Install MinIO 2.3.0 on vSphere 8.0

November 2026

MinIO 2.3.0 is a vSphere Supervisor Service. It installs the MinIO Operator and the MinIO plug-in for the vSphere Client on a Supervisor, and the plug-in creates and manages **MinIO AIStor** tenants in vSphere Namespaces, on vSAN Direct storage. 2.3.0 needs an **AIStor license**.

2.3.0 also fixes the CVEs reported against the earlier plug-in and operator images: the plug-in is built with Go 1.26.10 and updated dependencies, and MinIO Operator v5.0.21 runs on a distroless image with no shell (use kubectl debug instead of kubectl exec to troubleshoot it).

Service definition	minio-service-definition-2.3.0.yaml
sha256	03c90a7b2138868e43d86be5a42ce7552646f69acb1ddb68116e7254977e32dc
Operator	quay.io/miniohq/operator:v5.0.21 sha256:415c5aa11bb5a4725365ac71f9e048538e2403da28906647d6f7bdf1f9629f74
Plug-in	quay.io/miniohq/vsphere:v2.3.0 sha256:1002faad686a070aae98d5d9be899019c61a95e0fe976dbe1affb6c4e93e6428
AIStor (tenants)	quay.io/minio/aistor/minio:RELEASE.2026-09-19T17-05-25Z

The operator and plug-in digests are their linux/amd64 images, which the definition pins (<image>:<tag>@sha256:<digest>); the copies in your registry must have the same ones.

Contents

- Requirements
- Before you start: copy the images to a private registry
- 1. Register the service
- 2. Install it on the Supervisor
- 3. Verify
- 4. Activate the AIStor license
- 5. Prepare a namespace for tenants
- Default tenant image
- 6. Create a tenant
- Service config keys
- Removing the service
- Known issues

Requirements

- **vSphere 8.0** with **vSAN**, and the disks MinIO will use claimed as **vSAN Direct**.
- **HA and DRS** enabled on the cluster.
- **Workload Management** enabled, with a Supervisor that is **Running** and **Ready**. Supervisor Services work with VDS networking from vSphere 8.0 U1; NSX is not required.
- A **private container registry** (e.g. Harbor) that the Supervisor reaches over HTTPS, with a certificate the Supervisor trusts, for the operator and plug-in images.

- Credentials for MinIO's private quay.io/miniohq repositories, to copy those images. Contact MinIO at dev@min.io or sales@min.io, or open a request in SUBNET.
- An **AIStor license**.
- Optional: kubectl with the vSphere plug-in, to check the Supervisor from a workstation:

```
kubectl vsphere login --server=<supervisor-address> \
  --vsphere-username administrator@vsphere.local --insecure-skip-tls-verify
kubectl get nodes
```

Before you start: copy the images to a private registry

Required. The Supervisor pulls the operator and plug-in images from a private registry of yours: copy them into one project of that registry before the install, and give the Supervisor that registry's credentials (step 2). registryName is that registry **and project**, e.g. registry.example.com/minio, and the definition pulls the images from it under their own names:

Image	Source	In the private registry (registryName: registry.example.com/minio)
Operator	quay.io/miniohq/operator:v5.0.21	registry.example.com/minio/ operator:v5.0.21@sha256:…
Plug-in	quay.io/miniohq/vsphere:v2.3.0	registry.example.com/minio/ vsphere:v2.3.0@sha256:…
AIStor (tenants' default)	quay.io/minio/aistor/ minio:RELEASE.2026-09-19T17-05-25Z (public)	optional, see step 3

1. **Create the project** in the private registry (in Harbor, e.g. minio) and a robot account that can pull from it.
2. **Copy the operator and plug-in images** into the project, with all their platforms, so that the digests the definition pins stay the same. Do not rebuild them. With skopeo, logged in to quay.io with the MinIO credentials and to the private registry:

   ```
   PROJECT=registry.example.com/minio
   skopeo copy --all docker://quay.io/miniohq/operator:v5.0.21 docker://$PROJECT/operator:v5.0.21
   skopeo copy --all docker://quay.io/miniohq/vsphere:v2.3.0 docker://$PROJECT/vsphere:v2.3.0
   ```

   Check that the pinned digests are there:

```
skopeo inspect --raw docker://$PROJECT/
operator@sha256:415c5aa11bb5a4725365ac71f9e048538e2403da28906647d6f7bdf1f9629f74 \
  >/dev/null && echo ok
skopeo inspect --raw docker://$PROJECT/
vsphere@sha256:1002faad686a070aae98d5d9be899019c61a95e0fe976dbe1affb6c4e93e6428 \
  >/dev/null && echo ok
```

3. **The AIStor image for the tenants: optional.** Tenants run AIStor, quay.io/minio/aistor/minio:RELEASE.2026-09-19T17-05-25Z, which is **public**: by default they pull it from quay.io. Copy it only if the ESXi hosts cannot reach quay.io, and set minioImage to the copy at the install (default tenant image):

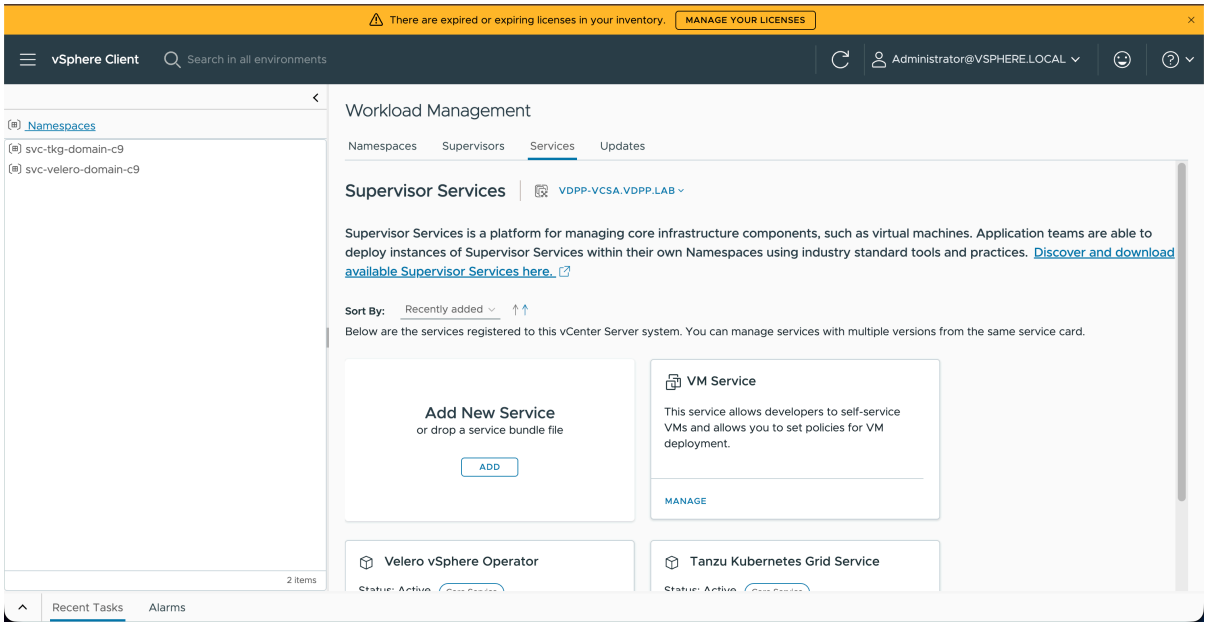
```
skopeo copy --all docker://quay.io/minio/aistor/minio:RELEASE.2026-09-19T17-05-25Z \
  docker://$PROJECT/aistor:RELEASE.2026-09-19T17-05-25Z
```

MinIO no longer provides the open-source minio/minio images, on quay.io or Docker Hub.

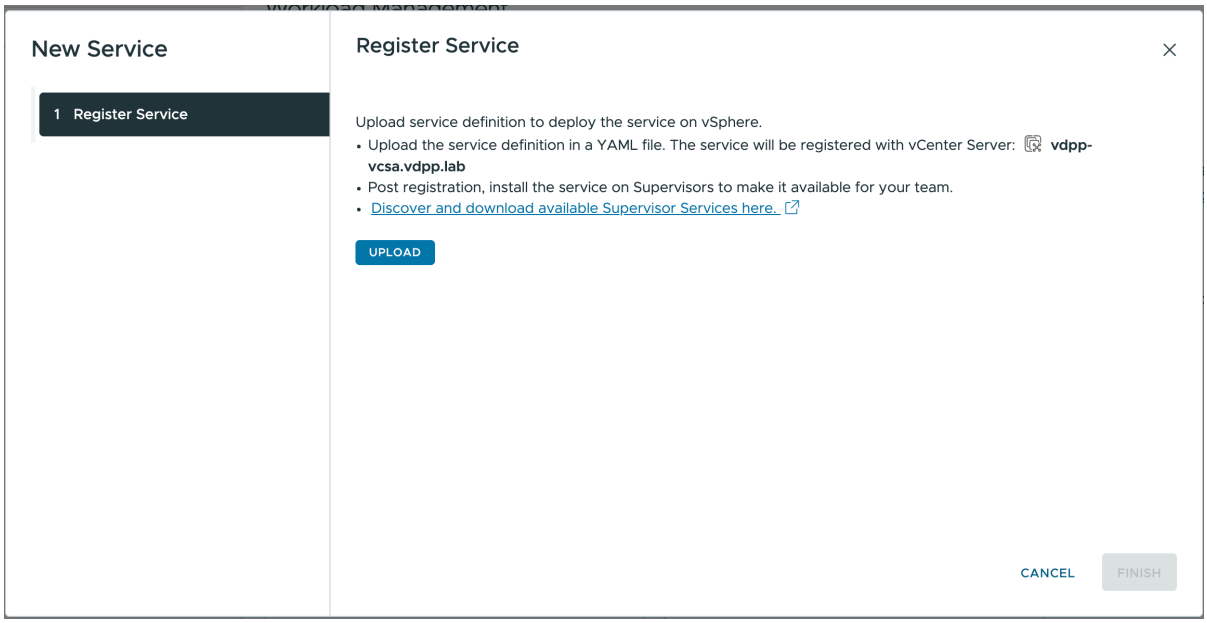
4. **The Supervisor must trust the registry's certificate.** A certificate from a private CA has to be added to the Supervisor before the install.

