# Deployment of Open-CAS

Install Open-CAS on the host who will join in the Kubernetes cluster.  
You can get more details about Open-CAS from [here](https://open-cas.github.io/).

## Prerequisites

OS:Linux

## Steps

* Download the Open CAS Linux project and change current directory to project folder

$wget <https://github.com/Open-CAS/open-cas-linux/releases/download/v21.6.1/open-cas-linux-21.06.1.0547.release.tar.gz>

$tar -xf open-cas-linux-21.06.1.0547.release.tar.gz  
$cd open-cas-linux-21.06.1.0547.release/

* Configure Open CAS Linux, compile it and install it

$./configure

$make

$make install

* Verify the kernel modules were inserted by checking their versions

$casadm -V

* Check the storage mounted on this host

$ll /dev/disk/by-id

For example:

$ll /dev/disk/by-id  
total 0  
drwxr-xr-x 2 root root 120 Oct 28 2021 ./  
drwxr-xr-x 7 root root 140 Oct 28 2021 ../  
lrwxrwxrwx 1 root root 9 Oct 28 13:59 virtio-rj9hex2e9zt131tm6i9i -> ../../vda  
lrwxrwxrwx 1 root root 10 Oct 28 13:59 virtio-rj9hex2e9zt131tm6i9i-part1 -> ../../vda1  
lrwxrwxrwx 1 root root 9 Oct 28 13:59 virtio-rj9iyr83m78doasn75ly -> ../../vdb  
lrwxrwxrwx 1 root root 9 Oct 28 13:59 virtio-rj9iyr83m78doasn75lz -> ../../vdc

* Choose block device as a caching device, for example:

$casadm -S -d /dev/disk/by-id/virtio-rj9iyr83m78doasn75ly

* The output should return the cache instance number, use it to add a backend device. For example, to use block device /vdc as a backend device to cache instance 1:

$casadm -A -d /dev/disk/by-id/virtio-rj9iyr83m78doasn75lz -i 1

* Verify CAS instance is operational with:

$casadm -L

fig:

## Automatic deployment

If you want to install Open-CAS on the host and create a new node to join the kubernetes cluster, a shell script is provided here, which integrates the installation methods of Open-CAS, kubectl, kubelet and kubedm.

#! /bin/bash  
if [[ "$#" -lt 1 ]];then  
 echo "must with at least one parameter:  
 --help usage for help  
 -p the dirrectory where admin.conf locates  
 default /root/itcast  
 -k install the kubernetes  
 -c install the openCAS"  
 exit 1  
fi  
  
if [[ $1 == "--help" ]];then  
 echo "it is the utils for install opencase/kubernetes"  
 echo "install the openCAS with ./utils -c"  
 echo "install the kubernetes(#the admin.config is in the /root/itcast by default#)"  
 echo "- with ./utils -k"  
 echo "install kubernetes having your dir with ./utils -k -p your-dir"  
 echo "you can install together with ./utils -k -p your-dir"  
 exit 0  
fi  
  
iskubernetes="0"  
isCAS="0"  
dir="/root/itcast"  
for (( index=0; index <= $#; index++ ))  
do  
 if [[ ${!index} == "-c" ]];then  
 isCAS="1"  
 elif [[ ${!index} == "-k" ]];then  
 iskubernetes="1"  
 elif [[ ${!index} == "-p" ]];then  
 i=$((index+1))  
 dir=${!i}  
 fi  
done  
echo "+++++(0 don't install 1 install)+++++++"  
echo "dir: $dir"  
echo "isCAS: $isCAS"  
echo "iskubernetes: $iskubernetes"  
echo "+++++++++++++++++++++++++++++"  
#exit 0  
if [ $iskubernetes == "1" ];then  
  
#You should set the download source of kubernetes here  
  
sudo ufw disable  
sudo swapoff -a  
setenforce 0  
sudo getenforce  
echo "net.bridge.bridge-nf-call-ip6tables = 1" > /etc/sysctl.d/kubernetes.conf  
echo "net.bridge.bridge-nf-call-iptables = 1" >> /etc/sysctl.d/kubernetes.conf  
echo "vm.swappiness = 0" >> /etc/sysctl.d/kubernetes.conf  
sudo modprobe br\_netfilter  
sudo sysctl -p /etc/sysctl.d/kubernetes.conf  
apt-get install -y kubelet=1.21.1-00 kubernetes-cni=0.8.7-00 kubeadm=1.21.1-00 kubectl=1.21.1-00  
systemctl enable kubelet && systemctl start kubelet  
kubectl version  
mkdir -p $HOME/.kube  
cp -i $dir/admin.conf /root/.kube/config  
chown $(id -u):$(id -g) /root/.kube/config  
echo "##################################"  
echo "\n"  
echo "the installation of kubernetes is over!!"  
echo "\n"  
echo "##################################"  
fi  
# -c for install Open-CAS  
#################OPEN-CAS###############  
  
if [ "$isCAS" == "1" ];then  
echo "###########################################"  
echo "\n"  
echo "==the next is the openCAS installation==\n"  
echo "\n"  
echo "###########################################"  
disk1=`ls -l /dev/disk/by-id|sed -n 4p |xargs|awk '{print $9}' `  
disk2=`ls -l /dev/disk/by-id|sed -n 5p |xargs|awk '{print $9}'`  
wget https://github.com/Open-CAS/open-cas-linux/releases/download/v21.6.1/open-cas-linux-21.06.1.0547.release.tar.gz  
tar -xf open-cas-linux-21.06.1.0547.release.tar.gz  
cd open-cas-linux-21.06.1.0547.release/  
./configure  
make  
make install  
echo "disk1: $disk1"  
echo "disk2: $disk2"  
casadm -S -d /dev/disk/by-id/$disk1  
casadm -A -d /dev/disk/by-id/$disk2 -i 1  
echo "install openCAS finished!"  
casadm -L  
fi  
exit 0

After that, you only need to join the cluster with "kubedm join", for example:

$kubeadm join 172.21.88.54:6443 --token abcdef.0123456789abcdef \ --discovery-token-ca-cert-hash sha256:52e99a0dfc203373bbb86fba869f8e6dc0b87773d7bfeb429a41d1f91e3de9f1

## Another way to deploy Open-CAS

You can also configure the cache instance by file instead of “casadm -S”, open the file /etc/opencas/opencas.conf, then follow these steps.

