Troubleshooting SUSE AI Factory

Troubleshooting SUSE AI Factory

Troubleshooting SUSE AI Factory involves diagnosing issues across RKE2, Rancher Prime, and containerized AI workloads. This guide covers fundamental troubleshooting principles, diagnostic data collection, and debugging techniques using `kubectl` and `Helm`.

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 What troubleshooting principles should I follow? 1

2 How do I collect diagnostics? 3

2.1 Essential information to collect 3

2.2 How to collect 4

2.3 Contact support 4

3 How do I troubleshoot related components? 5

4 How do I debug issues? 6

4.1 Validating SUSE AI Factory Operator and UI extension components 6

Validating that SUSE AI Factory Operator is running 6 • Validating that the UI extension is installed 6

4.2 Inspecting objects created by the operator 7

Listing SUSE AI Factory Custom Resources 7 • Listing Fleet GitOps and Helm deployment objects 7 • Inspecting operator-created object definitions and status 8

4.3 Find the namespace and pods 8

4.4 Get the complete definition of an object 9

4.5 Describe detailed resource information and related events 10

4.6 Inspect logs 10

4.7 Viewing events 11

4.8 Inspecting Helm charts 11

5 How do I uninstall SUSE AI Factory including CRDs? 13

5.1 Back up SUSE AI Factory resources and configuration (optional) 13

5.2 Uninstall using Helm 14

5.3 Uninstall using Rancher Prime 15

6 What happens if a registry credentials secret is deleted? 18

1 What troubleshooting principles should I follow?

Before diving into component-specific issues, consider the following general principles.

Check logs

Logs are the primary source of information. Most of the time, errors are self-explanatory and contain hints on what failed.

Check clocks

Having clock differences between systems can lead to all kinds of different errors. Ensure clocks are in sync.

Boot issues

If the system is stuck during boot, note down the last messages displayed. Access the console (physical or via BMC) to observe boot messages.

Network issues

Verify network interface configuration (`ip a`), routing table (`ip route`), test connectivity from/to other nodes and external services (`ping`, `nc`). Ensure firewall rules are not blocking necessary ports.

Verify component status

Use `kubectl get` and `kubectl describe` for Kubernetes resources. Use `kubectl get events --sort-by='.lastTimestamp' -n <NAMESPACE>` to see the events on a particular Kubernetes namespace. Use `kubectl get events --sort-by='.metadata.creationTimestamp' -n <NAMESPACE>` to sort events by creation time on a particular Kubernetes namespace.

Verify services status

Use `systemctl status <SERVICE>` for systemd services.

Check syntax

Software expects certain structure and syntax on configuration files. For YAML files, for example, use `yamllint` or similar tools to verify the proper syntax.

Isolate the problem

Try to narrow down the issue to a specific component or layer (for example, network, storage, OS or Kubernetes).

Documentation

Always refer to the official SUSE AI Factory documentation (https://documentation.suse.com/suse-ai-factory/) and also upstream documentation for detailed information.

Known issues

For each SUSE AI Factory release, the release notes contain a "Known issues" section detailing issues that will be fixed in future releases.

2 How do I collect diagnostics?

When contacting SUSE Support, providing comprehensive diagnostic information is crucial.

2.1 Essential information to collect

Detailed problem description

What happened, when did it happen, what were you doing, what is the expected behavior, and what is the actual behavior?

Steps to reproduce

Can you reliably reproduce the issue? If so, list the exact steps.

Component versions

SUSE AI Factory version, components versions (RKE2, Rancher Prime).

Relevant logs

- `journalctl` output (filtered by service if possible, or full boot logs).
- Kubernetes pod logs (`kubectl logs`).

System information

- `uname -a`
- `df -h`
- `ip a`
- `/etc/os-release`

Configuration files

Relevant configuration files such as Helm chart values, ConfigMaps, etc.

Kubernetes information

Nodes, Services, Deployments, etc.