1. Register the service

1. Check the definition: shasum -a 256 minio-service-definition-2.3.0.yaml must print the sha256 above.
2. In the vSphere Client, open **Menu → Workload Management → Services**. The **Supervisor Services** page lists one card per registered service, and the **Add New Service** card.



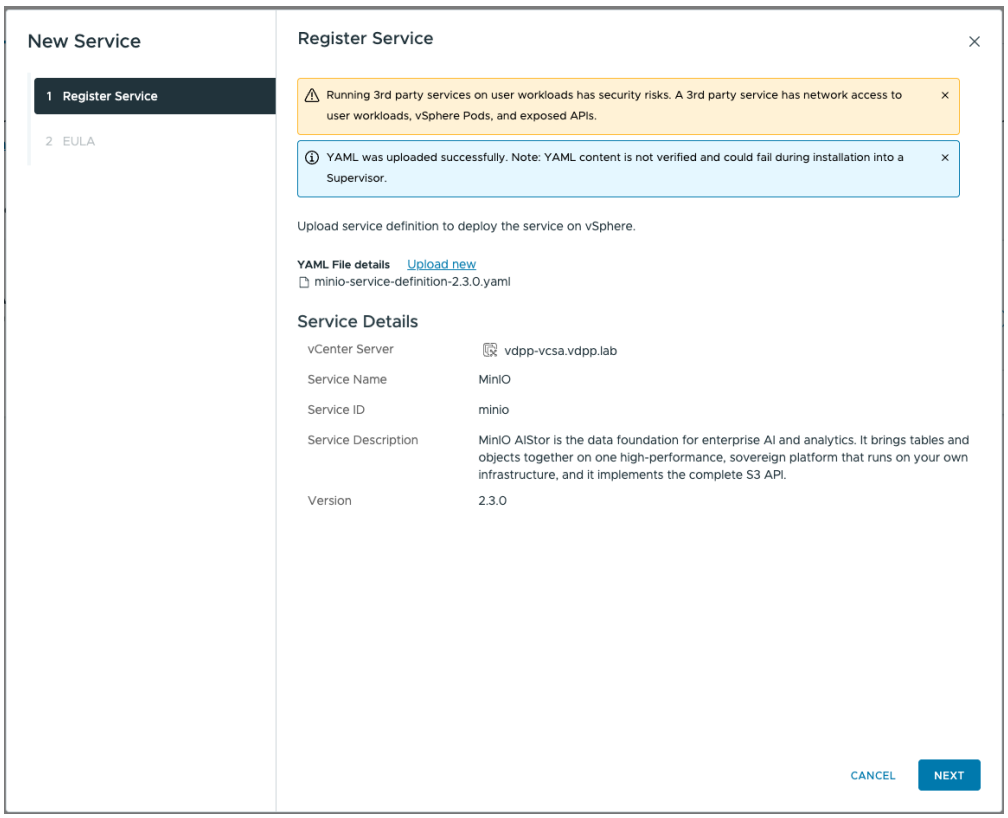
3. Click **ADD** on the **Add New Service** card, or drop the YAML file on it. The **New Service** wizard opens on **Register Service**, with an **UPLOAD** button.



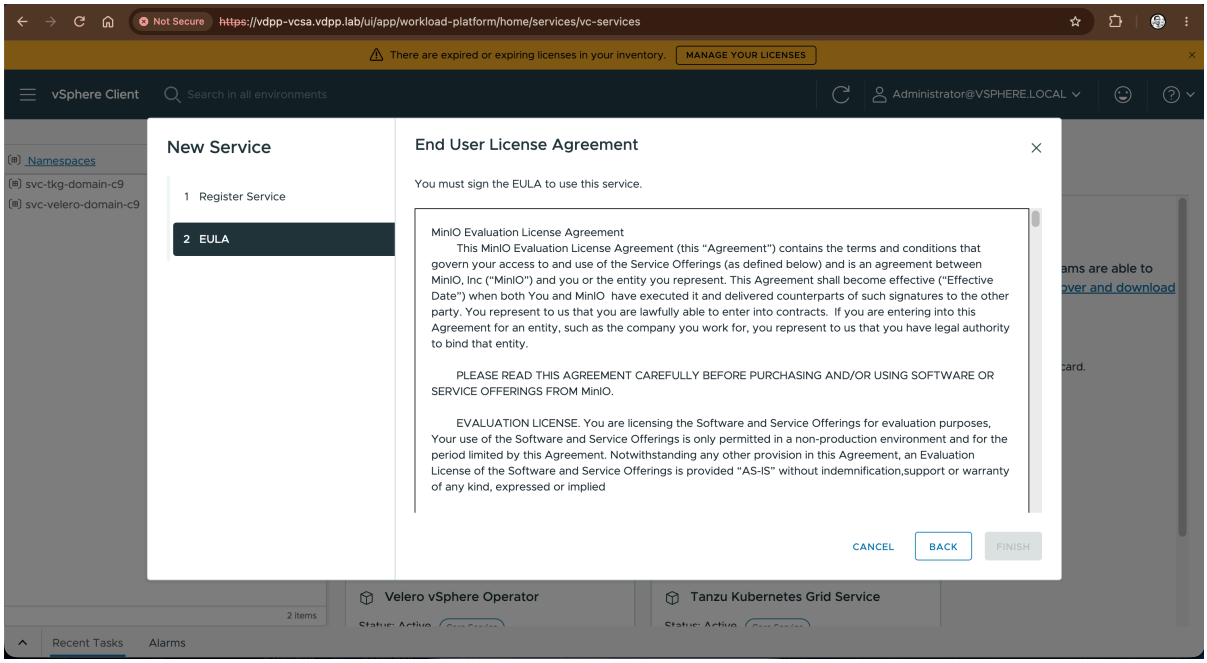
4. Click **UPLOAD** and select minio-service-definition-2.3.0.yaml. The step warns that third-party services have network access to user workloads, confirms "YAML was uploaded successfully" (vCenter does not verify its content), and shows **Service Details**:

   - **vCenter Server**: your vCenter;
   - **Service Name MinIO**, **Service ID minio**;
   - **Service Description**: "MinIO AIStor is the data foundation for enterprise AI and analytics. It brings tables and objects together on one high-performance, sovereign platform that runs on your own infrastructure, and it implements the complete S3 API.";
   - **Version 2.3.0**.

   Click **NEXT**.



5. **EULA**: read the license agreement carried in the definition, accept it, and click **FINISH**.

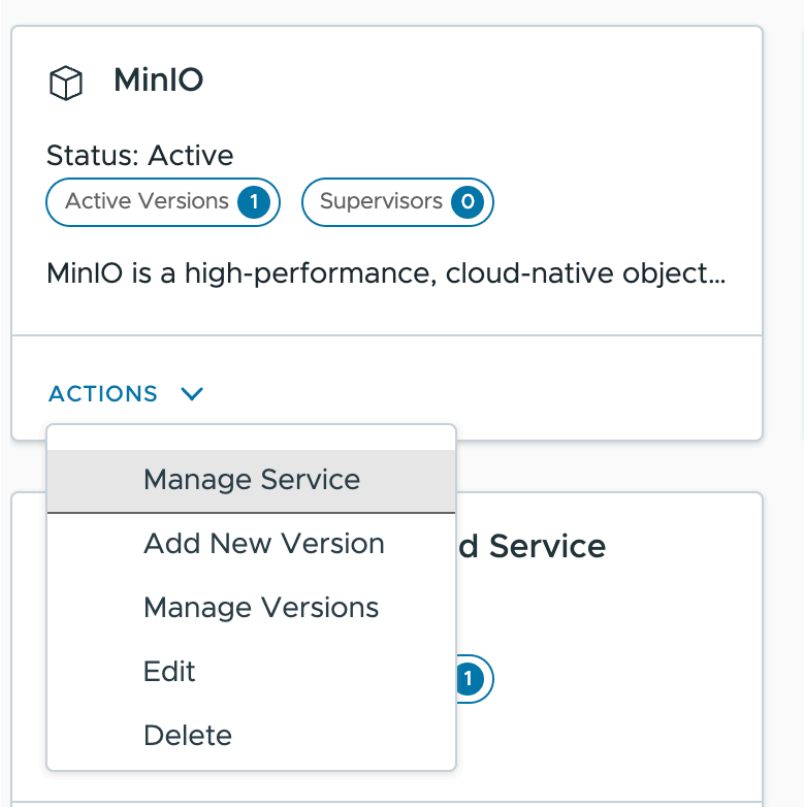


The Services page confirms **"Service 'MinIO' is successfully registered."** and shows the **MinIO** card. If an earlier MinIO version is already registered, add 2.3.0 to its card with **ACTIONS → Add New Version** instead of **Add New Service**; the upload and EULA steps are the same.