1. Edit and configure caches and cores as desired via the opencas.conf file, for example:

$ vim /etc/opencas/opencas.conf

version=19.3.0

# Version tag has to be first line in this file

#

# Open CAS configuration file - for reference on syntax

# of this file please refer to appropriate documentation

# NOTES:

# 1) It is required to specify cache/core device using links in

# /dev/disk/by-id/, preferably those using device WWN if available:

# /dev/disk/by-id/wwn-0x123456789abcdef0

# Referencing devices via /dev/sd\* is prohibited because

# may result in cache misconfiguration after system reboot

# due to change(s) in drive order. It is allowed to use /dev/cas\*-\*

# as a device path.

## Caches configuration section

[caches]

## Cache ID Cache device Cache mode Extra fields (optional)

## Uncomment and edit the below line for cache configuration

1 /dev/disk/by-id/virtio-rj9i0d0dpg03g47l1myr WB

## Core devices configuration

[cores]

## Cache ID Core ID Core device Extra fields (optional)

## Uncomment and edit the below line for core configuration

1 1 /dev/disk/by-id/virtio-rj9i0d0dpg03g47l1mys

## To specify use of the IO Classification file, place content of the following line in the

## Caches configuration section under Extra fields (optional)

## ioclass\_file=/etc/opencas/ioclass-config.csv

## If given cache/core pair is especially slow to start up, often doesn't come back

## up after reboot or you simply don't care if it does and don't want it to have

## an effect on your boot you can mark cores as lazy to prevent Open CAS from

## dropping boot to emergency shell because of this core failure. To do this

## put following line under in Extra fields (optional) section of core config:

## lazy\_startup=true

## NOTE: This will cause open-cas.service to not wait for marked core while

## starting up - this option should be used with care to prevent races with

## other services for devices (e.g. mounts based on FS labels)

2.Execute the following:

$ casctl init

## IO Classification

Open CAS Linux provides the ability to control data caching with classification granularity. Open CAS Linux can analyze every IO on the fly to determine whether the requested block is filesystem metadata or data and, if it is data, the size of the destination file. Using this information the administrator can determine the best IO class configuration settings for the typical workload and can set which IO classes to cache and not to cache, and set a priority level for each IO class. As a result, when it becomes necessary to evict a cache line, the software will evict cache lines of the lowest priority first (a major improvement compared to traditional LRU eviction). You can get more details from [here](https://open-cas.github.io/guide_io_classification.html).

To enable IO classification and selective allocation, first look at the provided example IO class configuration file and edit it to suit your needs:

Open the file /etc/opencas/io-class-config.csv

When you have your own needs, you can create a new file, for example:

$ vim /etc/opencas/io-class-minio.csv

IO class id,IO class name,Eviction priority,Allocation

0,unclassified,22,1

1,metadata&done,0,1

2,extension:meta&done,2,1

11,file\_size:le:4096&done,9,1

12,file\_size:le:16384&done,10,1

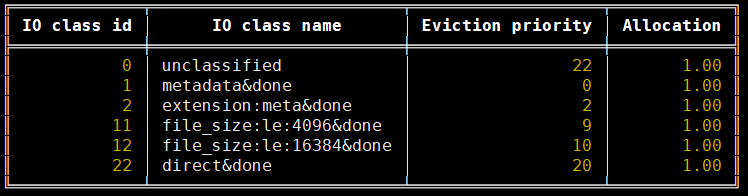
22,direct&done,20,1

After you have completed your changes to the example IO class configuration and saved the file, you must load the configuration for your cache device with ID number represented by <ID> from file <FILE>, for example:

$ casadm --io-class --load-config --cache-id 1 -f io-class-minio.csv

Verify that the configuration file was successfully loaded, for example:

$ casadm --io-class --list --cache-id 1



Comment: The IO class named “metadata” has the highest priority, which means the metadata of the file system will be the last to be evicted in cache. When a high speed storage media like Optane configured by Open-CAS as the cache device, the process like traversal or access of the file system will be accelerated.

Comment: MinIO’s metadata are files with the extension of meta in the file system. So we can improve the overall performance of MinIO by setting higher eviction priority for the file with the extension meta. When we use MinIO to traverse and read files, MinIO’s metadata will be frequently used, so we prefer the metadata of MinIO are always in cache.

You can also put the configuration in the file /etc/opencas/opencas.conf, for example:

version=19.3.0

# Version tag has to be first line in this file

#

# Open CAS configuration file - for reference on syntax

# of this file please refer to appropriate documentation

# NOTES:

# 1) It is required to specify cache/core device using links in

# /dev/disk/by-id/, preferably those using device WWN if available:

# /dev/disk/by-id/wwn-0x123456789abcdef0

# Referencing devices via /dev/sd\* is prohibited because

# may result in cache misconfiguration after system reboot

# due to change(s) in drive order. It is allowed to use /dev/cas\*-\*

# as a device path.

## Caches configuration section

[caches]

## Cache ID Cache device Cache mode Extra fields (optional)

## Uncomment and edit the below line for cache configuration

1 /dev/disk/by-id/virtio-rj9i0d0dpg03g47l1myr WB ioclass\_file=/etc/opencas/ioclass-minio.csv

## Core devices configuration

[cores]

## Cache ID Core ID Core device Extra fields (optional)

## Uncomment and edit the below line for core configuration

1 1 /dev/disk/by-id/virtio-rj9i0d0dpg03g47l1mys

## To specify use of the IO Classification file, place content of the following line in the

## Caches configuration section under Extra fields (optional)

## ioclass\_file=/etc/opencas/ioclass-config.csv

## If given cache/core pair is especially slow to start up, often doesn't come back

## up after reboot or you simply don't care if it does and don't want it to have

## an effect on your boot you can mark cores as lazy to prevent Open CAS from

## dropping boot to emergency shell because of this core failure. To do this

## put following line under in Extra fields (optional) section of core config:

## lazy\_startup=true

## NOTE: This will cause open-cas.service to not wait for marked core while

## starting up - this option should be used with care to prevent races with

## other services for devices (e.g. mounts based on FS labels)

~

# Deployment of Direct-CSI

You can get more details about Direct-CSI from [here](https://github.com/minio/direct-csi).

