります(いくつかの詳細さのレベルがあります)。

- 7.10.1項 「ノードまたリソースの最近のイベントの表示」
- 7.10.2項 「クラスタレポートのための履歴エクスプローラーの使用」
- 7.10.3項 「履歴エクスプローラーの遷移詳細の表示」

7.10.1 ノードまたリソースの最近のイベントの表示

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、監視機能 > 状態 の順に選択します。Resources (リソース)とNodes (ノード)が一覧表示されます。
3. リソースの最近のイベントを表示するには
 - a. Resources (リソース)をクリックして、それぞれのリソースを選択します。
 - b. リソースの操作列で、下矢印ボタンをクリックして最近のイベントを選択します。Hawk2では、新しいウィンドウが開き、最新のイベントのテーブルビューが表示されます。
4. ノードの最近のイベントを表示するには
 - a. Nodes (ノード)をクリックして、それぞれのノードを選択します。
 - b. ノードの操作列で、最近のイベントを選択します。

Hawk2では、新しいウィンドウが開き、最新のイベントのテーブルビューが表示されます。

🔍 最近のイベント: alice

RC	リソース	操作	前回変更日	状態	コール	実行	完了
0	dummy1	dummy1_start_0	2016年10月25日 火曜日 18:10:49	開始しました	18	20ms	✓
0	dummy1	dummy1_monitor_10000	2016年10月25日 火曜日 18:10:49	開始しました	19	26ms	✓
0	dummy2	dummy2_stop_0	2016年10月25日 火曜日 18:10:49	停止 (無効)	15	23ms	✓
0	dummy2	dummy2_monitor_10000	2016年10月25日 火曜日 18:10:49	停止 (無効)	13	19ms	✓

7.10.2 クラスタレポートのための履歴エクスプローラーの使用

左のナビゲーションバーからトラブルシューティング > 履歴を選択して、履歴エクスプローラーにアクセスします。履歴エクスプローラーでは、詳細なクラスタレポートを作成し、遷移情報を表示できます。次のオプションが表示されます。

生成

特定の時刻のクラスタレポートを作成します。Hawk2では、`crm report` コマンドをコールして、レポートを生成します。

アップロード

crmシェルで直接作成したか、異なるクラスタ上で作成した`crm report`アーカイブをアップロードできます。

レポートを生成するか、アップロードした後で、これらのレポートはレポートの下に表示されます。レポートのリストから、レポートの詳細を表示したり、レポートをダウンロードしたり、削除したりできます。



図 7.19: HAWK2 - 履歴エクスプローラーのメインビュー

手順 7.31: クラスタレポートの生成またはアップロード

1. Hawk2にログインします。

`https://HAWKSERVER:7630/`

2. 左のナビゲーションバーから、トラブルシューティング > 履歴の順に選択します。
生成ビューに履歴エクスプローラー画面が開きます。デフォルトでは、レポートの提案される時間フレームは最後の時刻です。
3. クラスタレポートを作成するには
 - a. レポートを直ちに開始するには、生成をクリックします。
 - b. レポートの時間フレームを変更するには、提案される時間フレームの任意の場所をクリックし、ドロップダウンボックスから別のオプションを選択します。Custom (カスタム)開始日、終了日、および時間をそれぞれ入力することもできます。レポートを開始するには、生成をクリックします。
レポートを終了した後で、そのレポートがレポートの下に表示されます。
4. クラスタレポートをアップロードするには、Hawk2でアクセス可能なファイルシステム上に`crm report`アーカイブがある必要があります。以下に手順を示します。
 - a. アップロードタブに切り替えます。
 - b. クラスタレポートアーカイブをブラウズし、アップロードをクリックします。

レポートがアップロードされると、そのレポートがレポートの下に表示されます。

5. レポートをダウンロードまたは削除するには、操作列のレポートの横の各アイコンをクリックします。
6. 履歴エクスプローラーのレポート詳細を表示するには、レポートの名前をクリックするか、操作列から表示を選択します。

SUSE Hawk

クラスタ詳細の表示

履歴エクスプローラー

レポート名: hawk_2017_08_14T09_21_34_00_00_2017_08_14T15_21_34_00_00

開始: 2017-08-14 09:23:19 UTC

終了: 2017-08-14 15:21:31 UTC

遷移: 18

ノードイベント

リソースイベント

```
2017-08-14T11:23:19.465929+02:00 kemter-2 lrmd[1892]: notice: executing - rsc:nfs action:stop call_id:113
2017-08-14T11:23:19.907638+02:00 kemter-2 nfsserver(nfs)[2441]: INFO: Stopping NFS server ...
2017-08-14T11:23:19.942108+02:00 kemter-2 nfsserver(nfs)[2441]: INFO: Stop: threads
2017-08-14T11:23:19.460195+02:00 kemter-3 crmd[1894]: notice: Initiating stop operation nfs_stop_0 on kemter-2
2017-08-14T11:23:19.858297+02:00 kemter-2 nfsserver(nfs)[2441]: INFO: Stop: rpc-statd
2017-08-14T11:23:19.978677+02:00 kemter-2 nfsserver(nfs)[2441]: INFO: Stop: nfs-idmapd
2017-08-14T11:23:20.076367+02:00 kemter-3 crmd[1894]: notice: Initiating start operation nfs_start_0 locally on kemter-3
2017-08-14T11:23:20.133732+02:00 kemter-3 lrmd[1891]: notice: executing - rsc:nfs action:start call_id:94
2017-08-14T11:23:20.011716+02:00 kemter-2 nfsserver(nfs)[2441]: INFO: Stop: nfs-mountd
2017-08-14T11:23:20.549624+02:00 kemter-2 nfsserver(nfs)[2441]: ERROR: rpcbind is not running
```

7. レポートボタンをクリックして、レポートのリストに戻ります。

履歴エクスプローラーのレポート詳細

- レポートの名前です。
- レポートの開始時刻。
- レポートの終了時刻。
- レポートによってカバーされるクラスタのすべての遷移の遷移数およびタイムライン。遷移の詳細を表示する方法については、[7.10.3項](#)を参照してください。
- ノードイベント。
- リソースイベント。

7.10.3 履歴エクスプローラーの遷移詳細の表示

各遷移ごとに、クラスタは状態のコピーを保存し、これを `pacemaker-schedulerd` への入力として提供します。このアーカイブがログ記録されるパス。すべての `pe-*` ファイルが DC (指定コーディネータ) 上に生成されます。DC はクラスタ内で変更可能なため、いくつかのノードからの `pe-*` ファイルがある場合があります。すべての `pe-*` ファイルは CIB の保存済みスナップショットであり、`pacemaker-schedulerd` による計算の入力として利用されます。

Hawk2 では、各 `pe-*` ファイルの名前と作成時刻、およびそれが作成されたノードを表示できます。また、履歴エクスプローラーでは、それぞれの `pe-*` ファイルに基づいて、次の詳細をビジュアル化できます。

履歴エクスプローラーでの遷移詳細

詳細

遷移に属するログインデータのスニペットを表示します。次のコマンドの出力を表示します(リソースエージェントのログメッセージを含む)。

```
crm history transition peinput
```

設定

`pe-*` ファイルが作成された時刻でクラスタ設定を表示します。

差分

選択した `pe-*` ファイルと次のファイル間の設定とステータスの相違を表示します。

ログ

遷移に属するログインデータのスニペットを表示します。次のコマンドの出力を表示します。

```
crm history transition log peinput
```

この出力には、`pacemaker-schedulerd`、`pacemaker-controld`、`pacemaker-execd` などのデーモンの詳細が示されています。

グラフ

遷移のグラフィカルな表現を示します。グラフをクリックすると、(`pacemaker-schedulerd` で実行したのと同じ) 計算がシミュレートされ、図表化されます。

手順 7.32: 遷移詳細の表示

1. Hawk2 にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、トラブルシューティング > 履歴の順に選択します。
レポートがすでに生成またはアップロードされている場合は、これらのレポートがレポートのリストに表示されます。そうでない場合は、[手順 7.31](#)で説明されるように、レポートを生成またはアップロードします。
3. レポート名をクリックするか、操作列から表示を選択して、[履歴エクスプローラーのレポート詳細](#)を開きます。
4. 遷移詳細にアクセスするには、下に示す遷移タイムラインの遷移ポイントを選択する必要があります。Previous (前へ)およびNext (次へ)アイコンをおよびZoom In (拡大)およびZoom Out (縮小)アイコンを使用して、興味ある遷移を探します。
5. `pe-input*`ファイルの名前、それが作成された時刻およびノードを表示するには、タイムラインの遷移ポイント上でマウスポインタを合わせます。
6. [履歴エクスプローラーでの遷移詳細](#)を表示するには、さらに知りたい遷移ポイントをクリックします。
7. 詳細、環境設定、差分、Logs (ログ)またはGraph (グラフ)を表示するには、[履歴エクスプローラーでの遷移詳細](#)で説明されるように、コンテンツを表示するには、各ボタンをクリックします。
8. レポートのリストに戻るには、レポートボタンをクリックします。

7.11 クラスタヘルスの確認

Hawk2は、クラスタの検査と問題点の検出を行うウィザードを提供します。分析が完了すると、Hawk2は詳細なクラスタレポートを作成します。Hawk2によるクラスタヘルスの確認とレポートの生成には、ノード間のパスワード不要のSSHアクセスが必要です。ない場合は、現在のノードのデータのみを収集します。`ha-cluster-bootstrap`パッケージが提供するブートストラップスクリプトでクラスタを設定した場合、パスワード不要のSSHアクセスは既に設定されています。手動で設定する必要がある場合は、[D.2項「パスワード不要のSSHアカウントの設定」](#)を参照してください。

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーから、環境設定 > ウィザード の順に選択します。
3. 基本的カテゴリを展開します。

4. Verify health and configuration (ヘルスと設定の検証)ウィザードを選択します。
5. 検証を選択して確認します。
6. クラスタのルートパスワードを入力して、適用をクリックします。Hawk2がレポートを生成します。

8 クラスタリソースの設定と管理(コマンドライン)

クラスタリソースを設定および管理するには、コマンドラインユーティリティであるcrmシェル(crmsh)またはWebベースのユーザインタフェースであるHawk2のいずれかを使用します。

この章では、**crm**コマンドラインツールを紹介し、このツールの概要、テンプレートの使用方法、そして、主にクラスタリソースの設定と管理(基本的なリソースと高度なリソース(グループとクローン)の作成、制約の設定、フェールオーバーノードとフェールバックノードの指定、リソース監視の設定、リソースの開始、クリーンアップ、または削除、および手動によるリソースの移行について説明します。



注記: ユーザの権限

クラスタを管理するには十分な権限が必要です。**crm**コマンドおよびそのサブコマンドは、**root**ユーザとして、またはCRM所有者ユーザとして実行される必要があります(通常はhaclusterユーザ)。

ただし、**user**オプションを使用することで、**crm**とそのサブコマンドを一般(権限のない)ユーザとして実行し、必要な場合はいつでも**sudo**を使用してIDを変更できます。たとえば、次の**crm**コマンドは、権限のあるユーザIDとしてhaclusterを使用します。

```
root # crm options user hacluster
```

/etc/sudoersを設定しておいて、**sudo**がパスワードを要求しないようにしておく必要があることに注意してください。

8.1 crmsh - 概要

crmコマンドには、リソース、CIB、ノード、リソースエージェントなどを管理するサブコマンドがあります。このコマンドには、例を組み込んだ詳細なヘルプシステムが用意されています。すべての例は、[付録B](#)で説明される命名規則に従います。



ヒント: 対話型crmプロンプト

crmを引数なしで(または1つのサブレベルのみを引数として)使用することにより、**crm**シェルは対話モードになります。このモードは、次のプロンプトで示されます。

```
crm(live/HOSTNAME)
```

読みやすくするために、このマニュアルでは対話型crmのプロンプトでホスト名を省略します。次の例のように、aliceなどの特定のノードで対話型シェルを実行する必要がある場合にのみホスト名を含めます。

```
crm(live/alice)
```

8.1.1 ヘルプの表示

ヘルプには複数の方法でアクセスできます。

- **crm**とそのコマンドラインオプションの使用方法を出力するには:

```
root # crm --help
```

- 使用可能なすべてのコマンドの一覧を表示するには:

```
root # crm help
```

- コマンドの参照情報だけでなく、他のヘルプセクションにアクセスするには:

```
root # crm help topics
```

- **configure**サブコマンドの詳細なヘルプテキストを表示するには:

```
root # crm configure help
```

- **configure**の**group**サブコマンドの構文、使用方法、例を印刷するには:

```
root # crm configure help group
```

これも同様です:

```
root # crm help configure group
```

helpサブコマンド(`--help`オプションと混同しないこと)のほとんどすべての出力によって、テキストビューアが開きます。このテキストビューアは上下にスクロール可能で、ヘルプテキストが読みやすくなっています。テキストビューアを閉じるには、< **q** >キーを押します。



ヒント: バッシュおよび対話型シェルでタブ補完機能を使用

crmshは、対話型シェルに対してだけでなく、バッシュでの直接的で完全なタブ補完機能をサポートしています。たとえば、「`crm help config` **<Tab>**」と入力してを押すと、対話型シェルと同様に単語が補完されます。

8.1.2 crmshのサブコマンドの実行

`crm`コマンドそのものは、次のように使用できます。

- **直接:** すべてのサブコマンドを`crm`に続け、**< Enter >**を押すと、ただちにその出力が表示されます。たとえば、`crm help ra`を入力すると、`ra`サブコマンド(リソースエージェント)に関する情報を取得できます。

サブコマンドは、その短縮形が固有である限り短縮できます。たとえば、`status`を`st`と短縮しても、crmshにはユーザが意図したサブコマンドとして認識されます。

パラメータを短縮する機能もあります。通常、パラメータは`params`キーワードを使用して追加します。`params`セクションが最初のセクションでほかにセクションがない場合、このセクションを省略できます。たとえば、次のような行があるとします。

```
root # crm primitive ipaddr ocf:heartbeat:IPaddr2 params ip=192.168.0.55
```

これは次の行と同等です。

```
root # crm primitive ipaddr ocf:heartbeat:IPaddr2 ip=192.168.0.55
```

- **crmシェルスクリプトとして使用:** Crmシェルスクリプトには`crm`のサブコマンドが含まれます。詳細については、8.1.4項「[crmshのシェルスクリプトの使用](#)」を参照してください。
- **crmshクラスタスクリプトとして使用:** これらは、メタデータ、RPMパッケージへの参照、設定ファイル、およびcrmshサブコマンドを1つのわかりやすい名前でバンドルしてまとめたものです。`crm script`コマンドを使用して管理します。

これらをcrmshシェルスクリプトと混同しないでください。両方に共通する目的はいくつかありますが、crmシェルスクリプトにはサブコマンドのみが含まれるのに対し、クラスタスクリプトにはコマンドの単純なエミュレーション以上の処理が組み込まれています。詳細については、[8.1.5項「crmshのクラスタスクリプトの使用」](#)を参照してください。

- **内部シェルとして対話式に使用:** 「crm」とタイプして、内部シェルに入ります。プロンプトが`crm(live)`に変化します。helpを使用すると、利用可能なサブコマンドの概要を取得できます。内部シェルにはさまざまなサブコマンドレベルがあり、1つのサブコマンドをタイプして **Enter** を押すことで、そのサブコマンドのレベルに「入る」ことができます。

たとえば、「resource」とタイプすると、リソース管理レベルに入ります。プロンプトは`crm(live)resource#`に変わります。内部シェルを終了するには、コマンド **quit**、**bye**、または**exit**を使用します。1レベル戻る場合は、**back**、**up**、**end**、または**cd**を使用します。

crm、そしてオプションを付けずにサブコマンドを入力して **Enter** を押すと、そのレベルに直接入ることができます。

内部シェルは、サブコマンドとリソースのタブによる完了もサポートします。コマンドの冒頭をタイプして **<Tab>** を押すと、crmがそのオブジェクトを完了します。

すでに説明した方法に加えて、crmshは、同期コマンド実行もサポートしています。これを有効にするには、**-w**オプションを使用します。crmを**-w**なしで起動した場合でも、後ほどユーザ初期設定の**wait**を**yes**に設定すれば(**options wait yes**)、有効にすることができます。このオプションが有効化される場合、crmは遷移が終了するまで待機します。処理が開始すると毎回、進行状況を示すための点が表示されます。同期コマンドの実行は**resource start**などのコマンドにのみ適用できます。



注記: 管理サブコマンドと設定サブコマンド間の相違

crmツールには管理機能(サブコマンド resources および node)があり、設定に使用できます(cib、configure)。

以降のサブセクションでは、crmツールの重要な側面について、その概要を示します。

8.1.3 OCFリソースエージェントに関する情報の表示

リソースエージェントはクラスタ設定で常に操作する必要があるため、`crm`ツールには、`ra`コマンドが含まれています。このコマンドを使用して、リソースエージェントの情報を表示し、リソースエージェントを管理します(詳細は6.3.2項「サポートされるリソースエージェントクラス」も参照)。

```
root # crm ra
crm(live)ra#
```

コマンド `classes` は、すべてのクラスとプロバイダを一覧表示します。

```
crm(live)ra# classes
lsb
ocf / heartbeat linbit lvm2 ocfs2 pacemaker
service
stonith
systemd
```

クラス(およびプロバイダ)に使用できるすべてのリソースエージェントの概要を取得するには、`list`コマンドを使用します。

```
crm(live)ra# list ocf
AoEtarget      AudibleAlarm   CTDB            ClusterMon
Delay          Dummy          EvmsSCC         Evmsd
Filesystem     HealthCPU      HealthSMART     ICP
IPaddr         IPaddr2        IPsrcaddr       IPv6addr
LVM            LinuxSCSI      MailTo          ManageRAID
ManageVE       Pure-FTPd      Raid1           Route
SAPDatabase    SAPIInstance   SendArp         ServeRAID
...
```

リソースエージェントの概要は、`info`で表示できます。

```
crm(live)ra# info ocf:linbit:drbd
This resource agent manages a DRBD* resource
as a master/slave resource. DRBD is a shared-nothing replicated storage
device. (ocf:linbit:drbd)

Master/Slave OCF Resource Agent for DRBD

Parameters (* denotes required, [] the default):

drbd_resource* (string): drbd resource name
    The name of the drbd resource from the drbd.conf file.

drbdconf (string, [/etc/drbd.conf]): Path to drbd.conf
```

```
Full path to the drbd.conf file.
```

```
Operations' defaults (advisory minimum):
```

```
start          timeout=240
promote        timeout=90
demote         timeout=90
notify        timeout=90
stop          timeout=100
monitor_Slave_0 interval=20 timeout=20 start-delay=1m
monitor_Master_0 interval=10 timeout=20 start-delay=1m
```

ビューアは、「**Q**」を押すと終了できます。



ヒント: **crm**の直接使用

前の例では、**crm**コマンドの内部シェルを使用しました。ただし、必ずしも、それを使用する必要はありません。該当するサブコマンドを**crm**に追加すれば、同じ結果が得られます。たとえば、すべてのOCFリソースエージェントを一覧するには、シェルに「**crm ra list ocf**」を入力すれば済みます。

8.1.4 **crmsh**のシェルスクリプトの使用

crmshシェルスクリプトは、**crmsh**サブコマンドをファイル内に列挙する便利な方法を提供します。これにより、特定の行をコメントしたり、これらのコメントを後で再生したりするのが簡単になります。**crmsh**シェルスクリプトには「**crmsh**サブコマンドのみ」を含めることができることに注意してください。他のコマンドは許可されていません。

crmshシェルスクリプトを使用するには、その前に特定のコマンドを使用してファイルを作成してください。たとえば、次のファイルにはクラスタのステータスが出力され、すべてのノードのリストが提供されます。

例 8.1: 単純な**CRMSH**シェルスクリプト

```
# A small example file with some crm subcommands
status
node list
```

ハッシュ記号(#)で始まる行はコメントなので、無視されます。行が長すぎる場合は、行末にバックスラッシュ(\)を挿入します。可読性を向上させるため、特定のサブコマンドに属する行をインデントすることをお勧めします。

このスクリプトを使用するには、次の方法のいずれかを使用します。

```
root # crm -f example.cli
root # crm < example.cli
```


8.1.5 crmshのクラスタスクリプトの使用

すべてのクラスタノードから情報を収集して変更をすべて展開することは、鍵となるクラスタ管理タスクです。複数のノードで同じ手順を手動で実行するのはミスを起こしがちであるため、代わりにcrmshクラスタスクリプトを使用できます。

これらを「crmshシェルスクリプト」と混同しないでください(8.1.4項「[crmshのシェルスクリプトの使用](#)」で説明)。

crmshシェルスクリプトとは対照的に、クラスタスクリプトでは次のような追加のタスクを実行します。

- 特定のタスクに必要なソフトウェアをインストールする。
- 設定ファイルを作成または変更する。
- 情報を収集し、クラスタの潜在的な問題をレポートする。
- 変更をすべてのノードに展開する。

crmshクラスタスクリプトは、他のクラスタ管理ツールを置き換えるものではなく、クラスタ全体に対して統合化された方法でこれらのタスクを実行できるようにします。詳細については、<http://crmsh.github.io/scripts/>  を参照してください。

8.1.5.1 使用方法

利用可能なすべてのクラスタのリストを取得するには、次のコマンドを実行します。

```
root # crm script list
```

スクリプトのコンポーネントを表示するには、**show**コマンドと、クラスタスクリプトの名前を使用します。次に例を示します。

```
root # crm script show mailto
mailto (Basic)
MailTo
```

```
This is a resource agent for MailTo. It sends email to a sysadmin
whenever a takeover occurs.
```

1. Notifies recipients by email in the event of resource takeover

```
id (required) (unique)
    Identifier for the cluster resource
email (required)
    Email address
subject
    Subject
```

showの出力には、タイトル、短い説明、および手順が含まれます。各手順は一連のステップに分かれており、これらのステップを指定された順序で実行します。

各ステップには、必須パラメータとオプションパラメータのリスト、および短い説明とそのデフォルト値が含まれます。

各クラスタスクリプトは、一連の共通パラメータを認識します。これらのパラメータは任意のスクリプトに渡すことができます。

表 8.1: 共通パラメータ

パラメータ	引数	説明
<u>action</u>	<u>INDEX</u>	設定した場合、1つのアクションのみを実行します (verifyによって返されたインデックス)。
<u>dry_run</u>	<u>BOOL</u>	設定した場合、実行のシミュレートのみを行います(デフォルト: no)。
<u>nodes</u>	<u>LIST</u>	スクリプト実行対象のノードのリスト。
<u>ポート</u>	<u>NUMBER</u>	接続先のポート。
<u>statefile</u>	<u>FILE</u>	シングルステップ実行の場合に、指定したファイルに状態を保存します。
<u>sudo</u>	<u>BOOL</u>	設定した場合、sudoパスワードを入力するようcrmによってプロンプトが表示され、必要に応じてsudoが

パラメータ	引数	説明
		使用されます(デフォルト: no)。
<u>timeout</u>	<u>NUMBER</u>	秒単位での実行タイムアウト(デフォルト: 600)。
<u>user</u>	<u>USER</u>	指定したユーザとしてスクリプトを実行します。

8.1.5.2 クラスタスクリプトの検証と実行

問題を避けるため、クラスタスクリプトを実行する前に、実行するアクションを確認してパラメータを検証します。クラスタスクリプトは一連のアクションを実行でき、さまざまな理由で失敗する可能性があります。そのため、実行前にパラメータを検証すると、問題の回避に役立ちます。

たとえば、mailtoリソースエージェントでは、固有の識別子と電子メールアドレスが必要です。これらのパラメータを検証するには、以下を実行します。

```
root # crm script verify mailto id=sysadmin email=tux@example.org
1. Ensure mail package is installed

    mailx

2. Configure cluster resources

    primitive sysadmin ocf:heartbeat:MailTo
        email="tux@example.org"
        op start timeout="10"
        op stop timeout="10"
        op monitor interval="10" timeout="10"

    clone c-sysadmin sysadmin
```

verifyは各ステップを出力し、指定したパラメータでプレースホルダを置き換えます。問題が見つかった場合、**verify**によってレポートされます。問題がなければ、**verify**コマンドを**run**に置き換えます。

```
root # crm script run mailto id=sysadmin email=tux@example.org
INFO: MailTo
INFO: Nodes: alice, bob
OK: Ensure mail package is installed
OK: Configure cluster resources
```

`crm status`を使用して、リソースがクラスタに統合されているかどうかを確認します。

```
root # crm status
[...]  
Clone Set: c-sysadmin [sysadmin]  
Started: [ alice bob ]
```

8.1.6 設定テンプレートの使用



注記: 非推奨に関する注意

設定テンプレートの使用は非推奨で、今後削除される予定です。設定テンプレートはクラスタスクリプトに置き換えられます。[8.1.5項「crmshのクラスタスクリプトの使用」](#)を参照してください。

設定テンプレートは、crmsh用の既成のクラスタ設定です。「リソーステンプレート」([8.3.3項「リソーステンプレートの作成」](#)の説明を参照)と混同しないでください。これらは「クラスタ」用のテンプレートで、crmシェル用ではありません。

設定テンプレートは、最小限の操作で、特定ユーザのニーズに合わせて調整できます。テンプレートで設定を作成する際には、警告メッセージでヒントが与えられます。これは、後から編集することができ、さらにカスタマイズできます。

次の手順は、簡単ですが機能的なApache設定を作成する方法を示しています。

1. `root`としてログインし、`crm`対話型シェルを開始します。

```
root # crm configure
```

2. 設定テンプレートから新しい設定を作成します。

- a. `template`サブコマンドに切り替えます。

```
crm(live)configure# template
```

- b. 使用可能な設定テンプレートを一覧します。

```
crm(live)configure template# list templates  
gfs2-base  filesystem  virtual-ip  apache      clvm        ocfs2       gfs2
```

- c. 必要な設定テンプレートを決めます。Apache設定が必要なので、`apache`テンプレートを選択し、`g-intranet`と名付けます。

```
crm(live)configure template# new g-intranet apache
INFO: pulling in template apache
INFO: pulling in template virtual-ip
```

3. パラメータを定義します。

a. 作成した設定を一覧表示します。

```
crm(live)configure template# list
g-intranet
```

b. 入力を必要とする最小限の変更項目を表示します。

```
crm(live)configure template# show
ERROR: 23: required parameter ip not set
ERROR: 61: required parameter id not set
ERROR: 65: required parameter configfile not set
```

c. 好みのテキストエディタを起動し、[ステップ 3.b](#)でエラーとして表示されたすべての行に入力します。

```
crm(live)configure template# edit
```

4. 設定を表示し、設定が有効かどうか確認します(太字のテキストは、[ステップ 3.c](#)で入力した設定によって異なります)。

```
crm(live)configure template# show
primitive virtual-ip ocf:heartbeat:IPAddr \
  params ip= "192.168.1.101"
primitive apache ocf:heartbeat:apache \
  params configfile= "/etc/apache2/httpd.conf"
monitor apache 120s:60s
group 「g-intranet」 \
  apache virtual-ip
```

5. 設定を適用します。

```
crm(live)configure template# apply
crm(live)configure# cd ..
crm(live)configure# show
```

6. 変更内容をCIBに送信します。

```
crm(live)configure# commit
```

詳細がわかっている場合は、コマンドをさらに簡素化できます。次のコマンドをシェルで使用して、上記の手順を要約できます。

```
root # crm configure template \  
new g-intranet apache params \  
configfile="/etc/apache2/httpd.conf" ip="192.168.1.101"
```

内部`crm`シェルに入っている場合は、次のコマンドを使用します。

```
crm(live)configure template# new intranet apache params \  
configfile="/etc/apache2/httpd.conf" ip="192.168.1.101"
```

ただし、このコマンドは、設定テンプレートから設定を作成するだけです。設定をCIBに適用したり、コミットすることはありません。

8.1.7 シャドーイング設定のテスト

シャドー構成は、異なる構成シナリオのテストに使用されます。複数のシャドウ設定を作成した場合は、1つ1つテストして変更を加えた影響を確認できます。

通常の処理は次のようになります。

1. `root`としてログインし、`crm`対話型シェルを開始します。

```
root # crm configure
```

2. 新しいシャドウ設定を作成します。

```
crm(live)configure# cib new myNewConfig  
INFO: myNewConfig shadow CIB created
```

シャドウCIBの名前を省略する場合は、一時名の`@tmp@`が作成されます。

3. 現在のライブ設定をシャドウ設定にコピーするには、次のコマンドを使用します。コピーしない場合は、このステップをスキップします。

```
crm(myNewConfig)# cib reset myNewConfig
```

このコマンドを使用すると、既存のリソースを後から編集する場合に、簡単に編集できます。

4. 通常どおり変更を行います。シャドウ設定の作成後は、すべての変更がシャドウ設定に適用されます。すべての変更を保存するには、次のコマンドを使用します。

```
crm(myNewConfig)# commit
```

5. ライブクラスタ設定が再び必要な場合は、次のコマンドでライブ設定に戻ります。

```
crm(myNewConfig)configure# cib use live  
crm(live)#
```

8.1.8 設定の変更のデバッグ

設定の変更をクラスタにロードする前に、変更内容を`ptest`でレビューすることを推奨します。`ptest`コマンドを指定すると、変更のコミットによって生じるアクションのダイアグラムを表示できます。ダイアグラムを表示するには、`graphviz`パッケージが必要です。次の例は監視操作を追加するスクリプトです。

```
root # crm configure
crm(live)configure# show fence-bob
primitive fence-bob stonith:apcsmart \
    params hostlist="bob"
crm(live)configure# monitor fence-bob 120m:60s
crm(live)configure# show changed
primitive fence-bob stonith:apcsmart \
    params hostlist="bob" \
    op monitor interval="120m" timeout="60s"
crm(live)configure# 「ptest」
crm(live)configure# commit
```

8.1.9 クラスタダイアグラム

クラスタダイアグラムを出力するには、コマンド`crm configure graph`を使用します。これにより現在の設定が現在のウィンドウに表示されるので、X11が必要になります。

SVG (Scalable Vector Graphics)を使用する場合は、次のコマンドを使用します。

```
root # crm configure graph dot config.svg svg
```

8.2 Corosync設定の管理

Corosyncは、ほとんどのHAクラスタの下層にあるメッセージング層です。`corosync`サブコマンドは、Corosync設定を編集および管理するためのコマンドを提供します。

たとえば、クラスタのステータスを一覧表示するには、`status`を使用します。

```
root # crm corosync status
Printing ring status.
Local node ID 175704363
RING ID 0
    id      = 10.121.9.43
    status  = ring 0 active with no faults
Quorum information
-----
```

```
Date: Thu May 8 16:41:56 2014
Quorum provider: corosync_votequorum
Nodes: 2
Node ID: 175704363
Ring ID: 4032
Quorate: Yes
```

Votequorum information

```
Expected votes: 2
Highest expected: 2
Total votes: 2
Quorum: 2
Flags: Quorate
```

Membership information

Nodeid	Votes	Name
175704363	1	alice.example.com (local)
175704619	1	bob.example.com

diff コマンドは非常に便利です。すべてのノード上のCorosync設定を比較し(別途記載のない場合)、それらの差異を出力します。

```
root # crm corosync diff
--- bob
+++ alice
@@ -46,2 +46,2 @@
-     expected_votes: 2
-     two_node: 1
+     expected_votes: 1
+     two_node: 0
```

詳細については、「http://crmsh.nongnu.org/crm.8.html#cmdhelp_corosync」を参照してください。

8.3 クラスタリソースの設定

クラスタの管理者は、クラスタ内のサーバ上の各リソースや、サーバ上で実行する各アプリケーションに対してクラスタリソースを作成する必要があります。クラスタリソースには、Webサイト、電子メールサーバ、データベース、ファイルシステム、仮想マシン、およびユーザが常時使用できるようにする他のサーバベースのアプリケーションまたはサービスなどが含まれます。

作成できるリソースタイプの概要については、6.3.3項「[リソースのタイプ](#)」を参照してください。

8.3.1 ファイルからのクラスタリソースのロード

設定の一部またはすべてをローカルファイルまたはネットワークURLからロードできます。次の3つの異なる方法を定義できます。

置き換え

このオプションは、現在の設定を新たなソース設定に置き換えます。

update

このオプションは、ソース設定のインポートを試みます。現在の設定に新たな項目を追加したり、既存の項目を更新したりします。

push

このオプションは、ソースからのコンテンツを現在の設定にインポートします(updateと同じ)。ただし、新しい設定で使用できないオブジェクトを削除します。

ファイル`mycluster-config.txt`から新しい設定をロードするには、次の構文を使用します。

```
root # crm configure load push mycluster-config.txt
```

8.3.2 クラスタリソースの作成

クラスタで使えるRA(リソースエージェント)には3種類あります(背景情報については6.3.2項「サポートされるリソースエージェントクラス」を参照)。新しいリソースをクラスタに追加するには、次の手順に従います。

1. rootとしてログインし、crmツールを開始します。

```
root # crm configure
```

2. プリミティブIPアドレスを設定します。

```
crm(live)configure# primitive myIP ocf:heartbeat:IPaddr \  
params ip=127.0.0.99 op monitor interval=60s
```

前のコマンドは「プリミティブ」に名前`myIP`を設定します。クラス(ここでは`ocf`)、プロバイダ(`heartbeat`)、およびタイプ(`IPaddr`)を選択する必要があります。さらに、このプリミティブでは、IPアドレスなどのパラメータが必要です。自分の設定に合わせてアドレスを変更してください。

3. 行った変更を表示して確認します。

```
crm(live)configure# show
```

4. 変更をコミットして反映させます。

```
crm(live)configure# commit
```

8.3.3 リソーステンプレートの作成

類似した設定で複数のリソースを作成する場合、リソーステンプレートを使用すれば作業が簡単になります。基本的なバックグラウンド情報については、6.5.3項「リソーステンプレートと制約」を参照してください。これらを、「通常の」テンプレート(8.1.6項「設定テンプレートの使用」で説明したもの)と混同しないでください。次の構文を知るには、**rsc_template**コマンドを使用してください。

```
root # crm configure rsc_template
usage: rsc_template <name> [<class>:[<provider>:]]<type>
      [params <param>=<value> [<param>=<value>...]]
      [meta <attribute>=<value> [<attribute>=<value>...]]
      [utilization <attribute>=<value> [<attribute>=<value>...]]
      [operations id_spec
        [op op_type [<attribute>=<value>...] ...]]
```

たとえば、次のコマンドは、ocf:heartbeat:Xenリソースと、デフォルト値および操作に由来するBigVMの名前を持つ新しいリソーステンプレートを作成します。

```
crm(live)configure# rsc_template BigVM ocf:heartbeat:Xen \
  params allow_mem_management="true" \
  op monitor timeout=60s interval=15s \
  op stop timeout=10m \
  op start timeout=10m
```

新しいリソーステンプレートを定義したら、それをプリミティブとして使用すること、または順序、コロケーション、またはrsc_ticketの制約で参照することができます。リソーステンプレートを参照するには、@記号を使用します。

```
crm(live)configure# primitive MyVM1 @BigVM \
  params xmfile="/etc/xen/shared-vm/MyVM1" name="MyVM1"
```

新しいプリミティブMy-VM1は、BigVMリソーステンプレートからすべてを継承します。たとえば、上の2つに等しいものは次のようになります。

```
crm(live)configure# primitive MyVM1 ocf:heartbeat:Xen \
  params xmfile="/etc/xen/shared-vm/MyVM1" name="MyVM1" \
  params allow_mem_management="true" \
  op monitor timeout=60s interval=15s \
  op stop timeout=10m \
  op start timeout=10m
```


オプションや操作を上書きしたい場合は、自分の(プリミティブの)定義を追加します。たとえば、次の新しいプリミティブMyVM2は監視操作のタイムアウトを2倍にしますが、その他はそのままだと残します。

```
crm(live)configure# primitive MyVM2 @BigVM \  
  params xmfile="/etc/xen/shared-vm/MyVM2" name="MyVM2" \  
  op monitor timeout=120s interval=30s
```

リソーステンプレートは、そのテンプレートから派生するすべてのプリミティブを表すものとして、制約で参照することができます。これにより、クラスタ設定をいっそう簡潔かつクリアに行うことができます。リソーステンプレートは、場所の制約を除くすべての制約から参照することができます。コロケーション制約には、複数のテンプレート参照を含めることはできません。

8.3.4 STONITHリソースの作成

crmからは、STONITHデバイスは単なる1つのリソースと認識されます。STONITHリソースを作成するには、次の手順に従います。

1. **root**としてログインし、**crm**対話型シェルを開始します。

```
root # crm configure
```

2. 次のコマンドで、すべてのSTONITHタイプのリストを取得します。

```
crm(live)# ra list stonith  
apcmaster          apcmastersnmp      apcsmart  
baytech            bladehpi            cyclades  
drac3              external/drac5      external/dracmc-telnet  
external/hetzner    external/hmchttp    external/ibmrsa  
external/ibmrsa-telnet external/ipmi        external/ippower9258  
external/kdumpcheck external/libvirt     external/nut  
external/rackpdu    external/riloe      external/sbd  
external/vcenter    external/vmware     external/xen0  
external/xen0-ha     fence_legacy        ibmhmc  
ipmilan              meatware             nw_rpc100s  
rcd_serial           rps10                suicide  
wti_mpc              wti_nps
```

3. 上記のリストからSTONITHタイプを選択し、利用できるオプションのリストを表示します。次のコマンドを実行します。

```
crm(live)# ra info stonith:external/ipmi  
IPMI STONITH external device (stonith:external/ipmi)
```

```
ipmitool based power management. Apparently, the power off method of ipmitool is intercepted by ACPI which then makes a regular shutdown. If case of a split brain on a two-node it may happen that no node survives. For two-node clusters use only the reset method.
```

Parameters (* denotes required, [] the default):

```
hostname (string): Hostname
    The name of the host to be managed by this STONITH device.
...
```

4. `stonith`クラス、ステップ3で選択したタイプ、および必要に応じて該当するパラメータを使用して、STONITHリソースを作成します。たとえば、次のコマンドを使用します。

```
crm(live)# configure
crm(live)configure# primitive my-stonith stonith:external/ipmi \
    params hostname="alice" \
    ipaddr="192.168.1.221" \
    userid="admin" passwd="secret" \
    op monitor interval=60m timeout=120s
```

8.3.5 リソース制約の設定

すべてのリソースを設定することは、ジョブのほんの一部です。クラスタが必要なすべてのリソースを認識しても、正しく処理できるとは限りません。たとえば、DRBDのスレーブノードにファイルシステムをマウントしないようにしてください(実際、DRBDでは失敗します)。このような情報をクラスタが利用できるように、制約を定義します。

制約の詳細については、6.5項「リソースの制約」を参照してください。

8.3.5.1 場所の制約

location コマンドは、リソースを実行できるノード、できないノード、または実行に適したノードを定義するものです。

この種類の制約は、各リソースに複数追加できます。すべての**location**制約は、所定のリソースに関して評価されます。`fs1`というIDを持つリソースを**alice**という名前のノード上で実行するプリファレンスを100にする簡単な例を次に示します。

```
crm(live)configure# location loc-fs1 fs1 100: alice
```

もう1つの例は、pingによる場所の設定です。

```
crm(live)configure# primitive ping ping \
    params name=ping dampen=5s multiplier=100 host_list="r1 r2"
crm(live)configure# clone cl-ping ping meta interleave=true
crm(live)configure# location loc-node_pref internal_www \
    rule 50: #uname eq alice \
    rule ping: defined ping
```

パラメータ `host_list` は、ping および カウント するホストのスペース区切りのリストです。場所の制約のもう1つの使用例は、「リソースセット」としてのプリミティブのグループ化です。これは、たとえば、いくつかのリソースがネットワーク接続のping属性によって異なるときに役立つ場合があります。以前は、`-inf/ping` ルールを設定で何度も重複して指定する必要があったため、設定内容が不必要に複雑でした。

次の例では、仮想IPアドレス `vip1` および `vip2` を参照して、リソースセット `loc-alice` を作成します。

```
crm(live)configure# primitive vip1 ocf:heartbeat:IPaddr2 params ip=192.168.1.5
crm(live)configure# primitive vip2 ocf:heartbeat:IPaddr2 params ip=192.168.1.6
crm(live)configure# location loc-alice { vip1 vip2 } inf: alice
```

ある場合には、`location` コマンドでリソースパターンを使用すると、より効率的で便利です。リソースパターンは、2つのスラッシュ間の正規表現です。たとえば、前に示した仮想IPアドレスは、次とすべて一致させることができます。

```
crm(live)configure# location loc-alice /vip.*/ inf: alice
```

8.3.5.2 コロケーション制約

`colocation` コマンドは、同じホストまたは別のホストで実行すべきリソースを定義するために使用します。

常に同じノードで実行する必要があるリソース、または同じノードで実行してはならないリソースを定義する場合には、それぞれ `+inf` または `-inf` のスコアを設定することだけが可能です。無限大以外のスコアの使用も可能です。その場合、コロケーションは「advisory」と呼ばれ、衝突が発生したときに他のリソースが停止しないようにするため、クラスタがそれらの制約に従わないこともあります。

たとえば、IDが `filesystem_resource` と `nfs_group` のリソースを常に同じホストで実行するには、次の制約を使用します。

```
crm(live)configure# colocation nfs_on_filesystem inf: nfs_group filesystem_resource
```

マスタスレーブ構成では、現在のノートがマスタかどうかと、リソースをローカルに実行しているかどうかを把握することが必要です。

8.3.5.3 依存性なしのリソースセットのコロケーション

同じノード上にリソースのグループを配置できると便利な場合があります(コロケーション制約を定義)、リソース間で困難な依存性を持つことはありません。

同じノード上にリソースを配置するが、これらの一方に障害が発生した場合のアクションがない場合は、**weak-bond**コマンドを使用します。

```
root # crm configure assist weak-bond RES1 RES2
```

weak-bondの実装により、指定されたリソースを持つダミーリソースとコロケーション制約が自動的に作成されます。

8.3.5.4 順序の制約

orderコマンドは、アクションのシーケンスを定義します。

リソースのアクションや操作の順序を指定することが必要な場合があります。たとえば、デバイスがシステムで利用できるようになるまで、ファイルシステムはマウントできません。順序の制約を使用して、開始、停止、マスタへの昇格など、別のリソースが特殊な条件を満たす直前または直後に、サービスを開始または停止できます。

順序の制約を設定するには、次のようなコマンドを**crm**シェルで使用します。

```
crm(live)configure# order nfs_after_filesystem Mandatory: filesystem_resource nfs_group
```

8.3.5.5 サンプル設定のための制約

このセクションで使用される例は、制約を追加しないと機能しません。すべてのリソースは、必ず、マスタであるDRBDリソースと同じマシンで実行される必要があります。DRBDリソースは、他のリソースが開始する前にマスタにする必要があります。マスタでないときに、drbdデバイスをマウントしようとするとう失敗します。次の制約を満たす必要があります。

- ファイルシステムは、常に、DRBDリソースのマスタと同じノード上に存在する必要があります。

```
crm(live)configure# colocation filesystem_on_master inf: \  
filesystem_resource drbd_resource:Master
```

- NFSサーバとIPアドレスは、ファイルシステムと同じノードに存在する必要があります。

```
crm(live)configure# colocation nfs_with_fs inf: \
nfs_group filesystem_resource
```

- NFSサーバとIPアドレスは、ファイルシステムがマウントされた後に開始されます。

```
crm(live)configure# order nfs_second Mandatory: \
filesystem_resource:start nfs_group
```

- ファイルシステムは、drbdリソースがこのノードのマスタに昇格した後にマウントされる必要があります。

```
crm(live)configure# order drbd_first Mandatory: \
drbd_resource:promote filesystem_resource:start
```

8.3.6 リソースフェールオーバーノードの指定

リソースフェールオーバーを判定するには、メタ属性migration-thresholdを使用します。すべてのノードで失敗回数がmigration-thresholdを超えている場合には、リソースは停止したままになります。例:

```
crm(live)configure# location rsc1-alice rsc1 100: alice
```

通常、rsc1はaliceで実行されます。そこで失敗すると、migration-thresholdがチェックされ、失敗回数と比較されます。失敗回数がmigration-threshold以上の場合、次の候補のノードにマイグレートします。

開始が失敗すると、start-failure-is-fatalオプションによっては、失敗回数がinfに設定されます。stopの失敗により、フェンシングが発生します。STONITHが定義されていない場合には、リソースは移行しません。

概要については、[6.5.4項「フェールオーバーノード」](#)を参照してください。

8.3.7 リソースフェールバックノードの指定(リソースの固着性)

ノードがオンライン状態に戻り、クラスタ内にある場合は、リソースが元のノードにフェールバックすることがあります。リソースを実行していたノードにリソースをフェールバックさせたくない場合や、リソースのフェールバック先として別のノードを指定する場合は、リソースの固着性の値を変更します。リソースの固着性は、リソースの作成時でも、その後も指定できます。

概要については、[6.5.5項「フェールバックノード」](#)を参照してください。

8.3.8 負荷インパクトに基づくリソース配置の設定

一部のリソースは、メモリの最小量など、特定の容量要件を持っています。要件が満たされていない場合、リソースは全く開始しないか、またはパフォーマンスを下げた状態で実行されます。

これを考慮に入れて、High Availability Extensionでは、次のパラメータを指定できます。

1. 一定のノードが「提供する」容量
2. 一定のリソースが「要求する」容量
3. リソースの配置に関する全体的なストラテジ

パラメータと設定の詳細な背景情報については、6.5.6項「負荷インパクトに基づくリソースの配置」を参照してください。

リソースの要件とノードが提供する容量を設定するには、使用属性を使用します。使用属性に任意の名前を付け、設定に必要なだけ名前/値のペアを定義します。いくつかのエージェントは、たとえばVirtualDomainなどの使用を更新します。

次の例では、クラスタのノードとリソースの基本設定がすでに完了していることを想定しています。さらに、特定のノードが提供する容量と特定のリソースが必要とする容量を設定します。

手順 8.1: `crm`で使用属性を追加または変更する

1. `root`としてログインし、`crm`対話型シェルを開始します。

```
root # crm configure
```

2. ノードが「提供する」容量を指定するには、次のコマンドを使用し、プレースホルダ`NODE_1`をノードの名前に置き換えます。

```
crm(live)configure# node NODE_1 utilization memory=16384 cpu=8
```

これらの値によって、`NODE_1`は16GBのメモリと8つのCPUコアをリソースに提供すると想定されます。

3. リソースが「要求する」容量を指定するには、次のコマンドを使用します。

```
crm(live)configure# primitive xen1 ocf:heartbeat:Xen ... \  
utilization memory=4096 cpu=4
```

これによって、リソースは`NODE_1`からの4,096のメモリ単位と4つのCPU単位を使用します。

4. **property**コマンドを使用して、配置ストラテジを設定します。

```
crm(live)configure# property ...
```

次の値を使用できます。

default (デフォルト値)

使用値は考慮しません。リソースは、場所のスコアに従って割り当てられます。スコアが同じであれば、リソースはノード間で均等に分散されます。

utilization

リソースの要件を満たすだけの空き容量がノードにあるかどうか決定する際に、利用率を確認します。ただし、負荷分散は、まだ、ノードに割り当てられたリソースの数に基づいて行われます。

minimal

リソースの要件を満たすだけの空き容量がノードにあるかどうか決定する際に、利用率を確認します。できるだけ少ない数のノードにリソースを集中しようとし、す(残りのノードの電力節約のため)。

balanced

リソースの要件を満たすだけの空き容量がノードにあるかどうか決定する際に、利用率を確認します。リソースを均等に分散して、リソースのパフォーマンスを最適化しようとし、す。



注記: リソース優先度の設定

使用できる配置ストラテジは、最善策であり、まだ、複雑なヒューリスティックソルバで、常に最適な割り当て結果を得るには至っていません。リソースの優先度を正しく設定して、最重要なリソースが最初にスケジュールされるようにしてください。