2.2 How to collect

For logs

Redirect command output to files (for example, `journalctl -u rke2-server > rke2_logs.txt`).

For Kubernetes resources

Use `kubectl get <RESOURCE> -o yaml > <RESOURCE_NAME>.yaml` to get detailed YAML definitions.

For Helm chart values

Redirect `helm get values` output to a file (for example, `helm get values <RELEASE_NAME> -n <NAMESPACE> -a > <RELEASE_NAME>-values.yaml`).

For system information

Collect output of the commands listed above.

For RKE2/Rancher Prime

Check the The Rancher v2.x Linux log collector script (https://www.suse.com/support/kb/doc/?id=000020191) article to run The Rancher v2.x Linux log collector script.

2.3 Contact support

Please check the articles available at How-to effectively work with SUSE Technical Support (https://www.suse.com/support/kb/doc/?id=000019452) and the support handbook located at SUSE Technical Support Handbook (https://www.suse.com/support/handbook/) for more details on how to contact SUSE support.

3 How do I troubleshoot related components?

Other SUSE AI Factory components troubleshooting guides can be consulted on their official documentation.

- RKE2 Known Issues and Limitations (https://documentation.suse.com/cloudnative/rke2/latest/en/known_issues.html)
- Rancher Prime Troubleshooting (https://documentation.suse.com/cloudnative/rancher-manager/latest/en/troubleshooting/general-troubleshooting.html)
- SUSE Storage Troubleshooting (https://documentation.suse.com/cloudnative/storage/latest/en/troubleshooting-maintenance/troubleshooting.html)
- SUSE Edge Troubleshooting (https://documentation.suse.com/suse-edge/3.7/html/edge/troubleshooting.html)
- SUSE Virtualization Troubleshooting (https://documentation.suse.com/cloudnative/virtualization/latest/en/troubleshooting/installation.html)
- SUSE Private Registry (https://documentation.suse.com/cloudnative/suse-private-registry/html/private-registry/pr-troubleshooting.html)
- Fleet Troubleshooting (https://documentation.suse.com/cloudnative/continuous-delivery/latest/en/troubleshooting.html)

You can also see SUSE Knowledgebase (https://www.suse.com/support/kb/).

4 How do I debug issues?

Because Rancher Prime runs on Kubernetes, `kubectl` is the primary tool for debugging workload issues.

4.1 Validating SUSE AI Factory Operator and UI extension components

Before troubleshooting workload issues, verify that SUSE AI Factory Operator and its UI components are properly installed and running.

4.1.1 Validating that SUSE AI Factory Operator is running

Check that the operator pod is in the `Running` state and ready in the `aif-operator` namespace:

```
> kubectl get pods -n aif-operator
NAME                            READY   STATUS    RESTARTS   AGE
aif-operator-7bf464987-mnrw5    1/1     Running   0          2d
```

Check the operator `Deployment` status:

```
> kubectl get deployment aif-operator -n aif-operator
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
aif-operator   1/1     1            1           108d
```

View the operator logs to verify successful reconciliation without errors:

```
> kubectl logs -n aif-operator -l app.kubernetes.io/name=aif-operator --tail=50
```

4.1.2 Validating that the UI extension is installed

The operator manages the UI extension lifecycle using the `InstallAIExtension` Custom Resource (CR). Verify that the extension CR is present and check its status:

```
> kubectl get installaiextensions.ai-factory.suse.com -A
NAME     PHASE       SOURCE   EXTENSION   VERSION   AGE
aif-ui   Installed   Helm     aif-ui      2.2.0     108d
```

Alternatively, use the short name `aifext`:

```
> kubectl get aifext -A
```

Inspect the detailed status and conditions of the extension installation:

```
> kubectl describe aifext -n aif-operator
```

In the Rancher Prime UI, navigate to *Extensions* › *Installed Extensions* and confirm that `SUSE AI Factory` is listed as installed and active.

