

Consulting Engagement Report

Vietnam International Bank - OpenShift Container Platform Non Prod Deployment

Version 2025-01-20-14-31-51



Table of Contents

1. Preface	4
1.1. Confidentiality, Copyright, and Disclaimer	4
1.2. Audience	4
1.3. Additional Background and Related Documents	4
1.4. Scripts and playbooks	4
1.5. Customer satisfaction survey (CSat)	4
2. Project Information	5
2.1. Originator	5
2.2. Owner	5
2.3. Document Conventions	5
2.3.1. Admonitions and Call-Outs	5
2.3.2. Single-line Commands and Code	5
2.3.3. Multi-line Commands, Code / File Contents	5
2.4. Additional Copies	6
2.5. Participants of the Engagement	6
2.5.1. Red Hat	6
2.5.2. Vietnam International Bank	6
3. Executive Summary	7
4. Overview	8
4.1. About VIB	8
4.2. Purpose And Engagement Approach	8
4.3. Engagement Requirement Criteria	8
5. Implementation Details	9
5.1. Deployed Architecture	10
5.1.1. Red Hat OpenShift Container Platform 4 - Non Prod Architecture	10
5.1.2. Diagrams	10
5.1.3. TLS/SSL Certificates	10
5.2. Technical Implementation	12
5.2.1. Red Hat OpenShift Container Platform 4	12
5.2.1.1. OpenShift Non Prod Environment	12
5.2.2. Prerequisites Preparation	13
5.2.3. Deploying Mirror Registry	13
5.2.3.1. Mirror Registry Installation and Configuration	13
5.2.4. Deploying OpenShift on VMware using User Provisioned Infrastructure (UPI)	18
5.2.4.1. Generate SSH key	18
5.2.4.2. Download Binaries	18

5.2.4.3. Pull Secret	18
5.2.4.4. Creating the Install Configuration File	19
5.2.4.5. Create Manifests and Ignition Configuration	20
5.2.4.6. Creating the Virtual Machines on vSphere	20
5.2.4.7. Provision OpenShift Servers	26
5.2.4.8. The Installation Process	30
5.2.4.9. Running oc Commands	31
5.2.4.10. Cluster Operators Deployment	31
5.2.5. Post Deployment Configurations	32
5.2.5.1. Node Labeling	32
5.2.5.2. Configure Taints on Nodes	32
5.2.5.3. Updating CA Bundle	33
5.2.5.4. Configuring Openshift Ingress	33
5.2.5.5. Authentication	34
5.2.5.6. Configuring NTP and Timezone	36
5.2.5.7. Configure Multus	38
5.2.5.8. Storage Deployment	39
5.2.5.9. Configuring OpenShift Monitoring Stack	41
5.2.5.10. Configuring OpenShift Logging Stack	42
5.2.5.11. Alertmanager configuration	47
5.2.5.12. Internal image registry configuration	48
5.2.5.13. Configuring ETCD backup script	51
5.2.6. CI/CD Pipeline	51
5.2.6.1. CI/CD Pipeline Development for Sample Application using Azure DevOps	51
5.3. Challenges, Resolutions, and Recommendations	68
5.3.1. Failed ODF Installation	68
5.3.1.1. Challenge	68
5.3.1.2. Immediate Resolution	68
5.3.1.3. Future Recommendation	68
6. Recommendations	69
6.1. Technical Next Steps	69
6.1.1. MachineConfigPool separation on Worker and Infra node	69
6.1.1.1. Indication	69
6.1.1.2. Recommendation	69
6.1.2. Setup VM Latency Sensitivity to High on Control Plane	69
6.1.2.1. Indication	69
6.1.2.2. Recommendation	69
6.1.3. Increase numbers and CPU capacity on infra node	69

6.1.3.1. Indication	69
6.1.3.2. Recommendation	69
Appendix A: Glossary	70
Appendix B: Red Hat Subscriptions Overview	71
B.1. Service Level Agreement	71
B.2. Socket Pairs	71
B.3. Virtualization	71
B.3.1. Virtual Datacenter Subscriptions	72
B.4. Layered products	72
B.5. Red Hat Enterprise Linux life cycle	72
Appendix C: Engaging Red Hat Support and Customer Service	75
C.1. Red Hat Customer Portal	75
C.2. Red Hat phone support	75
C.3. Escalation contacts	76
C.4. Customer Service	76
C.5. Production support terms of service	76
C.6. Red Hat Support severity level definitions	77
C.6.1. Severity 1 (urgent)	77
C.6.2. Severity 2 (high)	77
C.6.3. Severity 3 (medium)	77
C.6.4. Severity 4 (low)	78
C.7. Tips for creating a good support case	78

Chapter 1. Preface

1.1. Confidentiality, Copyright, and Disclaimer

This is a customer-facing document between Red Hat, Inc. and Vietnam International Bank ("Client").

Copyright © 2025 Red Hat, Inc. All Rights Reserved. No part of the work covered by the copyright herein may be reproduced or used in any form or by any means – graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems – without permission in writing from Red Hat except as is required to share this information as provided with the aforementioned confidential parties.

This document is not a quote and does not include any binding commitments by Red Hat. If acceptable, a formal quote can be issued upon request, which will include the scope of work, cost, and any customer requirements as necessary.

This Consulting Engagement Report (CER) contains both Red Hat Confidential Information and Client Confidential Information subject to the confidentiality terms of the agreement governing the Services provided by Red Hat.

1.2. Audience

This document is intended for Client technical staff responsible for the environment.

1.3. Additional Background and Related Documents

This document does not contain step by step details of installation or other tasks, as they are covered in the relevant documentation on <http://access.redhat.com/>. Links to the appropriate documents will be made when required.

1.4. Scripts and playbooks

Any scripts provided are being provided as-is, without any form of support or warranty.
All provided scripts can be modified by the customer at will.

1.5. Customer satisfaction survey (CSat)

Red Hat is committed to a quality customer experience and continuous improvement in our services. Your feedback is the most important factor in determining if we've met your expectations and what we can do to improve. You will be receiving a customer satisfaction survey at the conclusion of our engagement. Please take just a few minutes to provide us with an honest snapshot of your experience. In the meantime, if there are any areas of concern, issues or problems on the project, please don't hesitate to reach out to Vidya Krishnaiah.

Chapter 2. Project Information

2.1. Originator

Red Hat Consulting

2.2. Owner

Red Hat Services

2.3. Document Conventions

2.3.1. Admonitions and Call-Outs



Additional detail about a topic will be called out with the "info" symbol.



Tips and advice will appear with a "lightbulb" admonition.



Warnings and strong recommendations will appear next to a caution exclamation.



Crucial advice and information will be appear next to a red exclamation.

2.3.2. Single-line Commands and Code

If instructed to type a single command, it will be displayed in a monospace font and highlighted like this example:

```
ipa -vvv cert-find --all
```

2.3.3. Multi-line Commands, Code / File Contents

If instructed to type multi-line commands, create or modify code blocks and or file contents the information will be monospaced and placed in a callout box like this example:

```
require 'sinatra'

get '/hi' do
  "Hello World!"
end
```

2.4. Additional Copies

To request additional copies of this Consulting Engagement Report, please contact the Red Hat Project Manager identified in [Participants of the Engagement](#) section or your current Red Hat account team.

2.5. Participants of the Engagement

2.5.1. Red Hat

Table 1. Red Hat Participants

Name	Role	E-mail address
Vidya Krishnaiah	Project Manager	vikrishn@redhat.com
Uday Shankar AD	Senior Architect	uad@redhat.com
Joaquin Fernandez Llamas	Senior Consultant	joaquin.fernandez@redhat.com
Vadim Griner	Senior Consultant	vgriner@redhat.com
Muhammad Aslam	Senior Architect	aslam@redhat.com
Sunil Sharmaa	Senior Consultant	sunsharm@redhat.com

2.5.2. Vietnam International Bank

Table 2. VIB Participants

Name	Role	E-mail address
Tien Nguyen Hung	IT Senior Project Manager	tien.nguyenhung1@vib.com.vn
Quoc Ngo Duc	Head of Azure Cloud Solutions	quoc.ngoduc@vib.com.vn
Dung Tran Anh	Azure and Google Cloud Operations Manager	dung.trananh1@vib.com.vn
Minh Tran Dang	Network Data Center and Wintel Senior Specialist	minh.trandang@vib.com.vn
Huy Nguyen Quang	Network Data Center and Wintel Senior Specialist	huy.nguyenquang10@vib.com.vn
Luat Dang Le Trong	Azure and Google Cloud Operations Engineer	luat.dangle@vib.com.vn

Chapter 3. Executive Summary

Red Hat Consulting was engaged by Vietnam International Bank to assist with assist with the implementation of OpenShift Container Platform as the NON Prod environment. This cluster purpose is to become a testbed environment prior to chchanges/deployment in the production environment

Chapter 4. Overview

4.1. About VIB

Vietnam International Bank (VIB) is one of Vietnam’s leading commercial banks, known for its innovative approach to financial services and customer-centric solutions. Established in 1996, VIB has grown to become a prominent player in the banking sector, offering a wide range of services including retail banking, corporate banking, loans, credit cards, and wealth management. The bank is recognized for leveraging advanced technology to enhance its digital banking offerings, providing seamless and secure experiences for its customers. With a focus on sustainable growth and operational excellence, VIB has consistently received awards for its achievements in digital transformation and customer service. Headquartered in Ho Chi Minh City, the bank continues to expand its footprint across Vietnam, supporting the country’s economic development while maintaining a commitment to corporate social responsibility.