5. 変更をコミットしてから、crmshを終了します。

```
crm(live)configure# commit
```

次の例は、同等のノードから成る3ノードクラスターと4つの仮想マシンを示しています。

```
crm(live)configure# node alice utilization memory="4000"  
crm(live)configure# node bob utilization memory="4000"  
crm(live)configure# node charlie utilization memory="4000"  
crm(live)configure# primitive xenA ocf:heartbeat:Xen \  
utilization hv_memory="3500" meta priority="10" \  

```



```

    params xmfile="/etc/xen/shared-vm/vm1"
crm(live)configure# primitive xenB ocf:heartbeat:Xen \
    utilization hv_memory="2000" meta priority="1" \
    params xmfile="/etc/xen/shared-vm/vm2"
crm(live)configure# primitive xenC ocf:heartbeat:Xen \
    utilization hv_memory="2000" meta priority="1" \
    params xmfile="/etc/xen/shared-vm/vm3"
crm(live)configure# primitive xenD ocf:heartbeat:Xen \
    utilization hv_memory="1000" meta priority="5" \
    params xmfile="/etc/xen/shared-vm/vm4"
crm(live)configure# property placement-strategy="minimal"

```

3ノードはすべてアクティブであり、まず、xenAがノードに配置され、次に、xenDが配置されます。xenBとxenCは、一緒に割り当てられるか、またはどちらか1つがxenDとともに割り当てられます。

1つのノードに障害が発生した場合、残りのノード上で利用できるメモリ合計が少なすぎて、これらのリソースすべてはホストできません。xenAは確実に割り当てられ、xenDも同様です。ただし、xenBまたはxenCのいずれかしか配置できず、その優先度は同じであるため、結果はまだ定義されていません。これを解決するためにも、どちらかに高い優先度を設定する必要があります。

8.3.9 リソース監視の設定

リソースを監視するには、2つの方法(opキーワードで監視処理を定義するか、monitorコマンドを使用するか)があります。次の例では、Apacheリソースを設定し、opキーワードの使用で60秒ごとに監視します。

```

crm(live)configure# primitive apache apache \
    params ... \
    「op monitor interval=60s timeout=30s」

```

同じことを次のようにしても実行できます。

```

crm(live)configure# primitive apache apache \
    params ...
crm(live)configure# monitor apache 60s:30s

```

概要については、[6.4項「リソース監視」](#)を参照してください。

8.3.10 クラスタリソースグループの設定

クラスタの一般的な要素の1つは、一緒の場所で見つける必要のあるリソースのセットです。連続的に開始し、逆の順序で停止します。この設定を簡単にするため、グループのコンセプトをサポートしています。次の例では、2つのプリミティブ(IPアドレスと電子メールリソース)を作成します。

1. **crm**コマンドをシステム管理者として実行します。プロンプトが`crm(live)`に変化します。
2. プリミティブを設定します。

```
crm(live)# configure
crm(live)configure# primitive Public-IP ocf:heartbeat:IPaddr \
    params ip=1.2.3.4 id= Public-IP
crm(live)configure# primitive Email systemd:postfix \
    params id=Email
```

3. 該当するIDを使用して、正しい順序で、プリミティブをグループ化します。

```
crm(live)configure# group g-mailsvc Public-IP Email
```

グループメンバーの順序を変更するには、**configure**サブコマンドから**modgroup**コマンドを使用します。プリミティブのEmailをPublic-IPの前に移動するには、次のコマンドを使用します(このコマンドは機能のデモのみを目的としています)。

```
crm(live)configure# modgroup g-mailsvc add Email before Public-IP
```

グループ(Emailなど)からリソースを削除する場合には、次のコマンドを使用します。

```
crm(live)configure# modgroup g-mailsvc remove Email
```

概要については、6.3.5.1項「グループ」を参照してください。

8.3.11 クローンリソースの設定

クローンは当初、IPアドレスのN個のインスタンスを開始し、負荷分散のためにクラスタ上に分散させる便利な方法と考えられていました。それらは、DLMとの統合、サブシステムおよびOCFS2のフェンシングなど、複数の目的にも有効であることがわかってきました。どのようなリソースでも、リソースエージェントがサポートしていれば、クローン化できます。

クローンリソースの詳細については、6.3.5.2項「クローン」を参照してください。

8.3.11.1 匿名クローンリソースの作成

匿名クローンリソースを作成するには、まずプリミティブリソースを作成して、それを **clone** コマンドで指定することです。次の操作を実行してください:

1. **root** としてログインし、**crm** 対話型シェルを開始します。

```
root # crm configure
```

2. 次のように、プリミティブを設定します。

```
crm(live)configure# primitive Apache ocf:heartbeat:apache
```

3. プリミティブをクローンします。

```
crm(live)configure# clone cl-apache Apache
```

8.3.11.2 プロモータブルクローンリソースの作成

プロモータブルクローンリソース(以前はマルチステートリソースと呼ばれていました)は、クローンが得意とするところです。これにより、インスタンスを2つの動作モード(active/passive、primary/secondary、またはmaster/slave)のいずれかに設定できます。

プロモータブルクローンリソースを作成するには、まずプリミティブリソースを作成してから、プロモータブルクローンリソースを作成します。プロモータブルクローンリソースは少なくとも、昇格および降格操作をサポートしている必要があります。

1. **root** としてログインし、**crm** 対話型シェルを開始します。

```
root # crm configure
```

2. プリミティブを作成します。必要に応じて間隔を変更します。

```
crm(live)configure# primitive my-rsc ocf:myCorp:myAppl \  
    op monitor interval=60 \  
    op monitor interval=61 role=Master
```

3. プロモータブルクローンリソースを作成します。

```
crm(live)configure# ms ms-rsc my-rsc
```

8.4 クラスタリソースの管理

`crm`ツールでは、クラスタリソースの設定が可能だけでなく、既存リソースを管理することもできます。移行のサブセクションで概要を示します。

8.4.1 クラスタリソースの表示

クラスタを管理するには、コマンド `crm configure show` で、クラスタ設定、グローバルオプション、プリミティブなどの現在のCIBオブジェクトを一覧表示します。

```
root # crm configure show
node 178326192: alice
node 178326448: bob
primitive admin_addr IPAddr2 \
    params ip=192.168.2.1 \
    op monitor interval=10 timeout=20
primitive stonith-sbd stonith:external/sbd \
    params pcmk_delay_max=30
property cib-bootstrap-options: \
    have-watchdog=true \
    dc-version=1.1.15-17.1-e174ec8 \
    cluster-infrastructure=corosync \
    cluster-name=hacluster \
    stonith-enabled=true \
    placement-strategy=balanced \
    standby-mode=true
rsc_defaults rsc-options: \
    resource-stickiness=1 \
    migration-threshold=3
op_defaults op-options: \
    timeout=600 \
    record-pending=true
```

多数のリソースがある場合、`show`の出力が冗長になります。出力を制限するには、リソースの名前を使用します。たとえば、プリミティブ `admin_addr` のみのプロパティを一覧表示するには、リソース名を `show` に付加します。

```
root # crm configure show admin_addr
primitive admin_addr IPAddr2 \
    params ip=192.168.2.1 \
    op monitor interval=10 timeout=20
```

ただし、特定のリソースの出力をさらに制限したい場合があります。これは、「フィルタ」を使用して実現できます。フィルタは特定のコンポーネントに出力を制限します。たとえば、ノードのみを一覧表示するには、`type:node` を使用します。

```
root # crm configure show type:node
node 178326192: alice
node 178326448: bob
```

プリミティブにも興味がある場合には、orオペレータを使用します。

```
root # crm configure show type:node or type:primitive
node 178326192: alice
node 178326448: bob
primitive admin_addr IPAddr2 \
    params ip=192.168.2.1 \
    op monitor interval=10 timeout=20
primitive stonith-sbd stonith:external/sbd \
    params pcmk_delay_max=30
```

さらに、特定の文字列で開始するオブジェクトを検索するには次の表記を使用します。

```
root # crm configure show type:primitive and and 'admin*'
primitive admin_addr IPAddr2 \
    params ip=192.168.2.1 \
    op monitor interval=10 timeout=20
```

使用可能なすべてのタイプを一覧表示するには、**crm configure show type:**と入力し、**<Tab>** キーを押します。Bash補完により、すべてのタイプのリストが表示されます。

8.4.2 新しいクラスタリソースの開始

新しいクラスタリソースを開始するには、そのIDが必要です。以下に手順を示します。

1. rootとしてログインし、crm対話型シェルを開始します。

```
root # crm
```

2. リソースレベルに切り替えます。

```
crm(live)# resource
```

3. startでリソースを開始し、**<Tab>** キーを押してすべての既知のリソースを表示します。

```
crm(live)resource# start ID
```

8.4.3 クラスタリソースの停止

1つ以上の既存のクラスタリソースを停止するには、各IDが必要です。以下に手順を示します。

1. rootとしてログインし、crm対話型シェルを開始します。

```
root # crm
```

2. リソースレベルに切り替えます。

```
crm(live)# resource
```

3. stopでリソースを停止し、`<Tab>` キーを押してすべての既知のリソースを表示します。

```
crm(live)resource# stop ID
```

複数のリソースを一度に停止できます。

```
crm(live)resource# stop ID1 ID2 ...
```

8.4.4 リソースのクリーンアップ

リソースは、失敗した場合は自動的に再起動しますが、失敗のたびにリソースの失敗回数が増加します。migration-thresholdがそのリソースに設定されている場合は、失敗の数が移行しきい値に達すると、そのリソースはノードで実行できなくなります。

1. シェルを開いて、rootユーザとしてログインします。
2. すべてのリソースのリストを取得します。

```
root # crm resource list
...
Resource Group: dlm-clvm:1
    dlm:1 (ocf:pacemaker:controld) Started
    clvm:1 (ocf:heartbeat:lvmlockd) Started
```

3. リソースdlmをクリーンアップするには、たとえば、以下の手順を実行します:。

```
root # crm resource cleanup dlm
```

8.4.5 クラスタリソースの削除

次の手順に従って、クラスタリソースを削除します。

1. rootとしてログインし、crm対話型シェルを開始します。

```
root # crm configure
```

2. 次のコマンドを実行して、リソースのリストを取得します。

```
crm(live)# resource status
```

たとえば、出力はこのようになります(ここで、myIPはリソースの該当するID)。

```
myIP      (ocf:IPaddr:heartbeat) ...
```

3. 該当するIDを持つリソースを削除します(これは、commitも含意します)。

```
crm(live)# configure delete YOUR_ID
```

4. 変更をコミットします。

```
crm(live)# configure commit
```

8.4.6 クラスタリソースの移行

リソースは、ハードウェアまたはソフトウェアに障害が発生した場合、クラスタ内の他のノードに自動的にフェールオーバー(つまり移行)するよう設定されていますが、Hawk2またはコマンドラインを使用して、手動でリソースを別のノードに移動することもできます。

この作業を行うには、migrateコマンドを使用します。たとえば、リソース ipaddress1 を bob というクラスタノードに移行するには、次のコマンドを使用します。

```
root # crm resource
crm(live)resource# migrate ipaddress1 bob
```

8.4.7 リソースのグループ化/タグ付け

タグは、コロケーションの作成や関係の順序付けを行わずに、複数のリソースをただちに参照する方法です。これは、概念的に関連するリソースをグループ化するのに役立つ場合があります。たとえば、データベースに関連するいくつかのリソースがある場合、databases というタグを作成し、データベースに関連するすべてのリソースをこのタグに追加します。

```
root # crm configure tag databases: db1 db2 db3
```

これにより、1つのコマンドですべてを起動できます。

```
root # crm resource start databases
```

同様に、すべてを停止することもできます。

```
root # crm resource stop databases
```

8.4.8 ヘルスステータスの取得

クラスタまたはノードの「ヘルス」ステータスは、「スクリプト」というもので表示できます。スクリプトは、ヘルスだけに限らず各タスクを実行できます。ただし、このサブセクションでは、ヘルスステータスを取得する方法に焦点を当てます。

health コマンドに関するすべての詳細を取得するには、**describe** を使用します。

```
root # crm script describe health
```

このコマンドは、すべてのパラメータの説明とリスト、およびそのデフォルト値を示します。スクリプトを実行するには、**run** を使用します。

```
root # crm script run health
```

スイートから1つのステップのみを実行したい場合は、**describe** コマンドの Steps カテゴリで、使用可能なすべてのステップを一覧表示できます。

たとえば、次のコマンドは、**health** コマンドの最初のステップを実行します。さらなる調査のために、出力が `health.json` に保存されます。

```
root # crm script run health
statefile='health.json'
```

上記のコマンドは、**crm cluster health** でも実行できます。

スクリプトに関する追加情報を表示するには、<http://crmsh.github.io/scripts/> を参照してください。

8.5 cib.xml から独立したパスワードの設定

クラスタ設定にパスワードなどの機密の情報が含まれている場合、それらをローカルファイルに保存する必要があります。こうしておけば、これらのパラメータがログに記録されたり、サポートレポートに漏洩することはありません。

secret を使用する前に、リソースの概要を確認するため、**show** コマンドを実行しておくといでしょう。

```
root # crm configure show
primitive mydb ocf:heartbeat:mysql \
  params replication_user=admin ...
```

上記のmydbリソースに対してパスワードを設定するには、次のコマンドを使用します。

```
root # crm resource secret mydb set passwd linux
INFO: syncing /var/lib/heartbeat/lrm/secrets/mydb/passwd to [your node list]
```

次のように、保存されたパスワードが返されます。

```
root # crm resource secret mydb show passwd
linux
```

パラメータは、ノード間で同期する必要があることに注意してください。**crm resource secret**コマンドを使用すれば、この処理が実行されます。秘密のパラメータを管理する場合には、このコマンドを使用することを強く推奨します。

8.6 履歴情報の取得

クラスタの履歴の調査は複雑な作業です。この作業を簡素化するために、**crmsh**には**history**コマンドとそのサブコマンドが含まれています。これは、SSHが正しく設定されていることが前提となります。

それぞれのクラスタは、状態を移動し、リソースを移行し、または重要なプロセスを開始します。これらすべてのアクションは、**history**のサブコマンドによって取得できます。

デフォルトでは、すべての**history**コマンドは過去1時間のイベントを確認します。このタイムフレームを変更するには、**limit**サブコマンドを使用します。構文は次のとおりです。

```
root # crm history
crm(live)history# limit FROM_TIME [TO_TIME]
```

有効な例として、次のようなものが挙げられます。

limit4:00pm,

limit16:00


どちらのコマンドも同じ意味で、今日の午後4時を表しています。

limit2012/01/12 6pm

2012年1月12日の午後6時。

limit"Sun 5 20:46"

今年の今月の5日日曜日の午後8時46分。

その他の例とタイムフレームの作成方法については、<http://labix.org/python-dateutil>  を参照してください。

infoサブコマンドでは、**crm report**によって使用されているすべてのパラメータが表示されます。

```
crm(live)history# info
Source: live
Period: 2012-01-12 14:10:56 - end
Nodes: alice
Groups:
Resources:
```

crm reportを特定のパラメータに制限するには、サブコマンド**help**で使用可能なオプションを表示します。

詳細レベルに絞り込んでいくには、サブコマンド**detail**とレベル数を使用します。

```
crm(live)history# detail 1
```

数値が大きいほど、レポートが詳細になっていきます。デフォルト値は0 (ゼロ)です。

ここまでのパラメータを設定したら、**log**を使用してログメッセージを表示します。

最後の遷移を表示するには、次のコマンドを使用します。

```
crm(live)history# transition -1
INFO: fetching new logs, please wait ...
```

このコマンドはログをフェッチし、**dotty**を(**graphviz** パッケージから)実行して、遷移グラフを表示します。シェルはログファイルを開きます。ログ内は、**↓** と **↑** カーソルキーでブラウズできます。

遷移グラフを表示する必要がない場合には、**nograph**オプションを使用します。

```
crm(live)history# transition -1 nograph
```

8.7 詳細の参照先

- **crm** マニュアルページ。
- アップストリームプロジェクトマニュアルにアクセスします(<http://crmsh.github.io/documentation>)。
- 詳しい例については、項目「Highly Available NFS Storage with DRBD and Pacemaker」を参照してください。

9 リソースエージェントの追加または変更

クラスタによる管理が必要なすべての作業は、リソースとして使用できなければなりません。主要なグループとして、リソースエージェントとSTONITHエージェントの2つがあります。両方のカテゴリで、エージェントの追加や所有が可能で、クラスタ機能を各自のニーズに合わせて拡張することができます。

9.1 STONITHエージェント

クラスタがノードの1つの誤動作を検出し、そのノードの削除が必要となることがあります。これを「フェンシング」と呼び、一般にSTONITHリソースで実行されます。



警告: 外部SSH/STONITHはサポートされていません

SSHが他のシステムの問題にどのように反応するのかを知る方法はありません。このため、外部SSH/STONITHエージェント(`stonith:external/ssh`など)は、運用環境ではサポートされていません。テスト目的でこのようなエージェントをまだ使用する場合は、`libglue-devel` パッケージの次のアップデートで加える変更は保存されません。

現在使用可能なすべてのSTONITHデバイス(ソフトウェア側から)のリストを入手するには、`crm ra list stonith`コマンドを使用します。お気に入りのエージェントが見つからない場合は、`-devel` パッケージから削除されました。STONITHデバイスおよびリソースエージェントの詳細については、[第10章「フェンシングとSTONITH」](#)を参照してください。

今のところ、STONITHエージェントの作成に関するマニュアルはありません。新しいSTONITHエージェントを作成する場合は、`cluster-glue` パッケージの次のアップデートで加える変更は保存されません。

9.2 OCFリソースエージェントの作成

`/usr/lib/ocf/resource.d/`で提供されているすべてのOCFリソースエージェントの詳細については、[6.3.2項「サポートされるリソースエージェントクラス」](#)を参照してください。各リソースエージェントは、それを制御する次の操作をサポートしている必要があります。

`start`

リソースを開始または有効化します。

stop

リソースを中止または無効化します。

status

リソースのステータスを返します。

monitor

statusと同様ですが、予期しない状態もチェックします。

validate

リソースの設定を検証します。

meta-data

リソースエージェントの情報をXMLで返します。

OCF RAの作成方法の一般的な手順は、次のとおりです。

1. テンプレートとして、`/usr/lib/ocf/resource.d/pacemaker/Dummy` ファイルをロードします。
2. 新しいリソースエージェントごとに新しいサブディレクトリを作成して、名前が競合しないようにします。たとえばリソースグループ `kitchen` にリソース `coffee_machine` がある場合、このリソースを `/usr/lib/ocf/resource.d/kitchen/` ディレクトリに追加します。このRAにアクセスするには、コマンド `crm` を実行します。

```
root # crm configure primitive coffee_1 ocf:coffee_machine:kitchen ...
```

3. 異なるシェル関数を実装し、異なる名前ファイルでファイルを保存します。

OCFリソースエージェントの作成についての詳細は、http://www.linux-ha.org/wiki/Resource_Agents を参照してください。コンセプトの特別な情報については、第1章「製品の概要」を参照してください。

9.3 OCF戻りコードと障害回復

OCF仕様によると、アクションが返す必要がある出口コードの厳密な定義があります。クラスタは常に、予期される結果に対する戻りコードを確認します。結果が予期された値と一致しない場合、アクションは失敗したとみなされ、回復処理が開始されます。障害回復には3種類あります。

表 9.1: 障害回復の種類

回復の種類	説明	クラスタが行うアクション
soft	一時的なエラーが発生しました。	リソースを再起動するか、新しい場所に移動させます。
hard	一時的ではないエラーが発生しました。エラーは、現在のノードに固有の場合があります。	リソースを他の場所に移動して、現在のノードで再試行されないようにします。
fatal	すべてのクラスタノードに共通の、一時的ではないエラーが発生しました。これは、不正な設定が指定されたことを示しています。	リソースを停止して、どのクラスタノードでも開始されないようにします。

アクションが失敗したと想定して、次の表では、異なるOCF戻りコードを概説します。また、エラーコードを受け取った場合にクラスタが開始する回復の種類も示しています。

表 9.2: OCF戻りコード

OCF戻りコード	OCFエイリアス	説明	回復の種類
0	OCF_SUCCESS	成功。コマンドは正常に完了しました。これは、すべてのstart、stop、promote、demoteコマンドの予期された結果です。	soft
1	OCF_ERR_GENERIC	汎用の「問題が発生した」ことを示すエラーコード。	soft
2	OCF_ERR_ARGS	リソースの設定がこのマシンで有効ではありません(たとえば、ノード上に見つからない場所/ツールを参照している場合)。	hard
3	OCF_ERR_UNIMPLEMENTED	要求されたアクションは実行されていません。	hard
4	OCF_ERR_PERM	リソースエージェントに、作業を完了できるだけの権限がありません。	hard

OCF戻りコード	OCFエイリアス	説明	回復の種類
5	OCF_ERR_-INSTALLED	リソースが必要とするツールがこのコンピュータにインストールされていません。	hard
6	OCF_ERR_-CONFIGURED	リソースの設定が無効です(たとえば、必要なパラメータがないなど)。	fatal
7	OCF_NOT_-RUNNING	リソースが実行されていません。クラスタは、どのアクションについてもこれを返すリソースを停止しようとしません。 このOCF戻りコードはリソース回復を必要することも必要としないこともあります。予期されたリソースの状態に依存します。予期されない場合は、 <u>soft</u> 回復を行います。	N/A
8	OCF_RUNNING_-MASTER	リソースはマスタモードで実行しています。	soft
9	OCF_FAILED_-MASTER	リソースはマスタモードですが、失敗しました。リソースは降格、停止され、再度開始されます(昇格されます)。	soft
その他	該当なし	カスタムエラーコード。	soft

10 フェンシングとSTONITH

フェンシングはHA(High Availability)向けコンピュータクラスタにおいて、非常に重要なコンセプトです。クラスタがノードの1つの誤動作を検出し、そのノードの削除が必要となることがあります。これを「フェンシング」と呼び、一般にSTONITHリソースで実行されます。フェンシングは、HAクラスタを既知の状態にするための方法として定義できます。

クラスタのすべてのリソースには、それぞれ状態が関連付けられています。たとえば、「リソースr1はaliceで起動されている」などです。HAクラスタでは、このような状態は「リソースr1はalice以外のすべてのノードで停止している」ことを示します。クラスタは各リソースが1つのノードでのみ起動されるようにするためです。各ノードはリソースに生じた変更を報告する必要があります。つまり、クラスタの状態は、リソースの状態とノードの状態の集まりです。

ノードまたはリソースの状態を十分に確定することができない場合には、フェンシングが発生します。クラスタが所定のノードで起こっていることを認識しない場合でも、フェンシングによって、そのノードが重要なリソースを実行しないようにできます。

10.1 フェンシングのクラス

フェンシングには、リソースレベルとノードレベルのフェンシングという、2つのクラスがあります。後者について、この章で主に説明します。

リソースレベルのフェンシング

リソースレベルのフェンシングにより、特定のリソースへの排他的アクセスが保証されます。この一般的な例として、SANファイバチャネルスイッチからのノードのゾーニングの変更(つまり、ノードのディスクへのアクセスのロックアウト)や、SCSI予約などの方法が挙げられます。例については、[11.10項「ストレージ保護のための追加メカニズム」](#)を参照してください。

ノードレベルのフェンシング

ノードレベルのフェンシングにより、障害が発生したノードから共有リソースに完全にアクセスできなくなります。このことは通常、そのノードをリセットする、または電源オフにするというような、極端な手段で行われます。

10.2 ノードレベルのフェンシング

Pacemakerクラスタにおけるノードレベルフェンシングの実装は、STONITH (Shoot The Other Node in the Head: 他のノードの即時強制終了)です。High Availability Extensionには`stonith`コマンドラインツールが付属し、これはクラスタ上のノードの電源をリモートでオフにする拡張インタフェースです。使用できるオプションの概要については、`stonith --help`を実行するか、または`stonith`のマニュアルページで詳細を参照してください。

10.2.1 STONITHデバイス

ノードレベルのフェンシングを使用するには、まず、フェンシングデバイスを用意する必要があります。High Availability ExtensionでサポートされているSTONITHデバイスのリストを取得するには、任意のノード上で次のコマンドのいずれかを実行します。

```
root # stonith -L
```

あるいは、

```
root # crm ra list stonith
```

STONITHデバイスは次のカテゴリに分類できます。

電源分配装置(PDU)

電源分配装置は、重要なネットワーク、サーバ、データセンター装置の電力と機能を管理する、重要な要素です。接続した装置のリモートロード監視と、個々のコンセントでリモート電源オン/オフのための電力制御を実行できます。

無停電電源装置(UPS)

電力会社からの電力の停電発生時に別個のソースから電力を供給することで、安定した電源から接続先の装置に緊急電力が提供されます。

ブレード電源制御デバイス

クラスタを一連のブレード上で実行している場合、ブレードエンクロージャの電源制御デバイスがフェンシングの唯一の候補となります。当然、このデバイスは1台のブレードコンピュータを管理する必要があります。

ライトアウトデバイス

ライトアウトデバイス(IBM RSA、HP iLO、Dell DRAC)は急速に広まっており、既製コンピュータの標準装備になる可能性さえあります。ただし、電源をホスト(クラスタノード)と共有するため、これらはUPSデバイスに内蔵されています。ノードに電力が供給さ

れないままでは、それを制御するデバイスも役に立ちません。その場合は、CRMがノードのフェンシングの試行を無限に継続し、他のすべてのリソース操作はフェンシング/STONITH操作の完了を待機することになります。

テストデバイス

テストデバイスは、テスト専用で使用されます。通常、ハードウェアにあまり負担をかけないようにしています。クラスタが運用に使用される前に、実際のフェンシングデバイスに交換される必要があります。

STONITHデバイスは、予算と使用するハードウェアの種類に応じて選択します。

10.2.2 STONITHの実装

SUSE® Linux Enterprise High Availability ExtensionでのSTONITHの実装は、2つのコンポーネントで構成されています。

pacemaker-fenced

pacemaker-fencedは、ローカルプロセスまたはネットワーク経由でアクセスできるデーモンです。stonithdは、フェンシング操作に対応するコマンド(`rest`、`power-off`、`power-on`)を受け入れます。フェンシングデバイスのステータスチェックも行います。

pacemaker-fencedデーモンは、高可用性クラスタの各ノードで実行されます。DCノードで実行されるpacemaker-fencedインスタンスは、pacemaker-controldからフェンシング要求を受け取ります。目的のフェンシング操作を実行するのは、このインスタンスとその他のpacemaker-fencedプログラムです。

STONITHプラグイン

サポートされているフェンシングデバイスごとに、そのデバイスを制御できるSTONITHプラグインがあります。STONITHプラグインはフェンシングデバイスへのインタフェースです。STONITHプラグイン(cluster-glueパッケージに含まれています)は、各ノードの/usr/lib64/stonith/pluginsにあります。(fence-agents パッケージもインストールした場合、そこに含まれているプラグインは、/usr/sbin/fence_*にインストールされます。)すべてのSTONITHプラグインはpacemaker-fencedからは同一のものと認識されますが、フェンシングデバイスの性質を反映しているため、大きな違いがあります。

一部のプラグインは、複数のデバイスをサポートします。代表的な例はipmilan (またはexternal/ipmi)で、IPMIプロトコルを実装し、このプロトコルをサポートする任意のデバイスを制御できます。

10.3 STONITHのリソースと環境設定

フェンシングをセットアップするには、1つまたは複数のSTONITHリソースを設定する必要があります。pacemaker-fencedデーモンでは設定は不要です。すべての設定はCIBに保存されます。STONITHリソースはクラスstonithのリソースです(6.3.2項「サポートされるリソースエージェントクラス」を参照)。STONITHリソースはSTONITHプラグインのCIBでの表現です。フェンシング操作の他、STONITHリソースはその他のリソースと同様、起動、停止、監視できます。STONITHリソースの開始または停止は、ノード上でSTONITHデバイスドライバのロードおよびアンロードが行われることを意味しています。開始と停止は管理上の操作であるため、フェンシングデバイス自体での操作にはなりません。ただし、監視は、デバイスのログイン操作になります(必要な場合にデバイスが動作していることを検証するため)。STONITHリソースが別のノードにフェールオーバーすると、対応するドライバがロードされて、現在のノードがSTONITHデバイスと通信できるようにされます。

STONITHリソースはその他のリソースと同様にして設定できます。これらの操作の詳細については、使用しているクラスタ管理ツールに応じて次のいずれかを参照してください。

- Hawk2: 7.5.6項「STONITHリソースの追加」
- crmsh: 8.3.4項「STONITHリソースの作成」

パラメータ(属性)のリストは、それぞれのSTONITHの種類に依存します。特定のデバイスのパラメータ一覧を表示するには、**stonith**コマンドを実行します。

```
stonith -t stonith-device-type -n
```

たとえば、ibmhmcデバイスタイプのパラメータを表示するには、次のように入力します。

```
stonith -t ibmhmc -n
```

デバイスの簡易ヘルプテキストを表示するには、**-h**オプションを使用します。

```
stonith -t stonith-device-type -h
```

10.3.1 STONITHリソースの設定例

以降では、**crm**コマンドラインツールの構文で作成された設定例を紹介します。これを適用するには、サンプルをテキストファイル(sample.txtなど)に格納して、実行します。

```
root # crm < sample.txt
```

crmコマンドラインツールでのリソースの設定については、第8章「クラスタリソースの設定と管理(コマンドライン)」を参照してください。

例 10.1: IBM RSAライトアウトデバイスの設定

IBM RSAライトアウトデバイスは、次のようにして設定できます。

```
configure
primitive st-ibmrsa-1 stonith:external/ibmrsa-telnet \
params nodename=alice ip_address=192.168.0.101 \
username=USERNAME password=PASSWORD
primitive st-ibmrsa-2 stonith:external/ibmrsa-telnet \
params nodename=bob ip_address=192.168.0.102 \
username=USERNAME password=PASSWORD
location l-st-alice st-ibmrsa-1 -inf: alice
location l-st-bob st-ibmrsa-2 -inf: bob
commit
```

この例では、location制約が使用されていますが、それは、STONITH操作が常に一定の確率で失敗するためです。したがって、(実行側でもあるノード上の) STONITH操作は信頼できません。ノードがリセットされていない場合、フェンシング操作結果について通知を送信できません。これを実行する方法は、操作が成功すると仮定して事前に通知を送信するほかありません。ただし操作が失敗した場合、問題が発生することがあります。したがって、規則によって pacemaker-fenced はホストの終了を拒否します。

例 10.2: UPSフェンシングデバイスの設定

UPSタイプのフェンシングデバイスの設定は、上記の例と似ています。詳細についてはここでは割愛します。すべてのUPSデバイスは、フェンシングのために、同じ機構を使用します。デバイスへのアクセス方法が異なる方法。古いUPSデバイスにはシリアルポートしかなく、通常、特別のシリアルケーブルを使用して1200ボーで接続していました。新型の多くは、まだシリアルポートがありますが、USBインタフェースまたはEthernetインタフェースも使用します。使用できる接続の種類は、プラグインが何をサポートしているかによります。

たとえば、apcmasterをapcsmartデバイスと、stonith -t stonith-device-type -n コマンドを使用して比較します。

```
stonith -t apcmaster -h
```

次の情報が返されます。

```
STONITH Device: apcmaster - APC MasterSwitch (via telnet)
NOTE: The APC MasterSwitch accepts only one (telnet)
connection/session a time. When one session is active,
subsequent attempts to connect to the MasterSwitch will fail.
For more information see http://www.apc.com/
List of valid parameter names for apcmaster STONITH device:
    ipaddr
    login
    password
For Config info [-p] syntax, give each of the above parameters in order as
```

```
the -p value.  
Arguments are separated by white space.  
Config file [-F] syntax is the same as -p, except # at the start of a line  
denotes a comment
```

今度は次のコマンドを使用します。

```
stonith -t apcsmart -h
```

次の結果が得られます。

```
STONITH Device: apcsmart - APC Smart UPS  
(via serial port - NOT USB!).  
Works with higher-end APC UPSes, like  
Back-UPS Pro, Smart-UPS, Matrix-UPS, etc.  
(Smart-UPS may have to be >= Smart-UPS 700?).  
See http://www.networkupstools.org/protocols/apcsmart.html  
for protocol compatibility details.  
For more information see http://www.apc.com/  
List of valid parameter names for apcsmart STONITH device:  
ttydev  
hostlist
```

最初のプラグインは、ネットワークポートとtelnetプロトコルを持つAPC UPSをサポートします。2番目のプラグインはAPC SMARTプロトコルをシリアル回線で使用します。これは多数のAPC UPS製品ラインでサポートされているものです。

例 10.3: KDUMPデバイスの設定

Kdumpは特殊なフェンシングデバイスに属し、実際にはフェンシングデバイスとは正反対のものです。このプラグインは、ノードでカーネルダンプが進行中かどうかをチェックします。進行中であればtrueを返し、ノードがフェンシングされたかのように動作します。

Kdumpプラグインは、別の実際のSTONITHデバイスと共に使用する必要があります(たとえば、`external/ipmi`など)。フェンシングメカニズムが正常に機能するには、実際のSTONITHデバイスがトリガされる前にKdumpをチェックするよう指定する必要があります。次の手順で示すように、**`crm configure fencing_topology`**を使用して、フェンシングデバイスの順序を指定してください。

1. `stonith:fence_kdump` リソースエージェント(パッケージ `fence-agents` によって提供)を使用して、Kdump機能が有効になっているすべてのノードを監視します。構成の例については、以下のリソースを参照してください。

```
configure  
primitive st-kdump stonith:fence_kdump \  
  params nodename="alice "\ ❶  
  pcmk_host_check="static-list" \  
  pcmk_reboot_action="off" \  
  pcmk_monitor_action="metadata" \  
  \
```

```
pcmk_reboot_retries="1" \  
timeout="60"  
commit
```

- ① 監視されるノードの名前。複数のノードを監視する必要がある場合は、追加のSTONITHリソースを設定します。特定のノードでフェンシングデバイスを使用しないようにするには、場所に対する制約を追加します。

フェンシングのアクションは、リソースのタイムアウトが経過すると始まります。

2. 各ノード上の`/etc/sysconfig/kdump`で、`kdump`プロセスが完了したときにすべてのノードに通知が送信されるように`KDUMP_POSTSCRIPT`を設定します。例:

```
/usr/lib/fence_kdump_send -i INTERVAL -p PORT -c 1 alice bob charlie [...]
```

`kdump`が完了すると、`kdump`を実行するノードが自動的に再起動します。

3. ネットワークが有効化された`fence_kdump_send`ライブラリに関する指定を含む、新しい`initrd`を記述します。`-f`オプションを使用して既存のファイルを上書きし、次のブートプロセスでその新規ファイルが使用されるようにします。

```
root # dracut -f -a kdump
```

4. `fence_kdump`リソース用のポートをファイアウォールで開きます。デフォルトポートは7410です。
5. 実際のフェンシングメカニズム(`external/ipmi`など)をトリガする前に`Kdump`がチェックされるようにするため、次のような設定を使用します。

```
fencing_topology \  
  alice: kdump-node1 ipmi-node1 \  
  bob: kdump-node2 ipmi-node2
```

`fencing_topology`の詳細:

```
crm configure help fencing_topology
```

10.4 フェンシングデバイスの監視

他のリソースと同様に、STONITHクラスのエージェントは、ステータスのチェックのための監視操作もサポートします。



注記: STONITHリソースの監視

STONITHリソースの監視は定期的に行われますが、頻繁ではありません。ほとんどのデバイスでは、少なくとも1800秒(30分)の監視間隔があれば十分です。

フェンシングデバイスはHAクラスタの不可欠な要素ですが、使用する必要が少ないほど好都合です。ブロードキャストトラフィックが多すぎると、しばしば、電源管理装置が影響を受けます。1分間に10本程度の接続しか処理できないデバイスもあります。2つのクライアントが同時に接続しようとする、混乱するデバイスもあります。大部分は、同時に複数のセッションを処理できません。

したがって、通常、フェンシングデバイスのステータスは数時間ごとにチェックすれば十分です。フェンシング操作の実行が必要となり、電源スイッチが故障する可能性は小さいものです。

監視操作の設定方法の詳細については、[8.3.9項「リソース監視の設定」](#)を参照してください(コマンドラインアプローチについて説明されている)。

10.5 特殊なフェンシングデバイス

実際のSTONITHデバイスを操作するプラグインに加えて、特殊目的のSTONITHプラグインも存在します。



警告: テスト専用

次に示すSTONITHプラグインの一部は、デモとテストだけを目的としています。次のデバイスは、現実のシナリオでは使用しないでください。使用すると、データが損なわれたり、予測できない結果が生じることがあります。

- [external/ssh](#)
- [ssh](#)


[fence_kdump](#)

このプラグインは、ノードでカーネルダンプが進行中かどうかをチェックします。進行中であれば`true`を返し、ノードがフェンシングされたかのように動作します。いずれにせよ、ダンプ中には、ノードはどのリソースも実行できません。これによって、すでに

ダウンしているがダンプ中(これは時間がかかります)であるノードのフェンシングを避けることができます。このプラグインは、別の実際のSTONITHデバイスとともに使用する必要があります。

設定の詳細については、[例10.3「kdumpデバイスの設定」](#)を参照してください。

external/sbd

これは自己フェンシングデバイスです。共有ディスクに挿入されることがある、いわゆる「ポイズンピル」に反応します。共有ストレージ接続が失われた場合、このデバイスはノードの動作を停止します。このSTONITHエージェントを使用してストレージベースのフェンシングを実装する方法については、[第11章、手順11.7「SBDを使用するようにクラスタを設定する」](#)を参照してください。詳細については、http://www.linux-ha.org/wiki/SBD_Fencing も参照してください。

重要: external/sbdおよびDRBD

external/sbd フェンシングメカニズムは、SBDパーティションが各ノードから直接読み取れることを要求します。そのため、SBDパーティションではDRBD*デバイスを使用してはなりません。

ただし、SBDパーティションが階層配置または複製されていない共有ディスク上にある場合には、DRBDクラスタでフェンシングメカニズムを使用することはできません。

external/ssh

別のソフトウェアベースの「フェンシング」メカニズムです。ノードは、rootとして、パスワードなしでお互いにログインする必要があります。このメカニズムは、1つのパラメータ hostlist をとり、ターゲットにするノードを指定します。これは、本当に障害のあるノードをリセットすることはできないので、実際のクラスタには使用しないでください。これは、テストとデモの目的にのみ使用します。これを共有ストレージに使用すると、データが破損します。

meatware

meatware ではユーザが操作を支援する必要があります。起動すると、meatware はノードのコンソールに表示されるCRIT重大度メッセージを記録します。その場合、オペレータはノードがダウンしていることを確認して、**meatclient(8)** コマンドを発行します。これにより meatware は、クラスタに対して、そのノードが機能しなくなっていることを伝えます。詳細については、[/usr/share/doc/packages/cluster-glue/README.meatware](#) を参照してください。

suicide

これはソフトウェアのみのデバイスで、**reboot**コマンドを使用して実行しているノードを再起動できます。これにはノードのオペレーティングシステムによる操作が必要で、特定の状況では失敗することがあります。したがって、このデバイスの使用は、極力避けてください。ただし、1ノードクラスタで使用する場合は安全です。

ディスクレスSBD

この設定は、共有ストレージなしのフェンシングメカニズムが必要なときに便利です。このディスクレスモードでは、SBDは共有デバイスに頼らず、ハードウェアウォッチドッグを使用してノードをフェンスします。ただし、ディスクレスSBDは2ノードクラスタ用のスプリットブレインシナリオには対応できません。このオプションは、「3つ以上」のノードを持つクラスタにのみ使用してください。

suicideは、「自分のホストを停止させない」というルールに対する唯一の例外です。

10.6 基本的な推奨事項

次の推奨事項のリストをチェックして、よく発生する間違いを避けてください。

- 複数の電源スイッチを並列に接続しないでください。
- STONITHデバイスとその設定をテストする際には、各ノードからプラグを1回抜いて、ノードのフェンシングが起こらないことを検証してください。
- リソースのテストは負荷のかかった状態で行って、タイムアウト値が適切であるかどうかを検証してください。タイムアウト値が短すぎると、(不必要な)フェンシング操作がトリガされることがあります。詳細については、[6.3.9項「タイムアウト値」](#)を参照してください。
- セットアップでは適切なフェンシングデバイスを使用してください。詳細については、[10.5項「特殊なフェンシングデバイス」](#)も参照してください。
- 1つ以上のSTONITHリソースを設定します。デフォルトでは、グローバルクラスタオプションの`stonith-enabled`は`true`に設定されています。STONITHリソースが定義されていない場合、クラスタはどのリソースも開始することを拒否します。STONITHリソースが定義されていない場合、クラスタはどのリソースを開始することも拒否します。
- グローバルクラスタオプション`stonith-enabled`を`false`に設定しないでください。これは、次の理由によります。

- STONITHが有効でないクラスタはサポートされていません。
- DLM/OCFS2は、決して発生しないフェンシング操作を待機して、永久にブロックし続けます。
- グローバルクラスタオプション`startup-fencing`を`false`に設定しないでください。デフォルトでは、これは次の理由で`true`に設定されています。クラスタの起動時に、あるノードが不明な状態になっていると、そのノードは、ステータスが明らかにされるまでフェンシングされます。

10.7 詳細の参照先

</usr/share/doc/packages/cluster-glue>

インストールしたシステムのこのディレクトリには、多数のSTONITHプラグインおよびデバイスのREADMEファイルが格納されています。

<http://www.linux-ha.org/wiki/STONITH> 

STONITHに関する情報。High Availability Linux Projectのホームページにあります。

<http://www.clusterlabs.org/pacemaker/doc/> 

- 『Pacemakerの説明』: Pacemakerの設定に必要なコンセプトを説明します。包括的で詳しい参照用情報です。

http://techthoughts.typepad.com/managing_computers/2007/10/split-brain-quo.html 

HAクラスタでのスプリットブレイン、クォーラム、フェンシングのコンセプトを説明する記事。

11 ストレージ保護とSBD

SBD (STONITH Block Device)は、共有ブロックストレージ(SAN、iSCSI、FCoEなど)を介したメッセージの交換を通じて、Pacemakerベースのクラスタのノードフェンシングメカニズムを提供します。これにより、フェンシングメカニズムが、ファームウェアバージョンの変更や特定のファームウェアコントローラへの依存から切り離されます。動作異常のノードが本当に停止したかどうかを確認するために、各ノードではウォッチドッグが必要です。特定の条件下では、ディスクレスモードで実行することにより、共有ストレージなしでSBDを使用することもできます。

`sapMDCsapVirtHostname ha-cluster-bootstrap` スクリプトは、フェンシングメカニズムとしてSBDを使用するオプションを用いて、クラスタを設定する自動化された方法を提供します。詳細については、項目「インストールおよびセットアップクイックスタート」を参照してください。ただし、SBDを手動で設定する場合、個々の設定に関するオプションが増えます。

この章では、SBDの背後にある概念について説明します。スプリットブレインシナリオの場合に潜在的なデータ破損からクラスタを保護するために、SBDが必要とするコンポーネントを設定する手順を説明します。

ノードレベルのフェンシングに加えて、LVM2排他アクティブ化やOCFS2ファイルロックのサポート(リソースレベルのフェンシング)など、ストレージ保護のための追加のメカニズムを使用することができます。これにより、管理上またはアプリケーション上の障害からシステムが保護されます。

11.1 概念の概要

SBDは、「Storage-Based Death」または「STONITHブロックデバイス」の略語です。

High Availabilityクラスタスタックの最優先事項は、データの整合性を保護することです。これは、データストレージへの非協調的な同時アクセスを防止することによって実現されます。クラスタスタックは、複数の制御メカニズムを使用してこの処理を行います。

ただし、ネットワークのパーティション分割やソフトウェアの誤動作により、クラスタでいくつものDCが選択される状況となる可能性があります。このいわゆるスプリットブレインシナリオが発生した場合は、データが破損することがあります。

STONITHによるノードフェンシングは、これを防ぐためのプライマリメカニズムです。ノードフェンシングメカニズムとしてSBDを使用することは、スプリットブレインシナリオの場合に、外部電源オフデバイスを使用せずにノードをシャットダウンする1つの方法です。

SBDのコンポーネントとメカニズム

SBDパーティション

すべてのノードが共有ストレージへのアクセスを持つ環境で、デバイスの小さなパーティションをSBDで使用できるようにフォーマットします。パーティションのサイズは、使用されるディスクのブロックサイズによって異なります(たとえば、512バイトのブロックサイズの標準SCSIディスクには1MB、4KBブロックサイズのDASDディスクには4MB必要です)。初期化プロセスでは、最大255のノードに対するスロットを備えたデバイス上にメッセージレイアウトが作成されます。

SBDデーモン

SBDは、そのデーモンの設定後、クラスタスタックの他のコンポーネントが起動される前に各ノードでオンラインになります。SBDデーモンは、他のすべてのクラスタコンポーネントがシャットダウンされた後で終了されます。したがって、クラスタリソースがSBDの監督なしでアクティブになることはありません。

メッセージ

このデーモンは、自動的に、パーティション上のメッセージスロットの1つを自分自身に割り当て、自分へのメッセージがないかどうか、そのスロットを絶えず監視します。デーモンは、メッセージを受信すると、ただちに要求に従います(フェンシングのための電源切断や再起動サイクルの開始など)。

また、デーモンは、ストレージデバイスへの接続性を絶えず監視し、パーティションが到達不能になった場合は、デーモン自体が終了します。このため、デーモンがフェンシングメッセージから切断されることはありません。これは、クラスタデータが別のパーティション上の同じ論理ユニットにある場合、追加障害ポイントになることはありません。ストレージ接続を失えば、ワークロードは終了します。

ウォッチドッグ

SBDを使用する場合は常に、正常動作するウォッチドッグが不可欠です。近代的なシステムは、ソフトウェアコンポーネントによって「チックル」または「フィード」される必要のある「hardware watchdog」をサポートします。ソフトウェアコンポーネント(この場合、SBDデーモン)は、ウォッチドッグにサービスパルスを定期的書き込むことによって、ウォッチドッグに「フィード」します。デーモンがウォッチドッグへのフィードを停止すると、ハードウェアでシステムが強制的に再起動されます。この機能は、SBDプロセス自体の障害(I/Oエラーで終了またはスタックするなど)に対する保護を提供します。

Pacemaker統合が有効になっている場合、デバイスの過半数が失われてもSBDはセルフフェンスを行いません。たとえば、クラスタにA、B、Cの3つのノードが含まれており、ネットワーク分割によってAには自分自身しか表示できず、BとCはまだ通信可能な状態であるとして、この場合、2つのクラスタパーティションが存在し、1つは過半数(B、C)であるためにクォーラムがあり、もう1つにはクォーラムがない(A)ことになります。過半数のフェンシングデバイスに到達できないときにこれが発生した場合、ノードAはすぐに自らダウンしますが、BとCは引き続き実行されます。

11.2 SBDの手動設定の概要

手動でストレージベースの保護を設定するには、次の手順に従う必要があります。これらは`root`として実行する必要があります。開始する前に、[11.3項「要件」](#)を確認してください。

1. ウォッチドッグのセットアップ
2. シナリオに応じて、1〜3台のデバイスとともにまたはディスクレスモードでSBDを使用してください。概要については、[11.4項「SBDデバイスの数」](#)を参照してください。詳細な設定については、以下に記載されています。
 - デバイスでのSBDの設定
 - ディスクレスSBDの設定
3. SBDとフェンシングのテスト