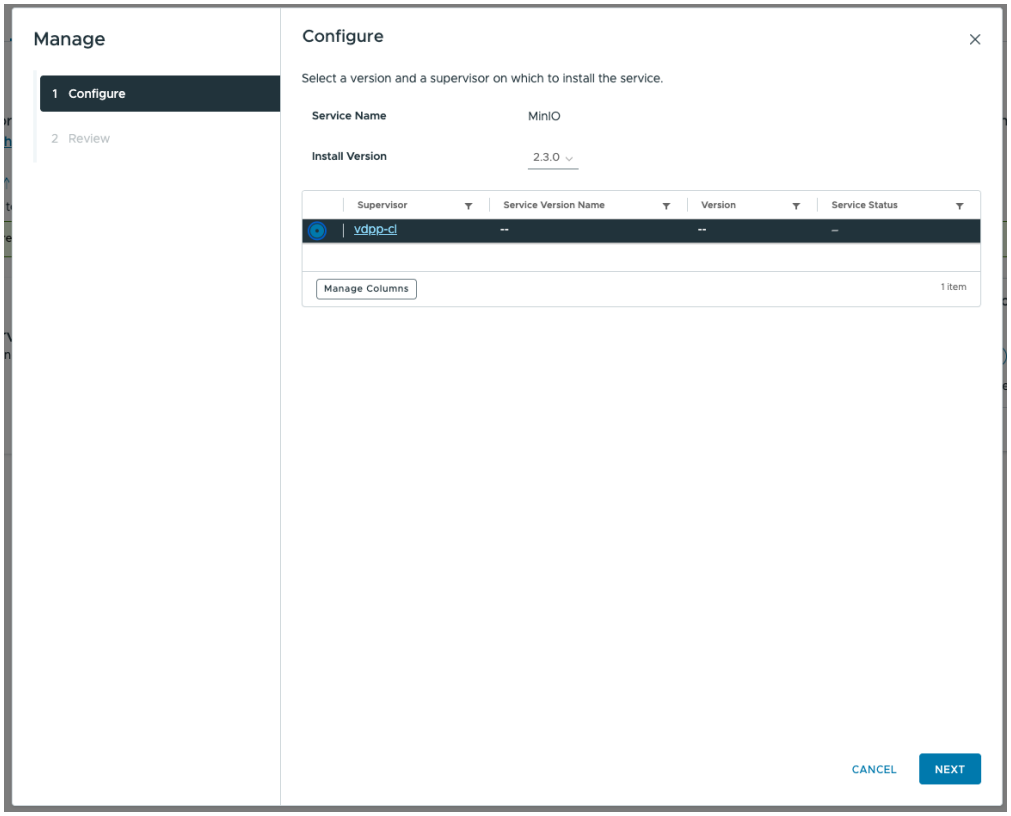
Registering only stores the definition in vCenter: nothing runs on a Supervisor until the service is installed on it.

2. Install it on the Supervisor

1. On the **MinIO** card, open **ACTIONS** → **Manage Service**.



2. **Configure** ("Select a version and a supervisor on which to install the service"): **Install Version 2.3.0**, select the Supervisor's radio button (here vdpp-cl; its **Service Version Name**, **Version** and **Service Status** are empty while MinIO is not installed on it), **NEXT**.

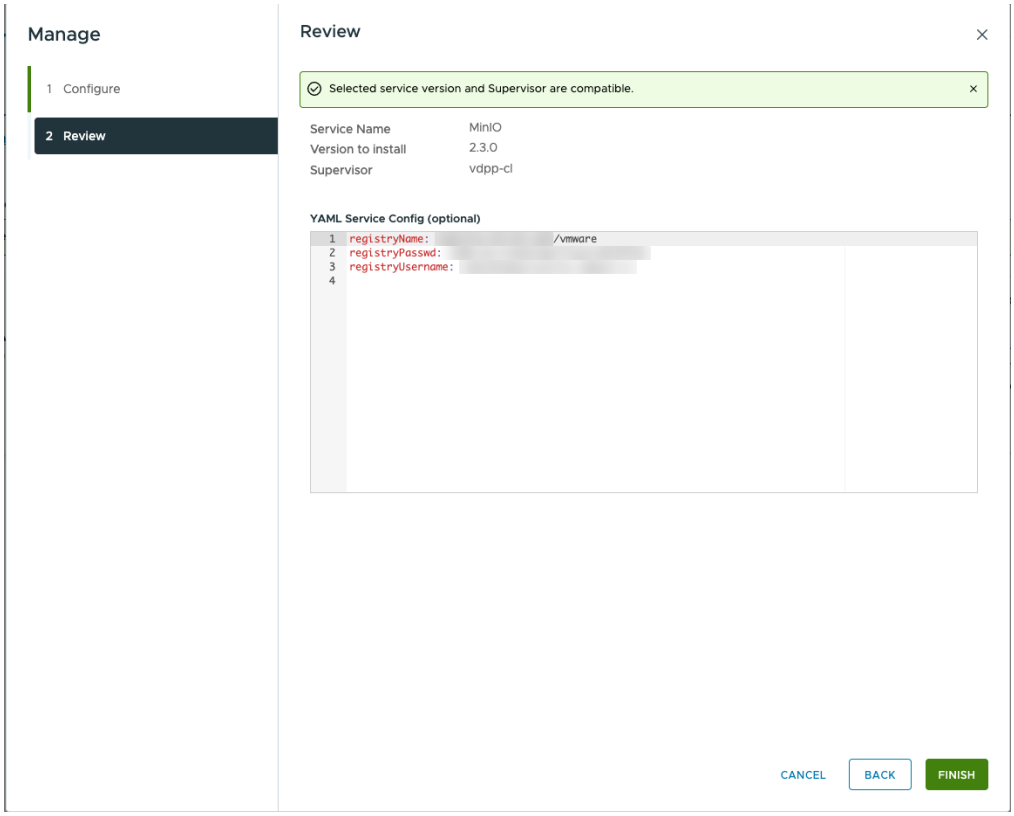


3. **Review**: check that it reports **"Selected service version and Supervisor are compatible."**, then enter the registry in **YAML Service Config (optional)**. The **Configure** step does not ask for it, and the YAML Service Config is the only place it goes. Despite the "(optional)" label, it is required: the operator and plug-in images come from your private registry, and the deployments only use the <prefix>-regcred pull secret when registryUsername is set.

```
registryName: registry.example.com/minio
registryUsername: <robot account>
registryPasswd: <its password>
# optional: the tenants' default image, instead of the public AIStor image
# minioImage: registry.example.com/minio/aistor:RELEASE.2026-09-19T17-05-25Z
```

> **Important: registryName must include the project** (registry.example.com/minio, not registry.example.com). The definition builds the image names as <registryName>/<image>: <registryName>/operator:v5.0.21 and <registryName>/vsphere:v2.3.0. The images were copied into a project, and a registry such as Harbor keeps every repository in a project. With the host alone, the Supervisor looks for registry.example.com/operator, which does not exist, and the operator and plug-in pods stay in ErrImagePull.

- registryName is without https:// or a trailing /. The pull secret is created for the registry host, so it covers the project.
- **minioImage (optional)** sets the image new tenants run when **Use custom image** is off in the Create Tenant wizard. Without it, that default is the public AIStor image, quay.io/minio/aistor/minio:RELEASE.2026-09-19T17-05-25Z (default tenant image).
- Quote the password ('...') if it starts with a YAML special character such as *, &, ! or @.
- The keys can be in any order. Service config keys lists all of them.



In this lab the images are in the project vmware, so registryName ends in /vmware; the registry host and the credentials are blurred.

4. Click **FINISH**.

3. Verify

1. **Workload Management → Supervisors → <supervisor> → Configure → Supervisor Services → Overview**: MinIO, version **2.3.0**, status **Configured**.

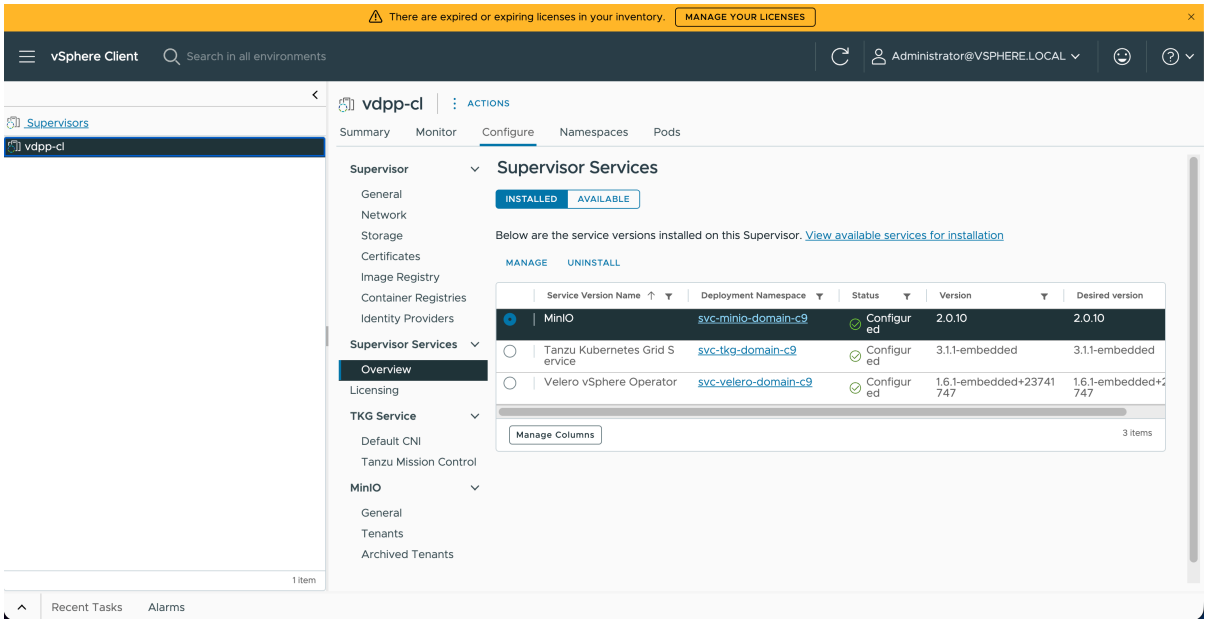
2. In the service namespace svc-minio-domain-c<N>:

```
kubectl get pods -n svc-minio-domain-c<N> \
  -o custom-
columns=NAME:.metadata.name,STATUS:.status.phase,IMAGES:.spec.containers[*].image
```

Expected: …-minio-operator-…, …-console-…, three …-vsphere-… and …-vsphere-controller-… **Running**, on <registryName>/operator:v5.0.21 and <registryName>/vsphere:v2.3.0; a …-regcred secret in the namespace.

