SUSE

Deploying SUSE AI Factory with NVIDIA

Deploying SUSE AI Factory with NVIDIA

Deploying SUSE AI Factory with NVIDIA on a RKE2 Kubernetes cluster standardizes artificial intelligence (AI) platform management. It replaces manual integration with version-controlled, reproducible blueprints to streamline enterprise workloads.

Important

For multi-cluster deployments, install SUSE AI Factory Operator on the same cluster where the main Rancher Prime server runs.

Publication Date: 2026-11-12

https://documentation.suse.com

Contents

1 How do I prepare for the deployment? 1

1.1 Installation overview 1

1.2 Install SLES 16 2

1.3 Install RKE2 2

1.4 Install Rancher Prime 2

1.5 Install GPU Operator 3

1.6 Connect the *downstream* cluster to Rancher Prime 3

1.7 Assign GPU resources 3

1.8 Install SUSE Security 5

1.9 Install SUSE Observability 5

2 How do I deploy SUSE AI Factory with NVIDIA? 6

2.1 How do I install SUSE AI Factory with NVIDIA using the Helm CLI? 6

2.2 How do I install SUSE AI Factory with NVIDIA using Rancher? 7

2.3 Can I provide repository credentials during the SUSE AI Factory with NVIDIA installation? 8

2.4 How do I deploy Run:ai for GPU workload orchestration? 10

3 Are there any post-deployment steps? 22

3.1 How do I access images from SUSE Application Collection? 22

3.2 How do I access images from SUSE Registry? 24

3.3 How do I access images from NVIDIA AI Library? 26

3.4 How do I verify access to a chart repository? 28

3.5 How do I access a Git repository for application configuration? 29

3.6 How do I trust a private CA certificate for a registry? 32

3.7 How do I change or remove secret references in the settings? 33

3.8 How do I add custom chart repositories? 33

3.9 How do I add blueprint catalogs? 36

3.10 How do I authorize access to Git-backed Rancher Prime catalogs? 41

4 How do I upgrade SUSE AI Factory with NVIDIA? 43

5 How do I uninstall SUSE AI Factory with NVIDIA? 47

5.1 Back up SUSE AI Factory with NVIDIA resources and configuration (optional) 47

5.2 Uninstall using Helm 48

5.3 Uninstall using Rancher Prime 49

1 How do I prepare for the deployment?

Preparation tasks depend on how much underlying infrastructure is already deployed.

1.1 Installation overview

This overview summarizes the SUSE AI Factory with NVIDIA installation scenarios and shows which deployment path to follow.

The SUSE AI Factory with NVIDIA installation process differs according to the following scenarios:

- You have clean cluster nodes prepared without SLES 16 operating system.
- You have SLES 16 operating system and RKE2 Kubernetes distribution with SUSE Rancher Prime installed.
- Your setup is ready for installing AI applications and blueprints.

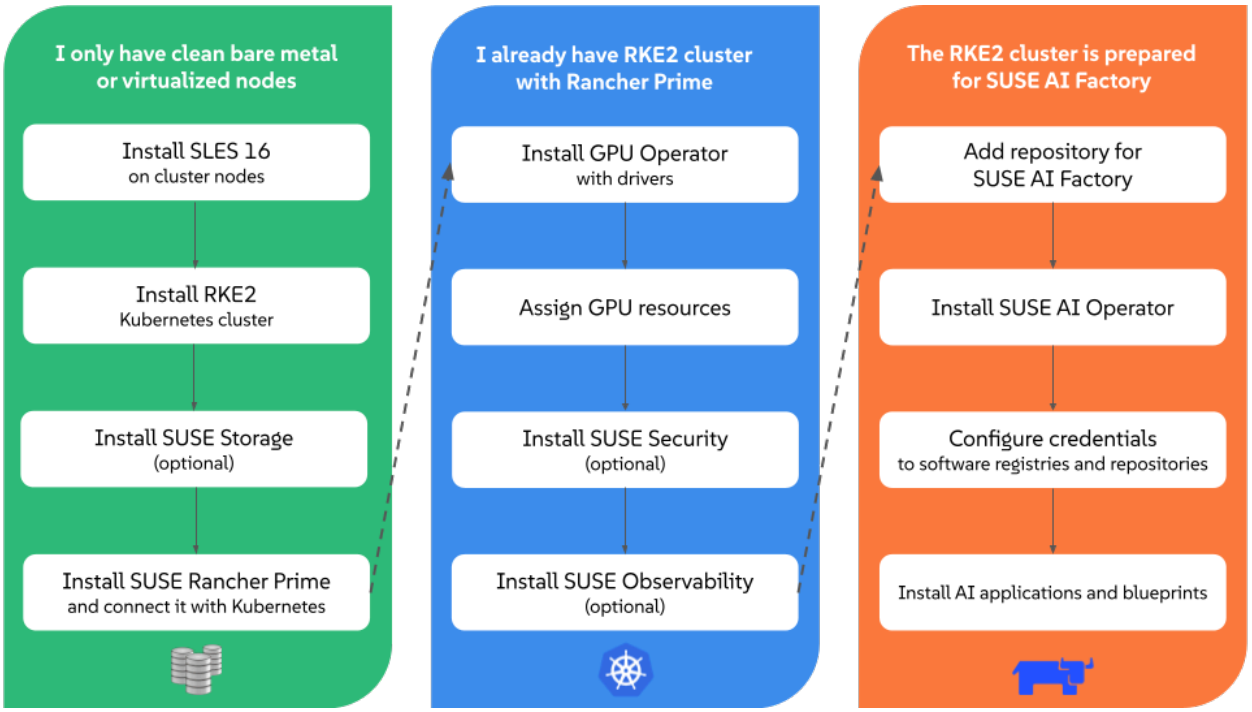

FIGURE 1.1: SUSE AI FACTORY WITH NVIDIA INSTALLATION PROCESS

1.2 Install SLES 16

Install and register SUSE Linux Enterprise Server 16 operating system on each cluster node. For supported installation methods, see:

- SUSE Linux Enterprise Server 16 Documentation (https://documentation.suse.com/sles/16.0/)
- SUSE Linux Enterprise Server 15.7 Deployment Guide (https://documentation.suse.com/sles/15-SP7/html/SLES-all/book-deployment.html)

1.3 Install RKE2

Important: Separate clusters for specific SUSE AI Factory with NVIDIA components

For production deployments, we strongly recommend deploying Rancher Prime, SUSE Observability, and workloads from the AI library to separate Kubernetes clusters.

Refer to RKE2 Installation Quickstart (https://documentation.suse.com/cloudnative/rke2/latest/en/install/quickstart.html) for details.

1.4 Install Rancher Prime

Refer to Rancher Prime Installation on Kubernetes Cluster (https://documentation.suse.com/cloudnative/rancher-manager/latest/en/installation-and-upgrade/install-rancher.html) for more details.

Important: Separate clusters for specific SUSE AI Factory with NVIDIA components

For production deployments, we strongly recommend deploying Rancher Prime, SUSE Observability, and workloads from the AI library to separate Kubernetes clusters.

1.5 Install GPU Operator

Install the GPU Operator on downstream clusters where GPU resources are available. Do not install it on the management cluster running SUSE Rancher Prime.

Note: Use third-party drivers

We recommend deploying the GPU Operator with third-party GPU drivers.

Refer to GPU Operator Installation (https://documentation.suse.com/cloudnative/rke2/latest/en/add-ons/gpu_operators.html) for more details.

1.6 Connect the *downstream* cluster to Rancher Prime

Refer to Registering Existing Clusters (https://documentation.suse.com/cloudnative/rancher-manager/latest/en/cluster-deployment/register-existing-clusters.html) for more details.

1.7 Assign GPU resources

Applications running on Kubernetes can request GPU resources when GPU acceleration is required.

After the GPU Operator is installed and configured, GPU resources become available to Kubernetes. Applications request these resources by defining resource requests and limits for their workloads.

Kubernetes schedules workloads that request GPU resources on nodes where the requested resources are available. Optionally, you can use node selection rules to restrict an application to specific nodes.

REQUIREMENTS

- RKE2 Kubernetes cluster with one or more nodes that provide supported GPU resources.
- A GPU Operator must be installed and configured to make GPU resources available to Kubernetes.
- Application deployments must support configuring Kubernetes resource requests and limits.

1.7.1 Requesting GPU resources

When GPU resources are available to Kubernetes, applications can request them by defining resource requests and limits. For NVIDIA GPUs, the GPU Operator typically makes the GPU resource available as `nvidia.com/gpu`.

For example, add the following configuration to the application's `values.yaml` file:

```
resources:
  requests:
    nvidia.com/gpu: 1
  limits:
    nvidia.com/gpu: 1
```

Specify the number of GPUs according to the requirements of the application and the available cluster resources.

When the workload is scheduled, Kubernetes schedules it on a node where the requested GPU resources are available and allocates the requested resources to the workload.

1.7.2 Optionally selecting specific GPU nodes

In some environments, you might want to restrict an application to specific GPU nodes. For example, this can be useful when a cluster contains nodes with different GPU models or when GPU nodes are reserved for particular workloads.

You can use Kubernetes labels and node selectors to control where the workload is scheduled.

For example, label a node:

```
> kubectl label node <GPU_NODE_NAME> accelerator=nvidia-gpu
```

Then configure the application to select nodes with that label:

```
nodeSelector:
  accelerator: nvidia-gpu
```

Node labels and selectors are optional. They are not required for Kubernetes to allocate GPU resources.

1.7.3 Verifying GPU assignment

The method used to verify GPU usage depends on the application.

First, verify that Kubernetes scheduled the workload successfully on a node that provides the requested GPU resources. You can inspect the workload and its assigned node by using `kubectl`. For example:

```
> kubectl describe pod <POD_NAME>
```

After the workload is running, use application-specific information to verify that the application detected and is using the assigned GPU. Depending on the application, this information might be available through logs, metrics, or the user interface.

For example, Ollama logs information about the available compute libraries and detected GPUs when GPU acceleration is enabled. The following example shows an Ollama log indicating that a compatible GPU was detected:

```
[...] source=gpu.go:204 msg="looking for compatible GPUs"
[...] source=types.go:105 msg="inference compute" id=GPU-... library=cuda compute=...
```

Refer to the documentation of the deployed application for application-specific instructions on verifying GPU detection and usage.

1.8 Install SUSE Security

Refer to SUSE Security Rancher Prime Deployment (https://documentation.suse.com/cloudnative/security/latest/en/rancher.html) for more details.

1.9 Install SUSE Observability

Refer to SUSE Observability Rancher Prime Deployment (https://documentation.suse.com/cloudnative/suse-observability/latest/en/k8s-suse-rancher-prime.html) for more details.

2 How do I deploy SUSE AI Factory with NVIDIA?

You can install SUSE AI Factory with NVIDIA on the Rancher Prime-enabled RKE2 Kubernetes cluster in the following ways:

- on the command line by using the Helm command
- using Rancher Prime UI

2.1 How do I install SUSE AI Factory with NVIDIA using the Helm CLI?

To install SUSE AI Factory with NVIDIA on the command line using Helm, proceed as follows.