11.3 要件

- ストレージベースのフェンシングには、最大3つのSBDデバイスを使用できます。1〜3台のデバイスを使用する場合、共有ストレージにすべてのノードからアクセス可能である必要があります。
- 共有ストレージデバイスのパスが永続的で、クラスタ内のすべてのノードで一致している必要があります。`/dev/disk/by-id/dm-uuid-part1-mpath-abcdef12345`などの固定デバイス名を使用してください。
- 共有ストレージはFC (ファイバチャネル)、FCoE (Fibre Channel over Ethernet)、またはiSCSI経由で接続できます。

- 共有ストレージセグメントが、ホストベースのRAID、LVM2、またはDRBD*を「使用してはなりません」。DRBDは分割できますが、SBDでは2つの状態が存在することはできないため、これはSBDにとって問題になります。クラスタマルチデバイス(クラスタMD)は、SBDには使用できません。
- ただし、信頼性向上のため、ストレージベースのRAIDとマルチパスの使用は推奨されます。
- 255を超えるノードでデバイスを共有しない限り、異なるクラスタ間でSBDデバイスを共有できます。
- 3つ以上のノードがあるクラスタの場合は、SBDを「ディスクレス」モードで 사용할こともできます。

11.4 SBDデバイスの数

SBDは、最大3つのデバイスの使用をサポートしています。

1つのデバイス

最も単純な実装です。すべてのデータが同じ共有ストレージ上にあるクラスタに適しています。

2つのデバイス

この設定は、主に、ホストベースのミラーリングを使用しているものの3つ目のストレージデバイスが使用できない環境で役立ちます。1つのミラーログにアクセスできなくなっても、SBDは終了せず、クラスタは引き続き実行できます。ただし、SBDにはストレージの非同期分割を検出できるだけの情報が与えられていないので、ミラーログが1つだけ使用可能なときにもう一方をフェンスすることはありません。つまり、ストレージアレイのいずれかがダウンしたときに、2つ目の障害に自動的に耐えることはできません。

3つのデバイス

最も信頼性の高い設定です。障害または保守による1台のデバイスの機能停止から回復できます。複数のデバイスが失われた場合、およびクラスタパーティションまたはノードの状態に応じて必要な場合にのみ、SBD自体が終了します。少なくとも2つのデバイスにまだアクセス可能な場合は、フェンシングメッセージを正常に送信できます。

この設定は、ストレージが1つのアレイに制約されていない、比較的複雑なシナリオに適しています。ホストベースのミラーリングソリューションでは、1つのミラーログに1つのSBDを設定し(自分自身はミラーしない)、iSCSI上に追加のタイブレーカを設定できます。

ディスクリス

この設定は、共有ストレージなしのフェンシングメカニズムが必要なときに便利です。このディスクリスモードでは、SBDは共有デバイスに頼らず、ハードウェアウォッチドッグを使用してノードをフェンスします。ただし、ディスクリスSBDは2ノードクラスタ用のスプリットブレインシナリオには対応できません。このオプションは、「3つ以上」のノードを持つクラスタにのみ使用してください。

11.5 タイムアウトの計算

フェンシングメカニズムとしてSBDを使用する場合、すべてのコンポーネントのタイムアウトを考慮することが重要です。それらのコンポーネントが相互に依存するためです。

ウォッチドッグのタイムアウト

このタイムアウトは、SBDデバイスの初期化中に設定されます。これは主にストレージのレイテンシに依存します。この時間内に大半のデバイスを正常に読み込む必要があります。それができない場合、そのノードでセルフフェンスを行うことがあります。



注記: マルチパスまたはiSCSIセットアップ

マルチパスセットアップまたはiSCSI上にSBDデバイスがある場合、パスの障害を検出して次のパスに切り替えるのに必要な時間に、タイムアウトを設定する必要があります。

またこれは、`/etc/multipath.conf`で`max_polling_interval`の値がウォッチドッグのタイムアウト未満でなければならないことを意味します。

msgwaitタイムアウト

このタイムアウトは、SBDデバイスの初期化中に設定されます。この時間が経過するとSBDデバイス上のノードのスロットに書き込まれたメッセージが配信されたとみなされる時間を定義します。タイムアウトは、ノードでセルフフェンスを行う必要があることを検出するのに十分な長さでなければなりません。

ただし、msgwaitタイムアウトが比較的長い場合、フェンシングアクションが戻る前にフェンスされたクラスタノードが再加入することがあります。これは、[手順 11.4](#)の[ステップ 4](#)で説明するように、SBD設定のSBD_DELAY_STARTパラメータを設定することで、緩和することができます。

CIBの`stonith-timeout`

このタイムアウトは、グローバルクラスタプロパティとしてCIBで設定されます。これは、STONITHアクション(再起動、オン、オフ)が完了するのを待つ時間の長さを定義します。

CIBの`stonith-watchdog-timeout`

このタイムアウトは、グローバルクラスタプロパティとしてCIBで設定されます。明示的に設定されていない場合は、デフォルトで0に設定されます。これは1〜3台のデバイスとともにSBDを使用するのに適しています。ディスクレスモードでSBDを使用する方法の詳細については、[手順11.8「ディスクレスSBDの設定」](#)を参照してください。

ウォッチドッグのタイムアウトを変更する場合は、他の2つのタイムアウトも調整する必要があります。次の「式」は、これら3つの値の関係を示しています。

例 11.1: タイムアウト計算の式

```
Timeout (msgwait) >= (Timeout (watchdog) * 2)
stonith-timeout = Timeout (msgwait) + 20%
```

たとえば、ウォッチドッグのタイムアウトを120に設定した場合、`msgwait`タイムアウトを240に設定し、`stonith-timeout`を288に設定します。

`ha-cluster-bootstrap` スクリプトを使用してクラスタを設定し、SBDデバイスを初期化する場合、これらのタイムアウト間の関係が自動的に考慮されます。

11.6 ウォッチドッグのセットアップ

SUSE Linux Enterprise High Availability Extensionには、ハードウェア固有のウォッチドッグドライバを提供する、いくつかのカーネルモジュールが付属しています。最もよく使用されるカーネルモジュールのリストについては、[よく使用されるウォッチドッグドライバ](#)を参照してください。

運用環境のクラスタでは、ハードウェア固有のウォッチドッグドライバを使用することをお勧めします。ただし、ハードウェアに適合するウォッチドッグがない場合、カーネルウォッチドッグモジュールとして`softdog`を使用することができます。

High Availability Extensionはウォッチドッグに「フィード」するソフトウェアコンポーネントとしてSBDデーモンを使用します。

11.6.1 ハードウェアウォッチドッグの使用

特定のシステムの正しいウォッチドッグカーネルモジュールを判断することは、容易ではありません。自動プロービングは頻繁に失敗します。その結果、正しいモジュールがロードされる前に、多くのモジュールがすでにロードされている状態になってしまいます。

表 11.1は、最もよく使用されるウォッチドッグドライバのリストです。ご使用のハードウェアがそこに一覧表示されていない場合は、ディレクトリ `/lib/modules/KERNEL_VERSION/kernel/drivers/watchdog` または `/lib/modules/KERNEL_VERSION/kernel/drivers/ipmi` にも選択肢のリストが表示されます。または、システム固有のウォッチドッグ設定の詳細について、ハードウェアまたはシステムのベンダーに問い合わせてください。

表 11.1: よく使用されるウォッチドッグドライバ

Hardware (ハードウェア)	ドライバ
HP	<code>hpwdt</code>
Dell、Lenovo (Intel TCO)	<code>iTCO_wdt</code>
富士通	<code>ipmi_watchdog</code>
IBMメインフレーム上のz/VMのVM	<code>vmwatchdog</code>
Xen VM (DomU)	<code>xen_xdt</code>
Generic	<code>softdog</code>

！ 重要: ウォッチドッグタイマへのアクセス

一部のハードウェアベンダーは、システムのリセット用にウォッチドッグを使用するシステム管理ソフトウェアを提供しています(たとえば、HP ASRデーモンなど)。ウォッチドッグがSBDで使用されている場合は、このようなソフトウェアを無効にします。他のソフトウェアは、ウォッチドッグタイマにアクセスしないでください。

手順 11.1: 正しいカーネルモジュールのロード

正しいウォッチドッグモジュールがロードされていることを確認するには、次の手順を実行します。

1. お使いのカーネルバージョンでインストールされているドライバをリストします。

```
root # rpm -ql kernel-VERSION | grep watchdog
```

2. カーネルに現在ロードされているウォッチドッグモジュールをリストします。

```
root # lsmod | egrep "(wd|dog)"
```

3. 結果が表示されたら、間違ったモジュールをアンロードします。

```
root # rmmod WRONG_MODULE
```

4. お使いのハードウェアに適合するウォッチドッグモジュールを有効にします。

```
root # echo WATCHDOG_MODULE > /etc/modules-load.d/watchdog.conf  
root # systemctl restart systemd-modules-load
```

5. ウォッチドッグモジュールが正しくロードされているかどうかをテストします。

```
root # lsmod | grep dog
```

6. ウォッチドッグデバイスが使用可能で機能しているかどうかを確認します。

```
root # ls -l /dev/watchdog*  
root # sbd query-watchdog
```

ウォッチドッグデバイスが使用できない場合は、ここで停止して、モジュール名とオプションを確認します。別のドライバを使用してみるのもいいかもしれません。

7. ウォッチドッグデバイスが機能しているかどうかを確認します。

```
root # sbd -w WATCHDOG_DEVICE test-watchdog
```

8. マシンを再起動して、カーネルモジュールが競合していないことを確認します。たとえば、ログに `cannot register ...` というメッセージが表示される場合は、このような競合するモジュールを示しています。このようなモジュールを無視するには、<https://documentation.suse.com/sles/html/SLES-all/cha-mod.html#sec-mod-modprobe-blacklist> を参照してください。

11.6.2 ソフトウェアウォッチドッグ(softdog)の使用

運用環境のクラスタでは、ハードウェア固有のウォッチドッグドライバを使用することをお勧めします。ただし、ハードウェアに適合するウォッチドッグがない場合、カーネルウォッチドッグモジュールとして `softdog` を使用することができます。

！ 重要: softdogの制限

softdogドライバはCPUが最低1つは動作中であることを前提とします。すべてのCPUが固まっている場合、システムを再起動させるsoftdogドライバのコードは実行されません。これに対して、ハードウェアウォッチドッグはすべてのCPUが固まっても動作し続けます。

手順 11.2: SOFTDOGカーネルモジュールのロード

1. softdogドライバを有効にします。

```
root # echo softdog > /etc/modules-load.d/watchdog.conf
```

2. `/etc/modules-load.d/watchdog.conf`に`softdog`モジュールを追加し、サービスを再起動します。

```
root # echo softdog > /etc/modules-load.d/watchdog.conf
root # systemctl restart systemd-modules-load
```

3. softdogウォッチドッグモジュールが正しくロードされているかどうかをテストします。

```
root # lsmod | grep softdog
```

11.7 デバイスでのSBDの設定

セットアップには次の手順が必要です。

1. SBDデバイスの初期化
2. SBD設定ファイルの編集
3. SBDサービスの有効化と起動
4. SBDデバイスのテスト
5. SBDを使用するようにクラスタを設定する

開始する前に、SBDに使用するブロックデバイスが、[11.3項](#)で指定された要件を満たしていることを確認してください。

SBDデバイスを設定するときは、いくつかのタイムアウト値を考慮する必要があります。詳細については、[11.5項「タイムアウトの計算」](#)を参照してください。

ノード上で実行しているSBDデーモンがウォッチドッグタイマを十分な速さで更新していない場合、ノード自体が終了します。タイムアウトを設定したら、個別の環境でテストしてください。

手順 11.3: SBDデバイスの初期化

共有ストレージでSBDを使用するには、まず1~3台のブロックデバイス上でメッセージングレイアウトを作成する必要があります。`sbd create`コマンドは、指定された1つまたは複数のデバイスにメタデータヘッダを書き込みます。また、最大255ノードのメッセージングスロットを初期化します。追加のオプションを指定せずに実行する場合、このコマンドはデフォルトのタイムアウト設定を使用します。



警告: 既存データの上書き

SBD用に使用するデバイスには、重要なデータが一切ないようにしてください。`sbd create`コマンドを実行すると、指定されたブロックデバイスの最初のメガバイトが、さらなる要求やバックアップなしに上書きされます。

1. SBDに使用するブロックデバイスを決定します。
2. 次のコマンドで、SBDデバイスを初期化します。

```
root # sbd -d /dev/SBD create
```

(`/dev/SBD`を実際のパス名で置き換えます。たとえば`/dev/disk/by-id/scsi-ST2000DM001-0123456_Wabcdefg`です)。

SBDに複数のデバイスを使用するには、`-d`オプションを複数回指定します。たとえば、次のようになります。

```
root # sbd -d /dev/SBD1 -d /dev/SBD2 -d /dev/SBD3 create
```

3. SBDデバイスがマルチパスグループにある場合は、`-1`と`-4`オプションを使用して、SBDに使用するタイムアウトを調整します。詳細については、[11.5項「タイムアウトの計算」](#)を参照してください。タイムアウトはすべて秒単位で指定します。

```
root # sbd -d /dev/SBD -4 180 ① -1 90 ② create
```

- ① `-4`オプションは`msgwait`タイムアウトを指定するために使用されます。上の例では、180秒に設定されます。

- ② `-l`オプションはwatchdogタイムアウトを指定するために使用されます。上の例では、90秒に設定されます。エミュレートされたウォッチドッグで使用可能な最小値は15秒です。

4. デバイスに書き込まれた内容を確認します。

```
root # sbd -d /dev/SBD dump
Header version      : 2.1
UUID                : 619127f4-0e06-434c-84a0-ea82036e144c
Number of slots     : 255
Sector size         : 512
Timeout (watchdog)  : 5
Timeout (allocate)  : 2
Timeout (loop)      : 1
Timeout (msgwait)   : 10
==Header on disk /dev/SBD is dumped
```

ご覧のように、タイムアウトがヘッダにも保存され、それらに関するすべての参加ノードの合意が確保されます。

SBDデバイスを初期化したら、SBD設定ファイルを編集し、次にそれぞれのサービスを有効にして起動し、変更を有効にします。

手順 11.4: SBD設定ファイルの編集

1. ファイル `/etc/sysconfig/sbd`を開きます。
2. 次のパラメータを検索します。 `SBD_DEVICE`をインストールします。
SBDメッセージを交換するために監視および使用するデバイスを指定します。
3. `SBD`をお使いのSBDデバイスに置き換えて、この行を編集します。

```
SBD_DEVICE="/dev/SBD"
```

1行目で複数のデバイスを指定する必要がある場合は、セミコロンで区切って指定します(デバイスの順序は任意で構いません)。

```
SBD_DEVICE="/dev/SBD1;/dev/SBD2;/dev/SBD3"
```

SBDデバイスがアクセス不能な場合は、SBDデーモンが開始できなくなり、クラスタの起動を抑制します。

4. 次のパラメータを検索します。 `SBD_DELAY_START`をインストールします。

遅延を有効または無効にします。設定 `SBD_DELAY_START` を `yes` に設定します (`msgwait` が比較的長い場合)。ただし、クラスタノードは非常に高速に起動します。このパラメータを `yes` に設定すると、ブート時に SBD の起動が遅れます。これは、仮想マシンで必要となることがあります。

SBD デバイスを SBD 設定ファイルに追加したら、SBD デーモンを有効にします。SBD デーモンは、クラスタスタックの不可欠なコンポーネントです。これは、クラスタスタックが実行されているときに、実行されている必要があります。したがって、`sbd` サービスは、`pacemaker` サービスが開始されるたびに依存関係として開始されます。

手順 11.5: SBD サービスの有効化と起動

1. 各ノードで、SBD サービスを有効にします。

```
root # systemctl enable sbd
```

これは、Pacemaker サービスが開始されるたびに、Corosync サービスと一緒に開始されます。

2. 各ノードでクラスタスタックを再起動します。

```
root # crm cluster restart
```

これによって、自動的に SBD デーモンの開始がトリガされます。

次の手順として、[手順 11.6](#)の説明に従って SBD デバイスをテストします。

手順 11.6: SBD デバイスのテスト

1. 次のコマンドを使用すると、ノードスロットとそれらの現在のメッセージが SBD デバイスからダンプされます。

```
root # sbd -d /dev/SBD list
```

ここで SBD を使用して起動したすべてのクラスタノードが表示されます。たとえば、2 ノードクラスタを使用している場合、両方のノードのメッセージスロットには `clear` と表示されます。

0	alice	clear
1	bob	clear

2. ノードの 1 つにテストメッセージを送信してみます。

```
root # sbd -d /dev/SBD message alice test
```

3. ノードがシステムログファイルにメッセージの受信を記録します。

```
May 03 16:08:31 alice sbd[66139]: /dev/SBD: notice: servant: Received command test
from bob on disk /dev/SBD
```

これによって、SBDがノード上で実際に機能し、メッセージを受信できることが確認されます。

最後のステップとして、[手順 11.7](#)の説明に従ってクラスタ設定を調整する必要があります。

手順 11.7: SBDを使用するようにクラスタを設定する

クラスタでSBDの使用を設定するには、クラスタ設定で次の操作を行う必要があります。

- 設定に適合する値に `stonith-timeout` パラメータを設定します。
- SBD STONITHリソースを設定します。

`stonith-timeout` の計算については、[11.5項「タイムアウトの計算」](#)を参照してください。

1. シェルを起動し、`root`または同等のものとしてログインします。
2. `crm configure`を実行します。
3. 次のように入力します。

```
crm(live)configure# property stonith-enabled="true" ①
crm(live)configure# property stonith-watchdog-timeout=0 ②
crm(live)configure# property stonith-timeout="40s" ③
```

- ① STONITHを使用しないクラスタはサポートされていないため、これがデフォルト設定になります。ただし、テスト目的でSTONITHが無効化されている場合は、再度このパラメータが`true`に設定されていることを確認してください。
 - ② 明示的に設定されていない場合、この値はデフォルトで0に設定されます。これは1～3台のデバイスとともにSBDを使用するのに適しています。
 - ③ SBDの`msgwait`タイムアウト値が30秒に設定されていた場合、`stonith-timeout`値は40が適切です。
4. 2ノードクラスタで、スプレッドブレインの場合、想定どおりに各ノードから他のノードにフェンシングが発行されます。両方のノードがほぼ同時にリセットされないようにするためには、次のフェンシング遅延を適用して、ノードのいずれか、または優先ノードが、フェンシングマッチに勝利するようにすることをお勧めします。3つ以上のノードを持つクラスタの場合は、これらの遅延を適用する必要はありません。

優先フェンシング遅延

`priority-fencing-delay` クラスタプロパティはデフォルトで無効になっています。遅延値を設定することにより、他のノードが失われ、リソース全体の優先度が高い場合に、そのノードをターゲットとするフェンシングが指定された時間だけ遅延されます。これは、スプリットブレインの場合、より重要なノードがフェンシングマッチに勝利することを意味します。

重要なリソースは優先メタ属性を使用して設定できます。計算時に、各ノードで実行されているリソースまたはインスタンスの優先度の値が合計されて考慮されます。昇格されたリソースインスタンスは、設定された基本優先度に1を加えたものになるため、昇格されていないインスタンスよりも高い値を受け取ります。

```
root # crm configure property priority-fencing-delay=30
```

`priority-fencing-delay` が使用される場合でも、ノードの優先度が等しい状況に対処するために、以下に説明するように、`pcmk_delay_base` または `pcmk_delay_max` を使用することもお勧めします。`priority-fencing-delay` の値は、`pcmk_delay_base / pcmk_delay_max` の最大値よりも大幅に大きく、できれば最大値の2倍にする必要があります。

予測可能な静的遅延

このパラメータはSTONITHアクションを実行する前に静的遅延を追加します。2ノードクラスタのスプリットブレイン下で同時にノードがリセットしないようにするために、遅延値が異なる別のフェンシングリソースを設定します。優先ノードは、より長いフェンシング遅延をターゲットとするパラメータでマーク付けすることができるため、フェンシングマッチに勝利します。これを正常に実行するには、各ノードに2つのプリミティブSTONITHデバイスを作成することが必須です。次の設定では、スプリットブレインシナリオの場合に、aliceが勝利して存続します。

```
crm(live)configure# primitive st-sbd-alice stonith:external/sbd params \
    pcmk_host_list=alice pcmk_delay_base=20
crm(live)configure# primitive st-sbd-bob stonith:external/sbd params \
    pcmk_host_list=bob pcmk_delay_base=0
```

動的なランダム遅延

このパラメータを使用すると、フェンシングデバイス上でSTONITHアクションに対するランダム遅延が追加されます。`pcmk_delay_max` パラメータは、特定のノードをターゲットとする静的な遅延ではなく、フェンシングリソース

を使用してフェンシングにランダム遅延を追加して、二重リセットを防止します。pcmk_delay_baseとは異なり、このパラメータは複数のノードをターゲットとする統一されたフェンシングリソースに対して指定できます。

```
crm(live)configure# primitive stonith_sbd stonith:external/sbd
params pcmk_delay_max=30
```

5. **show**で変更内容をレビューします。

6. **commit**で変更を送信し、**exit**でcrmライブ設定を終了します。

リソースの起動後、SBDを使用するためにクラスタが正常に設定されます。ノードをフェンスする必要がある場合にこの方法を使用します。

11.8 ディスクレスSBDの設定

ディスクレスモードでSBDを動作させることができます。このモードでは、次の場合にウォッチドッグデバイスを使用してノードをリセットします。クォーラムが失われた場合、監視されているデーモンが失われて回復しなかった場合、またはノードでフェンシングが必要であるとPacemakerが判断した場合。ディスクレスSBDは、クラスタの状態、クォーラム、およびいくつかの合理的な前提に応じた、ノードの「セルフフェンシング」に基づいています。STONITH SBDリソースプリミティブはCIBでは必要ありません。

！ 重要: クラスタノード数

2ノードクラスタのフェンシングメカニズムとしてディスクレスSBDを使用「しないでください」。3つ以上のノードを持つクラスタにのみディスクレスSBDを使用してください。ディスクレスモードのSBDでは、2ノードクラスタのスプリットブレインシナリオを処理できません。2ノードクラスタにディスクレスSBDを使用する場合は、[第12章「QDeviceとQNetd」](#)で説明するようにQDeviceを使用します。

手順 11.8: ディスクレスSBDの設定

1. ファイル `/etc/sysconfig/sbd`を開き、次のエントリを使用します。

```
SBD_PACEMAKER=yes
SBD_STARTMODE=always
SBD_DELAY_START=no
SBD_WATCHDOG_DEV=/dev/watchdog
SBD_WATCHDOG_TIMEOUT=5
```


共有ディスクが使用されないため、SBD_DEVICEエントリは必要ありません。このパラメータがない場合、sbddサービスはSBDデバイスのウォッチャプロセスを開始しません。

2. 各ノードで、SBDサービスを有効にします。

```
root # systemctl enable sbd
```

これは、Pacemakerサービスが開始されるたびに、Corosyncサービスと一緒に開始されます。

3. 各ノードでクラスタスタックを再起動します。

```
root # crm cluster restart
```

これによって、自動的にSBDデーモンの開始がトリガされます。

4. パラメータ have-watchdog=true が自動的に設定されているかどうかを確認します。

```
root # crm configure show | grep have-watchdog
have-watchdog=true
```

5. **crm configure**を実行し、crmシェルで次のクラスタプロパティを設定します。

```
crm(live)configure# property stonith-enabled="true" ①
crm(live)configure# property stonith-watchdog-timeout=10 ②
```

- ① STONITHを使用しないクラスタはサポートされていないため、これがデフォルト設定になります。ただし、テスト目的でSTONITHが無効化されている場合は、再度このパラメータがtrueに設定されていることを確認してください。
- ② ディスクレスSBDの場合、このパラメータはゼロであってはなりません。これは、どれくらいの時間が経ったらフェンシングターゲットがすでにセルフフェンスを行ったとみなされるのかを定義します。したがって、その値は/etc/sysconfig/sbdのSBD_WATCHDOG_TIMEOUTの値以上である必要があります。SUSE Linux Enterprise High Availability Extension 15から、stonith-watchdog-timeoutを負の値に設定した場合、Pacemakerは自動的にこのタイムアウトを計算し、SBD_WATCHDOG_TIMEOUTをインストールします。

6. **show**で変更内容をレビューします。

7. **commit**で変更を送信し、**exit**でcrmライブ設定を終了します。

11.9 SBDとフェンシングのテスト

SBDがノードフェンシング目的で期待どおりに機能するかどうかをテストするには、次のいずれかまたはすべての方法を使用します。

ノードのフェンシングを手動でトリガする

ノード `NODENAME` のフェンシングアクションをトリガするには:

```
root # crm node fence NODENAME
```

当該ノードがフェンシングされているかどうか、および `stonith-watchdog-timeout` をインストールします。

SBD障害のシミュレーション

1. SBD inquisitorのプロセスIDを特定します。

```
root # systemctl status sbd
● sbd.service - Shared-storage based fencing daemon

   Loaded: loaded (/usr/lib/systemd/system/sbd.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2018-04-17 15:24:51 CEST; 6 days ago
     Docs: man:sbd(8)
  Process: 1844 ExecStart=/usr/sbin/sbd $SBD_OPTS -p /var/run/sbd.pid watch (code=exited, status=0/SUCCESS)
 Main PID: 1859 (sbd)
    Tasks: 4 (limit: 4915)
   CGroup: /system.slice/sbd.service
          └─「1859 sbd: inquisitor」

[...]
```

2. SBD inquisitorプロセスを終了することにより、SBD障害をシミュレーションします。この例では、SBD inquisitorのプロセスIDは1859です。

```
root # kill -9 1859
```

当該ノードは積極的にセルフフェンスを行います。他のノードは、当該ノードの喪失を認識し、`stonith-watchdog-timeout`をインストールします。

監視動作の障害によるフェンシングのトリガ

通常の設定では、リソース「停止動作」の障害によって、フェンシングがトリガされます。フェンシングを手動でトリガするために、リソース停止動作の障害を発生させることができます。あるいは、以下に説明するように、リソース「監視動作」の設定を一時的に変更して、監視障害を発生させることができます。

1. リソース監視操作に`on-fail=fence`プロパティを設定します。

```
op monitor interval=10 on-fail=fence
```

2. 監視動作の障害を発生させます(たとえば、リソースがサービスに関連する場合は、それぞれのデーモンを終了させます)。この障害により、フェンシングアクションがトリガされます。

11.10 ストレージ保護のための追加メカニズム

STONITHによるノードフェンシング以外に、リソースレベルでストレージ保護を実現する方法があります。たとえば、SCSI-3とSCSI-4は永続予約を使用しますが、`sfex`はロック機構を提供します。両方の方法について以下のサブセクションで説明します。

11.10.1 `sg_persist`リソースの設定

SCSI仕様3および4では、「永続予約」が定義されています。これらはSCSIプロトコル機能であり、I/Oフェンシングとフェールオーバーに使用できます。この機能は、`sg_persist` Linux コマンドで実装されます。



注記: SCSIディスクの互換性

`sg_persist`のバックギングディスクは、SCSIディスクとの互換性が必要です。`sg_persist`は、SCSIディスクやiSCSI LUNなどのデバイスでのみ機能します。IDE、SATA、またはSCSIプロトコルをサポートしないブロックデバイスでは、使用「しない」でください。

続行する前に、お使いのディスクが永続予約をサポートしているかどうかを確認してください。次のコマンドを使用します(`DISK`をデバイス名で置き換えてください)。

```
root # sg_persist -n --in --read-reservation -d /dev/DISK
```

結果に、ディスクが永続予約をサポートしているかが示されます。

- サポートされているディスク:

```
PR generation=0x0, there is NO reservation held
```

- サポートされていないディスク:

```
PR in (Read reservation): command not supported
Illegal request, Invalid opcode
```

上記のようなエラーメッセージが表示された場合は、古いディスクをSCSIと互換性のあるディスクに交換してください。それ以外の場合は、以下の手順に従います。

1. プリミティブリソース `sg_persist` を作成するには、`root` として次のコマンドを実行します。

```
root # crm configure
crm(live)configure# primitive sg sg_persist \
    params devs="/dev/sdc" reservation_type=3 \
    op monitor interval=60 timeout=60
```

2. `sg_persist` プリミティブをマスタ-スレーブグループに追加します。

```
crm(live)configure# ms ms-sg sg \
    meta master-max=1 notify=true
```

3. いくつかのテストをします。リソースがマスタ/スレーブ構成のステータスにある場合、マスタインスタンスが実行されているクラスタノードでは `/dev/sdc1` をマウントして書き込みを行えますが、スレーブインスタンスが実行されているクラスタノードでは書き込みが禁止されます。

4. Ext4 のファイルシステムプリミティブを追加します。

```
crm(live)configure# primitive ext4 ocf:heartbeat:Filesystem \
    params device="/dev/sdc1" directory="/mnt/ext4" fstype=ext4
```

5. `sg_persist` マスタとファイルシステムリソースの間に、次の順序関係とコロケーションを追加します。

```
crm(live)configure# order o-ms-sg-before-ext4 Mandatory: ms-sg:promote ext4:start
crm(live)configure# colocation col-ext4-with-sg-persist inf: ext4 ms-sg:Master
```

6. `show changed` コマンドで、すべての変更内容を確認します。

7. 変更をコミットします。.

詳細については、`sg_persist` のマニュアルページを参照してください。

11.10.2 sfexを使用した排他的なストレージアクティブ化の保証

このセクションでは、共有ストレージへのアクセスを1つのノードに排他的にロックする低レベルの追加メカニズムであるsfexを紹介します。ただし、sfexは、STONITHと置き換えることはできないので注意してください。sfexには共有ストレージが必要なので、上記で説明したSBDノードフェンシングメカニズムは、ストレージの別のパーティションでを使用することをお勧めします。

設計上、sfexは、同時実行が必要なワークロード(OCFS2など)では使用できません。これは、従来のフェールオーバースタイルのワークロードに対する保護の層として機能します。これは、実際にはSCSI-2予約と似ていますが、もっと一般的です。

11.10.2.1 概要

共有ストレージ環境では、ストレージの小さなパーティションが1つ以上のロックの保存用に確保されます。

ノードは、保護されたリソースを取得する前に、まず、保護ロックを取得する必要があります。順序はPacemakerによって強制されます。sfexコンポーネントは、Pacemakerがスプリットブレイン条件に制約されても、ロックが2回以上付与されないことを保証します。

ノードのダウンが永続的にロックをブロックせず、他のノードが続行できるように、これらのロックも定期的に更新される必要があります。

11.10.2.2 セットアップ

次に、sfexで使用する共有パーティションの作成方法と、CIBでsfexロック用にリソースを設定する方法を説明します。1つのsfexパーティションは任意の数のロックを保持でき、ロックごとに1KBのストレージスペースを割り当てる必要があります。デフォルトでは、**sfex_init**はパーティション上にロックを1つ作成します。

！ 重要: 要件

- sfex用の共有パーティションは、保護するデータと同じ論理ユニットにある必要があります。
- 共有されたsfexパーティションは、ホストベースのRAIDやDRBDを使用してはなりません。
- LVM2論理ボリュームを使用することは可能です。

手順 11.9: SFEXパーティションの作成

1. sfexで使用する共有パーティションを作成します。このパーティションの名前を書き留め、以降の手順の/dev/sfexをこの名前で置き換えます。
2. 次のコマンドでsfexメタデータを作成します。

```
root # sfex_init -n 1 /dev/sfex
```

3. メタデータが正しく作成されたかどうか検証します。

```
root # sfex_stat -i 1 /dev/sfex ; echo $?
```

現在、ロックがかかっていないので、このコマンドは、2を返すはずです。

手順 11.10: SFEXロック用リソースの設定

1. sfexロックは、CIB内のリソースを介して表現され、次のように設定されます。

```
crm(live)configure# primitive sfex_1 ocf:heartbeat:sfex \  
# params device="/dev/sfex" index="1" collision_timeout="1" \  
lock_timeout="70" monitor_interval="10" \  
# op monitor interval="10s" timeout="30s" on-fail="fence"
```

2. sfexロックによってリソースを保護するには、保護対象のリソースとsfexリソース間の必須の順序付けと配置の制約を作成します。保護対象のリソースがfilesystem1というIDを持つ場合は、次のようになります。

```
crm(live)configure# order order-sfex-1 Mandatory: sfex_1 filesystem1  
crm(live)configure# colocation col-sfex-1 inf: filesystem1 sfex_1
```

3. グループ構文を使用する場合は、sfexリソースを最初のリソースとしてグループに追加します。

```
crm(live)configure# group LAMP sfex_1 filesystem1 apache ipaddr
```

11.11 詳細の参照先

- [man sbd](#)
- http://www.linux-ha.org/wiki/SBD_Fencing 

12 QDeviceとQNetd

QDeviceとQNetdはクォーラムの決定に参加します。アービトレータcorosync-qnetdからの支援により、corosync-qdeviceは設定可能な投票数を提供するため、クラスタは標準のクォーラムルールで許可されているよりも多くのノード障害に耐えることができます。2ノードクラスタにはcorosync-qnetdとcorosync-qdeviceを展開することを強くお勧めしますが、ノード数が偶数のクラスタには一般的にQNetdとQDeviceを使用することもお勧めします。

12.1 概念の概要

クラスタノード間のクォーラを計算するのと比較して、QDevice-and-QNetdアプローチには次の利点があります。

- ノード障害が発生した場合の持続可能性が向上します。
- 投票に影響する独自のヒューリスティックスクリプトを作成できます。これは、SAPアプリケーションなどの複雑なセットアップに特に役立ちます。
- 複数のクラスタに投票を提供するようにQNetdサーバを設定できます。
- 2ノードクラスタでディスクレスSBDを使用できます。
- スプリットブレイン状況下で偶数のノードを持つクラスタ、特に2ノードクラスタのクォーラムの決定に役立ちます。

QDevice/QNetdを使用したセットアップは、次のコンポーネントとメカニズムで構成されます。

QDEVICE/QNETDコンポーネントとメカニズム

QNetd (corosync-qnetd)

クラスタの一部ではないsystemdサービス(デーモン、「QNetdサーバ」)。systemdサービスは、corosync-qdeviceデーモンに投票を提供します。

セキュリティを向上させるため、corosync-qnetdは、クライアント証明書の確認にTLSと連携することができます。

QDevice (corosync-qdevice)

Corosyncとともに実行されている各クラスタノード上のsystemdサービス(デーモン)。これはcorosync-qnetdのクライアントです。その主な用途は、クラスタが標準のクォーラムルールが許可するよりも多くのノード障害に耐えることができるようにすることです。

QDeviceはさまざまなアービトラータと連携するように設計されています。ただし、現在では、QNetdのみがサポートされています。

アルゴリズム

QDeviceは投票の割り当て方法の動作を決定する、さまざまなアルゴリズムをサポートしています。現在は、以下が存在します。

- FFSplit (「fifty-fifty split」) がデフォルトです。これは、ノード数が偶数のクラスタに使用されます。クラスタが2つの同じパーティションに分割される場合、このアルゴリズムによりヒューリスティックスの確認と他の要因に基づいて、パーティションのいずれかに1票が提供されます。クラスタが2つの同様のパーティションに分割される場合、このアルゴリズムによりヒューリスティックスの確認と他の要因に基づいて、パーティションのいずれかに1票が提供されます。
- LMS (「last man standing」) では、QNetdサーバを確認できる残っている唯一のノードに1票が与えられます。QNetdサーバを確認できる残っている唯一のノードである場合に、1票が返されます。したがって、このアルゴリズムは、アクティブな1つのノードのみを持つクラスタがクォーラムに達した状態を維持する必要がある場合に役立ちます。

ヒューリスティックス

QDeviceは一連のコマンド(「heuristics」)をサポートしています。コマンドは、クラスタサービスの起動、クラスタメンバーシップの変更、corosync-qnetdへの接続の成功時にローカルで実行されるか、オプションで定期的に行われます。ヒューリスティックスは `quorum.device.heuristics` キー(corosync.confファイル内)または `--qdevice-heuristics-mode` オプションを使用して設定できます。両方が `off` (デフォルト)、`sync`、および `on` の値を認識しています。`sync` と `on` の違いは、先に示したコマンドを定期的に補足的に行うことができるということです。

すべてのコマンドが正常に実行された場合にのみ、ヒューリスティックスは合格したとみなされ、そうでない場合は、失敗したとみなされます。ヒューリスティックスの結果は、corosync-qnetdに送信され、そこでクォーラムに達するパーティションを決定するための計算に使用されます。

Tiebreaker

これは同じヒューリスティックスの結果の場合でも、クラスタパーティションが完全に等しい場合のフォールバックとして使用されます。最小、最大、または特定のノードIDに設定できます。

12.2 要件と前提条件

QDeviceとQNetdを設定する前に、次のように環境を準備しておく必要があります。

- クラスタノードのほかに、QNetdサーバになる別のマシンが必要です。12.3項「QNetdサーバのセットアップ」を参照してください。
- Corosyncが使用するものとは異なる物理ネットワーク。QDeviceがQNetdサーバに到達することをお勧めします。理想としては、QNetdサーバはメインクラスタとは別のラックに配置するか、少なくとも別のPSUに配置し、Corosyncリングと同じネットワークセグメントには配置しないでください。

12.3 QNetdサーバのセットアップ

QNetdサーバはクラスタスタックの一部ではなく、クラスタの実際のメンバーでもありません。そのため、このサーバにリソースを移動することはできません。

QNetdサーバはほとんど「ステートフリー」です。通常、設定ファイル`/etc/sysconfig/corosync-qnetd`内を変更する必要はありません。デフォルトでは、`corosync-qnetd`サービスは、グループ`coroqnetd`のユーザ`coroqnetd`としてデーモンを実行します。これにより、`root`としてデーモンを実行する必要がなくなります。

QNetdサーバを作成するには、次の手順に従います。

1. QNetdサーバになるマシン上に、SUSE Linux Enterprise Server 15 SP3をインストールします。
2. QNetdサーバにログインして、次のパッケージをインストールします。

```
root # zypper install corosync-qnetd
```

`corosync-qnetd`サービスを手動で開始する必要はありません。ブートストラップスクリプトは、`qdevice`ステージ中の起動プロセスを処理します。

QNetdサーバはQDeviceクライアント`corosync-qdevice`からの接続を受け入れる準備ができています。さらに設定する必要はありません。

12.4 QDeviceクライアントをQNetdサーバに接続する

QNetdサーバを設定した後で、クライアントを設定して実行することができます。クラスタのインストール中にQNetdサーバにクライアントを接続するか、後で追加することができます。次の手順では、後者の方法を使用します。2つのクラスタノード(aliceとbob)およびQNetdサーバ(charlie)を持つクラスタを想定しています。

1. すべてのノードで、パッケージ `corosync-qdevice` がインストールされていることを確認します。`zypper` をインストールします。

```
root # zypper install corosync-qdevice
```

2. aliceで、クラスタを初期化します。

```
root # crm cluster init -y
```

3. bobで、クラスタに参加します。

```
root # crm cluster join -c alice -y
```

4. aliceとbobで、qdeviceステージをブートストラップします。ほとんどの場合、デフォルト設定で問題ありません。少なくとも `--qnetd-hostname` とQNetdサーバのホスト名またはIP アドレス(この場合はCharlie)を指定します。

```
root # crm cluster init qdevice --qnetd-hostname=charlie
```

デフォルトの設定を変更する場合は、コマンド `crm cluster init qdevice --help` を使用してすべての可能なオプションのリストを取得します。QDeviceに関連するすべてのオプションは、`--qdevice-NAME` で開始されます。

デフォルト設定を使用している場合、先に示したコマンドはTLSが有効で、FFSplitアルゴリズムを使用するQDeviceを作成します。

12.5 ヒューリスティックスを使用したQDeviceの設定

投票の決定方法をさらに制御する必要がある場合は、ヒューリスティックスを使用します。ヒューリスティックスは、並行して実行される一連のコマンドです。

この目的のため、コマンド `crm cluster init qdevice` はオプション `--qdevice-heuristics` を提供します。絶対パスを使用して1つ以上のコマンド(セミコロンで区切る)を渡すことができます。

たとえば、ヒューリスティックチェック用の独自のコマンドが `/usr/sbin/my-script.sh` にある場合は、次のようにクラスタノードのいずれかで実行できます。

```
root # crm cluster init qdevice --qdevice-hostname=charlie \  
      --qdevice-heuristics=/usr/sbin/my-script.sh \  
      --qdevice-heuristics-mode=on
```

コマンドは、Shell、Python、またはRubyなどの任意の言語で記述することができます。成功した場合は、0 (ゼロ) を返し、失敗した場合はエラーコードを返します。

一連のコマンドを渡すこともできます。すべてのコマンドが正常に終了した場合のみ(戻りコードが0)、ヒューリスティックスが渡されます。

`--qdevice-heuristics-mode=on` オプションを使用すると、ヒューリスティックスコマンドを定期的に実行できます。

12.6 クォーラムステータスの確認と表示

例12.1「QDeviceのステータス」に示すように、クラスタノードのいずれかでクォーラムステータスを問い合わせることができます。QDeviceノードのステータスが表示されます。

例 12.1: QDEVICEのステータス

```
root # corosync-quorumtool ①  
Quorum information  
-----  
Date: ...  
Quorum provider: corosync_votequorum  
Nodes: 2 ②  
Node ID: 3232235777 ③  
Ring ID: 3232235777/8  
Quorate: Yes ④  
  
Votequorum information  
-----  
Expected votes: 3  
Highest expected: 3  
Total votes: 3  
Quorum: 2  
Flags: Quorate Qdevice  
  
Membership information
```

Nodeid	Votes	Qdevice Name
3232235777	1	A,V,NMW 192.168.1.1 (local) ⑤
3232235778	1	A,V,NMW 192.168.1.2 ⑤
0	1	Qdevice

- ① 同一の結果の代替として、`crm corosync status quorum`コマンドを使用することもできます。
- ② 予想されるノード数。この例では、2ノードクラスタです。
- ③ `corosync.conf`で、ノードIDが明示的に指定されていないため、このIDはIPアドレスの32ビットの整数表現です。この例では、値3232235777はIPアドレス192.168.1.1を表しています。
- ④ クォーラムステータス。この場合、クラスタにはクォーラムがあります。
- ⑤ 各クラスタノードのステータスは以下を意味します。

A (アクティブ)またはNA (非アクティブ)

QDeviceとCorosync間の接続ステータスを示します。QDeviceとCorosync間にハートビートがある場合は、アクティブ(A)として表示されます。

V (投票)またはNV (投票しない)

クォーラムデバイスがノードに投票(文字V)したかどうかを示します。文字Vは、両方のノードが互いに通信できることを意味します。スプリットブレイン状態では、一方のノードがVに設定され、他方のノードはNVに設定されます。

MW (マスターwins)またはNMW(master winsでない)

クォーラムデバイス `master_wins` フラグが設定されているかどうかを示します。デフォルトでは、フラグが設定されていないため、NMW (master winsでない)が表示されます。詳細については、マニュアルページ `votequorum_qdevice_master_wins(3)`を参照してください。

NR (未登録)

クラスタがクォーラムデバイスを使用していないことを示します。

QNetdサーバのステータスを問い合わせると、例12.2「QNetdサーバのステータス」に示すのと同様の出力が得られます。

例 12.2: QNETDサーバのステータス

```
root # corosync-qnetd-tool ①
Cluster "hacluster": ②
  Algorithm:      Fifty-Fifty split ③
  Tie-breaker:    Node with lowest node ID
```

```

Node ID 3232235777: ④
  Client address:      ::ffff:192.168.1.1:54732
  HB interval:        8000ms
  Configured node list: 3232235777, 3232235778
  Ring ID:            aa10ab0.8
  Membership node list: 3232235777, 3232235778
  Heuristics:         Undefined (membership: Undefined, regular: Undefined)
  TLS active:         Yes (client certificate verified)
  Vote:               ACK (ACK)
Node ID 3232235778:
  Client address:      ::ffff:192.168.1.2:43016
  HB interval:        8000ms
  Configured node list: 3232235777, 3232235778
  Ring ID:            aa10ab0.8
  Membership node list: 3232235777, 3232235778
  Heuristics:         Undefined (membership: Undefined, regular: Undefined)
  TLS active:         Yes (client certificate verified)
  Vote:               No change (ACK)

```

- ① 同一の結果の代替として、`crm corosync status qnetd`コマンドを使用することもできます。
- ② `totem.cluster_name`セクションの設定ファイル/`etc/corosync/corosync.conf`で設定されているクラスタの名前。
- ③ 現在使用されているアルゴリズム。この例では、`FFSplit`です。
- ④ これは、IPアドレスが`192.168.1.1`のノードのエントリです。TLSがアクティブです。

12.7 詳細の参照先

QDeviceとQNetdに関する追加情報については、`corosync-qdevice(8)`と`corosync-qnetd(8)`のマニュアルページを参照してください。

13 アクセス制御リスト

crmシェル(crmsh)またはHawk2などのクラスタ管理ツールは、rootユーザまたはhaclientグループ内のユーザが使用できます。デフォルトで、これらのユーザは完全な読み込み/書き込みのアクセス権を持ちます。アクセスを制限するか、または詳細なアクセス権を割り当てるには、「アクセス制御リスト」(ACL)を使用できます。

アクセス制御リストは、順序付けされたアクセスルールセットで構成されています。各ルールにより、クラスタ設定の一部への読み込みまたは書き込みアクセスの許可、またはアクセスの拒否が行われます。ルールは通常、組み合わせて特定の役割を生成し、ユーザを自分のタスクに一致する役割に割り当てることができます。



注記: CIB構文検証バージョンとACLとの違い

このACLマニュアルは、pacemaker-2.0以上のCIB構文バージョンでCIBを検証する場合にのみ適用します。この検証方法およびCIBバージョンのアップグレード方法の詳細については、[注記: CIB構文バージョンのアップグレード](#)を参照してください。

13.1 要件と前提条件

クラスタでACLの使用を開始する前に、次の条件が満たされていることを確認します。

- NIS、Active Directoryを使用するか、またはすべてのノードに同じユーザを手動で追加して、クラスタ内のすべてのノード上に同じユーザがいることを確認します。
- ACLでアクセス権を変更したいすべてのユーザがhaclientグループに属している必要があります。
- すべてのユーザが絶対パス/usr/sbin/crmでcrmshを実行する必要があります。
- 権限のないユーザがcrmshを実行する場合は、/usr/sbinを使用して、PATH変数を展開する必要があります。

！ 重要: デフォルトのアクセス権

- ACLはオプションの機能です。デフォルトでは、ACLの使用は無効になっています。
- ACL機能が無効化された場合、`root`および`haclient`グループに属するすべてのユーザは、クラスタ設定への完全な読み込み/書き込みアクセス権を持ちます。
- ACLが有効化され、設定される場合でも、`root`およびデフォルトのCRM所有者`haclient`「は両方とも、「常に」」クラスタ設定への完全なアクセス権を持ちます。

13.2 概念の概要

アクセス制御リストは、順序付けされたアクセスルールセットで構成されています。各ルールにより、クラスタ設定の一部への読み込みまたは書き込みアクセスの許可、またはアクセスの拒否が行われます。ルールは通常、組み合わせて特定の役割を生成し、ユーザを自分のタスクに一致する役割に割り当てることができます。ACLの役割はCIBへのアクセス権を表すルールのセットです。ルールは次の要素で構成されています。

- `read`、`write`、または`deny`のようなアクセス権。
- ルールを適用する場所の指定。種類、ID参照、またはXPath式を使用して指定できます。XPathはXMLドキュメントでノードを選択するための言語です。 <http://en.wikipedia.org/wiki/XPath> を参照してください。

通常、ACLを役割にバンドルし、システムユーザ(ACLターゲット)に特定の役割を割り当てると便利です。ACLルールを作成するためには、次の方法があります。

- 13.7項「XPath式によるACLルールの設定」。ACLルールを作成するためには、その記述言語であるXMLの構造を理解している必要があります。
- 13.8項「短縮によるACLルールの設定」。簡略構文を作成し、ACLルールが一致するオブジェクトに適用します。