4.2. Purpose And Engagement Approach

Red Hat Consulting was engaged by VIB to assist with OpenShift Container Platform Non Prod Deployment which seeks to have Red Hat OpenShift Container Platform deployed on their on-premises data center and running on top of VMware infrastructure

4.3. Engagement Requirement Criteria

Requirement	Details
Cluster deployed on top of Vsphere cluster	<ul style="list-style-type: none"> -
ODF will be deployed on the cluster using Multus	<ul style="list-style-type: none"> • ODF will be configured in worker nodes • ODF Multus will be configured with option 2 (separated private network)
Router, monitoring, and logging stack will use a dedicated infra node	<ul style="list-style-type: none"> -

Chapter 5. Implementation Details

5.1. Deployed Architecture

This content describes the architecture of the deployed and affected systems during this engagement.

5.1.1. Red Hat OpenShift Container Platform 4 - Non Prod Architecture

5.1.2. Diagrams

Diagram below shows high level architecture of the deployed cluster

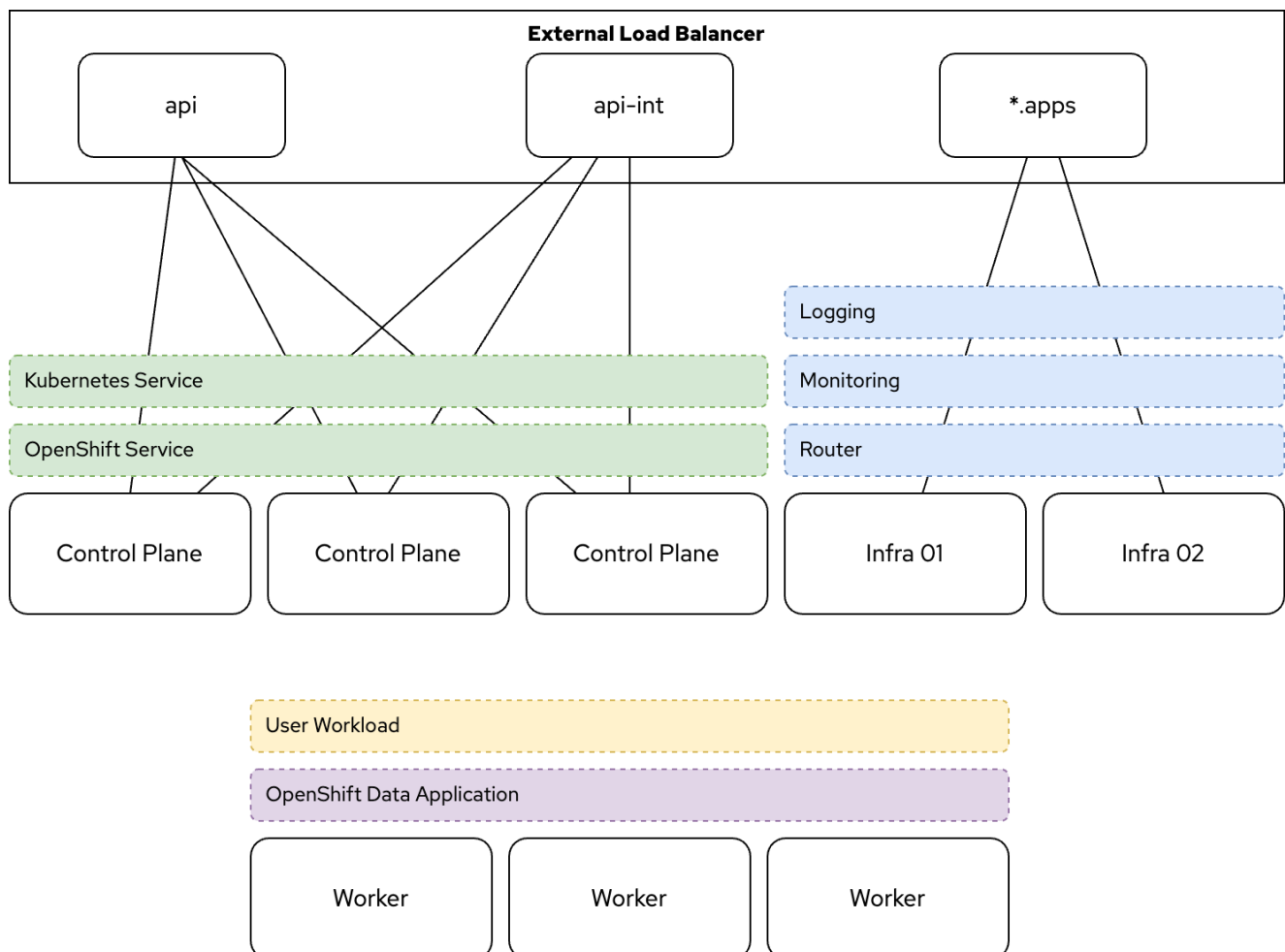


Figure 1. Cluster Diagram

5.1.3. TLS/SSL Certificates

The below table provides details on the TLS/SSL certificates that have been deployed as part of the engagement. To avoid an interruption in service ensure that the certificates are renewed before the expiration date.

Table 3. TLS/SSL Certificates Information

Application	Certificate Location		Certificate Authority	Expiration Date
<i>Ingress Default Certificate</i>	Server	<i>*.apps.workload-01.nonprod-ocp.vib</i>	<i>VIB Signed</i>	<i>11/Feb/2033</i>
	Certificate	<i>ingresscustomcertificatet</i>		

5.2. Technical Implementation

5.2.1. Red Hat OpenShift Container Platform 4

5.2.1.1. OpenShift Non Prod Environment

This cluster of OpenShift will be used as a development environment

5.2.1.1.1. Domain Information

Domain: **nonprod-ocp.vib**

Cluster Name: **workload-01**

5.2.1.1.2. Network Information

Table 4. Network Data

Name	CIDR Block or Value	Comments
Network	/24	10.50.136.0
Pod Network	/15	100.96.0.0
Service Network	/16	100.200.0.0
Host Subnet Length	22	

5.2.1.1.3. Node Information

Table 5. Nodes Data

Server FQDN	vCPU	RAM (GB)	Disk1 (GB)	Role	IP Address
master01	8	16	300	Master	10.50.136.11
master02	8	16	300	Master	10.50.136.12
master03	8	16	300	Master	10.50.136.13
infra01	16	64	300	Infra Router	10.50.136.21
infra02	16	64	300	Infra Router	10.50.136.22
worker01	16	64	300	Worker and Storage	10.50.136.51
worker02	16	64	300	Worker and Storage	10.50.136.52
worker03	16	64	300	Worker and Storage	10.50.136.53

5.2.1.1.4. Storage Information

Table 6. Storage Data

Application	Storage Type	Backing Storage	Size (GB)
Alert Manager	Block	ocs-storagecluster-ceph-rbd	10GB x 2
Metrics	Block	ocs-storagecluster-ceph-rbd	500GB x 2
Logging	Block	Nutanix volumes	500GB x 3
Registry	Files	Nutanix Files	512GB x 1

Table 7. Storage Class

Name	Provisioner	ReclaimPolicy
local-storageclass	kubernetes.io/no-provisioner	Delete
ocs-storagecluster-ceph-rbd	openshift-storage.rbd.csi.ceph.com	Delete
ocs-storagecluster-ceph-rgw	openshift-storage.ceph.rook.io/bucket	Delete
ocs-storagecluster-cephfs	openshift-storage.cephfs.csi.ceph.com	Delete
openshift-storage.noobaa.io	openshift-storage.noobaa.io/obc	Delete