1. Identify the operator chart version number that you want to install. The latest version is 2.4.0.
2. Review the `aif-operator` Helm chart values.

```
> helm show values oci://ghcr.io/suse/chart/aif-operator \
  --version 2.4.0
```

3. Install SUSE AI Factory Operator. The installation can take a few minutes.

```
> helm install aif-operator \
  oci://ghcr.io/suse/chart/aif-operator:2.4.0 \
  --namespace aif-operator \
  --create-namespace
```

4. Verify that the operator is correctly installed and related pods running. The output should be similar to the following:

```
> kubectl get pods -A | grep aif
aif-operator                aif-operator-7bf464987-mnrw5
cattle-ui-plugin-system     aif-ui-server-865c7fc84c-bmf7w
```

2.2 How do I install SUSE AI Factory with NVIDIA using Rancher?

To deploy SUSE AI Factory with NVIDIA through the graphical interface, add the OCI repository and install the operator chart. This procedure walks you through setting up the repository endpoint and configuring the installation in the Rancher Prime UI.

Important

You must have Rancher Prime Administrator privileges to perform this task.

1. **Add a new repository for SUSE AI Factory with NVIDIA.**
    a. From the main left panel, select *local* › *Apps* › *Repositories* and click *Create*.
    b. Select *OCI Repository* and enter your preferred repository name, for example, `suse-aif-operator`.
    c. Enter `oci://ghcr.io/suse/chart/aif-operator` as an OCI repository URL.
    d. Leave `None` for authentication type and confirm with *Create*.
2. **Install the `SUSE AI Operator` application.**
    a. Navigate to *Apps* › *Charts* and select the new `suse-aif-operator` filter in the left repository panel.
    b. Click *Refresh all repositories* and the `aif-operator` application tile appears.
    c. Click the application's tile and then the *Install this version* button.
    d. Select the `aif-operator` namespace and specify name for the workload.
    e. To customize detailed installation options, activate *Customize Helm options before install*. Confirm with *Next*.
    f. Review or update the installation chart and install the application with the *Install* button in the bottom right. You can watch the installation progress in the newly opened log window.
    g. Verify that the application is listed in *Apps* › *Installed Apps*.

2.3 Can I provide repository credentials during the SUSE AI Factory with NVIDIA installation?

To access applications and blueprints from the SUSE Application Collection, SUSE Registry and NVIDIA AI Librarys, you need to provide valid credentials. By default, you can configure the credentials after the SUSE AI Factory with NVIDIA installation is complete as described in *Chapter 3, Are there any post-deployment steps?*.

However, you can specify the credentials during the SUSE AI Factory with NVIDIA installation process as well. The result is identical—specifying credentials during the installation creates the secrets and configure the settings in Rancher Prime.

Warning

Storing credentials in a plain-text values YAML file poses a security risk. Always restrict the file permissions of your custom values files to prevent unauthorized local access. Avoid passing passwords directly in command-line arguments to prevent them from being recorded in your shell history.

For production environments, we recommend using secure secrets-management tools, such as the `helm-secrets` plug-in with Mozilla SOPS, or the External Secrets Operator.

Tip

User names for specific registries are preconfigured:

- SUSE Registry uses `regcode`.
- NVIDIA AI Library uses `$oauthtoken`.

2.3.1 Providing registry credentials during Helm CLI installation

1. Download and display the default values for the SUSE AI Factory with NVIDIA installation.

```
> helm show values oci://ghcr.io/suse/chart/aif-operator --version 2.4.0
```

2. Create the `aif-values.yaml` file and restrict its permissions so that only you can read and write to it:

```
> touch aif-values.yaml && chmod 600 aif-values.yaml
```

3. Copy the section that deals with registry credentials into the file, for example:

```
credentials:
  # credential for oci://dp.apps.rancher.io/charts
  applicationCollection:
    username: ""
    password: ""
  # credential to pull NGC artifacts
  # e.g. https://helm.ngc.nvidia.com/nvidia
  nvidia:
    username: "$oauthtoken"
    password: ""
  # credential for oci://registry.suse.com/ai/charts
  suseRegistry:
    username: "regcode"
    password: ""
```

4. Provide the credentials to individual registries in the related `username:` and `password:` fields.

5. Save the file and install SUSE AI Factory with NVIDIA as described in *Section 2.1, "How do I install SUSE AI Factory with NVIDIA using the Helm CLI?"* by specifying the custom values file, for example:

```
> helm install aif-operator \
  oci://ghcr.io/suse/chart/aif-operator:2.4.0 \
  --namespace aif-operator \
  --create-namespace \
  -f aif-values.yaml
```

2.3.2 Providing registry credentials during installation in SUSE Rancher Prime

Important

You must have Rancher Prime Administrator privileges to perform this task.

1. Start the SUSE AI Factory with NVIDIA installation as described in *Section 2.2, "How do I install SUSE AI Factory with NVIDIA using Rancher?"*.

2. On the installation screen, activate *Customize Helm options before install*.

3. Update the part of the installation chart that deals with registry credentials by providing credentials to individual registries.

```
credentials:
  # credential for oci://dp.apps.rancher.io/charts
  applicationCollection:
    username: ""
    password: ""
  # credential to pull NGC artifacts
  # e.g. https://helm.ngc.nvidia.com/nvidia
  nvidia:
    username: "$oauthtoken"
    password: ""
  # credential for oci://registry.suse.com/ai/charts
  suseRegistry:
    username: "regcode"
    password: ""
```

4. Continue the installation by clicking the *Install* button in the bottom right.

2.4 How do I deploy Run:ai for GPU workload orchestration?

Run:ai (Run:ai official documentation (https://docs.nvidia.com/run-ai/self-hosted/documentation/getting-started/overview)) is a graphics processing unit (GPU) orchestration and optimization platform designed to maximize compute resource use for artificial intelligence (AI) workloads in a Kubernetes environment. Installing Run:ai via SUSE AI Factory enables dynamic GPU resource allocation, efficient workload scheduling, and centralized cluster management.

2.4.1 Prerequisites

- **DNS resolution and Ingress controller**: The Run:ai Control Plane and cluster URLs must be fully qualified domain names (FQDNs) resolvable via public or private DNS. IP addresses are not supported. In a Kubernetes environment, you must configure a supported Ingress controller

(such as traefik) and DNS records pointing to the Ingress IP address or host name. For more information about Ingress controller requirements, see the Run:ai Kubernetes Ingress Controller requirements (https://run-ai-docs.nvidia.com/self-hosted/getting-started/installation/install-using-helm/cp-system-requirements#kubernetes-ingress-controller).

- **TLS certificates**: A configured `cert-manager` instance with a valid certificate authority (CA) or issuer is required. While Run:ai supports local or custom CAs, using Let's Encrypt with an HTTP-01 or DNS-01 challenge solver simplifies certificate management.
- **Prometheus Operator**: Custom Resource Definitions (CRDs) and monitoring components provided by the Prometheus Operator are required for the Run:ai Cluster installation. You can deploy the Prometheus Operator from SUSE Application Collection (https://apps.rancher.io/applications/prometheus-operator) (recommended) or install the upstream kube-prometheus-stack (https://github.com/prometheus-community/helm-charts). The Control Plane does not require this stack.

Important: Limitations

Due to the complexity of Run:ai, blueprints currently do not support deploying either the Control Plane or the Cluster. Deploy them from the *SUSE AI Factory* › *Apps* catalog instead.

2.4.1.1 Configure DNS resolution for Ingress

To allow external access to the Run:ai control plane, you must route your FQDNs to the Kubernetes cluster's Ingress controller. RKE2 deploys the traefik Ingress Controller by default. For detailed information on supported Ingress controllers, see the Run:ai Kubernetes Ingress Controller requirements (https://run-ai-docs.nvidia.com/self-hosted/getting-started/installation/install-using-helm/cp-system-requirements#kubernetes-ingress-controller).

1. Retrieve the external IP address or host name of the Ingress controller service.

```
> kubectl get service -n kube-system rke2-traefik
```

2. In your DNS provider's management console, create a DNS record (A or CNAME) pointing your FQDN (for example, `runai-control-plane.examplecloudev.com`) to the retrieved IP address or host name.

2.4.1.2 Install cert-manager (optional)

If cert-manager is not yet installed, install it from SUSE Application Collection using the following steps:

1. Create a manifest file named `cert-manager-deployment.yaml` with the following content. Adjust the credentials to match your environment.

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: cert-manager
---
apiVersion: v1
data:
  .dockerconfigjson: "<YOUR_ENCODED_APPCO_CREDENTIAL_HERE>"
kind: Secret
metadata:
  name: application-collection
  namespace: cert-manager
type: kubernetes.io/dockerconfigjson
---
apiVersion: v1
data:
  .dockerconfigjson: "<YOUR_ENCODED_APPCO_CREDENTIAL_HERE>"
kind: Secret
metadata:
  name: application-collection
  namespace: kube-system
type: kubernetes.io/dockerconfigjson
---
apiVersion: helm.cattle.io/v1
kind: HelmChart
metadata:
  name: cert-manager
  namespace: kube-system
spec:
  chart: oci://dp.apps.rancher.io/charts/cert-manager
  version: "1.20.0"
  targetNamespace: cert-manager
  dockerRegistrySecret:
    name: application-collection
  valuesContent: |-
    global:
      imagePullSecrets:
        - application-collection
    crds:
```

```
      enabled: true
```

2. Apply the manifest file to install cert-manager.

```
> kubectl apply -f cert-manager-deployment.yaml
```

2.4.1.3 Create a ClusterIssuer

Create a ClusterIssuer to issue TLS certificates for the Run:ai endpoints. The following example configures Let's Encrypt using an HTTP-01 challenge solver with traefik Ingress.

1. Create a manifest file named `cluster-issuer.yaml` with the following content. Adjust the `email` field to match your email address.

```
---
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: suse-ai-cluster-issuer
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: "tux@example.com"
    privateKeySecretRef:
      name: suse-ai-letsencrypt-key
    solvers:
    - http01:
        ingress:
          ingressClassName: traefik
```

Note

Alternatively, you can use a DNS-01 challenge solver (for example, with Cloudflare or AWS Route 53) by supplying your DNS provider credentials, or use an internal enterprise Certificate Authority.

2. Apply the manifest file to create the ClusterIssuer.

