MINIO

Install MinIO 2.0.13 on vSphere 8.0

November 2026

2.0.13 is the first working release after 2.0.10. 2.0.11 and 2.0.12 are skipped: their plug-in UI does not load in the vSphere Client, which 2.0.13 fixes (changelog).

Service definition	minio-service-definition-2.0.13.yaml, built with make supervisor-service
sha256	f31d852e88c8a0582d0a2798c51ce9eee3b9bfcc04b5035f20508e8b8baf4d92
Operator	quay.io/miniohq/operator:v5.0.19 (**needs pull credentials**)
Plug-in	quay.io/miniohq/vsphere:v2.0.13
MinIO (tenants' default)	<registryName>/minio:RELEASE.2025-09-07T16-13-09Z (**no longer provided by MinIO**: see below)

Prerequisites, the YAML Service Config keys and removal are in vsphere-8-reference.md.

Default tenant image

New tenants run the image the plug-in is configured with (OPERATOR_MINIO_IMAGE on the minio-vsphere deployment in svc-minio-domain-c<N>), unless **Use custom image** is on in the Create Tenant wizard. The definition builds it from registryName:

YAML Service Config	Default tenant image
registryName: registry.example.com/minio	registry.example.com/minio/minio:RELEASE.2025-09-07T16-13-09Z
without registryName	quay.io/minio/minio:RELEASE.2025-09-07T16-13-09Z

The 2.0.13 definition published before (sha256 c35812f7…) used minio/minio:RELEASE.2025-09-07T16-13-09Z, which resolves to Docker Hub.

MinIO no longer provides this image: neither quay.io nor Docker Hub serves it. Push a MinIO image of your own into the registry project, either under that name, so that tenants find it by default, or under another release, selected with **Use custom image** (step 5.4). 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}'
```

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:v5.0.19	registry.example.com/minio/operator:v5.0.19@sha256:…
Plug-in	quay.io/miniohq/vsphere:v2.0.13	registry.example.com/minio/vsphere:v2.0.13@sha256:…
Tenants' default MinIO image	minio/minio:RELEASE.2025-09-07T16-13-09Z, **no longer provided**	registry.example.com/minio/minio:RELEASE.2025-09-07T16-13-09Z

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. The quay.io/miniohq repositories are private: for credentials to copy from them, contact MinIO at dev@min.io or sales@min.io, or open a request in SUBNET. With skopeo, logged in to quay.io and to the private registry:

   ```
   PROJECT=registry.example.com/minio
   skopeo copy --all docker://quay.io/miniohq/operator:v5.0.19 docker://$PROJECT/operator:v5.0.19
   skopeo copy --all docker://quay.io/miniohq/vsphere:v2.0.13 docker://$PROJECT/vsphere:v2.0.13
   ```

   Check that the digests the definition pins are there (also listed in CHANGELOG.md):

   ```
   skopeo inspect --raw docker://$PROJECT/
   operator@sha256:4802b402436466bb6c37767659d47ff05c3b7b10a3d4f6f2d3f80089e7be229d \
     >/dev/null && echo ok
   skopeo inspect --raw docker://$PROJECT/
   vsphere@sha256:5f467dead6998e8b8c66ce153d288713f66780dbea82b705eb6d887b9130b73d \
     >/dev/null && echo ok
   ```



3. **The MinIO image for the tenants.**

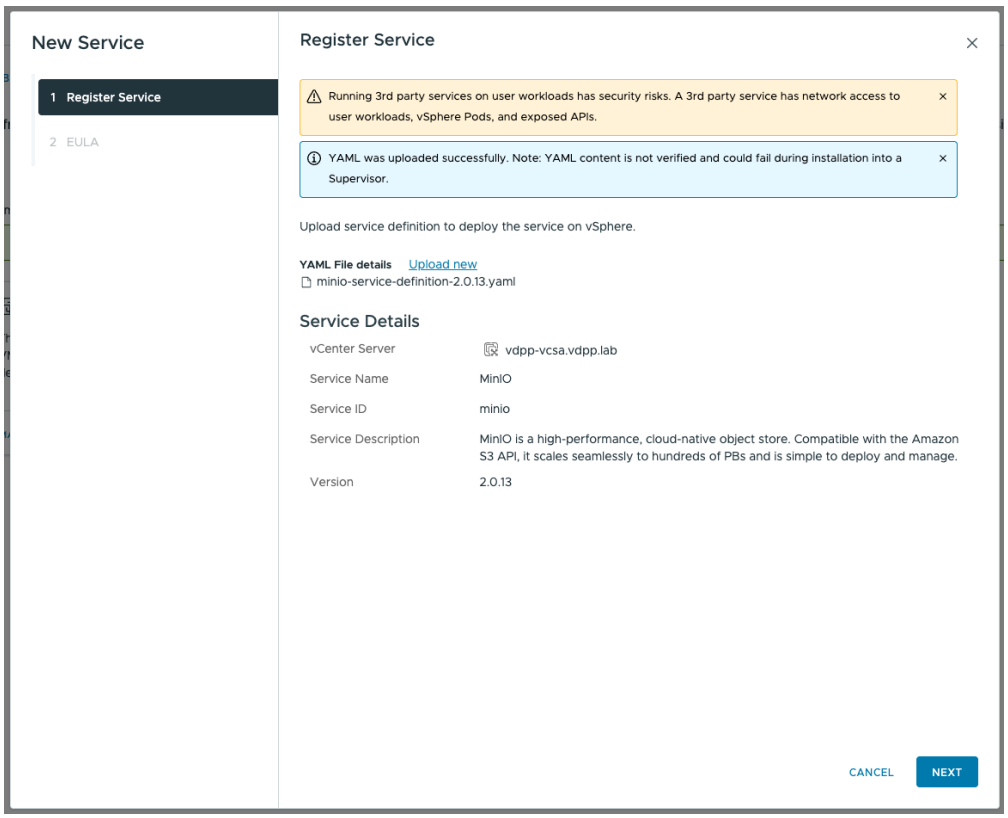
> **Warning: MinIO no longer provides the open-source minio/minio images.** Neither quay.io nor Docker Hub serves them any more, including RELEASE.2025-09-07T16-13-09Z, the 2.0.13 default. To run open-source MinIO you need a copy of your own: push it into the same project as $PROJECT/minio:<release>. Tenants run $PROJECT/minio:RELEASE.2025-09-07T16-13-09Z by default, or another release selected with **Use custom image** (step 5.4).

   The AIStor image, quay.io/minio/aistor/minio:RELEASE.2026-09-19T17-05-25Z, is public: copying it into the project is optional. It needs an AIStor license, which the plug-in manages from 2.3.0 (2.3.0 guide).

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-2.0.13.yaml must print the sha256 above.
2. In the vSphere Client, open **Menu** → **Workload Management** → **Services** (screenshot).
3. Click **ADD** on the **Add New Service** card, or drop the YAML file on it. The **New Service** wizard opens on **Register Service** (screenshot).
4. Click **UPLOAD** and select the YAML file. 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, Service Name **MinIO**, Service ID **minio**, the description and Version **2.0.13**. Click **NEXT**.



5. **EULA**: read the **MinIO Evaluation License Agreement**, accept it, and click **FINISH** (screenshots).

The Services page confirms **"Service 'MinIO' is successfully registered."** and shows a **MinIO** card: **Status: Active**, **Active Versions 1**, **Supervisors 0** (as in the 2.0.10 guide).

2. Install it on the Supervisor

1. On the **MinIO** card, open **ACTIONS → Manage Service**. This is the entry that installs the service on Supervisors (the ACTIONS menu).

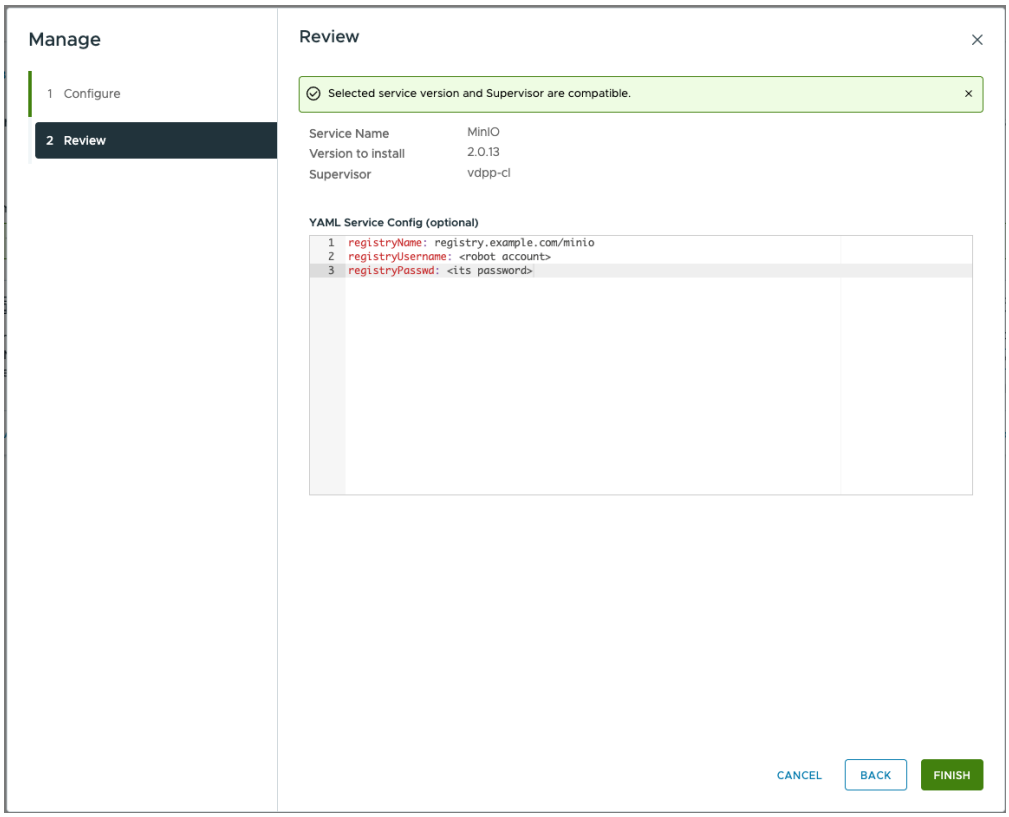
2. **Manage** wizard, **Configure** step: **Install Version 2.0.13**, select the Supervisor's radio button, **NEXT** (screenshot).

3. **Review**: check that it reports **"Selected service version and Supervisor are compatible."**, then enter the pull credentials in **YAML Service Config (optional)**. The **Configure** step does not ask for them, and the YAML Service Config is the only place they go. Despite the "(optional)" label, they are 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 (Pull credentials):

```
registryName: registry.example.com/minio
registryUsername: <robot account>
registryPasswd: <its password>
```

> **Important: registryName must include the project** (registry.example.com/minio, not registry.example.com). The definition builds every image name as <registryName>/<image>: <registryName>/operator:v5.0.19, <registryName>/vsphere:v2.0.13 and <registryName>/minio:<release> for tenants. 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 the registry **and project** holding the copied images (copy the images), without https:// or a trailing /. The pull secret is created for the registry host, so it covers the project.
- Quote the password ('...') if it starts with a YAML special character such as *, &, ! or @.
- Without these keys, the operator and plug-in pods stay in ErrImagePull.



4. Click **FINISH**.

3. Verify

1. **Workload Management → Supervisors → <supervisor> → Configure → Supervisor Services → Overview**: MinIO, version **2.0.13**, 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-…, …-vsphere-… and …-vsphere-controller-… **Running**, on <registryName>/operator:v5.0.19 and <registryName>/vsphere:v2.0.13; a …-regcred secret in the namespace.