13.3 クラスタでのACLの使用の有効化

ACLの設定を開始する前に、ACLの使用を「有効にする」必要があります。有効にするには、`crmsh`で次のコマンドを使用します。

```
root # crm configure property enable-acl=true
```

または、[手順13.1「Hawk2でのACLの使用の有効化」](#)で説明するように、Hawk2を使用します。

手順 13.1: HAWK2でのACLの使用の有効化

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーで、クラスタ設定を選択して、グローバルクラスタオプションとそれらの現在の値を表示します。
3. クラスタ設定の下にある空のドロップダウンボックスをクリックし、enable-aclを選択してパラメータを追加します。デフォルト値Noで追加されます。
4. 値をYesに設定して変更を適用します。

13.4 読み込み専用monitor役割の作成

次のサブセクションでは、Hawk2またはcrmシェルのいずれかでmonitor役割を定義することにより、読み込み専用アクセスを設定する方法について説明します。

13.4.1 Hawk2による読み込み専用monitor役割の作成

次の手順は、monitor役割を定義し、それをユーザに割り当てることで、クラスタ設定への読み込み専用アクセスを設定する方法を示しています。または、[手順13.4「monitor役割を追加して、crmshを持つユーザに割り当てる」](#)で説明されているように、crmshを使用してこの操作を実行することもできます。

手順 13.2: HAWK2によるMONITOR役割の追加

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーで、役割を選択します。
3. 作成をクリックします。
4. 固有な「役割ID」として、monitorなどを入力します。

5. アクセス権利として、Readを選択します。
6. Xpathとして、XPath式/cibを入力します。

SUSE Hawk クラスタ詳細の表示

役割の作成

役割ID: monitor

ルール: [+ - ↑ ↓]

権利: read

XPath: /cib

オブジェクトタイプ:

参照:

[作成] [戻る]

ACLの役割

ACLの役割は、CIBへのアクセス権を表すルールセットです。
各ルールは次の要素で構成されます。

- アクセス権 (読み込み、書き込み、または拒否)
- ルールを適用する場所の指定 (XPath式、タイプ、またはID参照)

役割の作成

役割ID: 固有のIDを定義します。

権利: アクセス権 (読み込み / 書き込み / 拒否) を選択します。

Xpath: アクセス権を適用するCIB要素のXPath式を入力します
(例: 場所の制約に適用する場合は `//constraints/rsc_location`)。

タイプ: アクセス権を適用するCIB XML要素の名前を入力します
(例: 場所の制約に適用する場合は `rsc_location`)。

参照: アクセス権を適用するCIB XML要素のIDを入力します
(例: ID `rsc1` を持つすべてのXML要素に適用する場合は、`rsc1`)。

7. 作成をクリックします。
この操作は、monitorの名前を持つ新しい役割を作成して、readの権利を設定し、XPath式/cibを使用してCIB内のすべての要素に適用します。
8. 必要に応じてプラスアイコンをクリックしてルールを追加し、個別のパラメータを指定します。
9. 上矢印や下矢印のボタンを使用して、個別のルールをソートできます。

手順 13.3: HAWK2によるターゲットへの役割割り当て

手順 13.2で作成した役割をシステムユーザ(ターゲット)に割り当てるには、次の手順に従います。

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーで、ターゲットを選択します。
3. システムユーザ(ACLターゲット)を作成するには、作成をクリックして、固有のターゲットIDを入力します(例: tux)。このユーザがhaclientグループに属することを確認します。
4. ターゲットに役割を割り当てるには、1つ以上の役割を選択します。

例では、手順 13.2 で作成した monitor 役割を選択します。



5. 選択内容を確認します。

リソースや制約に対するアクセス権を設定するには、13.8項「短縮によるACLルールの設定」で説明したように、短縮構文も使用できます。

13.4.2 crmshによる読み込み専用monitor役割の作成

次の手順は、monitor 役割を定義し、それをユーザに割り当てることで、クラスタ設定への読み込み専用アクセスを設定する方法を示しています。

手順 13.4: MONITOR役割を追加して、CRMSHを持つユーザに割り当てる

1. root としてログインします。
2. crmshの対話モードを開始します。

```
root # crm configure
crm(live)configure#
```

3. ACLの役割を次のとおり定義します。

- a. role コマンドを使用して、新しい役割を定義します。

```
crm(live)configure# role monitor read xpath="/cib"
```

前のコマンドは、`monitor`の名前を持つ新しい役割を作成して、`read`の権利を設定し、XPath式/`cib`を使用してCIB内のすべての要素に適用します。必要な場合は、アクセス権およびXPath引数をさらに追加できます。

b. 必要に応じてさらに役割を追加します。

4. 役割を1つ以上のACLターゲットに割り当てます。このACLターゲットは、該当のシステムユーザです。これらのシステムユーザが`haclient`グループに属していることを確認します。

```
crm(live)configure# acl_target tux monitor
```

5. 変更を確認します。

```
crm(live)configure# show
```

6. 変更をコミットします。:

```
crm(live)configure# commit
```

リソースや制約に対するアクセス権を設定するには、[13.8項「短縮によるACLルールの設定」](#)で説明したように、短縮構文も使用できます。

13.5 ユーザの削除

次のサブセクションでは、Hawk2またはcrmshのいずれかで、ACLから既存のユーザを削除する方法について説明します。

13.5.1 Hawk2によるユーザの削除

ACLからユーザを削除するには、次の手順に従います。

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーで、ターゲットを選択します。
3. システムユーザ(ACLターゲット)を削除するには、操作列の下にあるごみ箱アイコンをクリックします。

4. ダイアログボックスを確認します。

13.5.2 crmshによるユーザの削除

ACLからユーザを削除するには、プレースホルダ`USER`をユーザの名前で置き換えます置換します。

```
root # crm configure delete USERNAME
```

別の方法として、`edit`サブコマンドを使用できます。

```
root # crm configure edit USERNAME
```

13.6 既存の役割の削除

次のサブセクションでは、Hawk2またはcrmshのいずれかで、既存の役割を削除する方法について説明します。



注記: 参照されているユーザを含む役割の削除

この役割にユーザを含めないでください。役割内にユーザへの参照がまだある場合、役割を削除できません。まず、ユーザへの参照を削除してから、役割を削除してください。

13.6.1 Hawk2による既存の役割の削除

役割を削除するには、次の手順に従います。

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーで、役割を選択します。
3. 役割を削除するには、操作列の下にあるごみ箱アイコンをクリックします。
4. ダイアログボックスを確認します。エラーメッセージが表示される場合は、役割が「空」であり、ユーザを参照していないことを確認してください。

13.6.2 crmshによる既存の役割の削除

既存の役割を削除するには、プレースホルダ`ROLE`を役割の名前で置き換えます。

```
root # crm configure delete ROLE
```

13.7 XPath式によるACLルールの設定

XPathによってACLルールを管理するには、その記述言語であるXMLの構造を理解する必要があります。XMLでクラスタ設定を表示する次のコマンドで構造を取得します(例 13.1を参照)。

```
root # crm configure show xml
```

例 13.1: XML内のクラスタ設定の例

```
<cib>
  <!-- ... -->
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair name="stonith-enabled" value="true" id="cib-bootstrap-options-stonith-enabled"/>
        [...]
      </cluster_property_set>
    </crm_config>
    <nodes>
      <node id="175704363" uname="alice"/>
      <node id="175704619" uname="bob"/>
    </nodes>
    <resources> [...] </resources>
    <constraints/>
    <rsc_defaults> [...] </rsc_defaults>
    <op_defaults> [...] </op_defaults>
  </configuration>
</cib>
```

XPath言語を使用して、このXMLドキュメント内のノードを見つけることができます。たとえば、ルートノード(`cib`)を選択するには、XPath式`/cib`を使用します。グローバルクラスタ設定を見つけるには、XPath式`/cib/configuration/crm_config`を使用します。

一例として、表13.1「オペレータ役割 - アクセスタイプおよびXPath式」は、「オペレータ」の役割を作成するためのパラメータ(アクセスタイプおよびXPath式)を示しています。この役割を持つユーザは、2番目の列で説明されるタスクのみ実行することができ、リソースを再構成することはできません(たとえば、パラメータや操作の変更など)。また、コロケーションや順序の制約の設定を変更することもできません。

表 13.1: オペレータ役割 - アクセスタイプおよびXPath式

タイプ	XPath/説明
書き込み	<pre>//crm_config//nvpair[@name='maintenance-mode']</pre> <p>クラスタ保守モードをオンまたはオフにします。</p>
書き込み	<pre>//op_defaults//nvpair[@name='record-pending']</pre> <p>保留中の操作を記録するかを選択します。</p>
書き込み	<pre>//nodes/node//nvpair[@name='standby']</pre> <p>ノードをオンラインまたはスタンバイモードで設定します。</p>
書き込み	<pre>//resources//nvpair[@name='target-role']</pre> <p>リソースを開始、停止、昇格または降格します。</p>
書き込み	<pre>//resources//nvpair[@name='maintenance']</pre> <p>リソースを保守モードにするかどうかを選択します。</p>
書き込み	<pre>//constraints/rsc_location</pre> <p>リソースをノードから別のノードにマイグレート/移動します。</p>
読み込み	<pre>/cib</pre> <p>クラスタのステータスを表示します。</p>

13.8 短縮によるACLルールの設定

XML構造を扱いたくないユーザ向けには、より簡単な方法があります。

たとえば、次のXPathを検討します。

```
//*[@id="rsc1"]
```

このXPathは、IDがrsc1であるXMLノードをすべて探し出します。

短縮構文はこのように書かれます。

```
ref:"rsc1"
```

これは制約にも使用できます。これが冗長なXPathです。

```
//constraints/rsc_location
```

短縮構文はこのように書かれます。

```
type:"rsc_location"
```

短縮構文はcrmshおよびHawk2で使用できます。CIBデーモンは一致するオブジェクトにACLルールを適用する方法を認識しています。

14 ネットワークデバイスボンディング

多くのシステムで、通常のEthernetデバイスの標準のデータセキュリティ/可用性の要件を超えるネットワーク接続の実装が望ましいことがあります。その場合、数台のEthernetデバイスを集めて1つのボンディングデバイスを設定できます。

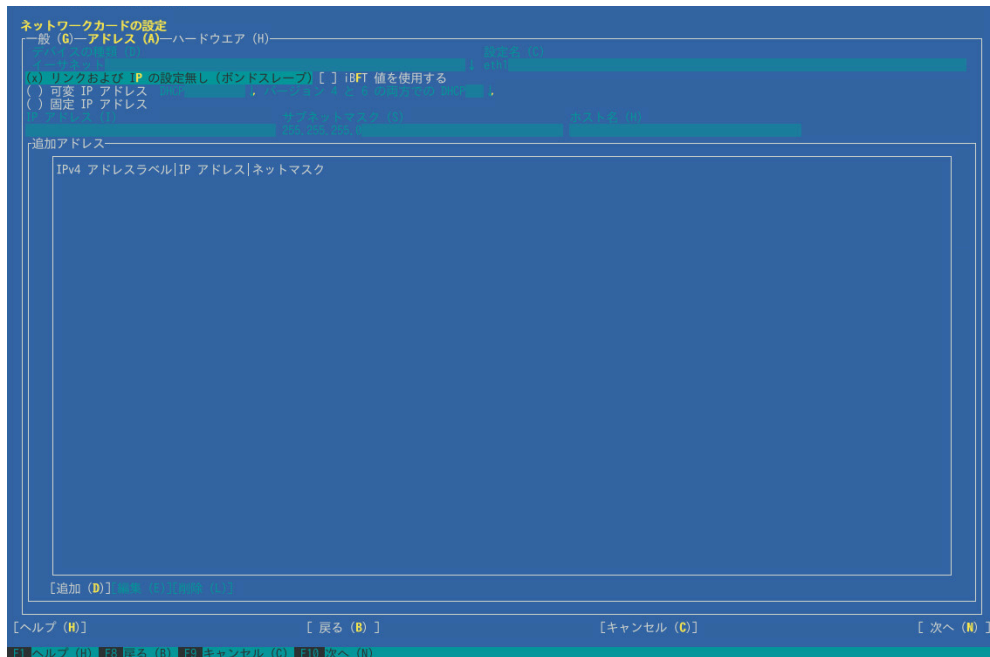
ボンディングデバイスの設定には、ボンディングモジュールオプションを使用します。ボンディングデバイスの振る舞いは、ボンディングデバイスのモードによって決定されます。デフォルトの動作は、`mode=active-backup`であり、アクティブなスレーブに障害が発生すると、別のスレーブデバイスがアクティブになります。

Corosyncの使用時は、クラスタソフトウェアでボンディングデバイスが管理されることはありません。したがって、ボンディングデバイスにアクセスする可能性のあるクラスタノードごとに、ボンディングデバイスを設定する必要があります。

14.1 YaSTによるボンディングデバイスの設定

ボンディングデバイスを設定するには、1つのボンディングデバイスに集めることができる数台のEthernetデバイスが必要です。以下に手順を示します。

1. `root`としてYaSTを開始し、システム > ネットワーク設定の順に選択します。
2. ネットワーク設定で、概要タブに切り替えて、使用可能なデバイスを表示します。
3. 1つのボンディングデバイスに集めるEthernetデバイスにIPアドレスが割り当てられているかどうかチェックします。割り当てられている場合は、それを変更します。
 - a. 選択するデバイスを選択して、編集をクリックします。
 - b. 開いているネットワークカードのセットアップダイアログのアドレスタブで、リンクとIPなしのセットアップ(ボンディングスレーブ)オプションを選択します。



c. 次へをクリックして、ネットワーク設定ダイアログのOverviewタブに戻ります。

4. 新しいボンディングデバイスを追加するには:

- a. 追加をクリックして、デバイスの型をボンドに変更します。次へで続行します。
- b. IPアドレスをボンディングデバイスに割り当てる方法を選択します。3つの方法から選択できます。
 - リンクとIPなしのセットアップ(ボンディングスレーブ)
 - 可変IPアドレス(DHCPまたはZeroconf)
 - 固定IPアドレス

ご使用の環境に適合する方法を使用します。Corosyncで仮想IPアドレスを管理する場合は、静的割り当てIPアドレスを選択し、インタフェースにIPアドレスを割り当てます。

- c. ボンドスレーブタブに切り替えます。
- d. **ステップ 3.b**でボンディングスレーブとして設定したEthernetデバイスが表示されます。ボンドに含めるEthernetデバイスを選択するには、ボンドスレーブと順序の下にある、各デバイスの前のチェックボックスを有効にします。



- e. ボンドドライバオプションを編集します。次のモードを使用できます。

balance-rr

パケットが正しい順序で転送されなくなる代わりに、負荷分散と耐障害性が提供されます。これは、TCPの再構築時などに遅延の原因になる場合があります。

active-backup

耐障害性を提供します。

balance-xor

負荷分散と耐障害性を提供します。

ブロードキャスト

耐障害性を提供します。

802.3ad

接続されるスイッチでサポートされる場合は、ダイナミックリンク集合を提供します。

balance-tlb

発信トラフィックの負荷分散を提供します。

balance-alb

使用中にハードウェアアドレスの変更が可能なネットワークデバイスを使用する場合は、着信トラフィックと発信トラフィックの負荷分散を提供します。

- f. ボンドドライバオプションには、パラメータ`miimon=100`を必ず追加します。このパラメータがなければ、リンクが定期的にチェックされないため、ボンディングドライバは、障害リンクで引き続きパケットを失う可能性があります。
5. 次へをクリックして、OKでYaSTを終了し、ボンディングデバイスの設定を完了します。YaSTが`/etc/sysconfig/network/ifcfg-bondDEVICENUMBER`に設定を書き込みます。

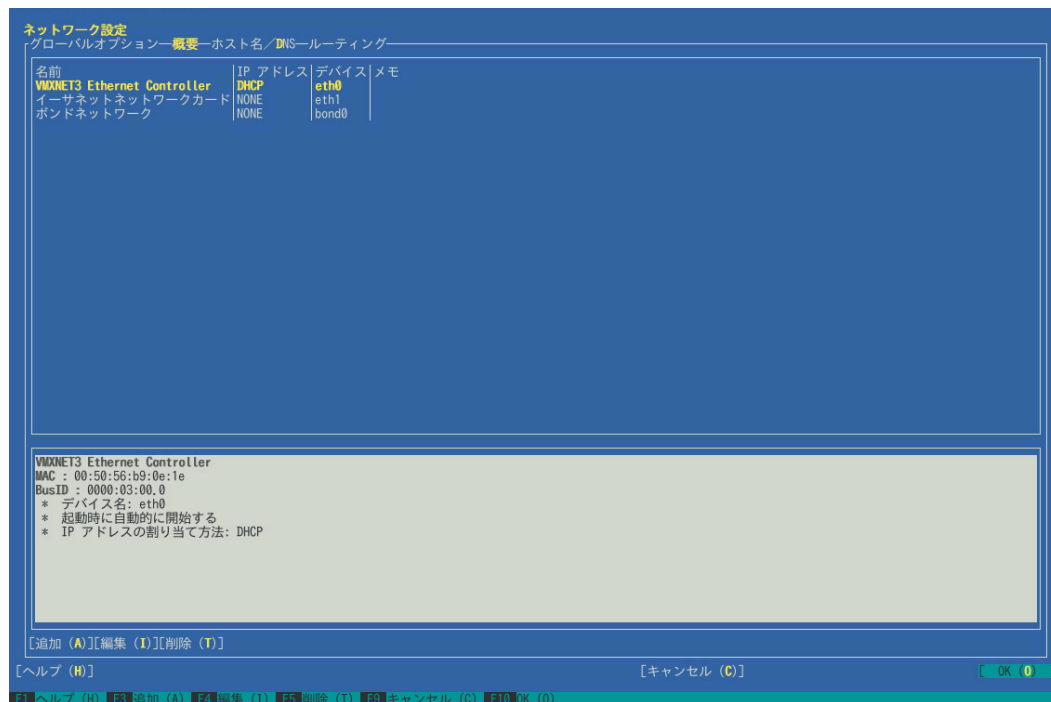
14.2 ボンディングスレーブのホットプラグ

ボンディングスレーブのインタフェースを別のものに置き換える必要が生じることがあります。たとえば、それぞれのネットワークデバイスに常に障害が発生する場合などです。解決方法として、ボンディングスレーブのホットプラグを設定します。デバイスをMACアドレスではなくバスIDによって一致させるために、`udev`ルールを変更する必要もあります。これにより、ハードウェアが許可する場合は、不具合のあるハードウェア(同じスロットにあるのにMACアドレスが異なるネットワークカードなど)を置き換えることができます。

手順 14.1: YASTによるボンディングスレーブのホットプラグの設定

手動の設定を行う場合は、『SUSE Linux Enterprise High Availability Extension Administration Guide』、「Basic Networking」の章の「.Hotplugging of Bonding Slaves」というセクションを参照してください。

1. `root`としてYaSTを開始し、システム > ネットワーク設定の順に選択します。
2. ネットワーク設定で、Overviewタブに切り替えて、すでに設定済みのデバイスを表示します。ボンディングスレーブがすでに設定済みの場合、メモ列にそのことが示されます。



3. 1つのボンディングデバイスに集められたEthernetデバイスのそれぞれに対して、次の手順を実行します。
 - a. 選択するデバイスを選択して、編集をクリックします。Network Card Setupダイアログが開きます。
 - b. 一般タブに切り替えて、デバイスのアクティブ化が「ホットプラグ」に設定されていることを確認します。
 - c. ハードウェアタブに切り替えます。
 - d. Udevルールで、変更をクリックしてBusIDオプションを選択します。
 - e. OKおよび次へをクリックして、ネットワーク設定ダイアログのOverviewタブに戻ります。この時点でEthernetデバイスエントリをクリックすると、下のペインにバスIDを含むデバイスの詳細が表示されます。
4. OKをクリックして変更を確定し、ネットワーク設定を終了します。

ブート時にネットワークセットアップはホットプラグスレーブを待機しませんが、ボンドの準備が整うのを待機します。これには少なくとも1つのスレーブが利用可能であることが必要です。スレーブインタフェースの1つがシステムから削除されると(NICドライバからアンバインド、NICドライバの`rmmod`、または実際のPCIホットプラグ取り外し)、カーネルによって

ボンドから自動的に削除されます。システムに新しいカードが追加されると(スロットのハードウェアが置換されると)、udevは、バスベースの永続名規則を適用することで名前を変更し、ifupを呼び出します。ifup呼び出しによって、ボンドに自動的に追加されます。

14.3 詳細の参照先

全モードおよび多数のオプションの詳細については、「Linux Ethernet Bonding Driver HOWTO」に記載されています。これは、kernel-sourceパッケージをインストールした後に参照できる/usr/src/linux/Documentation/networking/bonding.txtファイルの内容です。

High Availabilityセットアップの場合は、そのファイルで説明されているmiimonおよびuse_carrierオプションが特に重要です。

15 負荷分散

「負荷分散」によって、外部のクライアントからは、サーバのクラスタが1つの大きな高速サーバであるかのように見えます。この単一サーバのように見えるサーバは、「仮想サーバ」と呼ばれます。このサーバは、着信要求をディスパッチする1つ以上のロードバランサと実際のサービスを実行しているいくつかの実際のサーバで構成されます。High Availability Extensionの負荷分散設定によって、高度にスケーラブルで可用性の高いネットワークサービス(Web、キャッシュ、メール、FTP、メディア、VoIPなど)を構築できます。

15.1 概念の概要

High Availability Extensionは、負荷分散の2つのテクノロジー(Linux仮想サーバ(LVS)およびHAProxy)をサポートしています。これらの主な相違点は、Linux仮想サーバがOSI第4層(トランスポート)でカーネルのネットワーク層を設定するのに対し、HAProxyは第7層(アプリケーション)のユーザスペースで実行されることにあります。このように、Linux仮想サーバは、より少ないリソースで、より高い負荷を処理します。それに対してHAProxyは、トラフィックを調査し、SSL停止を実行して、トラフィックのコンテンツに基づいたディパッチに関する決定を行います。

一方、Linux仮想サーバには、IPVS (IP Virtual Server)およびKTCPPVS (Kernel TCP Virtual Server)という2つの異なるソフトウェアが組み込まれています。IPVSは第4層の負荷分散を提供するのに対し、KTCPPVSは第7層の負荷分散を提供します。

この項では、高可用性と組み合わせた負荷分散について概説してから、Linux仮想サーバとHAProxyについて簡単に説明します。最後に、追加情報を紹介します。

実際のサーバとロードバランサは、高速LANまたは地理的に分散されたWANのいずれでも、相互に接続できます。ロードバランサは、さまざまなサーバに要求をディスパッチします。ロードバランサによって、クラスタの平行サービスが1つのIPアドレス(仮想IPアドレスまたはVIP)上の仮想サービスであるかのように見えます。要求のディスパッチでは、IP負荷分散技術か、アプリケーションレベル負荷分散技術を使用できます。クラスタ内のノードのトランスペアレントな追加または削除によって、システムのスケーラビリティが達成されます。

ノードまたはサービスの障害検出と仮想サーバシステム全体の適切な再設定によって、常に高い可用性が実現されます。

いくつかの負荷分散戦略があります。ここに、Linux仮想サーバに適した第4層の各戦略を示します。

- **ラウンドロビン.** 最も簡単な戦略は、各接続を異なるアドレスに順番に指定することです。たとえば、DNSサーバは指定のホスト名に対するいくつかのエントリを持つことができます。DNSラウンドロビンでは、DNSサーバは循環しながらそれらのエントリすべてを順番に返します。このように、異なるクライアントは異なるアドレスを表示します。
- **「最良の」サーバの選択.** これにはいくつかのデメリットがありますが、「応答する最初のサーバ」または「負荷の最も少ないサーバ」アプローチで分散を実装できます。
- **サーバあたりの接続数の分散.** ユーザとサーバ間のロードバランサは、複数のサーバ間でユーザ数を分割できます。
- **位置情報.** 近くのサーバにクライアントをダイレクトすることができます。

ここに、HAProxyに適した第7層の各戦略を示します。

- **URI.** HTTPコンテンツを調査し、この特定のURIに最適なサーバにディスパッチします。
- **URLパラメータ、RDPクッキー.** セッションパラメータ(ポストパラメータの場合もある)、またはRDP(リモートデスクトッププロトコル)セッションクッキーのHTTPコンテンツを調査し、このセッションを提供するサーバにディパッチします。

一部の重複はありますが、HAProxyはLVS/ipvsadmが不十分なシナリオで使用できます(およびその逆もあり)。

- **SSL停止.** フロントエンドロードバランサは、SSL層を処理できます。このため、クラウドノードは、SSLキーにアクセスする必要はなく、ロードバランサのSSLアクセラレータを利用できます。
- **アプリケーションレベル.** HAProxyはアプリケーションレベルで動作するため、コンテンツストリームによって負荷分散の決定に影響を与えることができます。これにより、クッキーや他のフィルタに基づいた永続化が許可されます。

一方、LVS/ipvsadmは、HAProxyで完全に置き換えることはできません。

- LVSは、ロードバランサがインバウンドストリーム内にものみ配置される「ダイレクトルーティング」をサポートし、アウトバウンドトラフィックは直接クライアントにルーティングされます。これにより、非対称環境でのスループットがかなり向上する可能性があります。
- LVSは、(conntrackdを介した)ステートフルな接続テーブルレプリケーションをサポートしています。これにより、クライアントおよびサーバに透過なロードバランサのフェールオーバーが可能になります。

15.2 Linux仮想サーバによる負荷分散の設定

以降のセクションでは、主要なLVSのコンポーネントと概念の概要を示します。その後、High Availability ExtensionでのLinux仮想サーバのセットアップ方法について説明します。

15.2.1 Director

LVSの主要コンポーネントは、ip_vs (またはIPVS)カーネルコードです。これはデフォルトカーネルの一部であり、Linuxカーネル(第4層スイッチング)内のトランスポート層負荷分散を実装します。IPVSコードを含むLinuxカーネルを実行するノードは、「ディレクター」と呼ばれます。ディレクターで実行されるIPVSコードは、LVSの必須機能です。

クライアントがディレクターに接続すると、着信要求がすべてのクラスタノードに負荷分散されます。つまり、ディレクターは、変更されたルーティングルール(LVSを機能させる)セットを使用して、パケットを実サーバに転送します。たとえば、ディレクターは、接続の送受信端でないと、受信確認を送信しません。ディレクターは、エンドユーザから実サーバ(要求を処理するアプリケーションを実行するホスト)にパケットを転送する特殊なルータとして動作します。

15.2.2 ユーザスペースのコントローラとデーモン

ldirectordデーモンは、Linux仮想サーバを管理し、負荷分散型仮想サーバのLVSクラスタ内の実サーバを監視するユーザスペースデーモンです。設定ファイル(以下を参照)は、仮想サービスとそれらに関連付けられた実サーバを指定し、LVSリダイレクタとしてサーバを設定する方法をldirectordに指示します。このデーモンは、その初期化時にクラスタの仮想サービスを生成します。

ldirectordデーモンは、既知のURLを定期的に要求し、応答を確認することにより、実サーバのヘルスを監視します。障害が発生した実サーバは、ロードバランサで使用可能なサーバのリストから削除されます。サービス監視は、ダウンしていたサーバが回復し、再度機能していることを検出すると、そのサーバを使用可能サーバリストに戻します。すべての実サーバがダウンする場合、Webサービスのリダイレクト先にするフォールバックサーバを指定できます。通常、フォールバックサーバは、ローカルホストであり、Webサービスが一時的に使用できないことについて緊急ページを表示します。

ldirectordはipvsadmツール(パッケージ ipvsadm)を使用して、Linuxカーネルの仮想サーバテーブルを操作します。

15.2.3 パケット転送

ディレクターがクライアントから実サーバにパケットを送信する方法は、3つあります。

Network Address Translation (NAT)

着信要求は仮想IPで着信します。宛先のIPアドレスとポートを、選択した実サーバのIPアドレスとポートに変更することで、着信要求は実サーバに転送されます。実サーバはロードバランサに応答を送信し、そのロードバランサが宛先IPアドレスを変更して、応答をクライアントへ転送します。その結果、エンドユーザは予期されたソースから応答を受信します。すべてのトラフィックはロードバランサを通過するので、通常、ロードバランサがクラスタのボトルネックになります。

IPトンネリング(IP-IPカプセル化)

IPトンネリングでは、あるIPアドレスにアドレス指定されたパケットを別のアドレス(別のネットワーク上でも可能)にリダイレクトできます。LVSは、IPトンネルを介して実サーバに要求を送信し(別のIPアドレスにリダイレクト)、実サーバは、独自のルーティングテーブルを使用して、クライアントに直接応答します。クラスタメンバは、さまざまなサブネットに属することができます。

直接ルーティング

エンドユーザからのパケットを、直接、実サーバに転送します。IPパケットは変更されないで、仮想サーバのIPアドレスのトラフィックを受け付けるように、実サーバを設定する必要があります。実サーバからの応答は、直接、クライアントに送信されます。実サーバとロードバランサは、同じ物理ネットワークセグメントに属する必要があります。

15.2.4 スケジューリングアルゴリズム

クライアントから要求された新しい接続に使用する実サーバの決定は、さまざまなアルゴリズムを使用して実装されます。それらは、モジュールとして使用可能であり、特定のニーズに合わせて調整できます。使用可能なモジュールの概要については、[ipvsadm\(8\)](#)のマニュアルページを参照してください。ディレクターは、クライアントから接続要求を受信すると、「スケジューラ」に基づいて実際のサーバをクライアントに割り当てます。スケジューラは、IPVSカーネルコードの一部として、次の新しい接続を取得する実際のサーバを決定します。

Linux仮想サーバのスケジューリングアルゴリズムの詳細については、<http://kb.linuxvirtualserver.org/wiki/IPVS> を参照してください。また、[ipvsadm](#)のマニュアルページで `--scheduler` を検索してください。

関連するHAProxy負荷分散戦略については、<http://www.haproxy.org/download/1.6/doc/configuration.txt> を参照してください。

15.2.5 YaSTによるIP負荷分散の設定

YaST IP負荷分散モジュールを使用して、カーネルベースのIP負荷分散を設定できます。このモジュールは、`ldirectord`のフロントエンドです。

IP負荷分散ダイアログにアクセスするには、`root`としてYaSTを開始し、高可用性 > IP負荷分散の順に選択します。または、コマンドラインで「**yast2 ip1b**」と入力して、`root`としてYaSTクラスタモジュールを起動します。

デフォルトのインストールには、環境設定ファイル`/etc/ha.d/ldirectord.cf`は含まれていません。このファイルは、YaSTモジュールによって作成されます。YaSTモジュール内で使用できるタブは、設定ファイル`/etc/ha.d/ldirectord.cf`の構造、グローバルオプションの定義、および仮想サービス用オプションの定義に対応しています。

設定例とその結果のロードバランサ/実サーバ間のプロセスについては、[例15.1「単純なldirectord設定」](#)を参照してください。



注記: グローバルパラメータと仮想サーバパラメータ

特定のパラメータを仮想サーバセクションとグローバルセクションの両方で指定した場合は、仮想サーバセクションで定義した値が、グローバルセクションで定義した値に優先します。

手順 15.1: グローバルパラメータの設定

次の手順では、重要なグローバルパラメータの設定方法を示します。個々のパラメータ（および、ここに記載されていないパラメータ）の詳細については、ヘルプをクリックするか、`ldirectord`のマニュアルページを参照してください。

1. 確認間隔で、`ldirectord`が各実サーバに接続していて、それらがまだオンラインかどうかを確認する間隔を定義します。
2. 確認タイムアウトで、最後の確認後に実サーバが応答する期限を設定します。
3. 障害発生回数では、`ldirectord`が、何回、実サーバに要求すると、確認が失敗したと見なされるか定義できます。
4. ネゴシエーションタイムアウトで、ネゴシエーション確認のタイムアウトを秒単位で定義します。
5. フォールバックで、すべての実サーバがダウンした場合にWebサービスのリダイレクト先にするWebサーバのホスト名とIPアドレスを入力します。
6. 実サーバへの接続ステータスがかわったら、システムにアラートを送信させたい場合は、有効な電子メールアドレスを電子メールアラートに入力します。

7. 電子メールアラート頻度で、実サーバにアクセスできない状態が続く場合、何秒後に電子メールアラートを繰り返すか定義します。
8. 電子メールアラートのステータスで、電子メールアラートを送信する必要のあるサーバのステータスを指定します。複数の状態を定義する場合は、カンマで区切ったリストを使用します。
9. 自動リロードで、変更の有無について、`ldirectord`に設定ファイルを継続的に監視させるかどうか定義します。yesに設定した場合は、変更のたびに、設定ファイルが自動的にリロードされます。
10. 休止スイッチで、障害が発生した実サーバをカーネルのLVSテーブルから削除するかどうか定義します。はいに設定すると、障害のあるサーバは削除されません。代わりに、それらの重み付けが0に設定され、新しい接続が受け入れられなくなります。すでに確立している接続は、タイムアウトするまで持続します。
11. ロギングに代替パスを使用する場合は、ログファイルでログファイルのパスを指定します。デフォルトでは、`ldirectord`は、そのログファイルを `/var/log/ldirectord.log` に書き込みます。

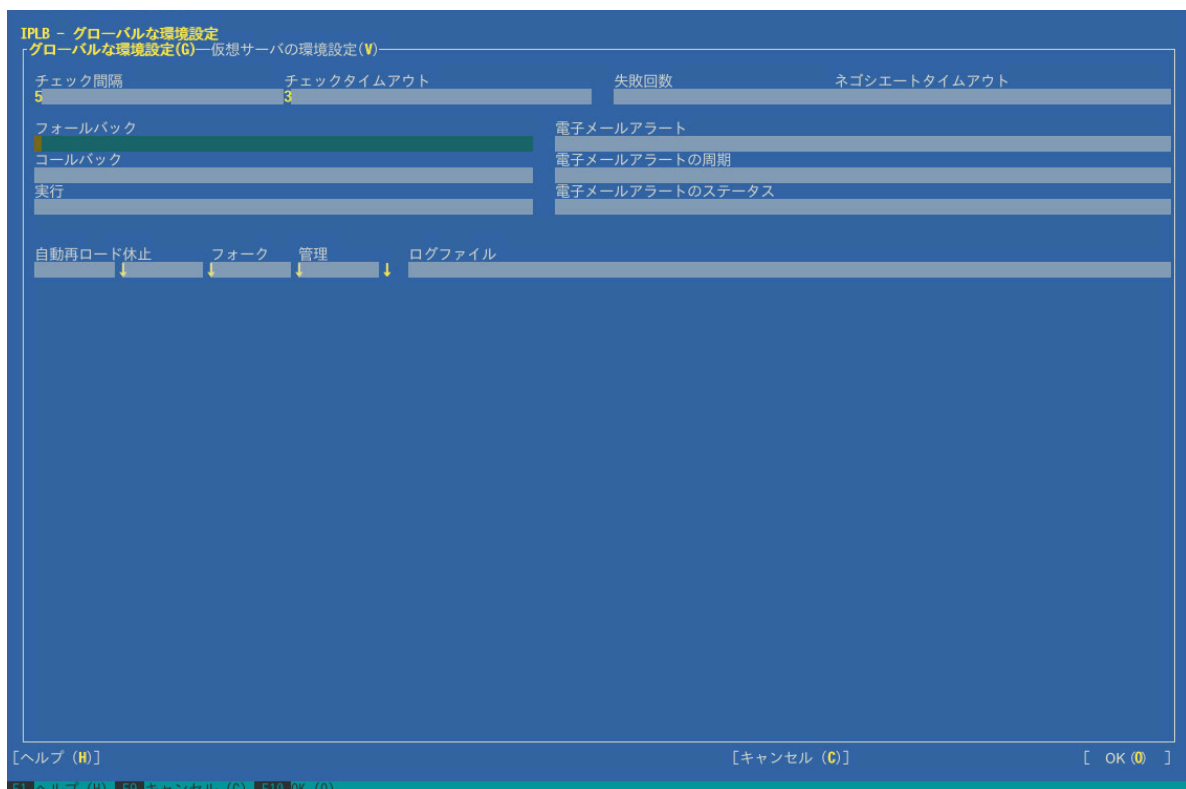


図 15.1: YAST IP負荷分散 - グローバルパラメータ

手順 15.2: 仮想サービスの設定

仮想サービスごとに、2、3のパラメータを定義することによって、1つ以上の仮想サービスを設定できます。次の手順で、仮想サービスの重要なパラメータを設定する方法を示します。個々のパラメータ(および、ここに記載されていないパラメータ)の詳細については、ヘルプをクリックするか、[ldirectord](#)のマニュアルページを参照してください。

1. YaST IP負荷分散モジュール内で、仮想サーバ設定タブに切り替えます。
2. 追加で新しい仮想サーバを追加するか、編集で既存の仮想サーバを編集します。新しいダイアログに、使用可能なオプションが表示されます。
3. 仮想サーバで、共有仮想IPアドレス(IPv4またはIPv6)とポートを入力します。これらのアドレスとポートで、ロードバランサと実サーバをLVSとしてアクセスできます。IPアドレスとポート番号の代わりに、ホスト名とサービスも指定できます。または、ファイアウォールマークを使用することもできます。ファイアウォールマークは、[VIP:port](#)サービスの任意の集まりを1つの仮想サービスにまとめる方法です。
4. 実サーバで実際のサーバを指定するには、サーバのIPアドレス(IPv4、IPv6、またはホスト名)、ポート(またはサービス名)、および転送方法を入力する必要があります。転送方法は、[gate](#)、[ipip](#)、または[masq](#)のいずれかにする必要があります([15.2.3項「パケット転送」](#)参照)。
追加ボタンをクリックし、実サーバごとに必要な引数を入力します。
5. 確認タイプで、実サーバがまだアクティブかどうかをテストするために実行する必要がある確認のタイプを選択します。たとえば、要求を送信し、応答に予期どおりの文字列が含まれているかどうか確認するには、[「ネゴシエーション」](#)を選択します。
6. 確認のタイプを [「ネゴシエーション」](#) に設定した場合は、監視するサービスのタイプも定義する必要があります。サービスドロップダウンボックスから選択してください。
7. 要求で、確認間隔中に各実サーバで要求されるオブジェクトへのURLを入力します。
8. 実サーバからの応答に一定の文字列(「I'm alive」メッセージ)が含まれているかどうか確認する場合は、一致する必要がある正規表現を定義します。正規表現を受信に入力します。実サーバからの応答にこの表現が含まれている場合、実サーバはアクティブとみなされます。
9. [ステップ 6](#)で選択したサービスのタイプによっては、認証のためのパラメータをさらに指定する必要があります。認証タイプタブに切り替えて、ログイン、パスワード、データベース、またはシークレットなどの詳細を入力します。詳細については、YaSTヘルプのテキストか、[ldirectord](#)のマニュアルページを参照してください。

10. その他タブに切り替えます。
11. ロードに使用するスケジューラを選択します。使用可能なスケジューラについては、[ipvsadm\(8\)](#)のマニュアルページを参照してください。
12. 使用するプロトコルを選択します。仮想サービスをIPアドレスとポートとして指定する場合は、プロトコルをtcpまたはudpのどちらかにする必要があります。仮想サービスをファイアウォールマークとして指定する場合は、プロトコルをfwmにする必要があります。
13. 必要な場合は、さらにパラメータを定義します。OKを選択して、設定を確認します。YaSTが設定を/etc/ha.d/ldirectord.cfに書き込みます。

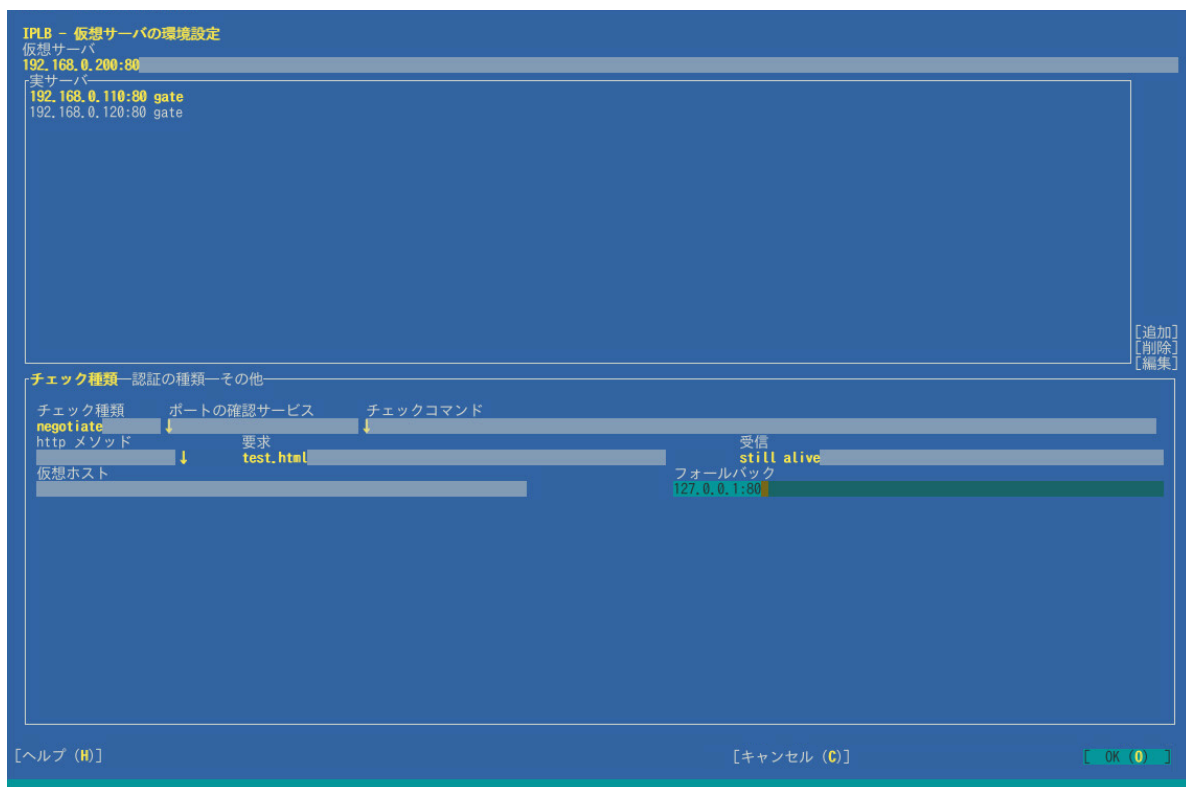


図 15.2: YAST IP負荷分散 - 仮想サービス

例 15.1: 単純なLDIRECTORD設定

図15.1「YaST IP負荷分散 - グローバルパラメータ」と図15.2「YaST IP負荷分散 - 仮想サービス」で示された値を使用すると、次のような設定になり、[/etc/ha.d/ldirectord.cf](#)で定義されます。

```
autoreload = yes ①
checkinterval = 5 ②
checktimeout = 3 ③
```

```
quiescent = yes ④
virtual = 192.168.0.200:80 ⑤
checktype = negotiate ⑥
fallback = 127.0.0.1:80 ⑦
protocol = tcp ⑧
real = 192.168.0.110:80 gate ⑨
real = 192.168.0.120:80 gate ⑨
receive = "still alive" ⑩
request = "test.html" ⑪
scheduler = wlc ⑫
service = http ⑬
```

- ① ldirectordが変更の有無について設定ファイルを継続的に確認するように定義します。
- ② 実サーバがまだオンラインかどうか確認するため、ldirectordが各実サーバに接続する間隔。
- ③ 最後の確認後、実サーバが応答しなければならない時間的な期限
- ④ 障害が発生した実サーバをカーネルのLVSテーブルから削除せず、代わりに、それらのサーバの重み付けを0に設定します。
- ⑤ LVSの仮想IPアドレス(VIP)。LVSはポート80で使用できます。
- ⑥ 実サーバがまだアクティブかどうかをテストするための確認のタイプ。
- ⑦ このサービス用のすべての実サーバがダウンしている場合に、Webサービスのリダイレクト先にするサーバ。
- ⑧ 使用するプロトコル。
- ⑨ ポート80で利用できる2つの実サーバが定義されています。パケットの転送方法がgateなので、直接ルーティングが使用されます。
- ⑩ 実サーバからの応答文字列内で一致する必要がある正規表現。
- ⑪ 確認間隔中に、各実サーバで要求されるオブジェクトへのURI。
- ⑫ 負荷分散に使用するスケジューラが選択されています。
- ⑬ 監視するサービスのタイプ

この設定を使用すると、次のような処理フローになります: ldirectordが、5秒ごとに(②)各実サーバに接続し、⑨と⑪で指定されているように、192.168.0.110:80/test.htmlまたは192.168.0.120:80/test.htmlを要求します。予期されたstill alive文字列(⑩)を、最後の確認から3秒以内(③)に実サーバから受信しない場合は、実サーバが使用可能なサーバのプールから削除されます。ただし、quiescent=yesが設定されているので(④)、実サーバは、LVSテーブルからは削除されません。代わりに、その重み付けが0に設定されます。その結果、この実サーバへの新しい接続は受け付けられなくなります。すでに確立されている接続は、タイムアウトするまで持続します。

15.2.6 追加設定

YaSTによる`ldirectord`の設定に加えて、LVS設定を完了するには、次の条件を満たす必要があります。

- 実サーバは、必要なサービスを提供するように正しく設定します。
- 負荷分散サーバは、IP転送を使用して実サーバにトラフィックをルーティングする必要があります。実サーバのネットワーク設定は、選択したパケット転送方法によって左右されます。
- 負荷分散サーバをシステム全体のシングルポイント障害にしないため、ロードバランサのバックアップを1つ以上セットアップする必要があります。クラスタ設定では、`ldirectord`にプリミティブリソースを設定して、ハードウェア障害の場合に`ldirectord`が他のサーバにフェールオーバーできるようにします。
- ロードバランサのバックアップにも、その作業を達成するために、`ldirectord`設定ファイルが必要なので、ロードバランサのバックアップとして使用するすべてのサーバ上で`/etc/ha.d/ldirectord.cf`が使用できるようにします。設定ファイルは、4.5項「すべてのノードへの設定の転送」で説明されているように、Csync2で同期できます。

15.3 HAProxyによる負荷分散の設定

次のセクションでは、HAProxyの概要とHigh Availabilityでのセットアップ方法について説明します。ロードバランサは、すべての要求をそのバックエンドサーバに分配します。あるマスタで障害が発生すると、スレーブがマスタになることを意味する、アクティブ/パッシブとして設定されます。このようなシナリオでは、ユーザは中断したことに気付きません。

このセクションでは、次のセットアップを使用します。

- ロードバランサ、IPアドレス `192.168.1.99`
- 仮想の浮動IPアドレス `192.168.1.99`
- サーバ(通常はWebコンテンツ用) `www.example1.com` (IP: `192.168.1.200`) および `www.example2.com` (IP: `192.168.1.201`)

HAProxyを設定するには、次の手順に従います。

1. 次のように `haproxy` パッケージの次のアップデートで加える変更は保存されません。
2. 次のコンテンツを含む `/etc/haproxy/haproxy.cfg` ファイルを作成します。

```
global ❶
```

```

maxconn 256
daemon

defaults ❷
    log      global
    mode     http
    option   httplog
    option   dontlognull
    retries  3
    option   redispatch
    maxconn  2000
    timeout  connect    5000  ❸
    timeout  client     50s   ❹
    timeout  server     50000 ❺

frontend LB
    bind 192.168.1.99:80 ❻
    reqadd X-Forwarded-Proto:\ http
    default_backend LB

backend LB
    mode http
    stats enable
    stats hide-version
    stats uri /stats
    stats realm Haproxy\ Statistics
    stats auth haproxy:password ❼
    balance roundrobin ❸
    option httpclose
    option forwardfor
    cookie LB insert
    option httpchk GET /robots.txt HTTP/1.0
    server web1-srv 192.168.1.200:80 cookie web1-srv check
    server web2-srv 192.168.1.201:80 cookie web2-srv check

```

- ❶ プロセスワイドでOS固有のオプションを含むセクション。

maxconn

プロセスあたりの同時接続の最大数。

デーモン

HAProxyがバックグラウンドで実行する推奨モード。

- ❷ セクションの宣言後に、他のすべてのセクションのデフォルトパラメータを設定するセクション。次の重要な行があります。

redispatch

接続が失敗した場合にセッションの再ディストリビューションを有効または無効にします。

log

イベントおよびトラフィックのログ記録を有効にします。

mode http

HTTPモードで動作します(HAProxyの推奨モード)このモードでは、サーバへの接続が実行される前に要求が分析されます。RFCに準拠しない要求は拒否されます。

option forwardfor

HTTP X-Forwarded-Forヘッダを要求に追加します。クライアントのIPアドレスを維持する場合は、このオプションが必要です。

- ③ サーバへの接続試行が成功するまで待機する最大時間。
- ④ クライアント側の最大非アクティブ時間。
- ⑤ サーバ側の最大非アクティブ時間。
- ⑥ フロントエンドおよびバックエンドセクションを1つに結合するセクション。

balance leastconn

負荷分散アルゴリズムを定義します。 <http://cbonte.github.io/haproxy-dconv/configuration-1.5.html#4-balance> を参照してください。

stats enable,

stats auth

(stats enableを使用して)統計レポートングを有効にします。authオプションは、特定のアカウントに対して認証された統計のログを記録します。

- ⑦ HAProxy Statisticレポートページの認証情報。
- ⑧ 負荷分散はラウンドロビン処理で動作します。

3. 設定ファイルをテストします。

```
root # haproxy -f /etc/haproxy/haproxy.cfg -c
```

4. Csync2の設定ファイル/etc/csync2/csync2.cfgに次の行を追加して、HAProxy設定ファイルが含まれていることを確認します。

```
include /etc/haproxy/haproxy.cfg
```


5. それを同期します。

```
root # csync2 -f /etc/haproxy/haproxy.cfg
root # csync2 -xv
```



注記

Csync2の設定部分は、HAノードが`ha-cluster-bootstrap`を使用して設定されたことを想定しています。詳細については、『インストールおよびセットアップクイックスタート』を参照してください。

6. Pacemakerによって起動されるため、HAProxyが両方のロードバランサ(`alice`および`bob`)で無効になっていることを確認します。

```
root # systemctl disable haproxy
```

7. 新しいCIBを設定します。

```
root # crm configure
crm(live)# cib new haproxy-config
crm(haproxy-config)# primitive haproxy systemd:haproxy \
    op start timeout=120 interval=0 \
    op stop timeout=120 interval=0 \
    op monitor timeout=100 interval=5s \
    meta target-role=Started
crm(haproxy-config)# primitive vip IPAddr2 \
    params ip=192.168.1.99 nic=eth0 cidr_netmask=23 broadcast=192.168.1.255 \
    op monitor interval=5s timeout=120 on-fail=restart
crm(haproxy-config)# group g-haproxy vip haproxy
```

8. 新しいCIBを確認し、エラーがあれば修正します。

```
crm(haproxy-config)# verify
```

9. 新しいCIBをコミットします。

```
crm(haproxy-config)# cib use live
crm(live)# cib commit haproxy-config
```

15.4 詳細の参照先

- <http://www.haproxy.org> 
- <http://www.linuxvirtualserver.org/> にあるプロジェクトのホームページ。

- ldirectordの詳細については、その総合的なマニュアルページを参照してください。
- LVS Knowledge Base: http://kb.linuxvirtualserver.org/wiki/Main_Page .