```
> kubectl apply -f cluster-issuer.yaml
```

2.4.2 Install Run:ai Control Plane

Determine a unique fully qualified domain name (FQDN) from your domain for the Run:ai Control Plane UI and API endpoint. For example, if your domain is `exampleclouddev.com`, use an FQDN such as `runai-control-plane.exampleclouddev.com`.

Important

Because the Run:ai Helm chart hardcodes the service URLs, both the Helm release name and the namespace for the Run:ai Control Plane must be `runai-backend`.

2.4.2.1 Create the Control Plane endpoint certificate

1. Create a manifest file named `runai-control-plane-endpoint-cert.yaml` with the following content. Replace the `commonName` and `dnsNames` fields with your own Run:ai endpoint URL.

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: runai-backend
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: runai-tls
  namespace: runai-backend
spec:
  commonName: "runai-control-plane.exampleclouddev.com"
  secretName: runai-tls-cert
  privateKey:
    rotationPolicy: Always
  issuerRef:
    name: suse-ai-cluster-issuer
    kind: ClusterIssuer
  dnsNames:
  - "runai-control-plane.exampleclouddev.com"
```

2. Apply the manifest file to create the certificate.

```
> kubectl apply -f runai-control-plane-endpoint-cert.yaml
```

2.4.2.2 Create the `runai-reg-creds` secret

Run:ai images are hosted in the secure `nvcr.io` registry. For SUSE AI Factory with NVIDIA with NVIDIA, the required NVIDIA registry service key is already provided in SUSE Customer Center (https://scc.suse.com/) under your subscription details, so no separate NVIDIA AI Enterprise license purchase is required.

```
> kubectl create secret docker-registry runai-reg-creds \
  --docker-server=nvcr.io \
  --docker-username='$oauthtoken' \
  --docker-password="<SERVICE_KEY>" \
  --namespace=runai-backend
```

2.4.2.3 Install Run:ai Control Plane

1. In the left navigation menu, select *SUSE AI Factory* › *Apps*.
2. Select *NVIDIA AI Library*, search for `run:ai`, and select the application tile.

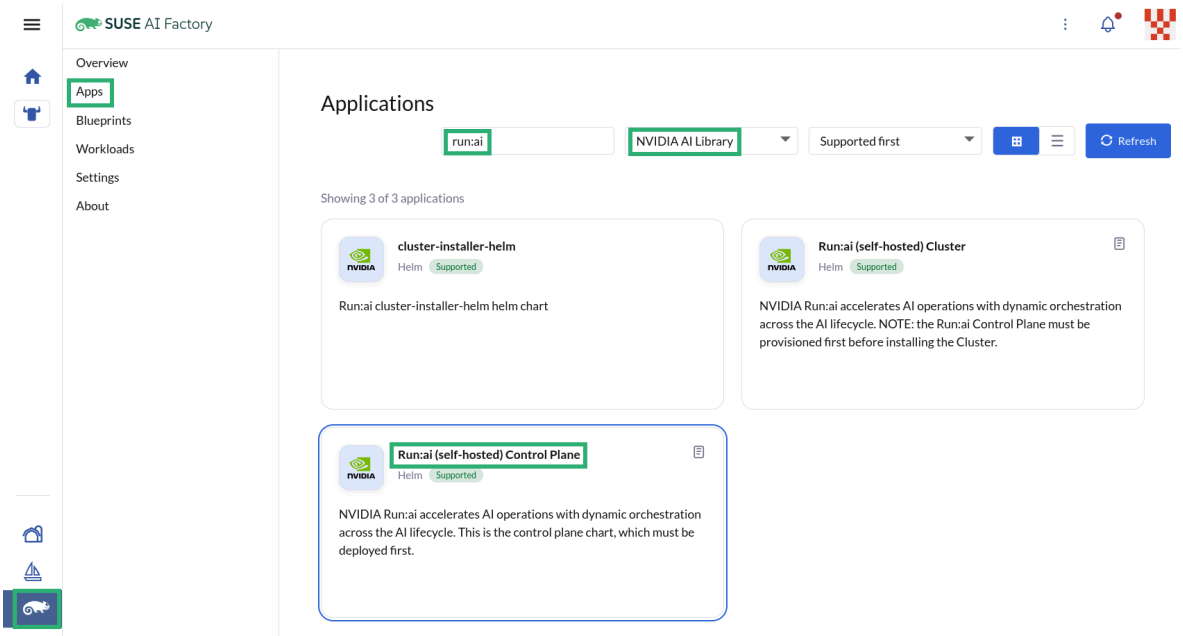

FIGURE 2.1: RUN:AI CONTROL PLANE APPLICATION

3. On the `Basic Information` screen, set both the `Instance Name` and `Namespace` fields to `runai-backend`.
4. On the `Target Cluster` screen, set the `Deployment Type` to `Helm`. Do not use the default `Fleet Bundle` option.

5. On the `Configuration` screen, override the default Helm values with your custom configuration, for example:

```
global:
  domain: "runai-control-plane.exampleclouddev.com"
  ingress:
    ingressClass: traefik
    tlsSecretName: runai-tls-cert
tenantsManager:
  config:
    adminUsername: "tux@example.com"
    adminPassword: "<PASSWORD>"
```

6. Review the configuration and select *Install*. The installation takes several minutes to complete.

7. Once the deployment is complete, open a web browser, navigate to the Run:ai UI endpoint, and log in with the credentials specified in your Helm values. For example, navigate to `https://runai-control-plane.exampleclouddev.com` and log in with the username `tux@example.com` and your configured password.

8. On your first login, accept the NVIDIA user agreement and follow the setup wizard to create your first cluster.

9. The Run:ai Cluster can be installed on the same Kubernetes cluster as the Control Plane. If you choose this option, the Cluster URL is identical to the Control Plane URL. Copy and save the installation instructions displayed on the screen.

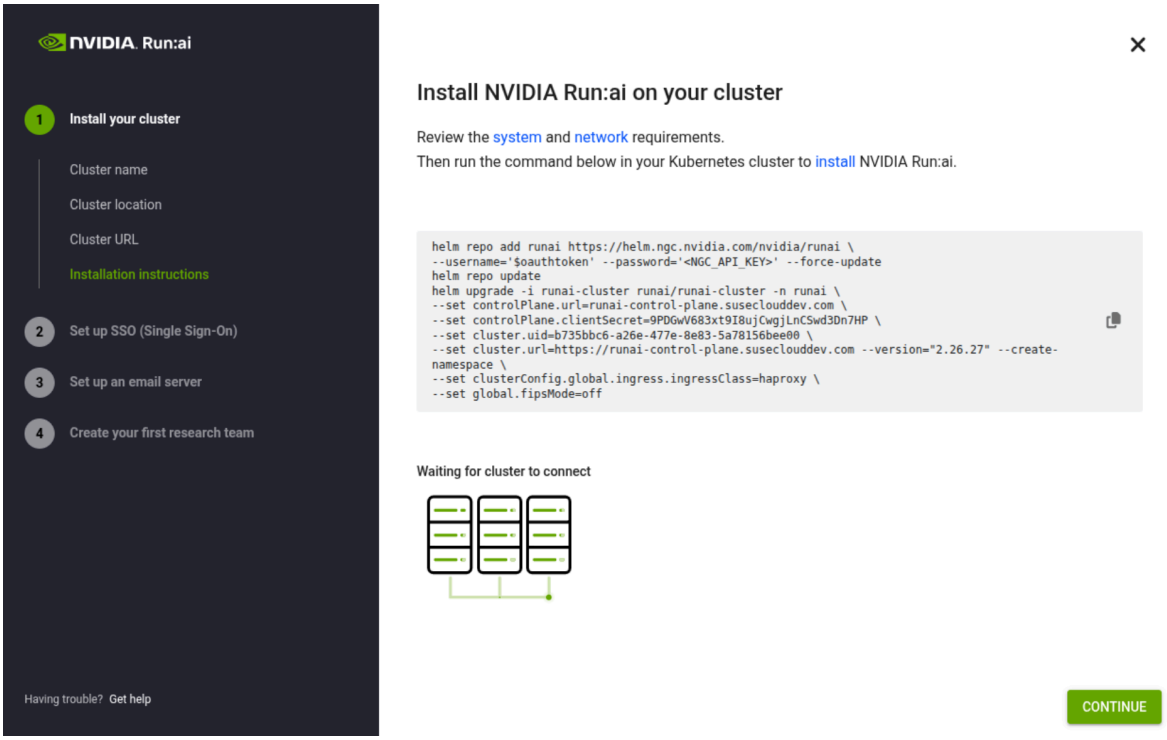

FIGURE 2.2: RUN:AI CLUSTER APPLICATION

Important

Do not run the installation commands displayed in the wizard. The Run:ai Cluster must be deployed using SUSE AI Factory instead. Record the following values from the wizard setup screen:

- Control Plane URL
- Control Plane Client Secret
- Cluster UID
- Cluster URL

Other wizard steps, such as setting up SAML (Security Assertion Markup Language) authentication or creating teams, are optional and can be configured later.

2.4.3 Install Run:ai Cluster

2.4.3.1 Prerequisites

- Before installing the Run:ai Cluster, you must deploy the Run:ai Control Plane (see *Section 2.4.2.3, "Install Run:ai Control Plane"*) and create your cluster.
- Obtain the following information from the Control Plane:
  - Control Plane URL
  - Control Plane Client Secret
  - Cluster UID
  - Cluster URL

2.4.3.2 Create the `runai` namespace

The Run:ai Cluster must be installed in the `runai` namespace.

```
> kubectl create namespace runai
```

2.4.3.3 Create the `runai-reg-creds` secret

Run:ai images are hosted in the secure `nvcr.io` registry. Create the `runai-reg-creds` secret in the `runai` namespace using the NVIDIA service key obtained from SUSE Customer Center to pull the images.

```
> kubectl create secret docker-registry runai-reg-creds \
  --docker-server=nvcr.io \
  --docker-username='$oauthtoken' \
  --docker-password="<SERVICE_KEY>" \
  --namespace=runai
```

2.4.3.4 Install Prometheus Operator

The Run:ai Cluster requires Custom Resource Definitions (CRDs) and monitoring components provided by the Prometheus Operator. SUSE recommends installing the Prometheus Operator from SUSE Application Collection (https://apps.rancher.io/applications/prometheus-operator). Alternatively, you can deploy the upstream Prometheus community stack.

1. Create a file named `prometheus-operator.yaml` with the following content. Adjust the credentials to match your environment.

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: monitoring
---
*apiVersion*: v1
data:
  .dockerconfigjson: "<YOUR_ENCODED_APPCO_CREDENTIAL_HERE>"
kind: Secret
metadata:
  name: application-collection
  namespace: monitoring
type: kubernetes.io/dockerconfigjson
---
apiVersion: v1
data:
  .dockerconfigjson: "<YOUR_ENCODED_APPCO_CREDENTIAL_HERE>"
kind: Secret
metadata:
  name: application-collection
  namespace: kube-system
type: kubernetes.io/dockerconfigjson
---
apiVersion: helm.cattle.io/v1
kind: HelmChart
metadata:
  name: prometheus-operator
  namespace: kube-system
spec:
  chart: oci://dp.apps.rancher.io/charts/prometheus-operator
  targetNamespace: monitoring
  dockerRegistrySecret:
    name: application-collection
  valuesContent: |-
    global:
```

```
      imagePullSecrets:
        - application-collection
```

2. Apply the manifest file to install the Prometheus Operator.

