# Datera OpenShift 3.11 Deployment Guide



## Document Revisions

| Date | Description |
|---|---|
| June 5, 2020 | Launch |

This document expands on the strengths and availability of Datera with OpenShift, and guides you on configuring the environment.

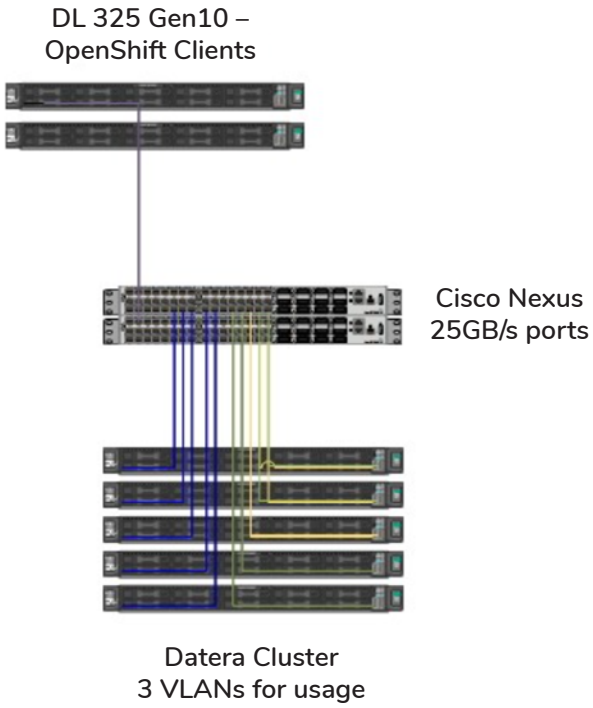
# 1. Datera Architecture

Datera uses standard X86 servers to provide low latency storage to clients. On each physical server, we typically utilize 2 front end interface ports on 1 NIC and 2 backend traffic port on 1 NIC.

Traffic between front and back end traffic can be separate through VLAN tagging on the switch. Front ports can be configured to use LACP for Bandwidth optimization. Please refer to the Datera Networking guide for further details.

Whether you deployed nodes on Layer 2 or as a BGP Peer on Layer3 networking, our behavior is identical. Please consult our support or System Engineers to help you determine the architecture needed.

An example of a Layer 2 storage environment below was used for this guide:




## Network details

**Front-side vlans**

Blue – subnet 172.17.100.0/24 – Datera PoC Front-side A
Blue – subnet 172.17.101.0/24 – Datera PoC Front-side B

**Back-side vlans**

Yellow – subnet 10.200.20.0/24 – Datera Storage 200
Yellow – subnet 10.220.20.0/24 – Datera Storage 220

DHCP is enabled for all 4 VLANs.
For iLOs - VLAN 104 – subnet 172.17.104.0/21
– iLO/Management – DHCP

# 2. Executive Summary & Introduction to Datera and RedHat's OpenShift.

Datera is a fully disaggregated scale-out storage platform, that runs over multiple standard protocols (iSCSI,Object/S3), combining both heterogeneous compute platform/framework flexibility (HPE, Dell, Fujitsu, Intel, and others) with rapid deployment velocity and access to data from anywhere.

Red Hat® OpenShift® is a hybrid cloud and enterprise Kubernetes application platform.

The Datera Data Services Platform natively supports powerful OpenShift™ integration via CSI. Datera leverages the Container Storage Interface (CSI) which gives OpenShift™/Kubernetes (K8s) enterprise customers the peace of mind of a future-proof data services platform that is ready for diverse and demanding workloads.

Datera enables OpenShift/K8s-based applications to expose enterprise storage system functionality via API plug-in to the Datera software-defined storage platform.

Datera gives OpenShift and Kubernetes (K8s) enterprise customers the peace of mind of a future-proof data services platform that is ready for diverse and demanding workloads — as K8s continues to dominate the container orchestration arena, it is likely to containerize higher-end workloads, as well.