## Prerequisites

Having a Kubernetes cluster, for example, we test everything in a cluster with five nodes:

$kubectl get nodes

NAME STATUS ROLES AGE VERSION

node1 Ready <none> 4d17h v1.21.1

node2 Ready <none> 4d18h v1.21.1

node3 Ready <none> 4d21h v1.21.1

node4 Ready <none> 4d23h v1.21.1

master Ready control-plane,master 5d v1.21.1

$kubectl version

Client Version: version.Info{Major:"1", Minor:"21", GitVersion:"v1.21.1", GitCommit:"5e58841cce77d4bc13713ad2b91fa0d961e69192", GitTreeState:"clean", BuildDate:"2021-05-12T14:18:45Z", GoVersion:"go1.16.4", Compiler:"gc", Platform:"linux/amd64"}

Server Version: version.Info{Major:"1", Minor:"21", GitVersion:"v1.21.1", GitCommit:"5e58841cce77d4bc13713ad2b91fa0d961e69192", GitTreeState:"clean", BuildDate:"2021-05-12T14:12:29Z", GoVersion:"go1.16.4", Compiler:"gc", Platform:"linux/amd64"}

Having plugin "krew" in kubectl, if don't have it, you can install it according to the information provided at this website.

https://krew.sigs.k8s.io/docs/user-guide/

## Steps

* Install Kubectl plugin

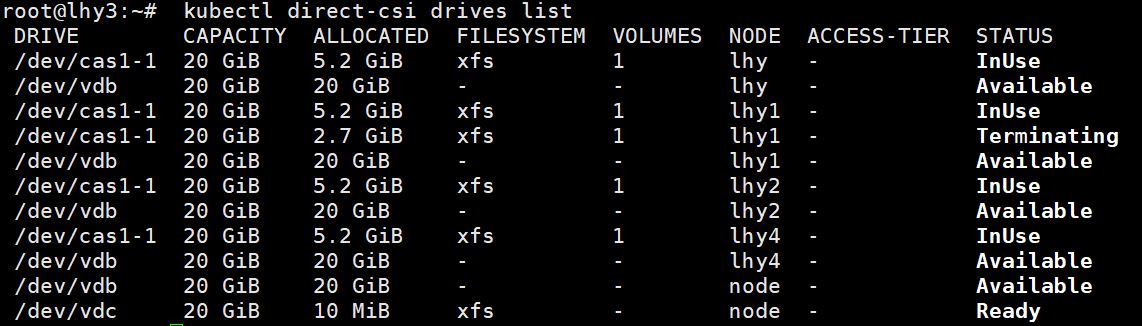
$kubectl krew install direct-csi

* Use the kubectl plugin, install direct-csi driver in your kubernetes cluster

$kubectl direct-csi install

* The kubectl plugin makes it easy to discover drives in your cluster

$kubectl direct-csi drives list

For example:  


* You can filter drives you need in all nodes, for example:

$kubectl direct-csi drives format --drives='/dev/cas1-1'

Then the drives' status you have chosen will turn to "Ready" form "Available".

After create minio tenant, the drives' status will turn to "InUse" from "Ready".

* Verify Installation

Check if all the pods are deployed correctly. i.e. they are 'Running'.

$ kubectl -n direct-csi-min-io get pods  
NAME READY STATUS RESTARTS AGE  
direct-csi-min-io-7f9475b58c-b8rsj 2/2 Running 0 2d18h  
direct-csi-min-io-7f9475b58c-tlw8x 2/2 Running 0 2d15h  
direct-csi-min-io-7f9475b58c-v7s7k 2/2 Running 6 2d18h  
direct-csi-min-io-7j6rk 3/3 Running 4 2d18h  
direct-csi-min-io-d7dnk 3/3 Running 0 2d14h  
direct-csi-min-io-mnc99 3/3 Running 0 2d17h  
direct-csi-min-io-qprz7 3/3 Running 0 2d18h  
direct-csi-min-io-xg4kp 3/3 Running 0 2d13h  
directcsi-conversion-webhook-794d98f585-4wvh6 1/1 Running 0 2d15h  
directcsi-conversion-webhook-794d98f585-c5nq5 1/1 Running 0 2d15h  
directcsi-conversion-webhook-794d98f585-jjnml 1/1 Running 0 2d18h

# Deploy the MinIO Operator and Create Tenants

## Install the MinIO Operator

* Download the [latest stable  
  version](https://github.com/minio/operator/releases/) of the MinIO   
  Kubernetes Plugin:

$wget https://github.com/minio/operator/releases/download/v4.2.2/kubectl-minio\_4.2.2\_linux\_amd64 -O kubectl-minio

$chmod +x kubectl-minio

# The following command may require sudo if the current user does not have root permissions

$mv kubectl-minio /usr/local/bin/

* Run the following command to verify installation:

$kubectl minio version

The output should reflect the installed version of the MinIO Kubernetes Plugin.

* Run the following command to initialize the Operator:

$kubectl minio init

* Run the following command to verify the status of the Operator:

$kubectl get pods -n minio-operator

* The output resembles the following:

NAME READY STATUS RESTARTS AGE

console-7c855c9789-x7llm 1/1 Running 0 2d15h

minio-operator-66849f98b7-8pv7r 1/1 Running 0 2d15h

## Access the operator console through the browser

* Run the following command to create a local proxy to the MinIO Operator  
  Console:

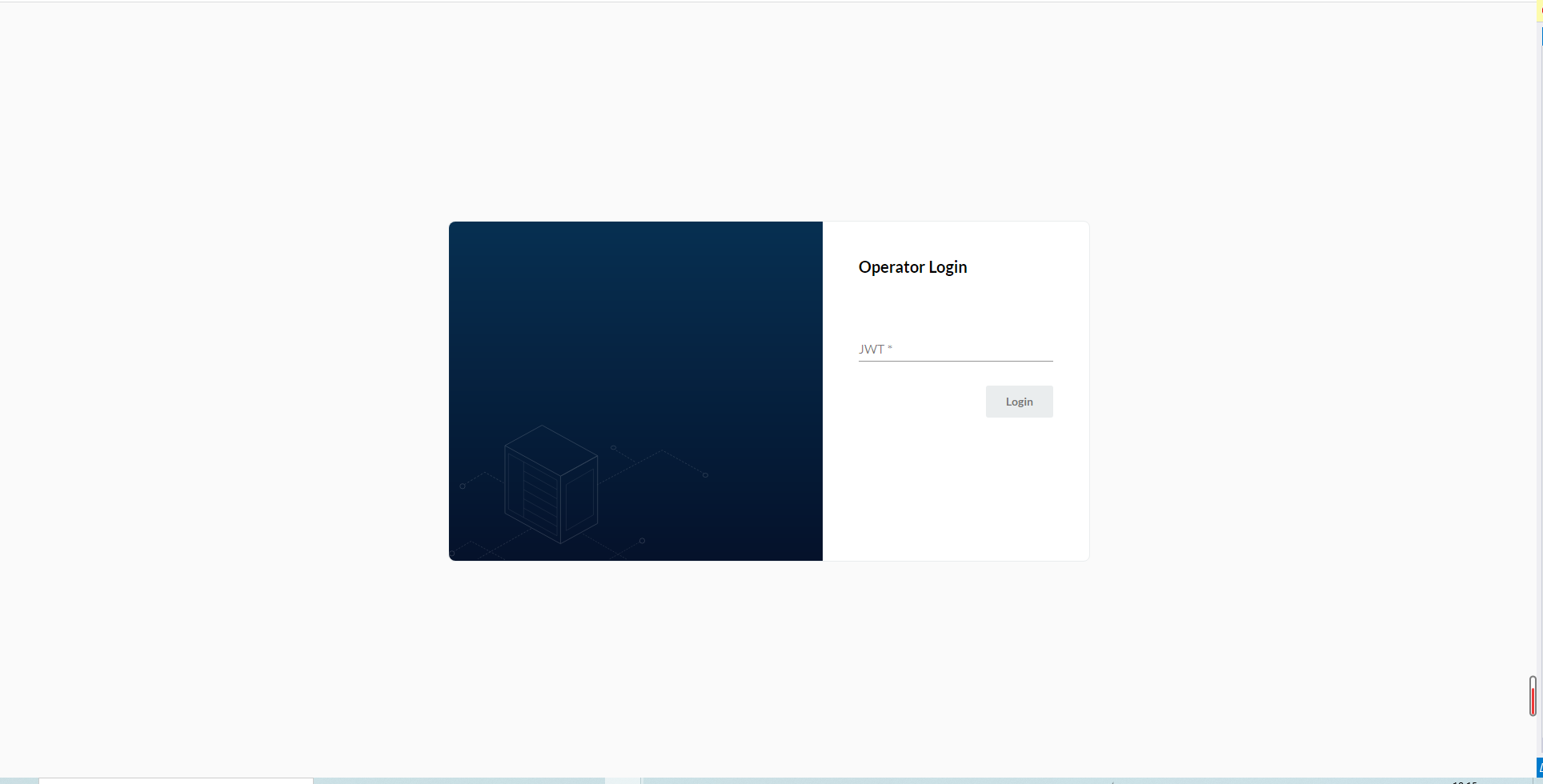
$kubectl minio proxy -n minio-operator

* The output resembles the following:

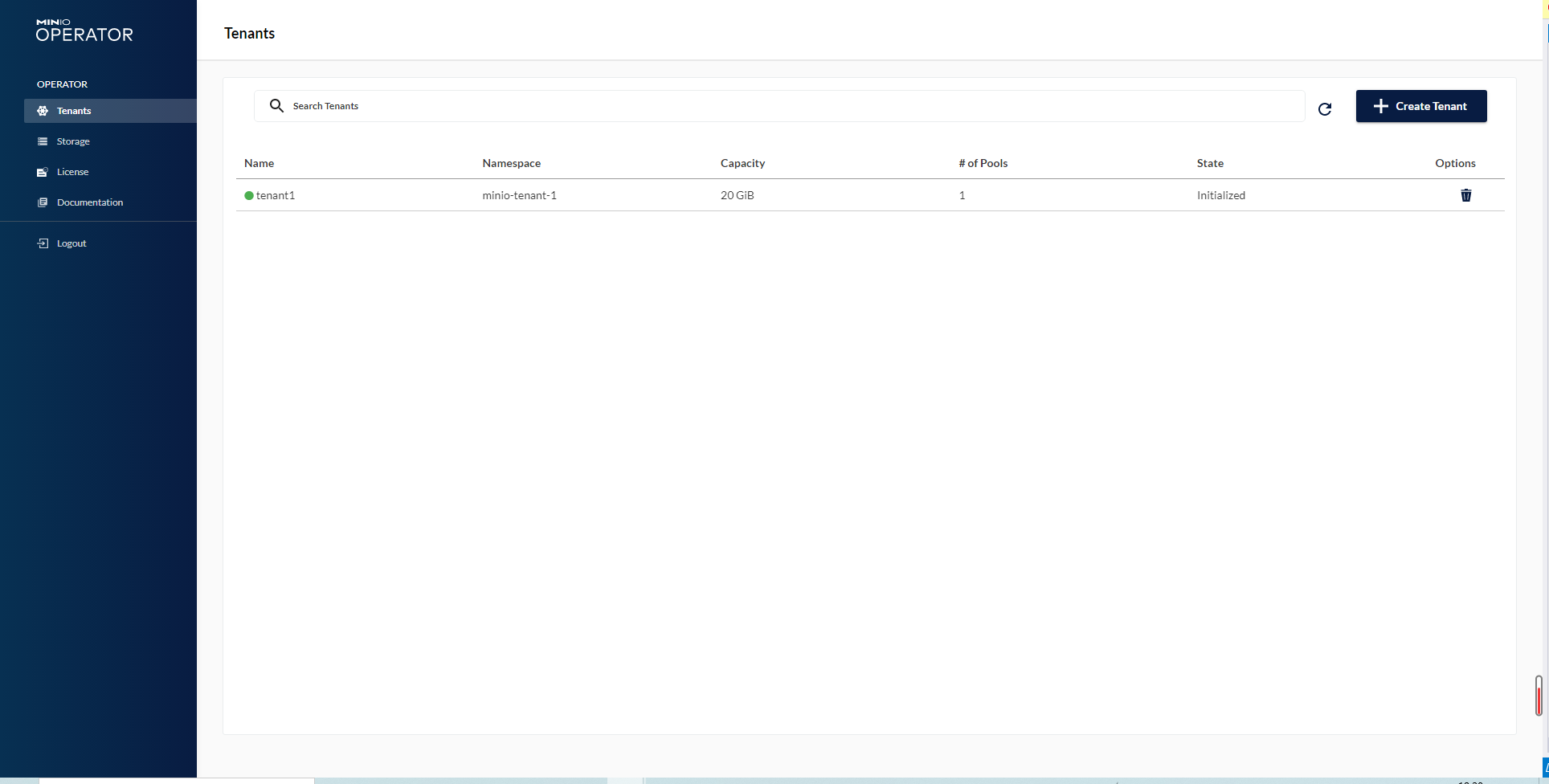
$kubectl minio proxy -n minio-operator

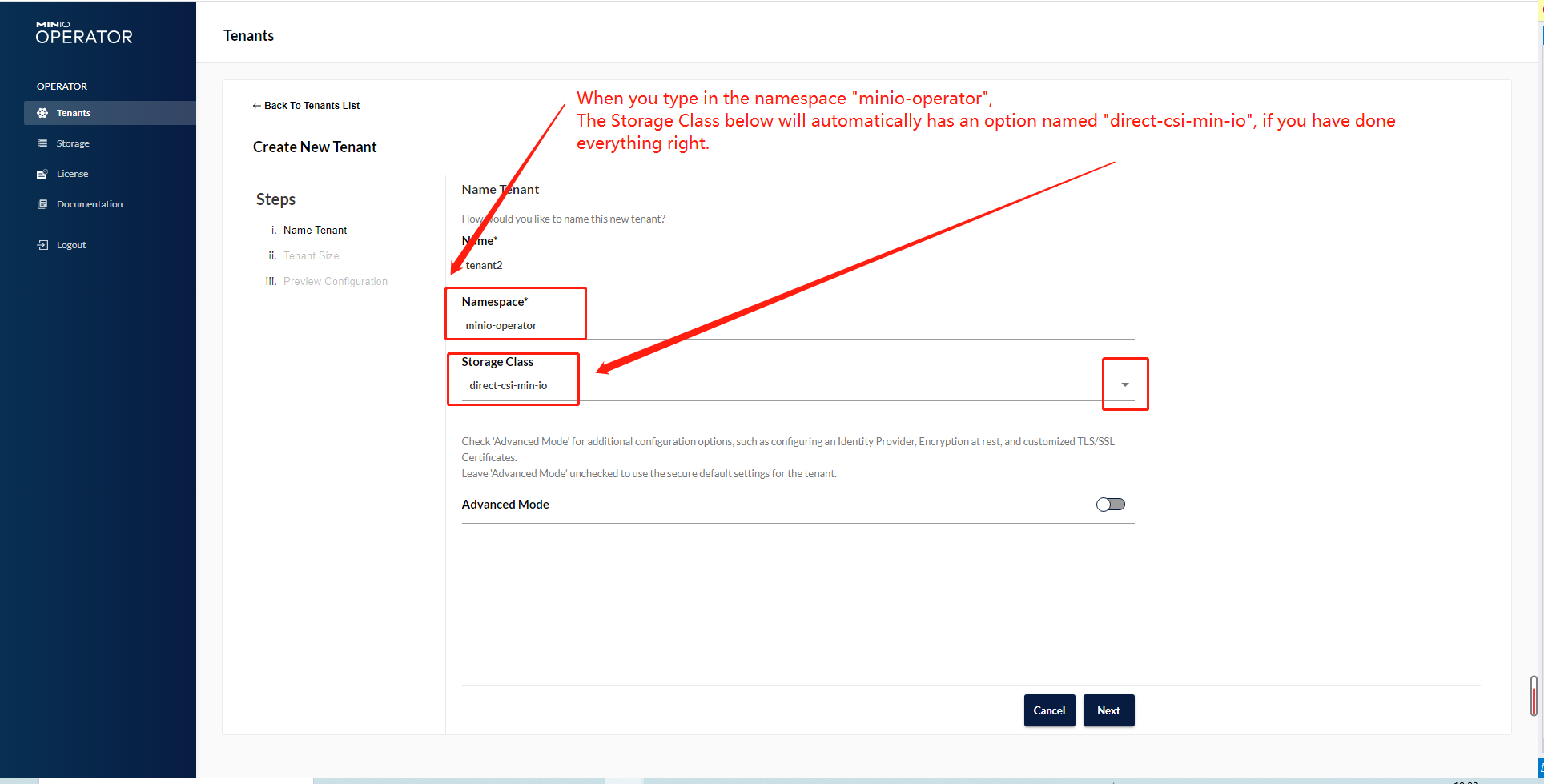
Starting port forward of the Console UI.  
  
To connect open a browser and go to http://localhost:9090  
  
Current JWT to login: eyJhbGciOiJSUzI1NiIsImtpZCI6Il9tbThJMzlMdnpDQV92bVI4Y1ZXZ195VGZoOWFaTS1DSlMzbktVc29iWkEifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJtaW5pby1vcGVyYXRvciIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJjb25zb2xlLXNhLXRva2VuLXF4bnE1Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImNvbnNvbGUtc2EiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJlZTIwMTBhOS1hMDdmLTRiZGEtYmVmMS01MTY3Zjk0YmFmMzYiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6bWluaW8tb3BlcmF0b3I6Y29uc29sZS1zYSJ9.GviXxBLfsBbinTrCTlH7S\_9ch8UmGMjMMq-mEzaK6GpmJBEu3S0kCvYLCsPvllHUI60dAPa0LyTq760AKLOYzoqXJxjX2sTw9G\_4Nqr41GMckAbp4NkxWL8h\_yQBq4V0PQ1ro0vojjIvbLJetib2UMN8UXIUVo\_giaRFHE0DkRDDRk4ohoH\_ora\_TU3I8V\_54HBMwLStPyK6iMD5ah0wc3M5EJ3i0y9DkPS-4v-0KbVuE2PH5YbTixDdldpT2BobHBlfCjU0kLaVjJceH7bc6yE007ehwppzqTb9BktW0z4kp2uASUh28SRfrRbm6PKpIpD8M6W7KufvDTVTMGb2cw  
Forwarding from 0.0.0.0:9090 -> 9090