```
> kubectl apply -f prometheus-operator.yaml
```

Tip

If you prefer to deploy the upstream `kube-prometheus-stack` instead of the SUSE Application Collection chart, you can use the following `HelmChart` manifest:

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: monitoring
---
apiVersion: helm.cattle.io/v1
kind: HelmChart
metadata:
  name: prometheus
  namespace: kube-system
spec:
  repo: https://prometheus-community.github.io/helm-charts
  chart: kube-prometheus-stack
  version: "87.19.1"
  targetNamespace: monitoring
  valuesContent: |-
    grafana:
      enabled: false
```

2.4.3.5 Install Run:ai cluster

1. In the left navigation menu, select *SUSE AI Factory* › *Apps*.
2. Select *NVIDIA AI Library*, search for `run:ai`, and select the `Run:ai (self-hosted) Cluster` application tile.
3. On the `Basic Information` screen, set the `Namespace` field to `runai-backend`. You can leave the `Instance Name` field with the default value `runai-cluster`.

4. On the `Target Cluster` screen, set the `Deployment Type` to `Helm`. Do not use the default `Fleet Bundle` option.

5. On the `Configuration` screen, override the default Helm values with your custom configuration. Use the values for `controlPlane.url`, `controlPlane.clientSecret`, `cluster.uid`, and `cluster.url` obtained in the previous step.

```
controlPlane:
  url: "runai-control-plane.exampleclouddev.com"
  clientSecret: "<SECRET_KEY>"
cluster:
  uid: "b735bbc6-a26e-477e-8e83-5a78156bee00"
  url: "https://runai-control-plane.exampleclouddev.com"
clusterConfig:
  global:
    ingress:
      ingressClass: traefik
```

6. Review the configuration and select *Install*.

2.4.4 Verify installation

After the Run:ai Cluster installation is complete, verify that the cluster is connected to the Run:ai Control Plane:

1. In a web browser, log in to the Run:ai Control Plane UI.

2. In the left navigation menu, select *Resources* › *Clusters*.

3. In the **Clusters** table, verify that your cluster is listed and its status is displayed as **Connected**.
   For more information about cluster status indicators and cluster management, see the Run:ai Clusters documentation (https://run-ai-docs.nvidia.com/self-hosted/infrastructure-setup/procedures/clusters#clusters-table).

3 Are there any post-deployment steps?

To run AI applications and blueprints, you must configure access credentials for multiple network registries. This guide walks you through acquiring user tokens, creating Kubernetes secrets in the local cluster, and mapping those secrets in SUSE AI Factory.

Important

If you already provided credentials for specific registries during the SUSE AI Factory with NVIDIA installation as described in *Section 2.3, "Can I provide repository credentials during the SUSE AI Factory with NVIDIA installation?"*, you can skip adding those credentials as described in this section.

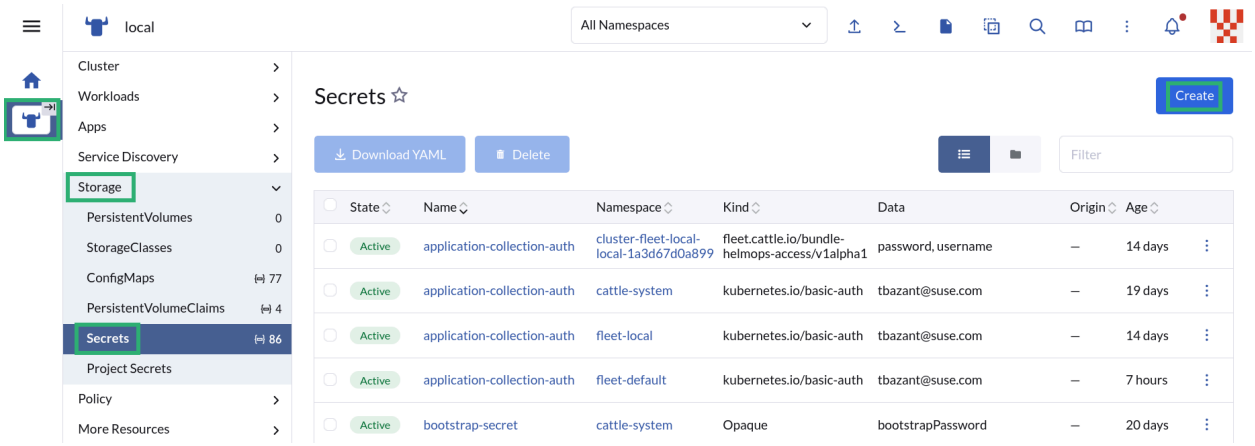

FIGURE 3.1: RANCHER PRIME SECRETS

3.1 How do I access images from SUSE Application Collection?

1. Visit SUSE Application Collection (https://apps.rancher.io/), sign in, create and save the user access token as described in the official documentation (https://docs.apps.rancher.io/get-started/authentication/).
2. Add the SUSE Application Collection credentials to the Rancher Prime *local* cluster.
   a. From the main left panel, select *local* › *Storage* › *Secrets*.
   b. Select *Create* in the top right and select *Opaque*.

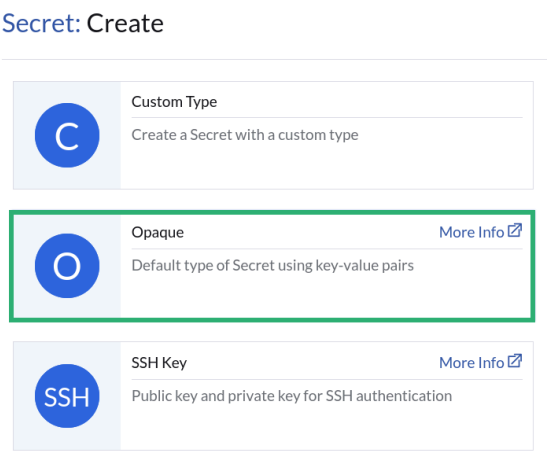

FIGURE 3.2: RANCHER PRIME SECRETS TYPES

c. Select the *aif-operator* namespace and enter your preferred secret name (referenced as `<APPCO_SECRETS_NAME>` in command line tasks) to identify the resource.

d. In the *Data* tab, add the key `user` with the value of your SUSE Application Collection username and `token` with your SUSE Application Collection token.

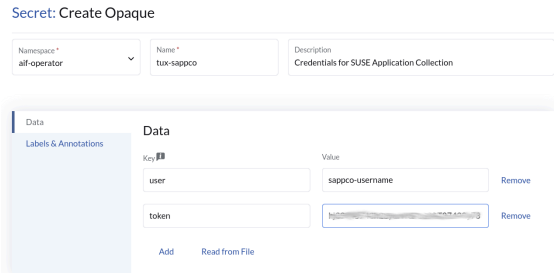

FIGURE 3.3: SUSE APPLICATION COLLECTION SECRETS

e. Confirm by selecting *Create* in the bottom right corner.

Tip: Creating the secrets on the command line

```
> kubectl create secret generic <APPCO_SECRETS_NAME> \
  --from-literal=user=<APPCO_USERNAME> \
  --from-literal=token=<APPCO_USER_TOKEN> \
  -n aif-operator
```

3. Assign SUSE Application Collection secrets to SUSE AI Factory.

a. From the main left panel, select *SUSE AI Factory* › *Settings*.

b. In the `SUSE Application Collection` section, select the secret name corresponding to your `<APPCO_SECRETS_NAME>` from the `Secret name` drop-down list.

c. Select `user` for *Username Secret* › *Secret Key* and `token` for *Access Token Secret* › *Secret Key*.

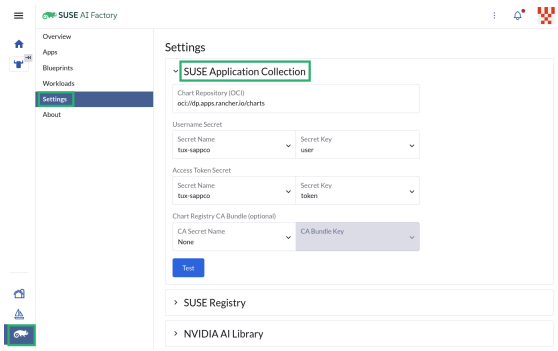

FIGURE 3.4: SUSE APPLICATION COLLECTION SECRETS

d. (*Optional*) Select *Test* to validate the credentials before saving, as described in *Section 3.4, "How do I verify access to a chart repository?"*.

e. Confirm by selecting *Apply* in the bottom right corner.

4. Navigate to *SUSE AI Factory* › *Apps* and select *SUSE AI Library* from the drop-down list in the top center of the screen.

5. Select the *Refresh* button in the top right. The list of available applications from the SUSE Application Collection appears.

3.2 How do I access images from SUSE Registry?

1. Visit SUSE Customer Center (https://scc.suse.com/), select the organization with an active SUSE AI Factory with NVIDIA subscription from the left sidebar, and save the registration code for the SUSE AI Factory with NVIDIA subscription.

2. Add the SUSE Registry credentials to the Rancher Prime *local* cluster.

a. From the main left panel, select *local* › *Storage* › *Secrets*.

b. Select *Create* in the top right and select *Opaque*.

c. Select the *aif-operator* namespace and enter your preferred secret name (referenced as `<REGISTRY_SECRETS_NAME>` in command line tasks) to identify the resource.

d. In the *Data* tab, add the key `user` with the value `regcode` and `token` with your SUSE Registry token.

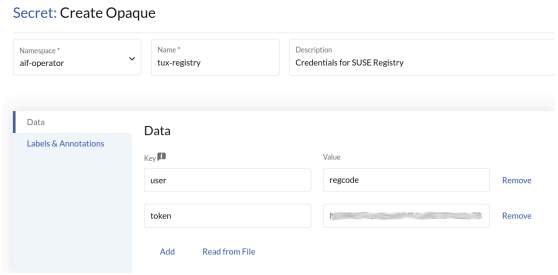

FIGURE 3.5: SUSE REGISTRY SECRETS

e. Confirm by selecting *Create* in the bottom right corner.

Tip: Creating the secrets on the command line

```
> kubectl create secret generic <REGISTRY_SECRETS_NAME> \
  --from-literal=user=regcode \
  --from-literal=token=<REGISTRY_USER_TOKEN> \
  -n aif-operator
```

3. Assign SUSE Registry secrets to SUSE AI Factory.

a. From the main left panel, select *SUSE AI Factory* › *Settings*.

b. In the `SUSE Registry` section, select the secret name corresponding to your `<REGISTRY_SECRETS_NAME>` from the `Secret name` drop-down list.

c. Select `user` for *Username Secret* › *Secret Key* and `token` for *Access Token Secret* › *Secret Key*.

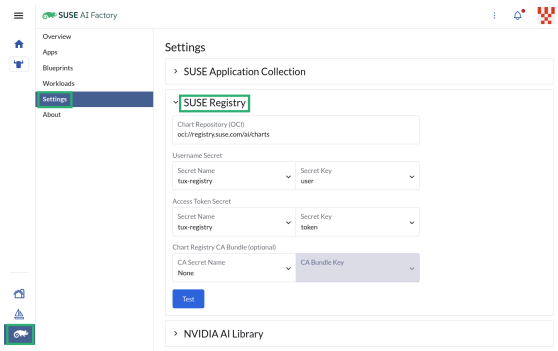

FIGURE 3.6: ASSIGN SUSE REGISTRY SECRETS

d. (*Optional*) Select *Test* to validate the credentials before saving, as described in *Section 3.4, "How do I verify access to a chart repository?"*.

e. Confirm by selecting *Apply* in the bottom right corner.