The Container Storage Interface (CSI) is a standard for exposing arbitrary block and file storage systems to containerized workloads on Container Orchestration Systems (COs) like Kubernetes. Using CSI third-party storage providers, such as Datera, can write and deploy plugins exposing new storage systems in Kubernetes without ever having to touch the core Kubernetes code.

Datera's CSI driver deeply integrates with the K8s runtime. It allows deploying entire stateful multi-site K8s clusters with a single K8s command and pushing application-specific telemetry to the Datera policy engine, so that it can intelligently adapt the data fabric. Datera's powerful storage classes, and policy driven workloads are a natural fit with Kubernetes, and our deep CSI integration will be covered in this paper.

Datera provides IT a private/hybrid cloud data platform to consolidate both traditional enterprise, bare metal, virtualized and modern cloud-native workloads. IT operators gain the flexibility to plan, deploy and scale their compute resources independently from their Datera storage resources, while application owners can self-service and consume infrastructure as they go.

# 3. Datera Product Overview

Datera provides IT a private/hybrid cloud data platform to consolidate both traditional enterprise, bare metal, virtualized and modern cloud-native workloads.

IT operators gain the flexibility to plan, deploy and scale their compute resources independently from their Datera storage resources, while application owners can self-service and consume infrastructure as they go.

The following table shows how Datera's policy driven, automated, disaggregated storage solution works seamlessly with OpenShift and K8s to offer the most performant, scalable, and flexible platform for your container storage workloads:

| OpenShift & K8s Concept | Datera Concept |
|---|---|
| Manifests | **Templates + CSI driver**<br>Declarative policy (intents) and telemetry (operationalization)<br>Label-based provisioning with seamless integration in K8s manifests |
| Namespaces | **Tenancy**<br>Governance (operationalization of policy)<br>Single authentication/access/quota mechanism |
| Quotas | **Tenancy + Quotas**<br>Fine-grained controls at tenant and volume level for sandboxing storage<br>Containment for noisy neighbors and rogue resource scaling<br>Makes K8s more safely consumable |
| Resource Pools<br>"Tainting" | **Tenancy + Resource Pools**<br>Ability to restrict media placement to a subset of nodes/resources |
| Storage Classes | **Application Classes and Instances + Live Data Mobility**<br>Just-in-time non-disruptive resource provisioning, driven by policy:<br>• No application downtime<br>• No need to respin pods<br>• No need to recreat PVs/PVCs<br>Live policy (label) changes in AppClasses and/or AppInstances |

# 4. Setting up OpenShift on VMware with Datera

We can begin by first defining storage capacity for usage, and create all the prerequisites for a dedicated Management node and worker nodes.

From the Datera side, we are going to be using SSD HP DL380 nodes with Datera 3.3.3 code.
The Virtual machines design will host RHEL 7.8 and OpenShift 3.11

In total, we will create 3 VM's:

Openshift® Master VM
Openshift® Infrastructure
VM Openshift® App VM

We will deploy the DNS and DHCP services on the Master node.

# 5. Configuring storage from Datera

Storage can be configured to multiple Openshift nodes. To create storage on the Datera side, simply grab the IQN for the OpenShift master and worker nodes.

RHEL needs the following utility to work with iSCSI.

```
iscsi-initiator-utils
```

After the utility is installed, it will generate the IQN name, and assign it under the following directory.
You should see something like this:

```
/etc/iscsi/initiator.name
```

If you are running firewalld or iptables, you need to make sure you add port 3260/tcp as exception (allow it through firewall) so that communication between client and iSCSI datastore is not blocked. With firewall you can do that as:

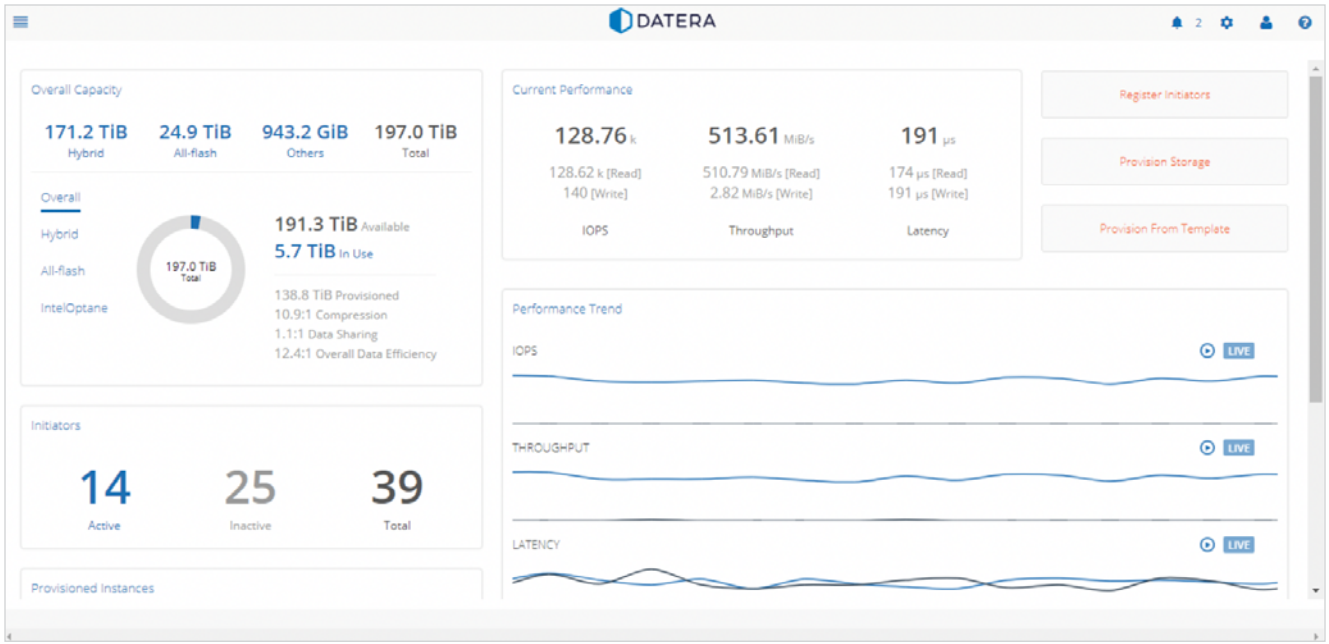
```
cat initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:7c4f89fede22


[root@server1 ~]#  firewall-cmd --add-port=3260/tcp --permanent
success
[root@server1 ~]#  firewall-cmd --reload success
[root@server1 ~]#  firewall-cmd --list-ports 3260/tcp
```