4.2 Inspecting objects created by the operator

SUSE AI Factory Operator manages custom resources to declare blueprints, AI workloads, settings and UI extensions. It also orchestrates Fleet bundles and Helm releases to deploy applications across clusters.

4.2.1 Listing SUSE AI Factory Custom Resources

List all available blueprints (both bundled and custom templates):

```
> kubectl get blueprints.ai-factory.suse.com -A
```

List all AI workload instances deployed across namespaces:

```
> kubectl get aiworkloads.ai-factory.suse.com -A
```

List configuration settings (registries, GitOps and API access):

```
> kubectl get settings.ai-factory.suse.com -n aif-operator
```

List UI extension installation resources:

```
> kubectl get installaiextensions.ai-factory.suse.com -A
```

4.2.2 Listing Fleet GitOps and Helm deployment objects

When deploying applications and blueprints, the operator generates GitOps bundles and Helm operations.

List Fleet bundles and their deployment status across clusters:

```
> kubectl get bundles.fleet.cattle.io -A
```

```
> kubectl get bundledeployments.fleet.cattle.io -A
```

List Helm releases and chart definitions:

```
> helm list -A
> kubectl get helmcharts.helm.cattle.io -A
```

4.2.3 Inspecting operator-created object definitions and status

Display the full YAML specification of a blueprint:

```
> kubectl get blueprint <BLUEPRINT_NAME> -o yaml
```

Display the full YAML specification and runtime status of an AI workload:

```
> kubectl get aiworkload <WORKLOAD_NAME> -n <NAMESPACE> -o yaml
```

Inspect detailed conditions and events for an AI workload:

```
> kubectl describe aiworkload <WORKLOAD_NAME> -n <NAMESPACE>
```

Inspect a Fleet bundle deployment state:

```
> kubectl describe bundle <BUNDLE_NAME> -n <FLEET_NAMESPACE>
```

4.3 Find the namespace and pods

List all installed pods sorted by their namespace.

```
> kubectl get pods -A
NAMESPACE                   NAME                                        READY   STATUS
 RESTARTS       AGE
aif-operator                aif-operator-7d6b8f6bb5-j6lkk               1/1     Running
 30 (79m ago)   33d
cattle-capi-system          capi-controller-manager-7654f98879-zrk2v    1/1     Running
 82 (79m ago)   115d
cattle-fleet-local-system   fleet-agent-598d897c9b-bshfg                1/1     Running
 69 (79m ago)   114d
cattle-fleet-system         fleet-controller-58f99fb59b-cv2qd           3/3     Running
 219 (79m ago)  115d
[...]
```

Filter lines for a specific application and use wider view.

```
> kubectl get pods -A -o wide | grep -i ollama
NAMESPACE  NAME                      READY   STATUS   RESTARTS  AGE            IP
    NODE     NOMINATED NODE   READINESS GATES
ollama     ollama-6ffbcfc6cf-nlrcr   1/1     Running  9         (89m ago) 13d
 10.42.0.177   debian   <none>           <none>
```

Tip

`kubectl get` can output the actual Kubernetes object as YAML or JSON, which is useful for debugging. Add `-o yaml` or `-o json` to the `kubectl` command line to retrieve more details about an object.

4.4 Get the complete definition of an object

Display the full YAML specification of a Deployment.

```
> kubectl get deployment ollama -n ollama -o yaml
```

Display the full YAML specification and runtime status of a specific Pod.

```
> kubectl get pod ollama-6ffbcfc6cf-nlrcr -n ollama -o yaml
```

Find and inspect the configuration of a Service.

```
> kubectl get svc -A | grep -i ollama
[...]
> kubectl get svc ollama -n ollama -o yaml
```

Locate and inspect the YAML definition of a Secret.

```
> kubectl get secret -A | grep ollama
[...]
> kubectl get secret sh.helm.release.v1.ollama.v1 -n ollama -o yaml
```