4. Navigate to *SUSE AI Factory* › *Apps* and select *SUSE AI Library* from the drop-down list in the top center of the screen.

5. Select the *Refresh* button in the top right. The list of available applications from the SUSE Registry appears.

3.3 How do I access images from NVIDIA AI Library?

1. Visit SUSE Customer Center and save the NVIDIA registration code for the SUSE AI Factory with NVIDIA subscription.

2. Add the NVIDIA registry credentials to the Rancher Prime *local* cluster.

   a. From the main left panel, select *local* › *Storage* › *Secrets*.

   b. Select *Create* in the top right and select *Opaque*.

   c. Select the *aif-operator* namespace and enter your preferred secret name (referenced as `<NVIDIA_SECRETS_NAME>` in command line tasks) to identify the resource.

   d. In the *Data* tab, add the key `user` with the value `$oauthtoken` and `token` with your NVIDIA registry token.

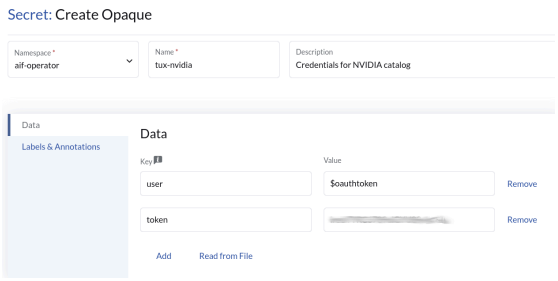

FIGURE 3.7: NVIDIA SECRETS

e. Confirm by selecting *Create* in the bottom right corner.

Tip: Creating NVIDIA secrets on the command line

```
> kubectl create secret generic <NVIDIA_SECRETS_NAME> \
  --from-literal=user=$oauthtoken \
  --from-literal=token=<NVIDIA_USER_TOKEN> \
  -n aif-operator
```

3. Assign NVIDIA registry secrets to SUSE AI Factory.

   a. From the main left panel, select *SUSE AI Factory* › *Settings*.

   b. In the `NVIDIA AI Library` section, select the secret name corresponding to your `<NVIDIA_SECRETS_NAME>` from the `Secret name` drop-down list.

   c. Select `user` for *Username Secret* › *Secret Key* and `token` for *Access Token Secret* › *Secret Key*.

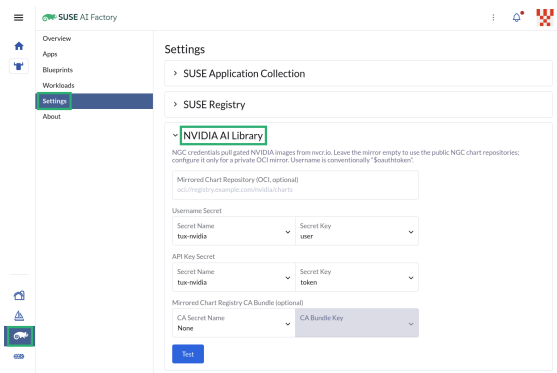

FIGURE 3.8: ASSIGN NVIDIA SECRETS

d. (*Optional*) Select *Test* to validate the credentials before saving, as described in *Section 3.4, "How do I verify access to a chart repository?"*.

e. Confirm by selecting *Apply* in the bottom right corner.

4. Navigate to *SUSE AI Factory* › *Apps* and select *NVIDIA AI Library* from the drop-down list in the top center of the screen.

5. Select the *Refresh* button in the top right. The list of available applications from the NVIDIA registry appears.

3.4 How do I verify access to a chart repository?

A successful registry login does not guarantee that you can read the charts in the repository, or that Rancher Prime has downloaded the repository index. Selecting *Test* in the `SUSE Application Collection`, `SUSE Registry` and `NVIDIA AI Library` sections therefore reports three separate results in the `Verify repository access` panel:

Registry connection (current form)

Whether the registry responds to the endpoint and credentials entered in the form.

Chart access (current form)

Whether a sample chart can be read with the endpoint, credentials and CA certificate entered in the form. The result names the chart and version that were tested, and explains failures such as denied access, missing content, an invalid index, TLS errors or timeouts.

Chart repositories (saved settings)

Whether the Rancher Prime `ClusterRepo` resources of the saved configuration have downloaded their index. This result is based on the saved settings, not on unsaved changes in the form.

1. From the main left panel, select *SUSE AI Factory* › *Settings*.

2. (*Optional*) If your private mirror does not contain the default sample chart, enter a chart that is available in your repository in the `Chart to test` field.

3. Select *Test* and review the results.

4. If you changed the form, select *Apply* and then *Test* again to verify the saved configuration in Rancher Prime.
5. If a repository index failed to download, for example, because you corrected the credentials or the mirror content, select *Refresh* next to the saved repository. This asks Rancher Prime to download the index again. Select *Test* again to check the result.

Note: Scope of the test

Test is read-only. It does not save your settings and does not refresh repositories. Only the listed chart and version in each repository are checked. The test does not verify all charts, full chart downloads or application container images.

An NVIDIA GPU Cloud (NGC) key may have access to certain NVIDIA team repositories but not to others. An unavailable optional team repository does not mean that the key is invalid for every NVIDIA application.

When a managed chart repository is unavailable, the `Overview` page lists it with links to **Settings** and to the repository in Rancher Prime. Applications from that repository remain visible on the `Apps` page, but you cannot install them until the repository is available again.

3.5 How do I access a Git repository for application configuration?

SUSE AI Factory with NVIDIA uses the `blueprints/` path for administrator-managed `Blueprint` resources and the `workloads/` path for GitOps resources that SUSE AI Factory with NVIDIA writes.

Important: Prepare the Git repository

Before you apply the repository URL, create both paths on the selected branch with content that Fleet can read. You can initialize each path with a `fleet.yaml` file. The file is optional when the path already contains deployable manifests.

Applying a non-empty repository URL creates the Fleet `GitRepo` resource and starts reconciliation immediately. For GitOps output, the selected credential must allow SUSE AI Factory with NVIDIA to write to the repository.

1. Add your Git Personal Access Token (PAT) to the Rancher Prime *local* cluster.

a. From the main left panel, select *local* › *Storage* › *Secrets*.

b. Select *Create* in the top right and select *Opaque*.

c. Select the *aif-operator* namespace and enter your preferred secret name (referenced as `<GIT_SECRETS_NAME>` in command line tasks) to identify the resource.

d. In the *Data* tab, add the key `token` and paste your PAT as a value.

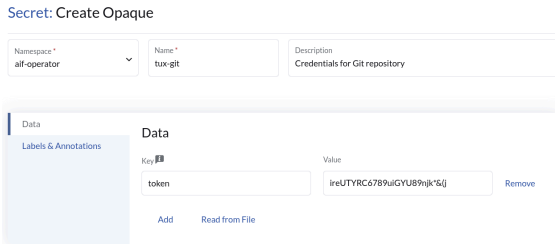

FIGURE 3.9: GIT SECRETS

e. Confirm by selecting *Create* in the bottom right corner.

Tip: Creating Git secrets on the command line

```
> kubectl create secret generic <GIT_SECRETS_NAME> \
  --from-literal=token=<GIT_PAT_TOKEN> \
  -n aif-operator
```

2. Add the Git PAT to SUSE AI Factory.

a. From the main left panel, select *SUSE AI Factory* › *Settings*.

b. In the `Fleet/GitOps` section, enter the Git repository URL and specify which branch to use.

c. If the Git server requires an account name, enter it in the `HTTPS Username (optional)` field.

d. Select the PAT secret name corresponding to your `<GIT_SECRETS_NAME>` from the `Credential Secret` drop-down list.

e. Select `token` for menu:Secret Key.

f. If the Git server uses a certificate issued by a private CA, select the secret and key that hold the PEM-encoded CA certificate.

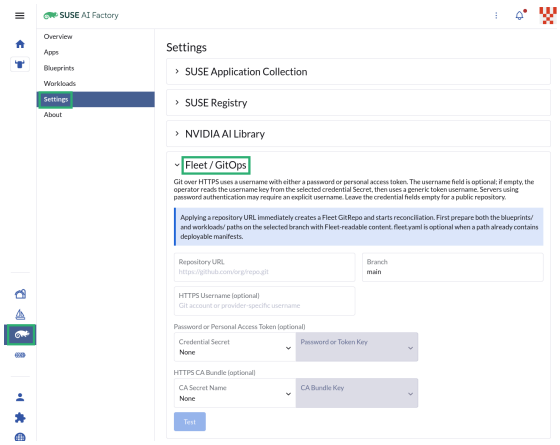

FIGURE 3.10: ASSIGN GIT SECRETS

g. (*Optional*) Select *Test* to validate the credentials before saving.

h. Confirm by selecting *Apply* in the bottom-right corner.

Note: Git authentication

SUSE AI Factory with NVIDIA uses HTTP Basic authentication for private HTTPS Git repositories. The selected password or personal access token is sent as the HTTP Basic authentication password.

When you select a credential, the user name is optional and resolves in the following order:

1. The value in the `HTTPS Username (optional)` field
2. The `username` key in the selected credential Secret
3. The value `token`

Git servers that validate the account name require an explicit user name in the field or Secret. Leave both the user name and credential fields empty for an anonymous repository.

Configurations created before SUSE AI Factory 2.2.0 that use the deprecated authentication type setting continue to work.

Git over SSH is not supported.

3.6 How do I trust a private CA certificate for a registry?

If you host SUSE Application Collection, SUSE Registry or NVIDIA artifacts in a local registry that uses a certificate issued by your organization's CA, provide that CA certificate to SUSE AI Factory instead of disabling TLS verification. SUSE AI Factory with NVIDIA propagates the certificate to Rancher Prime and Fleet so that chart repositories and workload deployments trust the registry.

1. Add the CA certificate to the Rancher Prime *local* cluster.
    a. From the main left panel, select *local* › *Storage* › *Secrets*.
    b. Select *Create* in the top right and select *Opaque*.
    c. Select the *aif-operator* namespace and enter your preferred secret name (referenced as `<REGISTRY_CA_SECRET_NAME>` in command line tasks).
    d. In the *Data* tab, add a key that holds the PEM-encoded certificate, for example, `ca.crt`.

Tip: Creating the CA secret on the command line

```
> kubectl create secret generic <REGISTRY_CA_SECRET_NAME> \
  --from-file=ca.crt=<PATH_TO_CA_CERTIFICATE> \
  -n aif-operator
```

2. Assign the CA certificate to the registry.
    a. From the main left panel, select *SUSE AI Factory* › *Settings*.
    b. In the `SUSE Application Collection`, `SUSE Registry` or `NVIDIA AI Library` section, select the secret name corresponding to your `<REGISTRY_CA_SECRET_NAME>` and the key that holds the certificate.
    c. Confirm by selecting *Apply* in the bottom-right corner.