3. Storage policies created by the service: minio-vsan-direct-thick and minio-vsan-sna-thick.

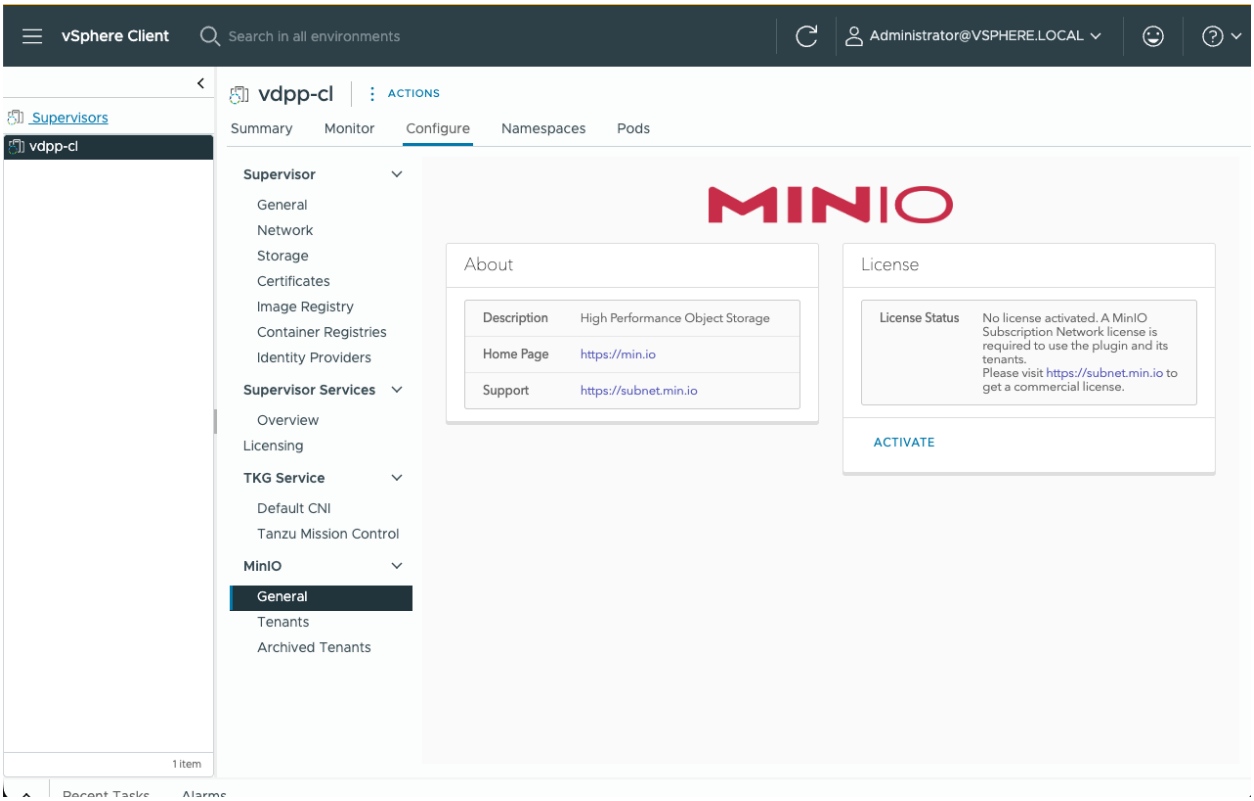
4. Log out of the vSphere Client and in again. The MinIO plug-in appears under **Workload Management → Supervisors → <supervisor> → Configure → MinIO**, with **General**, **Tenants** and **Archived Tenants**.



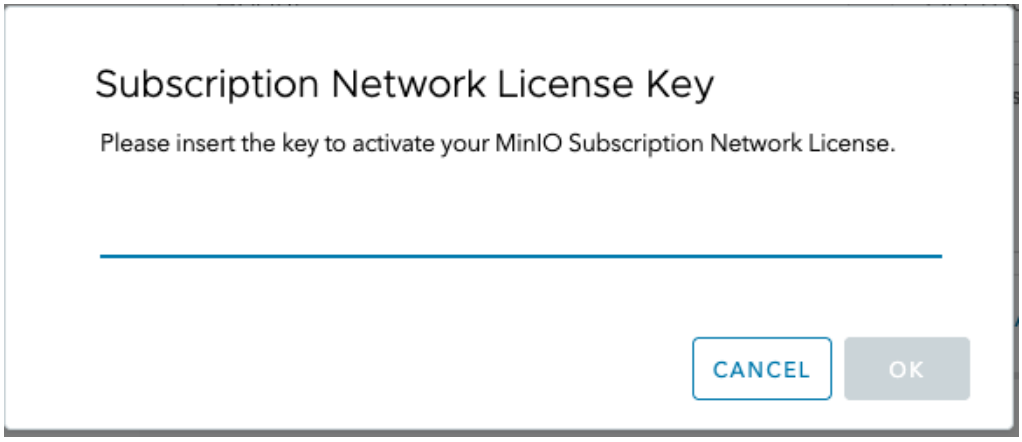
If the MinIO group does not appear, see Known issues.

4. Activate the AIStor license

MinIO → General shows **About** and **License**. Until a license is activated, **License Status** reports "No license activated. A MinIO Subscription Network license is required to use the plugin and its tenants.", and the plug-in only serves the license screens: tenants cannot be created. There is no trial period.

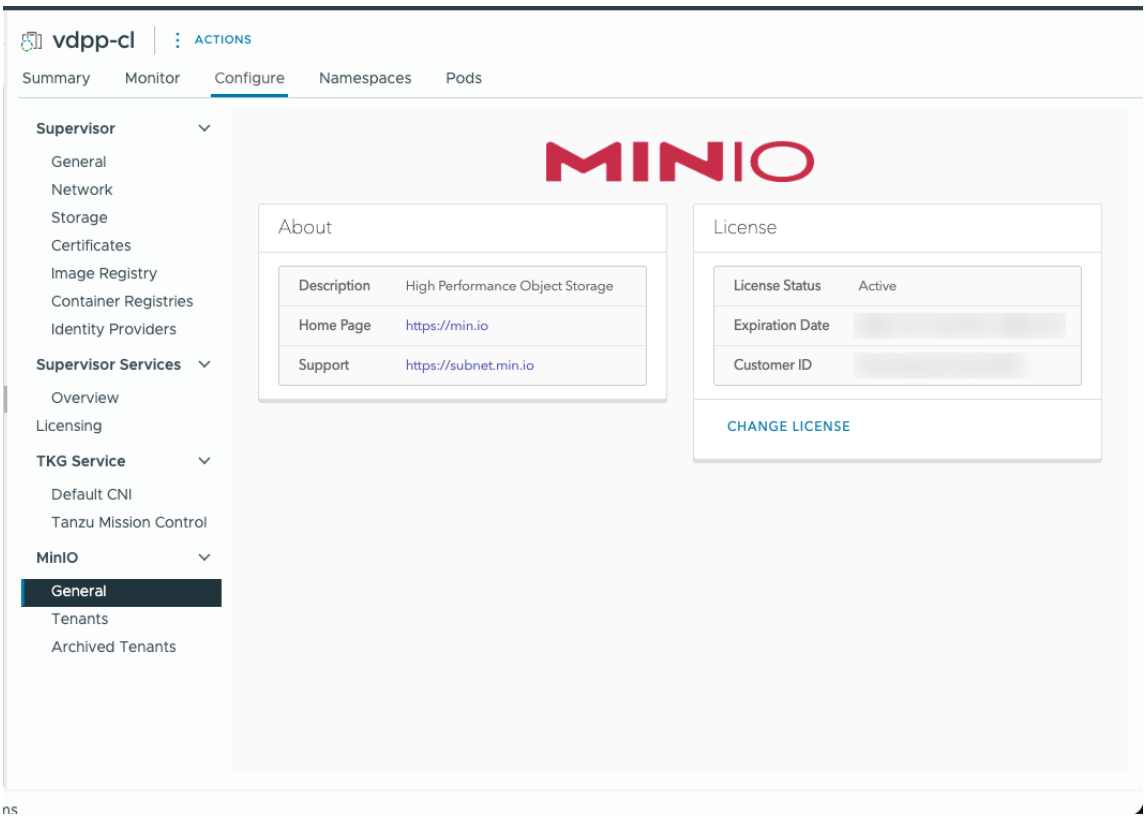


1. Click **ACTIVATE**. The **Subscription Network License Key** dialog opens ("Please insert the key to activate your MinIO Subscription Network License."); **OK** stays disabled until a key is entered.



2. Paste the **AIStor license** and click **OK**. SUBNET license keys of earlier releases are rejected.

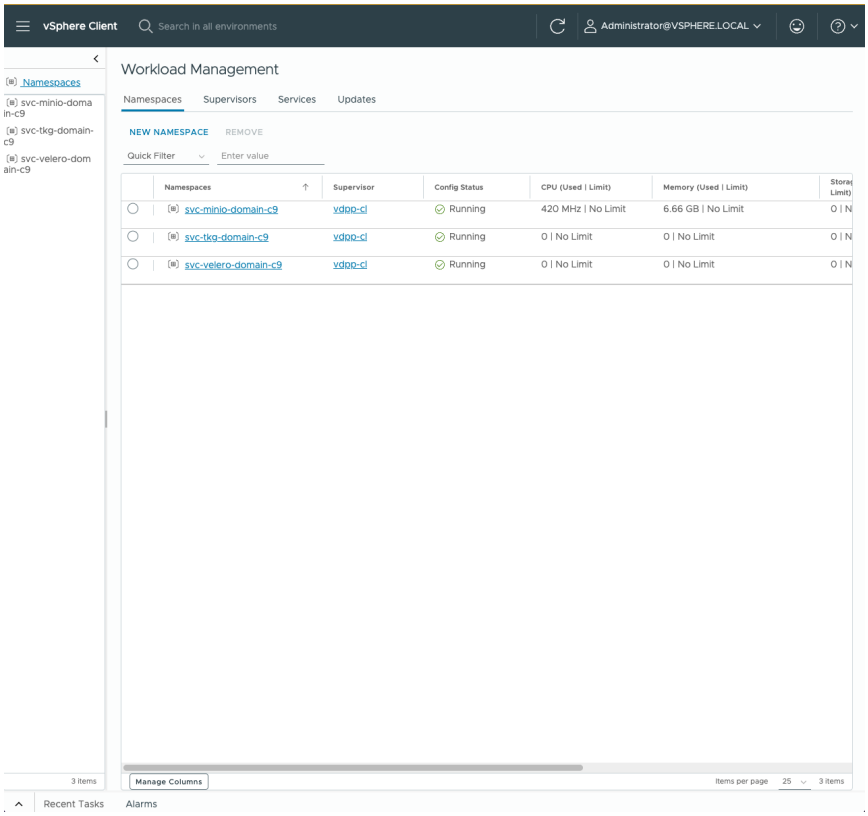
License then shows **License Status: Active**, the **Expiration Date** and the **Customer ID** from the license, and the button becomes **CHANGE LICENSE**. The key is stored in the secret minio-license-secret in svc-minio-domain-c<N>.