Locate and view the contents of a ConfigMap.

```
> kubectl get configmap -A | grep ollama
[...]
> kubectl get configmap kube-root-ca.crt -n ollama -o yaml
```

4.5 Describe detailed resource information and related events

The `kubectl describe` command displays human-readable details about a resource, including runtime conditions and recent cluster events:

- **Scheduling issues:** Explains why a Pod remains in `Pending` state (for example, insufficient CPU/memory, missing GPU resources, or unmet node selector and taint constraints).
- **Image pull failures:** Highlights why container images cannot be fetched, distinguishing between invalid tags, network timeouts, or missing authentication secrets (`ErrImagePull`, `ImagePullBackOff`).
- **Volume and mount issues:** Shows persistent volume claims (PVC) binding status, missing secrets or configmaps required for mounts, and storage attachment errors.
- **Environment and configuration:** Lists resolved environment variables, command arguments, resource limits/requests, and active annotations or labels.
- **Health checks and probes:** Reports readiness, liveness and startup probe failures along with exit codes and error messages from failing check scripts.
- **Container states and restart reasons:** Details whether containers are running, terminated or crashing (`CrashLoopBackOff`), including exit codes and time stamps of previous terminations.
- **Events:** Displays chronological Kubernetes events related to the object (such as successful assignment, pulling, creation or warning/failure notices from controllers).

```
> kubectl describe pod ollama-6ffbcfc6cf-nlrcr -n ollama
```

4.6 Inspect logs

Print the current stdout and stderr logs of a Pod.

```
> kubectl logs ollama-6ffbcfc6cf-nlrcr -n ollama
```

Stream (follow) logs in real time to monitor ongoing activity.

```
> kubectl logs -f ollama-6ffbcfc6cf-nlrcr -n ollama
```

Retrieve logs from a previously crashed or terminated container instance to identify failure causes.

```
> kubectl logs ollama-6ffbcfc6cf-nlrcr -n ollama --previous
```

4.7 Viewing events

Events sometimes offer more useful information than logs.

View events for the whole cluster.

```
> kubectl events -A
```

View events for a specific pod.

```
> kubectl events -n ollama --for pod/ollama-6ffbcfc6cf-nlrcr
```

4.8 Inspecting Helm charts

Because Kubernetes applications are commonly Helm-based, the `helm` command can reveal useful information too.

List all deployed Helm releases across all namespaces.

```
> helm list -A
```

Display the deployment status, revision number, and post-installation notes for a specific release.

```
> helm status ollama -n ollama
NAME: ollama
LAST DEPLOYED: Mon Sep 21 12:12:05 2026
NAMESPACE: ollama
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
[...]
```

Output the rendered Kubernetes resource manifests generated by the chart.

```
> helm get manifest ollama -n ollama
```

Display the user-supplied values used when installing or upgrading the release.

```
> helm get values ollama -n ollama
```

Save the user-supplied values to a file.

```
> helm get values ollama -n ollama > ollama-values.yaml
```

Save all values including chart defaults to a file.

```
> helm get values ollama -n ollama -a > ollama-all-values.yaml
```

Retrieve all information about the release, including notes, custom values, and generated manifests.

```
> helm get all ollama -n ollama
```

5 How do I uninstall SUSE AI Factory including CRDs?

To uninstall SUSE AI Factory, 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 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 resources and configuration (optional)

Before uninstalling SUSE AI Factory, 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 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 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.