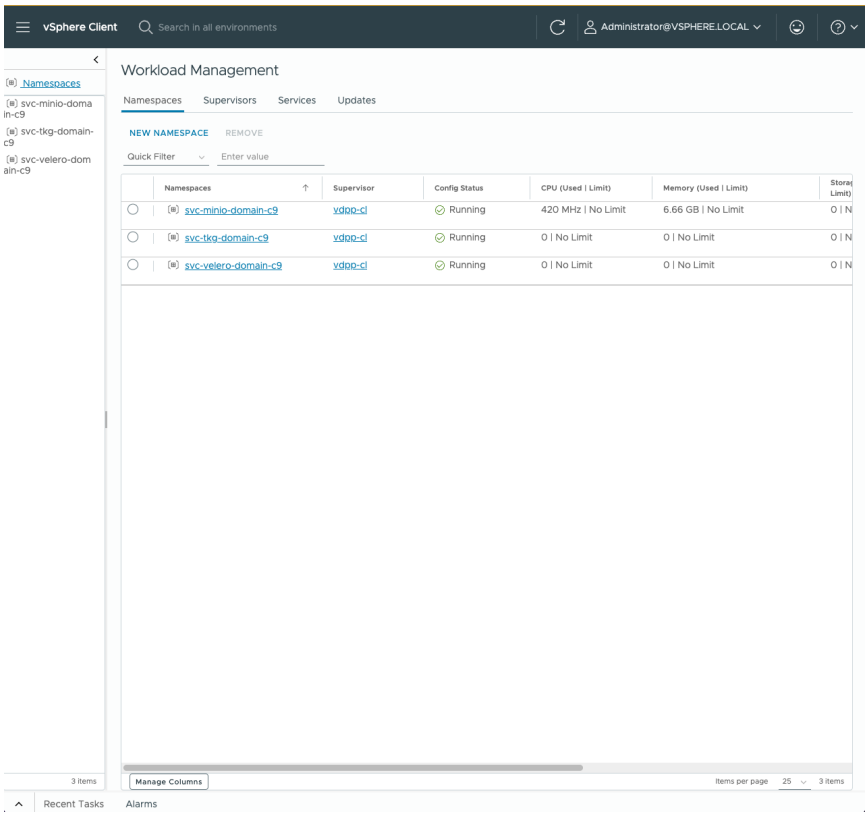
3. Storage policies created by the service: minio-vsan-direct-thick and minio-vsan-sna-thick.

4. The MinIO plug-in UI loads in the vSphere Client (log out and in): **Workload Management → Supervisors → <supervisor> → Configure → MinIO**. This is the check that fails on 2.0.11 and 2.0.12.

4. 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.

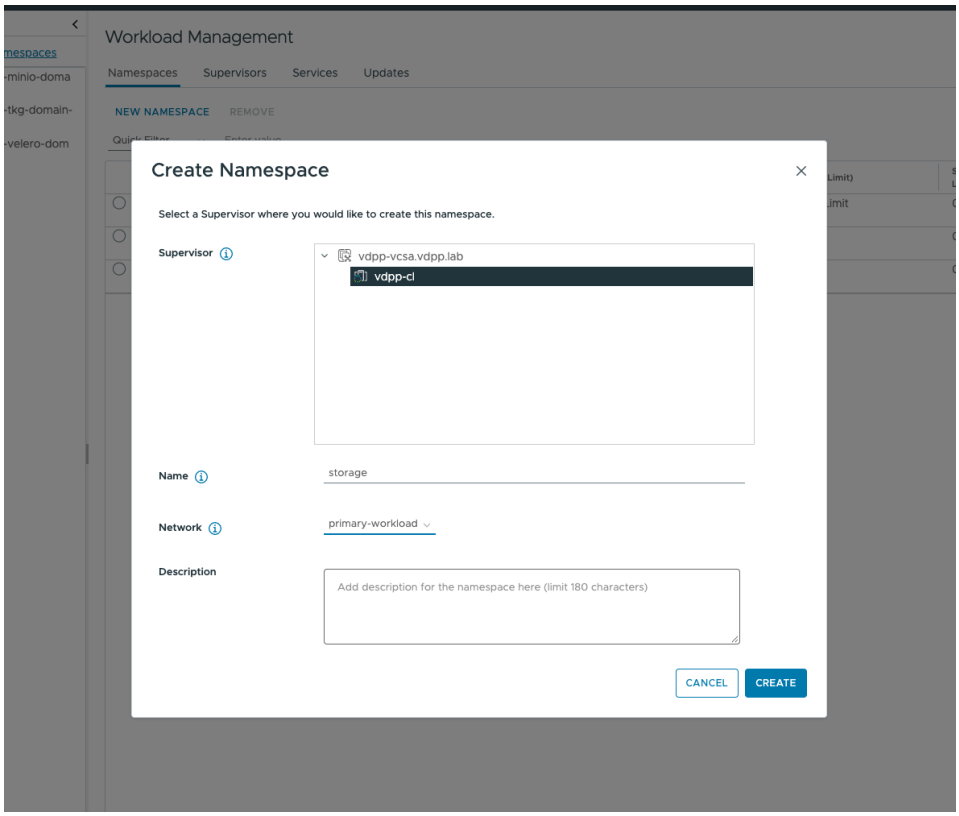
1. **Menu → Workload Management → Namespaces**. After the install it lists the service namespaces only: svc-minio-domain-c<N>, svc-tkg-domain-c<N> and svc-velero-domain-c<N>, all **Running**. Click **NEW NAMESPACE**.



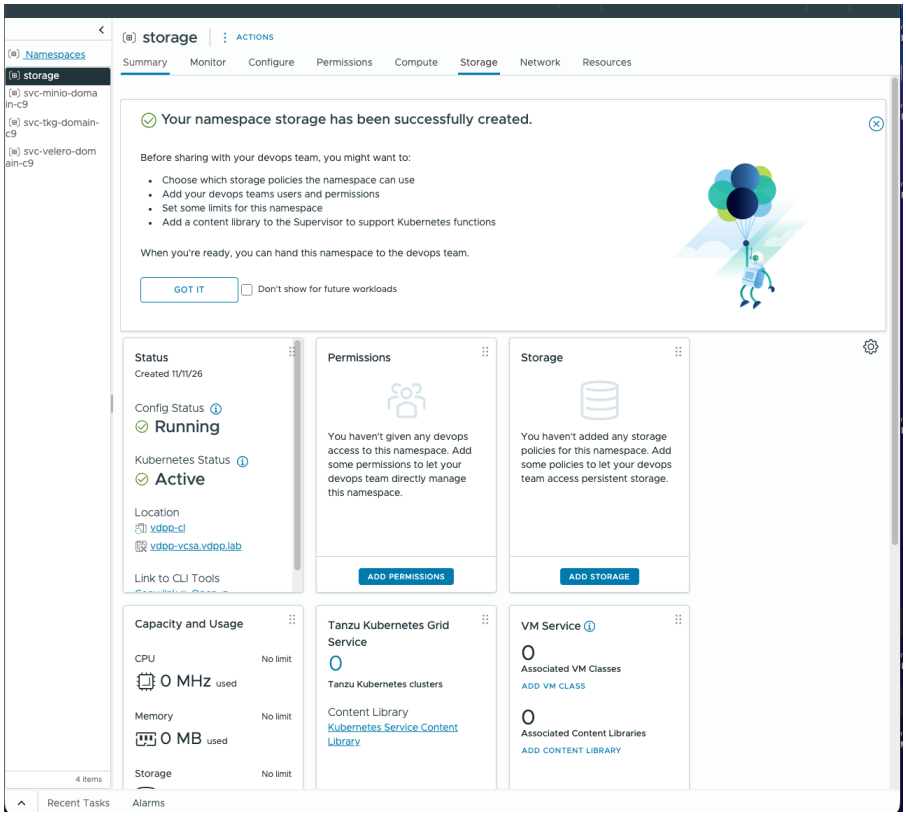
2. In **Create Namespace**:

- **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.

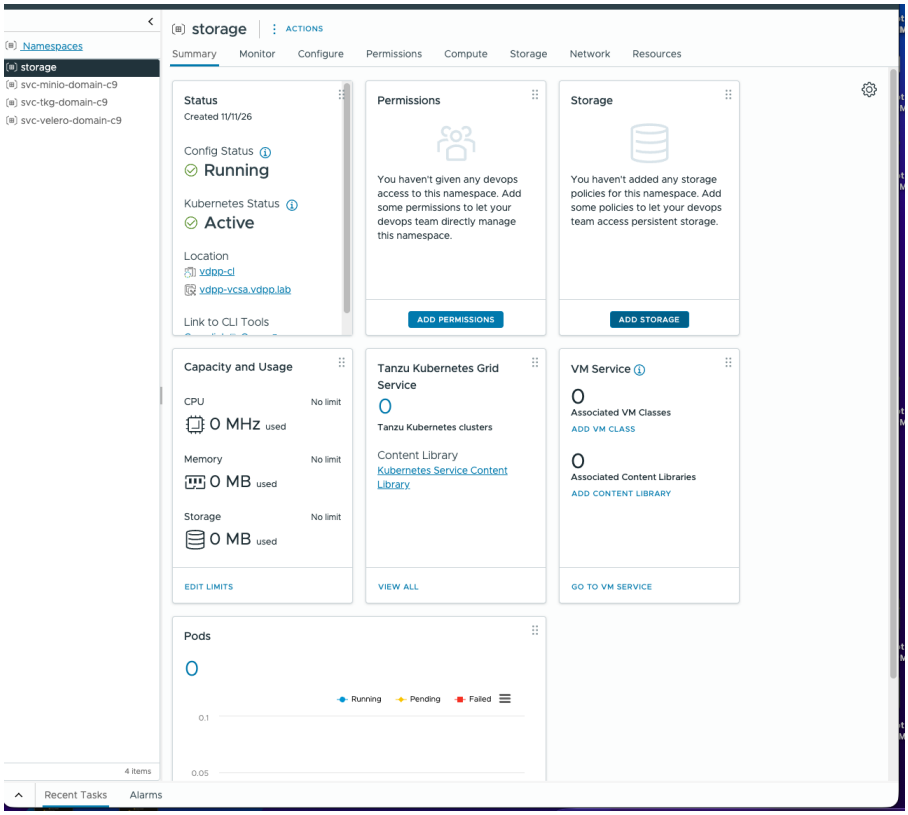
Click **CREATE**.



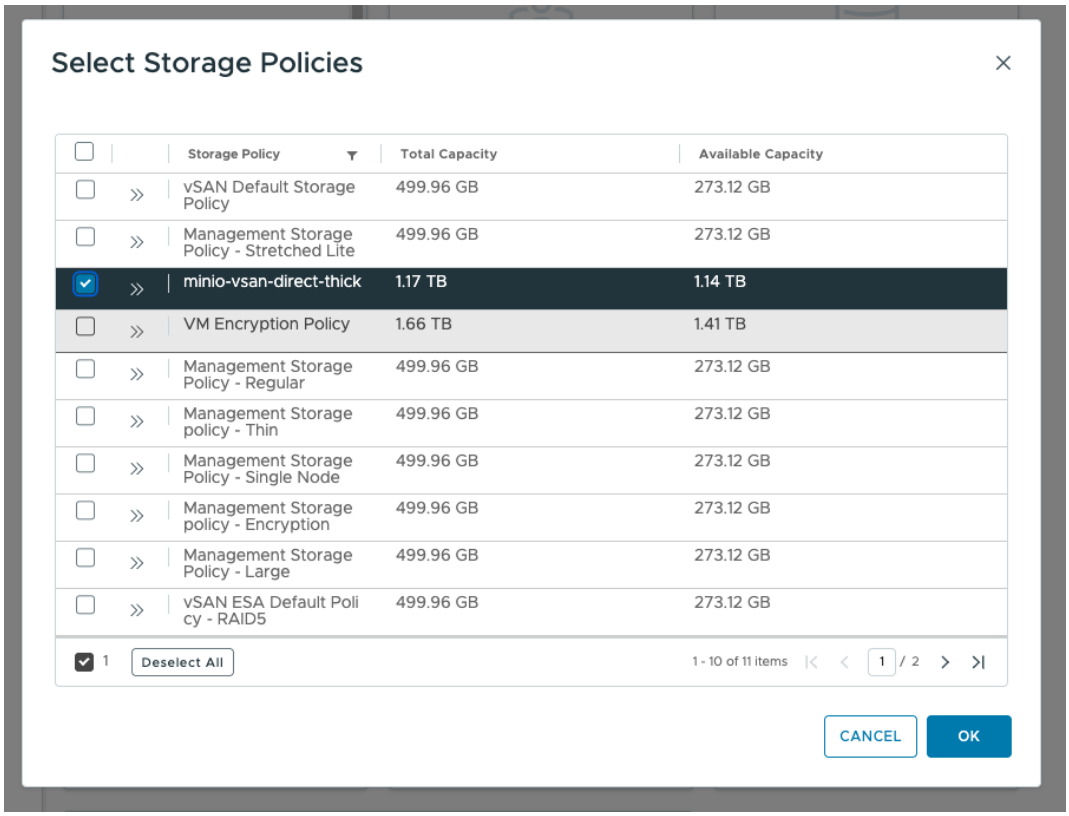
3. The namespace opens on **Summary** with "Your namespace storage has been successfully created.", **Config Status: Running**, **Kubernetes Status: Active**, and empty **Permissions** and **Storage** cards. Click **GOT IT** to close the banner.