5.2.2. Prerequisites Preparation

The following services are pre-requisites for OpenShift 4.16 installation on vSphere ([see more on the official docs for OpenShift](#)) :

- DHCP (If 4.16 >= 4.6 and you intend to provision nodes with UPI using static IPs DHCP isn't required)
- DNS
- Load Balancer
- Webserver (to host bootstrap ignition file)
- vSphere 7 and up

5.2.3. Deploying Mirror Registry

5.2.3.1. Mirror Registry Installation and Configuration

Since the OpenShift cluster will be deployed using a disconnected method, a local mirror registry needs to be provisioned first

5.2.3.1.1. Download and Install Mirror Registry for Red Hat OpenShift

1. Download the mirror-registry.tar.gz package for the latest version of the mirror registry for Red Hat OpenShift found on [the OpenShift console Downloads page](#)
2. Deploy the containerized mirror registry

```
./mirror-registry install --quayHostname <host_example_com> --quayRoot <example_directory_name>
```

and enter all the necessary information

3. Use the user name and password generated during installation to log into the registry by running the following command:

```
podman login -u init -p <password> <host_example_com>:8443 --tls-verify=false
```

5.2.3.1.2. Mirroring Images Using oc-mirror

5.2.3.1.2.1. Installing Packages

1. Download the oc-mirror plugin from [the OpenShift console Downloads page](#)
2. Extract the package, set as executable, and move it to /usr/local/bin folder

```
tar xvzf oc-mirror.tar.gz
chmod +x oc-mirror
sudo mv oc-mirror /usr/local/bin/
```

3. Try running the oc-mirror by run the help command

```
oc-mirror --help
```

5.2.3.1.2.2. Setup credentials

1. Go to [Red Hat Console](#) and get the pull-secret
2. Convert the pull-secret into a json format

```
cat ./pull-secret | jq . > pull_secret.json
```

The content of pull_secret.json will look like the following example

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3B1bnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3B1bnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

3. Copy the file as ~/.docker/config.json

```
mkdir .docker
cp pull_secret.json .docker/config.json
```

4. Encode the mirror registry authentication into base64 and add it to the .docker/config.json file

```
echo -n '<user_name>:<password>' | base64 -w0
```

Edit the .docker/config.json file and add the information

```
"auths": {
  "<mirror_registry_url>": {
    "auth": "result_of_base64_encode",
    "email": "you@example.com"
  }
},
```


5.2.3.1.2.3. Replace the mirror registry TLS

The following describe step on how to replace the SSL certificate in the mirror registry:

1. Stop the quay-app service

```
systemctl stop quay-app.service
```

2. Check the quay folder location using the following command

```
cat /etc/systemd/system/quay-app.service | grep -i conf
```

3. Navigate to the directory indicated by the path and back up the existing ssl.key and ssl.cert files:

```
cd /path/to/quay-config
mv ssl.cert ssl.cert.bak
mv ssl.key ssl.key.bak
```

4. Copy VIB certificate and key to the quay-config folder

```
cp ssl.cert /path/to/quay-config/ssl.cert
cp ssl.key /path/to/quay-config/ssl.key
chown 1001:1001 ssl.cert ssl.key
```

5. Start the registry mirror

```
systemctl start quay-app.service
```

5.2.3.1.2.4. Creating the image set configuration

1. Use the oc mirror init command to create a template for the image set configuration and save it to a file called imageset-config.yaml

```
oc-mirror init > imageset-config.yaml
```

2. Edit the imageset-config.yaml file. In this example, we mirror for 2 versions of OpenShift: 4.16 and 4.17 and download all packages in the Red Hat Operator Index

```
kind: ImageSetConfiguration
```

```

apiVersion: mirror.openshift.io/v1alpha2
archiveSize: 4
storageConfig:
  registry:
    imageURL: <>mirror-registry-url>/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  platform:
    channels:
      - name: stable-4.16
        type: ocp
      - name: stable-4.17
        type: ocp
    graph: true
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.16
      full: true
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.17
      full: true
  additionalImages:
    - name: registry.redhat.io/ubi9/ubi:latest
  helm: {}

```

During deployment we found that mirroring all the packages (setting the full to true) resulting in package mirror failed. We then change the ImageSet to only download specific packages

3. Mirror the imageset to a the mirror registry

```
oc mirror --config=./imageset-config.yaml docker://<mirror-registry-url>
```

4. Check if the image already exist using Quay UI, and check the following files from terminal

```

ls oc-mirror-workspace/results-1670920047
catalogSource-redhat-operator-index.yaml charts imageContentSourcePolicy.yaml mapping.txt release-signatures

```

5.2.3.1.3. Configuring the cluster to use the resources generated by oc-mirror

1. Apply YAML files inside the result folder

```

oc apply -f ./oc-mirror-workspace/results-1670920047/
oc apply -f ./oc-mirror-workspace/results-1670920047/release-signatures/

```

5.2.4. Deploying OpenShift on VMware using User Provisioned Infrastructure (UPI)

The procedure used to deploy the OpenShift cluster on vSphere is a subset of the documentation found online at:

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.16/installing/installing_vsphere/upi/installing-vsphere.html

In order to deploy OpenShift on vSphere, the following steps needs to be performed.

5.2.4.1. Generate SSH key

See: [Generating a key pair for cluster node SSH access](#)

The specific command used for this engagement:

```
ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa
```

5.2.4.2. Download Binaries

See: [Obtaining the installation program](#)

And

See: [Installing the OpenShift CLI by downloading the binary](#)

The specific commands used for this engagement:

```
mkdir ~/bin

OCP4_BASEURL=https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest
LATEST_VERSION=$(curl -s ${OCP4_BASEURL}/release.txt | grep 'Version: ' | awk '{print $2}')
curl -s ${OCP4_BASEURL}/openshift-client-linux-$LATEST_VERSION.tar.gz | tar -xzf - -C ~/bin oc kubectl
curl -s ${OCP4_BASEURL}/openshift-install-linux-$LATEST_VERSION.tar.gz | tar -xzf - -C ~/bin/ openshift-install
```

5.2.4.3. Pull Secret

Pull secrets are specific each Red Hat account, so the customer was directed to log in to console.redhat.com and obtain a pull secret from the appropriate link at:

<https://console.redhat.com/openshift/install/vsphere/user-provisioned>.

For this engagement the pull secret was stored in the file `~/ocp4_pull_secret`

5.2.4.4. Creating the Install Configuration File

See: [Manually creating the installation configuration file](#)

Create install-config.yaml using the following contents: .install-config.yaml

```
apiVersion: v1
baseDomain: nonprod-ocp.vib
compute:
- name: worker
  replicas: 3
controlPlane:
  name: master
  replicas: 3
metadata:
  name: workload-01
networking:
  clusterNetwork:
    - cidr: 100.96.0.0/15
    hostPrefix: 22
  machineNetwork:
    - cidr: 10.50.136.0/24
  networkType: OVNKubernetes
  serviceNetwork:
    - 100.200.0.0/16
platform: {}
imageDigestSources:
- mirrors:
  - dr-ocp-mirror-registry-vm-01.north.vib.corp:8443/ocp4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - dr-ocp-mirror-registry-vm-01.north.vib.corp:8443/ocp4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
pullSecret: <redacted>
sshKey: <redacted>
```



The installer will automatically delete the **install-config.yaml** file when it is executed. It is strongly recommended that a backup be created prior to moving to the next step to avoid the need to re-create the file in the event it needs to be used again.

A backup of the installation file was created with the following command

```
cp install-config.yaml ../install-config.yaml.bkp
```

5.2.4.5. Create Manifests and Ignition Configuration

See: [Creating the Kubernetes manifest and Ignition config files](#)

The specific command used for this engagement (manifests are created in the current directory):

```
openshift-install create manifests
```

To ensure that masters are not schedulable the following command was used to modify the generated manifest files:

```
sed -i 's/mastersSchedulable: true/mastersSchedulable: false/g' manifests/cluster-scheduler-02-config.yml
```

The Ignition configuration was created using the following commands:

```
openshift-install create ignition-configs

cat <<EOF > append-bootstrap.ign
{
  "ignition": {
    "config": {
      "append": [
        {
          "source": "http://WEBSERVERIP:8080/ocp/ignition/bootstrap.ign",
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
EOF
```

5.2.4.6. Creating the Virtual Machines on vSphere

See: [Creating Red Hat Enterprise Linux CoreOS \(RHCOS\) Machines in vSphere](#)

The specific commands used for this engagement to copy **bootstrap.ign** to the webserver:

```
sudo mkdir -p /var/www/html/ocp/ignition/  
sudo cp bootstrap.ign /var/www/html/ocp/ignition/
```

This command confirms that webserver is hosting the bootstrap ignition file and that it is accessible:

```
curl http://WEBSERVERIP:8080/ocp/ignition/bootstrap.ign
```

This command was used to generate files in base64:

```
for i in append-bootstrap master worker  
do  
base64 -w0 < $i.ign > $i.64  
done
```

5.2.4.6.1. Import OVA with vSphere

Access the vCenter web UI



You will need to connect to vSphere using credentials with privileges to create/upload templates to the target datacenter.