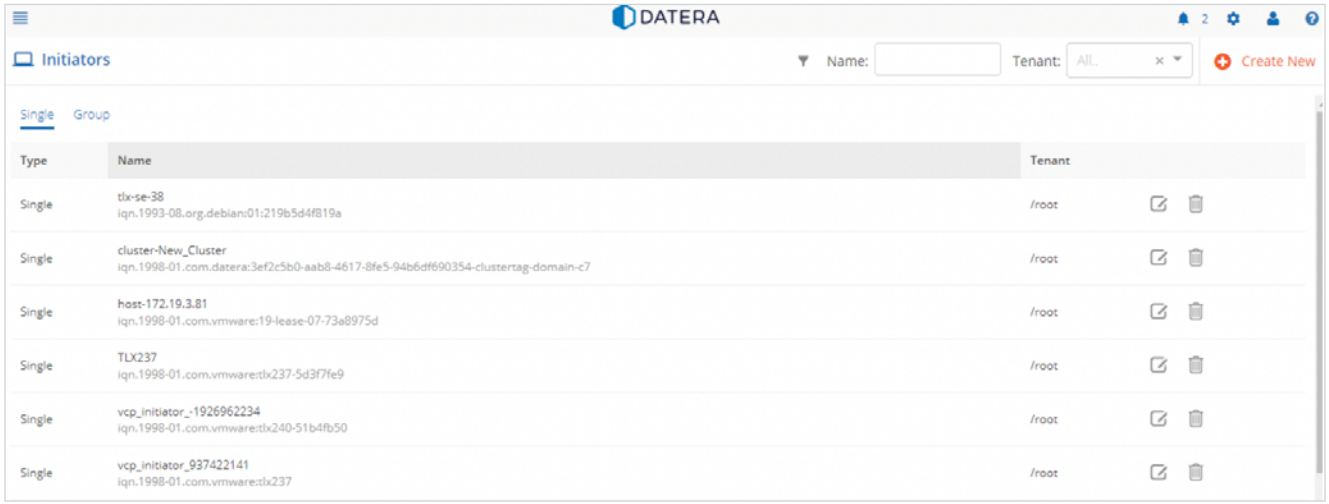
# 6. Provisioning Storage on Datera

These tasks can be done through the rest API or alternatively, through an orchestrator. Now that you have the iSCSI names, we can begin to register the IQNs in the Datera UI Start by:

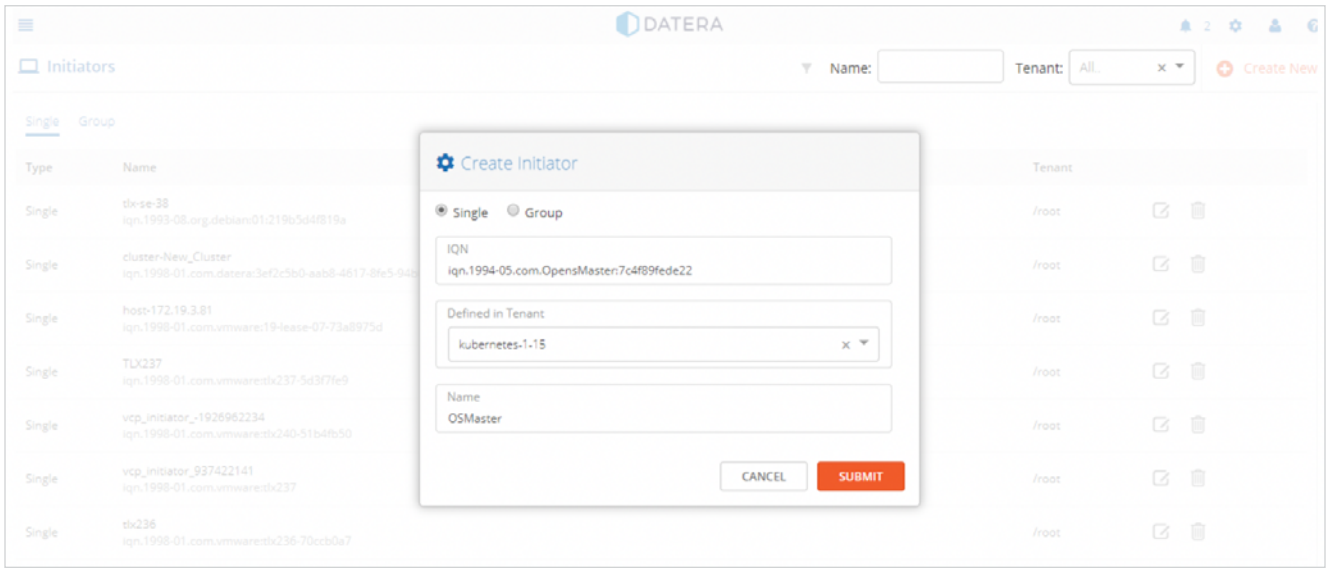
Log into your Datera System...