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



5. In **Select Storage Policies**, tick **minio-vsan-direct-thick** and click **OK**. It is backed by the vSAN Direct disks (1.17 TB in the lab); MinIO's erasure coding provides the redundancy. The service also created minio-vsan-sna-thick, on the vSAN datastore (the two policies).



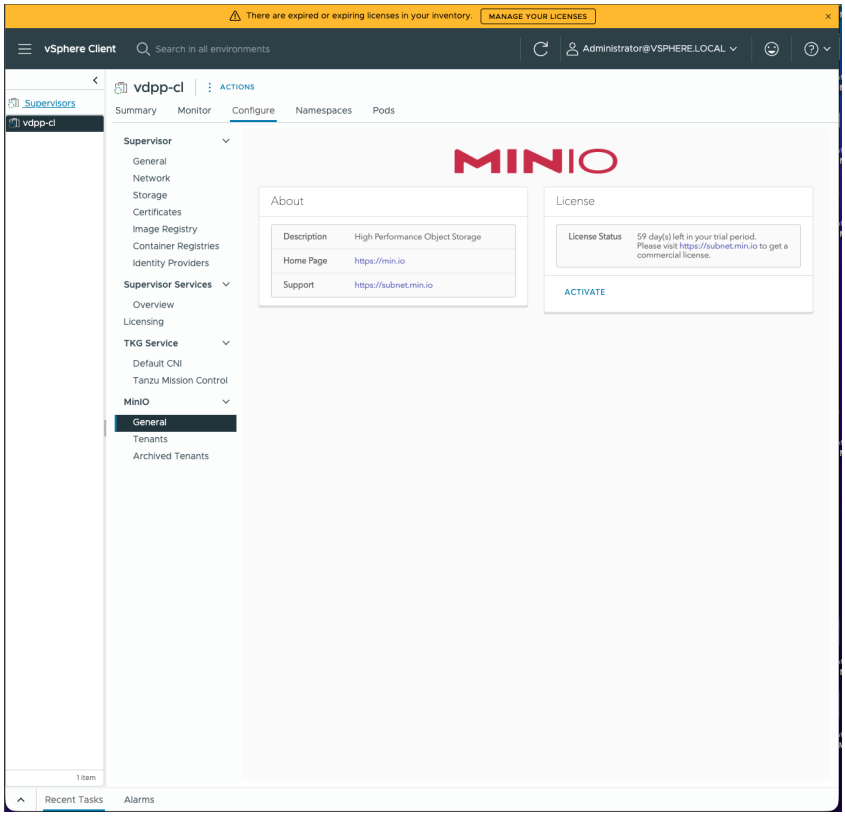
6. Optional: **Permissions** tab → **ADD** the users or groups who will create tenants, with **Can edit**. Members of vsphere.local\administrators, such as administrator@vsphere.local, already have edit rights.

The storage policy makes the Kubernetes StorageClass minio-vsan-direct-thick usable in the namespace.

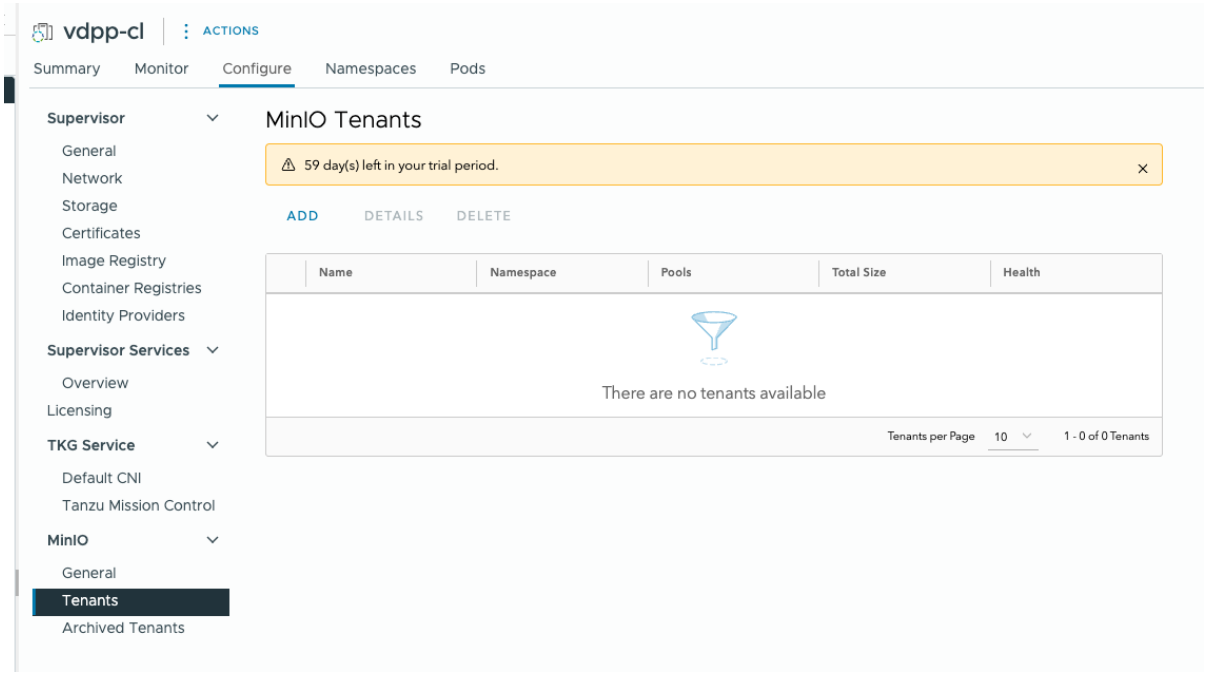
5. Create a tenant

All MinIO pages are under **Workload Management** → **Supervisors** → **<supervisor>** → **Configure** → **MinIO**.

1. **MinIO** → **General** shows **About** and **License**. Like 2.0.10, 2.0.13 runs a **60-day trial** without a license ("59 day(s) left in your trial period" the day after the install), and **ACTIVATE** takes a SUBNET license key (license). The trial is enough to create tenants.



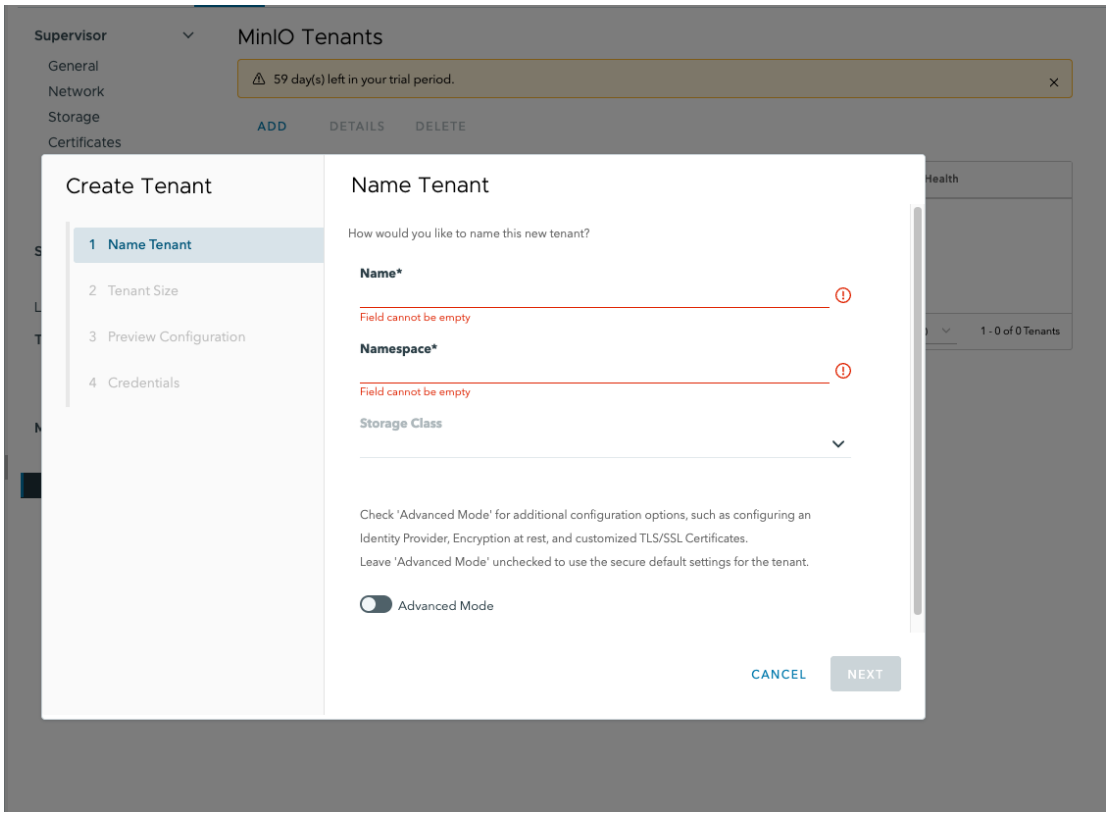
2. **MinIO → Tenants** lists the tenants of every namespace (**Name**, **Namespace**, **Pools**, **Total Size**, **Health**) and starts with "There are no tenants available". Click **ADD**.