Click the icon resembling a stack of paper to navigate to “VMs and Templates”.

From there right click on your datacenter and select "**New Folder → New VM and Template Folder**".

Name this new folder the name of your cluster id: workload-01

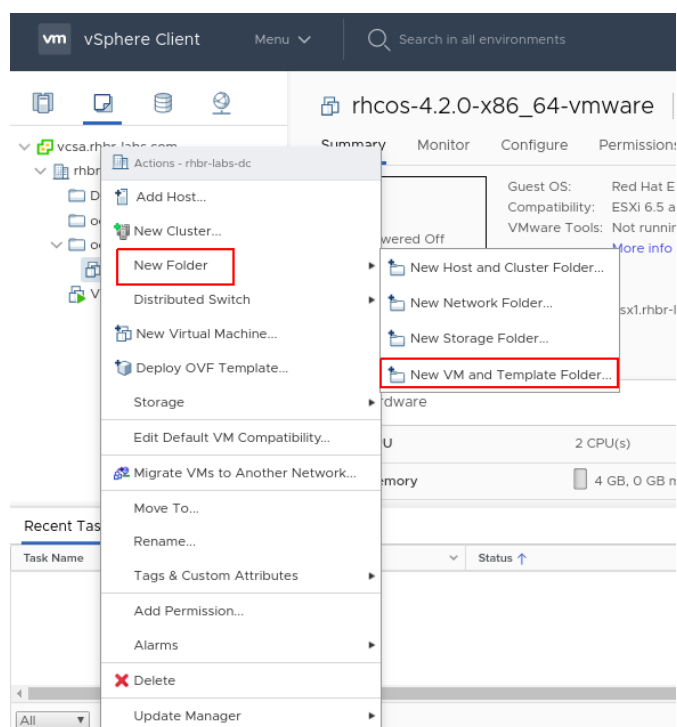


Figure 2. Creating Folder

Import the OVA by right clicking the folder and selecting "**Deploy OVF Template**".

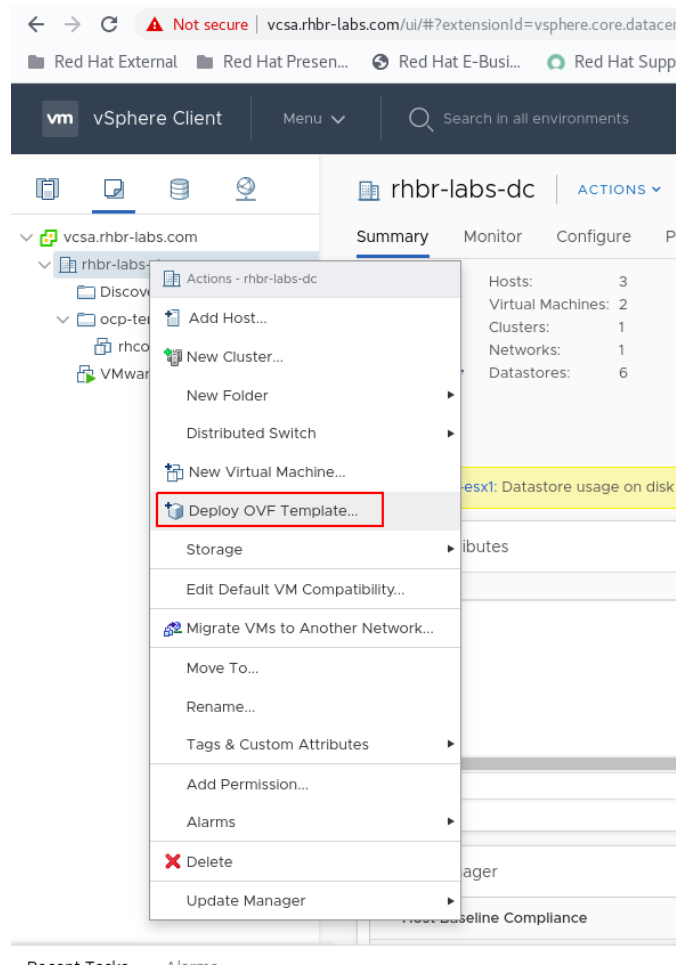


Figure 3. Deploying OVA

Add the url to RHCOS OVA ([see here](#)) and click "**NEXT**":

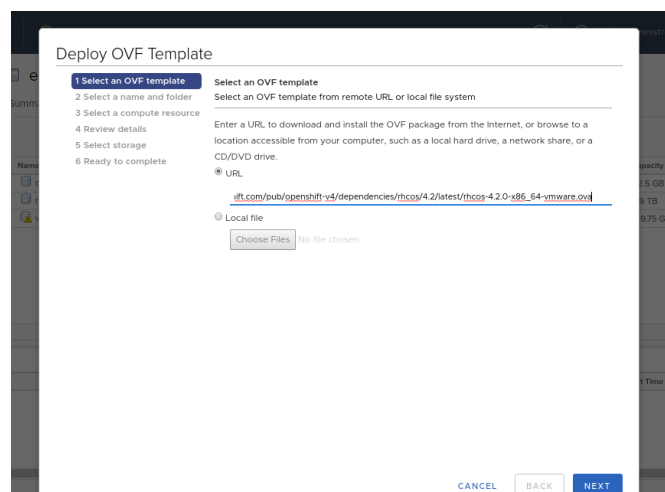


Figure 4. Deploying OVA

Select the folder you created in the previous step and click "**NEXT**":

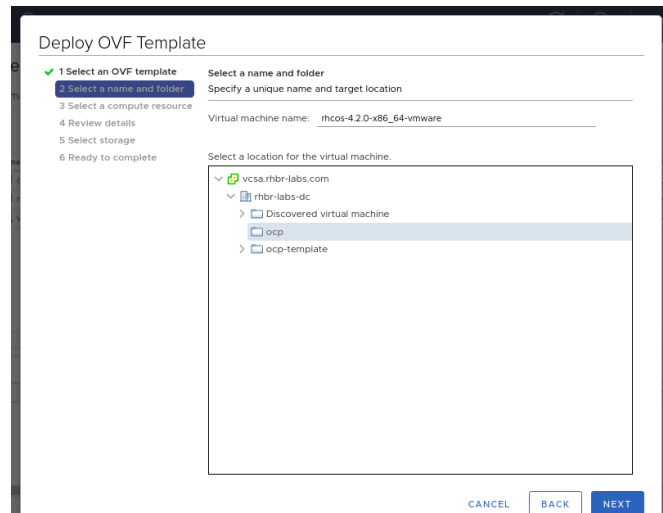


Figure 5. Deploying OVA

Select the compute resource and click "**NEXT**":

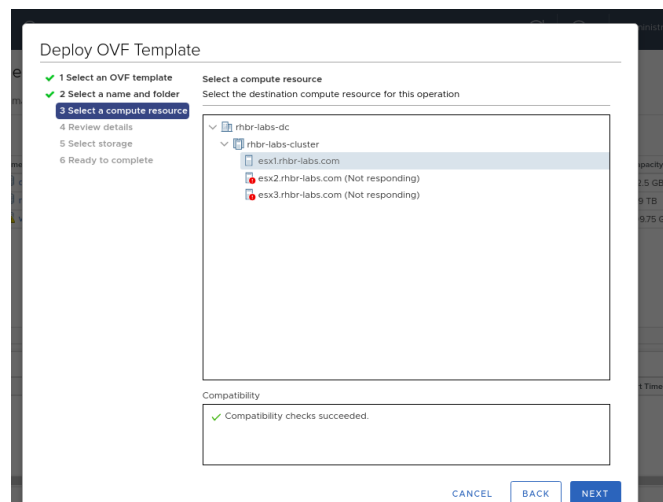


Figure 6. Deploying OVA

Select the datastore specified in the installation config file earlier:

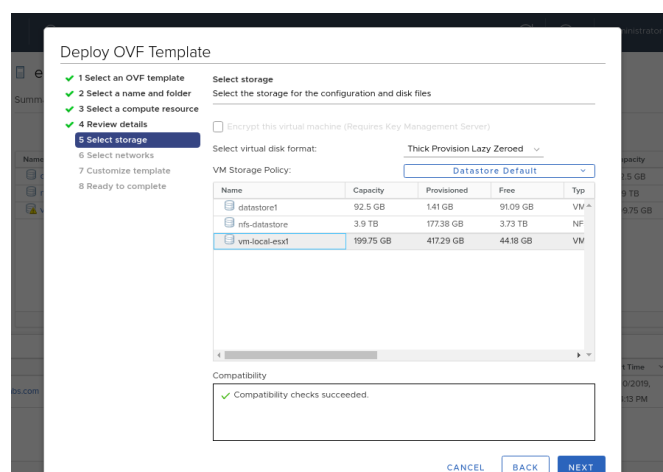


Figure 7. Deploying OVA

Select the network and click "**NEXT**":

Figure 8. Deploying OVA

Don't fill anything yet (these parameters will be filled further). Click "**NEXT**".