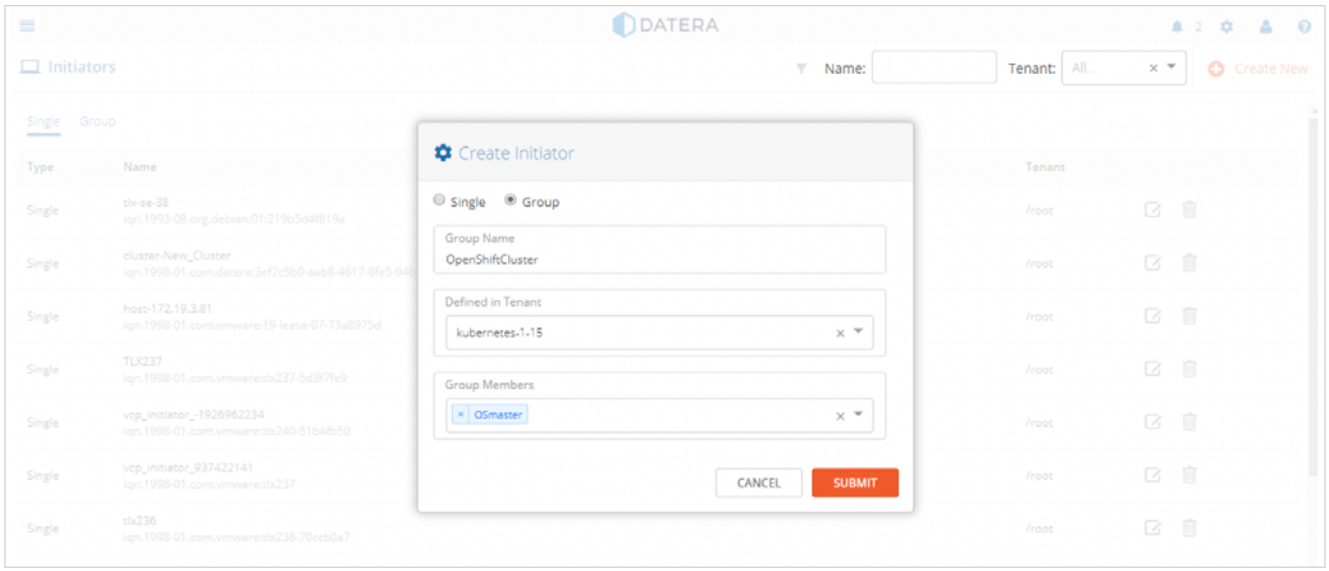

Select the 3 horizontal lines on the top left, or hamburger, and select initiators:



Start by registering the Master, as well as any workers needed. (just repeat for all machines needed)