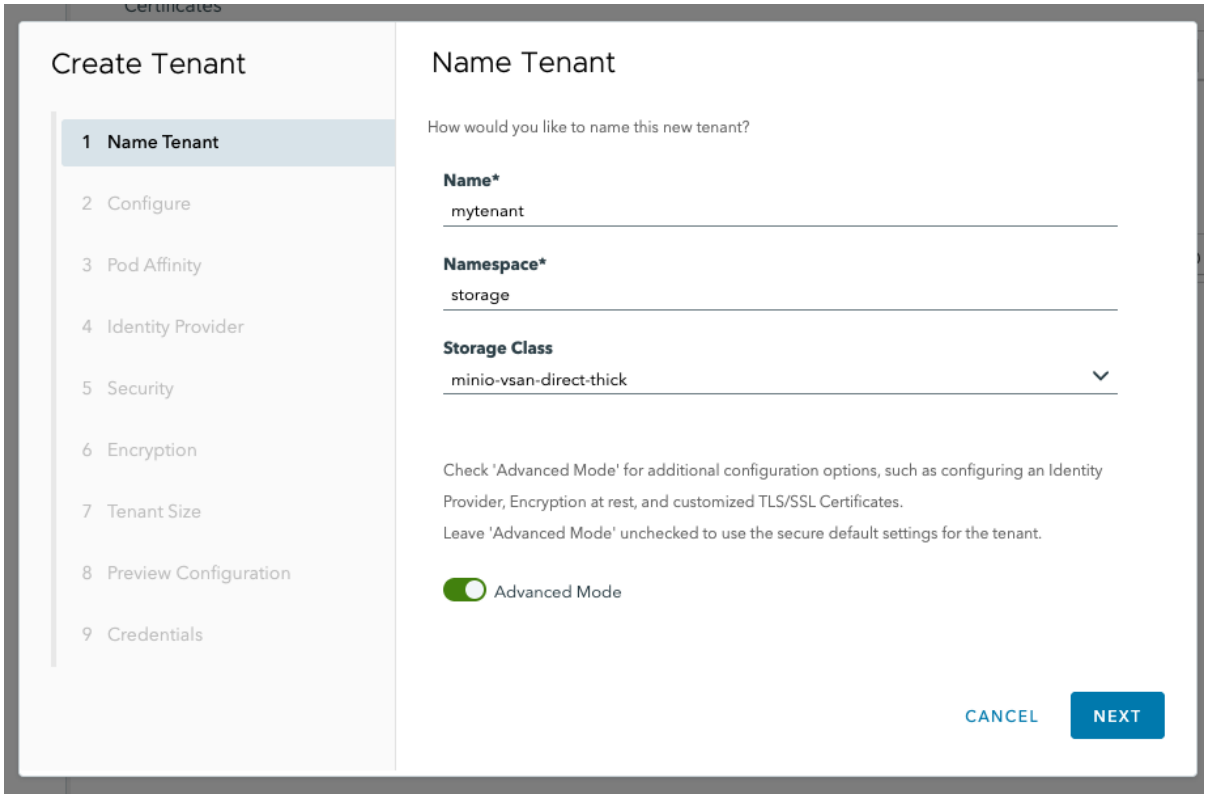
3. **Create Tenant** has four steps: **Name Tenant**, **Tenant Size**, **Preview Configuration**, **Credentials**. On **Name Tenant**:

- **Name**: the tenant name, e.g. mytenant;
- **Namespace**: type the namespace from step 4 (storage);
- **Storage Class**: enabled once the namespace is entered; choose **minio-vsan-direct-thick**.

NEXT stays disabled until Name and Namespace are filled.



Advanced Mode off uses the secure defaults. On, it adds the steps for an identity provider, encryption at rest and custom TLS certificates, and the wizard has nine steps: **Name Tenant**, **Configure**, **Pod Affinity**, **Identity Provider**, **Security**, **Encryption**, **Tenant Size**, **Preview Configuration**, **Credentials**.

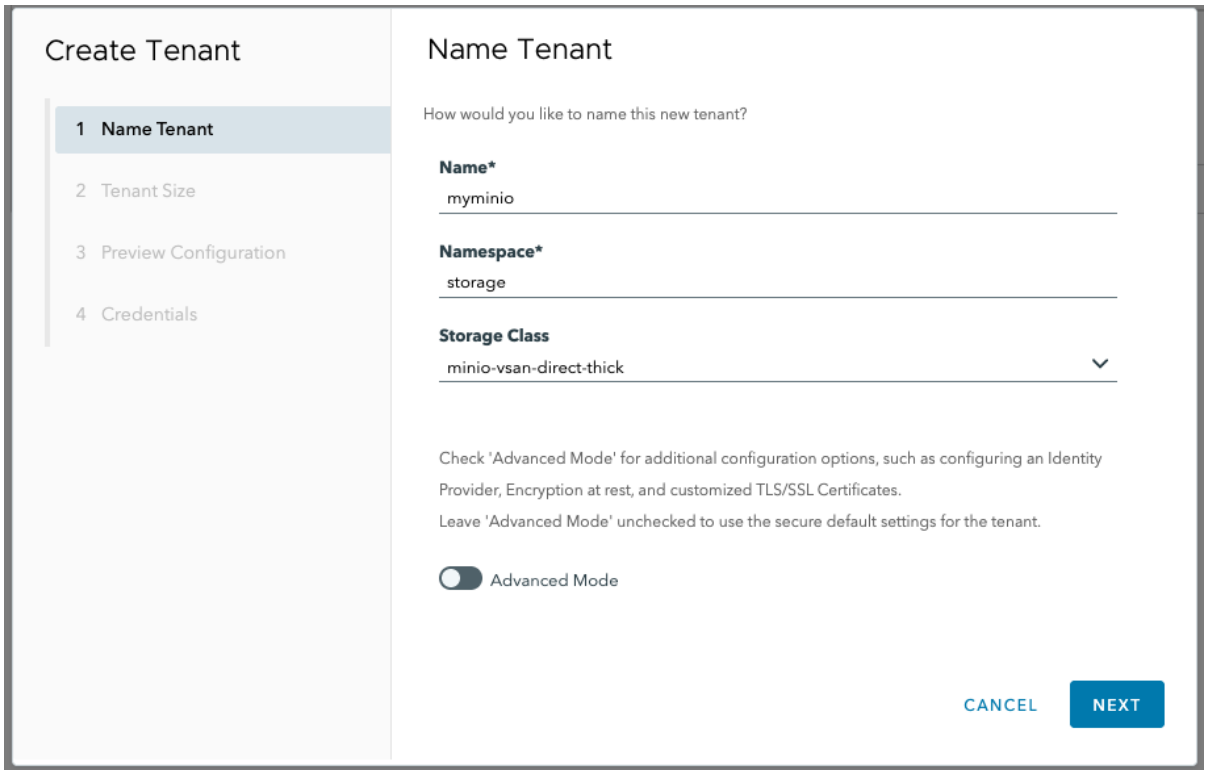


The next steps depend on the mode: basic mode, which takes the defaults, or Advanced Mode, steps 4 to 12.

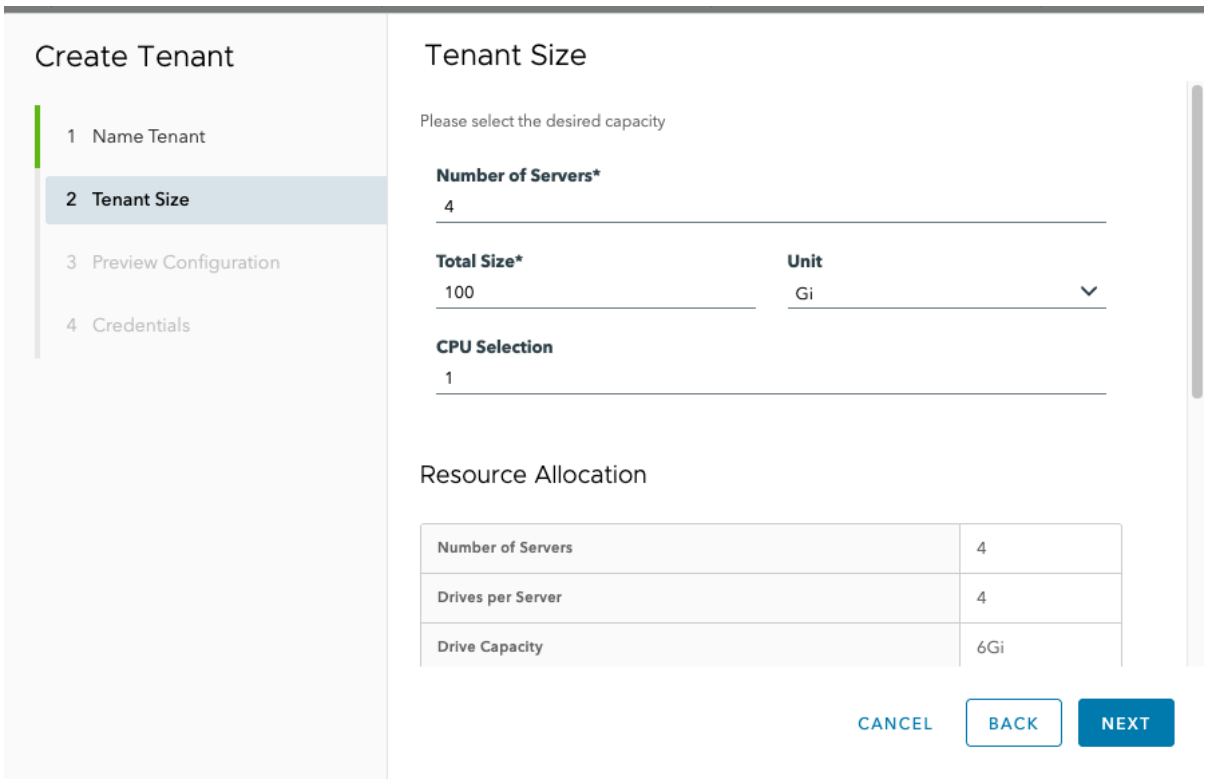
Basic mode: the default image

With **Advanced Mode** off the tenant gets the secure defaults (TLS with AutoCert, built-in users, Pod Anti-affinity) and the default tenant image. With registryName set, that is **<registryName>/minio:RELEASE.2025-09-07T16-13-09Z**, e.g. registry.example.com/minio/minio:RELEASE.2025-09-07T16-13-09Z: it must be in your registry project under that name, or the MinIO pods stay in ErrImagePull. The tenant gets the service's pull secret, which covers it and the operator image.

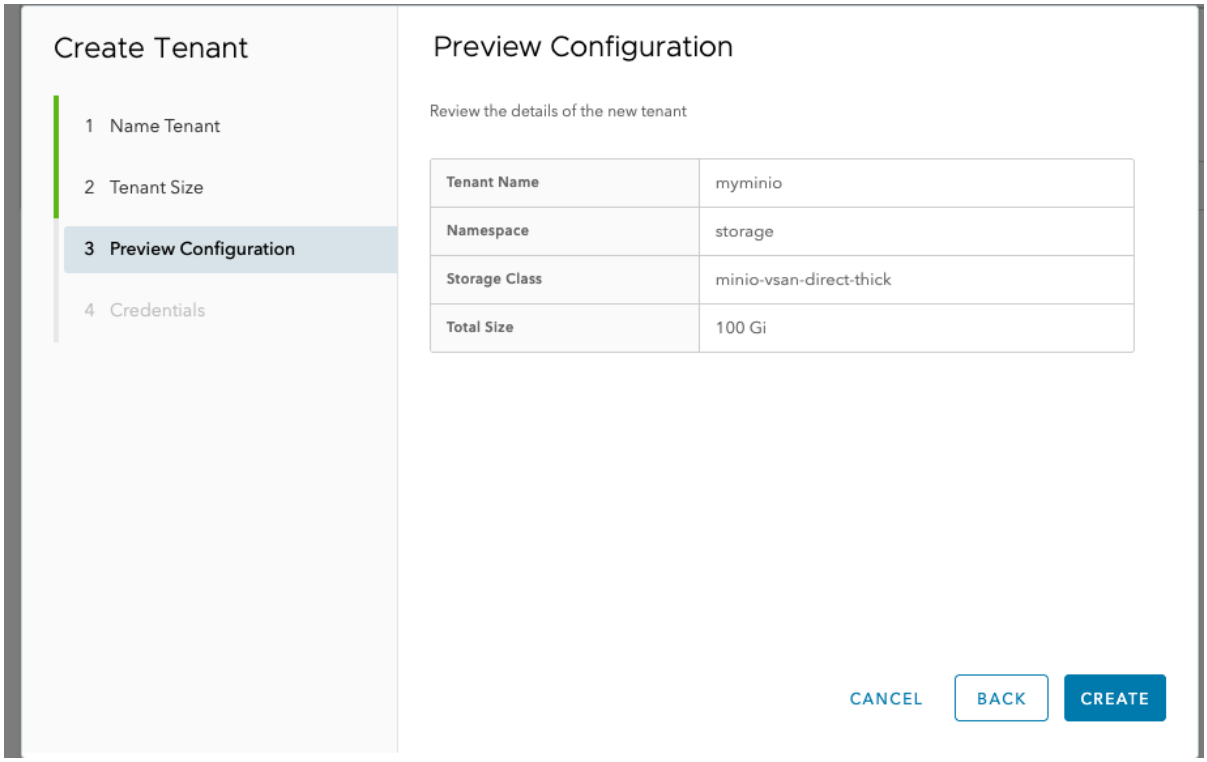
1. **Name Tenant**: Name (e.g. myminio), Namespace storage, Storage Class minio-vsan-direct-thick, **Advanced Mode** off. The wizard has four steps. **NEXT**.