Note: Certificate rotation

When you replace the certificate or the credentials of a registry, SUSE AI Factory with NVIDIA refreshes the corresponding chart repositories automatically.

3.7 How do I change or remove secret references in the settings?

Each credential or CA certificate field on the `Settings` page references a key in a secret in the `aif-operator` namespace. A reference is valid only when both the secret and the key are selected.

- **To change a reference**, select another secret and key, and confirm with *Apply*.
- **To remove a reference**, set the secret to `None` and confirm with *Apply*. This also removes references that were set by the `credentials` Helm values of the `aif-operator` chart or by the `rancherCatalog.token` and `rancherCatalog.tokenSecret` values.

If a field has a secret selected but no key, *Apply* is blocked. The message names each incomplete field, for example, `NVIDIA AI Library - Username Secret: select a key in Secret "nvidia-registry", or set the Secret to None.`, and opens its section. Select a key or set the secret to `None`, and select *Apply* again. The operator API and the `Settings` CRD reject incomplete references as well.

Note: Automatically assigned registry secrets

SUSE AI Factory with NVIDIA automatically assigns registry credentials from secrets with the following names in the `aif-operator` namespace:

- SUSE Application Collection: `appco` or `application-collection`
- SUSE Registry: `suse-registry`
- NVIDIA AI Library: `nvidia` or `nvidia-registry`

If you set the credentials of a registry to `None` while such a secret exists, the operator assigns it again. To keep the credentials removed, delete the secret or rename it.

3.8 How do I add custom chart repositories?

In addition to the built-in SUSE AI and NVIDIA repositories, you can add your own Helm, OCI and Git chart repositories. For each custom repository, SUSE AI Factory with NVIDIA creates a Rancher Prime `ClusterRepo` resource with the same name and lists its applications on the `Apps` page under a separate library.

1. If the repository requires authentication, add its credentials to the Rancher Prime *local* cluster.

    a. From the main left panel, select *local* › *Storage* › *Secrets*.

    b. Select *Create* in the top right and select *Opaque*.

    c. Select the *aif-operator* namespace and enter your preferred secret name (referenced as `<CUSTOM_REPO_SECRETS_NAME>` in command line tasks) to identify the resource.

    d. In the *Data* tab, add the key `user` with your user name and `token` with your password or token. For a Git repository accessed over SSH, add a key that holds the SSH private key instead.

    e. Confirm by selecting *Create* in the bottom right corner.

Tip: Creating the secrets on the command line

```
> kubectl create secret generic <CUSTOM_REPO_SECRETS_NAME> \
  --from-literal=user=<USERNAME> \
  --from-literal=token=<PASSWORD_OR_TOKEN> \
  -n aif-operator
```

2. Add the repository to SUSE AI Factory.

    a. From the main left panel, select *SUSE AI Factory* › *Settings*.

    b. In the `Repositories` section, select *Add Repository*.

    c. Enter the `Repository Name` and, optionally, a `Display Name (optional)` that is shown in the catalog.

    d. Select the `Repository Type`: `Helm`, `OCI` or `Git`.

    e. For a Helm or OCI repository, enter the `Repository URL`, for example, `https://example.com/charts` or `oci://example.com/charts`. For a Git repository, enter the `Git Repository URL` and the `Git Branch`.

    f. If the repository requires authentication, select the secret and key in `Username Secret (optional)` and `Token/Password Secret (optional)`. For a Git repository accessed over SSH, select the secret and key in `SSH Private Key Secret (optional)` instead.

    g. If the repository uses a certificate issued by a private CA, select the secret and key that hold the PEM-encoded CA certificate in `CA Bundle Secret (optional)`.

h. (*Optional*) Select *Test* to check the credentials, reachability and access to a sample chart.

i. Confirm by selecting *Apply* in the bottom right corner.

3. Navigate to *SUSE AI Factory* › *Apps* and select the library of your repository from the drop-down list in the top center of the screen.

Custom repositories are marked with a `Custom` badge in the `Repositories` section. To edit a custom repository, change its settings and select *Apply*. To delete it, select the delete icon of the repository, confirm with *Delete*, and select *Apply*.

Note: Reserved repository names

A custom repository cannot use the name of a built-in SUSE AI Factory with NVIDIA repository, of a default Rancher Prime repository such as `rancher-charts`, of an NVIDIA team repository, or of any other existing `ClusterRepo` resource.

Note: Single-chart OCI repositories

If the OCI repository URL already ends with the chart name, for example, `oci://ghcr.io/nvidia/openshell/helm-chart`, SUSE AI Factory with NVIDIA uses the URL as is and does not append the chart name again. Use this to add the NVIDIA OpenShell charts. The OpenShell gateway chart requires the Agent Sandbox CRDs and controller (`agents.x-k8s.io`) to be installed on the target cluster first.

Note: Testing repositories on private networks

Tests of unsaved repositories cannot reach private network addresses. If the repository is hosted on a private network, for example, a mirror in an air-gapped environment, select *Apply* first and then *Test*.

Note: Known limitations

- A custom HTTPS repository that uses a certificate issued by a private CA is not trusted through the CA bundle alone. As a workaround, enable `Skip TLS verification (development only)` for that repository.
- Anonymous tests against OCI registries that require repository-scoped tokens, such as GitHub Container Registry, report an authentication failure even though the charts can be pulled anonymously.
- Chart access is checked only for `https://` Helm and `oci://` repositories. A plain `http://` Helm repository can pass the connection test but reports a configuration error for chart access.
- Testing is not available for Git repositories accessed over SSH.
- Charts from Git repositories are subject to a size limit, as described in *Section 3.10, "How do I authorize access to Git-backed Rancher Prime catalogs?"*.

3.9 How do I add blueprint catalogs?

A blueprint catalog is a Git repository that contains `Blueprint` resources and a `BlueprintCatalog` resource that describes the catalog and lists its blueprints. SUSE AI Factory with NVIDIA synchronizes each catalog into the cluster through its own Fleet `GitRepo` resource. Blueprints from a catalog appear on the `Blueprints` page, where you can filter them by catalog. The blueprints bundled with SUSE AI Factory with NVIDIA are listed under `SUSE Blueprints`.

EXAMPLE 3.1: EXAMPLE `BlueprintCatalog` RESOURCE

```
apiVersion: ai-factory.suse.com/v1alpha1
kind: BlueprintCatalog
metadata:
  name: partner-acme
spec:
  displayName: ACME Blueprints
  blueprints:
    - name: acme-chatbot
```

```
apiVersion: ai-factory.suse.com/v1alpha1
kind: BlueprintCatalog
```

```
metadata:
  name: partner-acme
spec:
  displayName: ACME Blueprints
  blueprints:
    - name: acme-chatbot
```

1. If the Git repository is private, add the password or personal access token to a secret in the `aif-operator` namespace, as described in *Section 3.5, "How do I access a Git repository for application configuration?"*. If the Git server uses a certificate issued by a private CA, add the CA certificate to a secret in the same namespace, as described in *Section 3.6, "How do I trust a private CA certificate for a registry?"*.
2. From the main left panel, select *SUSE AI Factory* › *Settings*.
3. In the `Blueprint Catalogs` section, select *Add Catalog* under `Custom Catalogs`.
4. Enter the catalog `Name`. The name must be unique, can be up to 40 characters long, and must consist of lowercase alphanumeric characters and hyphens.
5. Enter the `Repository URL`, the `Branch`, and the directories that contain the blueprint files in `Paths (comma-separated)`.
6. If the repository is private, select the secret and key in `Password or Personal Access Token (optional)`.
7. If the Git server uses a certificate issued by a private CA, select the secret and key in `HTTPS CA Bundle (optional)`.
8. (*Optional*) Select *Test* to check access to the repository.
9. Confirm by selecting *Apply* in the bottom right corner.

To remove a catalog, select *Remove catalog* and then *Apply*. SUSE AI Factory with NVIDIA removes the corresponding Fleet `GitRepo` resource together with the blueprints of the catalog.

Tip: Configuring catalogs with Helm

You can also define blueprint catalogs when you install or upgrade the operator by using the `blueprintCatalogs` value of the `aif-operator` chart:

```
blueprintCatalogs:
  - name: partner-acme
```

```
    repoURL: https://github.com/acme/blueprints.git
    branch: main
    paths: [catalog/blueprints]
    credSecretRef: {name: acme-git, key: token}
    caBundleSecretRef: {name: acme-ca, key: ca.crt}
```

The referenced secrets must exist in the operator namespace.

Catalogs declared in `blueprintCatalogs` are managed by Helm. The `Settings` page shows them as `Managed by Helm` without edit or delete controls, and the operator API refuses to change them. Change or remove them only with `helm upgrade`, and pass `--reuse-values` so that Helm keeps the other values you set. Because Helm replaces lists rather than merging them, list every catalog that you want to keep.

Note: Combining Helm-declared catalogs with catalogs added on the `Settings` page

Keeping catalogs added on the `Settings` page next to catalogs declared in `blueprintCatalogs` requires server-side apply. Helm 4 on a fresh installation uses server-side apply. Helm 3, and Helm 4 upgrading a release first installed with Helm 3, use client-side apply, which drops the catalogs added on the `Settings` page whenever the Helm-declared list changes. Run one `helm upgrade --server-side=true` to switch the release to server-side apply; later upgrades keep it. This does not apply while `blueprintCatalogs` is empty.

Adding a catalog on the `Settings` page with the same name as a Helm-declared catalog but a different repository, branch or paths makes `helm upgrade` fail with a conflict. Remove or rename that catalog before you upgrade.

After `helm uninstall`, the retained `Settings` resource keeps the Helm-declared catalogs read-only. To manage them on the `Settings` page again, remove the annotation:

```
> kubectl annotate settings.ai-factory.suse.com settings -n aif-operator \
  ai-factory.suse.com/helm-managed-
```

Note: Catalog blueprints are read-only

Blueprints synchronized from a catalog cannot be edited or deleted in SUSE AI Factory with NVIDIA. To customize one, clone it as described in How do I clone a blueprint? (https://documentation.suse.com/suse-ai-factory/latest/html/AI-Factory-usage/ai-factory-managing-blueprints.html#ai-factory-managing-blueprints-copy). You can also over-

ride values and exclude components when you deploy it, as described in How do I install blueprints? (https://documentation.suse.com/suse-ai-factory/latest/html/AI-Factory-usage/ai-factory-managing-blueprints.html#ai-factory-managing-blueprints-install).

Blueprint `source` values, including `Partner` and `Community`, are declared by the blueprint author and are not verified by SUSE AI Factory with NVIDIA.

3.9.1 SUSE partner blueprint catalog

SUSE AI Factory with NVIDIA synchronizes the SUSE partner blueprint catalog by default. The `aif-operator` chart creates the Fleet `GitRepo` resource `fleet-local/aif-partner-blueprints`, which synchronizes the `partners` directory of the `main` branch of the SUSE partner-blueprints repository (https://github.com/suse/partner-blueprints). The `SUSE Partner Catalog` row in the `Blueprint Catalogs` section of *SUSE AI Factory* › *Settings* shows whether the catalog is synchronized.