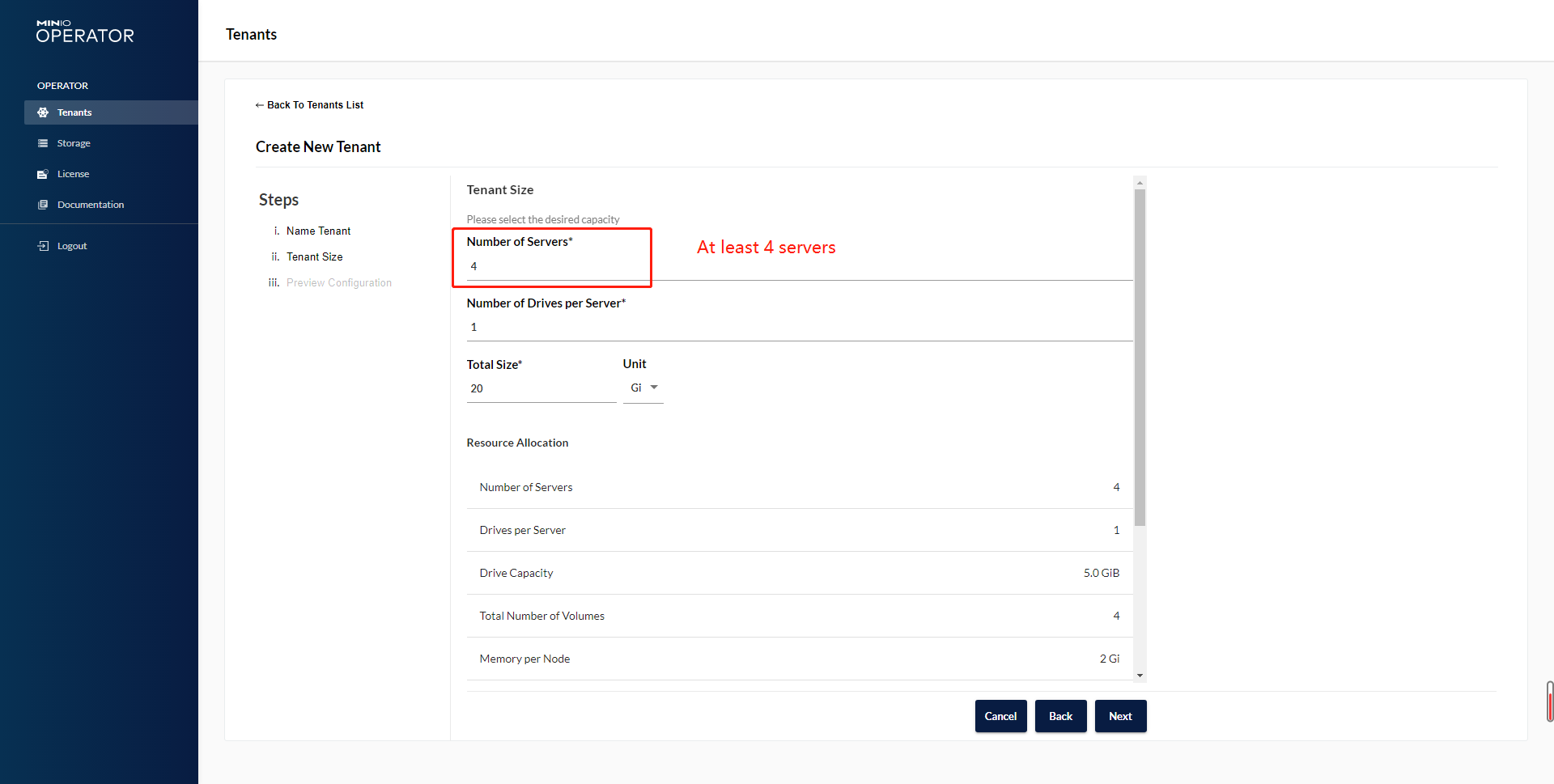
Open your browser, and type in http:// your public IP address of the server:9090