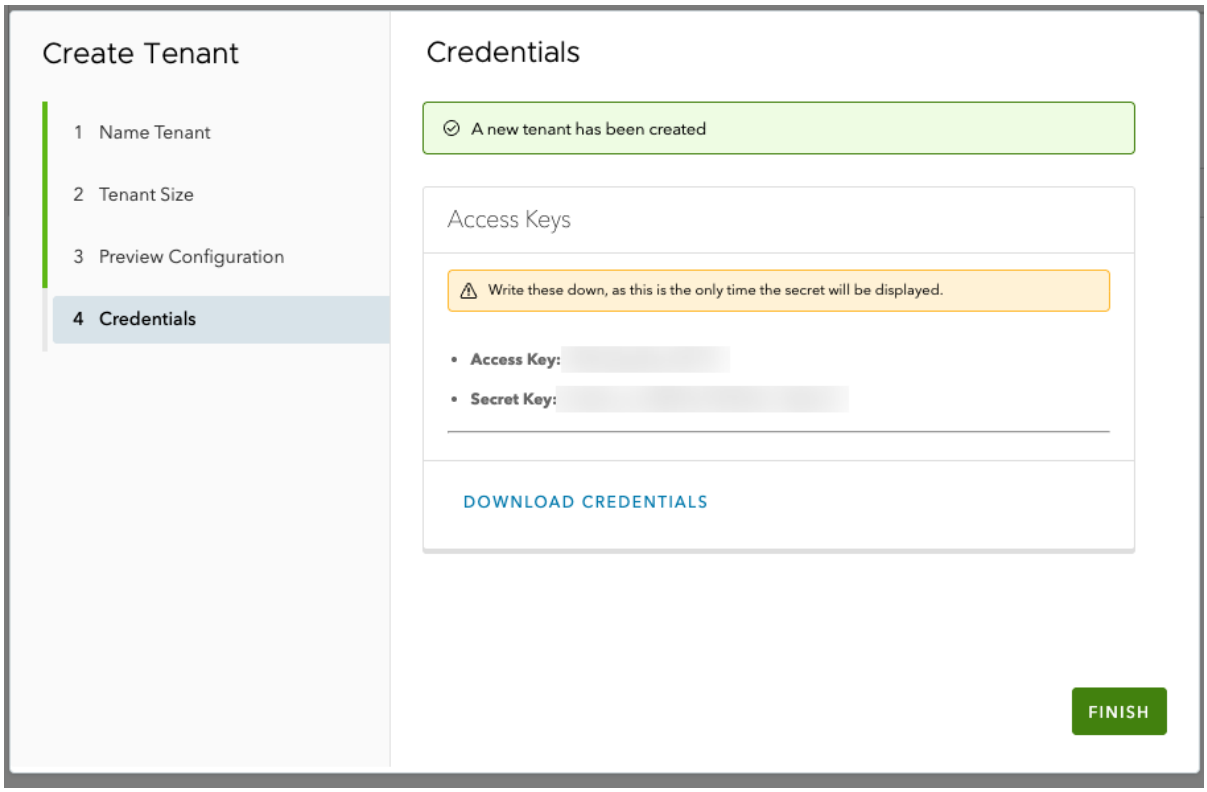
2. **Tenant Size**: **Number of Servers** (at least 4), **Total Size** with its **Unit**, and **CPU Selection**; **Resource Allocation** shows how the plug-in spreads the capacity (here 4 servers × 4 drives). Lower **CPU Selection** from its default, a host's whole allocatable CPU, or the pods stay **Pending** on small hosts (the lab tenant uses 1). **NEXT**.



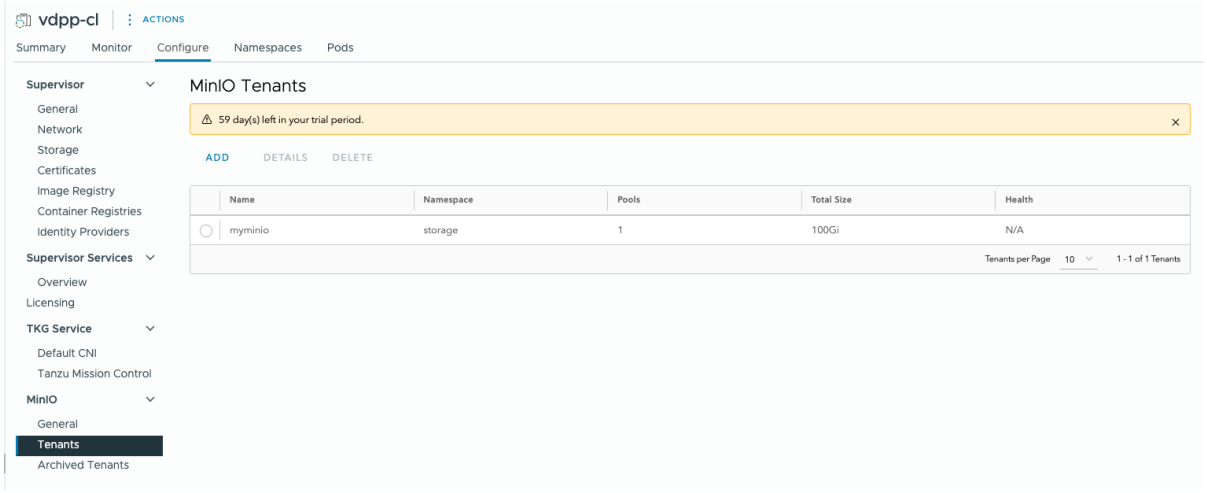
3. **Preview Configuration**: Tenant Name, Namespace, Storage Class and Total Size. There is no MinIO Image line: the tenant takes the default. **CREATE**.



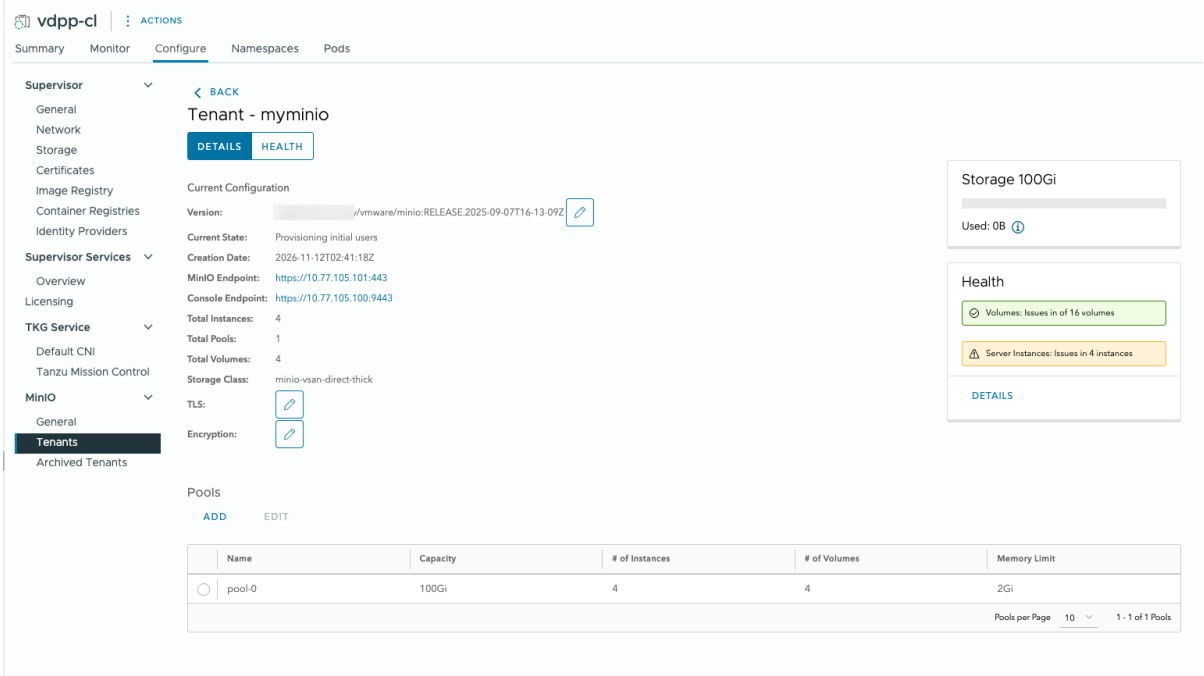
4. **Credentials**: "A new tenant has been created", with the tenant's **Access Key** and **Secret Key**, shown only this once: note them or use **DOWNLOAD CREDENTIALS**, then **FINISH**.



5. **MinIO → Tenants** lists **myminio** in **storage**, 1 pool, 100Gi; **Health** shows **N/A** until its pods are running.



6. **MinIO → Tenants → myminio → DETAILS**: **Version** is the default image, <registryName>/minio:RELEASE.2025-09-07T16-13-09Z. Right after the creation the tenant is in **Provisioning initial users** and the **Health** card shows issues in the server instances; both clear once the pods are running (step 13 shows a running tenant).