The catalog includes blueprints from partners such as Tensormesh, Traefik Labs and Zentience Labs (AlchemiStudio). For the current list, refer to the catalog file (https://github.com/SUSE/partner-blueprints/blob/main/partners/catalog.yaml) and the `README` file of each partner in the repository. Because the catalog is synchronized from the `main` branch, new partner blueprints appear without upgrading SUSE AI Factory with NVIDIA. Support for each partner blueprint is provided by the contributing partner.

The partner catalog is a chart resource and is not listed under `Custom Catalogs`. Upgrading the chart never changes the catalogs added on the `Settings` page. If you added a catalog on the `Settings` page that points to the partner repository, remove it. Otherwise, two `GitRepo` resources synchronize the same blueprints and conflict.

The catalog is configured with the following values of the `aif-operator` chart:

Value	Description	Default
`partnerBlueprints.enabled`	Synchronize the partner catalog.	`true`
`partnerBlueprints.git.repoURL`	Git repository to synchronize from.	`https://github.com/suse/partner-blueprints.git`
`partnerBlueprints.git.branch`	Branch to synchronize.	`main`

Value	Description	Default
`partnerBlueprints.git.paths`	Repository directories to scan.	`[partners]`

To disable the partner catalog, run the following command. Fleet then removes the synchronized partner blueprints.

```
> helm upgrade aif-operator oci://ghcr.io/suse/chart/aif-operator \
  -n aif-operator \
  --reuse-values \
  --set partnerBlueprints.enabled=false
```

Note: Partner catalog in air-gapped environments

An air-gapped cluster cannot reach GitHub, so the `GitRepo` resource reports clone errors until you disable the catalog or point it at an internal mirror. If the mirror requires no credentials and its certificate is trusted by Fleet, set `partnerBlueprints.git.repoURL` to the URL of the mirror. If the mirror requires credentials or a private CA certificate, set `partnerBlueprints.enabled=false` and add the mirror as an ordinary catalog, either on the `Settings` page or in `blueprintCatalogs`:

```
partnerBlueprints:
  enabled: false
blueprintCatalogs:
  - name: partner-blueprints
    repoURL: https://gitea.example.com/mirrors/partner-blueprints.git
    branch: main
    paths: [partners]
    credSecretRef: {name: git-credentials, key: token}
    caBundleSecretRef: {name: private-ca, key: ca.crt}
```

Always pass `--reuse-values` when you change these values. Without it, Helm drops every value set earlier, and `--reset-values` enables the partner catalog again.

3.10 How do I authorize access to Git-backed Rancher Prime catalogs?

Applications and blueprint components can be deployed from Git-backed Rancher Prime `ClusterRepo` resources, including the built-in `rancher-charts` and `rancher-ui-plugins` repositories. To resolve charts from these repositories, SUSE AI Factory needs a Rancher Prime API token so that the operator can read the Rancher Prime catalog on the management cluster.

1. From the main left panel, select *SUSE AI Factory* › *Settings*.
2. In the `Rancher API Access` section, select *Authorize*. This mints a Rancher Prime API token as the signed-in user.
3. (*Optional*) Select *Test* to validate the token before saving.
4. Confirm by selecting *Apply* in the bottom-right corner.

Note: Token expiry

The section shows the token expiry and warns before it lapses. The maximum lifetime is limited by your Rancher Prime token settings. When the token expires, select *Authorize* again to renew it.

Note: CA certificate

The Rancher Prime CA certificate is discovered automatically, so TLS verification of the Rancher Prime catalog endpoint stays enabled by default.

Tip: Chart size limit

Charts resolved from a Git-backed `ClusterRepo`, including custom Git chart repositories, are delivered as an embedded Fleet `Bundle` and are subject to the Kubernetes object-size limit. SUSE AI Factory with NVIDIA stores the chart files compressed, so most charts fit even if they are larger than 1 MiB unpacked. For example, a chart of 2 MB unpacked is stored in about 0.43 MB. The following limits apply:

- The chart archive (`.tgz`) can be up to 4 MiB.
- The unpacked chart can be up to 32 MiB.
- The compressed chart files stored in the `Bundle` can be up to 1 MiB.

A chart that exceeds a limit fails to deploy with a `ChartTooLarge` error.

4 How do I upgrade SUSE AI Factory with NVIDIA?

To upgrade SUSE AI Factory with NVIDIA and its template dependencies, use the Helm command-line interface. This procedure explains how to apply custom override configurations, automatically pull dependency updates, and safely deploy the latest operator version.

Important

New releases may include breaking changes that require manual intervention. Always review the companion `README` file for release-specific instructions before upgrading.

To upgrade SUSE AI Factory with NVIDIA on the command line, follow these steps:

1. (*Optional*) Identify the operator chart version number that you want to upgrade to, for example:

   ```
   > skopeo list-tags docker://ghcr.io/suse/chart/aif-operator
   ```

   The latest version is 2.4.0.

2. Save all values of the AIF Operator Helm chart to `values.yaml` and customize them if required. Omit `--version` to display the latest chart.

   ```
   > helm show values oci://ghcr.io/suse/chart/aif-operator \
     --version 2.4.0 \
     --output yaml \
     > values.yaml
   ```

3. Upgrade SUSE AI Factory Operator. Omit `--version` to upgrade to the latest chart.

   ```
   > helm upgrade --install \
     aif-operator oci://ghcr.io/suse/chart/aif-operator \
     -n aif-operator \
     --version 2.4.0 \
     -f values.yaml
   ```

   The upgrade can take a few minutes.

Note: Custom resource definitions on upgrade

CRDs are installed on first installation through the standard `crds/` directory of the chart. Because Helm never upgrades that directory, the chart applies CRD schema changes through a pre-install and pre-upgrade job (`crds.manageWithJob=true`). The job only creates or updates CRDs; it never deletes them.

The job requires the `kubectl` image defined by the `crds.image` values. In an air-gapped environment, mirror this image or override the registry.

In restricted environments where the installer cannot create cluster-scoped RBAC, set `crds.rbac.create=false`, create the service account, cluster role and cluster role binding in advance, and set `crds.serviceAccountName` to the name of the service account in the release namespace.

To skip the job entirely, set `crds.manageWithJob=false`. You are then responsible for applying the CRDs from the chart's `crds/` directory with `kubectl apply --server-side -f crds/` before each upgrade.

Note: Rancher Prime readiness check

Before the operator is installed or upgraded, a pre-install and pre-upgrade job waits until the required Fleet and Rancher Prime catalog CRDs and the `cattle-ui-plugin-system` namespace are available. The job waits for as long as the installer allows, for example as set by `helm --timeout` or by the Fleet `helm.timeoutSeconds` option. If Rancher Prime does not become ready within that time, the installation or upgrade fails.

The job uses the `kubectl` image defined by the `global.kubectl.image` values, by default `registry.suse.com/suse/kubectl:1.35`. In an air-gapped environment, mirror this image or override the registry.

Important: Changes to review before upgrading to 2.4.0

- **Upgrade the operator and the UI extension to matching versions.** The fixes for `None` and partial secret references on the `Settings` page require both the operator and the UI extension in version 2.4.0.

- **The `Settings` and `InstallAIExtension` CRDs are updated.** The `name` and `key` of a secret reference must not be empty. The CRD job applies this change during the upgrade. On Kubernetes 1.30 and later, a `Settings` resource that already stores a secret reference

without a key, or a key without a secret, can still be updated as long as that field is not changed. Complete or clear such references the next time you edit the settings, as described in *Section 3.7, "How do I change or remove secret references in the settings?"*.

- **New installations use a random default namespace.** The installation wizards suggest a namespace such as `<APP>-k3x9q2a` instead of `<APP>-system`. If your scripts or policies expect `<APP>-system`, enter that namespace explicitly in the wizard. Existing workloads keep their namespace.

Important: Changes to review when upgrading from 2.3.0

- **The partner blueprint catalog is enabled by default.** The chart creates the Fleet `GitRepo` resource `fleet-local/aif-partner-blueprints`, which synchronizes the SUSE partner blueprints from GitHub. If you already added a catalog on the `Settings` page that points to the same repository, remove it after upgrading. Otherwise, two `GitRepo` resources synchronize the same blueprints and conflict. In air-gapped environments, disable the catalog or point it at a mirror, as described in *Section 3.9.1, "SUSE partner blueprint catalog"*.

- **Catalogs declared in `blueprintCatalogs` are read-only outside Helm.** Change or remove them only with `helm upgrade`, and always pass `--reuse-values` so that Helm keeps the other values you set. If a catalog added on the `Settings` page has the same name as a Helm-declared catalog but a different repository, branch or paths, `helm upgrade` fails with a server-side apply conflict. Remove or rename that catalog first.

- **Server-side apply is required to combine both kinds of catalogs.** If `blueprintCatalogs` is not empty and you also add catalogs on the `Settings` page, upgrade with server-side apply. Releases first installed with Helm 3, including releases upgraded through Rancher Prime `Apps`, keep using client-side apply, which drops the catalogs added on the `Settings` page. Run one `helm upgrade --server-side=true` to switch the release to server-side apply; later upgrades keep it.

- **Blueprints with `source: Community` require 2.3.1 or newer.** Operator version 2.3.0 and earlier reject this value. If you publish blueprint catalogs, do not use it until 2.3.1 is the minimum operator version that you support.

Important: Changes to review when upgrading from 2.2.x

- **Upgrade the operator and the UI extension to matching versions.** Repository chart-access checks, blueprint customization and partner icons require both the operator and the UI extension in version 2.3.0 or newer.
- **Mirror the new `kubectl` image in air-gapped environments.** The Rancher Prime readiness job uses the image defined by `global.kubectl.image`.
- **New CRD and CRD changes.** The `blueprintcatalogs.ai-factory.suse.com` CRD is added, and the `AIWorkload`, `Blueprint` and `Settings` CRDs are updated. The CRD job applies these changes during the upgrade.
- **The status of application workloads deployed with Helm may change.** Workloads that previously showed `Running` may show `Pending` or `Degraded` after the upgrade if their pods are not ready.
- **The `aif-ui-config` ConfigMap is retained when the UI extension is uninstalled.** If you install the UI extension chart on its own, without the operator, over an existing `aif-ui-config` ConfigMap, run `helm install` or `helm upgrade` with the `--take-ownership` option (Helm 3.15 or newer).
- **Custom repository names are reserved.** Custom chart repositories cannot use the names of built-in SUSE AI Factory with NVIDIA repositories, default Rancher Prime repositories, NVIDIA team repositories or other existing `ClusterRepo` resources.
- **Blueprints with `source: Partner` require 2.3.0 or newer.** Operator version 2.2.0 rejects this value. If you publish blueprint catalogs, do not use it until 2.3.0 is the minimum operator version that you support.