16 Geoクラスタ(マルチサイトクラスタ)

SUSE® Linux Enterprise High Availability Extension 15 SP3は、ローカルクラスタとメトロエリアクラスタのほかに、地理的に離れたクラスタ(Geoクラスタ。マルチサイトクラスタとも呼ばれます)もサポートしています。これは、それぞれひとつのローカルクラスタで持った複数の地理的に離れたサイトを持てることを意味します。これらクラスタ間のフェールオーバーは、より高いレベルのエンティティである**booth**によって管理されます。Geoクラスタの使用方法和設定方法の詳細については、項目「Geo Clusteringのクイックスタート」と『管理ガイド』を参照してください。

17 保守タスクの実行

クラスタノードで保守タスクを実行するには、そのノードで実行中のリソースを停止し、それらを移動するか、あるいはそのノードをシャットダウンするか再起動する必要がある場合があります。また、クラスタからリソースの制御を一時的に引き継ぐか、またはリソースを実行中のままにしてクラスタサービスを停止することもある場合があります。

この章では、負の影響を及ぼすことなくクラスタノードを手動で切断する方法について説明します。また、クラスタスタックが保守タスクを実行するために提供するさまざまなオプションの概要についても説明します。

17.1 クラスタノードを切断する意味

SUSE Linux Enterprise Server High Availability Extension 15 SP2では、クラスタサービスの開始および停止方法が変更されました。SUSEでは、`crm`シェルの使用を推奨しています。`systemctl`を使用した古いコマンドはまだ使用できますが、熟練ユーザにのみ推奨されません。詳細については、SUSEのブログ記事「<https://www.suse.com/c/suse-high-availability-cluster-services-how-to-stop-start-or-view-the-status/>」を参照してください。

ステータスを開始、停止、または表示するための推奨される方法は次のとおりです。

`crm cluster start`

1つのノードでクラスタサービスを開始します

`crm cluster stop`

1つのノードでクラスタサービスを停止します

`crm cluster restart`

1つのノードでクラスタサービスを再起動します

`crm cluster status`

1つのノードでクラスタスタックのステータスを表示します

先に示したコマンドは、ユーザ`root`、または必要な権限を持つユーザとして実行します。

クラスタノードをシャットダウンまたは再起動する(またはノード上でPacemakerサービスを停止する)場合、次のプロセスがトリガされます。

- ノード上で実行されているリソースは停止されるか、ノードから移動します。
- リソースの停止が失敗するか、タイムアウトする場合、STONITHメカニズムはノードをフェンシングし、シャットダウンします。

手順 17.1: クラスタノードの手動による再起動

ノードをシャットダウンまたは再起動する前に、順序だった方法でノードのサービスをオフにしたい場合は、次の操作を実行します。

1. 再起動またはシャットダウンするノードで、rootまたは同等な権限でログインします。
2. ノードをstandbyモードにします。

```
root # crm -w node standby
```

このようにすると、サービスはPacemakerクラスタサービスのシャットダウンタイムアウトによって制限されることなく、ノードをオフに移行できます。

3. 以下を使用してクラスタの状態を確認します。

```
root # crm status
```

standbyモード状態の各ノードが示されます。

```
[...]
Node bob: standby
[...]
```

4. そのノードでクラスタサービスを停止します。

```
root # crm cluster stop
```

5. ノードを再起動します。

ノードが再びクラスタに参加しているかどうかを確認するには:

1. rootまたは同等の権限でノードにログインします。
2. クラスタサービスが開始されているかどうかを確認します。

```
root # crm cluster status
```

3. 開始されていない場合は、開始します。

```
root # crm cluster start
```

4. 以下を使用してクラスタの状態を確認します。

```
root # crm status
```

ノードが再びオンラインになっていることが示されます。

17.2 保守タスクのためのさまざまなオプション

Pacemakerはシステム保守を実行するためのさまざまなオプションを提供しています。

クラスタを保守モードにする

グローバルクラスタプロパティ `maintenance-mode` により、すべてのリソースを瞬時に保守状態にすることができます。クラスタはモニタリングを停止し、ステータスが追跡されなくなります。

ノードを保守モードにする

このオプションにより、特定のノードで実行されているすべてのリソースを瞬時に保守状態にすることができます。クラスタはモニタリングを停止し、ステータスが追跡されなくなります。

ノードをスタンバイモードにする

スタンバイモードのノードはリソースを実行できなくなります。ノード上で実行されているすべてのリソースは移動するか停止されます(他のノードがリソースを実行する資格がない場合)。また、ノード上のすべての監視操作は停止されます(`role="Stopped"`に設定された操作を除く)。

別のノードで実行されているサービスを提供し続けながら、クラスタ内の1台のノードを停止する必要がある場合は、このオプションを使用できます。

リソースを保守モードにする

リソースに対してこのモードが有効な場合、リソースの監視操作はトリガされません。このリソースで管理されるサービスに手動で介入する必要があり、その間にリソースの監視操作をクラスタに実行させない場合は、このオプションを使用します。

リソースを非管理対象モードにする

`is-managed` メタ属性により、リソースを一時的にクラスタスタックによって管理されている状態から「解放」することができます。これは、このリソースによって管理されるサービスに手動で介入できることを意味します(たとえば、コンポーネントを調整するなど)。ただし、クラスタはリソースの「監視」と障害の報告を継続して行います。クラスタによるリソースの「監視」を停止したい場合は、代わりにリソース単位の保守モードを使用します([リソースを保守モードにする](#)を参照してください)。

17.3 保守作業の準備と終了



警告: データ損失の危険

テストまたは保守作業を実行する必要がある場合は、以下の一般的な手順に従います。従わない場合、リソースが順序だった方法で起動できない、クラスタノード間でCIBが同期されない、データ損失などの、望ましくない負の影響が及ぼされるリスクがあります。

1. 開始する前に、17.2項で概説されている、自分の状況に適したオプションを選択します。
2. Hawk2またはcrmshを使用してこのオプションを適用します。
3. 保守タスクまたはテストを実行します。
4. 終了したら、リソース、ノードまたはクラスタを「通常」の操作状態に戻します。

17.4 クラスタを保守モードにする

クラスタをcrmシェル上で保守モードにするには、次のコマンドを使用します。

```
root # crm configure property maintenance-mode=true
```

保守作業が完了した後で、クラスタを通常モードに戻すには、次のコマンドを使用します。

```
root # crm configure property maintenance-mode=false
```

手順 17.2: クラスタをHAWK2を使用して保守モードにする

1. 7.2項「ログイン」で説明したように、Webブラウザを起動してクラスタにログインします。
2. 左のナビゲーションバーで、[クラスタ設定] を選択します。
3. [CRMの環境設定] グループで、空のドロップダウンボックスからmaintenance-mode属性を選択し、プラスアイコンをクリックして追加します。

4. `maintenance-mode=true`を設定するには、`maintenance-mode`の隣のチェックボックスをオンにして、変更を確認します。
5. クラスタ全体の保守作業が完了したら、`maintenance-mode`属性の隣のチェックボックスをオフにします。
この時点から、High Availability Extensionはクラスタ管理をもう一度引き継ぎます。

17.5 ノードを保守モードにする

crmシェル上でノードを保守モードにするには、次のコマンドを使用します。

```
root # crm node maintenance NODENAME
```

保守作業が完了した後で、ノードを通常モードに戻すには、次のコマンドを使用します。

```
root # crm node ready NODENAME
```

手順 17.3: ノードをHAWK2を使用して保守モードにする

1. 7.2項「ログイン」で説明したように、Webブラウザを起動してクラスタにログインします。
2. 左のナビゲーションバーで、[クラスタステータス] を選択します。
3. 個々のノードのビューのいずれかで、ノードの隣のレンチアイコンをクリックして、[保守] を選択します。
4. 保守タスクが終了したら、ノードの横にあるレンチアイコンをクリックして、準備完了を選択します。

17.6 ノードをスタンバイモードにする

ノードをcrmシェル上でスタンバイモードにするには、次のコマンドを使用します。

```
root # crm node standby NODENAME
```

保守作業が完了した後でノードをオンライン状態に戻すには、次のコマンドを使用します。

```
root # crm node online NODENAME
```


手順 17.4: ノードをHAWK2を使用してスタンバイモードにする

1. 7.2項「ログイン」で説明したように、Webブラウザを起動してクラスタにログインします。
2. 左のナビゲーションバーで、[クラスタステータス] を選択します。
3. 個々のノードのビューのいずれかで、ノードの隣のレンチアイコンをクリックして、スタンバイを選択します。
4. ノードの保守タスクを完了します。
5. スタンバイモードを無効化するには、そのノードの隣のレンチアイコンをクリックして準備完了を選択します。

17.7 リソースを保守モードにする

crmシェル上でリソースを保守モードにするには、次のコマンドを使用します。

```
root # crm resource maintenance RESOURCE_ID true
```

保守作業が完了した後で、リソースを通常モードに戻すには次のコマンドを使用します。

```
root # crm resource maintenance RESOURCE_ID false
```

手順 17.5: リソースをHAWK2を使用して保守モードにする

1. 7.2項「ログイン」で説明したように、Webブラウザを起動してクラスタにログインします。
2. 左のナビゲーションバーで、[リソース] を選択します。
3. 保守モードまたは非管理対象モードにするリソースを選択し、そのリソースの隣のレンチアイコンをクリックして、[リソースの編集] を選択します。
4. [メタ属性] カテゴリが開きます。
5. 空のドロップダウンリストからmaintenance属性を選択し、プラスアイコンをクリックして追加します。
6. maintenanceの隣のチェックボックスをオンにして、maintenance属性をyesに設定します。

7. 変更内容を確認します。
8. 該当するリソースの保守作業が完了したら、そのリソースの maintenance 属性の隣のチェックボックスをオフにします。
リソースは、この時点から再びHigh Availability Extensionソフトウェアによって管理されます。

17.8 リソースを非管理対象モードにする

crmシェル上でリソースを非管理対象モードにするには、次のコマンドを使用します。

```
root # crm resource unmanage RESOURCE_ID
```

保守作業が完了した後で再び管理対象モードにするには、次のコマンドを使用します。

```
root # crm resource manage RESOURCE_ID
```

手順 17.6: リソースをHAWK2を使用して非管理対象モードにする

1. 7.2項「ログイン」で説明したように、Webブラウザを起動してクラスタにログインします。
2. 左ナビゲーションバーから、状態を選択し、リソースリストに移動します。
3. 操作列で、変更したいリソースの横にある下矢印アイコンをクリックして編集を選択します。
リソース設定画面が開きます。
4. メタ属性の下で、空のドロップダウンボックスからis-managedエントリを選択します。
5. その値をNoに設定し、適用をクリックします。
6. 保守タスクが終了した後で、is-managedをYesに設定し(デフォルト値です)、変更を適用します。
リソースは、この時点から再びHigh Availability Extensionソフトウェアによって管理されます。

17.9 保守モード中のクラスタノードの再起動



注記: 意味

クラスタまたはノードが保守モードの場合、クラスタリソースを任意に停止したり再起動したりできます。High Availability Extensionはこれらを再起動しようとしません。ノード上のPacemakerサービスを停止する場合、(Pacemakerの管理対象クラスタリソースとして最初に起動された)すべてのデーモンとプロセスの実行は継続されます。

クラスタまたはノードが保守モードのときに、ノード上でPacemakerサービスを起動しようとする場合、Pacemakerはリソースごとに1つのワンショット監視操作(「probe」)を開始し、そのノードで現在どのリソースが実行されているかを評価します。ただし、リソースのステータスを決定する以外の操作は行いません。

クラスタまたはノードが保守モードのときにノードを切断する場合は、次のようにします。

1. 再起動またはシャットダウンするノードで、rootまたは同等な権限でログインします。
2. DLMリソース(またはDLMに依存するその他のリソース)が存在するときは、Pacemakerサービスを停止する前にそれらのリソースを明示的に停止してください。

```
crm(live)resource# stop RESOURCE_ID
```

その理由は、Pacemakerを停止すると、DLMが依存するメンバーシップとメッセージングサービスを持つCorosyncサービスも停止するからです。Corosyncが停止した場合、DLMリソースではスプリットブレインシナリオが発生したと見なされ、フェンシング操作がトリガされます。

3. そのノードでPacemakerサービスを停止します。

```
root # crm cluster stop
```

4. ノードをシャットダウンするか再起動します。

III ストレージとデータレプリケーション

- 18 分散ロックマネージャ(DLM:Distributed Lock Manager) **265**
- 19 OCFS2 **268**
- 20 GFS2 **278**
- 21 DRBD **283**
- 22 Cluster Logical Volume Manager (Cluster LVM) **301**
- 23 クラスタマルチデバイス(Cluster MD) **316**
- 24 Sambaクラスタリング **321**
- 25 Rear (Relax-and-Recover)による障害復旧 **330**

18 分散ロックマネージャ(DLM:Distributed Lock Manager)

カーネル内の分散ロックマネージャ(DLM)は、OCFS2、GFS2、Cluster MD、および Cluster LVM (lvmlockd)によって使用されるベースコンポーネントで、各層でアクティブ/アクティブ構成のストレージが提供されます。

18.1 DLM通信のプロトコル

シングルポイント障害を回避するには、高可用性クラスタに対する冗長通信パスが重要となります。これはDLM通信においても当てはまります。ネットワークボンディング(Link Aggregation Control Protocol、LACP)が何らかの理由で使用できない場合、Corosyncで冗長通信チャンネル(2番目のリング)を定義することを強くお勧めします。詳細については、[手順 4.3「冗長通信チャンネルの定義」](#)を参照してください。

/etc/corosync/corosync.confの設定によって、DLMはその通信にTCPプロトコルを使用するか、SCTPプロトコルを使用するかを判断します。

- `rrp_mode`を`none`に設定する場合(冗長リング設定が無効であることを意味する)、DLMは自動的にTCPを使用します。ただし、冗長通信チャンネルを使用しない場合には、TCPリンクがダウンすると、DLM通信は失敗します。
- `rrp_mode`が`passive`に設定され(これは通常の設定です)、/etc/corosync/corosync.confの2番目の通信リングが正しく設定されている場合、DLMは自動的にSCTPを使用します。この場合、DLMメッセージングには、SCTPによって提供される冗長性機能があります。

18.2 DLMクラスタリソースの設定

DLMはPacemakerからのクラスタメンバーシップサービスを使用し、それらのサービスはユーザスペースで実行されます。したがって、DLMは、クラスタ内の各ノードに存在するクローンリソースとして設定する必要があります。



注記: いくつかの解決策のためのDLMリソース

OCFS2、GFS2、Cluster MD、およびCluster LVM (lvmlockd)のすべてがDLMを使用するため、DLMに1つのリソースを設定するだけで十分です。DLMリソースはクラスタ内のすべてのノード上で実行されるので、リソースはクローンリソースとして設定されます。

OCFS2およびCluster LVMの両方を含むセットアップがある場合、OCFS2およびCluster LVMの両方に「1つの」DLMリソースを設定するだけで十分です。

手順 18.1: DLMのベースグループの設定

設定は複数のプリミティブおよび1つのベースクローンを含むベースグループで設定されます。ベースグループとベースクローンはどちらも、後でさまざまなシナリオで使用できます(例: OCFS2およびCluster LVM)。必要に応じてそれぞれのプリミティブを持つベースグループを拡張する必要があるだけです。ベースグループは内部コロケーションおよび順序付けを持つため、個々のグループ、クローン、その依存性をいくつも指定する必要がなく、セットアップ全体を容易にします。

クラスタ内の1つのノードについて、次の手順を実行してください。

1. シェルを起動し、rootまたは同等のものとしてログインします。
2. crm configureを実行します。
3. 次のコマンドを入力して、DLMのプリミティブリソースを作成します。

```
crm(live)configure# primitive dlm ocf:pacemaker:controld \  
op monitor interval="60" timeout="60"
```

4. DLMリソースおよび追加のストレージ関連のリソース用にベースグループを作成します。

```
crm(live)configure# group g-storage dlm
```

5. g-storageグループのクローンを作成して、すべてのノードで実行できるようにします。

```
crm(live)configure# clone cl-storage g-storage \  
meta interleave=true target-role=Started
```

6. showで変更内容をレビューします。
7. すべて正しければ、commitで変更を送信し、exitでcrmライブ設定を終了します。



注記: STONITHを無効にする際のエラー

STONITHを使用しないクラスタはサポートされていません。グローバルクラスタオプション `stonith-enabled` を テストまたはトラブルシューティングの目的で `false` に設定すると、DLMリソースとそれに依存するすべてのサービス(Cluster LVM、GFS2、OCFS2 など)は起動できません。

19 OCFS2

OCFS 2 (Oracle Cluster File System 2)は、Linux 2.6以降のカーネルに完全に統合されている汎用ジャーナリングファイルシステムです。Oracle Cluster File System 2を利用すれば、アプリケーションバイナリファイル、データファイル、およびデータベースを、共有ストレージ中のデバイスに保管することができます。このファイルシステムには、クラスタ中のすべてのノードが同時に読み書きすることができます。ユーザスペース管理デーモンは、クローンリソースを介して管理され、HAスタック(特に、CorosyncおよびDLM (Distributed Lock Manager))との統合を実現します。

19.1 特長と利点

OCFS2は、たとえば、次のストレージソリューションに使用できます。

- 一般のアプリケーションとワークロード。
- クラスタ中のXENイメージ。Xen仮想マシンと仮想サーバは、クラスタサーバによってマウントされたOCFS2ボリュームに保存できます。これによって、サーバ間でXen仮想マシンを素早く容易に移植できます。
- LAMP (Linux、Apache、MySQL、およびPHP | Perl | Python)スタック。

OCF2は、高パフォーマンスでシンメトリックなパラレルクラスタファイルシステムとして、次の機能をサポートします。

- アプリケーションのファイルを、クラスタ内のすべてのノードで使用できます。ユーザは、クラスタ中のOracle Cluster File System 2ボリュームに1回インストールするだけで構いません。
- すべてのノードが、標準ファイルシステムインタフェースを介して、同時並行的に、ストレージに直接読み書きできるので、クラスタ全体に渡わたって実行されるアプリケーションの管理が容易になります。

- ファイルアクセスがDLMを介して調整されます。ほとんどの場合、DLMによる制御は適切に機能しますが、アプリケーションの設計によっては、アプリケーションとDLMがファイルアクセスの調整で競合すると、スケーラビリティが制限されることがあります。
- すべてのバックエンドストレージで、ストレージのバックアップ機能を利用することができます。共有アプリケーションファイルのイメージを簡単に作成することができるため、災害発生時でも素早くデータを復元することができます。

Oracle Cluster File System 2には、次の機能も用意されています。

- メタデータのキャッシュ処理。
- メタデータのジャーナル処理。
- ノード間にまたがるファイルデータの整合性。
- 最大4KBのマルチブロックサイズ、最大1MBのクラスタサイズ、4PB(ペタバイト)の最大ボリュームサイズをサポートします。
- 32台までのクラスタノードをサポート。
- データベースのパフォーマンスを向上する非同期、直接I/Oのサポート。



注記: OCFS2用サポート

OCFS2は、SUSE Linux Enterprise High Availability Extensionによって提供される、pcmk (Pacemaker)スタックと併用する場合にのみ、SUSEによってサポートされます。o2cbスタックと組み合わせた場合、SUSEはOCFS2をサポートしません。

19.2 OCFS2のパッケージと管理ユーティリティ

SUSE® Linux Enterprise Server 15 SP3上のHigh Availability Extensionには、OCFS2カーネルモジュール(ocfs2)が自動的にインストールされます。OCFS2を使用するには、クラスタ内の各ノードに次のパッケージがインストールされていることを確認してください。 ocfs2-tools および使用しているカーネル用の一致する ocfs2-kmp-* パッケージ。

sapMDCsapVirtHostname ocfs2-tools パッケージには、次に示すOFS2ボリュームの管理ユーティリティがあります。構文については、各マニュアルページを参照してください。

表 19.1: OCFS2ユーティリティ

OCFS2ユーティリティ	説明
debugfs.ocfs2	デバッグのために、OCFS2ファイルシステムの状態を調査します。
fsck.ocfs2	ファイルシステムにエラーがないかをチェックし、必要に応じてエラーを修復します。
mkfs.ocfs2	デバイス上にOCFS2ファイルシステムを作成します。通常は、共有物理/論理ディスク上のパーティションに作成します。
mounted.ocfs2	クラスタシステム上のすべてのOCFS2ボリュームを検出、表示します。OCFS2デバイスをマウントしているシステム上のすべてのノードを検出、表示するか、またはすべてのOCFS2デバイスを表示します。
tuneufs.ocfs2	ボリュームラベル、ノードスロット数、すべてのノードスロットのジャーナルサイズ、およびボリュームサイズなど、OCFS2ファイルのシステムパラメータを変更します。

19.3 OCFS2サービスとSTONITHリソースの設定

OCFS2ボリュームを作成する前に、DLMおよびSTONITHリソースをクラスタ内のサービスとして設定する必要があります。

次の手順では、**crm**シェルを使用してクラスタリソースを設定します。19.6項「[Hawk2でのOCFS2リソースの設定](#)」で説明されているように、リソースの設定にはHawk2を使用することもできます。

手順 19.1: STONITHリソースの設定



注記: 必要なSTONITHデバイス

フェンシングデバイスを設定する必要があります。STONITHなしでは、設定内に配置されたメカニズム(external/sbdなど)は失敗します。

1. シェルを起動し、rootまたは同等のものとしてログインします。
2. 手順11.3「SBDデバイスの初期化」で説明されるとおり、SBDパーティションを作成します。
3. crm configureを実行します。
4. external/sdbをフェンシングデバイスとして設定し、/dev/sdb2を共有ストレージ上のハートビートとフェンシング専用のパーティションにします。

```
crm(live)configure# primitive sbd_stonith stonith:external/sbd \  
params pcmk_delay_max=30 meta target-role="Started"
```
5. showで変更内容をレビューします。
6. すべて正しければ、commitで変更を送信し、exitでcrmライブ設定を終了します。

DLMに対するリソースグループの設定の詳細については、[手順18.1「DLMのベースグループの設定」](#)を参照してください。

19.4 OCFS2ボリュームの作成

19.3項「OCFS2サービスとSTONITHリソースの設定」で説明されているように、DLMクラスタリソースを設定したら、システムがOCFS2を使用できるように設定し、OCFS2ボリュームを作成します。



注記: アプリケーションファイルとデータファイル用のOCFS2ボリューム

一般に、アプリケーションファイルとデータファイルは、異なるOCFS2ボリュームに保存することを推奨します。アプリケーションボリュームとデータボリュームのマウント要件が異なる場合は、必ず、異なるボリュームに保存します。

作業を始める前に、OCFS2ボリュームに使用するブロックデバイスを準備します。デバイスは空き領域のままにしてください。

次に、[手順19.2「OCFS2ボリュームを作成し、フォーマットする」](#)で説明されているように、mkfs.ocfs2で、OCFS2ボリュームを作成し、フォーマットします。そのコマンドの重要なパラメータは、[表19.2「重要なOCFS2パラメータ」](#)に一覧されています。詳細情報とコマンド構文については、mkfs.ocfs2のマニュアルページを参照してください。

表 19.2: 重要なOCFS2パラメータ

OCFS2パラメータ	説明と推奨設定
ボリュームラベル(<u>-L</u>)	異なるノードへのマウント時に、正しく識別できるように、一意のわかりやすいボリューム名を指定します。ラベルを変更するには、 tunefs.ocfs2 ユーティリティを使用します。
クラスタサイズ(<u>-C</u>)	クラスタサイズは、ファイルに割り当てられる、データ保管領域の最小単位です。使用できるオプションと推奨事項については、 mkfs.ocfs2 のマニュアルページを参照してください。
ノードスロット数(<u>-N</u>)	<p>同時にボリュームをマウントできる最大ノード数を指定します。各ノードについて、OCFS2はジャーナルなどの個別のシステムファイルを作成します。ボリュームにアクセスするノードに、リトルエンディアン形式のノード(AMD64/Intel 64など)とビッグエンディアン形式のノード(S/390xなど)が混在しても構いません。</p> <p>ノード固有のファイルは、ローカルファイルと呼ばれます。ローカルファイルには、ノードスロット番号が付加されます。たとえば、<code>journal:0000</code>は、スロット番号0に割り当てられたノードに属します。</p> <p>各ボリュームを同時にマウントすると予想されるノード数に従って、各ボリュームの作成時に、そのボリュームの最大ノードスロット数を設定します。tunefs.ocfs2ユーティリティを使用して、必要に応じてノードスロットの数を増やします。値を減らすことはできず、1つのノードスロットが約100MiBのディスク容量を消費することに注意してください。</p>

OCFS2パラメータ	説明と推奨設定
	-Nパラメータを指定しない場合、ノードスロット数はファイルシステムのサイズに基づいて決定されます。デフォルト値については、 mkfs.ocfs2 のマニュアルページを参照してください。
ブロックサイズ(-b)	ファイルシステムがアドレス可能な領域の最小単位を指定します。ブロックサイズは、ボリュームの作成時に指定します。使用できるオプションと推奨事項については、 mkfs.ocfs2 のマニュアルページを参照してください。
特定機能のオン/オフ(--fs-features)	カンマで区切った機能フラグリストを指定できます。 mkfs.ocfs2 は、そのリストに従って、それらの機能セットを含むファイルシステムを作成しようとします。機能をオンにするには、その機能をリストに入れます。機能をオフにするには、その名前の前に no を付けます。 使用できるすべてのフラグの概要については、 mkfs.ocfs2 のマニュアルページを参照してください。
事前定義機能(--fs-feature-level)	事前定義されたファイルシステム機能セットから選択できます。使用できるオプションについては、 mkfs.ocfs2 のマニュアルページを参照してください。

[mkfs.ocfs2](#)によるボリュームの作成およびフォーマット時に機能を指定しない場合は、[backup-super](#)、[sparse](#)、[inline-data](#)、[unwritten](#)、[metaecc](#)、[indexed-dirs](#)、および[xattr](#)の各機能がデフォルトで有効になります。

手順 19.2: OCFS2ボリュームを作成し、フォーマットする

クラスタノードの「1つ」だけで、次の手順を実行します。

1. 端末ウィンドウを開いて、[root](#)としてログインします。

2. クラスタがオンラインであることをコマンド `crm status` で確認します。
3. `mkfs.ocfs2`ユーティリティを使用して、ボリュームを作成およびフォーマットします。
このコマンドの指定形式については、`mkfs.ocfs2`マニュアルページを参照してください。
たとえば、最大32台のクラスタノードをサポートする新しいOCFS2ファイルシステムを `/dev/sdb1`上に作成するには、次のコマンドを使用します。

```
root # mkfs.ocfs2 -N 32 /dev/sdb1
```

19.5 OCFS2ボリュームのマウント

OCFS2ボリュームは、手動でマウントするか、クラスタマネージャでマウントできます(手順 19.4「クラスタリソースマネージャでOCFS2ボリュームをマウントする」参照)。

手順 19.3: OCFS2ボリュームを手動でマウントする

1. 端末ウィンドウを開いて、`root`としてログインします。
2. クラスタがオンラインであることをコマンド `crm status` で確認します。
3. コマンドラインから、`mount`コマンドを使ってボリュームをマウントします。



警告: 手動マウントによるOCFS2デバイス

OCFS2ファイルシステムをテスト目的で手動マウントした場合、そのファイルシステムは、いったんマウント解除してから、クラスタリソースで使用してください。

手順 19.4: クラスタリソースマネージャでOCFS2ボリュームをマウントする

High AvailabilityソフトウェアでOCFS2ボリュームをマウントするには、クラスタ内で `ocfs2`ファイルシステムのリソースを設定します。次の手順では、`crm`シェルを使用してクラスタリソースを設定します。19.6項「Hawk2でのOCFS2リソースの設定」で説明されているように、リソースの設定にはHawk2を使用することもできます。

1. シェルを起動し、`root`または同等のものとしてログインします。
2. `crm configure`を実行します。
3. OCFS2ファイルシステムをクラスタ内のすべてのノードにマウントするように、Pacemakerを設定します。

```
crm(live)configure# primitive ocfs2-1 ocf:heartbeat:Filesystem \
```

```
params device="/dev/sdb1" directory="/mnt/shared" fstype="ocfs2" \  
op monitor interval="20" timeout="40" \  
op start timeout="60" op stop timeout="60" \  
meta target-role="Started"
```

4. ocfs2-1プリミティブを手順18.1「DLMのベースグループの設定」で作成したg-storageグループに追加します。

```
crm(live)configure# modgroup g-storage add ocfs2-1
```

addサブコマンドは、デフォルトで新しいグループメンバーを追加します。ベースグループの内部コロケーションおよび順序付けによって、Pacemakerは、すでに実行しているdlmリソースも持つノード上でocfs2-1リソースのみ起動します。

5. showで変更内容をレビューします。
6. すべて正しければ、commitで変更を送信し、exitでcrmライブ設定を終了します。

19.6 Hawk2でのOCFS2リソースの設定

crmシェルを使用して、DLM、およびOCFS2のファイルシステムリソースを手動で設定する代わりに、Hawk2のセットアップウィザードのOCFS2テンプレートを使用することもできます。

！ 重要: 手動設定とHawk2との相違点

セットアップウィザードのOCFS2テンプレートには、STONITHリソースの設定が含まれません。ウィザードを使用する場合でも、手順19.1「STONITHリソースの設定」で説明されているように、共有ストレージ上でSBDパーティションを作成し、STONITHリソースを設定する必要があります。

また、HawkセットアップウィザードのOCFS2テンプレートを使用すると、手順18.1「DLMのベースグループの設定」および手順19.4「クラスタリソースマネージャでOCFS2ボリュームをマウントする」で説明されている手動設定とは若干異なるリソース設定になります。

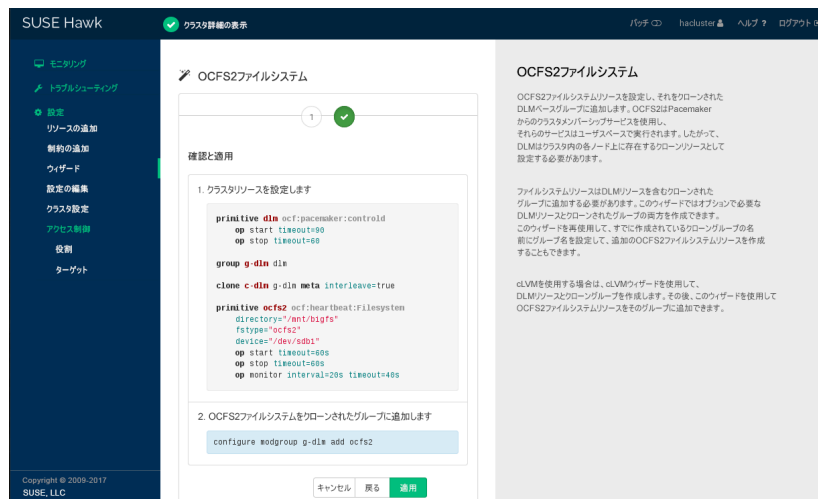
手順 19.5: HAWK2のウィザードでのOCFS2リソースの設定

1. Hawk2にログインします。

```
https://HAWKSERVER:7630/
```

2. 左のナビゲーションバーで、ウィザードを選択します。

3. ファイルシステムカテゴリを展開して、OCFS2 File Systemを選択します。
4. 画面の指示に従います。オプションについての情報が必要な場合には、オプションをクリックすると、Hawk2は簡単なヘルプテキストを表示します。最後の設定手順が完了したら、Verify (検証)を選択して、入力した値を検証します。
CIBに適用する設定スニペットやその他の必要な変更がウィザードに表示されます。



5. 適用予定の変更を確認します。すべてが希望どおりの場合は、変更を適用します。
画面上のメッセージが、アクションに成功したかどうかを示します。

19.7 OCFS2ファイルシステム上でクォータを使用する

OCFS2ファイルシステム上でクォータを使用するには、適切なクォータ機能またはオプションを使用して、ファイルシステムを作成し、マウントします。オプションは`ursquota` (個々のユーザのためのクォータ)または`grpquota` (グループのためのクォータ)です。これらの機能は後ほど、`tunefs.ocfs2`を使用して、マウントされていないファイルシステムで有効にすることもできます。

ファイルシステムで適切なクォータ機能が有効にされている場合、ファイルシステムは、そのメタデータで、各ユーザ(または)グループが使用しているスペースの量とファイルの数を追跡します。OCFS2はクォータ情報をファイルシステムの内部メタデータとして扱うので、`quotacheck(8)`プログラムを実行する必要はありません。すべての機能は`fsck.ocf2`、およびファイルシステムドライバ自体に組み込まれています。

各ユーザまたはグループに課せられている制限の強制を有効にするには、他のファイルシステムでの場合と同様に、`quotaon(8)`を実行します。

パフォーマンス上の理由で、各クラスタノードはクォータの計算をローカルに行い、この情報を、10秒ごとに共通の中央ストレージに同期するようになっています。この間隔は、`tunefs.ocfs2`と、`usrquota-sync-interval`および`grpquota-sync-interval`オプションで調整することができます。クォータ情報は必ずしも常に正確というわけではないので、複数のクラスタノードを並列に運用している場合、ユーザまたはグループがクォータ制限をいくらか超えることもあります。

19.8 詳細の参照先

OCFS2の詳細については、次のリンクを参照してください。

<https://ocfs2.wiki.kernel.org/> 

OCFS2プロジェクトホームページ。

<http://oss.oracle.com/projects/ocfs2/> 

Oracleサイトにある以前のOCFS2プロジェクトのホームページ

<http://oss.oracle.com/projects/ocfs2/documentation> 

プロジェクトの以前のドキュメントホームページ。

20 GFS2

Global File System 2 (GFS2)は、Linuxコンピュータクラスタ用の共有ディスクファイルシステムです。GFS2により、すべてのノードが同じ共有ブロックストレージに直接同時にアクセスすることができます。GFS2には、非接続運用モードがなく、クライアント役割やサーバ役割也没有ありません。GFS2クラスタのすべてのノードがピアとして機能します。GFS2は、クラスタノードを32台までサポートします。クラスタでGFS2を使用する場合は、ハードウェアが共有ストレージへのアクセスを許可し、ロックマネージャがストレージへのアクセスを制御する必要があります。

SUSEでは、パフォーマンスが主要な要件の1つである場合は、クラスタ環境でOCFS2 over GFS2を使用することを推奨しています。弊社のテストは、このような設定では、GFS2と比較してOCFS2の方がパフォーマンスに優れていることを示しています。

20.1 GFS2パッケージおよび管理ユーティリティ

GFS2を使用するには、`gfs2-utils` および使用しているカーネル用の一致する `gfs2-kmp-*` パッケージが、クラスタの各ノードにインストールされていることを確認します。

`sapMDCsapVirtHostname gfs2-utils` パッケージには、次に示すGFS2ボリュームの管理ユーティリティがあります。構文については、各マニュアルページを参照してください。

表 20.1: GFS2ユーティリティ

GFS2ユーティリティ	説明
<code>fsck.gfs2</code>	ファイルシステムにエラーがないかをチェックし、必要に応じてエラーを修復します。
<code>gfs2_jadd</code>	GFS2ファイルシステムにジャーナルを追加します。
<code>gfs2_grow</code>	GFS2ファイルシステムを拡張します。
<code>mkfs.gfs2</code>	デバイス上にGFS2ファイルシステムを作成します。通常は、共有デバイスまたはパーティションになります。

GFS2ユーティリティ	説明
tunegfs2	UUID、 <u>label</u> 、 <u>lockproto</u> 、 <u>locktable</u> などのGFS2ファイルシステムパラメータを表示および操作できます。

20.2 GFS2サービスとSTONITHリソースの設定

GFS2ボリュームを作成する前に、DLMおよびSTONITHリソースを設定する必要があります。

手順 20.1: STONITHリソースの設定



注記: 必要なSTONITHデバイス

フェンシングデバイスを設定する必要があります。STONITHなしでは、設定内に配置されたメカニズム(external/sbdなど)は失敗します。

1. シェルを起動し、rootまたは同等のものとしてログインします。
2. [手順11.3「SBDデバイスの初期化」](#)で説明されるとおり、SBDパーティションを作成します。
3. `crm configure`を実行します。
4. external/sdbをフェンシングデバイスとして設定し、/dev/sdb2を共有ストレージ上のハートビートとフェンシング専用のパーティションにします。

```
crm(live)configure# primitive sbd_stonith stonith:external/sbd \
    params pcmk_delay_max=30 meta target-role="Started"
```

5. `show`で変更内容をレビューします。
6. すべて正しければ、`commit`で変更を送信し、`exit`でcrmライブ設定を終了します。

DLMに対するリソースグループの設定の詳細については、[手順18.1「DLMのベースグループの設定」](#)を参照してください。

20.3 GFS2ボリュームの作成

20.2項「GFS2サービスとSTONITHリソースの設定」で説明されているように、DLMをクラスターソースとして設定したら、システムがGFS2を使用できるように設定し、GFS2ボリュームを作成します。



注記: アプリケーションファイルとデータファイル用のGFS2ボリューム

一般に、アプリケーションファイルとデータファイルは、異なるGFS2ボリュームに保存することをお勧めします。アプリケーションボリュームとデータボリュームのマウント要件が異なる場合は、必ず、異なるボリュームに保存します。

作業を始める前に、GFS2ボリュームに使用するブロックデバイスを準備します。デバイスは空き領域のままにしてください。

次に、[手順20.2「GFS2ボリュームの作成とフォーマット」](#)で説明されているように、**mkfs.gfs2**で、GFS2ボリュームを作成し、フォーマットします。そのコマンドの重要なパラメータは、[表20.2「重要なGFS2パラメータ」](#)に一覧されています。詳細情報とコマンド構文については、**mkfs.gfs2**のマニュアルページを参照してください。

表 20.2: 重要なGFS2パラメータ

GFS2パラメータ	説明と推奨設定
ロックプロトコル名 (-p)	使用するロックングプロトコルの名前。使用可能なロックングプロトコルはlock_dlm (共有ストレージ用)です。またはローカルファイルシステム(1ノードのみ)としてGFS2を使用している場合は、lock_nolockプロトコルを指定できます。このオプションを指定しない場合、lock_dlmプロトコルであるとみなされます。
ロックテーブル名 (-t)	使用しているロックモジュールに適切はロックテーブルフィールド。 <code>clustername:fsname</code> です。 <code>clustername</code> は、クラスタ設定ファイル/etc/corosync/corosync.confのクラスタ名と一致している必要があります。このクラスタのメンバーだけが、このファイルシステムの使用を許可されます。 <code>fsname</code> は、このGFS2ファイルシステムと作成された他のファイルシステムを区別するために使用される固有のファイルシステム名です(1~16文字)。

GFS2パラメータ	説明と推奨設定
ジャーナル数(-j)	作成するgfs2_mkfs用のジャーナル数。ファイルシステムをマウントするマシンごとに少なくとも1つのジャーナルが必要です。このオプションを指定しない場合、1つのジャーナルが作成されます。

手順 20.2: GFS2ボリュームの作成とフォーマット

クラスタノードの「1つ」だけで、次の手順を実行します。

1. 端末ウィンドウを開いて、rootとしてログインします。
2. クラスタがオンラインであることをコマンド `crm status` で確認します。
3. `mkfs.gfs2` ユーティリティを使用して、ボリュームを作成およびフォーマットします。
このコマンドの構文については、`mkfs.gfs2` マニュアルページを参照してください。
たとえば、最大32台のクラスタノードをサポートする新しいGFS2ファイルシステムを `/dev/sdb1` 上に作成するには、次のコマンドを使用します。

```
root # mkfs.gfs2 -t hacluster:mygfs2 -p lock_dlm -j 32 /dev/sdb1
```

`hacluster` 名は、ファイル `/etc/corosync/corosync.conf` (これはデフォルトです) のエントリ `cluster_name` に関係します。

20.4 GFS2ボリュームのマウント

GFS2ボリュームは、手動でマウントするか、クラスタマネージャでマウントできます(手順 20.4 「クラスタマネージャによるGFS2ボリュームのマウント」を参照)。

手順 20.3: GFS2ボリュームの手動によるマウント

1. 端末ウィンドウを開いて、rootとしてログインします。
2. クラスタがオンラインであることをコマンド `crm status` で確認します。
3. コマンドラインから、`mount` コマンドを使ってボリュームをマウントします。



警告: 手動マウントによるGFS2デバイス

GFS2ファイルシステムをテスト目的で手動マウントした場合、そのファイルシステムは、いったんマウント解除してから、クラスタリソースで使用してください。

手順 20.4: クラスタマネージャによるGFS2ボリュームのマウント

High AvailabilityソフトウェアでGFS2ボリュームをマウントするには、クラスタ内でOCFファイルシステムのリソースを設定します。次の手順では、**crm**シェルを使用してクラスタリソースを設定します。リソースの設定には、Hawk2を使用することもできます。

1. シェルを起動し、rootまたは同等のものとしてログインします。
2. **crm configure**を実行します。
3. GFS2ファイルシステムをクラスタ内のすべてのノードにマウントするように、Pacemakerを設定します。

```
crm(live)configure# primitive gfs2-1 ocf:heartbeat:Filesystem \  
  params device="/dev/sdb1" directory="/mnt/shared" fstype="gfs2" \  
  op monitor interval="20" timeout="40" \  
  op start timeout="60" op stop timeout="60" \  
  meta target-role="Stopped"
```

4. 手順18.1「DLMのベースグループの設定」で作成したdlmプリミティブとgfs2-1プリミティブから構成されるベースグループを作成します。グループをクローンします。

```
crm(live)configure# group g-storage dlm gfs2-1  
  clone cl-storage g-storage \  
  meta interleave="true"
```

ベースグループの内部コロケーションおよび順序付けによって、Pacemakerは、すでに実行しているdlmリソースも持つノード上でgfs2-1リソースのみ起動します。

5. **show**で変更内容をレビューします。
6. すべて正しければ、**commit**で変更を送信し、**exit**でcrmライブ設定を終了します。

21 DRBD

「分散複製ブロックデバイス」(DRBD*)を使用すると、IPネットワーク内の2つの異なるサイトに位置する2つのブロックデバイスのミラーを作成できます。Corosyncと共に使用すると、DRBDは分散高可用性Linuxクラスタをサポートします。この章では、DRBDのインストールとセットアップの方法を示します。

21.1 概念の概要

DRBDは、プライマリデバイス上のデータをセカンダリデバイスに、データの両方のコピーが同一に保たれるような方法で複製します。これは、ネットワーク型のRAID 1と考えてください。DRBDは、データをリアルタイムでミラーリングするので、そのレプリケーションは連続的に起こります。アプリケーションは、実際そのデータがさまざまなディスクに保存されるということを知る必要はありません。

DRBDは、Linuxカーネルモジュールであり、下端のI/Oスケジューラと上端のファイルシステムの間には存在しています(図21.1「Linux内でのDRBDの位置」参照)。DRBDと通信するには、高レベルのコマンド `drbdadm` を使用します。柔軟性を最大にするため、DRBDには、低レベルのツール `drbdsetup` が付いてきます。