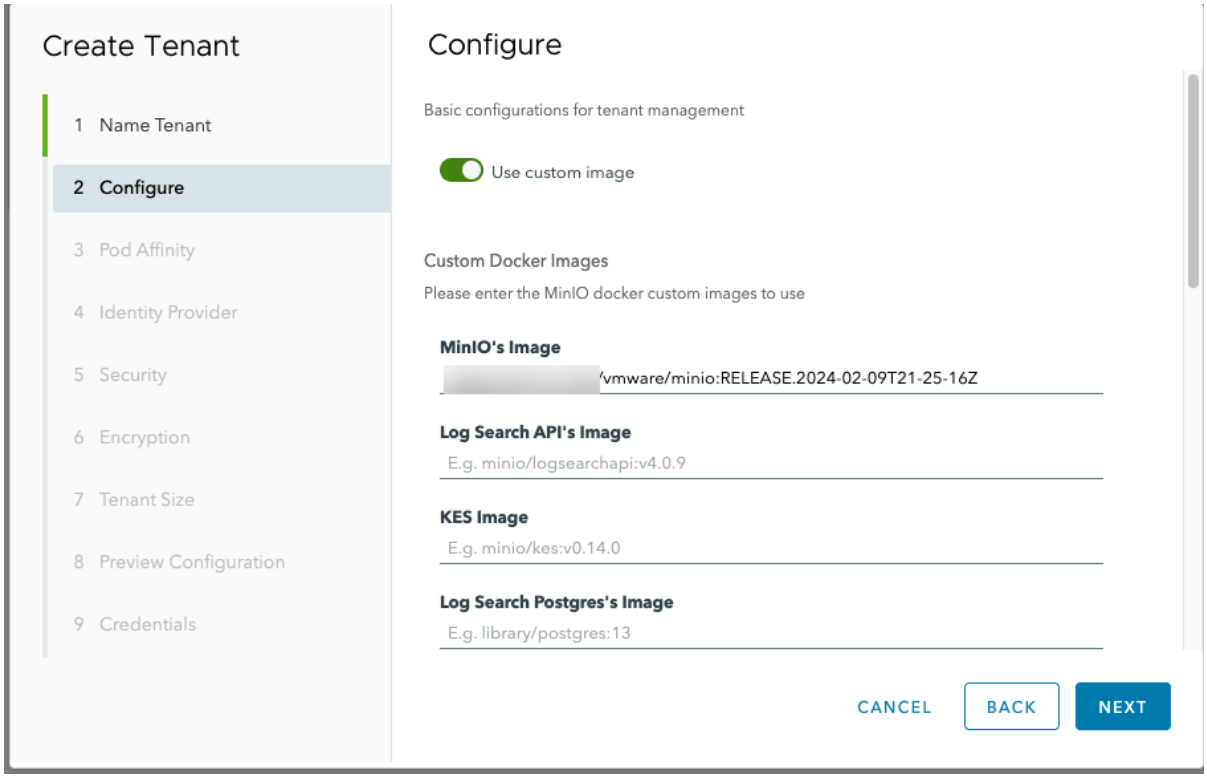


Advanced Mode

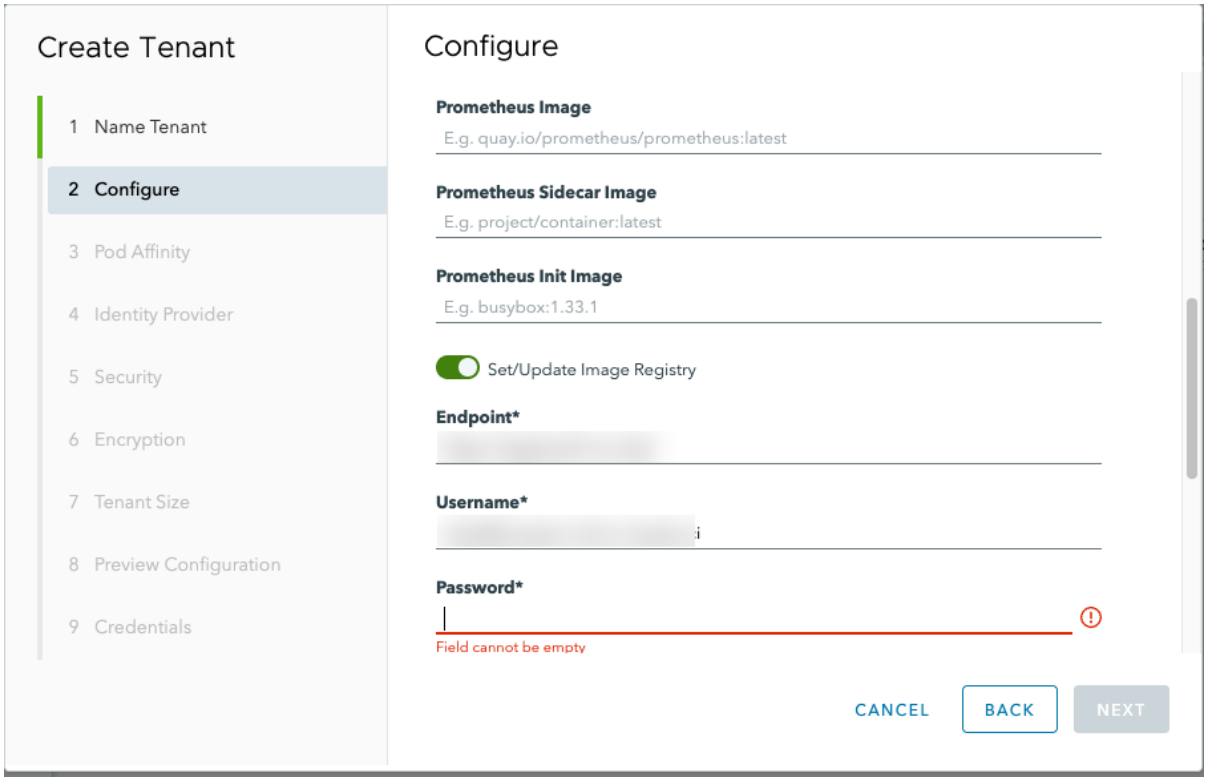
With **Advanced Mode** on, the wizard adds steps 4 to 8; **Tenant Size**, **Preview Configuration** and **Credentials** follow as steps 9 to 11.

4. **Configure**: leave **Set/Update Image Registry** off. With **Use custom image** off too, the tenant runs the default tenant image, <registryName>/minio:RELEASE.2025-09-07T16-13-09Z, which must be in your registry project; if it is not, turn **Use custom image** on and enter a MinIO image you have there. The plug-in gives a tenant created without **Set/Update Image Registry** a copy of the service's pull secret (<prefix>-regcred, with your registry's credentials) in its namespace, and the tenant pods pull with it: the MinIO image and the operator image they also run (the sidecar container and the validate-arguments init container). A registry set on the tenant replaces that secret, and the operator image then fails to pull (Known issues).

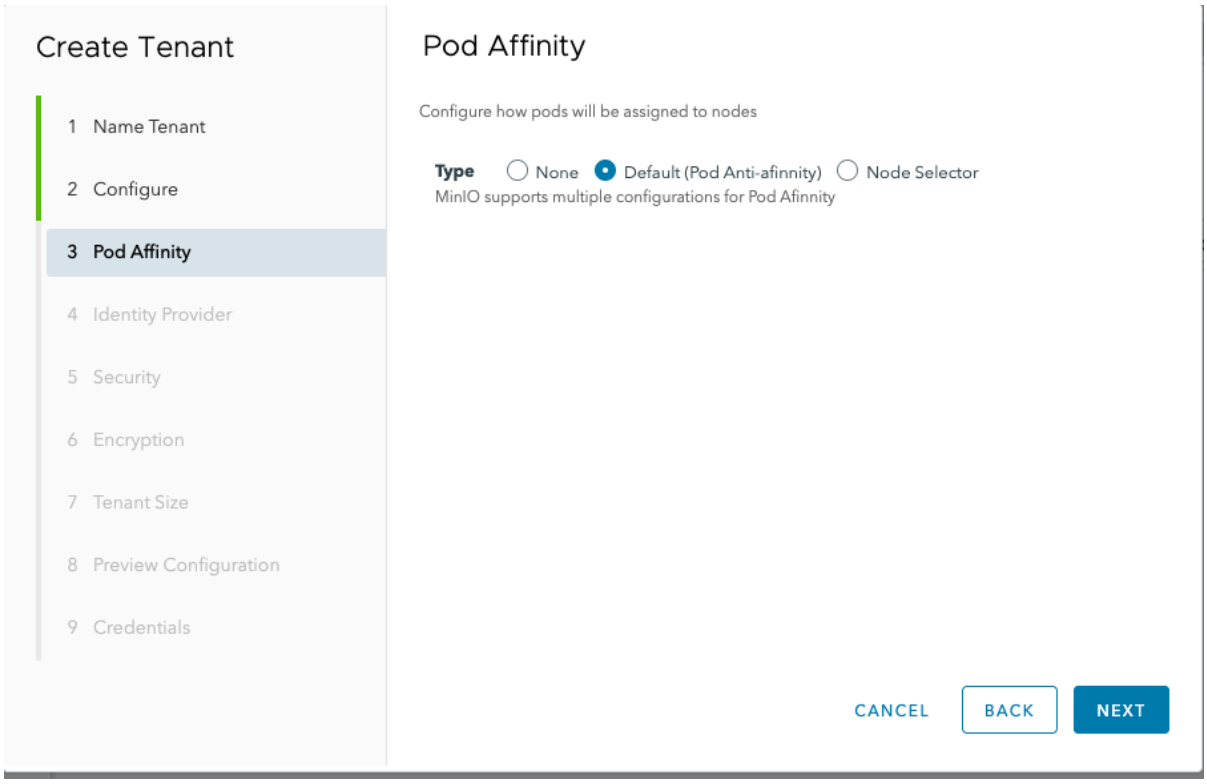
   Use custom image sets another MinIO image under **Custom Docker Images → MinIO's Image**. Take it from the service's registry project, e.g. registry.example.com/minio/minio:RELEASE.2024-02-09T21-25-16Z; the other image fields (Log Search, KES, Postgres, Prometheus) can stay empty.



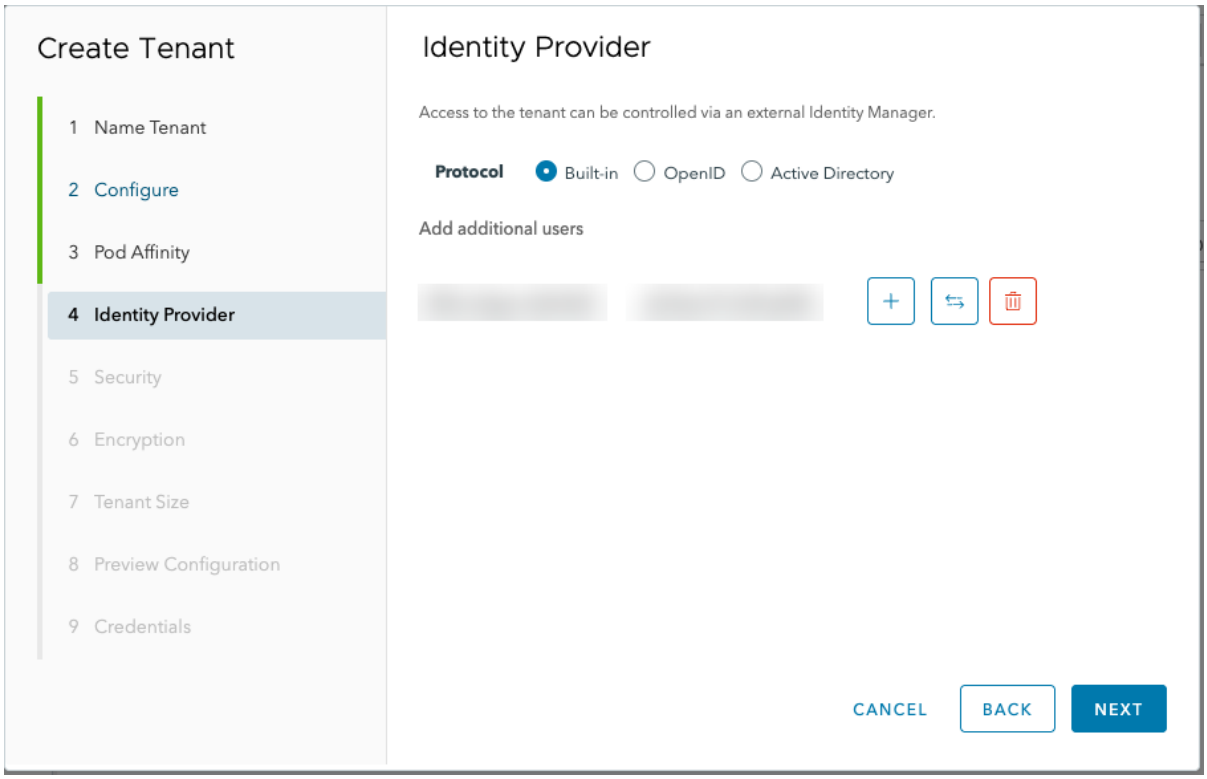
Set/Update Image Registry (**Endpoint**, **Username**, **Password**) gives the tenant the credentials of another registry. Do not use it in 2.0.13: the tenant then loses the service's pull secret, and its pods cannot pull the operator image (Known issues).