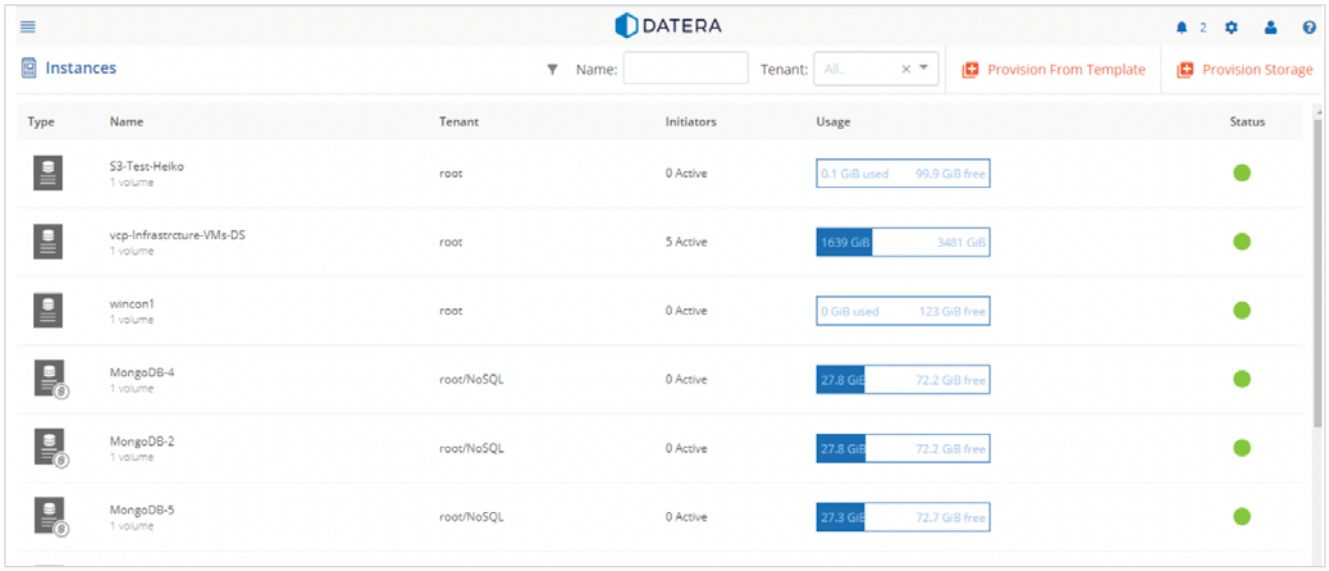

Also create a group to simplify:



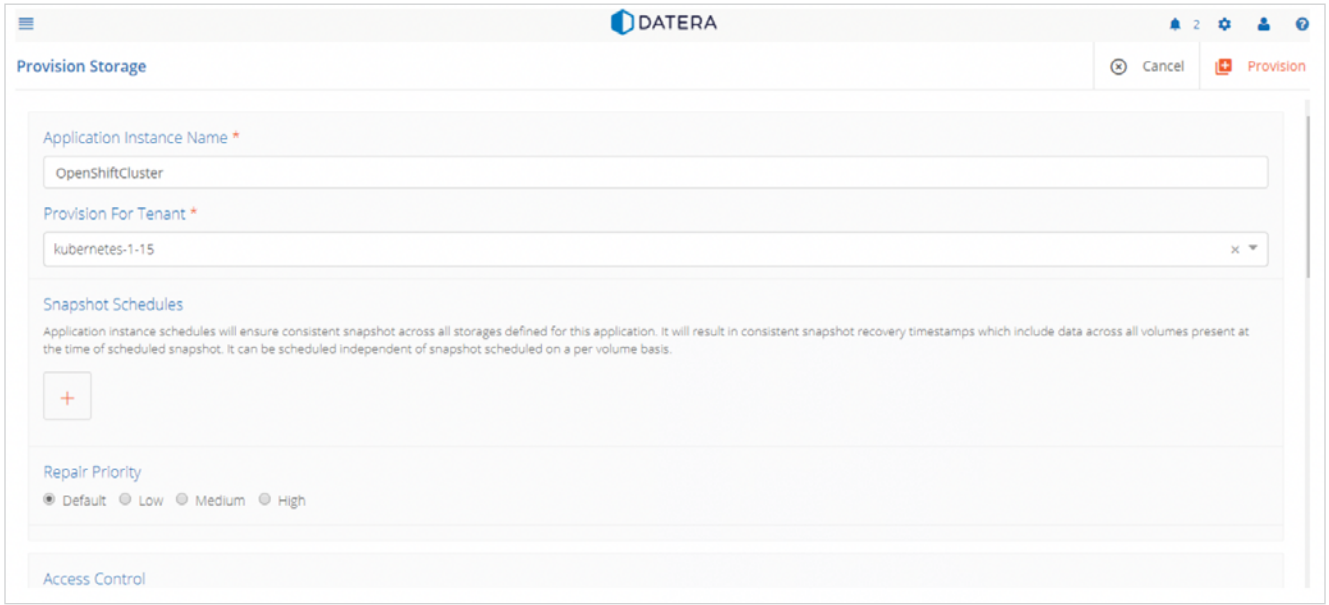
Add the servers underneath it by selecting them under Group Members.

Next, we will want to assign at least 3 volumes to the OpenShift Cluster. This is completely up to you to decide the size of the configuration.