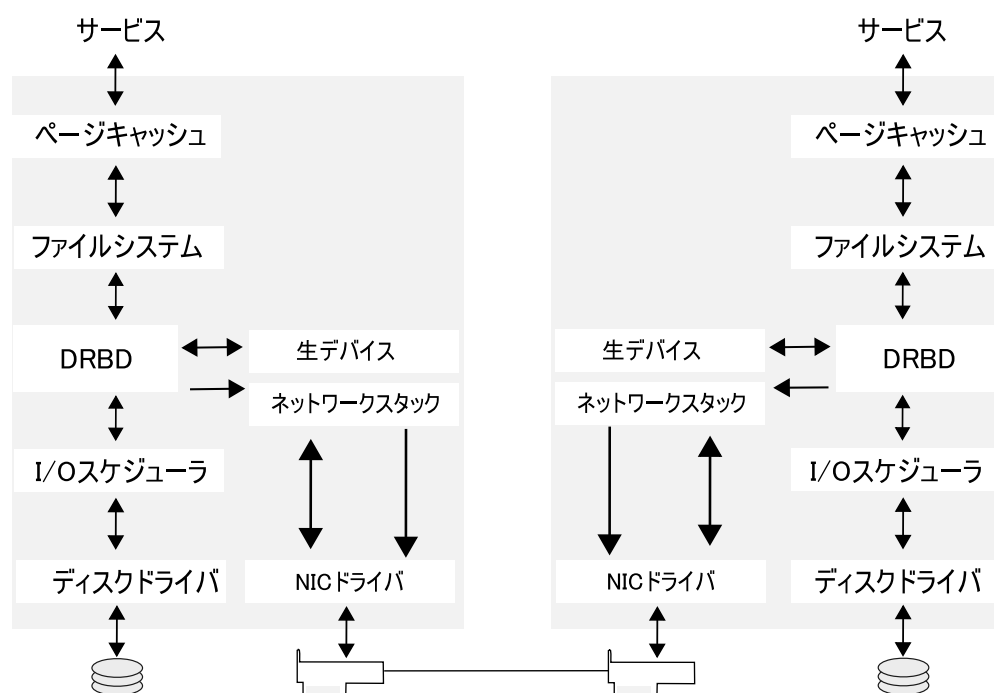


図 21.1: LINUX内でのDRBDの位置

！ 重要: 暗号化されないデータ

ミラー間のデータトラフィックは暗号化されません。データ交換を安全にするには、接続に仮想プライベートネットワーク(VPN)ソリューションを導入する必要があります。

DRBDでは、Linuxでサポートされる任意のブロックデバイスを使用できます。通常は次のデバイスです。

- パーティションまたは完全なハードディスク
- ソフトウェアRAID
- LVM (Logical Volume Manager)
- EVMS (Enterprise Volume Management System)

DRBDは、デフォルトでは、DRBDノード間の通信にTCPポート7788以上を使用します。使用しているポートの通信がファイアウォールで許可されていることを確認してください。

まず、DRBDデバイスを設定してから、その上にファイルシステムを作成する必要があります。ユーザデータに関することはすべて、rawデバイスではなく、`/dev/drbdN`デバイスを介してのみ実行される必要があります。これは、DRBDが、メタデータ用にrawデバイスの最後の部分を使用するからです。rawデバイスを使用すると、データが矛盾する原因となります。

udevの統合により、`/dev/drbd/by-res/RESOURCES`の形式でシンボリックリンクも取得されます。このリンクは、より簡単に使用でき、デバイスのマイナー番号を誤って記憶しないように安全対策が講じられています。

たとえば、rawデバイスのサイズが1024MBの場合、DRBDデバイスは、1023MBしかデータ用に使用できません。70KBは隠され、メタデータ用に予約されています。rawディスクを介した既存のキロバイトへのアクセスは、それがユーザデータ用でないので、すべて失敗します。

21.2 DRBDサービスのインストール

パートI「インストール、セットアップ、およびアップグレード」で説明されているように、High Availability Extensionをネットワーククラスタの両方のSUSE Linux Enterprise Serverマシンにインストールします。High Availability Extensionをインストールすると、DRBDプログラムファイルもインストールされます。

クラスタスタック全体を必要とせず、のみを使用したい場合、パッケージ `drbd`、`drbd-kmp-FLAVOR`、`drbd-utils`、および `yast2-drbd` をインストールします。

`drbdadm`の操作を簡素化するには、Bash補完サポートを使用します。現在のシェルセッションでこのサポートを有効にするには、次のコマンドを挿入します。


```
root # source /etc/bash_completion.d/drbdadm.sh
```

root用に永続的に使用するには、ファイル`/root/.bashrc`を拡張し、前の行を挿入します。

21.3 DRBDサービスの設定



注記: 必要な調整

次の手順では、サーバ名としてaliceとbobを使用し、クラスタリソース名としてr0を使用します。aliceをプライマリノードとして設定し、`/dev/sda1`をストレージとして設定します。必ず、手順を変更して、ご使用のノード名とファイルの名前を使用してください。

次の項では、aliceとbobという2つのノードがあり、それぞれがTCPポート7788を使用するものと想定しています。ファイアウォールでこのポートが開いているようにしてください。

1. システムを準備します。

- a. Linuxノード内のブロックデバイスを準備し、(必要な場合は)パーティション分割しておいてください。
- b. ディスクに、必要のなくなったファイルシステムがすでに含まれている場合は、次のコマンドでファイルシステムの構造を破壊します。

```
root # dd if=/dev/zero of=YOUR_DEVICE count=16 bs=1M
```

破壊する、より多くのファイルシステムがある場合は、DRBDセットアップに含むすべてのデバイス上でこのステップを繰り返します。

- c. クラスタがすでにDRBDを使用している場合は、クラスタを保守モードにします。

```
root # crm configure property maintenance-mode=true
```

クラスタがすでにDRBDを使用している場合に、この手順をスキップすると、ライブ設定の構文エラーによってサービスがシャットダウンされます。

別の方法として、`drbdadm -c FILE`を使用して設定ファイルをテストすることもできます。

2. 次のいずれかの方法を選択してDRBDを設定します。

- 21.3.1項 「手動によるDRBDの設定」
- 21.3.2項 「YaSTによるDRBDの設定」

3. Csync2 (デフォルト)を設定している場合、DRBD設定ファイルは、同期に必要なファイルのリストにすでに含まれています。同期するには、次のコマンドを使用します。

```
root # csync2 -xv /etc/drbd.d/
```

Csync2を設定していない場合(または使用しない場合)には、DRBD設定ファイルを手動で他のノードにコピーしてください。

```
root # scp /etc/drbd.conf bob:/etc/
root # scp /etc/drbd.d/* bob:/etc/drbd.d/
```

4. 初期同期を実行します(21.3.3項 「DRBDリソースの初期化とフォーマット」を参照してください)。
5. クラスタの保守モードフラグをリセットします。

```
root # crm configure property maintenance-mode=false
```

21.3.1 手動によるDRBDの設定



注記: 「自動プロモート」機能の限定サポート

DRBD9機能の「自動プロモート」は、マスタ/スレーブ接続の代わりにクローンとファイルシステムリソースを使用できます。ファイルシステムがマウントされているときにこの機能を使用すると、DRBDは自動的にプライマリモードに変わります。

自動プロモート機能は現在サポートが限定されています。DRBD 9では、SUSEはDRBD-8でもサポートされていた使用例と同じ使用例をサポートしています。3つ以上のノードでのセットアップなど、それを超える使用例はサポートされていません。

DRBDを手動で設定するには、次の手順に従います。

手順 21.1: DRBDの手動設定

DRBDバージョン8.3以降、設定ファイルは、複数のファイルに分割され、/etc/drbd.d/ディレクトリに保存されています。

1. `/etc/drbd.d/global_common.conf` ファイルを開きます。このファイルには、すでにいくつかのグローバルな事前定義値が含まれています。`startup` セクションに移動し、次の3行を挿入します。

```
startup {
    # wfc-timeout degr-wfc-timeout outdated-wfc-timeout
    # wait-after-sb;
    wfc-timeout 100;
    degr-wfc-timeout 120;
}
```

これらのオプションは、ブート時のタイムアウトを減らすために使用します。詳細については、<https://docs.linbit.com/docs/users-guide-9.0/#ch-configure> を参照してください。

2. `/etc/drbd.d/r0.res` ファイルを作成します。状況に応じて行を変更して保存します。

```
resource r0 { ①
    device /dev/drbd0; ②
    disk /dev/sda1; ③
    meta-disk internal; ④
    on alice { ⑤
        address 192.168.1.10:7788; ⑥
        node-id 0; ⑦
    }
    on bob { ⑤
        address 192.168.1.11:7788; ⑥
        node-id 1; ⑦
    }
    disk {
        resync-rate 10M; ⑧
    }
    connection-mesh { ⑨
        hosts alice bob;
    }
}
```

- ① 必要なサービスへのいくつかの関連付けを許可するDRBDリソースの名前。たとえば、`nfs`、`http`、`mysql_0`、`postgres_wal` など。ここでは、一般的な名前 `r0` が使用されています。
- ② DRBD用デバイス名とそのマイナー番号。
先に示した例では、マイナー番号0がDRBDに対して使用されています。udev統合スクリプトは、シンボリックリンク (`/dev/drbd/by-res/nfs/0`) を提供します。または、設定のデバイスノード名を省略し、代わりに次のラインを使用します。
`drbd0 minor 0` (`/dev/` は必要に応じて指定します) または `/dev/drbd0`

- ③ ノード間で複製されるrawデバイス。ただし、この例では、デバイスは両方のノードで「同じ」です。異なるデバイスが必要な場合は、`disk`パラメータを`on`ホストに移動します。
 - ④ `meta-disk`パラメータには、通常、値`internal`が含まれますが、メタデータを保持する明示的なデバイスを指定することもできます。詳細については、<https://docs.linbit.com/docs/users-guide-9.0/#s-metadata> を参照してください。
 - ⑤ `on`セクションでは、この設定文が適用されるホストを記述します。
 - ⑥ それぞれのノードのIPアドレスとポート番号。リソースごとに、通常、`7788`から始まる別個のポートが必要です。1つのDRBDリソースに対して両方のポートが同じである必要があります。
 - ⑦ 複数のノードを設定する際は、ノードIDが必要です。ノードIDは、別々のノードを区別するための固有の負でない整数です。
 - ⑧ 同期レート。このレートは、ディスク帯域幅およびネットワーク帯域幅の3分の1に設定します。これは、再同期を制限するだけで、レプリケーションは制限しません。
 - ⑨ 同一メッシュのすべてのノードを設定します。`hosts`パラメータには、同じDRBDセットアップを共有するすべてのホスト名が含まれます。
3. 環境設定ファイルの構文をチェックします。次のコマンドがエラーを返す場合は、ファイルを検証します。

```
root # drbdadm dump all
```

4. 21.3.3項「DRBDリソースの初期化とフォーマット」に進んでください。

21.3.2 YaSTによるDRBDの設定

YaSTを使用して、DRBDの初期セットアップを開始できます。DRBDセットアップの作成後、生成されたファイルを手動で調整できます。

ただし、設定ファイルを変更した後にYaST DRBDモジュールを使用しないでください。DRBDモジュールでサポートされているのは、限られた一連の基本設定のみです。再度DRBDモジュールを使用すると、変更内容がモジュールに表示されない可能性があります。

YaSTを使ってDRBDを設定するには、次の手順に従います。

手順 21.2: YASTを使用してDRBDを設定

1. YaSTを起動して、設定モジュール高可用性 > DRBDを選択します。すでにDRBDを設定していた場合、YaSTはそのことを警告します。YaSTは設定を変更し、古いDRBD設定ファイルを*.YaSTsaveとして保存します。
2. 起動設定 > ブートのブートフラグは、そのままにしてください(デフォルトではoff)。Pacemakerがこのサービスを管理するので変更しないでください。
3. ファイアウォールが実行中の場合は、ファイアウォールでポートを開くを有効にします。
4. リソース設定エントリに移動します。追加をクリックして、新しいリソースを作成します(see 図21.2「リソースの環境設定」を参照してください)。

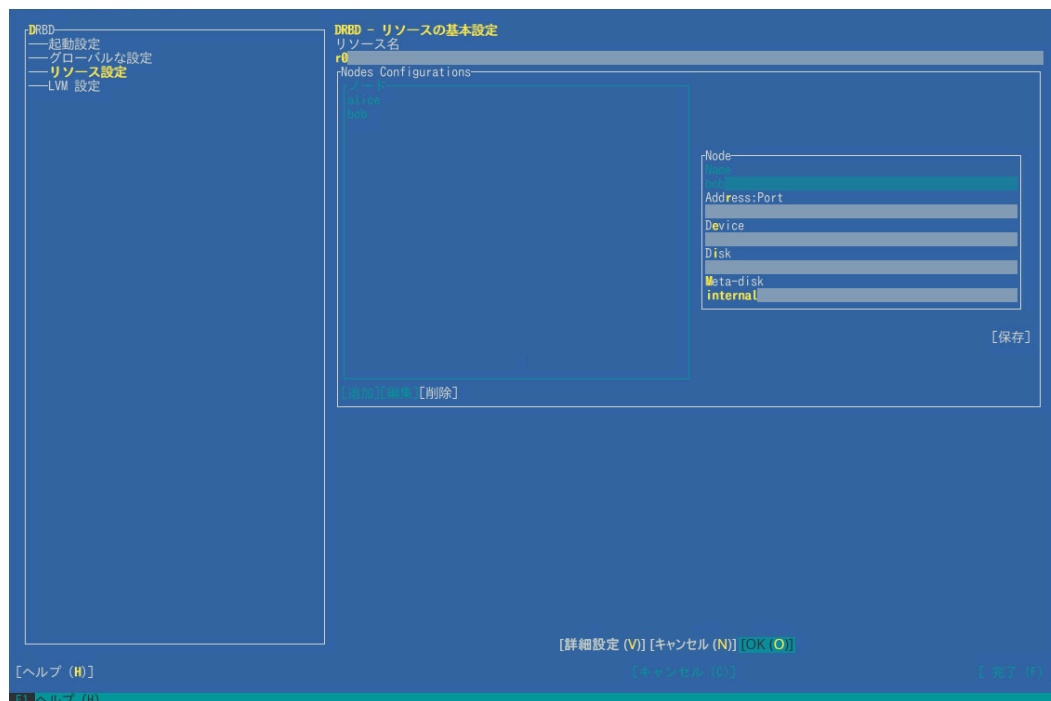


図 21.2: リソースの環境設定

次のパラメータを設定する必要があります。

リソース名

DRBDリソースの名前(必須)。

名前

関連するノードのホスト名。

アドレス:ポート

それぞれのノードのIPアドレスとポート番号(デフォルトは7788)

デバイス

複製されたデータにアクセスするためのブロックデバイスパス。デバイスにマイナー番号が使用されている場合は、関連付けられたブロックデバイスの名前は `/dev/drbdX` になることが普通です。Xはデバイスのマイナー番号です。デバイスにマイナー番号が使用されていない場合は、必ずデバイス名の後に `minor 0` を追記します。

ディスク

両方のノード間で複製されるrawデバイス。LVMを使用する場合、LVMデバイス名を挿入します。

メタディスク

メタディスクは、値 `internal` に設定されるか、またはインデックスで拡張された、drbdで必要なメタデータを保持する明示的なデバイスを指定します。複数のDRBDリソースに実際のデバイスを使用することもできます。たとえば、最初のリソースに対してメタディスクが `/dev/sda6[0]` の場合、`/dev/sda6[1]` を2番目のリソースに使用できます。ただし、このディスク上で各リソースについて少なくとも128MBのスペースが必要です。メタデータの固定サイズによって、複製できる最大データサイズが制限されます。

これらのオプションはすべて、`/usr/share/doc/packages/drbd/drbd.conf` ファイルの例と **drbd.conf (5)** のマニュアルページで説明されています。

5. 保存をクリックします。
6. 追加をクリックして、2番目のDRBDリソースを入力し、保存をクリックして終了します。
7. OKと完了をクリックして、[リソース設定] を閉じます。
8. DRBDでLVMを使用する場合、LVM設定ファイルでオプションをいくつか変更する必要があります(LVM Configuration (LVMの設定)エントリを参照してください)。この変更は、YaST DRBDモジュールを使用して自動的に実行できます。DRBDリソースのローカルホストのディスク名およびデフォルトのフィルタはLVMフィルタで拒否されます。LVMデバイスをスキャンできるのは `/dev/drbd` のみです。

たとえば、`/dev/sda1`をDRBDディスクとして使用している場合、そのデバイス名がLVMフィルタの最初のエントリとして挿入されます。フィルタを手動で変更するには、`Modify LVM Device Filter Automatically` (LVMデバイスフィルタを自動的に変更)チェックボックスをクリックします。

9. 完了をクリックして、変更を保存します。

10. 21.3.3項「DRBDリソースの初期化とフォーマット」に進んでください。

21.3.3 DRBDリソースの初期化とフォーマット

システムを準備してDRBDを設定したら、ディスクの初回の初期化を行います。

1. 「両」ノード(aliceとbob)でメタデータストレージを初期化します。

```
root # drbdadm create-md r0
root # drbdadm up r0
```

2. DRBDリソースの初期再同期を短縮する場合は、次のことを確認します。

- すべてのノード上のDRBDデバイスが同じデータを持つ場合(たとえば、21.3項「DRBDサービスの設定」に示すように`dd`コマンドでファイルシステム構造を破壊することによる)、次のコマンドを使用して初期再同期をスキップします(両ノード)。

```
root # drbdadm new-current-uuid --clear-bitmap r0/0
```

状態は`Secondary/Secondary UpToDate/UpToDate`です

- その後、次のステップに進みます。

3. プライマリノードのaliceから再同期プロセスを開始します。

```
root # drbdadm primary --force r0
```

4. 以下を使用してステータスをチェックします。

```
root # drbdadm status r0
r0 role:Primary
  disk:UpToDate
  bob role:Secondary
  peer-disk:UpToDate
```

5. DRBDデバイスの上にファイルシステムを作成します。たとえば、次のように指定します。

```
root # mkfs.ext3 /dev/drbd0
```

6. ファイルシステムをマウントして使用します。

```
root # mount /dev/drbd0 /mnt/
```

21.4 DRBD 8から DRBD 9への移行

DRBD 8 (SUSE Linux Enterprise High Availability Extension 12 SP1に付属)とDRBD 9 (SUSE Linux Enterprise High Availability Extension 12 SP2に付属)間で、メタデータフォーマットが変更されました。DRBD 9では、以前のメタデータファイルが新しいフォーマットに自動的に変換されません。

12 SP2に移行した後で、DRBDを開始する前に、DRBDメタデータをバージョン9フォーマットに手動で変換します。これを実行するには、`drbdadm create-md`を使用します。設定を変更する必要はありません。



注記: 限定サポート

DRBD 9では、SUSEはDRBD-8でもサポートされていた使用例と同じ使用例をサポートしています。3つ以上のノードでのセットアップなど、それを超える使用例はサポートされていません。

DRBD 9は、バージョン8と互換性を持つように切り替わります。3つ以上のノードの場合、DRBDバージョン9固有のオプションを使用するため、メタデータを再作成する必要があります。

スタックされたDRBDリソースがある場合は、詳細について[21.5項「スタックされたDRBDデバイスの作成」](#)も参照してください。

新しいリソースを再作成せずにデータを保持し、新しいノードを追加することを許可する場合は、次の操作を実行します。

1. 1つのノードをスタンバイモードで設定します。
2. ノードのすべてでDRBDパッケージのすべてを更新します。[21.2項「DRBDサービスのインストール」](#)を参照してください。
3. リソース設定に新しいノード情報を追加します。

- すべての onセクションごとにnode-id。
- connection-mesh セクションには、ホストパラメータのすべてのホスト名が含まれます。 hosts パラメータで制御します。

手順21.1「DRBDの手動設定」のサンプル設定を参照してください。

4. internalをmeta-diskキーとして使用する場合は、DRBDディスクのスペースを拡大します。LVMのようなスペースの拡大をサポートするデバイスを使用します。別の方法として、メタデータ用の外部ディスクに変更して、meta-disk DEVICE;を使用します。
5. 新しい設定に基づいてメタデータを再作成します。

```
root # drbdadm create-md RESOURCE
```

6. スタンバイモードをキャンセルします。

21.5 スタックされたDRBDデバイスの作成

スタックされたDRBDデバイスには少なくとも一方のデバイスがDRBDリソースでもある2つの他のデバイスが含まれます。すなわち、DRBDは既存のDRBDリソースの最上部に追加のノードを追加します(図21.3「リソースのスタッキング」を参照してください)。このようなレプリケーションセットアップは、バックアップおよび障害復旧目的に使用できます。

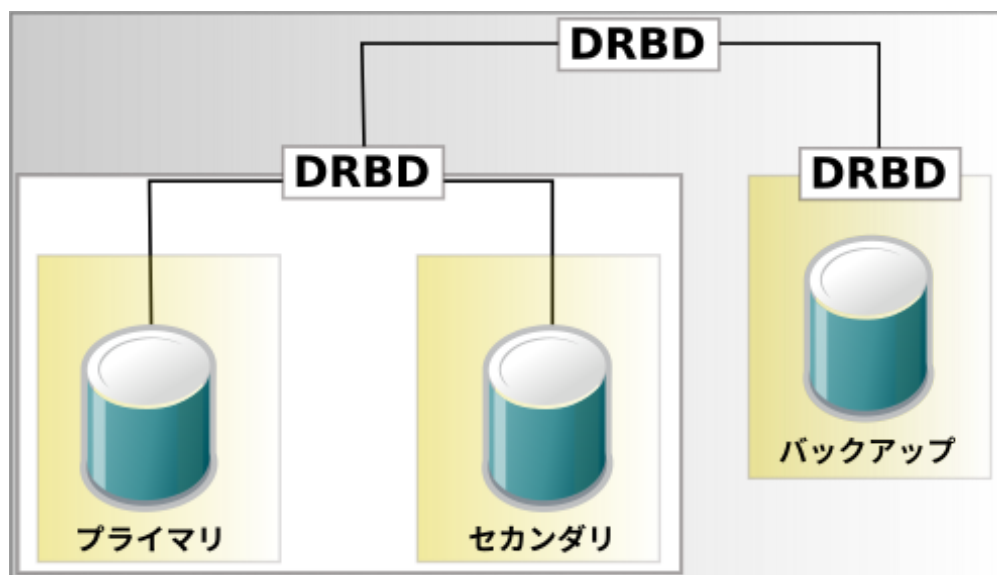


図 21.3: リソースのスタッキング

Three-way レプリケーションは、非同期(DRBDプロトコルA)と同期レプリケーション(DRBDプロトコルC)を使用します。非同期部分はスタックされたリソースに使用され、同期部分はバックアップに使用されます。

ご使用の運用環境ではスタックされたデバイスを使用します。たとえば、最上部にDRBDデバイス/dev/drbd0とスタックされたデバイス/dev/drbd10がある場合、ファイルシステムは/dev/drbd10上に作成されます。詳細については、[例21.1「3ノードのスタックされたDRBDリソースの設定」](#)を参照してください。

例 21.1: 3ノードのスタックされたDRBDリソースの設定

```
# /etc/drbd.d/r0.res
resource r0 {
    protocol C;
    device      /dev/drbd0;
    disk        /dev/sda1;
    meta-disk internal;

    on amsterdam-alice {
        address    192.168.1.1:7900;
    }

    on amsterdam-bob {
        address    192.168.1.2:7900;
    }
}

resource r0-U {
    protocol A;
    device      /dev/drbd10;

    stacked-on-top-of r0 {
        address    192.168.2.1:7910;
    }

    on berlin-charlie {
        disk        /dev/sda10;
        address     192.168.2.2:7910; # Public IP of the backup node
        meta-disk internal;
    }
}
```

21.6 リソースレベルのフェンシングの使用

DRBDレプリケーションリンクが途切れた場合、PacemakerはDRBDリソースを別のノードにプロモートしようとします。Pacemakerが古いデータでサービスを開始しないようにするため、例21.2「クラスタ情報ベース(CIB)を使用したリソースレベルのフェンシングを含むDRBDの設定」に示すようにDRBD設定ファイルでリソースレベルのフェンシングを有効にします。

例 21.2: クラスタ情報ベース(CIB)を使用したリソースレベルのフェンシングを含むDRBDの設定

```
resource RESOURCE {
  net {
    fencing resource-only;
    # ...
  }
  handlers {
    fence-peer "/usr/lib/drbd/crm-fence-peer.9.sh";
    after-resync-target "/usr/lib/drbd/crm-unfence-peer.9.sh";
    # ...
  }
  ...
}
```

DRBDレプリケーションリンクが切断されると、DRBDは以下を実行します。

1. DRBDは`crm-fence-peer.9.sh`スクリプトを呼び出します。
2. スクリプトはクラスタマネージャに連絡します。
3. スクリプトはこのDRBDリソースに関連付けられたPacemakerリソースを判断します。
4. スクリプトは、このDRBDリソースが他のノードにプロモートされないことを確認します。リソースは現在アクティブなノード上にとどまります。
5. レプリケーションリンクがもう一度接続され、DRBDがその同期プロセスを完了すると、制限が解除されます。クラスタマネージャは自由にリソースをプロモートできるようになります。

21.7 DRBDサービスのテスト

インストールと設定のプロシージャが予期どおりの結果となった場合は、DRBD機能の基本的なテストを実行できます。このテストは、DRBDソフトウェアの機能を理解する上でも役立ちます。

1. alice上でDRBDサービスをテストします。

a. 端末コンソールを開き、rootとしてログインします。

b. aliceにマウントポイント(/srv/r0など)を作成します。

```
root # mkdir -p /srv/r0
```

c. drbdデバイスをマウントします。

```
root # mount -o rw /dev/drbd0 /srv/r0
```

d. プライマリノードからファイルを作成します。

```
root # touch /srv/r0/from_alice
```

e. aliceでディスクをマウント解除します。

```
root # umount /srv/r0
```

f. aliceで次のコマンドを入力して、aliceのDRBDサービスを降格します。

```
root # drbdadm secondary r0
```

2. bob上でDRBDサービスをテストします。

a. 端末コンソールを開き、bobでrootとしてログインします。

b. bobで、DRBDサービスをプライマリに昇格します。

```
root # drbdadm primary r0
```

c. bobで、bobがプライマリかどうかチェックします。

```
root # drbdadm status r0
```

d. bobで、/srv/r0などのマウントポイントを作成します。

```
root # mkdir /srv/r0
```

e. bobで、DRBDデバイスをマウントします。

```
root # mount -o rw /dev/drbd0 /srv/r0
```

f. aliceで作成したファイルが存在していることを確認します。

```
root # ls /srv/r0/from_alice
```

/srv/r0/from_aliceファイルが一覧に表示されている必要があります。

3. サービスが両方のノードで稼動していれば、DRBDの設定は完了です。

4. 再度、aliceをプライマリとして設定します。

a. bobで次のコマンドを入力して、bobのディスクをマウント解除します。

```
root # umount /srv/r0
```

b. bobで次のコマンドを入力して、bobのDRBDサービスを降格します。

```
root # drbdadm secondary r0
```

c. aliceで、DRBDサービスをプライマリに昇格します。

```
root # drbdadm primary r0
```

d. aliceで、aliceがプライマリかどうかチェックします。

```
root # drbdadm status r0
```

5. サービスを自動的に起動させ、サーバに問題が発生した場合はフェールオーバーさせるためには、Pacemaker/CorosyncでDRBDを高可用性サービスとして設定できます。SUSE Linux Enterprise 15 SP3のインストールと設定については、[パートII「設定および管理」](#)を参照してください。

21.8 DRBDデバイスの監視

DRBDには、リアルタイム監視用の**drbdmon**ユーティリティが付属しています。このユーティリティにより、設定されているすべてのリソースの状態と、それらのリソースで発生している問題が示されます。

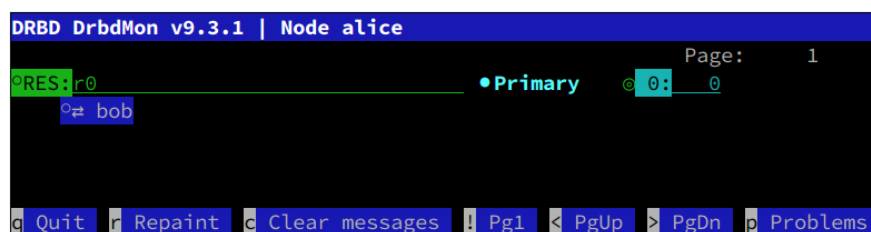


図 21.4: drbdmonが示す正常な接続

問題がある場合、**drbdadm**に次のエラーメッセージが表示されます。

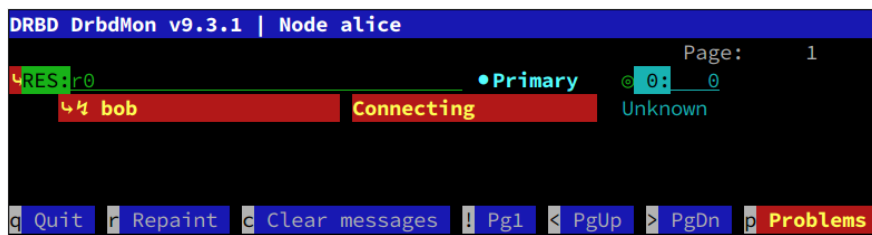


図 21.5: drbdmonが示す異常な接続

21.9 DRBDのチューニング

DRBDをチューニングするには、いくつかの方法があります。

1. メタデータ用には外部ディスクを使用します。これは便利ですが、保守作業は煩雑になります。
2. `sysctl`を介して受信および送信バッファ設定を変更することで、ネットワーク接続を調整します。
3. DRBD設定で`max-buffers`、`max-epoch-size`、またはその両方を変更します。
4. IOパターンに応じて、`al-extents`の値を増やします。
5. ハードウェアRAIDコントローラとBBU (「バッテリバックアップユニット」)を併用する場合、`no-disk-flushes`、`no-disk-barrier`、および`no-md-flushes`の設定が有効な場合があります。
6. ワークロードに従って読み込みバランスを有効にします。詳しくは「<https://www.linbit.com/en/read-balancing/>」を参照してください。

21.10 DRBDのトラブルシューティング

DRBDセットアップには、多数のコンポーネントが使用され、別のソースから問題が発生することがあります。以降のセクションでは、一般的なシナリオをいくつか示し、さまざまなソリューションを推奨します。

21.10.1 設定

初期のDRBDセットアップが予期どおりに機能しない場合は、おそらく、環境設定に問題があります。

環境設定の情報を取得するには:

1. 端末コンソールを開き、rootとしてログインします。
2. drbdadmに-dオプションを指定して、環境設定ファイルをテストします。次のコマンドを入力します。

```
root # drbdadm -d adjust r0
```

adjustオプションのドライ実行では、drbdadmは、DRBDリソースの実際の設定を使用中のDRBD環境設定ファイルと比較しますが、コールは実行しません。出力をレビューして、エラーのソースおよび原因を確認してください。

3. /etc/drbd.d/*ファイルとdrbd.confファイルにエラーがある場合は、そのエラーを修正してから続行してください。
4. パーティションと設定が正しい場合は、drbdadmを-dオプションなしで、再度実行します。

```
root # drbdadm adjust r0
```

このコマンドは、環境設定ファイルをDRBDリソースに適用します。

21.10.2 ホスト名

DRBDの場合、ホスト名の大文字と小文字が区別され(Node0はnode0とは異なるホストであるとみなされる)、カーネルに格納されているホスト名と比較されます(uname -n出力を参照)。

複数のネットワークデバイスがあり、専用ネットワークデバイスを使用したい場合、おそらく、ホスト名は使用されたIPアドレスに解決されません。この場合は、パラメータdisable-ip-verificationを使用します。

21.10.3 TCPポート7788

システムがピアに接続できない場合は、ローカルファイアウォールに問題のある可能性があります。DRBDは、デフォルトでは、TCPポート7788を使用して、もう一方のノードにアクセスします。このポートを両方のノードからアクセスできるかどうか確認してください。

21.10.4 DRBDデバイスが再起動後に破損した

DRBDサブシステムが実際のどのデバイスが最新データを保持しているか認識していない場合、スプリットブレイン受験に変更されます。この場合、それぞれのDRBDサブシステムがセカンダリとして起動され、互いに接続しません。この場合、ログ記録データに、次のメッセージが出力されることがあります。

```
Split-Brain detected, dropping connection!
```

この状況を解決するには、廃棄するデータを持つノードで、次のコマンドを入力します。

```
root # drbdadm secondary r0
```

状態が`WFconnection`の場合、最初に切断します。

```
root # drbdadm disconnect r0
```

最新のデータを持つノードで、次のコマンドを入力します。

```
root # drbdadm connect --discard-my-data r0
```

このコマンドは、あるノードのデータをピアのデータで上書きすることによって問題を解決するため、両方のノードで一貫したビューが得られます。

21.11 詳細の参照先

DRBDについては、次のオープンソースリソースを利用できます。

- プロジェクトホームページ<http://www.drbd.org>。
- 項目「Highly Available NFS Storage with DRBD and Pacemaker」を参照してください。
- http://clusterlabs.org/wiki/DRBD_HowTo_1.0 (Linux Pacemaker Cluster Stack Projectによる)。
- このディストリビューションで利用できるDRBDのマニュアルページは、[`drbd\(8\)`](#)、[`rbdmeta\(8\)`](#)、[`drbdsetup\(8\)`](#)、[`drbdadm\(8\)`](#)、[`drbd.conf\(5\)`](#)です。
- コメント付きのDRBD設定例が、[`/usr/share/doc/packages/drbd-utils/drbd.conf.example`](#)にあります。
- さらに、クラスタ間のストレージ管理を容易にするために、「DRBD-Manager」(<https://www.linbit.com/en/drbd-manager/>)に関する最新の通知を参照してください。

22 Cluster Logical Volume Manager (Cluster LVM)

クラスタ上の共有ストレージを管理する場合、ストレージサブシステムに行った変更を各ノードに伝える必要があります。Logical Volume Manager 2 (LVM2)はローカルストレージの管理に多用されており、クラスタ全体のボリュームグループのトランスペアレントな管理をサポートするために拡張されています。複数のホスト間で共有されるボリュームグループは、ローカルストレージと同じコマンドを使用して管理できます。

22.1 概念の概要

Cluster LVMは、さまざまなツールと連携します。

Distributed Lock Manager (DLM)

クラスタ全体のロックにより、複数のホスト間の共有リソースへのアクセスを調整します。

Logical Volume Manager (LVM2)

LVM2はディスクスペースの仮想プールを提供し、1つの論理ボリュームを複数のディスクにわたって柔軟に分散できます。

Cluster Logical Volume Manager (Cluster LVM)

Cluster LVMという用語は、LVM2がクラスタ環境で使用されていることを示しています。このため、共有ストレージ上のLVM2メタデータを保護するには、ある程度の設定調整が必要になります。SUSE Linux Enterprise 15以降、クラスタ拡張では、よく知られているclvmdの代わりに、lvmlockdを使用します。lvmlockdの詳細については、[lvmlockd](#)コマンドのマニュアルページ([man 8 lvmlockd](#))を参照してください。

ボリュームグループと論理ボリューム

ボリュームグループ(VG)と論理ボリューム(LV)は、LVM2の基本的な概念です。ボリュームグループは、複数の物理ディスクのストレージプールです。論理ボリュームはボリュームグループに属し、ファイルシステムを作成できるエラスティックボリュームと見なすことができます。クラスタ環境には、共有VGという概念があります。共有VGは共有ストレージで構成され、複数のホストで同時に使用することができます。

22.2 クラスタLVMの設定

以下の要件が満たされていることを確認します。

- 共有ストレージデバイス(Fibre Channel、FCoE、SCSI、iSCSI SAN、DRBD*などで提供されているデバイス)が使用できること
- パッケージ `lvm2` と `lvm2-lockd` がインストールされていることを確認します。
- SUSE Linux Enterprise 15以降では、LVM2クラスタ拡張として `clvmd` ではなく `lvmlockd` を使用しています。 `clvmd` デーモンが実行されていないことを確認します。実行されていると、 `lvmlockd` の起動に失敗します。

22.2.1 クラスタリソースの作成

クラスタ内で共有VGを設定するには、1つのノードで次の基本手順を実行します。

- DLMリソースの作成
- `lvmlockd` リソースの作成
- 共有VGおよびLVの作成
- LVM-activateリソースの作成

手順 22.1: DLMリソースの作成

1. シェルを起動して、 `root` としてログインします。
2. クラスタリソースの現在の設定を確認します。

```
root # crm configure show
```

3. すでにDLMリソース(および対応するベースグループおよびベースクローン)を設定済みである場合、[手順22.2「lvmlockdリソースの作成」](#)で続きます。
そうでない場合は、[手順18.1「DLMのベースグループの設定」](#)で説明されているように、DLMリソース、および対応するベースグループとベースクローンを設定します。

手順 22.2: LVMLOCKDリソースの作成

1. シェルを起動して、 `root` としてログインします。
2. 次のコマンドを実行して、このリソースの使用状況を確認します。

```
root # crm configure ra info lvmlockd
```

3. lvmlockdリソースを次のように設定します。

```
root # crm configure primitive lvmlockd ocf:heartbeat:lvmlockd \  
  op start timeout="90" \  
  op stop timeout="100" \  
  op monitor interval="30" timeout="90"
```

4. lvmlockdリソースがすべてのノードで起動されるようにするには、[手順22.1「DLMリソースの作成」](#)で作成したストレージ用の基本グループにプリミティブリソースを追加します。

```
root # crm configure modgroup g-storage add lvmlockd
```

5. 変更内容をレビューします。

```
root # crm configure show
```

6. リソースが正常に実行されているかどうかを確認します。

```
root # crm status full
```

手順 22.3: 共有VGおよびLVの作成

1. シェルを起動して、rootとしてログインします。
2. すでに2つの共有ディスクがあると仮定し、それらに対して共有VGを作成します。

```
root # vgcreate --shared vg1 /dev/sda /dev/sdb
```

3. LVを作成します。最初は有効にしないでください。

```
root # lvcreate -an -L10G -n lv1 vg1
```

手順 22.4: LVM-ACTIVATEリソースの作成

1. シェルを起動して、rootとしてログインします。
2. 次のコマンドを実行して、このリソースの使用状況を確認します。

```
root # crm configure ra info LVM-activate
```

このリソースは、VGのアクティブ化を管理します。共有VGにおけるLVアクティブ化には、排他モードと共有モードという2つの異なるモードがあります。デフォルト設定は排他モードです。このモードは通常、ext4などのローカルファイルシステムでLVを使用するときに使用します。共有モードは、OCFS2などのクラスタファイルシステムでのみ使用します。

3. VGのアクティブ化を管理するようにリソースを設定します。使用しているシナリオに応じて、次のいずれかのオプションを選択します。

- ローカルファイルシステムの使用に対して排他アクティブ化モードを使用します。

```
root # crm configure primitive vg1 ocf:heartbeat:LVM-activate \  
  params vgname=vg1 vg_access_mode=lvmlockd \  
  op start timeout=90s interval=0 \  
  op stop timeout=90s interval=0 \  
  op monitor interval=30s timeout=90s
```

- OCFS2に対して共有アクティブ化モードを使用し、クローニングされたg-storageグループに追加します。

```
root # crm configure primitive vg1 ocf:heartbeat:LVM-activate \  
  params vgname=vg1 vg_access_mode=lvmlockd activation_mode=shared \  
  op start timeout=90s interval=0 \  
  op stop timeout=90s interval=0 \  
  op monitor interval=30s timeout=90s  
root # crm configure modgroup g-storage add vg1
```

4. リソースが正常に実行されているかどうかを確認します。

```
root # crm status full
```

22.2.2 シナリオ: SAN上でiSCSIを使用するCluster LVM

次のシナリオでは、iSCSIターゲットをいくつかのクライアントにエクスポートする2つのSANボックスを使用します。一般的なアイデアが、[図22.1「Cluster LVMを使用した共有ディスクのセットアップ」](#)で説明されています。

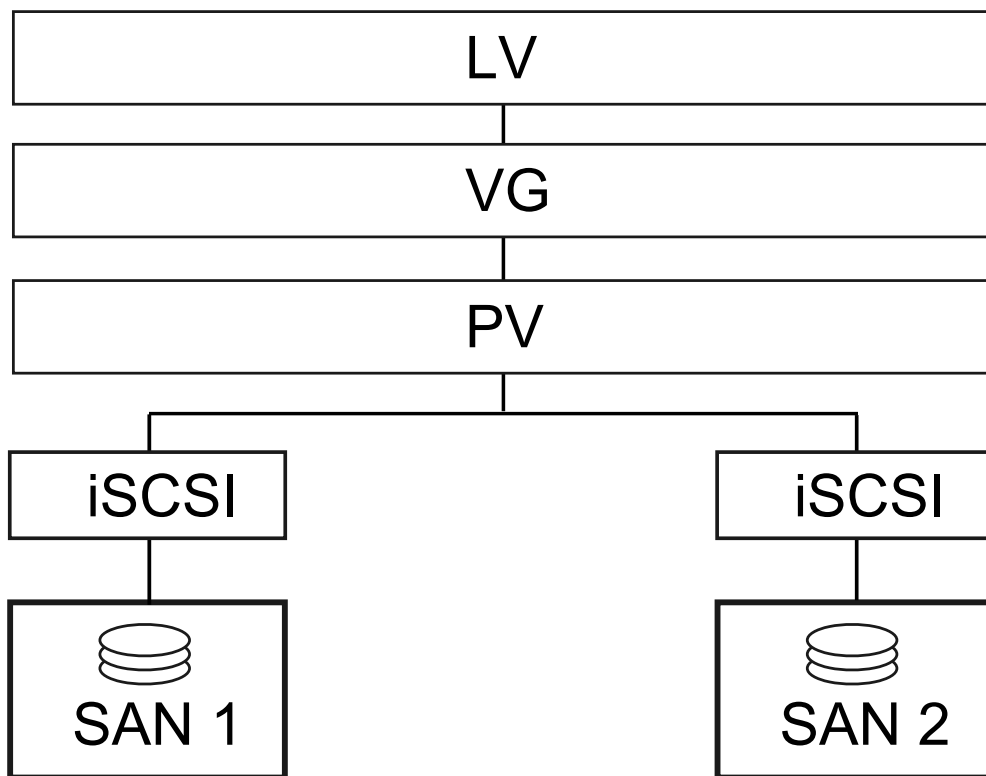


図 22.1: CLUSTER LVMを使用した共有ディスクのセットアップ



警告: データ損失

以降の手順を実行すると、ディスク上のデータはすべて破壊されます。

まず、1つのSANボックスだけ設定します。各SANボックスは、そのiSCSIターゲットをエクスポートする必要があります。以下に手順を示します。

手順 22.5: iSCSIターゲット(SAN上)を設定する

1. YaSTを実行し、ネットワークサービス > iSCSI LIO Target (iSCSI LIOターゲット)の順にクリックしてiSCSIサーバモジュールを起動します。
2. コンピュータがブートするたびにiSCSIターゲットを起動したい場合は、ブート時を選択し、そうでない場合は、手動を選択します。
3. ファイアウォールが実行中の場合は、ファイアウォールでポートを開くを有効にします。
4. グローバルタブに切り替えます。認証が必要な場合は、受信または送信(あるいはその両方)の認証を有効にします。この例では、認証なしを選択します。
5. 新しいiSCSIターゲットを追加します。

- a. ターゲットタブに切り替えます。
- b. 追加をクリックします。
- c. ターゲットの名前を入力します。名前は、次のようにフォーマットされます。

```
iqn.DATE.DOMAIN
```

フォーマットに関する詳細は、セクション3.2.6.3.1のタイプ「iqn」(iSCSI修飾名) (<http://www.ietf.org/rfc/rfc3720.txt>)を参照してください。

- d. より説明的な名前にしたい場合は、さまざまなターゲットで一意であれば、識別子を変更できます。
 - e. 追加をクリックします。
 - f. パスにデバイス名を入力し、Scsiidを使用します。
 - g. 次へを2回クリックします。
6. 警告ボックスではいを選択して確認します。
 7. 環境設定ファイル/etc/iscsi/iscsid.confを開き、パラメータnode.startupをautomaticに変更します。

次の手順に従って、iSCSIイニシエータを設定します。

手順 22.6: iSCSIイニシエータの環境設定

1. YaSTを実行し、ネットワークサービス > iSCSIイニシエータの順にクリックします。
2. コンピュータがブートするたびに、iSCSIイニシエータを起動したい場合は、ブート時を選択し、そうでない場合は、手動を選択します。
3. 検出タブに切り替え、検出ボタンをクリックします。
4. IPアドレスとiSCSIターゲットのポートを追加します(手順22.5「iSCSIターゲット(SAN上)を設定する」参照)。通常は、ポートを既定のままにし、デフォルト値を使用できます。
5. 認証を使用する場合は、受信および送信用のユーザ名およびパスワードを挿入します。そうでない場合は、認証なしを選択します。
6. 次へを選択します。検出された接続が一覧されます。
7. 完了をクリックして続行します。

8. シェルを開いて、rootとしてログインします。

9. iSCSIイニシエータが正常に起動しているかどうかテストします。

```
root # iscsiadm -m discovery -t st -p 192.168.3.100
192.168.3.100:3260,1 iqn.2010-03.de.jupiter:san1
```

10. セッションを確立します。

```
root # iscsiadm -m node -l -p 192.168.3.100 -T iqn.2010-03.de.jupiter:san1
Logging in to [iface: default, target: iqn.2010-03.de.jupiter:san1, portal:
192.168.3.100,3260]
Login to [iface: default, target: iqn.2010-03.de.jupiter:san1, portal:
192.168.3.100,3260]: successful
```

ls SCSIでデバイス名を表示します。

```
...
[4:0:0:2]    disk    IET      ...    0    /dev/sdd
[5:0:0:1]    disk    IET      ...    0    /dev/sde
```

3番目の列にIETを含むエントリを捜します。この場合、該当するデバイスは、/dev/sddと/dev/sdeです。

手順 22.7: 共有ボリュームグループの作成

1. 手順22.6「iSCSIイニシエータの環境設定」のiSCSIイニシエータを実行したノードの1つで、rootシェルを開きます。

2. ディスク/dev/sddおよび/dev/sdeに共有ボリュームグループを作成します。

```
root # vgcreate --shared testvg /dev/sdd /dev/sde
```

3. 必要に応じて、論理ボリュームを作成します。

```
root # lvcreate --name lv1 --size 500M testvg
```

4. ボリュームグループをvgdisplayで確認します。

```
--- Volume group ---
VG Name                testvg
System ID
Format                 lvm2
Metadata Areas         2
Metadata Sequence No   1
VG Access               read/write
VG Status               resizable
MAX LV                 0
```

```

Cur LV          0
Open LV          0
Max PV           0
Cur PV          2
Act PV           2
VG Size          1016,00 MB
PE Size          4,00 MB
Total PE         254
Alloc PE / Size  0 / 0
Free PE / Size   254 / 1016,00 MB
VG UUID          UCyWw8-2jqV-enuT-KH4d-NXQI-JhH3-J24anD

```

5. コマンド **vgs** を使用してボリュームグループの共有状態を確認します。

```

root # vgs
VG      #PV #LV #SN Attr   VSize   VFree
vgshared 1   1   0 wz--ns 1016.00m 1016.00m

```

Attr列には、ボリューム属性が表示されます。この例では、ボリュームグループは書き込み可能(w)、サイズ変更可能(z)であり、割り当てポリシーは通常(n)であり、共有リソース(s)です。詳細については、**vgs**のマニュアルページを参照してください。

ボリュームを作成してリソースを起動したら、`/dev/testvg`の下に、新しいデバイス名(`/dev/testvg/lv1`など)を作成する必要があります。これは、LVがアクティブ化され使用できることを示します。

22.2.3 シナリオ: DRBDを使用したCluster LVM

市、国、または大陸の各所にデータセンターが分散している場合は、次のシナリオを使用できます。

手順 22.8: DRBDでクラスタ対応ボリュームグループを作成する

1. プライマリ/プライマリDRBDリソースを作成する

- a. まず、手順21.1「DRBDの手動設定」の説明に従って、DRBDデバイスをプライマリ/セカンダリとしてセットアップします。ディスクの状態が両方のノードでup-to-dateであることを確認します。**drbdadm status**を使用してこれをチェックします。
- b. 次のオプションを環境設定ファイル(通常は、`/etc/drbd.d/r0.res`)に追加します。

```

resource r0 {
    net {

```



```
    allow-two-primaries;
  }
  ...
}
```

- c. 変更した設定ファイルをもう一方のノードにコピーします。たとえば、次のように指定します。

```
root # scp /etc/drbd.d/r0.res venus:/etc/drbd.d/
```

- d. 「両方」のノードで、次のコマンドを実行します。

```
root # drbdadm disconnect r0
root # drbdadm connect r0
root # drbdadm primary r0
```