Figure 9. Deploying OVA

Click "**FINISH**" in the next screen

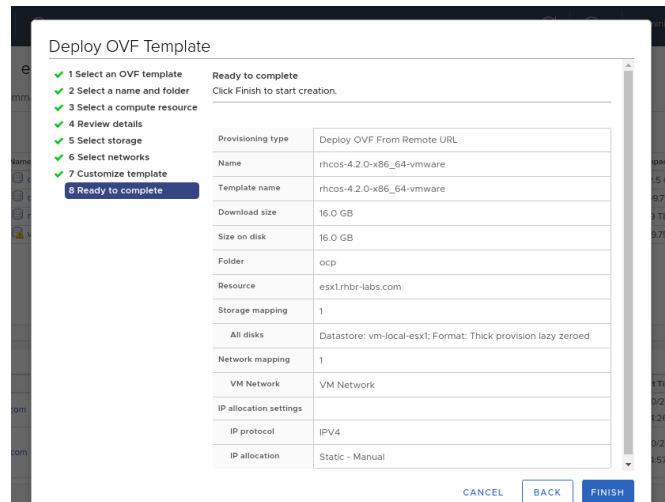


Figure 10. Deploying OVA



NEVER start up the template itself. The ignition process only runs on first boot, so booting the template would cause ignition files provided afterward to be ignored.

5.2.4.7. Provision OpenShift Servers

Right click on the OVA and select **Clone** → **Clone to Virtual Machine**

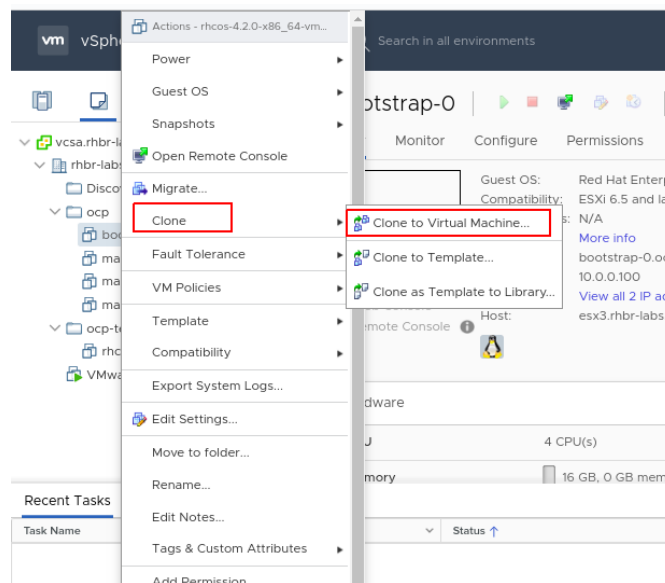


Figure 11. Clone to VM

Select the folder you created before, input the VM name and click **"NEXT"**.



The VM name must match the name configured in DHCP and DNS.

Folder: workload-01
VM Name: bootstrap-0

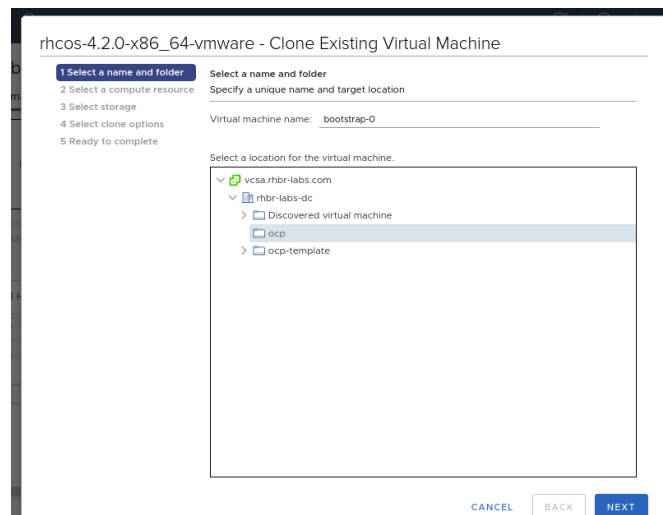


Figure 12. Clone to VM

Select the compute resource and click **"NEXT"**:

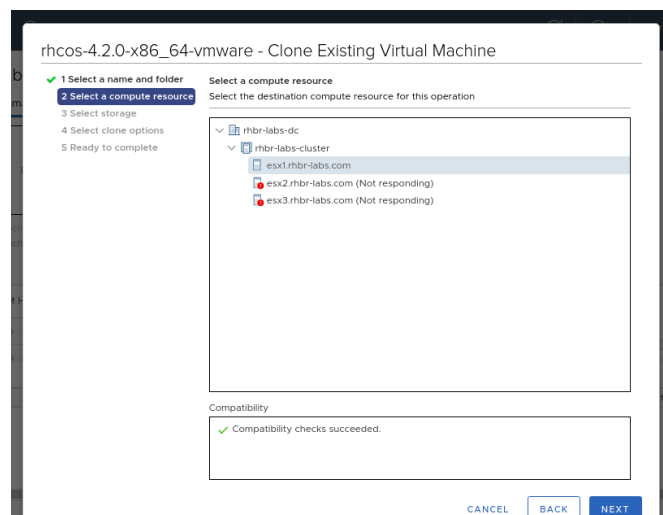


Figure 13. Clone to VM

Select the datastore and select disk format as **"Thin Provision"**:

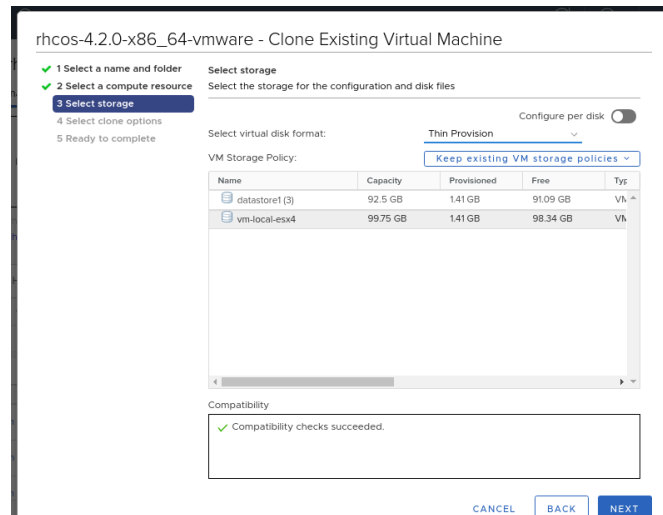


Figure 14. Clone to VM

Enable the option "**Customize this virtual machine's hardware**"

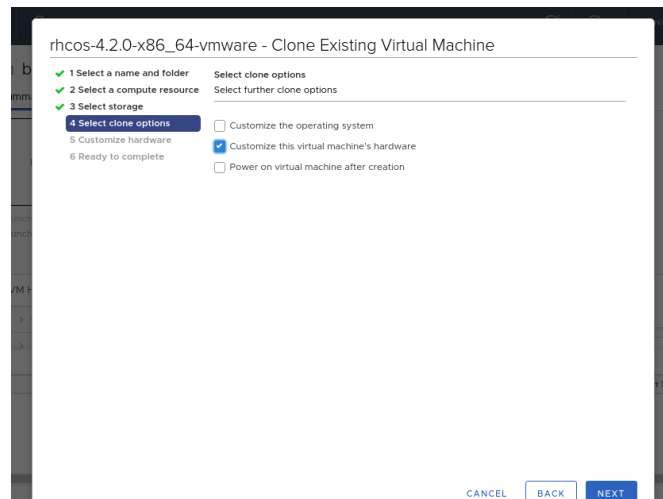


Figure 15. Clone to VM

In the next screen input the following parameters:

CPU: 4
 Memory: 16 GB
 - Enable "Reserve all guest memory" option
 Hard Disk: 120 GB
 Network Adapter 1:
 - MAC Address: Manual - <MAC ADDRESS RESERVED IN DHCP>

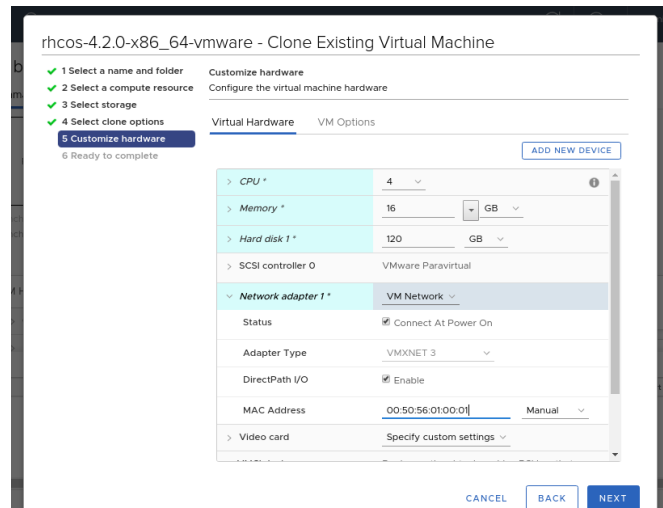


Figure 16. Clone to VM

Click in "**VM Options**" tab and expand "**Advanced**" accordion:

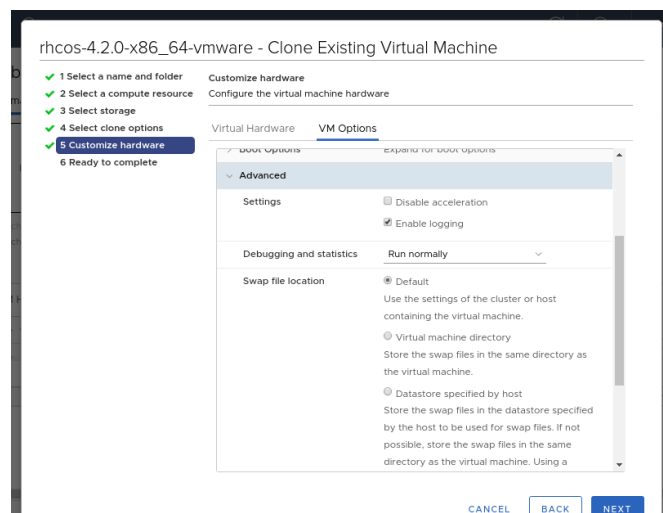


Figure 17. Clone to VM

In "**Latency Sensitivity**" select High and click in "**Edit Configuration...**" button.

Click in the "**ADD CONFIGURATION PARAMS**" button and add the following parameters:

```
guestinfo.ignition.config.data=<content of append_bootstrap.64 file>
guestinfo.ignition.config.data.encoding=base64
disk.EnableUUID=TRUE
```

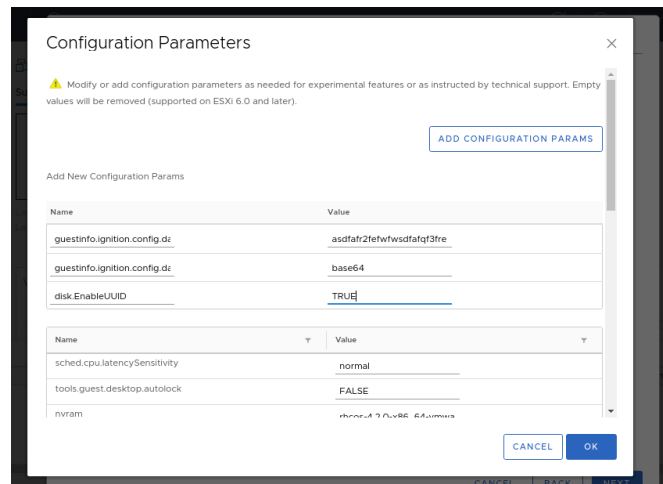
If using static IP addresses and if 4.16 >= 4.6 set the following parameter before booting the VM:

```
guestinfo.afterburn.initrd.network-kargs=ip=<ipcfg>
```

[See: Static IP configuration for vSphere using OVA](#)



If using static IPs the parameter `afterburn.initrd.network-kargs` only applies to the first boot.



Configuration Parameters

Modify or add configuration parameters as needed for experimental features or as instructed by technical support. Empty values will be removed (supported on ESXi 6.0 and later).

ADD CONFIGURATION PARAMS

Add New Configuration Params

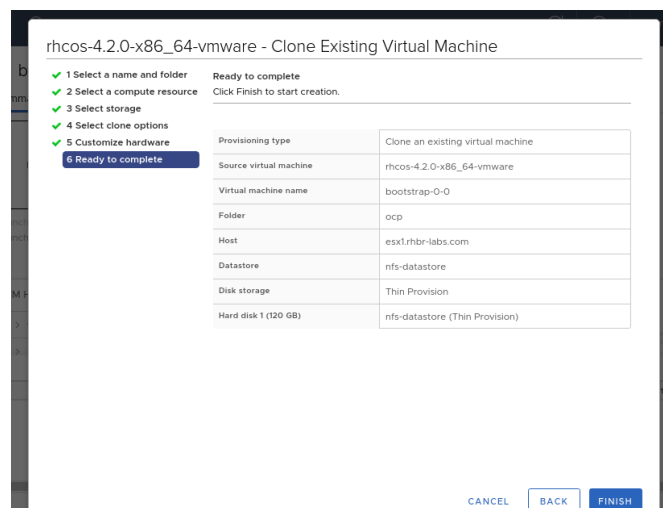
Name	Value
guestinfo.ignition.config.dir	asdfaf2fefwfsdafqf3fre
guestinfo.ignition.config.dir	base64
disk.EnableUUID	TRUE

Name	Value
sched.cpu.latencySensitivity	normal
tools.guest.desktop.autolock	FALSE
nvram	rhcos-4.2.0-x86_64-vmware

CANCEL OK

Figure 18. Clone to VM

Click "**NEXT**" then "**FINISH**" to create the bootstrap machine.



rhcos-4.2.0-x86_64-vmware - Clone Existing Virtual Machine

1 Select a name and folder
2 Select a compute resource
3 Select storage
4 Select clone options
5 Customize hardware
6 Ready to complete

Ready to complete
Click Finish to start creation.

Provisioning type	Clone an existing virtual machine
Source virtual machine	rhcos-4.2.0-x86_64-vmware
Virtual machine name	bootstrap-0-0
Folder	ocp
Host	esx1.rhbr-labs.com
Datastore	nfs-datastore
Disk storage	Thin Provision
Hard disk 1 (120 GB)	nfs-datastore (Thin Provision)

CANCEL BACK FINISH

Figure 19. Clone to VM

Repeat the process above to deploy each of the VMs

Once all of the VMs have been deployed, start them up.

5.2.4.8. The Installation Process

[See: Waiting for the bootstrap process to complete](#)

The following command was used to install OpenShift with the configuration file created earlier:

```
[user0@infra-services ocp]$ openshift-install wait-for bootstrap-complete --log-level debug
...
DEBUG Bootstrap status: complete
INFO It is now safe to remove the bootstrap resources
```

After "INFO" message about removing bootstrap resources is displayed, the bootstrap VM and its associated disk can be shut down and removed from vSphere.

The process can take up to 20 minutes. If this message is not displayed within that timeframe, see the troubleshooting tips here: [troubleshooting.adoc](#)!

After bootstrap completion, the following command was used to verify the installation:

```
openshift-install wait-for install-complete --log-level debug
```

The output of the command:

```
...
DEBUG Route found in openshift-console namespace: console
DEBUG Route found in openshift-console namespace: downloads
DEBUG OpenShift console route is created
INFO Install complete!
...
```

5.2.4.9. Running oc Commands

See: [Logging in to the cluster](#)

The following command will copy the Kubernetes configuration to the logged-in user's profile and allow the use of the "oc" command on the newly deployed cluster:

```
mkdir ~/.kube/
cp auth/kubeconfig ~/.kube/config
```

5.2.4.10. Cluster Operators Deployment

See: [Initial Operator configuration](#)

Many operators are deployed as part of the installation process.