5 How do I uninstall SUSE AI Factory with NVIDIA?

To uninstall SUSE AI Factory with NVIDIA, you need to perform the following tasks:

- Optionally, back up configuration, secrets and custom resources.
- Uninstall SUSE AI Factory Operator.
- Remove installation leftovers.
- Optionally, remove Custom Resource Definitions (CRDs).

You can uninstall SUSE AI Factory with NVIDIA either using Rancher Prime or the command line.

Note: What uninstalling removes

Uninstalling the operator chart removes the bundled blueprints, the SUSE partner blueprint catalog, and the UI extension. The CRDs, the `Settings` resource, and the `aif-ui-config` ConfigMap of the UI extension are deliberately retained so that a reinstallation preserves your configuration and existing workload resources.

Remove them explicitly with the steps below if you want a complete removal.

5.1 Back up SUSE AI Factory with NVIDIA resources and configuration (optional)

Before uninstalling SUSE AI Factory with NVIDIA, back up your configuration, secrets and custom resources to preserve them for future use or reinstallation:

1. **Back up Helm chart values**
   Export the values used to deploy `SUSE AI Factory Operator`:

   ```
   > helm get values aif-operator -n aif-operator -a > aif-operator-values-backup.yaml
   ```

2. **Back up settings**
   Export the `Settings` custom resource in the `aif-operator` namespace:

   ```
   > kubectl get settings settings -n aif-operator -o yaml > aif-settings-backup.yaml
   ```

3. **Back up secrets**

Export all secrets (including registry credentials, Git access tokens, and CA certificates) configured in the `aif-operator` namespace:

```
> kubectl get secrets -n aif-operator -o yaml > aif-secrets-backup.yaml
```

4. **Back up UI extension configuration**

   Export the `aif-ui-config` ConfigMap from the `cattle-ui-plugin-system` namespace:

   ```
   > kubectl get configmap aif-ui-config -n cattle-ui-plugin-system -o yaml > aif-ui-
   config-backup.yaml
   ```

5. **Back up SUSE AI Factory with NVIDIA custom resources**

   Export all existing blueprints, blueprint catalogs, AI workloads, and extension install resources:

   ```
   > kubectl get blueprints,blueprintcatalogs,aiworkloads,installaiextensions -A -o
    yaml > aif-cr-backup.yaml
   ```

5.2 Uninstall using Helm

1. **Uninstall SUSE AI Factory Operator**

   a. Uninstall the SUSE AI Factory extension and the operator.

   ```
   > helm uninstall aif-operator -n aif-operator
   ```

   b. Verify that the resources have been removed by running the following commands.

   ```
   > kubectl get all -n aif-operator
   > kubectl get aifext
   > helm list -n aif-operator
   ```

2. **Remove installation leftovers**

   a. Remove the related settings resource.

   ```
   > kubectl delete settings settings -n aif-operator
   ```

   b. Remove the configuration of the UI extension.

   ```
   > kubectl delete configmap aif-ui-config -n cattle-ui-plugin-system
   ```

3. To perform a complete removal of all resources, **remove CRDs**.

a. Delete SUSE AI Factory with NVIDIA CRDs.

```
> kubectl delete crd \
  aiworkloads.ai-factory.suse.com \
  blueprintcatalogs.ai-factory.suse.com \
  blueprints.ai-factory.suse.com \
  installaiextensions.ai-factory.suse.com \
  settings.ai-factory.suse.com
```

5.3 Uninstall using Rancher Prime

1. **Uninstall the SUSE AI Factory extension and the operator.**

   a. From the main left panel, select the cluster you want to uninstall from, then *Apps* › *Installed Apps*.

   b. From the list of installed applications, activate the checkbox of `aif-operator` and click *Delete* above the list.

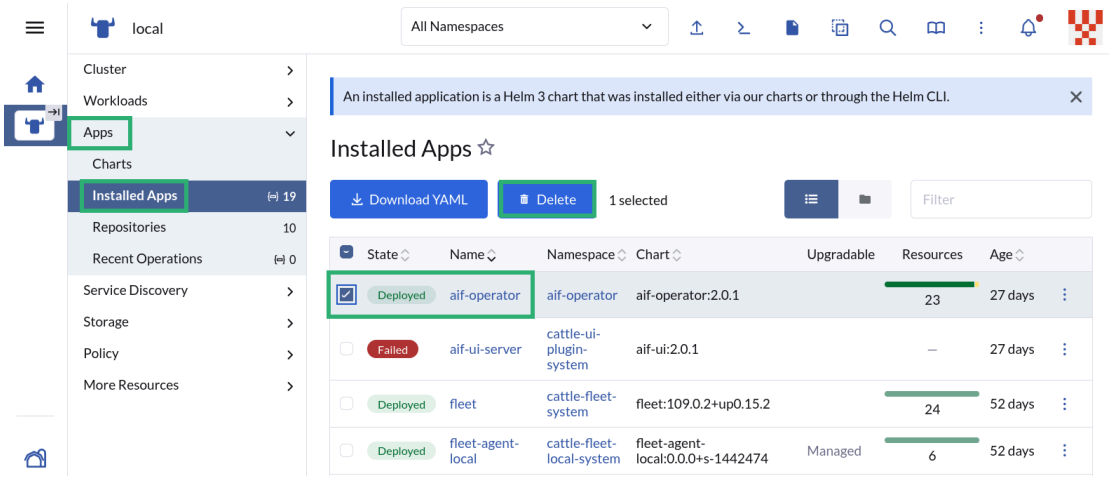

FIGURE 5.1: UNINSTALLING SUSE AI FACTORY OPERATOR

   c. Confirm the deletion and wait until it is finished.

2. **Remove installation leftovers**

   a. From the same cluster menu, select *More Resources* › *ai-factory.suse.com* › *Settingses*.

   b. Activate the checkbox of `aif-operator` and click *Delete* above the list.

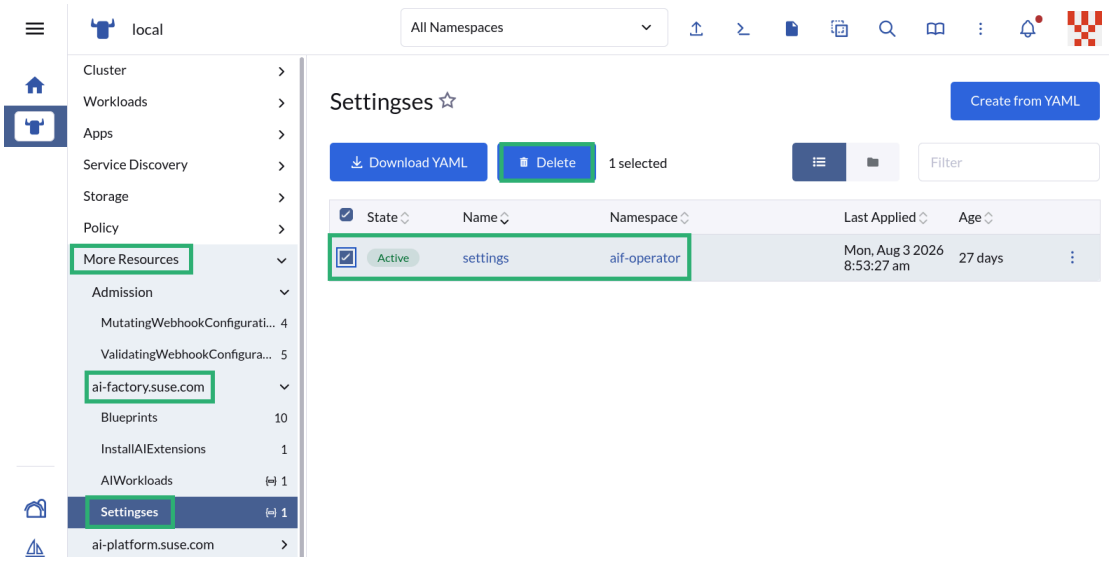

FIGURE 5.2: UNINSTALLING SUSE AI FACTORY OPERATOR LEFTOVERS

c. Confirm the deletion.

d. From the same cluster menu, select *Storage* › *ConfigMaps*.

e. Activate the checkbox of `aif-ui-config` in the `cattle-ui-plugin-system` namespace and click *Delete* above the list.

f. Confirm the deletion.

3. To perform a complete removal of all resources, **remove CRDs**.

a. From the same cluster menu, select *More Resources* › *API* › *CustomResourceDefinitions*.

b. Filter the list by entering `ai-factory.suse.com` in the search box.

c. Activate the checkbox of all `ai-factory.suse.com` resources and click *Delete* above the list.

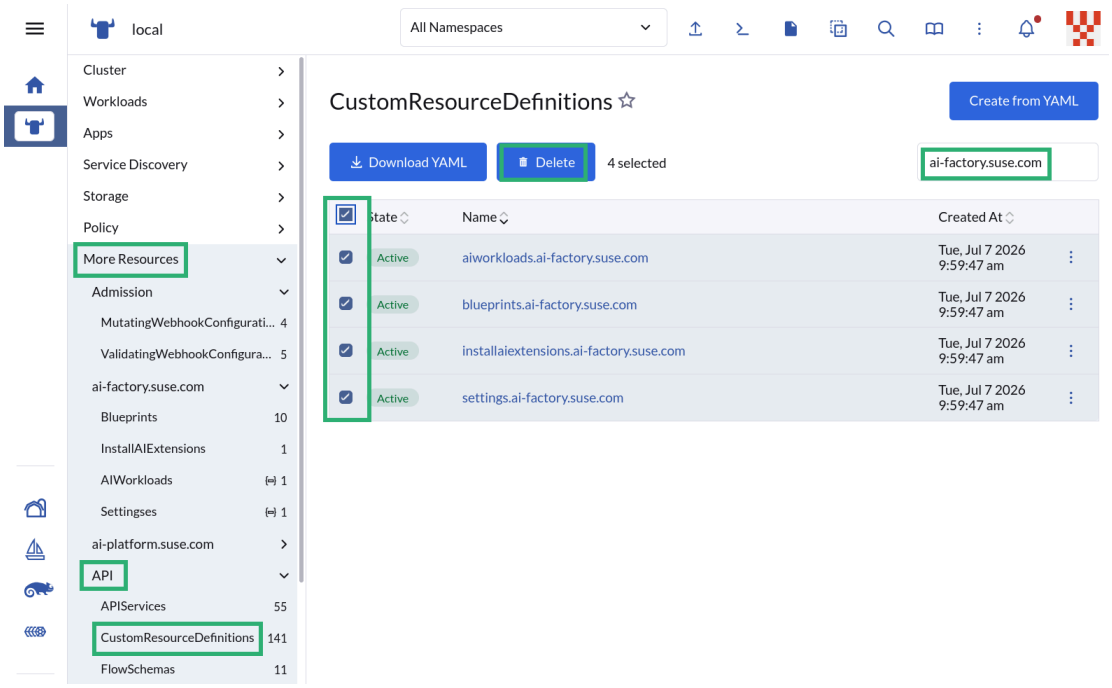

FIGURE 5.3: UNINSTALLING SUSE AI FACTORY OPERATOR CRDS

d. Confirm the deletion.