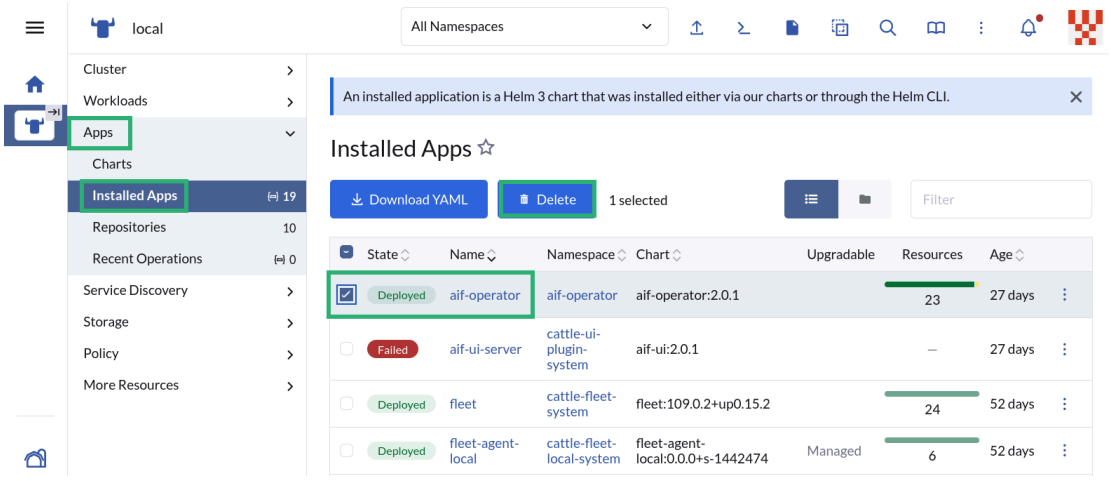

FIGURE 5.1: UNINSTALLING SUSE AI FACTORY OPERATOR

   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.

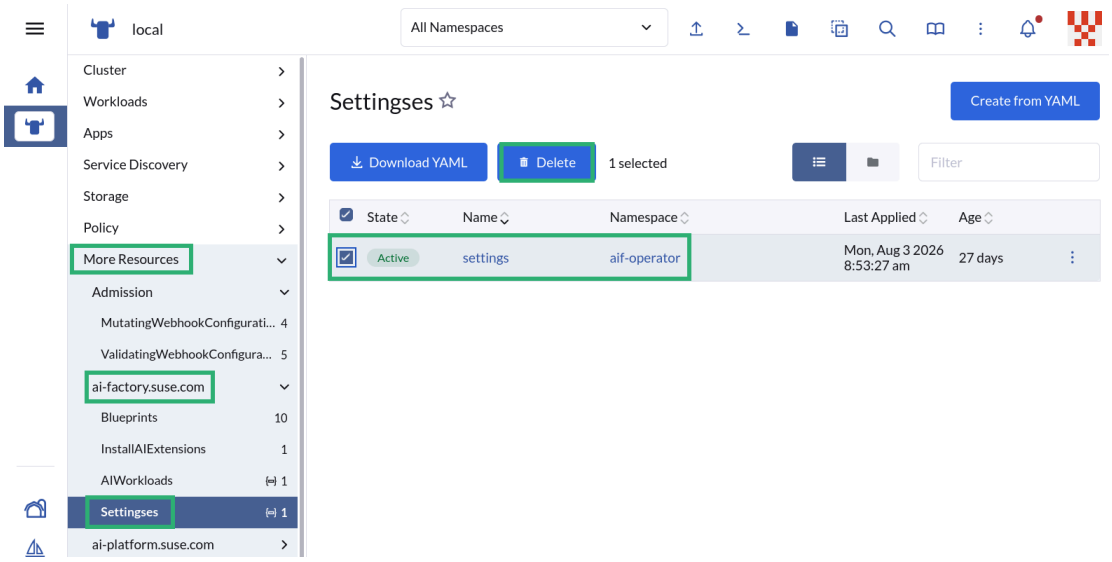

FIGURE 5.2: UNINSTALLING SUSE AI FACTORY OPERATOR LEFTOVERS

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.