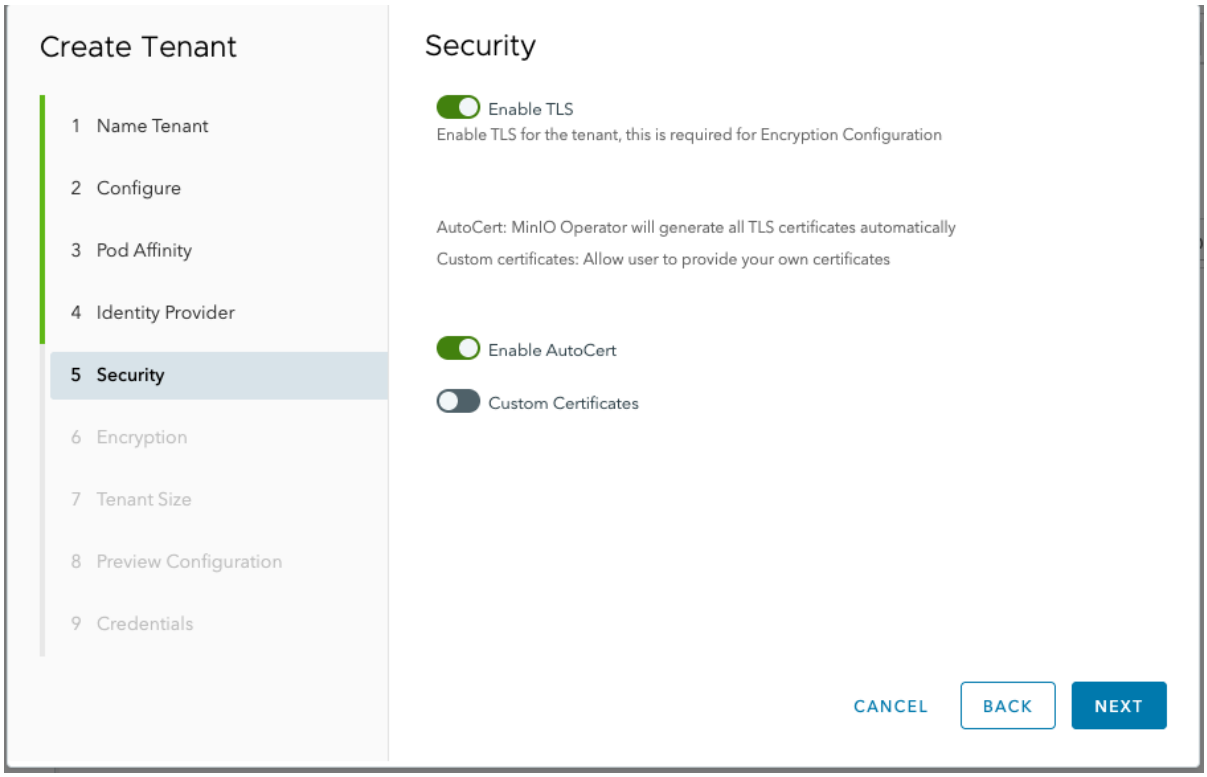
5. **Pod Affinity**: keep **Default (Pod Anti-affinity)**, which places each MinIO server on a different host. **None** and **Node Selector** are the alternatives.



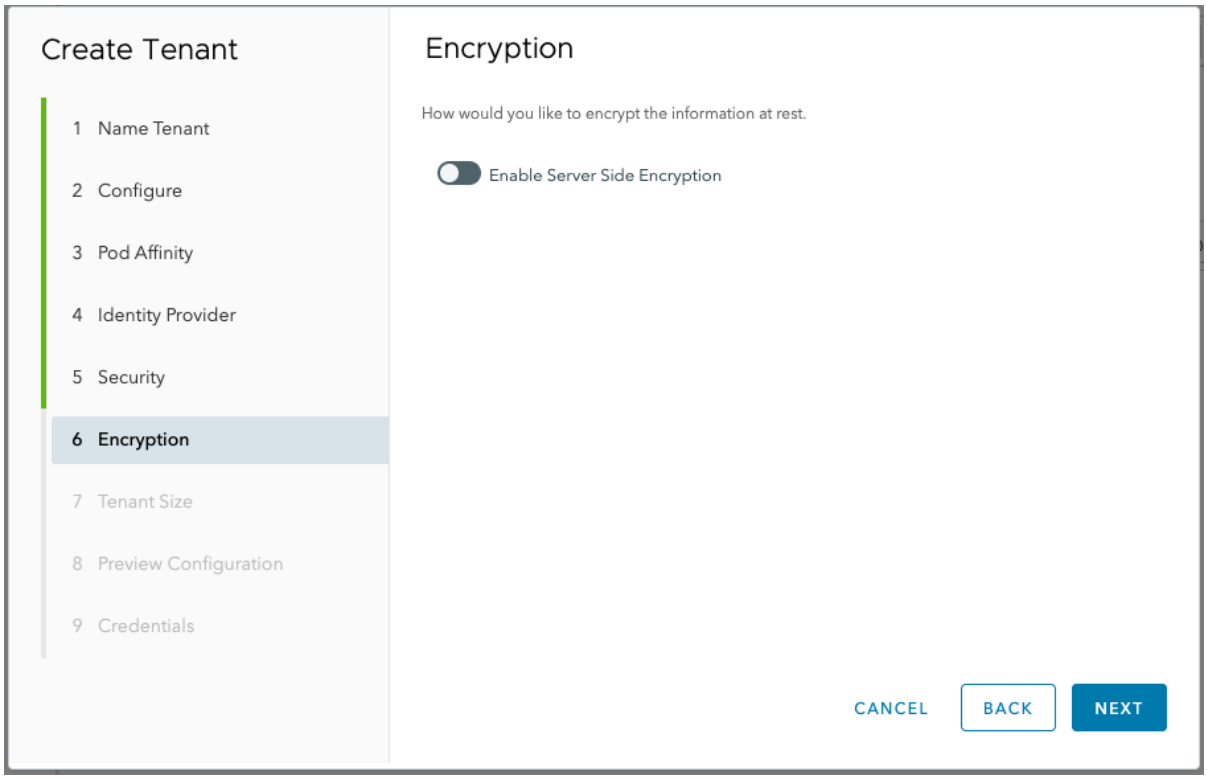
6. **Identity Provider**: **Built-in** keeps the users in MinIO. The wizard fills in a first user (access key and secret key); **+** adds more, the arrows button generates new keys and the bin removes one. **OpenID** and **Active Directory** connect an external identity provider instead.



7. **Security**: **Enable TLS** and **Enable AutoCert** are on, so the MinIO Operator generates the tenant's certificates. **Custom Certificates** lets you provide your own. TLS is required for encryption.

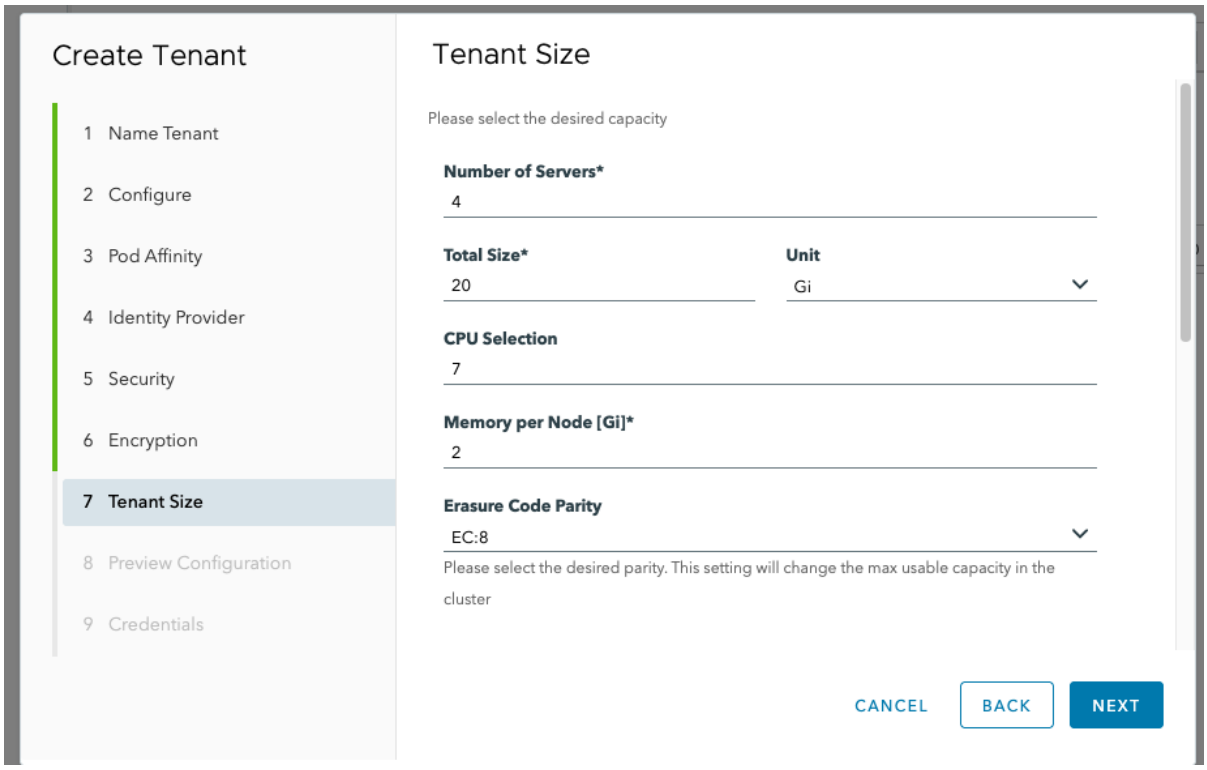


8. **Encryption**: **Enable Server Side Encryption** is off. Turn it on to encrypt data at rest with a KMS through KES.

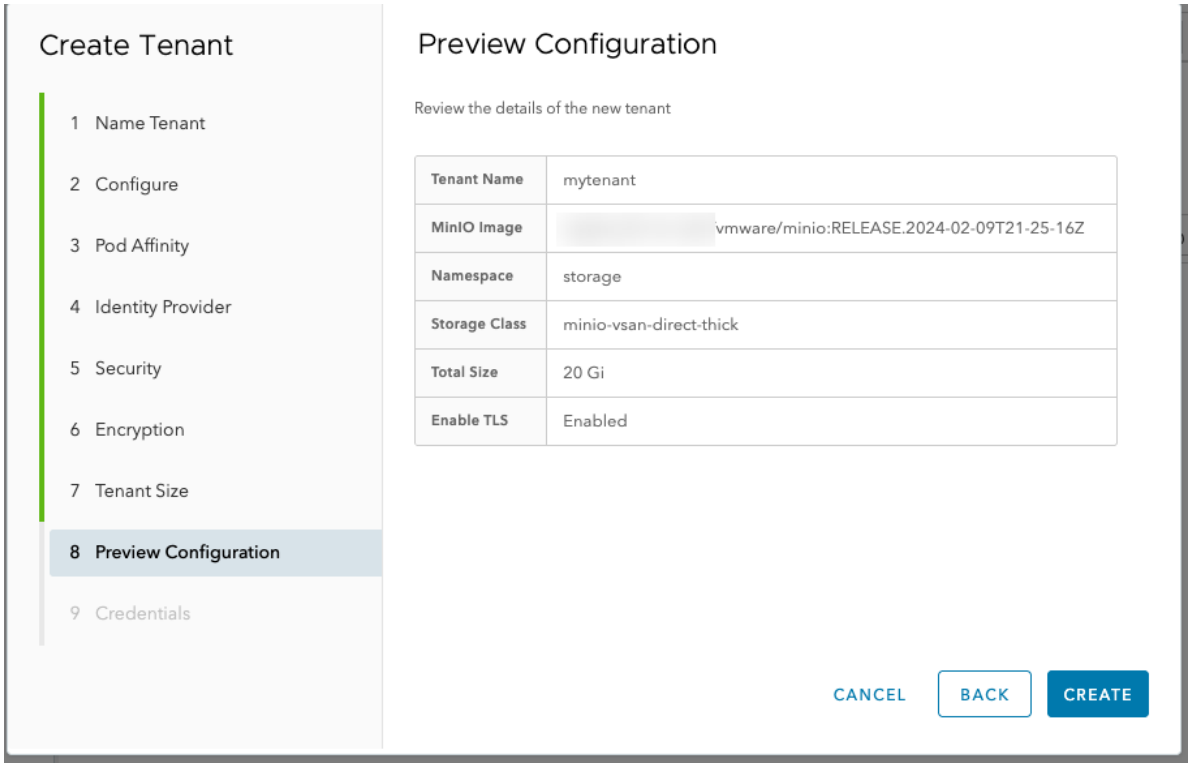


9. **Tenant Size**: **Number of Servers** (at least 4), **Total Size** with its **Unit**, **CPU Selection**, **Memory per Node [Gi]** and **Erasure Code Parity**, which sets the usable capacity. The lab tenant uses 4 servers, 20 Gi, 2 Gi of memory per node and EC:8.