Select Instances from the Hamburger in the upper right corner:

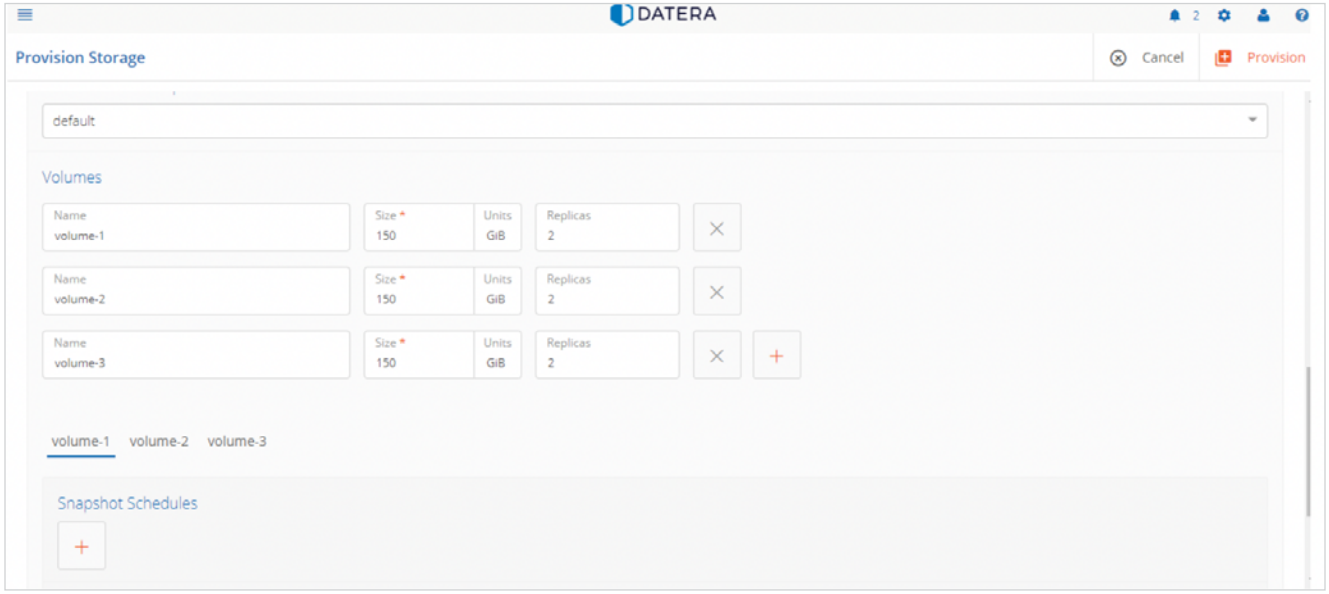



Select Provision Storage from the top right.
Give it a name, and define the tenant.



Define the Initiator Group (this will make it easier to add systems in the future if you expand the cluster)


Add the volumes:




Once you are done, select Provision.

# 7. Prerequisites

Installing OCP requires:

At least three physical or virtual RHEL 7+ machines, with fully qualified domain names (either real world or within a network) and password - less SSH access to each other.

These systems must be reachable through DNS names.

A valid Red Hat subscription, to register the nodes for Repos.

For a smaller deployment, you can register hostnames for all the members of the OCP in the /etc/hosts files. It would look something like this for –

```
172.19.111.237 osmaster.tlx.daterainc.com
172.19.112.242 worker1.tlx.daterainc.com
172.19.112.241 worker2.tlx.daterainc.com
```

If you are deploying it in a larger environment, a Wildcard DNS resolution that resolves the domain to the IP of the node is required, with attributes set accordingly.

Once these prerequisites are configured, you must set up an OCP install.

### Setting up repositories

On all nodes, use subscription-manager to enable the repositories for OCP and attach them.

```
subscription-manager attach --pool=8a85f98c601232d90160125e9c28041
subscription-manager attach --pool=8a85f98c600b1aeb01600b48068608f3
subscription-manager repos --enable="rhel-7-server-rpms" \
--enable="rhel-7-server-extras-rpms" \
--enable="rhel-7-server-ose-3.11-rpms" \
--enable="rhel-7-server-ansible-2.9-rpms"
```

Now that we registered the repos, we will install the Ansible.