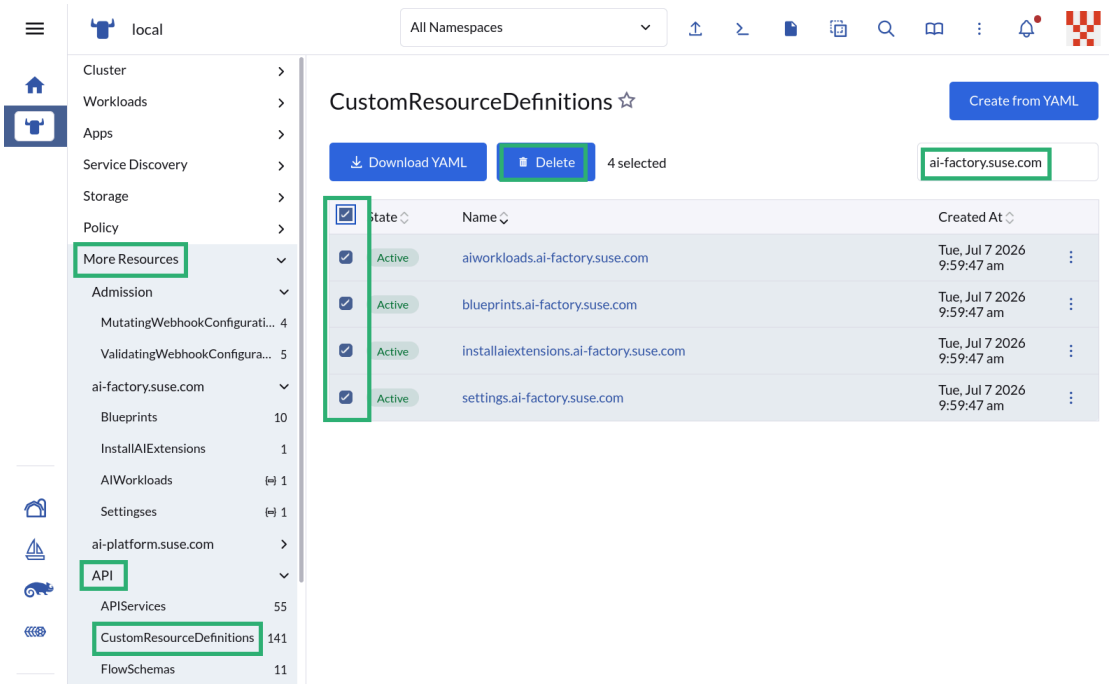

FIGURE 5.3: UNINSTALLING SUSE AI FACTORY OPERATOR CRDS

d. Confirm the deletion.

6 What happens if a registry credentials secret is deleted?

If a container registry credentials Secret (referenced by `imagePullSecrets`) is removed or invalid, currently running pods continue to operate normally because credentials are only verified during image pull operations.

However, subsequent actions that require pulling an image — such as pod restarts, rolling deployments, rescheduling or scaling operations — may fail with `ErrImagePull` or `ImagePullBackOff` under the following conditions:

- **Pod scheduled on a different node:** The new node does not have the image cached locally in its container runtime storage and cannot pull it without valid credentials.
- `imagePullPolicy: Always`**:** The Kubernetes agent Kubelet is forced to verify the remote registry manifest on every start, which fails with an HTTP 401 (Unauthorized) error.
- **Image cache invalidation:** If an image previously cached on a node has been pruned by Kubelet image garbage collection, the node cannot re-fetch the image.

Note

If a pod is restarted on the **same node**, uses `imagePullPolicy: IfNotPresent`, and the image is still present in the node's local container runtime cache, the container starts successfully without querying the registry.

To resolve `ImagePullBackOff` caused by missing credentials, re-create the registry secret and ensure it is referenced in the pod specification or the default ServiceAccount for the namespace. Refer to Post-deployment steps (https://documentation.suse.com/suse-ai-factory/latest/html/AI-Factory-deployment/ai-factory-post-deployment.html) for more details.