The query below shows the operators deployed by the installation process for this engagement:

<replace this example with actual output from the customer's environment>

```
[user0@infra-services ocp]$ watch -n 10 'oc get clusteroperators'
Every 10.0s: oc get clusteroperators
infra-services.rhbr-labs.com: Mon Dec 16 20:43:44 2019
```

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication		Unknown	Unknown	True	3m1s
cloud-credential	4.16.14	True	False	False	6m54s
console	4.16.14	Unknown	True	False	11s
dns	4.16.14	True	False	False	6m20s
image-registry		False	False	True	11s
ingress	unknown	False	True	False	11s
insights	4.16.14	True	False	False	6m53s
kube-apiserver	4.16.14	True	False	False	4m24s
kube-controller-manager	4.16.14	True	False	False	4m18s
kube-scheduler	4.16.14	True	False	False	4m16s
machine-api	4.16.14	True	False	False	6m56s
machine-config	4.16.14	True	False	False	6m18s
marketplace		False	True	False	12s
monitoring		Unknown	True	Unknown	14s
network	4.16.14	True	False	False	5m57s
node-tuning	4.16.14	True	False	False	2m50s
openshift-apiserver	4.16.14	True	False	False	2m9s
openshift-controller-manager	4.16.14	True	False	False	3m7s
openshift-samples		False	False		9s
operator-lifecycle-manager	4.16.14	True	False	False	5m52s
operator-lifecycle-manager-catalog	4.16.14	True	False	False	5m52s
operator-lifecycle-manager-packageserver	4.16.14	True	False	False	3m7s
service-ca	4.16.14	True	False	False	6m46s
service-catalog-apiserver	4.16.14	True	False	False	2m57s
service-catalog-controller-manager	4.16.14	True	False	False	3m

5.2.5. Post Deployment Configurations

The point of this section is to describe post installation configuration done

5.2.5.1. Node Labeling

[See: Update label on nodes](#)

Create label for infrastructure node

```
for server in infra01 infra02; do oc label node $server node-role.kubernetes.io/infra=""
```

5.2.5.2. Configure Taints on Nodes

[See: Adding taint and tolerations using machineset](#)

Set taints for infrastructure node

```
oc adm taint nodes -l node-role.kubernetes.io/infra node-role.kubernetes.io/infra=reserved:NoSchedule
```

5.2.5.3. Updating CA Bundle

[See: Updating the CA bundle](#)

Vietnam International Bank has generated the certificate in pfx format. We need to extract the CA from the pfx file

```
openssl pkcs12 -in nonprod.pfx -cacerts -nokeys -chain -out vib-ca.pem
```

We create config map from the CA certificate

```
oc create configmap vib-ca --from-file=ca-bundle.crt=vib-ca.pem -n openshift-config
```

Update the proxy/cluster

```
oc patch proxy/cluster --type=merge --patch='{"spec":{"trustedCA":{"name":"vib-ca"}}}'
```

5.2.5.4. Configuring Openshift Ingress

5.2.5.4.1. Replacing Default Ingress Certificate

[See: Replacing the default ingress certificate](#)

The ingress certificate and the private key also need to be extracted from the pfx files.

Extract private key

```
openssl pkcs12 -in nonprod.pfx -nocerts -out nonprod.protected.key
# Enter passphrase
# Now we need to create another key without passphrase
openssl rsa -in nonprod.protected.key -out nonprod.key
# Enter passphrase again
```

Extract certificate from pfx file

```
openssl pkcs12 -in nonprod.pfx -clcerts -nokeys -out nonprod.crt
```

Create secret contains the certificate

```
oc create secret tls custom-ingress --cert=nonprod.crt --key=nonprod.key -n openshift-ingress
```

Update the ingress controller configuration

```
oc patch ingresscontroller.operator default --type=merge -p '{"spec":{"defaultCertificate": {"name": "custom-ingress"}}}'  
-n openshift-ingress-operator
```

5.2.5.4.2. Configure Ingress Tolerations

[See: Controlling pod placement using taints](#)

and

[See: add a nodePlacement and Toleration to default ingresscontroller](#)

To arrange ingress pods to be placed on router nodes the following actions are done

```
oc edit -n openshift-ingress-operator ingresscontroller/default
```

Add the following lines into spec section

```
nodePlacement:  
  nodeSelector:  
    matchLabels:  
      node-role.kubernetes.io/infra: ""  
  tolerations:  
  - effect: NoSchedule  
    key: node-role.kubernetes.io/infra  
    value: reserved
```

Save and exit vim

Delete pods in openshift-ingress namespace one by one to trigger pod rescheduling

5.2.5.5. Authentication

5.2.5.5.1. Configuring HTPasswd

[See: Configuring an htpasswd identity provider](#)

```
#Install httpd tools
yum install httpd-tools -y

#Create file called users.htpasswd with single username and password
htpasswd -c -B -b users.htpasswd onprem.ocp.adm <password>

#Create secret in the OpenShift
oc create secret generic htpass-secret --from-file=htpasswd=users.htpasswd -n openshift-config
```

Edit OpenShift OAuth

```
oc edit OAuth
```

and add the following entry

```
spec:
  identityProviders:
  - htpasswd:
      fileName:
        name: htpass-secret
      mappingMethod: claim
      name: local-user
      type: HTPasswd
```

Assign admin a cluster-admin roles

```
oc adm policy add-cluster-role-to-user cluster-admin admin
```

5.2.5.5.2. Configuring OpenID Authentication

[See: Configuring oidc identity provider](#)

Prior to create OpenID authentication, the following steps needs to be done on Entra ID: - Register a new application in Entra ID for authentication. - Configure the application registration in Entra ID to include optional and group claims in tokens.

Create secret object containing OpenID client secret

```
oc create secret generic openid-client-secret-shzrv --from-literal=clientSecret=<secret> -n openshift-config
```

Edit OpenShift OAuth

```
oc edit OAuth
```

and add the following entry

```
spec:
  identityProviders:
  - mappingMethod: claim
    name: VIB-SSO
    openID:
      claims:
        email:
        - email
        name:
        - name
        preferredUsername:
        - upn
        - email
      clientID: f2477598-e5dd-4421-a888-71684f262eb1
      clientSecret:
        name: openid-client-secret-shzrv
      extraScopes: []
      issuer: https://login.microsoftonline.com/662ce8bb-85db-4ba2-ba98-be222318b83e/v2.0
```

5.2.5.6. Configuring NTP and Timezone

5.2.5.6.1. Configure NTP

See: [Configuring chrony time service](#)

Download and install butane

```
#Download butane
curl https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane --output butane

#Make it executable
chmod +x butane
mv butane /usr/local/bin/butane
```

Create the following butane file

Listing 1. 99-worker-chrony.bu

```
variant: openshift
version: 4.12.0
```

```

metadata:
  name: 99-worker-chrony
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        server 10.50.90.93 iburst
        server 10.50.90.94 iburst
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtsync
        logdir /var/log/chrony

```

Use Butane to generate a MachineConfig object file

```
butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

Copy the worker manifest and change the role and name to master

```
cp 99-worker-chrony.yaml 99-master-chrony.yaml
```

Final end files

Listing 2. 99-worker-chrony.yaml

```

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 99-worker-chrony
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          compression: gzip
          source:
            data: ;base64,H4sIAAAAAAAC/2TLQQ6CQAwF0H1P0RPM1KALjgNDwcaRmb+VhNubGFyxfnmu2BXcSbpLGiQNPdv0gQdd5PaXGbbEYLU57yNytSmXB9p25J/Q
            a3yqh765S8I9IYofW6Ha1tlnraeh74BAAD//9aVfLuCAAAA
          mode: 420

```

```

overwrite: true
path: /etc/chrony.conf

```

Listing 3. 99-master-chrony.yaml

```

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-master-chrony
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          compression: gzip
          source:
            data: ;base64,H4sIAAAAAAAC/2TLQQ6CQAwF0H1P0RPM1KALjgNDwcaRMB+VhNubGFyxfnmu2BXcSbpLGiQNPdv0gQdd5PaXGbbEY1U57yNytSmXB9p25J/Q
            a3yqh765S8I9IYofW6Ha1tlnraeh74BAAD//9aVfluCAAAA
          mode: 420
          overwrite: true
          path: /etc/chrony.conf

```

Apply both machineconfig

```

oc create -f 99-worker-chrony.yaml
oc create -f 99-master-chrony.yaml

```

5.2.5.7. Configure Multus

Prior to configuring ODF, an additional NetworkAttachmentDefinition needs to be created

The following NetworkAttachmentDefinition was created in the cluster