- e. ノードのステータスをチェックします。

```
root # drbdadm status r0
```

2. `lvmlckd`リソースをペースメーカーの環境設定でクローンとして保存し、DLMクローンリソースに依存させます。詳細については、[手順22.1「DLMリソースの作成」](#)を参照してください。次に進む前に、クラスタでこれらのリソースが正しく機動していることを確認してください。`crm status`またはWebインタフェースを使用して、実行中のサービスを確認します。

3. `pvccreate`コマンドで、LVM用に物理ボリュームを準備します。たとえば、`/dev/drbd_r0`デバイスでは、コマンドは次のようになります。

```
root # pvccreate /dev/drbd_r0
```

4. 共有ボリュームグループを作成します。

```
root # vgcreate --shared testvg /dev/drbd_r0
```

5. 必要に応じて、論理ボリュームを作成します。おそらく、論理ボリュームのサイズの変更が必要になります。たとえば、次のコマンドで、4GBの論理ボリュームを作成します。

```
root # lvcreate --name lv1 -L 4G testvg
```

6. VG内の論理ボリュームは、ファイルシステムのマウントまたはraw用として使用できるようになりました。論理ボリュームを使用しているサービスにコロケーションのための正しい依存性があることを確認し、VGをアクティブ化したら論理ボリュームの順序付けを行います。

このような設定手順を終了すると、LVM2の環境設定は他のスタンドアロンワークステーションと同様に行えます。

22.3 有効なLVM2デバイスの明示的な設定

複数のデバイスが同じ物理ボリュームの署名を共有していると思われる場合(マルチパスデバイスやDRBDなどのように)、LVM2がPVを走査するデバイスを明示的に設定しておくことをお勧めします。

たとえば、コマンド **vgcreate** で、ミラーリングされたブロックデバイスを使用する代わりに物理デバイスを使用すると、DRBDで混乱が生じます。これにより、DRBDがスプリットブレイン状態になる可能性があります。

LVM2用の単一のデバイスを非アクティブ化するには、次の手順に従います。

1. ファイル `/etc/lvm/lvm.conf` を編集し、`filter` から始まる行を検索します。
2. そこに記載されているパターンは正規表現として処理されます。冒頭の「a」は走査にデバイスパターンを受け入れることを、冒頭の「r」はそのデバイスパターンのデバイスを拒否することを意味します。
3. `/dev/sdb1` という名前のデバイスを削除するには、次の表現をフィルタルールに追加します。

```
"r|^/dev/sdb1$|"
```

完全なフィルタ行は次のようになります。

```
filter = [ "r|^/dev/sdb1$|", "r|/dev/.*/by-path/.*/", "r|/dev/.*/by-id/.*/",  
          "a/.*/" ]
```

DRBDとMPIOデバイスは受け入れ、その他のすべてのデバイスは拒否するフィルタ行は次のようになります。

```
filter = [ "a|/dev/drbd.*|", "a|/dev/.*/by-id/dm-uuid-mpath-.*/", "r/.*/" ]
```

4. 環境設定ファイルを書き込み、すべてのクラスターノードにコピーします。

22.4 Mirror LVからCluster MDへのオンラインマイグレーション

SUSE Linux Enterprise High Availability Extension 15以降、Cluster LVMでの`cmirrord`の使用は非推奨になりました。クラスタ内のミラー論理ボリュームはCluster MDに移行することを強く推奨します。Cluster MDはクラスタマルチデバイスの略称で、クラスタ用のソフトウェアベースのRAIDストレージソリューションです。

22.4.1 マイグレーション前のサンプルセットアップ

次のサンプルセットアップが存在するとします。

- aliceおよびbobというノードからなる2ノードクラスタ
- cluster-vg2というボリュームグループから作成された、test-lvという名前のミラー論理ボリューム
- /dev/vdbおよび/dev/vdcというディスクからなるボリュームグループcluster-vg2

```
root # lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                                253:0    0   40G  0 disk
├─vda1                             253:1    0    4G  0 part [SWAP]
└─vda2                             253:2    0   36G  0 part /
vdb                                253:16   0   20G  0 disk
├─cluster--vg2-test--lv_mlog_mimage_0 254:0    0    4M  0 lvm
│   └─cluster--vg2-test--lv_mlog        254:2    0    4M  0 lvm
│       └─cluster--vg2-test--lv        254:5    0   12G  0 lvm
└─cluster--vg2-test--lv_mimage_0      254:3    0   12G  0 lvm
    └─cluster--vg2-test--lv            254:5    0   12G  0 lvm
vdc                                253:32   0   20G  0 disk
├─cluster--vg2-test--lv_mlog_mimage_1 254:1    0    4M  0 lvm
│   └─cluster--vg2-test--lv_mlog        254:2    0    4M  0 lvm
│       └─cluster--vg2-test--lv        254:5    0   12G  0 lvm
└─cluster--vg2-test--lv_mimage_1      254:4    0   12G  0 lvm
    └─cluster--vg2-test--lv            254:5    0   12G  0 lvm
```

！ 重要: マイグレーションにおける障害の発生を避ける

マイグレーション手順を実施する前に、論理ボリュームおよび物理ボリュームで使用可能な容量と、どの程度使用されているかを確認してください。論理ボリュームの容量がすべて使い果たされている場合、マイグレーションは失敗し、ターゲットボリュームに関する `insufficient free space` エラーが発生します。こうしたマイグレーション障害に対応するための回避策は、ミラーログのオプションに応じて異なります。

- **ミラーログ自体が(`mirrored`オプションを使用して)ミラーリングされ、ミラーログの格納先と同じデバイスに割り当てられている場合。** (たとえば、SUSE Linux Enterprise High Availability Extension 11または12で、各バージョンの『管理ガイド (<https://documentation.suse.com/sle-ha/12-SP5/html/SLE-HA-all/cha-ha-clvm.html#sec-ha-clvm-config-cmirrord>) 』の説明に従って、`cmirrord`を設定した論理ボリュームを作成したケース。)

デフォルトでは、`mdadm`が実行されることにより、デバイスとアレイデータのマイグレーションがそれぞれ開始される前に、ある程度の空き領域が確保されます。マイグレーション中には、この未使用のパディング領域をチェックして、その容量を削除することも可能です。その際には、**ステップ 1.d**と次の段落で説明する `data-offset` オプションを利用できます。

`data-offset`の実行時には、Cluster MDがそのメタデータを書き込めるよう、デバイス上に十分な余裕を持たせる必要があります。一方で、オフセット自体の容量は抑える必要があります。デバイスの残りの領域で、移行する物理ボリュームのエクス Tent をすべて格納しなければならないからです。使用可能なボリューム領域は、ボリューム全体からミラーログを差し引いた残りの領域なので、オフセットのサイズはミラーログよりも小さくする必要があります。

`data-offset`を128kBに設定することを推奨します。オフセット値を指定しない場合、デフォルト値の1kB (1024バイト)が使用されます。

- **ミラーログが(`disk`オプションを使用して)別のデバイスに記録されている場合、あるいは(`core`オプションを使用して)メモリ上に格納されている場合:** マイグレーションを始める前に、物理ボリュームの容量を増やすか、(物理ボリュームの空き領域を確保するために)論理ボリュームのサイズを減らします。

22.4.2 Mirror LVをCluster MDにマイグレートする

以下の手順は、22.4.1項「マイグレーション前のサンプルセットアップ」に基づいています。セットアップ環境に応じて手順を変更し、LV、VG、ディスク、Cluster MDデバイスの名前はご使用のリソースに置き換えてください。

このマイグレーションではダウンタイムは発生しません。そのため、マイグレーション中でもファイルシステムをマウントすることができます。

1. aliceノードで、次の手順を実行します。

a. ミラー論理ボリュームtest-lvをリニア論理ボリュームに変換します。

```
root # lvconvert -m0 cluster-vg2/test-lv /dev/vdc
```

b. 物理ボリューム/dev/vdcをボリュームグループcluster-vg2から削除します。

```
root # vgreduce cluster-vg2 /dev/vdc
```

c. この物理ボリュームをLVMから削除します。

```
root # pvremove /dev/vdc
```

lsblkを実行すると、次のように出力されます。

NAME			MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
vda	253:0	0	40G	0	disk			
└─vda1	253:1	0	4G	0	part		[SWAP]	
└─vda2	253:2	0	36G	0	part		/	
vdb	253:16	0	20G	0	disk			
└─cluster--vg2-test--lv	254:5	0	12G	0	lvm			
vdc	253:32	0	20G	0	disk			

d. ディスク/dev/vdcを持つCluster MDデバイス/dev/md0を作成します。

```
root # mdadm --create /dev/md0 --bitmap=clustered \
--metadata=1.2 --raid-devices=1 --force --level=mirror \
/dev/vdc --data-offset=128
```

data-offsetオプションを使用する理由については、[重要: マイグレーションにおける障害の発生を避ける](#)を参照してください。

2. bobノードで、次のMDデバイスをアセンブルします。

```
root # mdadm --assemble md0 /dev/vdc
```

2ノードより多いクラスタでは、クラスタ内の残りのノードについても、この手順を実行してください。

3. aliceノードに戻って、次の手順を実行します。

- a. LVM用の物理ボリュームとして、MDデバイス/dev/md0を初期化します。

```
root # pvcreate /dev/md0
```

- b. MDデバイス/dev/md0をボリュームグループcluster-vg2に追加します。

```
root # vgextend cluster-vg2 /dev/md0
```

- c. ディスク/dev/vdbのデータをデバイス/dev/md0に移動します。

```
root # pvmove /dev/vdb /dev/md0
```

- d. 物理ボリューム/dev/vdbをボリュームgroup cluster-vg2から削除します。

```
root # vgreduce cluster-vg2 /dev/vdb
```

- e. LVMがデバイスを物理ボリュームとして認識しなくなるように、このデバイスのラベルを削除します。

```
root # pvremove /dev/vdb
```

- f. /dev/vdbをMDデバイス/dev/md0に追加します。

```
root # mdadm --grow /dev/md0 --raid-devices=2 --add /dev/vdb
```

22.4.3 マイグレーション後のサンプルセットアップ

lsblkを実行すると、次のように出力されます。

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPPOINT
vda	253:0	0	40G	0	disk	
├vda1	253:1	0	4G	0	part	[SWAP]
└vda2	253:2	0	36G	0	part	/
vdb	253:16	0	20G	0	disk	
└md0	9:0	0	20G	0	raid1	
└cluster--vg2-test--lv	254:5	0	12G	0	lvm	
vdc	253:32	0	20G	0	disk	
└md0	9:0	0	20G	0	raid1	
└cluster--vg2-test--lv	254:5	0	12G	0	lvm	

22.5 詳細の参照先

lvmlockdの詳細については、[lvmlockd](#)コマンドのマニュアルページ([man 8 lvmlockd](#))を参照してください。

詳細な情報は、<http://www.clusterlabs.org/wiki/Help:Contents> にあるPacemakerメーリングリストから取得できます。

23 クラスタマルチデバイス(Cluster MD)

クラスタマルチデバイス(Cluster MD)は、クラスタ用のソフトウェアベースのRAIDストレージソリューションです。現在、Cluster MDでは、クラスタにRAID1ミラーリングの冗長性を提供しています。SUSE Linux Enterprise High Availability Extension 15 SP3では、テクノロジープレビューとしてRAID10が含まれています。RAID10を試したい場合は、関連する`mdadm`コマンドで`mirror`を`10`に置き換えてください。この章では、Cluster MDの作成および使用方法を示します。

23.1 概念の概要

Cluster MDは、クラスタ環境内のRAID1の使用をサポートします。Cluster MDで使用するディスクまたはデバイスは、各ノードからアクセスされます。Cluster MDの一方のデバイスに障害が発生した場合、実行時に他方のデバイスに置き換えることができ、同じ量の冗長性を提供するために再同期されます。Cluster MDでは、調整とメッセージングのためにCorosyncと分散ロックマネージャ(DLM)が必要です。

Cluster MDデバイスは、他の通常のMDデバイスのようにブート時に自動的に開始されません。クラスタ化されたデバイスはリソースエージェントを使用して開始し、DLMリソースが開始されていることを確認する必要があります。

23.2 クラスタ化されたMD RAIDデバイスの作成

要件

- Pacemakerを使用して実行中のクラスタ。
 - DLMのリソースエージェント(DLMの設定方法については、[手順18.1「DLMのベースグループの設定」](#)を参照してください)。
 - 少なくとも2つの共有ディスクデバイス。デバイス障害の場合は自動的にフェールオーバーするスペアとして追加のデバイスを使用できます。
 - インストール済みパッケージ `cluster-md-kmp-default` をインストールします。
1. クラスタの各ノードでDLMリソースが稼動していることを確認し、リソース状態を確認するには、次のコマンドを実行します。


```
root # crm_resource -r dlm -W
```

2. Cluster MDデバイスを作成します。

- 既存の通常のRAIDデバイスがない場合、次のコマンドを実行し、DLMリソースが稼動しているノードでCluster MDデバイスを作成します。

```
root # mdadm --create /dev/md0 --bitmap=clustered \  
--metadata=1.2 --raid-devices=2 --level=mirror /dev/sda /dev/sdb
```

Cluster MDはバージョン1.2のメタデータでのみ動作します。このため、`--metadata`オプションを使用してバージョンを指定することが推奨されます。他の役立つオプションについては、`mdadm`のマニュアルページを参照してください。`/proc/mdstat`で再同期の進捗状況を監視します。

- 既存の通常のRAIDがすでにある場合は、最初に既存のビットマップをクリアしてからクラスタ化されたビットマップを作成します。

```
root # mdadm --grow /dev/mdX --bitmap=none  
root # mdadm --grow /dev/mdX --bitmap=clustered
```

- オプションで、自動フェールオーバーのためのスペアデバイスを使用してCluster MDデバイスを作成するには、1つのクラスタノード上で次のコマンドを実行します。

```
root # mdadm --create /dev/md0 --bitmap=clustered --raid-devices=2 \  
--level=mirror --spare-devices=1 /dev/sda /dev/sdb /dev/sdc --  
metadata=1.2
```

3. UUIDおよび関連するmdパスを取得します。

```
root # mdadm --detail --scan
```

このUUIDはスーパーブロックに保存されているUUIDと一致する必要があります。UUIDの詳細については、`mdadm.conf`のマニュアルページを参照してください。

4. `/etc/mdadm.conf`を開いて、mdデバイス名とそのデバイス名に関連付けられているデバイスを追加します。前のステップのUUIDを使用します。

```
DEVICE /dev/sda /dev/sdb  
ARRAY /dev/md0 UUID=1d70f103:49740ef1:af2afce5:fcf6a489
```

5. Csync2の設定ファイル`/etc/csync2/csync2.cfg`を開き、`/etc/mdadm.conf`を追加します。

```
group ha_group
{
    # ... list of files pruned ...
    include /etc/mdadm.conf
}
```

23.3 リソースエージェントの設定

CRMリソースを次のように設定します。

1. Raid1プリミティブを作成します。

```
crm(live)configure# primitive raider Raid1 \
    params raidconf="/etc/mdadm.conf" raiddev=/dev/md0 \
    force_clones=true \
    op monitor timeout=20s interval=10 \
    op start timeout=20s interval=0 \
    op stop timeout=20s interval=0
```

2. raiderリソースをDLM用に作成したストレージのベースグループに追加します。

```
crm(live)configure# modgroup g-storage add raider
```

addサブコマンドは、デフォルトで新しいグループメンバーを追加します。

まだ実行されてない場合は、g-storageグループのクローンを作成して、すべてのノードで実行できるようにします。

```
crm(live)configure# clone cl-storage g-storage \
    meta interleave=true target-role=Started
```

3. showで変更内容をレビューします。

4. すべて正しいと思われる場合は、commitで変更を送信します。

23.4 デバイスの追加

デバイスを既存のアクティブなCluster MDデバイスに追加するには、最初にそのデバイスが各ノード上で「表示可能」であることを、コマンドを実行して確認します(cat /proc/mdstat)。デバイスが表示できない場合、コマンドは失敗します。

1つのクラスタノード上で次のコマンドを使用します。

```
root # mdadm --manage /dev/md0 --add /dev/sdc
```

追加された新しいデバイスの動作は、Cluster MDデバイスの状態によって異なります。

- ミラーリングされたデバイスの1つのみがアクティブである場合、新しいデバイスはミラーリングされたデバイスの2番目のデバイスになり、回復処理が開始されます。
- Cluster MDデバイスの両方のデバイスがアクティブな場合、新たに追加されたデバイスはスペアデバイスになります。

23.5 一時的に障害が発生したデバイスの再追加

多くの場合、障害は一時的なもので、単一ノードに限定されます。任意のノードでI/O処理中に障害が発生した場合、クラスタ全体のデバイスがfailedとマーク付けされます。

たとえば、あるノードでケーブル障害が発生した場合などに、このようになることがあります。問題を修正した後で、デバイスを再追加できます。新しいパーツを追加してデバイス全体を同期するのではなく、古いパーツのみが同期されます。

デバイスを再追加するには、1つのクラスタノード上で次のコマンドを実行します。

```
root # mdadm --manage /dev/md0 --re-add /dev/sdb
```

23.6 デバイスの削除

置き換えるために実行時にデバイスを削除する前に、次の操作を実行します。

1. `/proc/mdstat`をイントロスペクトしてデバイスに障害が発生しているか確認します。デバイスの前にある(F)を探します。
2. 1つのクラスタノード上で次のコマンドを実行して、デバイスに障害発生させます。

```
root # mdadm --manage /dev/md0 --fail /dev/sda
```

3. 1つのクラスタノード上で次のコマンドを実行して障害が発生したデバイスを削除します。

```
root # mdadm --manage /dev/md0 --remove /dev/sda
```

23.7 障害復旧サイトで通常のRAIDとしてCluster MDをアセンブルする障害復旧サイトで通常のRAIDとしてCluster MDのアセンブルする

障害復旧の際には、障害復旧サイトのインフラストラクチャにPacemakerクラスタスタックがないにもかかわらず、アプリケーションが既存のCluster MDディスク上のデータまたはバックアップからデータにアクセスする必要があるという状況に直面する場合があります。

`-U no-bitmap`オプションを指定して`--assemble`操作を使用し、それに応じてRAIDディスクのメタデータを変更することで、Cluster MD RAIDを通常のRAIDに変換できます。

データ復旧サイトですべてのアレイをアセンブルする方法の例を以下に示します。

```
while read i; do
  NAME=`echo $i | sed 's/.*name=//'|awk '{print $1}'|sed 's/.*:/'`
  UUID=`echo $i | sed 's/.*UUID=//'|awk '{print $1}'`
  mdadm -AR "/dev/md/$NAME" -u $UUID -U no-bitmap
  echo "NAME =" $NAME ", UUID =" $UUID ", assembled."
done < <(mdadm -Es)
```

24 Sambaクラスタリング

クラスタ対応のSambaサーバは、異種混合ネットワークにHigh Availabilityソリューションを提供します。この章では、背景情報とクラスタ対応Sambaサーバの設定方法を説明します。

24.1 概念の概要

TDB (Trivial Database)は、長年にわたって、Sambaによって使用されてきました。TDBでは、複数のアプリケーションが同時に書き込むことができます。すべての書き込み操作を正常に実行し、互いに衝突させないため、TDBは、内部ロッキングメカニズムを使用しています。

CTDB (Cluster Trivial Database)は、既存のTDBの小規模な拡張です。CTDBは、プロジェクトによって、「一時データの保存のために、Sambaなどのプロジェクトによって使用されるTDBデータベースのクラスタ実装」として説明されています。

各クラスタノードは、ローカルCTDBデーモンを実行します。Sambaは、そのTDBに直接書き込むのではなく、そのローカルCTDBデーモンと通信します。それらのデーモンは、ネットワークを介してメタデータを交換しますが、実際の読み取り/書き込み操作は、高速ストレージでローカルコピー上で行われます。CTDBの概念は、[図24.1「CTDBクラスタの構造」](#)に表示されています。



注記: Samba専用CTDB

CTDBリソースエージェントの現在の実装では、Sambaの管理のためだけにCTDBを設定します。他の機能(IPフェールオーバーなど)はすべて、Pacemakerで設定する必要があります。

CTDBは、完全に同種のクラスタに関してのみサポートされます。たとえば、クラスタのすべてのノードが同じアーキテクチャを持つ必要があります。x86とAMD64を混合することはできません。

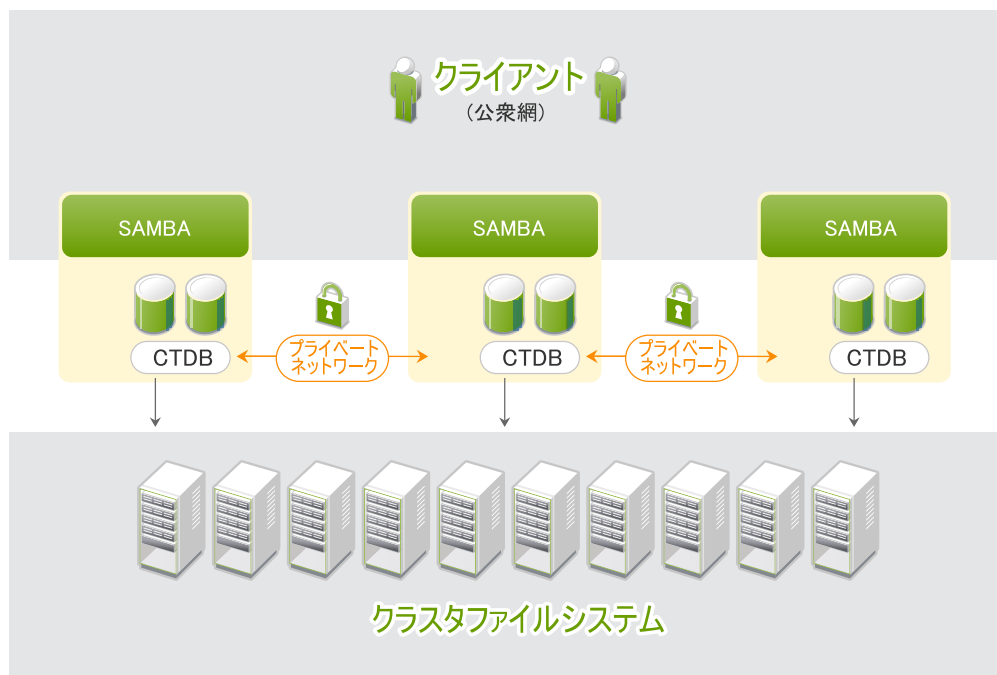


図 24.1: CTDBクラスタの構造

クラスタ対応Sambaサーバは、一定のデータを共有する必要があります。

- UnixのユーザとグループIDをWindowsのユーザとグループに関連付けるマッピングテーブル。
- ユーザーデータベースをすべてのノード間で同期する必要があります。
- Windowsドメイン内のメンバーサーバの参加情報をすべてのノードで利用できる必要があります。
- メタデータをすべてのノードで利用できる必要があります(アクティブSMBセッション、共有接続、各ロックなど)。

クラスタ対応SambaサーバがN+1ノードを持っている場合、Nノードだけのサーバより高速になることを目的としています。1つのノードは、クラスタ非対応のSambaサーバより遅くなることはありません。

24.2 基本的な設定



注記: 変更された設定ファイル

CTDBリソースエージェントは、自動的に`/etc/sysconfig/ctdb`を変更します。`crm ra info CTDB`を使用して、CTDBリソースに指定できるすべてのパラメータを一覧表示してください。

クラスタ対応Sambaサーバをセットアップするには、次の手順に従います。

手順 24.1: クラスタ対応SAMBAサーバの基本セットアップ

1. クラスタを準備します。

- a. 次のパッケージがインストールされていることを確認してから進んでください:
`ctdb`、`tdb-tools`、および `samba` (`smb`および`nmb`リソースに必要)。
- b. このガイドの [パートII「設定および管理」](#) で説明されているように、クラスタ (Pacemaker、OCFS2)を設定します。
- c. OCFS2などの共有ファイルシステムを設定し、マウントします(たとえば、`/srv/clusterfs`にマウント)。詳細については、[第19章「OCFS2」](#)を参照してください。
- d. POSIX ACLをオンにする場合は、それを有効にします。

- 新しいOCFS2ファイルシステムの場合は、次のコマンドを使用します。

```
root # mkfs.ocfs2 --fs-features=xattr ...
```

- 既存のOCFS2ファイルシステムの場合は、次のコマンドを使用します。

```
root # tuneefs.ocfs2 --fs-feature=xattr DEVICE
```

ファイルシステムリソースには、必ず、`acl`オプションを指定します。次のように、`crm`シェルを使用します。

```
crm(live)configure# primitive ocfs2-3 ocf:heartbeat:Filesystem params  
options="acl" ...
```

- e. `ctdb`、`smb`、`nmb`の各サービスが無効になるようにします。

```
root # systemctl disable ctdb  
root # systemctl disable smb
```



```

    op stop interval="0" timeout="100"
crm(live)configure# primitive nmb systemd:nmb \
    op start timeout="60" interval="0" \
    op stop timeout="60" interval="0" \
    op monitor interval="60" timeout="60"
crm(live)configure# primitive smb systemd:smb \
    op start timeout="60" interval="0" \
    op stop timeout="60" interval="0" \
    op monitor interval="60" timeout="60"
crm(live)configure# group g-ctdb ctdb nmb smb
crm(live)configure# clone cl-ctdb g-ctdb meta interleave="true"
crm(live)configure# colocation col-ctdb-with-clusterfs inf: cl-ctdb cl-clusterfs
crm(live)configure# order o-clusterfs-then-ctdb Mandatory: cl-clusterfs cl-ctdb
crm(live)configure# commit

```

7. クラスタ対応のIPアドレスを追加します。

```

crm(live)configure# primitive ip ocf:heartbeat:IPAddr2 params ip=192.168.2.222 \
    unique_clone_address="true" \
    op monitor interval="60" \
    meta resource-stickiness="0"
crm(live)configure# clone cl-ip ip \
    meta interleave="true" clone-node-max="2" globally-unique="true"
crm(live)configure# colocation col-ip-with-ctdb 0: cl-ip cl-ctdb
crm(live)configure# order o-ip-then-ctdb 0: cl-ip cl-ctdb
crm(live)configure# commit

```

`unique_clone_address`が`true`に設定されている場合、IPAddr2リソースエージェン
トはクローンIDを指定のアドレスに追加し、3つの異なるIPアドレスを設定します。
これらは通常必要とされませんが、負荷分散に役立ちます。この項目の詳細について
は、[15.2項「Linux仮想サーバによる負荷分散の設定」](#)を参照してください。

8. 変更をコミットします。

```

crm(live)configure# commit

```

9. 結果を確認します。

```

root # crm status
Clone Set: cl-storage [dlm]
    Started: [ factory-1 ]
    Stopped: [ factory-0 ]
Clone Set: cl-clusterfs [clusterfs]
    Started: [ factory-1 ]
    Stopped: [ factory-0 ]
Clone Set: cl-ctdb [g-ctdb]
    Started: [ factory-1 ]
    Started: [ factory-0 ]
Clone Set: cl-ip [ip] (unique)

```

```
ip:0      (ocf:heartbeat:IPAddr2):      Started factory-0
ip:1      (ocf:heartbeat:IPAddr2):      Started factory-1
```

10. クライアントコンピュータからテストを行います。次のコマンドをLinuxクライアントで実行して、システムからファイルをコピーしたり、システムにファイルをコピーできるかどうか確認します。

```
root # smbclient //192.168.2.222/myshare
```

24.3 Active Directoryドメインへの追加

Active Directory (AD)は、Windowsサーバシステムのディレクトリサービスです。

次の手順は、CTDBクラスタをActive Directoryドメインに追加する方法を概説しています。

1. 手順24.1「クラスタ対応Sambaサーバの基本セットアップ」の説明に従って、CTDBリソースを作成します。
2. 次のように `samba-winbind` パッケージの次のアップデートで加える変更は保存されません。
3. `winbind`サービスを無効にします。

```
root # systemctl disable winbind
```

4. `winbind`クラスタリソースを定義します。

```
root # crm configure
crm(live)configure# primitive winbind systemd:winbind \
    op start timeout="60" interval="0" \
    op stop timeout="60" interval="0" \
    op monitor interval="60" timeout="60"
crm(live)configure# commit
```

5. `g-ctdb`グループを編集して、`nmb`と`smb`リソースの間に`winbind`を挿入します。

```
crm(live)configure# edit g-ctdb
```

`:w` (`vim`)でエディタを保存して閉じます。

6. Active Directoryドメインのセットアップ方法については、Windows Serverのマニュアルを参照してください。この例では、次のパラメータを使用します。

ADおよびDNSサーバ	win2k3.2k3test.example.com
-------------	----------------------------

ADドメイン	2k3test.example.com
クラスタADメンバーのNETBIOS名	CTDB-SERVER

7. 手順24.2「Active Directoryへの参加」

最後に、クラスタをActive Directoryサーバに参加させます。

手順 24.2: ACTIVE DIRECTORYへの参加

1. 次のファイルが、すべてのクラスタホストにインストールされるように、Csync2設定に含まれていることを確認します。

```
/etc/samba/smb.conf
/etc/security/pam_winbind.conf
/etc/krb5.conf
/etc/nsswitch.conf
/etc/security/pam_mount.conf.xml
/etc/pam.d/common-session
```

この作業には、YaSTのCsync2の設定モジュールを使用することもできます。4.5項「すべてのノードへの設定の転送」を参照してください。

2. YaSTを実行し、ネットワークサービスエントリからWindowsドメインメンバーシップモジュールを開きます。
3. ドメインまたはワークグループの設定を入力して、OKをクリックして終了します。

24.4 クラスタ対応Sambaのデバッグとテスト

クライアント対応Sambaサーバのデバッグには、次のツールを使用できます。これらのツールは、さまざまなレベルで動作します。

ctdb_diagnostics

このツールを実行して、クラスタ対応Sambaサーバを診断します。詳細なデバッグメッセージが出力されるので、発生している問題を追跡するのに役立ちます。

ctdb_diagnosticsコマンドは、次のファイルを検索します。これらのファイルは、すべてのノードで利用できる必要があります。

```
/etc/krb5.conf
/etc/hosts
/etc/ctdb/nodes
/etc/sysconfig/ctdb
```

```
/etc/resolv.conf
/etc/nsswitch.conf
/etc/sysctl.conf
/etc/samba/smb.conf
/etc/fstab
/etc/multipath.conf
/etc/pam.d/system-auth
/etc/sysconfig/nfs
/etc/exports
/etc/vsftpd/vsftpd.conf
```

/etc/ctdb/public_addressesファイルと/etc/ctdb/static-routesファイルが存在する場合は、それらもチェックされます。

ping_pong

ping_pongでは、ファイルシステムがCTDBに適合しているかどうかチェックできます。このコマンドは、クラスタファイルシステムの一定のテスト(コヒーレンスやパフォーマンスなどのテスト)を実行して(http://wiki.samba.org/index.php/Ping_pong参照)、高負荷の状況下におけるクラスタの動作を示す情報を提供します。

send_arpツールおよびSendArpリソースエージェント

SendArpリソースエージェントは、/usr/lib/heartbeat/send_arp(または/usr/lib64/heartbeat/send_arp)にあります。**send_arp**ツールはGratuitous ARP (余計なアドレス解決プロトコル)パケットを送信し、他のマシンのARPテーブルを更新するために使用できます。これは、フェールオーバープロセス後の通信問題の識別に役立ちます。Sambaのクラスタ化されたIPアドレスを表示しているのにも関わらず、ノードに接続またはpingできない場合は、**send_arp**コマンドを使用して、ノードはARPテーブルの更新のみが必要であるのかをテストします。

詳細については、<https://gitlab.com/wireshark/wireshark/-/wikis/home>を参照してください。

クラスタファイルシステムの特定の側面をテストするには、次の手順に従います。

手順 24.3: クラスタファイルシステムのコヒーレンスとパフォーマンスをテストする

- 1つのノードで**ping_pong**コマンドを開始します。プレースホルダNはノード数+1で置き換えます。共有ストレージではABSPATH/data.txtファイルが使用可能で、すべてのノード上でアクセスできます(ABSPATHは絶対パスを示しています)。

```
ping_pong ABSPATH/data.txt N
```

1つのノードでだけ実行しているので、ロッキングレートは非常に高いと予想してください。プログラムがロッキングレートを出力しない場合は、クラスタファイルシステムを置き換えます。

2. 同じパラメータを使用して、別のノードで**ping_pong**の2つ目のコピーを開始します。ロックングレートが大幅に下がることを予想できます。使用しているクラスタファイルシステムに次のどれかが当てはまる場合は、クラスタファイルシステムを置き換えます。
 - **ping_pong**がロックングレート(秒単位)を出力しない、
 - 2つのインスタンスのロックングレートがほぼ同じではない。
 - 2つ目のインスタンスの開始後にロックングレートが下がらなかった。
3. **ping_pong**の3つ目のコピーを開始します。もう1つノードを追加し、ロックングレートの変化に注目します。
4. **ping_pong**コマンドを1つずつ終了させます。単一ノードの状態に戻るまで、ロックングレートの増加が観察されるはずですが、予想したような振る舞いが見られなかった場合には、第19章「OCFS2」に記されている詳細を参照してください。

24.5 詳細の参照先

- <http://www.linux-ha.org/doc/man-pages/re-ra-CTDB.html> 
- http://wiki.samba.org/index.php/CTDB_Setup 
- <http://ctdb.samba.org> 
- http://wiki.samba.org/index.php/Samba_%26_Clustering 

25 Rear (Relax-and-Recover)による障害復旧

Relax-and-Recover (旧称「ReaR」。この章ではRearと略記)は、システム管理者による使用を意図した障害復旧フレームワークです。Rearは、障害発生時に保護対象となる特定の運用環境に合わせて調整する必要があるBashスクリプトのコレクションです。

特別な設定を必要とせずにそのまま使用できる障害復旧ソリューションはありません。したがって、障害が発生する「前に」準備しておくことが不可欠です。

25.1 概念の概要

以降のセクションでは、障害復旧の一般的な概念を述べ、Rearを使用した復旧を実現するために必要となる基本的な手順について説明します。また、Rearの要件に関するいくつかの指針、知っておくべき制限事項、およびシナリオとバックアップツールについても紹介します。

25.1.1 障害復旧プランの作成

最悪のシナリオが発生する「前に」、ITインフラストラクチャに重大なリスクがあるかどうかの分析、予算の見積もり、障害復旧プランの作成などの対応策を講じます。障害復旧プランを手元に用意していない場合は、以下の手順ごとに情報を入手します。

- **リスクの分析。** インフラの確かなリスク分析を実施します。可能性のあるすべての脅威を一覧表示し、深刻度を評価します。これらの脅威が発生する可能性を判断し、優先順位を設定します。可能性と影響の簡単な分類を使用することを推奨します。
- **予算のプランニング。** 分析結果には、どのリスクが耐えうるもので、どれがビジネスにとって致命的であるかを全般的に示します。リスクを最小限にする方法およびそれに要するコストを自問して検討します。会社の規模に応じて、IT予算全体の2～15%を障害復旧に使用します。
- **障害復旧プランの作成。** チェックリストの作成、手順のテスト、優先順位の設定と割り当て、ITインフラのインベントリ調査を行います。インフラのサービスが失敗した際、問題に対処する方法を定義します。
- **テスト。** 念入りのプランを定義したら、それをテストします。最低でも1年に1度テストします。ご使用のメインITインフラと同じテストハードウェアを使用します。

25.1.2 障害復旧とは

運用環境に存在するシステムが、ハードウェアの損傷、誤設定、ソフトウェア上の問題など、原因がどのようなものであっても破損した場合は、システムを再作成する必要があります。再作成は、同じハードウェア上または互換性のある代替ハードウェア上で実行できます。バックアップからファイルを復元するだけでは、システムを再作成することにはなりません。システムの再作成では、パーティション、ファイルシステム、マウントポイントの面からのシステムのストレージ作成や、ブートローダの再インストールなどの作業も必要になります。

25.1.3 Rearによる障害復旧

システムが正常に稼働しているときに、ファイルのバックアップを作成し、復旧メディア上に復旧システムを作成します。復旧システムには、復旧インストーラが収められています。

システムが破損した場合は、損傷したハードウェアを必要に応じて交換し、復旧メディアから復旧システムをブートして復旧インストーラを起動します。復旧インストーラによるシステムの再作成では、まず、ストレージにパーティション、ファイルシステム、マウントポイントを作成し、続いてバックアップからファイルを復元します。最後に、ブートローダを再インストールします。

25.1.4 Rearの要件

Rearを使用するには、運用環境を実行するマシンおよびそれと同一のテストマシンが必要です。つまり、同一のシステムが2台以上必要になります。ここでいう「同一」とは、たとえば、ネットワークカードを、同じカーネルドライバを使用する他のネットワークカードに置き換えることができるということです。



警告: 同一のドライバが必要

運用環境のドライバと同じドライバを使用していないハードウェアコンポーネントは、Rearでは同一のコンポーネントとは見なされません。

25.1.5 Rearバージョンの更新

SUSE Linux Enterprise High Availability Extension 15 SP3に付属するパッケージ `rear23a` をインストールします。



注記: 変更ログでの重要な情報の検索

バグ修正、非互換性、および他の問題に関する情報はすべて、パッケージの変更ログで検索できます。障害復旧手順を再検証する必要がある場合は、Rearのより最新のパッケージバージョンも確認することをお勧めします。

Rearには次の問題がありますので注意してください。

- UEFIシステムで障害復旧を許可するには、少なくともRearバージョン 1.18.aおよびパッケージ `ebiso` をインストールします。このバージョンのみが新しいヘルパーツール `/usr/bin/ebiso` をサポートします。このヘルパーツールは、UEFIブート可能RearシステムISOイメージの作成に使用されます。
- 特定のRearバージョンによる障害復旧手順をテスト済みで、その手順が十分に機能しているのであれば、Rearを更新しないでください。Rearパッケージをそのまま保持し、障害復旧手法を変更しないようにします。
- Rearの各バージョン更新は、インストール済みのバージョンが誤って別のバージョンに置き換えられることがないよう、相互に意図的に衝突する別個のパッケージとして提供されています。

次の場合に、既存の障害復旧手順を完全に再検証する必要があります。

- Rearバージョンの更新ごとに。
- Rearを手動で更新する場合。
- Rearで使用されるソフトウェアごとに。
- `parted`、`btrfs`、などの低レベルのシステムコンポーネントを更新する場合。

25.1.6 Btrfsに伴う制限事項

Btrfsを使用する場合は次の制限事項が発生します。

システムにサブボリュームは存在しても、スナップショットのサブボリュームが存在しない場合

Rearバージョン1.17.2.a以上が必要です。このバージョンは、スナップショットのサブボリュームが存在しない「通常の」Btrfsサブボリューム構造の再作成をサポートしています。

システムにスナップショットのサブボリュームが存在する場合



警告

ファイルベースのバックアップソフトウェアでは、Btrfsスナップショットのサブボリュームを通常どおりにはバックアップできず、復元することも「できません」。

Btrfsにはコピーオンライト機能があることから、Btrfsファイルシステム上にある最近のスナップショットサブボリュームはディスク容量をほとんど必要としません。一方、ファイルベースのバックアップソフトウェアを使用すると、これらのファイルは完全なファイルとしてバックアップされます。最終的に、これらのファイルはその本来のファイルサイズで2回バックアップされることになります。したがって、元のシステム上に以前に存在していたときと同じ状態にスナップショットを復元することができません。

SLEシステムに適合するRear設定が必要な場合

たとえば、SLE12 GA、SLE12 SP1、およびSLE12 SP2の設定には、いくつかの異なる不適合Btrfsのデフォルト構造があります。そのため、適合するRear設定ファイルを使用することはたいへん重要です。[/usr/share/rear/conf/examples/SLE12*-btrfs-example.conf](#)のサンプルファイルを参照してください。

25.1.7 シナリオとバックアップのツール

Rearでは、ハードディスク、フラッシュディスク、DVD/CD-RなどのローカルメディアからのブートやPXEを介したブートが可能な障害復旧システム(システム固有の復旧インストーラなど)を作成できます。バックアップデータは、[例 25.1](#)に説明があるNFSなどのネットワークファイルシステムに保存できます。

Rearは、ファイルのバックアップに取って代わるツールではなく、それを補完するツールです。Rearは、汎用的な**tar**コマンドのほか、いくつかのサードパーティのバックアップツールをデフォルトでサポートしています。このようなバックアップツールとして、Tivoli Storage Manager、QNetix Galaxy、Symantec NetBackup、EMC NetWorker、HP DataProtectorなどがあります。バックアップツールとしてEMC NetWorkerを使用したRearの設定例については[例 25.2](#)を参照してください。

25.1.8 基本手順

障害発生時にRearを使用して効果的な復旧を実現するには、以下の基本手順を実行する必要があります。

Rearおよびバックアップソリューションのセットアップ

この手順では、Rear設定ファイルの編集、Bashスクリプトの調整、使用するバックアップソリューションの設定などのタスクを実行します。

復旧インストールシステムの作成

保護対象のシステムが稼働しているときに、`rear mkbackup`コマンドを使用して、ファイルのバックアップを作成し、システム固有のRear復旧インストーラなどの復旧システムを生成します。

復旧プロセスのテスト

Rearを使用して障害復旧メディアを作成した場合は、その障害復旧プロセスを必ず十分にテストしておくようにします。ここでは、運用環境を構成するハードウェアと「同一の」ハードウェアを備えたテストマシンの使用が不可欠です。詳細については、[25.1.4 項「Rearの要件」](#)を参照してください。

障害からの復旧

障害が発生した場合は、必要に応じて損傷したハードウェアを交換します。続いて、Rear復旧システムをブートし、`rear recover`コマンドで復旧インストーラを起動します。

25.2 Rearおよびバックアップソリューションのセットアップ

Rearをセットアップするには、少なくともRear設定ファイル`/etc/rear/local.conf`を編集する必要があります。さらに、Rearフレームワークを構成するBashスクリプトも必要に応じて編集します。

特に、Rearが実行する以下のタスクの定義が必要です。

- **システムをUEFIを使用してブートする場合。** システムをUEFIブートローダを使用してブートする場合は、パッケージ `ebiso` をインストールし、次の行を `/etc/rear/local.conf` に追加します。

- **ファイルをバックアップする方法および障害復旧システムを作成して保存する方法。** これは、`/etc/rear/local.conf`で設定する必要があります。
- **正確な再作成を必要とする対象(パーティション、ファイルシステム、マウントポイントなど)。** これは、`/etc/rear/local.conf`で定義できます(たとえば、除外対象を定義できます)。標準とは異なるシステムを再作成するには、Bashスクリプトの拡張が必要になることがあります。
- **復旧プロセスの仕組み。** Rearで復旧インストーラを生成する方法の変更やRear復旧インストーラによる実行タスクとの適合を可能にするには、Bashスクリプトの編集が必要です。

Rearを設定するには、`/etc/rear/local.conf`設定ファイルに目的のオプションを追加します(これまで使用されてきた設定ファイル`/etc/rear/sites.conf`は、パッケージから削除されました。なお、このファイルを前回のセットアップから引き継いでいる場合、Rearでは引き続きこのファイルが使用されます)。

すべてのRear設定変数とそのデフォルト値は、`/usr/share/rear/conf/default.conf`に設定されています。`/etc/rear/local.conf`などで設定されているユーザ設定に合わせたサンプルファイルのいくつか(*`example.conf`)は、`examples`サブディレクトリ下にあります。詳細については、Rearのマニュアルで該当のページを参照してください。

適合するサンプル設定ファイルをまずはテンプレートとして使用し、必要に応じて修正することで、個別の設定ファイルを作成します。いくつかのサンプル設定ファイルからさまざまなオプションをコピーし、システムに適合した特定の`/etc/rear/local.conf`ファイルにそれらをペーストします。特定の構成向けに利用できる変数の概要などが含まれるため、元のサンプル設定ファイルをそのまま使用しないでください。

例 25.1: NFSサーバを使用したファイルバックアップの保存

Rearはさまざまなシナリオで使用できます。以下の例では、ファイルのバックアップを収めるストレージとしてNFSサーバを使用しています。

1. SUSE Linux Enterprise Server 15 SP3の『管理ガイド (<https://documentation.suse.com/sles-15/html/SLES-all/cha-nfs.html>)』の説明に従って、YaSTを使用してNFSサーバを設定します。
2. 目的のNFSサーバの設定を`/etc/exports`ファイルで定義します。NFSサーバ上でバックアップデータの保存先とするディレクトリに、適切なマウントオプションが設定されていることを確認します。例:

```
/srv/nfs *([...],rw,no_root_squash,[...])
```

`/srv/nfs`をNFSサーバ上のバックアップデータへのパスに置き換えて、マウントオプションを調整します。`rear mkbbackup`コマンドが`root`として実行されるので、`no_root_squash`が必要になる場合があります。

3. さまざまなBACKUPパラメータ(設定ファイル`/etc/rear/local.conf`に記述されています)を調整して、該当のNFSサーバ上にRearからファイルのバックアップを保存できるようにします。この例は、インストールしたシステムの`/usr/share/rear/conf/examples/SLE*-example.conf`にあります。

例 25.2: サードパーティのバックアップツール(EMC NETWORKERなど)の使用

`tar`の代わりにサードパーティのバックアップツールを使用するには、Rear設定ファイルを適切に設定する必要があります。

以下は、EMC NetWorkerを使用する場合の設定例です。この設定スニペットを`/etc/rear/local.conf`に追加し、それぞれのセットアップに応じて調整します。

```
BACKUP=NSR
OUTPUT=ISO
BACKUP_URL=nfs://host.example.com/path/to/rear/backup
OUTPUT_URL=nfs://host.example.com/path/to/rear/backup
NSRSERVER=backupserver.example.com
RETENTION_TIME="Month"
```

25.3 復旧インストールシステムの作成

25.2項の説明に従ってRearを設定した後、Rear復旧インストーラを持つ復旧インストールシステムを作成したうえで、以下のコマンドを使用してファイルのバックアップを作成します。

```
rear -d -D mkbbackup
```

このコマンドでは以下の手順が実行されます。

1. ターゲットシステムを分析し、ディスクのレイアウト(パーティション、ファイルシステム、マウントポイント)やブートローダに関する情報を中心として必要な情報を収集する。
2. 最初の手順で収集した情報を使用して、ブート可能な復旧システムを作成する。ここで得られるRear復旧インストーラは、障害から保護する個々のシステム「専用」のインストーラです。このインストーラは、この固有のシステムを再作成する目的でのみ使用できます。
3. 設定済みのバックアップツールを呼び出し、システムとユーザファイルをバックアップする。

25.4 復旧プロセスのテスト

復旧システムを作成した後、運用マシンと同一のハードウェアを備えたテストマシンで復旧プロセスをテストします。25.1.4項「Rearの要件」も参照してください。テストマシンの設定が適切で、メインマシンの代わりとして機能できることを確認します。



警告: 運用マシンと同一のハードウェア上での包括的なテスト

マシン上で障害復旧プロセスを十分にテストする必要があります。復旧手順を定期的にテストし、すべてが想定どおりに機能することを確認します。

手順 25.1: テストマシン上での障害復旧の実行

1. 25.3項で作成した復旧システムをDVDやCDに書き込み、復旧メディアを作成します。PXEを介したネットワークブートとすることもできます。
2. 復旧メディアからテストマシンをブートします。
3. メニューからRecover (復旧)を選択します。
4. `root`としてログインします(パスワードは必要なし)。
5. 次のコマンドを入力して復旧インストーラを起動します。

```
rear -d -D recover
```