5. Prepare a namespace for tenants

A MinIO tenant runs in a vSphere Namespace that can use a vSAN Direct storage policy. The Create Tenant wizard asks for that namespace, so create it first.

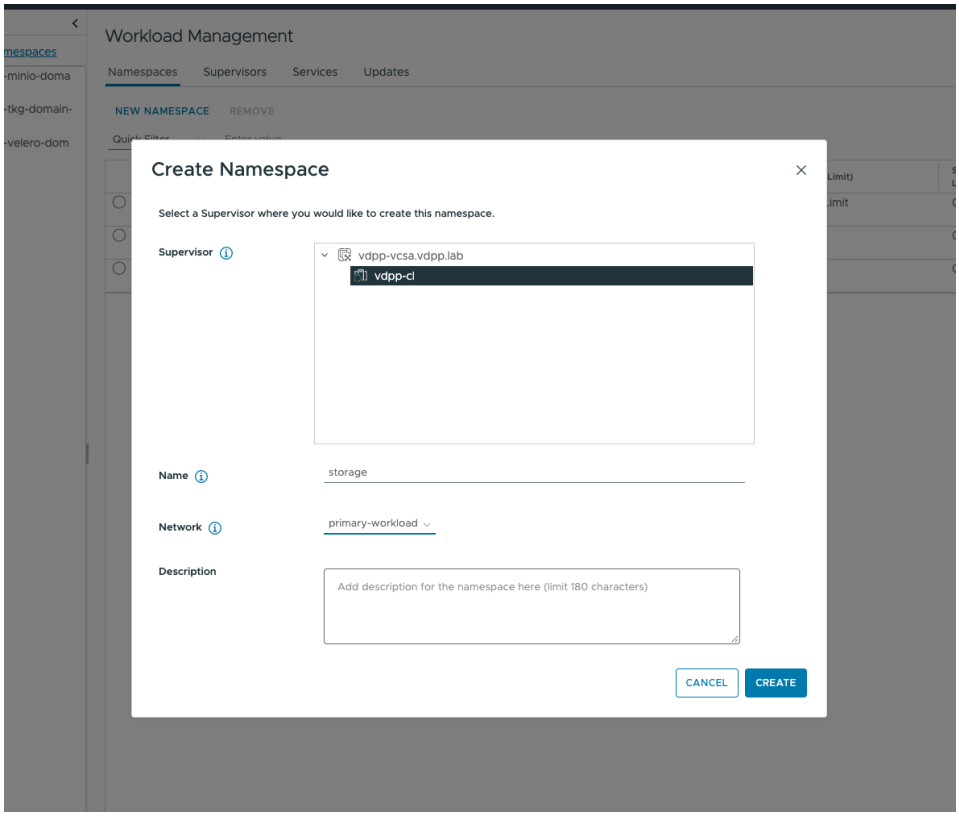
1. **Menu → Workload Management → Namespaces**. After the install it lists the service namespaces, all **Running**. Click **NEW NAMESPACE**.



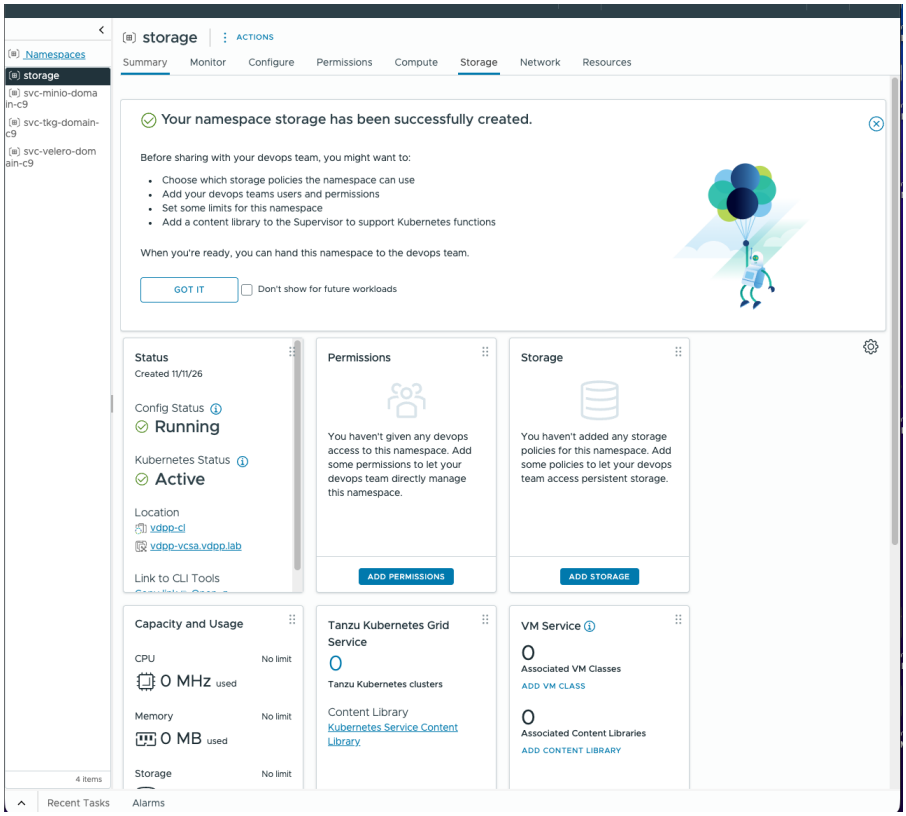
2. In **Create Namespace**:

   - **Supervisor**: the Supervisor under its vCenter (here vdpp-cl under vdpp-vcsa.vdpp.lab);
   - **Name**: e.g. storage;
   - **Network**: the workload network, here primary-workload;
   - **Description**: optional, up to 180 characters.

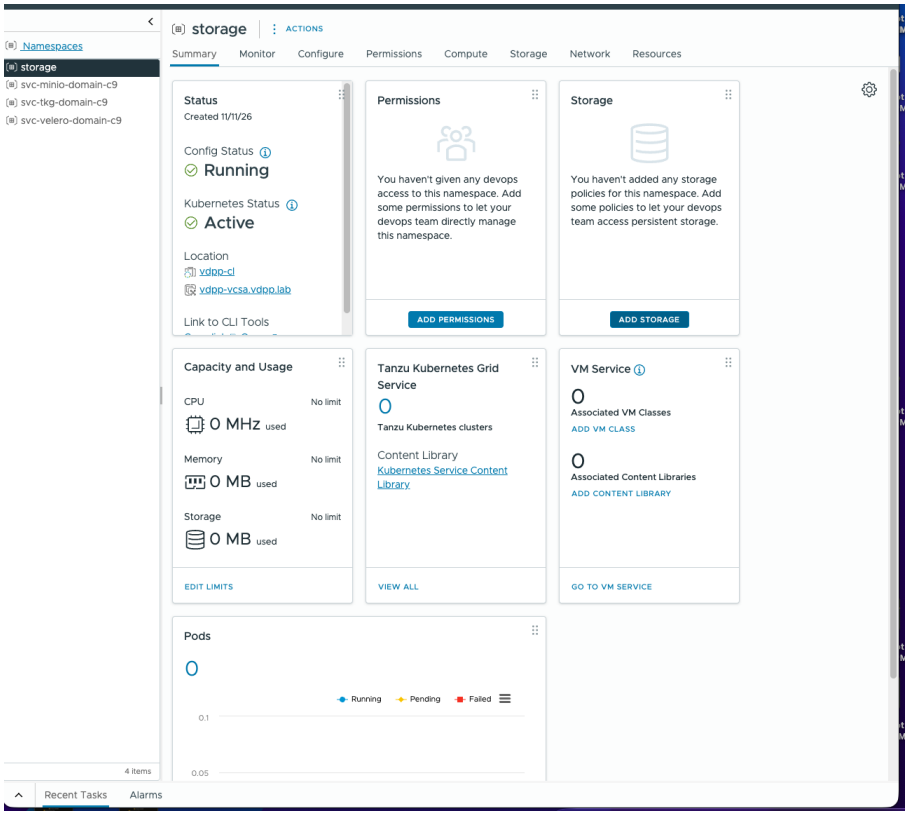
   Click **CREATE**.



3. The namespace opens on **Summary** with "Your namespace storage has been successfully created.", **Config Status: Running**, **Kubernetes Status: Active**, and empty **Permissions** and **Storage** cards. Click **GOT IT** to close the banner.