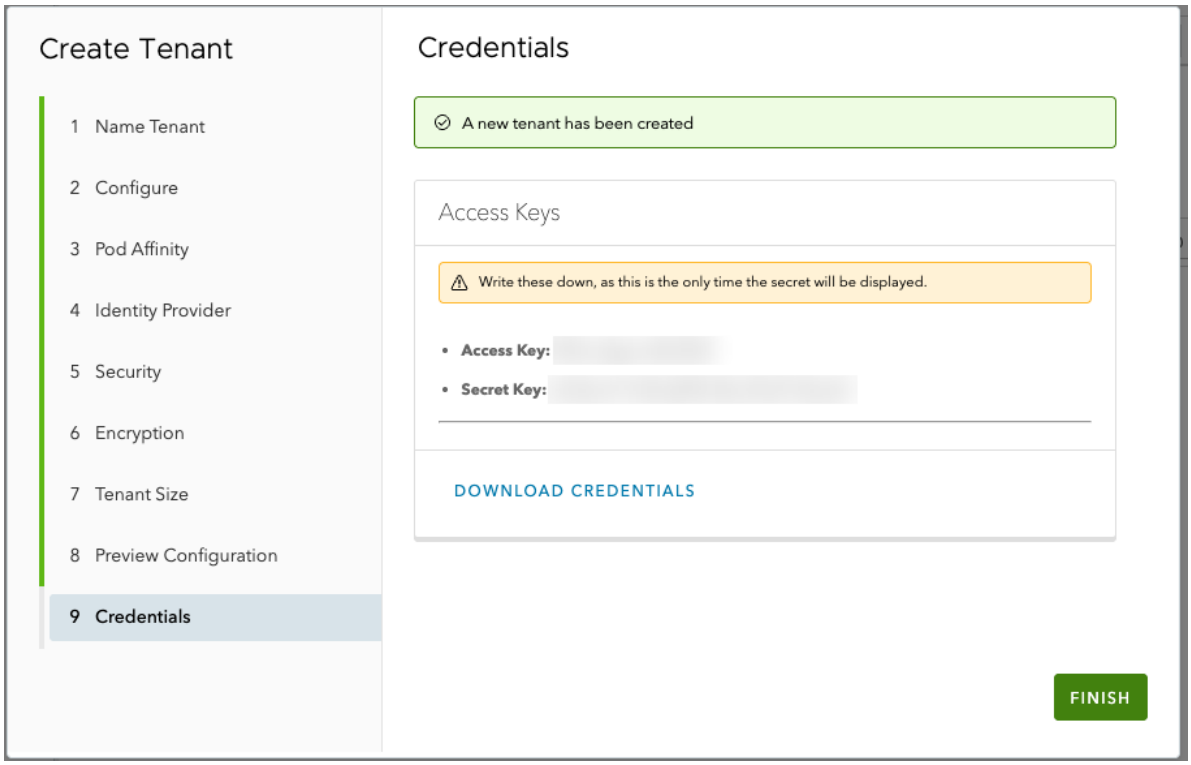
   CPU Selection defaults to a host's whole allocatable CPU (7 on the lab's 8-core hosts). vSphere Pods reserve what they request, so on small hosts lower it, or the MinIO pods stay **Pending** (Edit Pool changes it afterwards).



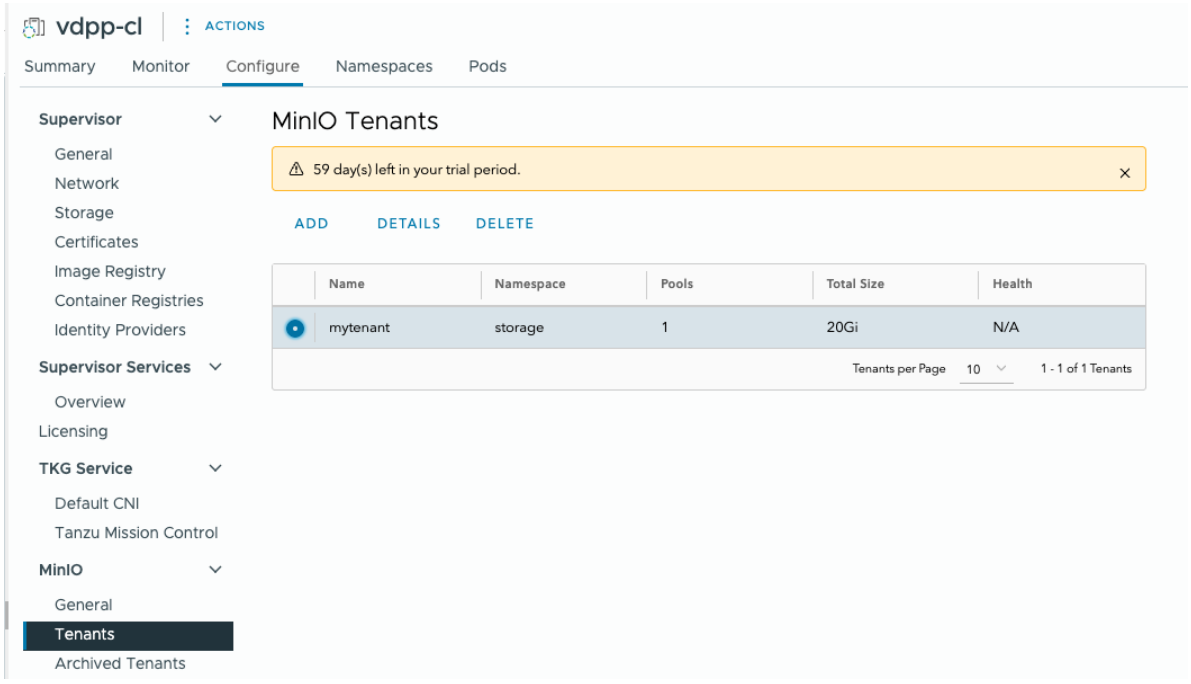
10. **Preview Configuration**: Tenant Name, MinIO Image, Namespace, Storage Class, Total Size and Enable TLS. Click **CREATE**.



11. **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**.

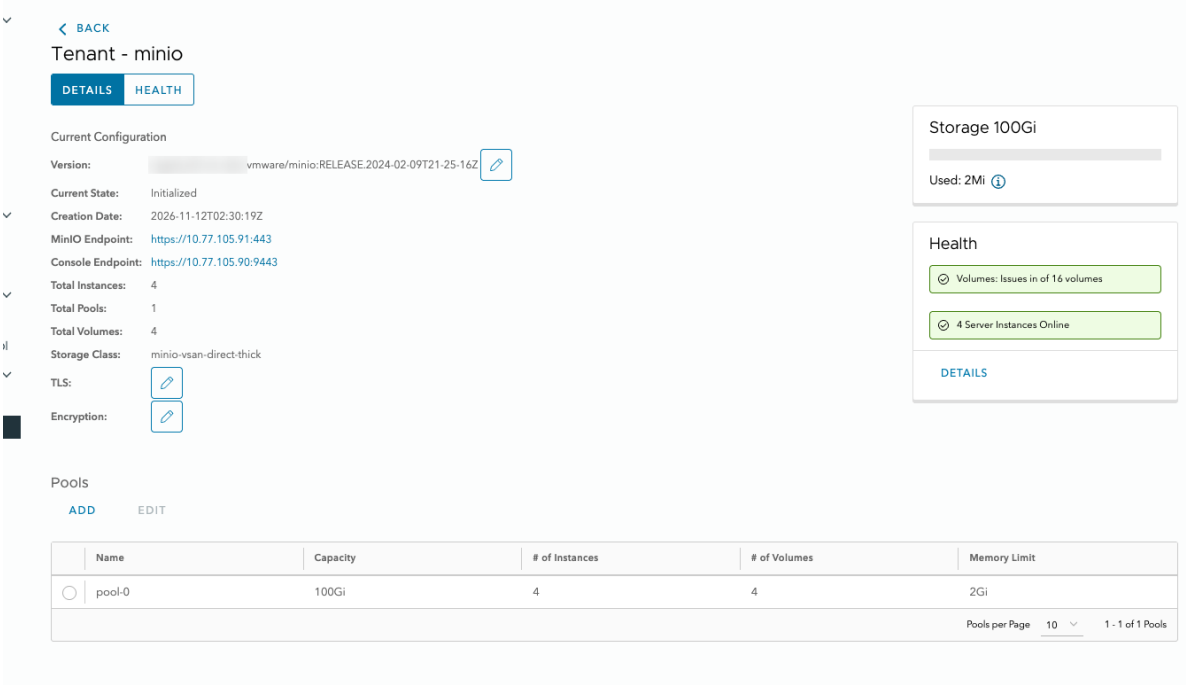


12. **MinIO → Tenants** lists the new tenant: **mytenant** in **storage**, 1 pool, 20Gi. **Health** shows **N/A** until its pods are running.

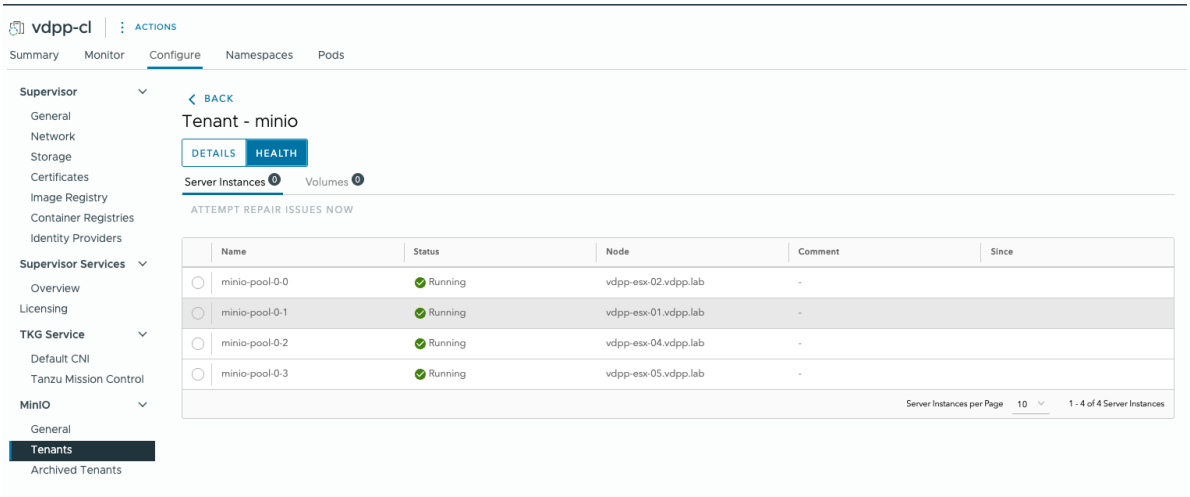


13. **MinIO → Tenants → <tenant>** opens the tenant page with **DETAILS** and **HEALTH**. **DETAILS** shows the current configuration: the MinIO image (**Version**), **Current State** (**Initialized** once it runs), the creation date, the **MinIO Endpoint** and **Console Endpoint**, the instances, pools, volumes and storage class, the **TLS** and **Encryption** edit buttons, and the **Pools** list, next to the **Storage** and **Health** cards. The pencil next to **Version** changes the MinIO image.

    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>.



    HEALTH lists the **Server Instances** and **Volumes** with their status, node and comments, and **ATTEMPT REPAIR ISSUES NOW**. A healthy tenant shows one Running instance per ESXi host (Default Pod Anti-affinity):



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. Serve the MinIO image from the service's registry and leave **Set/Update Image Registry** off (private registry).
- Without registryUsername in the YAML Service Config, the operator pod is expected to stay in ErrImagePull (Pull credentials).