このプロセスでRearが実行する手順の詳細については[回復プロセス](#)を参照してください。

6. 復旧プロセスが完了した後、システムが正常に再作成されたかどうか、および運用環境で元のシステムの代替として機能するかどうかを確認します。

25.5 障害からの復旧


障害が発生した場合には、必要に応じて損傷したハードウェアを取り替えます。次に、[手順 25.1](#)の説明に従って、修復したマシンまたは元のシステムの代替として機能することをテスト済みの同一構成のマシンを使用して手順を進めます。

`rear recover`コマンドでは以下の手順が実行されます。

回復プロセス

1. ディスクのレイアウト(パーティション、ファイルシステム、およびマウントポイント)を復元する。
2. バックアップからシステムとユーザファイルを復元する。
3. ブートローダを復元する。

25.6 詳細の参照先

- http://en.opensuse.org/SDB:Disaster_Recovery 
- [rear](#) マニュアルページ
- </usr/share/doc/packages/rear/>

IV 付録

- A トラブルシューティング **340**
- B 命名規則 **350**
- C クラスタ管理ツール(コマンドライン) **351**
- D rootアクセスなしでのクラスタレポートの実行 **353**

A トラブルシューティング

時として理解しにくい奇妙な問題が発生することがあります。High Availabilityでの実験を開始したときには、特にそうです。それでも、High Availabilityの内部プロセスを詳しく調べるために使用できる、いくつかのユーティリティがあります。この章では、さまざまなソリューションを推奨します。

A.1 インストールと最初のステップ

パッケージのインストールやクラスタのオンライン化では、次のように問題をトラブルシュートします。

HAパッケージはインストールされているか

クラスタの構成を管理に必要なパッケージは、High Availability Extensionで使用できるHigh Availabilityインストールパターンに付属しています。

High Availability Extensionが各クラスタノードにSUSE Linux Enterprise Server 15 SP3の拡張としてインストールされているか、High Availabilityパターンが『インストールおよびセットアップクイックスタート』で説明するように各マシンにインストールされているか、確認します。

初期設定がすべてのクラスタノードについて同一か

相互に通信するため、第4章「YaSTクラスタモジュールの使用」で説明するように、同じクラスタに属するすべてのノードは同じ`bindnetaddr`、`mcastaddr`、`mcastport`を使用する必要があります。

`/etc/corosync/corosync.conf`で設定されている通信チャンネルとオプションがすべてのクラスタノードに関して同一かどうか確認します。

暗号化通信を使用する場合は、`/etc/corosync/authkey`ファイルがすべてのクラスタノードで使用可能かどうかを確認します。

すべての`corosync.conf`設定(`nodeid`以外)が同一で、すべてのノードの`authkey`ファイルが同一でなければなりません。

ファイアウォールで`mcastport`による通信が許可されているか

クラスタノード間の通信に使用される`mcastport`がファイアウォールでブロックされている場合、ノードは相互に認識できません。YaSTまたはブートストラップスクリプト(第4章「YaSTクラスタモジュールの使用」と項目「インストールおよびセットアップクイックスタート」でそれぞれ説明)で初期セットアップを行っているときに、ファイアウォール設定は通常、自動的に調整されます。

mcastportがファイアウォールでブロックされないようにするには、各ノードのファイアウォールの設定を確認します。

PacemakerとCorosyncが各クラスタノードで開始されているか

通常、Pacemakerを開始すると、Corosyncサービスも開始します。両方のサービスが実行されているかどうかを確認するには、次のコマンドを実行します。

```
root # crm cluster status
```

両方のサービスが実行されていない場合は、次のコマンドを実行して開始します。

```
root # crm cluster start
```

A.2 ログ記録

ログファイルはどこにあるか

Pacemakerのログファイルは、/var/log/pacemakerディレクトリに書き込まれます。Pacemakerのメインログファイルは、/var/log/pacemaker/pacemaker.logです。ログファイルが見つからない場合は、Pacemaker独自の設定ファイルである/etc/sysconfig/pacemakerのログ設定を確認してください。PCMK_logfileがそこで設定されている場合、Pacemakerはこのパラメータで定義したパスを使用します。すべての関連ログファイルを表示するクラスタ全体のレポートが必要な場合は、詳細については、すべてのクラスタノードの分析を含むレポートを作成するにはどうしたらよいですか。を参照してください。

監視を有効にしているのに、ログファイルに監視操作の記録が残っていないのはなぜですか。

pacemaker-execdデーモンは、エラーが発生しない限り、複数の監視操作をログに記録しません。複数の監視操作をすべてログ記録すると、多量のノイズが発生してしまいます。そのため、複数の監視操作は、1時間に1度だけ記録されます。

failedメッセージだけが出ました。詳細情報を取得できますか。

コマンドに--verboseパラメータを追加してください。これを複数回行くと、デバッグ出力が非常に詳細になります。役立つヒントについては、ログ記録データ(sudo journalctl -n)を参照してください。

ノードとリソースすべての概要を確認するにはどうしたらよいですか。

crm_monコマンドを使用してください。次のコマンドは、リソース操作履歴(-oオプション)と非アクティブなリソース(-r)を表示します。

```
root # crm_mon -o -r
```

表示内容は、ステータスが変わると、更新されます(これをキャンセルするには、`Ctrl-C` を押します)。次に例を示します

例 A.1: 停止されたリソース

```
Last updated: Fri Aug 15 10:42:08 2014
Last change: Fri Aug 15 10:32:19 2014
Stack: corosync
Current DC: bob (175704619) - partition with quorum
Version: 1.1.12-ad083a8
2 Nodes configured
3 Resources configured

Online: [ alice bob ]

Full list of resources:

my_ipaddress      (ocf:heartbeat:Dummy): Started bob
my_filesystem     (ocf:heartbeat:Dummy): Stopped
my_webserver      (ocf:heartbeat:Dummy): Stopped

Operations:
* Node bob:
  my_ipaddress: migration-threshold=3
    + (14) start: rc=0 (ok)
    + (15) monitor: interval=10000ms rc=0 (ok)
* Node alice:
```

『Explained (Pacemaker)』PDF (<http://www.clusterlabs.org/pacemaker/doc/> から入手可能)では、「How are OCF Return Codes Interpreted?」セクションで3つの異なる復元タイプを説明しています。

ログはどのように表示しますか。

クラスタで発生している現象をより詳しく表示するには、次のコマンドを使用します。

```
root # crm history log [NODE]
```

`NODE`は、調べたいノードに置き換えるか、空のままにします。詳細については、[A.5項「履歴」](#)を参照してください。

A.3 リソース

リソースはどのようにクリーンアップしますか。

次のコマンドを使用してください。

```
root # crm resource list
```

```
crm resource cleanup rscid [node]
```

ノードを指定しないと、すべてのノードでリソースがクリーンアップされます。詳細については、[8.4.4項「リソースのクリーンアップ」](#)を参照してください。

現在既知のリソースを一覧表示するにはどうしたらよいですか。

コマンド `crm_resource list` を使用して、現在のリソースの情報を表示できます。

リソースを設定しましたが、いつも失敗します。なぜですか。

OCFスクリプトを確認するには、たとえば、次の `ocf-tester` コマンドを使用します。

```
ocf-tester -n ip1 -o ip=YOUR_IP_ADDRESS \  
/usr/lib/ocf/resource.d/heartbeat/IPaddr
```

パラメータを増やすには、`-o` を複数回使用します。必須パラメータとオプションパラメータのリストは、`crm ra info AGENT` の実行によって取得できます。たとえば、次のようにします。

```
root # crm ra info ocf:heartbeat:IPaddr
```

`ocf-tester` を実行する場合は、その前に、リソースがクラスターで管理されていないことを確認してください。

リソースがフェールオーバーせず、エラーが出ないのはなぜですか。

終端ノードは `unclean` (アンクリーン) と考えられる場合があります。その場合には、それをフェンシングする必要があります。STONITHリソースが動作していない、または存在しない場合、残りのノードはフェンシングが実行されるのを待機することになります。フェンシングのタイムアウトは通常長いので、問題の兆候がはっきりと現れるまでには (仮に現れたとしても)、かなり長い時間がかかることがあります。

さらに別の可能性としては、単にこのノードでのリソースの実行が許可されていないという場合があります。このことは、過去にエラーが発生し、それが正しく「解決」されていないために生じることがあります。または、以前に行った管理上の操作が原因である場合もあります。つまり、負のスコアを持つ場所の制約のためです。そのような場所の制約は、たとえば、`crm resource migrate` コマンドによって挿入されることがあります。

リソースがどこで実行されるかを予測できないのはなぜですか。

リソースに対して場所の制約が設定されていない場合、その配置は、(ほとんど)ランダムなノード選択によって決まります。どのノードでリソースを実行することが望ましいか、常に明示的に指定することをお勧めします。このことは、「すべての」リソースに対して、場所の初期設定を行う必要があるという意味ではありません。関連する(コロケーション)リソースのセットに対して優先指定を設定すれば十分です。ノードの優先指定は次のようになります。

A.4 STONITHとフェンシング

STONITHリソースが開始しないのはなぜですか。

開始(または有効化)操作には、デバイスのステータスのチェックが含まれます。デバイスの準備ができていない場合、STONITHリソースの開始は失敗します。

同時に、STONITHプラグインは、ホストリストを生成するように要求されます。リストが空の場合、STONITHリソースが対象にできるものがないことになるので、いずれにせよシューティングは行われません。STONITHが動作しているホストの名前は、リストから除外されます。ノードが自分自身をシューティングすることはできないからです。

停電デバイスのような、シングルホスト管理デバイスを使用する場合、フェンシングの対象とするデバイスではSTONITHリソースの動作を許可しないようにしてください。- INFINITYの、ノードの場所優先設定(制約)を使用してください。クラスタは、STONITHリソースを、起動できる別の場所に移動します。その際にはそのことが通知されます。

STONITHリソースを設定したのにフェンシングが行われないのはなぜですか。

それぞれのSTONITHリソースは、ホストリストを持つ必要があります。このリストは、手動でSTONITHリソースの設定に挿入される場合、またはデバイス自体から取得される場合があります(たとえば出力名から)。この点は、STONITHプラグインの性質に応じて決まります。`pacemaker-fenced`は、このリストを基に、どのSTONITHリソースがターゲットノードのフェンシングを行えるかを判断します。ノードがリストに含まれている場合に限って、STONITHリソースはノードのシューティング(フェンシング)を行えます。

`pacemaker-fenced`は、動作しているSTONITHリソースから提供されたホストリスト内にノードを見つけられなかった場合、他のノードの`pacemaker-fenced`インスタンスに問い合わせます。他の`pacemaker-fenced`インスタンスのホストリストにもターゲットノードが含まれていなかった場合、フェンシング要求は、開始ノードでタイムアウトのために終了します。

STONITHリソースが失敗することがあるのはなぜですか。

ブロードキャストトラフィックが多すぎると、電源管理デバイスが機能しなくなることがあります。監視操作を少なくして、余裕を持たせてください。フェンシングが一時的にのみ必要な場合(必要が生じないのが最善ですが)、デバイスのステータスは数時間に1回チェックすれば十分です。

また、この種のデバイスの中には、同時に複数の相手と通信するのを拒否するものもあります。このことは、ユーザが端末またはブラウザセッションを開いたままにしている、クラスタがステータスのテストを行おうとした場合には、問題となり得ます。

A.5 履歴

障害の発生したリソースからステータス情報またはログを取得するにはどうしたらよいですか。

history コマンド、およびそのサブコマンド **resource** を使用します。

```
root # crm history resource NAME1
```

これにより、指定したリソースのみの完全な遷移ログが得られます。ただし、複数のリソースを調査することも可能です。その場合、最初のリソース名の後に目的のリソース名を追加します。

一定の命名規則(を参照してください)に従っていれば、**resource** コマンドでリソースのグループを調査するのが容易になります。たとえば、次のコマンドは、**db** で始まるすべてのプリミティブを調査します。

```
root # crm history resource db*
```

/var/cache/crm/history/live/alice/ha-log.txt のログファイルを表示します。

履歴の出力を減らすにはどうしたらよいですか。

history コマンドには、次の2つのオプションがあります。

- **exclude** を使用する
- **timeframe** を使用する

exclude コマンドを使用すると、追加の正規表現を設定して、ログから特定のパターンを除外できます。たとえば、次のコマンドは、SSH、systemd、およびカーネルのメッセージをすべて除外します。

```
root # crm history exclude ssh|systemd|kernel.
```

timeframe コマンドを使用して、出力を特定の範囲に制限します。たとえば、次のコマンドは、8月23日12:00～12:30のイベントをすべて表示します。

```
root # crm history timeframe "Aug 23 12:00" "Aug 23 12:30"
```

後で検査できるように「セッション」を保存するにはどうしたらよいですか。

詳しい調査を要するバグまたはイベントが発生した場合、現在のすべての設定を保存しておく役に立ちます。このファイルをサポートに送信したり、**bzless** で表示したりできます。例:

```
crm(live)history# timeframe "Oct 13 15:00" "Oct 13 16:00"
```

```
crm(live)history# session save tux-test
crm(live)history# session pack
Report saved in '/root/tux-test.tar.bz2'
```

A.6 Hawk2

自己署名証明書の置き換え置換

Hawk2の最初の起動で自己署名証明書に関する警告が発行されるのを避けるには、自動生成された証明書を、独自の証明書または公式認証局(CA)によって署名された証明書で置き換えてください。

1. /etc/hawk/hawk.keyを秘密鍵で置き換えます。
2. /etc/hawk/hawk.pemをHawk2が提供する証明書で置き換えます。
3. Hawk2サービスを再起動して、新しい証明書を再ロードします。

```
root # systemctl restart hawk-backend hawk
```

root:haclientにファイルの所有権を変更して、そのファイルがグループにアクセスできるようにします。

```
chown root:haclient /etc/hawk/hawk.key /etc/hawk/hawk.pem
chmod 640 /etc/hawk/hawk.key /etc/hawk/hawk.pem
```

A.7 その他

すべてのクラスタノードでコマンドを実行するにはどうしたらよいですか。

この作業を実行するには、psshコマンドを使用します。必要であれば pssh をインストールします。ファイル(たとえば hosts.txt)を作成し、その中に操作する必要があるノードのIPアドレスまたはホスト名を含めます。sshを使用して hosts.txt ファイルに含まれている各ホストにログインしていることを確認します。準備ができたなら、psshを実行します。hosts.txt ファイルを(オプション -h で)指定し、対話モードを使用してください(オプション -i)。次のようになります。

```
pssh -i -h hosts.txt "ls -l /corosync/*.conf"
[1] 08:28:32 [SUCCESS] root@venus.example.com
-rw-r--r-- 1 root root 1480 Nov 14 13:37 /etc/corosync/corosync.conf
[2] 08:28:32 [SUCCESS] root@192.168.2.102
```



```
-rw-r--r-- 1 root root 1480 Nov 14 13:37 /etc/corosync/corosync.conf
```

クラスタはどのような状態でしょうか。

クラスタの現在のステータスを確認するには、`crm_mon`か`crm status`のどちらかを使用します。これによって、現在のDCと、現在のノードに認識されているすべてのノードとリソースが表示されます。

クラスタ内の一部のノードが相互に通信できないのはなぜですか。

これにはいくつかの理由が考えられます。

- まず、設定ファイル`/etc/corosync/corosync.conf`を調べます。マルチキャストまたはユニキャストアドレスがクラスタ内のすべてのノードで同一かどうか確認します(キー`mcastaddr`を含む`interface`セクションを調べてください)。
- ファイアウォール設定を確認します。
- スイッチがマルチキャストまたはユニキャストアドレスをサポートしているか確認します。
- ノード間の接続が切断されていないかどうか確認します。その原因の大半は、ファイアウォールの設定が正しくないことです。また、これは「スプリットブレイン」の理由にもなり、クラスタがパーティション化されます。

OCFS2デバイスをマウントできないのはなぜですか。

ログメッセージ(`sudo journalctl -n`)に次の行があるか確認してください。

```
Jan 12 09:58:55 alice pacemaker-execd: [3487]: info: RA output: [...]
ERROR: Could not load ocfs2_stackglue
Jan 12 16:04:22 alice modprobe: FATAL: Module ocfs2_stackglue not found.
```

この場合、カーネルモジュール`ocfs2_stackglue.ko`がありません。インストールしたカーネルに応じて、パッケージ`ocfs2-kmp-default`、`ocfs2-kmp-pae`、または`ocfs2-kmp-xen`をインストールします。

すべてのクラスタノードの分析を含むレポートを作成するにはどうしたらよいですか。

`crm`シェルで、`crm report`を使用してレポートを作成します。このツールは以下を収集します。

- クラスタ全体のログファイル
- パッケージ状態
- DLM/OCFS2状態
- システム情報

- CIB履歴
- コアダンプレポートの解析(debuginfoパッケージがインストールされている場合)

通常は、次のコマンドで**crm report**を実行します。

```
root # crm report -f 0:00 -n alice -n bob
```

このコマンドは、ホストaliceおよびbob上の午前0時以降のすべての情報を抽出し、現在のディレクトリに**crm_report-DATE.tar.bz2**という名前の*.tar.bz2アーカイブを作成します(例: **crm_report-Wed-03-Mar-2012**)。特定のタイムフレームのみを対象とする場合は、**-t**オプションを使用して終了時間を追加します。



警告: 機密の情報は削除してください

crm reportツールは、CIBとPE入力ファイルから機密の情報を削除しようと試みますが、すべてを知ることはできません。機密性の高い情報がある場合は、**-p**オプションを使用して追加のパターンを指定してください(マニュアルページを参照)。ログファイルと**crm_mon**、**ccm_tool**、および**crm_verify**の出力は、フィルタされません。

データをいずれの方法でも共有する前に、アーカイブをチェックして、公表したくない情報があればすべて削除してください。

さらに追加のオプションを使用して、コマンドの実行をカスタマイズします。たとえば、Pacemakerクラスタがある場合は、確実にオプション**-A**を追加する必要がありますでしょう。別のユーザがクラスタに対するパーミッションを持っている場合は、(**root**および**hacluster**に加えて)**-u**オプションを使用してこのユーザを指定します。非標準のSSHポートを使用する場合は、**-X**オプションを使用して、ポートを追加します(たとえば、ポート3479では、**-X "-p 3479"**を使用)。その他のオプションは、**crm report**のマニュアルページに記載されています。

crm reportで、関連するすべてのログファイルを分析し、ディレクトリ(またはアーカイブ)を作成したら、**ERROR**という文字列(大文字)があるかどうかログファイルをチェックします。レポートの最上位ディレクトリにある最も重要なファイルは次のとおりです。

analysis.txt

すべてのノードで同一である必要があるファイルを比較します。

corosync.txt

Corosync設定ファイルのコピーを格納します。

crm_mon.txt

crm_mon コマンドの出力を格納します。

description.txt

ノード上のすべてのクラスタパッケージのバージョンを格納します。ノード固有の sysinfo.txt ファイルもあります。これは最上位ディレクトリにリンクしています。

このファイルは、発生した問題を説明して <https://github.com/ClusterLabs/crmsh/issues> に送信するためのテンプレートとして使用できます。

members.txt

すべてのノードのリストです。

sysinfo.txt

関連するすべてのパッケージ名とそのバージョンのリストが記述されています。さらに、元のRPMパッケージとは異なる設定ファイルのリストもあります。ノード固有のファイルは、ノードの名前を持つサブディレクトリに保存されます。ここには、それぞれのノードのディレクトリ /etc のコピーが保存されます。引数を単純化する必要がある場合は、設定ファイル /etc/crm/crm.conf のセクション report にデフォルト値を設定します。詳細は、マニュアルページ man 8 crmsh_hb_report に記載されています。

A.8 詳細の参照先

クラスタリソースの設定、およびHigh Availabilityクラスタの管理とカスタマイズなど、Linuxの高可用性に関するその他の情報については、<http://clusterlabs.org/wiki/Documentation> を参照してください。

B 命名規則

このガイドでは、クラスタノードと名前、クラスタリソース、および制約に次の命名規則を使用します。

クラスタノード

クラスタノードは名を使用します:。
alice、bob、charlie、doro、およびeris

クラスタサイトの名前

クラスタサイトには、都市の名前が付けられます:
アムステルダム、ベルリン、キャンベラ、福岡、ギザ、ハノイ、およびイスタンブール

クラスタリソース

プリミティブ	プレフィックスなし
グループ	プレフィックスg-
クローン	プレフィックスcl-
プロモータブルクローン	(以前はマルチステートリソースと呼ばれていました) プレフィックスms-

制約

順序の制約	プレフィックスo-
場所の制約	プレフィックスloc-
コロケーション制約	プレフィックスcol-

C クラスタ管理ツール(コマンドライン)

High Availability Extensionには、クラスタをコマンドラインから管理する際に役立つ、包括的なツールセットが付属しています。この章では、CIBおよびクラスタリソースでのクラスタ構成を管理するために必要なツールを紹介します。リソースエージェントを管理する他のコマンドラインツールや、セットアップのデバッグ(およびトラブルシューティング)に使用するツールについては、[付録A トラブルシューティング](#)で説明されています。



注記: crmshの使用

これは、エキスパート専用のツールです。通常、crmシェル(crmsh)を使用したクラスタ管理が推奨されている方法です。

次のリストは、クラスタ管理に関連するいくつかの作業を示しており、これらの作業を実行するために使用するツールを簡単に説明しています。

クラスタのステータスの監視

`crm_mon`コマンドでは、クラスタのステータスと設定を監視できます。出力には、ノード数、uname、UUID、ステータス、クラスタで設定されたリソース、それぞれの現在のステータスが含まれます。`crm_mon`の出力は、コンソールに表示したり、HTMLファイルに出力したりできます。`status`セクションのないクラスタ設定ファイルが指定された場合、`crm_mon`はファイルに指定されたノードとリソースの概要を作成します。このツールの使用法およびコマンド構文に関する詳細については、[crm_mon](#)マニュアルページを参照してください。

CIBの管理

`cibadmin`コマンドは、CIBを操作するための低レベル管理コマンドです。CIBのすべてまたは一部のダンプ、CIBのすべてまたは一部の更新、すべてまたは一部の変更、CIB全体の削除、その他のCIB管理操作に使用できます。このツールの使用法およびコマンド構文に関する詳細については、[cibadmin](#)マニュアルページを参照してください。

設定の変更の管理

`crm_diff`コマンドは、XMLパッチの作成と適用をサポートします。クラスタの環境設定の2つのバージョンの違いを視覚的に確認する場合や、変更を保存しておき、後で[cibadmin](#)を使用して適用する場合には便利です。このツールの使用法およびコマンド構文に関する詳細については、[crm_diff](#)マニュアルページを参照してください。

CIB属性の操作

crm_attributeコマンドで、CIBで使用されているノード属性およびクラスタ設定オプションを問い合わせて操作できます。このツールの使用法およびコマンド構文に関する詳細については、crm_attributeマニュアルページを参照してください。

クラスタ設定の検証

crm_verifyコマンドは、設定データベース(CIB)の整合性およびその他の問題を確認します。設定を含むファイルを確認したり、実行中のクラスタに接続したりできます。2種類の問題を報告します。エラーを解決しないと High Availability Extensionが正常に機能できず、警告の解決は管理者が担当します。crm_verifyは新規または変更された設定の作成を支援します。実行中のクラスタのCIBのローカルコピーを作成し、編集し、crm_verifyを使用して検証し、新規設定をcibadminを使用して適用できます。このツールの使用法およびコマンド構文に関する詳細については、crm_verifyマニュアルページを参照してください。

リソース設定の管理

crm_resourceコマンドは、クラスタ上でリソース関連のさまざまなアクションを実行します。設定されたリソースの定義の変更、リソースの始動と停止、リソースの削除およびノード間でのマイグレートを実行できます。このツールの使用法およびコマンド構文に関する詳細については、crm_resourceマニュアルページを参照してください。

リソースの失敗回数の管理

crm_failcountコマンドは、所定のノードのリソースごとの失敗回数を問い合わせます。このツールは、失敗回数のリセットにも使用でき、リソースが頻繁に失敗したノード上で再度実行できるようにします。このツールの使用法およびコマンド構文に関する詳細については、crm_failcountマニュアルページを参照してください。

ノードのスタンバイステータスの管理

crm_standbyコマンドは、ノードのスタンバイ属性を操作します。スタンバイモードのノードはすべて、リソースをホストすることができず、そのノードにあるリソースは削除する必要があります。スタンバイモードはカーネルの更新などの保守作業を行う場合に便利です。ノードを再びクラスタの完全にアクティブなメンバーにするには、ノードからスタンバイ属性を削除します。このツールの使用法およびコマンド構文に関する詳細については、crm_standbyマニュアルページを参照してください。

D rootアクセスなしでのクラスタレポートの実行

すべてのクラスタノードはSSHによって互いにアクセスする必要があります。`crm report` (トラブルシューティング用)などのツールおよびHawk2の履歴エクスプローラは、ノード間でパスワード不要のSSHアクセスを必要とします。それがない場合、現在のノードからしかデータを収集できません。

パスワード不要のSSH rootアクセスが規定要件を順守しない場合は、次善策を使用してクラスタレポートを実行できます。これは次の基本手順で構成されます。

1. 専用のローカルユーザアカウント(`crm report`実行用)を作成する。
2. できれば標準以外のSSHポートを使用して、そのユーザアカウントにパスワード不要のSSHアクセスを設定する。
3. そのユーザに`sudo`を設定する。
4. そのユーザとして`crm report`を実行する。

デフォルトでは、`crm report`を実行すると、まずrootとしてリモートノードへのログインを試行し、続いてユーザ`hacluster`としてログインを試行します。ただし、ローカルセキュリティポリシーによってSSHを使用したrootログインが禁止されている場合、スクリプトの実行はすべてのリモートノードで失敗します。ユーザ`hacluster`としてスクリプトを実行しようとしても失敗します。これはサービスアカウントであり、そのシェルはログインが禁止されている`/bin/false`に設定されているためです。High Availabilityクラスタのすべてのノードで`crm report`スクリプトを正常に実行する唯一のオプションは、専用のローカルユーザを作成することだけです。

D.1 ローカルユーザアカウントの作成

次の例では、コマンドラインから`hareport`という名前のローカルユーザを作成します。パスワードは、セキュリティ要件を満たしていれば何でも構いません。または、YaSTでユーザアカウントを作成してパスワードを設定することもできます。

手順 D.1: クラスタレポート実行用の専用ユーザアカウントの作成

1. シェルを起動し、ホームディレクトリ`/home/hareport`を持つユーザ`hareport`を作成します。

```
root # useradd -m -d /home/hareport -c "HA Report" hareport
```

2. ユーザのパスワードを設定します。

```
root # passwd hareport
```

3. プロンプトが表示されたら、ユーザのパスワードを入力し、確認のために再度入力します。

❗ 重要: 各クラスタノードで同じユーザが必要

すべてのノードで同じユーザアカウントを作成するため、各ノードで上の手順を繰り返してください。

D.2 パスワード不要のSSHアカウントの設定

手順 D.2: 非標準ポートのSSHデーモンの設定SSHデーモンに対する非標準ポートの設定

デフォルトでは、SSHデーモンおよびSSHクライアントはポート22で通信およびリスンします。ネットワークセキュリティガイドラインによって、デフォルトのSSHポートを大きい番号の別のポートに変更することが要求されている場合は、デーモンの設定ファイル`/etc/ssh/sshd_config`を変更する必要があります。

1. デフォルトポートを変更するには、ファイルで`Port`行を検索してコメント解除し、目的に応じて編集します。たとえば、次のように設定します。

```
Port 5022
```

2. 組織において`root`ユーザが他のサーバにアクセスすることが許可されていない場合、このファイルで`PermitRootLogin`エントリを検索してコメント解除し、`no`に設定します。

```
PermitRootLogin no
```

3. または、次のコマンドを実行して、各行をファイルの末尾に追加します。

```
root # echo "PermitRootLogin no" >> /etc/ssh/sshd_config
root # echo "Port 5022" >> /etc/ssh/sshd_config
```

4. `/etc/ssh/sshd_config`を変更したら、SSHデーモンを再起動して新しい設定を有効にします。

```
root # systemctl restart sshd
```

！ 重要: 各クラスタノードで同じ設定が必要

各クラスタノードで、このSSHデーモンの設定を繰り返してください。

手順 D.3: SSHクライアントに対する非標準ポートの設定非標準ポートのSSHクライアントの設定

クラスタ内のすべてのノードでSSHポートを変更する場合、SSH設定ファイル`/etc/ssh/sshd_config`を変更するのが便利です。

1. デフォルトポートを変更するには、ファイルでPort行を検索してコメント解除し、目的に応じて編集します。たとえば、次のように設定します。

```
Port 5022
```

2. または、次のコマンドを実行して、各行をファイルの末尾に追加します。

```
root # echo "Port 5022" >> /etc/ssh/sshd_config
```

📎 注記: 1つのノードでのみ設定が必要

このSSHクライアントの設定は、クラスタレポートを実行するノードでのみ必要です。

または、`-X`オプションを使用してカスタムSSHポートで**crm report**を実行することも、**crm report**がデフォルトでカスタムSSHポートを使用するように設定することもできます。詳細については、[手順D.5「カスタムSSHポートを使用したクラスタレポートの生成」](#)を参照してください。

手順 D.4: 共有SSH鍵の設定

パスワードを要求されずに、SSHを使用して他のサーバにアクセスできます。これは一見、安全ではないように見えますが、ユーザは自身の公開鍵が共有されているサーバにしかアクセスできないため、実際は非常に安全なアクセス方法です。共有鍵は、その鍵を使用するユーザとして作成する必要があります。

1. クラスタレポートの実行用に作成したユーザアカウントでノードの1つにログインします(上の例では、このユーザアカウントは**hareport**です)。
2. 新しい鍵を生成します。

```
hareport > ssh-keygen -t rsa
```

このコマンドは、デフォルトで2,048ビットの鍵を生成します。鍵のデフォルトの場所は `~/.ssh/` です。鍵にパスフレーズを設定するよう求められます。ただし、パスワードなしでログインする場合は、鍵にパスフレーズがあってはならないため、パスフレーズを入力しないでください。

3. 鍵が生成されたら、公開鍵を他の「それぞれの」ノードにコピーします(鍵を作成したノードを「含む」)。

```
hareport > ssh-copy-id -i ~/.ssh/id_rsa.pub HOSTNAME_OR_IP
```

このコマンドでは、各サーバのDNS名、エイリアス、またはIPアドレスを使用できます。コピー中に、各ノードのホスト鍵を受諾するよう求められるので、hareportユーザーアカウントのパスワードを入力する必要があります(パスワードを入力する必要があるのは、ここだけです)。

4. 鍵をすべてのクラスターノードと共有したら、パスワード不要のSSHを使用して、ユーザhareportとして他のノードにログインできるかどうかをテストします。

```
hareport > ssh HOSTNAME_OR_IP
```

証明書の受諾やパスワードの入力を要求されることなく、自動的にリモートサーバに接続されます。



注記: 1つのノードでのみ設定が必要

毎回同じノードからクラスターレポートを実行する場合は、上の手順は、このノードでのみ実行すれば十分です。そうでない場合は、各ノードでこの手順を繰り返します。

D.3 sudoの設定

sudo コマンドは、通常のユーザを素早く `root` にしてコマンドを発行できるようにします。パスワードの入力は、必要な場合と不要な場合があります。すべてのルートレベルのコマンドに `sudo` アクセスを付与することも、特定のコマンドにのみ付与することもできます。一般的には、`sudo` はエイリアスを使用してコマンド文字列全体を定義します。

`sudo` を設定するには、**visudo** (viでは「ありません」) またはYaSTを使用します。



警告: viは使用しない

コマンドラインからsudoを設定するには、**visudo**を使用して、rootとしてsudoersファイルを編集する必要があります。他のエディタを使用すると、構文エラーやファイルパーミッションエラーが発生し、sudoを実行できないことがあります。

1. rootとしてログインします。
2. /etc/sudoersファイルを開くため、「**visudo**」と入力します。
3. カテゴリHost alias specification、User alias specification、Cmnd alias specification、およびRunas alias specificationを探します。
4. 次のエントリを/etc/sudoers内のそれぞれのカテゴリに追加します。

```
Host_Alias CLUSTER = alice,bob,charlie ❶
User_Alias HA = hareport ❷
Cmnd_Alias HA_ALLOWED = /bin/su, /usr/sbin/crm report * ❸
Runas_Alias R = root ❹
```

- ❶ ホストエイリアスは、sudoユーザがコマンド発行権利を持つサーバ(またはサーバの範囲)を定義します。ホストエイリアスでは、DNS名またはIPアドレスを使用するか、ネットワーク範囲全体を指定できます(例: 172.17.12.0/24)。アクセスの範囲を制限するには、クラスタノードのホスト名のみを指定する必要があります。
 - ❷ ユーザエイリアスでは、複数のローカルユーザアカウントを1つのエイリアスに追加できます。ただし、この場合、使用するアカウントは1つだけであるため、エイリアスの作成を避けることができます。上の例では、クラスタレポート実行用に作成したhareportユーザを追加しています。
 - ❸ コマンドエイリアスは、ユーザが実行できるコマンドを定義します。これは、非ルートユーザが**sudo**を使用する際にアクセスできるコマンドを制限する場合に便利です。この場合、hareportユーザアカウントには、コマンドcrm reportおよびsuに対するアクセスが必要です。
 - ❹ runasエイリアスは、コマンドの実行に使用するアカウントを指定します。この場合は、rootです。
5. 次の2行を検索します。

```
Defaults targetpw
ALL ALL=(ALL) ALL
```

これらは、作成したい設定と衝突するため、無効にします。

```
#Defaults targetpw
#ALL ALL=(ALL) ALL
```

6. User privilege specificationを探します。
7. 上のエイリアスを定義したら、そこに次のルールを追加できます。

```
HA CLUSTER = (R) NOPASSWD:HA_ALLOWED
```

NOPASSWDオプションは、ユーザhareportがパスワードを入力せずにクラスタレポートを実行できるようにします。

！ 重要: 各クラスタノードで同じsudo設定が必要

クラスタ内のすべてのノードでこのsudo設定を行う必要があります。sudoに他の変更は必要なく、再起動が必要なサービス也没有ありません。

D.4 クラスタレポートの生成

上で行った設定でクラスタレポートを実行するには、ノードの1つにユーザhareportとしてログインする必要があります。クラスタレポートを起動するには、crm reportコマンドを使用します。例:

```
root # crm report -f 0:00 -n "alice bob charlie"
```

このコマンドは、指定したノード上の午前0時以降の情報をすべて抽出し、現在のディレクトリにpcmk-DATE.tar.bz2という名前の*.tar.bz2アーカイブを作成します。

手順 D.5: カスタムSSHポートを使用したクラスタレポートの生成

1. カスタムSSHポートを使用する場合、crm reportで-Xを使用して、クライアントのSSHポートを変更します。たとえば、カスタムSSHポートが5022の場合、次のコマンドを使用します。

```
root # crm report -X "-p 5022" [...]
```

2. crm reportのカスタムSSHポートを永続的に設定するには、crm対話型シェルを開始します。

```
crm options
```

3. 次のように入力します。

```
crm(live)options# set core.report_tool_options "-X -oPort=5022"
```

用語集

AutoYaST

AutoYaSTは、ユーザの介入なしで、1つ以上のSUSE Linux Enterpriseシステムを自動的にインストールするためのシステムです。

bindnetaddr (バインドネットワークアドレス)

Corosyncエグゼクティブのバインド先のネットワークアドレス。

boothd (ブースデーモン)

Geoクラスタ内のそれぞれの参加クラスタおよびアービトラータが、サービスであるboothdを実行します。これは、別のサイトで実行しているブースデーモンに接続し、接続性の詳細を交換します。

CCM (コンセンサスクラスタメンバーシップ)

CCMは、どのノードがクラスタを設定するか決定し、この情報をクラスタで共有します。ノードまたはクォーラムの新規追加および損失は、CCMによって通知されます。CCMモジュールはクラスタの各ノード上で実行されます。

CIB (クラスタ情報ベース)

クラスタの設定とステータス(クラスタオプション、ノード、リソース、制約、相互の関係性)の全体的な表現。XMLで作成され、メモリに常駐します。マスタCIBは、**DC (指定コーディネータ)**で保持および保守され、他のノードに複製されます。CIBに対する通常の読み書き操作は、マスタCIBによってシリアルに処理されます。

conntrackツール

カーネル内の接続トラッキングシステムとやり取りできるようにして、iptablesでの「ステートフルな」パケット検査を可能にします。High Availability Extensionによって、クラスタノード間の接続ステータスを同期化するために使用されます。

CRM (クラスタリソースマネージャ)

高可用性クラスタにおける外部とのやり取りをすべて統括する管理エンティティ。High Availability Extensionでは、PacemakerをCRMとして使用します。pacemaker-controldとして実装されるCRMは、多数のコンポーネント(CRM自身のノードとその他のノード両方のローカルリソースマネージャ、非ローカルCRM、管理コマンド、フェンシング機能、メンバーシップ層)と対話します。

crmsh

コマンドラインユーティリティcrmshは、クラスタ、ノード、およびリソースを管理します。

詳細については、[第8章「クラスタリソースの設定と管理\(コマンドライン\)」](#)を参照してください。

Csync2

クラスタ内のすべてのノード、およびGeoクラスタ全体に設定ファイルを複製するために使用できる同期ツールです。

DC (指定コーディネータ)

DCは、クラスタ内にあるすべてのノードの中から選択されます。この操作は、DCがまだ指定されていない場合や、現在のDCがなんらかの理由でクラスタを離脱した場合に行われます。DCは、ノードのフェンシングやリソースの移動など、クラスタ全体におよぶ変更が必要かどうかを判断できる、クラスタ内で唯一のエンティティです。その他すべてのノードは、現在のDCから設定とリソース割り当て情報を取得します。

DLM (分散ロックマネージャ)

DLMは、クラスタファイルシステムのディスクアクセスを調整し、ファイルロックングを管理して、パフォーマンスと可用性を向上します。

DRBD

DRBD®は、高可用性クラスタを構築するためのブロックデバイスです。ブロックデバイス全体が専用ネットワーク経由でミラーリングされ、ネットワークRAID-1として認識されます。

Geoクラスタ

それぞれにローカルクラスタを持つ、複数の地理的に離れたサイトで構成されます。サイトはIPによって交信します。サイト全体のフェールオーバーはより高いレベルのエンティティ(ブース)によって調整されます。Geoクラスタは限られたネットワーク帯域幅および高レイテンシに対応する必要があります。ストレージは同期的にレプリケートされます。

Geoクラスタ(地理的に離れたクラスタ)

[Geoクラスタ](#)を参照してください。

LRM (ローカルリソースマネージャ)

ローカルリソースマネージャは、各ノードのPacemaker層とリソース層の間に存在し、`pacemaker-execd`デーモンとして実装されます。このデーモンにより、Pacemakerでのリソースの起動、停止、監視が可能になります。

mcastaddr (マルチキャストアドレス)

Corosyncエグゼクティブによるマルチキャストに使用されるIPアドレス。このIPアドレスはIPv4またはIPv6のいずれかに設定できます。

mcastport (マルチキャストポート)

クラスタ通信に使用されるポート。

multicast (マルチキャスト)

ネットワーク内で一対多数の通信に使用される技術で、クラスタ通信に使用できます。Corosyncはマルチキャストとユニキャストの両方をサポートしています。

pacemaker-controld (クラスタコントローラデーモン)

CRMは、pacemaker-controldというデーモンとして実装されます。crmdは各クラスタノード上にインスタンスを持ちます。マスタとして動作するpacemaker-controldインスタンスを1つ選択することにより、クラスタのすべての意思決定が一元化されます。選択したpacemaker-controldプロセス(またはそのプロセスが実行されているノード)で障害が発生したら、新しいpacemaker-controldプロセスが確立されます。

PE (ポリシーエンジン)

ポリシーエンジンは、`pacemaker-schedulerd`デーモンとして実装されます。クラスタ遷移が必要になると、`pacemaker-schedulerd`はクラスタの現在の状態と設定を基に、クラスタの次の状態を計算します。また、次の状態を達成するためにどんなアクションを行う必要があるかも決定します。

RA (リソースエージェント)

プロキシとして機能してリソースを管理する(リソースの開始、停止、監視などを行う)スクリプト。High Availability Extensionでは、別の種類のリソースエージェントがサポートされています。詳細については、[6.3.2項「サポートされるリソースエージェントクラス」](#)を参照してください。

Rear (Relaxおよび回復)

障害復旧イメージを作成するための管理ツールセット。

RRP (冗長リングプロトコル)

ネットワーク障害の一部または全体に対する災害耐性のために、複数の冗長ローカルエリアネットワークが使用できるようになります。この方法では、ひとつのネットワークが作動中である限り、クラスタ通信を維持できます。Corosyncはトータム冗長リングプロトコルをサポートします。

SBD (STONITHブロックデバイス)

共有ブロックストレージ(SAN、iSCSI、FCoEなど)を介したメッセージの交換を通じて、ノードフェンシングメカニズムを提供します。ディスクレスモードでも使用できます。動作異常のノードが本当に停止したかどうかを確認するために、各ノードではハードウェアまたはソフトウェアのウォッチドッグが必要です。

SFEX (共有ディスクファイル排他制御)

SFEXはSANにおけるストレージ保護機能を提供します。

SPOF (シングルポイント障害)

失敗するとクラスタ全体の障害をトリガしてしまう、クラスタのコンポーネント。

STONITH

「Shoot the other node in the head」の略です。動作異常のノードをシャットダウンすることでクラスタに問題を発生させないようにするフェンシングメカニズムを指しています。Pacemakerクラスタにおけるノードレベルフェンシングの実装は、STONITHです。このPacemakerにはフェンシングサブシステム`pacemaker-fenced`が内蔵されています。

アクティブ/アクティブ、アクティブ/パッシブ

サービスがノード上で実行される方法の概念。アクティブ/パッシブシナリオでは、1つ以上のサービスがアクティブノード上で実行され、パッシブノードはアクティブノードの失敗を待機します。アクティブ/アクティブでは、各ノードがアクティブであると同時にパッシブです。たとえば、一部のサービスは実行されていますが、それ以外のサービスは他のノードから引き継ぐことができます。DRBDのプライマリ/セカンダリとデュアルプライマリに類似しています。

アービトレータ

Geoクラスタ内の追加インスタンスで、サイト間にまたがるリソースのフェールオーバーなどの決定に関する合意の形成を手助けします。アービトレータは専用モードで1つまたは複数のブースインスタンスを実行する単一のマシンです。

クォーラム

クラスタでは、クラスタパーティションは、ノード(投票)の大多数を保有する場合、クォーラムを持つ(「定数に達している」と定義されます。クォーラムはただ1つのパーティションで識別されます。複数の切断されたパーティションまたはノードが処理を続行してデータおよびサービスが破損されないようにする、アルゴリズムの一部です(スプリットブレイン)。クォーラムはフェンシングの前提条件で、このためクォーラムは一意になります。

クラスタ

「ハイパフォーマンス」クラスタは、結果を早く出すためにアプリケーション負荷を共有するコンピュータ(実際または仮想のコンピュータ)のグループです。「高可用性」クラスタは、サービスの可用性を最大にすることを第一に設計されています。

クラスタスタック

クラスタを構成するソフトウェアテクノロジーとコンポーネントのアンサンブル。

クラスタパーティション

1つ以上のノードとその他のクラスタ間で通信が失敗した場合は、常にクラスタパーティションが発生します。クラスタのノードはパーティションに分割されますが、アクティブなままです。これらは同じパーティション内のノードのみと通信可能で、切り離されたノードは認識しません。つまり、他のパーティションのノードの損失は確認できないため、スプリットブレインシナリオが作成されます([スプリットブレイン](#)も参照)。

スイッチオーバー

クラスタ内の他のノードへの、予定されたオンデマンドのサービス移動。[フェールオーバー](#)を参照してください。

スプリットブレイン

クラスタノードが(ソフトウェアまたはハードウェア障害によって)互いに認識しない2つ以上のグループに分割される場合のシナリオです。STONITHによって、スプリットブレインがクラスタ全体に悪影響をおよぼさなくなります。「パーティションされたクラスタ」シナリオとも呼ばれます。

スプリットブレインという用語は、DRBDでも使用されますが、2つのノードに異なるデータが含まれることを意味します。

チケット

Geoクラスタで使用されるコンポーネント。チケットは指定のクラスタサイトの特定のリソースを実行する権利を付与します。チケットは1度に1つのサイトだけが所有できます。リソースを特定のチケットに依存させることができます。定義されたチケットがサイトで使用できる場合のみそれぞれのリソースが始動します。その逆に、チケットが削除されると、そのチケットに依存するリソースが自動的に停止します。

ノード

クラスタのメンバーで、ユーザには見えない(実際または仮想の)コンピュータ。

フェンシング

孤立または失敗したクラスタメンバーによる共有リソースへのアクセス防止の概念を示しています。フェンシングには、リソースレベルフェンシングおよびノードレベルフェンシングという、2つのクラスがあります。リソースレベルのフェンシングにより、特定のリソースへの排他的アクセスが保証されます。ノードレベルフェンシングでは、障害が発生したノードから共有リソースに完全にアクセスできなくなります。また、リソースから、ステータスが不明なノードを実行できなくなります。このことは通常、そのノードをリセットする、または電源オフにするというような、極端な手段で行われます。

フェールオーバー

リソースまたはノードが1台のマシンで失敗し、影響を受けるリソースが別のノードで起動されたときに発生します。

フェールオーバードメイン

ノード障害の発生時にクラスタサービスを実行することができる、指定されたクラスタノードのサブセット。

ブース

Geoクラスタのサイト間のフェールオーバープロセスを管理するインスタンス。その目的は、1つのサイトのみでマルチサイトリソースをアクティブにすることです。これは、サイトをダウンする必要がある場合、クラスタサイト間のフェールオーバードメインとして処理される、いわゆるチケットを使用することで可能になります。

メトロクラスタ

すべてのサイトがファイバチャネルで接続された、複数の建物またはデータセンターにわたってストレッチできる単一のクラスタ。ネットワークの遅延時間は通常は短くなります(約20マイルの距離で<5ms)。ストレージは頻繁にレプリケートされます(ミラーリングまたは同期レプリケーション)

ユニキャスト

ひとつのあて先ネットワークにメッセージを送信する技術Corosyncはマルチキャストとユニキャストの両方をサポートしています。Corosyncでは、ユニキャストはUDP-unicast (UDPU)として実装されます。

リソース

Pacemakerに認識されている、任意のタイプのサービスまたはアプリケーション。IPアドレス、ファイルシステム、データベースなどです。

「リソース」という用語は、DRBDでも使用されており、レプリケーション用の一般的な接続を使用しているブロックデバイスのセットの名前を表します。

ローカルクラスタ

1つのロケーション内の単一のクラスタ(たとえば、すべてのノードが1つのデータセンターにある)。ネットワークの遅延時間は無視できます。ストレージは通常、すべてのノードに同時にアクセスされます。

同時実行違反

クラスタ内の1つのノードだけで実行する必要があるリソースが、複数のノード上で実行されています。

既存のクラスタ

「既存のクラスタ」という用語は、1つ以上のノードで構成されるクラスタを指すものとして使用されます。既存のクラスタは、通信チャネルを定義する基本的なCorosync設定を持ちますが、必ずしもリソース設定を持つとは限りません。

負荷分散

複数のサーバを同じサービスに参加させて、同じ作業を行わせる機能。

障害

自然、人、ハードウェアのエラー、ソフトウェアのバグなどによって引き起こされる重要なインフラストラクチャの想定外の障害

障害復旧

障害復旧は、障害発生後、ビジネス機能を通常とおりの、安定した状態に修復するプロセスです。

障害復旧プラン

ITインフラストラクチャへの影響を最小限に抑えながら障害から復旧する戦略。

E GNU licenses

This appendix contains the GNU Free Documentation License version 1.2.

GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format

whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in

quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History"; Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections

as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <https://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.