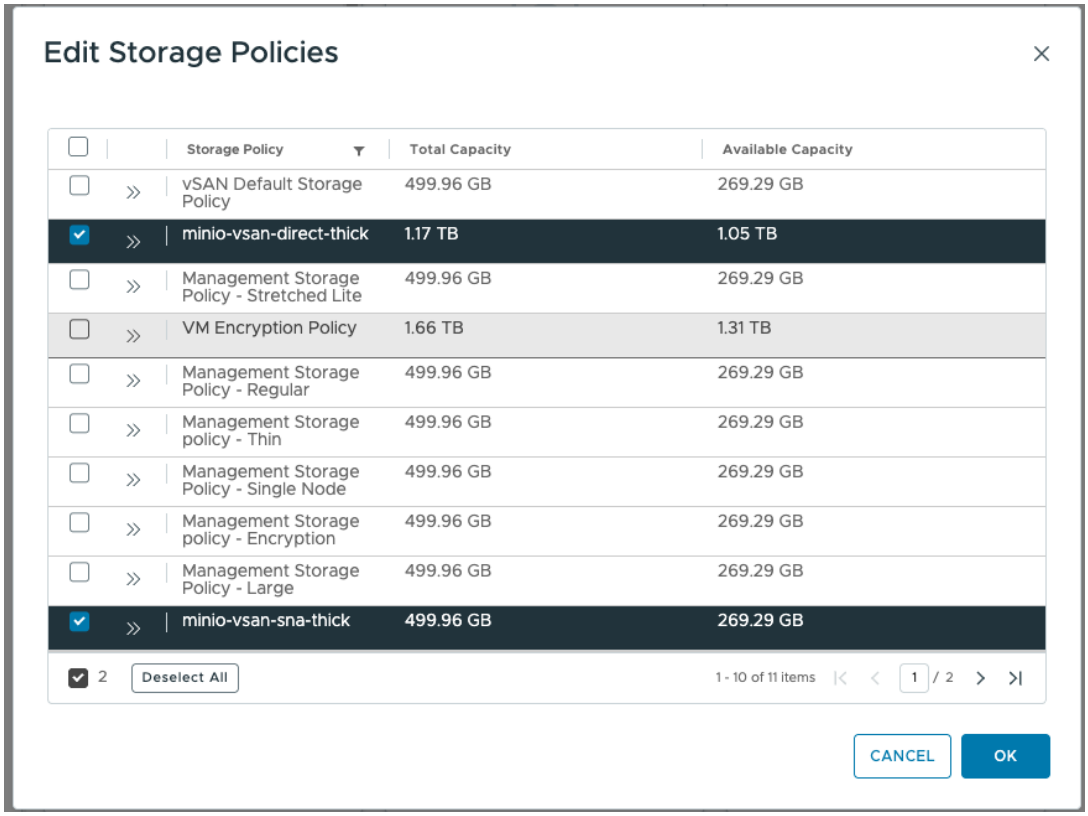
4. On the **Storage** card, click **ADD STORAGE**.



5. In **Select Storage Policies**, tick the two policies the MinIO service created, and click **OK**:

   - **minio-vsan-direct-thick**: the vSAN Direct disks, one datastore per disk; MinIO's erasure coding provides the redundancy. This is the storage class for MinIO tenants.
   - **minio-vsan-sna-thick**: the vSAN datastore, with shared-nothing placement.

   To change them later, the namespace's **Storage** card opens the same list as **Edit Storage Policies**.

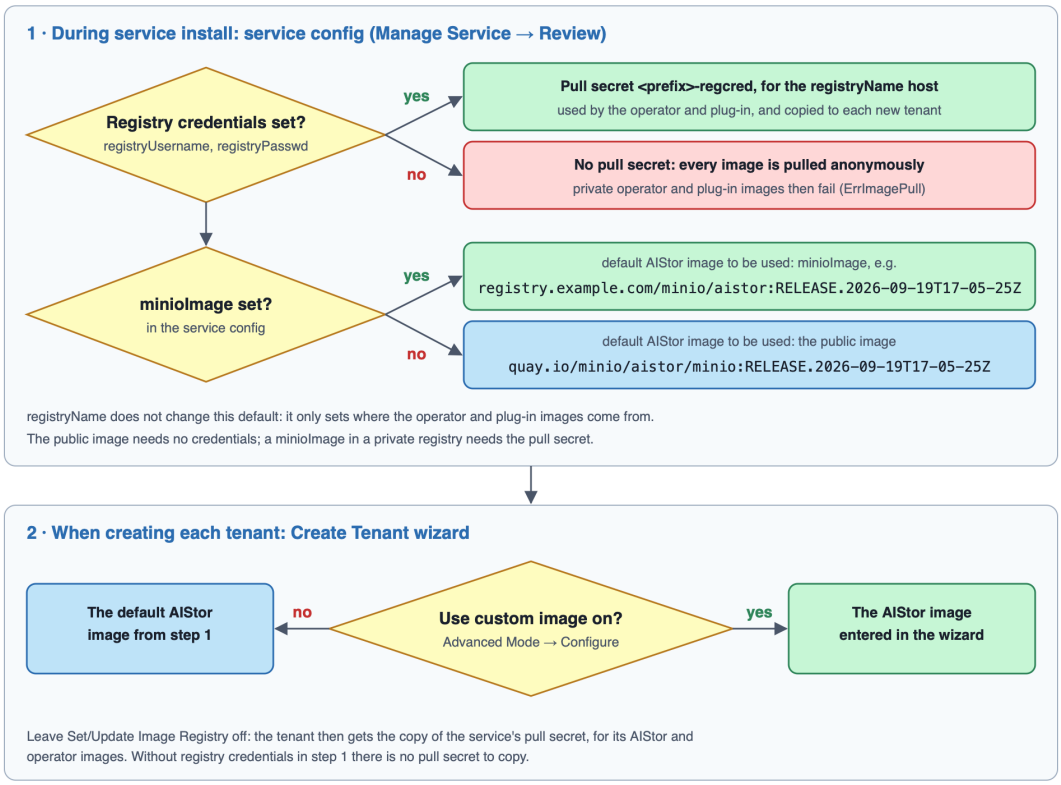


6. Optional: **Permissions** tab → **ADD** the users or groups who will create tenants, with **Can edit**. Members of vsphere.local\administrators, such as administrator@vsphere.local, already have edit rights.

The storage policies make the Kubernetes StorageClasses minio-vsan-direct-thick and minio-vsan-sna-thick usable in the namespace.

Default tenant image

Before creating tenants, check which AIStor image they will run. It is decided in two places: the service config you entered in step 2 sets the **default** (minioImage), and the Create Tenant wizard can override it per tenant (**Use custom image**):



- **Registry credentials** (registryUsername, registryPasswd) **create the pull secret** <prefix>-regcred, for the registryName host. The operator and plug-in pull with it, and the plug-in copies it into each new tenant's namespace. Without them there is no pull secret: every image is pulled anonymously, so the private operator and plug-in images fail with ErrImagePull, and a minioImage in a private registry fails too. The public AIStor image works either way.
- **minioImage only changes the tenants' default image.** It does not move the operator or plug-in images, and existing tenants keep their image.
- **registryName only moves the operator and plug-in images**, to <registryName>/operator and <registryName>/vsphere. It does not change the default tenant image.
- **The public image needs no credentials**, but the ESXi hosts must reach quay.io. Without that access, copy it into your registry project and set minioImage to the copy, e.g. registry.example.com/minio/aistor:RELEASE.2026-09-19T17-05-25Z; the tenants' pull secret, for the registry host, covers it.

To check the default on an installed service:

```
kubectl -n svc-minio-domain-c<N> get deploy minio-vsphere \
  -o jsonpath='{.spec.template.spec.containers[*].env[?
(@.name=="OPERATOR_MINIO_IMAGE")].value}'
```

6. Create a tenant

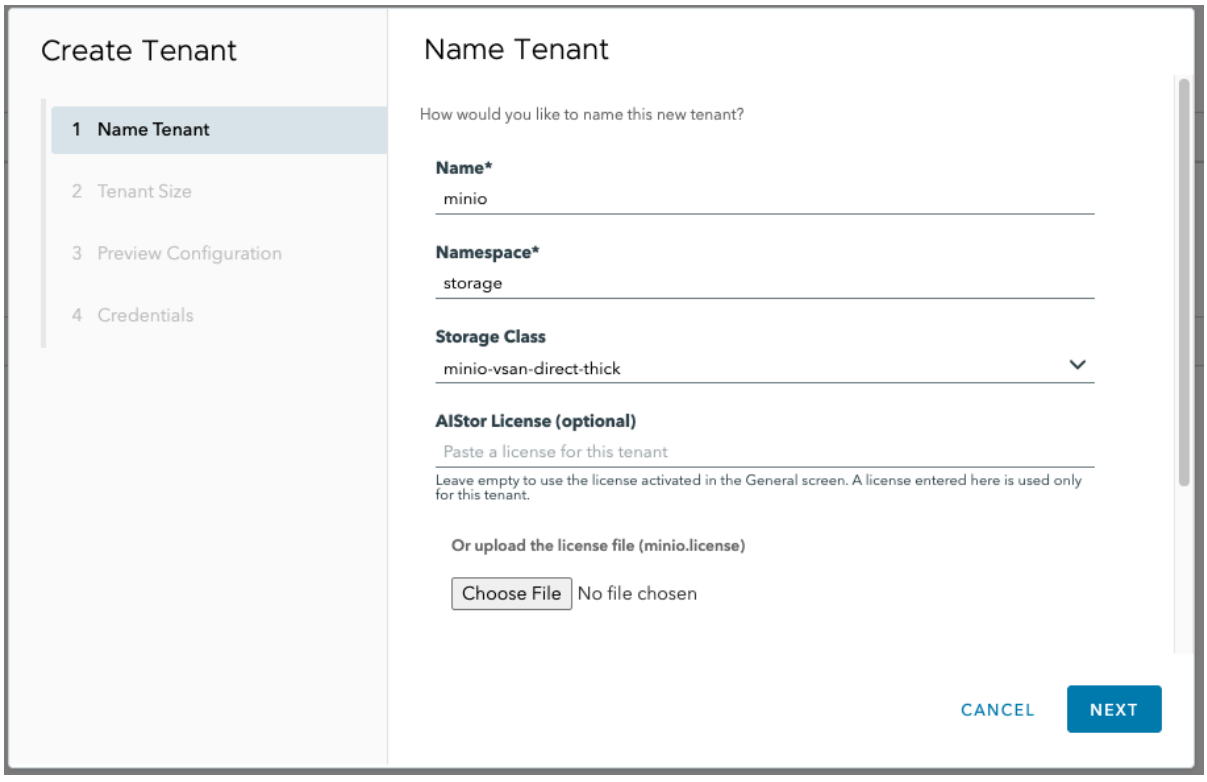
All MinIO pages are under **Workload Management → Supervisors → <supervisor> → Configure → MinIO**. **MinIO → Tenants** lists the tenants of every namespace (**Name**, **Namespace**, **Pools**, **Total Size**, **Health**). Click **ADD** to open **Create Tenant**.