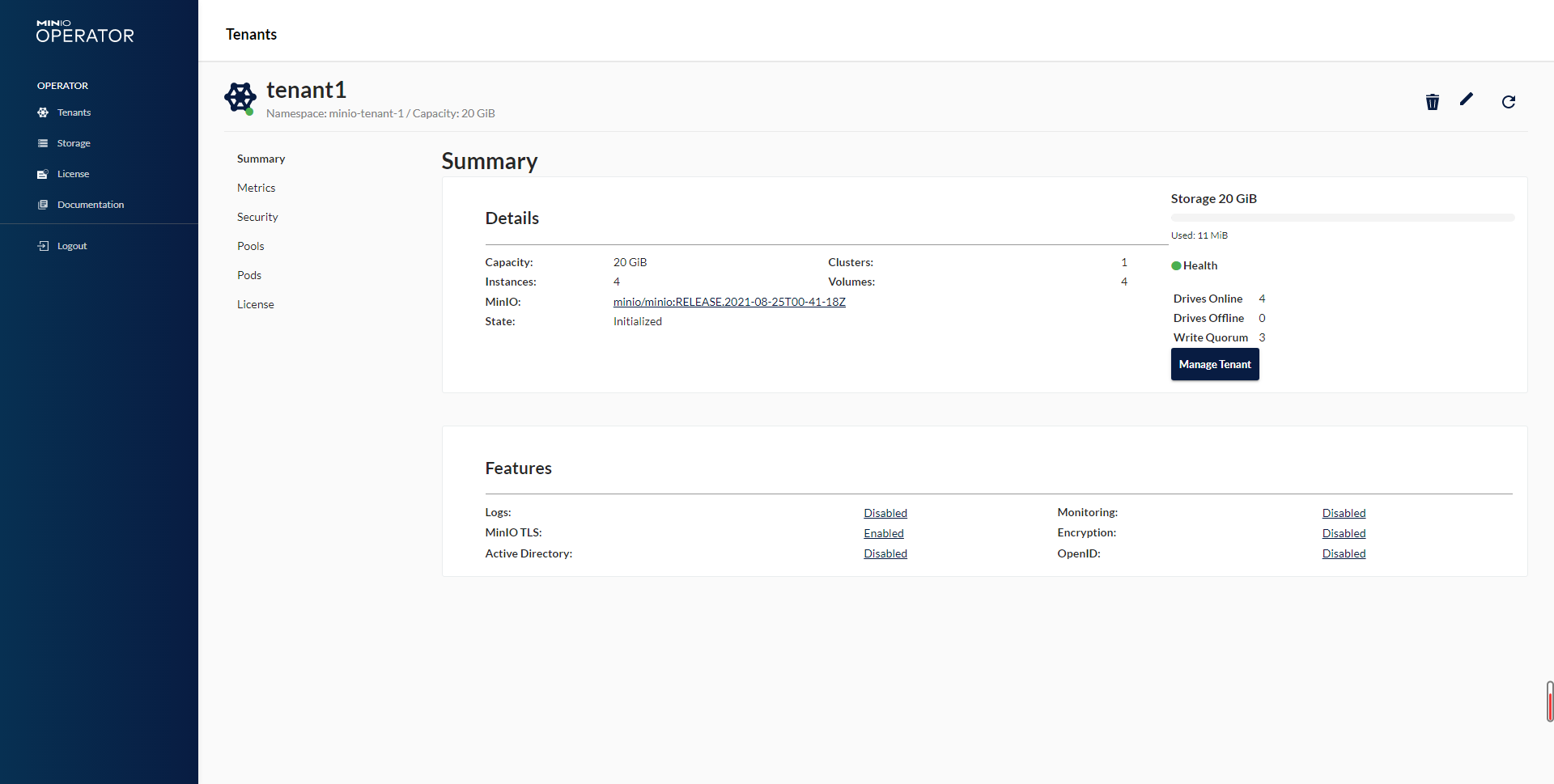
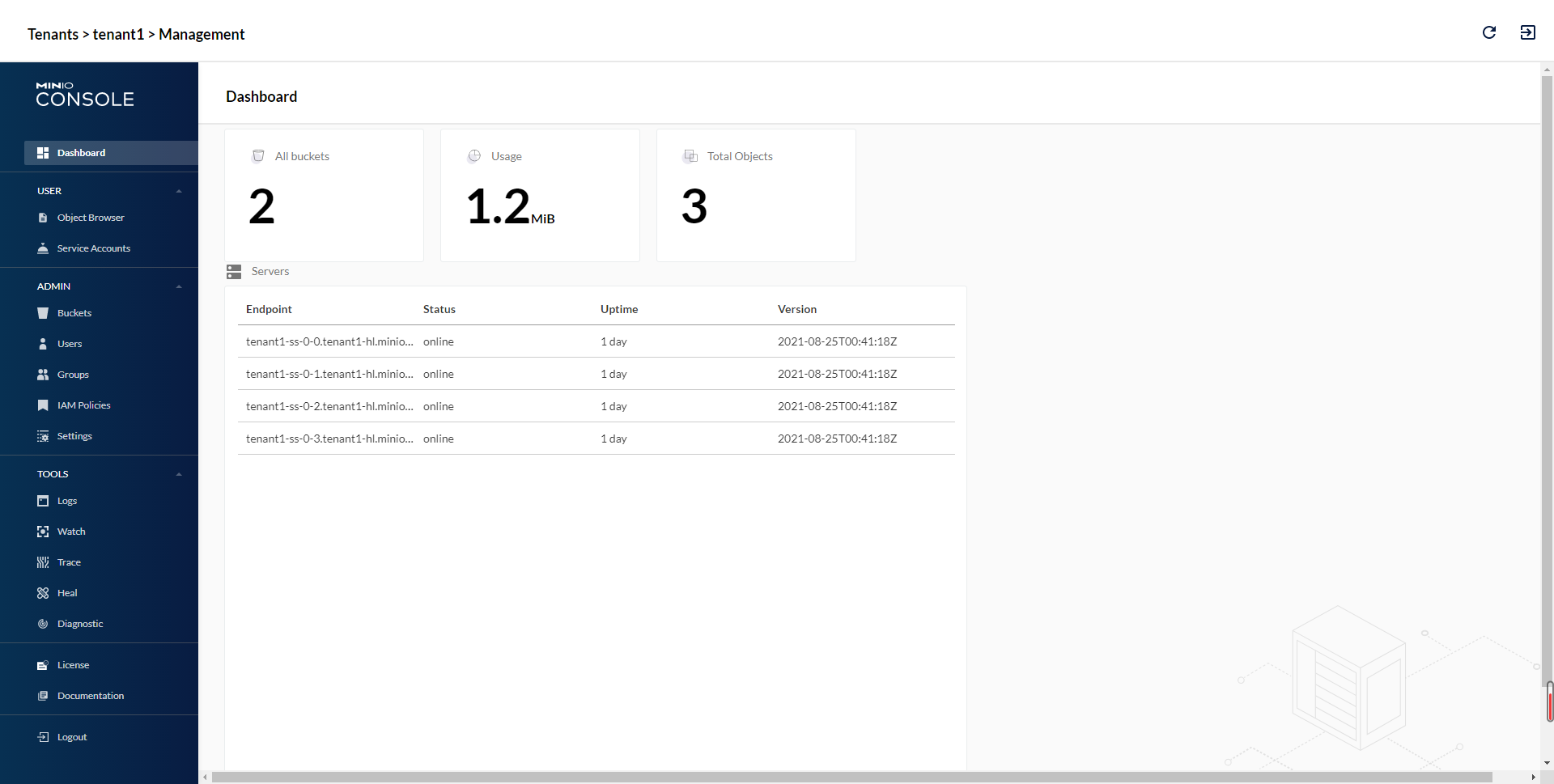
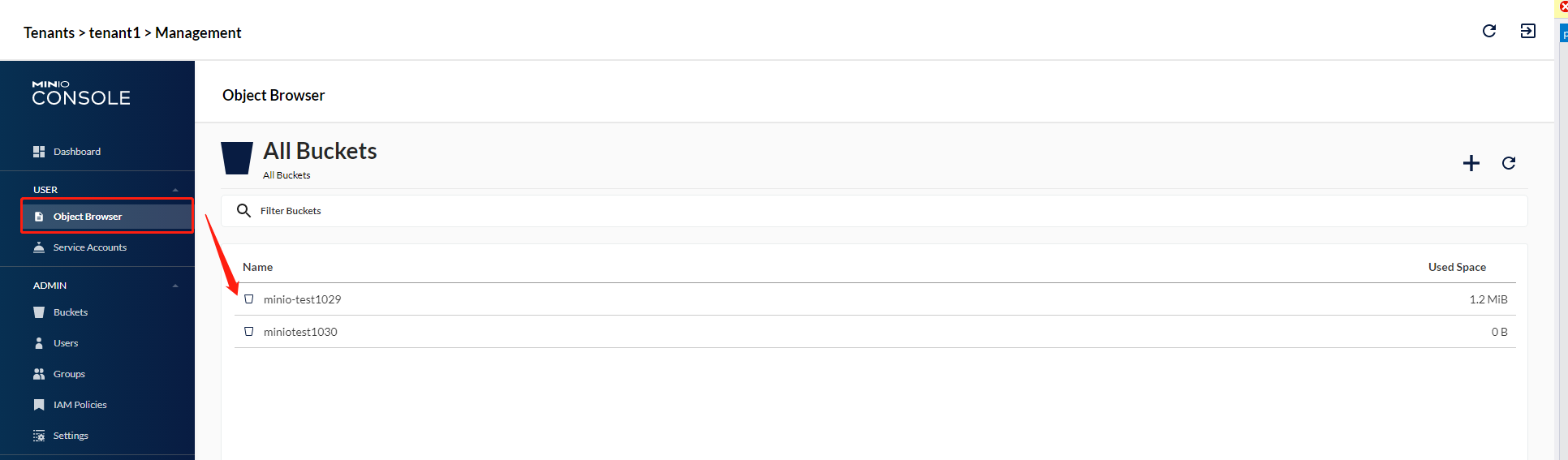
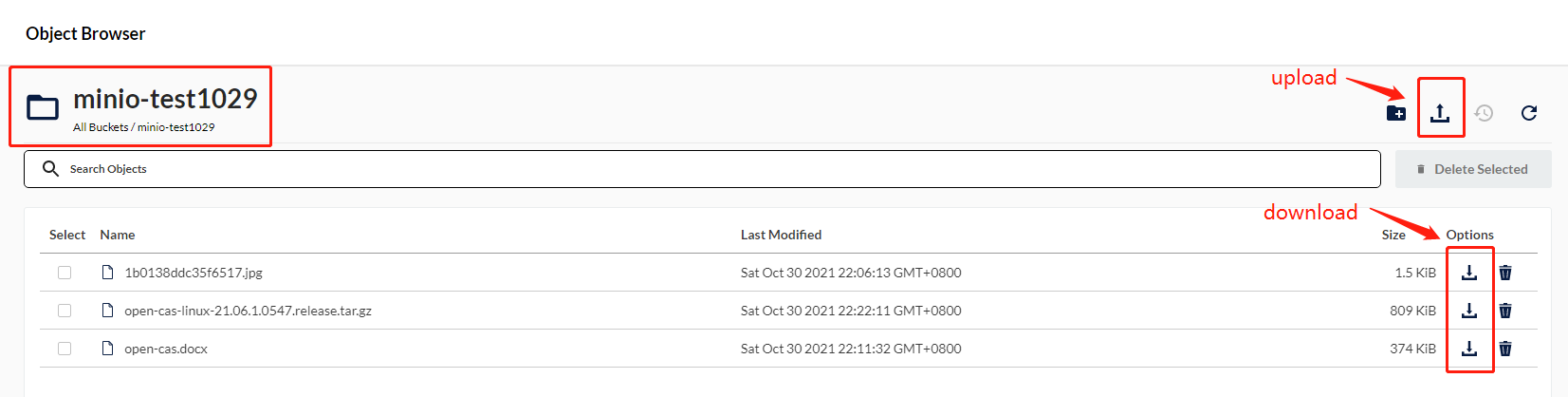
When you login in, you will see:



Click "Create Tenant" to create a new tenant:  




You can get more details of deploying a minio tenant from [here](https://docs.min.io/minio/k8s/tenant-management/deploy-minio-tenant.html#deploy-minio-tenant).

The tenant you create looks like this:  
   
Then you click "Manage Tenant" to load the MinIO Console:  
   
You can see the buckets throuth this:  
   
You can upload your files to this bucket and download files from it:  


## Access the operator console through command line

You can also create and manage your minio tenant by command line in your terminal.

* Create minio tenant:

$kubectl minio tenant create tenant1 --namespace minio-tenant-1 --storage-class direct-csi-min-io --servers 4 --volumes 4 --capacity 20Gi -o > tenant.yaml  
$kubectl apply -f tenant.yaml  
$kubectl get pod -n minio-tenant-1 -owide  
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES  
tenant1-ss-0-0 1/1 Running 0 45h 10.244.6.17 lhy1 <none> <none>  
tenant1-ss-0-1 1/1 Running 0 45h 10.244.7.4 lhy <none> <none>  
tenant1-ss-0-2 1/1 Running 0 45h 10.244.1.16 lhy4 <none> <none>  
tenant1-ss-0-3 1/1 Running 0 45h 10.244.5.24 lhy2 <none> <none>

* Check the accesskey and secretkey from tenant.yaml:

$cat tenant.yaml

---

apiVersion: v1

data:

accesskey: MDI3NmVmOTUtNjdjNS00NWI3LWI2ZmQtYzc5YTIxY2Y1NjFl

secretkey: MjI4YTkwOGEtYWEyZS00ZjBhLWI3YTEtNjM0ZWFkZjNhMThi

kind: Secret

metadata:

creationTimestamp: null

name: tenant1-creds-secret

namespace: minio-tenant-1

---

$echo MDI3NmVmOTUtNjdjNS00NWI3LWI2ZmQtYzc5YTIxY2Y1NjFl | base64 -d

0276ef95-67c5-45b7-b6fd-c79a21cf561e

$echo MjI4YTkwOGEtYWEyZS00ZjBhLWI3YTEtNjM0ZWFkZjNhMThi | base64 -d

228a908a-aa2e-4f0a-b7a1-634eadf3a18b

* Download the minio client:

$wget https://dl.min.io/client/mc/release/linux-amd64/mc

$chmod +x mc

• Manage the minio tenant you created:

$kubectl get svc -n minio-tenant-1

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

minio NodePort 10.107.42.103 <none> 443:30232/TCP 45h

tenant1-console NodePort 10.103.59.238 <none> 9443:32645/TCP 45h

tenant1-hl ClusterIP None <none> 9000/TCP 45h

$./mc --insecure alias set myminio https://10.107.42.103 0276ef95-67c5-45b7-b6fd-c79a21cf561e 228a908a-aa2e-4f0a-b7a1-634eadf3a18b

"0276ef95-67c5-45b7-b6fd-c79a21cf561e" and "228a908a-aa2e-4f0a-b7a1-634eadf3a18b" are access key and secret key respectively.

* Show the bucket in your tenant:

$./mc --insecure ls myminio

[2021-10-29 22:09:19 CST] 0B minio-test1029/

[2021-10-30 21:20:54 CST] 0B miniotest1030/

* Upload file:

$./mc --insecure cp ./../open-cas-linux-21.06.1.0547.release.tar.gz myminio/minio-test1029/

* You can get help by "--help":

$./mc --help

NAME:

mc - MinIO Client for cloud storage and filesystems.

USAGE:

mc [FLAGS] COMMAND [COMMAND FLAGS | -h] [ARGUMENTS...]

COMMANDS:

alias set, remove and list aliases in configuration file

ls list buckets and objects

mb make a bucket

rb remove a bucket

cp copy objects

mirror synchronize object(s) to a remote site

cat display object contents

head display first 'n' lines of an object

pipe stream STDIN to an object

share generate URL for temporary access to an object

find search for objects

sql run sql queries on objects

stat show object metadata

mv move objects

tree list buckets and objects in a tree format

du summarize disk usage recursively

retention set retention for object(s)

legalhold manage legal hold for object(s)

diff list differences in object name, size, and date between two buckets

rm remove objects

version manage bucket versioning

ilm manage bucket lifecycle

encrypt manage bucket encryption config

event manage object notifications

watch listen for object notification events

undo undo PUT/DELETE operations

anonymous manage anonymous access to buckets and objects

tag manage tags for bucket and object(s)

replicate configure server side bucket replication

admin manage MinIO servers

update update mc to latest release

GLOBAL FLAGS:

--autocompletion install auto-completion for your shell

--config-dir value, -C value path to configuration folder (default: "/root/.mc")

--quiet, -q disable progress bar display

--no-color disable color theme

--json enable JSON lines formatted output

--debug enable debug output

--insecure disable SSL certificate verification

--help, -h show help

--version, -v print the version

TIP:

Use 'mc --autocompletion' to enable shell autocompletion

VERSION:

RELEASE.2021-10-07T04-19-58Z