### Install OpenShift Container Platform

We will need to deploy Ansible on the Master node. In our example, it's the node with the OS master, but that is an architecture purely dependent on you. Granted there could only be one master.

Run the following to install the required packages on the master:

```
# yum -y install wget git net-tools bind-utils iptables-services bridge-utils bash-completion \
kexec-tools sos psacct
# yum -y update
# reboot
# yum -y install openshift-ansible
```

Now, we will set up the container packages: To install CRI-O:

```
# yum -y install cri-o
```

Docker:

```
# yum -y install docker
```

### Configuring password-less SSH access

Before running the installer on the Master, set up password-less SSH access. This is required by the installer to gain access to the machines. On the Master, run the following command in the Linux prompt:

```
# ssh-keygen
```

Press enter on each prompt

```
[root@OSmaster ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
0a:2a:d7:98:d1:f5:ee:55:48:7f:52:57:27:69:1e:02 root@OSmaster.tlx.daterainc.com
The key's randomart image is:
+--[ RSA 2048]----+
|           E.  ..o|
|             . +.o|
|       .    .  +...|
|    . . . . o ... |
|   . o   Q . + .  |
|    * . o   . o   |
|. = . . . .       |
| o     . .        |
|        .         |
+-----------------+
```

Follow the prompts and press Enter when asked for pass phrase.
Once done, copy the key to all worker nodes –

```
# ssh-copy-id root@worker1.tlx.daterainc.com
```

### Run the installation playbooks

Log in to your Master where the openshift-ansible package is installed.

Edit the example inventory to use your host names and save. By default, it's in /etc/ansible/hosts

The default example configuration:

```
# Ex 1: Ungrouped hosts, specify before any group headers.
## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group
## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110
```

The modified configuration to reflect the environment:

```
[workers]
worker1 ansible_ssh_host = 172.19.112.242
worker2 ansible_ssh_host = 172.19.112.241
```

### Run the `prerequisites.yml` playbook using your inventory file:

OpenShift Container Platform deployment

```
$ ansible-playbook -i <inventory_file> /usr/share/ansible/openshift-ansible/playbooks/prerequisites.yml
```

Next, execute the `deploy_cluster.yml` playbook using your inventory file:

```
$ ansible-playbook-i <inventory_file> /usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml
```

After a successful install, go ahead and set up user roles and groups.

### Understanding roles and authentication

On installation, there are no roles or user accounts created in OCP.
We will need to setup accounts for usage.

On the Master, run the Following:

```
$ oc login -u system:admin
```

Note: Execute all commands on the Master, unless mentioned

Logging in the first time, the system:admin user's configuration file gets created.

There is no password for this system account.

Following command will determine node status:

```
$ oc get nodes
```

Note: For helpful docs: https://docs.openshift.com/container-platform/4.3/welcome/index.html

### Verifying the installation

After the installation completes, login to the Master node using the default admin credentials:

```
# oc login
Username: system:admin
```

Switch to the default created project.

```
# oc project default
```

Verify the web console is installed correctly, default port is 8443, the web console would be found at https://osmaster-dns-name:8443/console

Login with admin and the password you set in the inventory file

```
/etc/ansible/hosts
```

# 8. Creating a new application

**If you want to a deploy a new application using CLI, in the master -**

```
# oc new-app tomcat
```

After this is successful, check the pods.

```
#
oc get pods
tomcat-1-6t7xp        1/1       Running   0          2d
```

Log in with:

```
# oc rsh <pod_name>
```

Note: For more information on how to use Openshift CLI, see:
https://docs.openshift.com/container-platform/3.7/cli_reference/get_started_cli.html

# 9. References

The Datera Storage Platform continues to integrate closely with key partners, and critical platforms. This document covered Datera's deployment and best practices with OpenShift. We also offer tight integration with OpenStack and are RHOSP certified.

For further details and resources, please navigate our site https://www.datera.io and reach out to your sales or support team for questions.