The tenant **HEALTH** screenshot shows the same screen in an earlier release.

1. **Name Tenant**:

   - **Name**: the tenant name, e.g. minio;
   - **Namespace**: the namespace from step 5 (storage);
   - **Storage Class**: enabled once the namespace is entered; choose **minio-vsan-direct-thick**;
   - **AIStor License (optional)**: leave it empty to use the license activated on **General**, or paste a license for this tenant only, or upload it under **Or upload the license file (minio.license)** with **Choose File**;
   - **Advanced Mode**: off uses the secure defaults (TLS with AutoCert, built-in users, Pod Anti-affinity) and the default tenant image. On, it adds the **Configure**, **Pod Affinity**, **Identity Provider**, **Security** and **Encryption** steps.

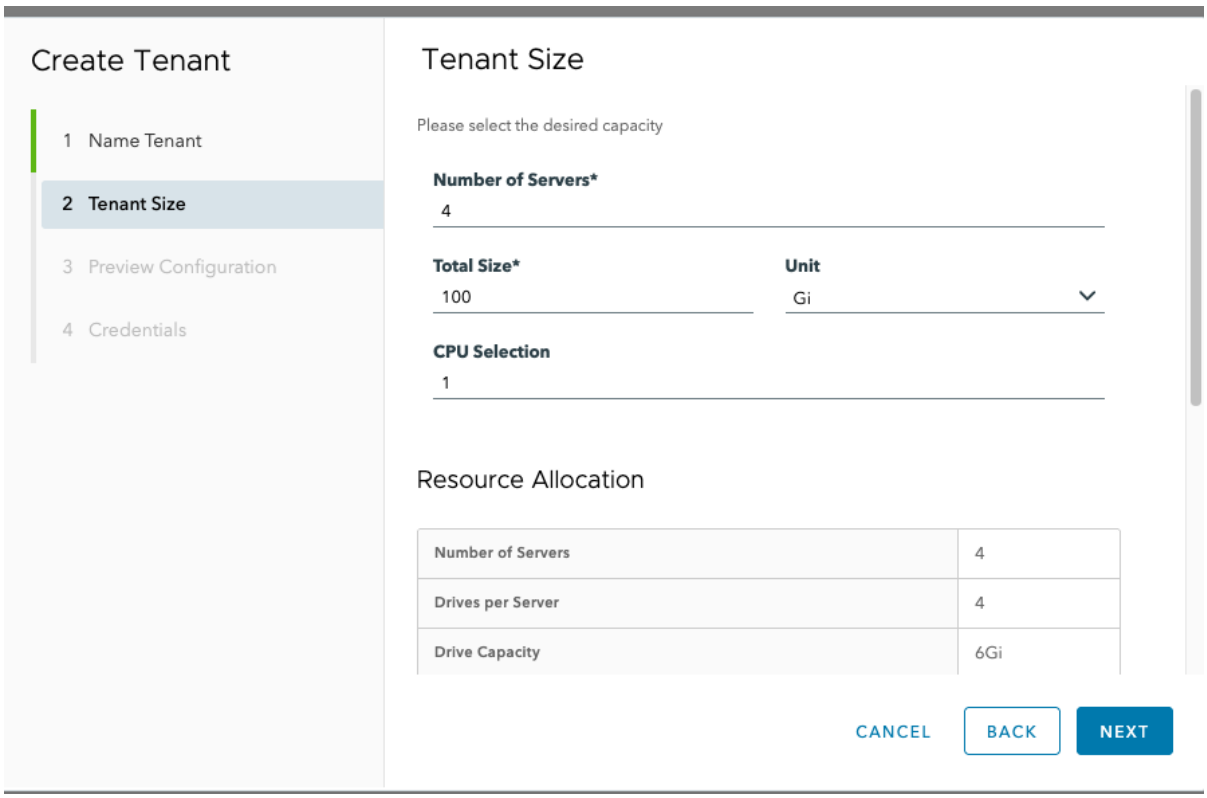
   NEXT stays disabled until Name and Namespace are filled.



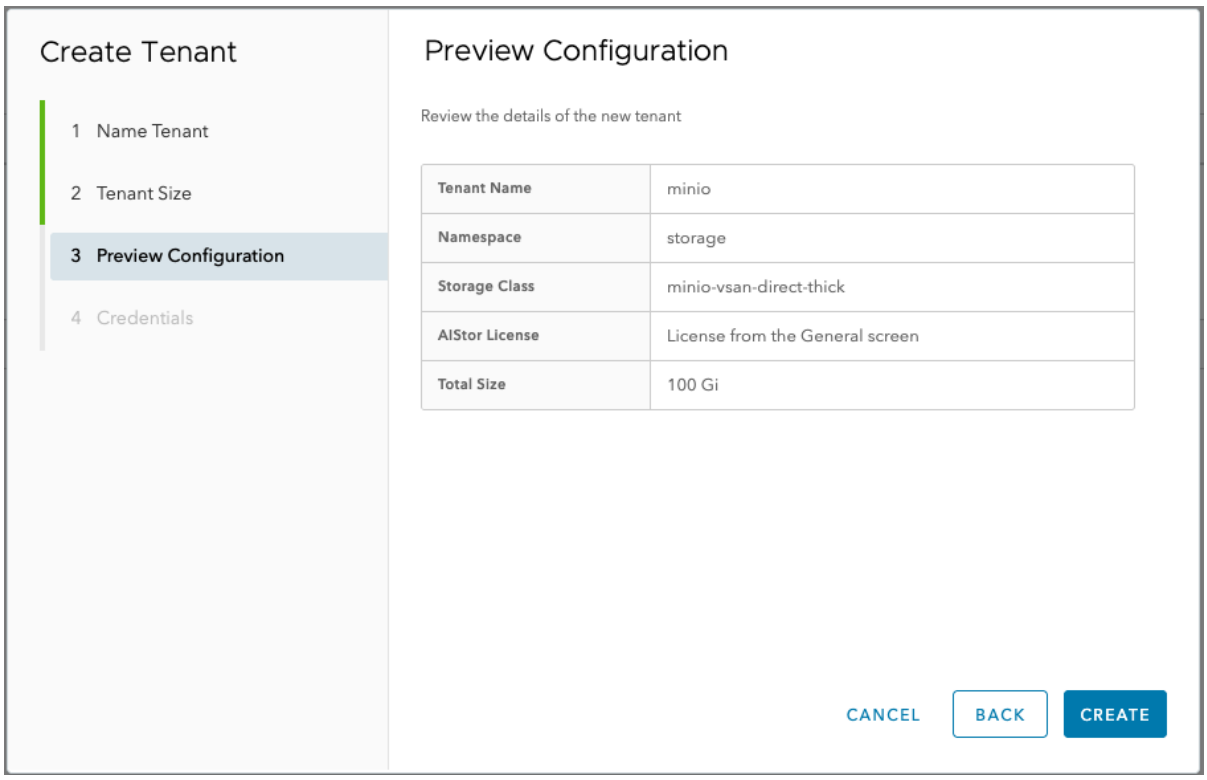
2. Advanced Mode only, **Configure**: leave **Set/Update Image Registry** off. With **Use custom image** off, the tenant runs the default tenant image; on, enter another AIStor image under **MinIO's Image**. The plug-in gives a tenant created without **Set/Update Image Registry** a copy of the service's pull secret in its namespace, and the tenant

pods pull with it: the AIStor image and the operator image they also run. A registry set on the tenant replaces that secret, and the operator image then fails to pull (Known issues).

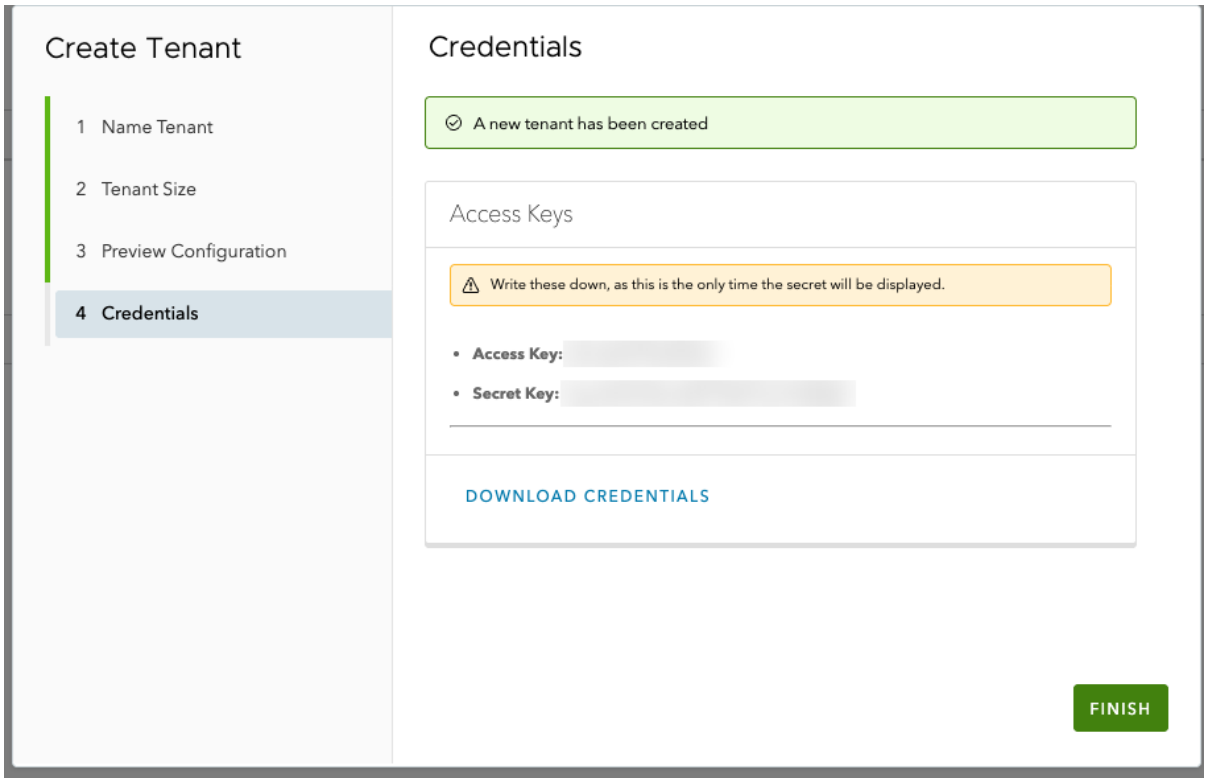
3. **Tenant Size**: **Number of Servers** (at least 4), **Total Size** with its **Unit**, and **CPU Selection**; **Resource Allocation** shows how the plug-in spreads the capacity over the vSAN Direct disks. **Lower CPU Selection** from its default, a host's whole allocatable CPU: vSphere Pods reserve what they request, and on small hosts the pods stay **Pending** with "DRS cannot find a host to power on or migrate the virtual machine". Advanced Mode also sets **Memory per Node** and the **Erasure Code Parity** here.



4. **Preview Configuration**: Tenant Name, Namespace, Storage Class, **AIStor License** ("License from the General screen", or the tenant's own) and Total Size. Click **CREATE**.

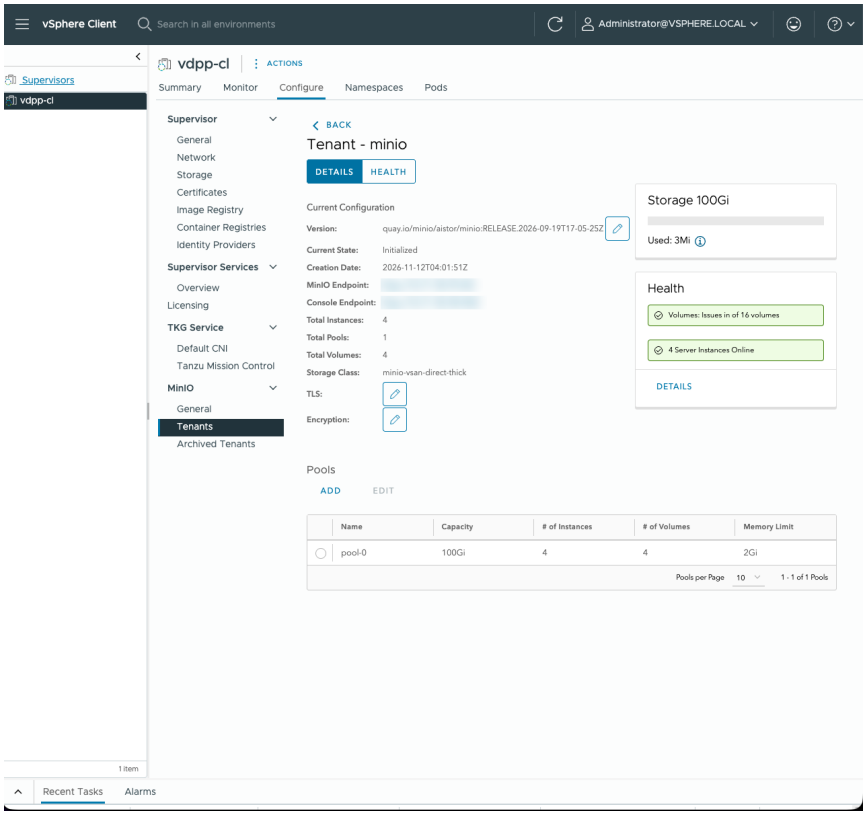


5. **Credentials**: "A new tenant has been created", with the tenant's **Access Key** and **Secret Key**. They are shown only this once: note them or use **DOWNLOAD CREDENTIALS**, then **FINISH**.

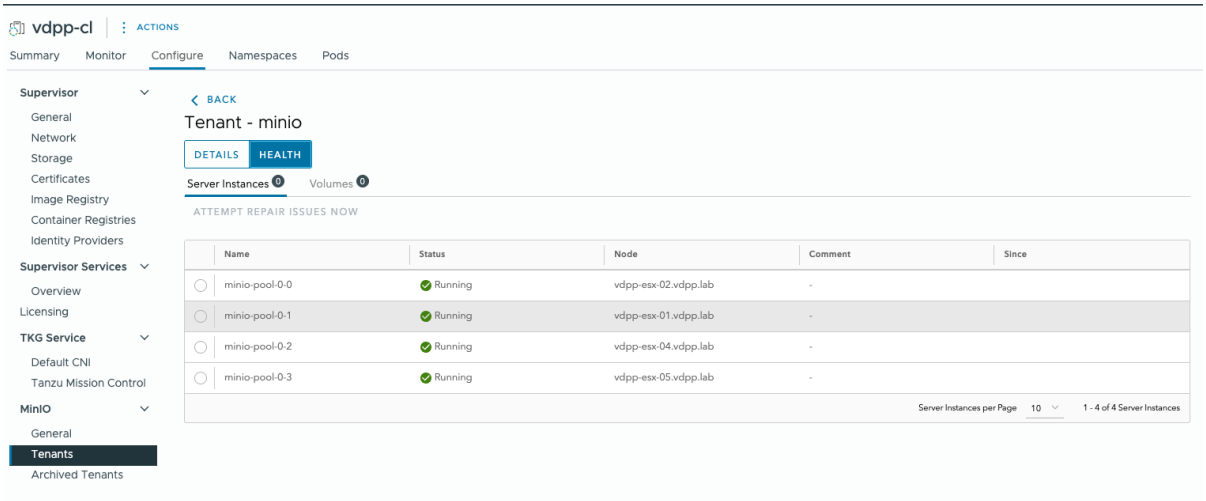


6. **MinIO → Tenants → <tenant>** opens the tenant page. **DETAILS** shows the AIStor image (**Version**), the state (**Provisioning initial users** right after the creation, **Initialized** once it runs), the **MinIO Endpoint** and **Console Endpoint**, the pools and the storage class. **HEALTH** lists the server instances and volumes: a healthy tenant shows one **Running** instance per ESXi host.

   The endpoints are listed in the vSphere Client on the tenant's **DETAILS** page (**Workload Management → Supervisors → <supervisor> → Configure → MinIO → Tenants → <tenant>**), as **MinIO Endpoint** (S3 API, port 443) and **Console Endpoint** (port 9443). They are external addresses that the Supervisor's load balancer gives the tenant's services. With kubectl, the same addresses are the EXTERNAL-IP of the tenant's services: kubectl get svc -n <namespace>.



   Here **Version** is the public AIStor image, the default when the service config has no minioImage.



A tenant without its own license uses the plug-in's: its secret <tenant>-aistor-license has the label vsphere.min.io/aistor-license-source: plugin (tenant when a license was entered in the wizard). AIStor without a valid license runs in offline mode and denies all S3 operations.

Service config keys

The **YAML Service Config (optional)** editor, on the **Review** step of **ACTIONS → Manage Service**, accepts these keys:

Key	Default	Effect
registryName	the quay.io images	registry and project holding the operator and plug-in images: <registryName>/operator, <registryName>/vsphere
registryUsername	—	with registryPasswd, creates the pull secret <prefix>-regcred for the registry host, used by every deployment and copied to tenants
registryPasswd	—	password for registryUsername
minioImage	quay.io/minio/aistor/minio:RELEASE.2026-09-19T17-05-25Z	the tenants' default AIStor image
rebuildTimer	360	passed to the plug-in controller
enableAutoFix	off	passed to the plug-in controller

Removing the service

1. Delete the MinIO tenants first.
2. **Supervisors → <supervisor> → Configure → Supervisor Services → Overview → MinIO → Uninstall**.
3. To remove the definition from vCenter: **Services → MinIO → ACTIONS → Manage Versions** to deactivate the versions, then **ACTIONS → Delete**.

Known issues

- **A tenant with its own image registry cannot pull the operator image.** The MinIO Operator adds its own image to every tenant pod (the sidecar container and the validate-arguments init container), from the service's registry. A tenant takes a single pull secret: with **Set/Update Image Registry** on, the plug-in gives it only the tenant registry's credentials instead of the service's, and the tenant pods stay in ErrImagePull with 401 UNAUTHORIZED from the service's registry. Use the default tenant image or an image from the service's registry, and leave **Set/Update Image Registry** off.
- **The MinIO group does not appear in the vSphere Client** although the service is **Configured**. The vSphere Client service may fail to load the plug-in: its log /var/log/vmware/vsphere-ui/logs/vsphere_client_virgo.log on the vCenter Server Appliance shows DEPLOYMENT_FAILED … io.min.vsphere … Plugin configuration with Reverse Proxy failed. Restart the vSphere Client service on the appliance, which signs out every session, then log in again:

```
service-control --restart vsphere-ui
```

- **"502 Bad Gateway" on MinIO → General.** The internal TLS certificate of the plug-in expired. Restart the plug-in deployment, wait for its pods to be Running, and reload the vSphere Client:

```
kubectl rollout restart deployment -n svc-minio-domain-c<N> minio-vsphere
```