```

apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ocs-cluster
  namespace: openshift-storage
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "macvlan",
    "master": "ens224",
    "mode": "bridge",
    "ipam": {

```

```

    "type": "whereabouts",
    "range": "100.201.0.024"
  }
}'

```

5.2.5.8. Storage Deployment

5.2.5.8.1. Local Storage Operator Deployment

Procedure

1. Log in to the OpenShift Web Console.
2. Click **Operators > OperatorHub**.
3. Type **local storage** in the **Filter by keyword** box to find the **Local Storage Operator** from the list of operators, and click on it.
4. Set the following options on the **Install Operator** page:
 1. Update channel as **stable**.
 2. Installation mode as **A specific namespace on the cluster**.
 3. Installed Namespace as **Operator recommended namespace openshift-local-storage**.
 4. Update approval as **Manual**.
5. Click **Install**.

Verification steps

- Verify that the Local Storage Operator shows a green tick indicating successful installation.

5.2.5.8.2. ODF Operator Deployment

Procedure

1. Log in to the OpenShift Web Console.
2. Click **Operators > OperatorHub**.
3. Scroll or type **OpenShift Data Foundation** into the **Filter by keyword** box to find the **OpenShift Data Foundation** Operator.
4. Click **Install**.
5. Set the following options on the **Install Operator** page:
 1. Update Channel as **stable-4.16**.
 2. Installation Mode as **A specific namespace on the cluster**.
 3. Installed Namespace as **Operator recommended namespace openshift-storage**.

4. Select Approval Strategy as **Manual**.
5. Ensure that the **Enable** option is selected for the **Console plugin**.
6. Click **Install**.

Verification steps

- After the operator is successfully installed, a pop-up with a message, **Web console update is available**, appears on the user interface. Click **Refresh web console** from this pop-up for the console changes to reflect.
- In the Web Console:
 - Navigate to Installed Operators and verify that the **OpenShift Data Foundation** Operator shows a green tick indicating successful installation.
 - Navigate to **Storage** and verify if the **Data Foundation** dashboard is available.

5.2.5.8.3. Configure ODF with Multus

Procedure

1. In the OpenShift Web Console, click **Operators > Installed Operators** to view all the installed operators. Ensure that the **Project** selected is **openshift-storage**.
2. Click on the **OpenShift Data Foundation** operator, and then click **Create StorageSystem**.
3. In the **Backing storage** page, perform the following:
 1. Select **Full Deployment** for the **Deployment type** option.
 2. Select the **Create a new StorageClass using the local storage devices** option.
 3. Select **Use Ceph RBD as the default StorageClass**.
4. In the **Create local volume set** page, provide the following information:
 1. Select **Full Deployment** for the **Deployment type** option.
 2. Select **Disks on selected nodes** and choose all the worker nodes
 3. Click **Next** and **Yes** to continue
5. In the Capacity and nodes page:
 1. In the **Configure Performance** section, choose **Balanced**
 2. Click **Next**
6. In the Security and network page:
 1. On the **Network** section, choose **Custom (Multus)**
 2. Select a **Cluster Network Interface** from the dropdown and choose the NAD created previously
 3. Leave the **Public Network Interface** blank and click **Next**
7. In the **Data Protection** page, click **Next**.

8. In the **Review and create** page, review the configuration details. To modify any configuration settings, click **Back** to go back to the previous configuration page.
9. Click **Create StorageSystem**.

Verification steps

- To verify the final Status of the installed storage cluster:
 1. In the OpenShift Web Console, navigate to **Installed Operators > OpenShift Data Foundation > Storage System**.
 2. Click **ocs-storagecluster-storagesystem > Resources**.
 3. Verify that the **Status** of the **StorageCluster** is **Ready** and has a green tick mark next to it.

5.2.5.9. Configuring OpenShift Monitoring Stack

[See: Configuring Monitoring Stack](#)

[See: Configuring persistent storage](#)

[See: Moving monitoring components to different nodes](#)

Set cluster monitoring configuration:

Listing 4. cluster-monitoring-configuration.yaml

```
apiVersion: v1
data:
  config.yaml: |
    enableUserWorkload: true
    prometheusOperator:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusK8s:
      retention: 30d
      volumeClaimTemplate:
        metadata:
          name: pvc-prometheus
        spec:
          storageClassName: ocs-storagecluster-ceph-rbd
          resources:
            requests:
              storage: 500Gi
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    alertmanagerMain:
      volumeClaimTemplate:
```

```

metadata:
  name: pvc-alertmanager
spec:
  resources:
    requests:
      storage: 10Gi
nodeSelector:
  node-role.kubernetes.io/infra: ""
kubeStateMetrics:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
telemetryClient:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
k8sPrometheusAdapter:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
openshiftStateMetrics:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
thanosQuerier:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
kind: ConfigMap

```

Create the "cluster-monitoring-configmap".

```
oc create -n openshift-monitoring configmap cluster-monitoring-config --from-file config.yaml=monitoring-storage.yaml
```

Check if the PVCs are created.

```
oc get pvc -n openshift-monitoring
```

Verify prometheus and alert manager is mounted using PVC

```

oc exec -n openshift-monitoring statefulset/prometheus-k8s -c prometheus -- df -h /prometheus
oc exec -n openshift-monitoring statefulset/alertmanager-main -c alertmanager -- df -h /alertmanager

```

5.2.5.10. Configuring OpenShift Logging Stack

[See: Installing OpenShift logging Configuring the logging collector](#)

To install Loki Operator:

- In the OpenShift Container Platform web console, click Operators → OperatorHub.
- Choose Loki Operator from the list of available Operators, and click Install.
- Ensure that the All namespaces on the cluster is selected under Installation Mode.
- You must specify the openshift-operators-redhat namespace. The openshift-operators namespace might contain Community Operators, which are untrusted and could publish a metric with the same name as an OpenShift Container Platform metric, which would cause conflicts.
- Select Enable operator recommended cluster monitoring on this namespace.
- This option sets the openshift.io/cluster-monitoring: "true" label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the openshift-operators-redhat namespace.
- Select stable as the Update Channel.
- Select an Approval Strategy. In this cluster, the "Automatic" option was chosen.
- Click Install.
- Verify that the Loki Operator installed by switching to the Operators → Installed Operators page.
- Ensure that Loki Operator is listed in all projects with a Status of Succeeded.

Create object bucket in noobaa to store logss

5.2.5.10.1. Create Object Bucket Claim

See: [How to install LokiStack On Prem using ODF](#)

Create object bucket claim which will be used to host the logging data

Listing 5. logging-bucket-odf.yaml

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  finalizers:
    - objectbucket.io/finalizer
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: logging-bucket-odf
  namespace: openshift-logging
spec:
  additionalConfig:
    bucketclass: noobaa-default-bucket-class
```

```
generateBucketName: loki-bucket-odf
objectBucketName: obc-openshift-logging-loki-bucket-odf
storageClassName: openshift-storage.noobaa.io
```

Get bucket properties and accessKey from the associated ConfigMap

```
BUCKET_HOST=$(oc get -n openshift-logging configmap loki-bucket-odf -o jsonpath='{.data.BUCKET_HOST}')
BUCKET_NAME=$(oc get -n openshift-logging configmap loki-bucket-odf -o jsonpath='{.data.BUCKET_NAME}')
BUCKET_PORT=$(oc get -n openshift-logging configmap loki-bucket-odf -o jsonpath='{.data.BUCKET_PORT}')
ACCESS_KEY_ID=$(oc get -n openshift-logging secret loki-bucket-odf -o jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 -d)
SECRET_ACCESS_KEY=$(oc get -n openshift-logging secret loki-bucket-odf -o jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 -d)
```

Build an object storage secret using variable from above

```
oc create -n openshift-logging secret generic lokistack-odf \
  --from-literal=access_key_id="${ACCESS_KEY_ID}" \
  --from-literal=access_key_secret="${SECRET_ACCESS_KEY}" \
  --from-literal=bucketnames="${BUCKET_NAME}" \
  --from-literal=endpoint="https://${BUCKET_HOST}:${BUCKET_PORT}"
```

Create the lokistack instance .lokistack-logging.yaml

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: lokistack-logging
  namespace: openshift-logging
spec:
  hashRing:
    type: memberlist
  limits:
    global:
      queries:
        queryTimeout: 3m
      retention:
        days: 7
  managementState: Managed
  size: 1x.extra-small
  storage:
    schemas:
      - effectiveDate: "2024-06-04"
        version: v13
    secret:
```

```

    name: lokistack-odf
    type: s3
  tls:
    caName: openshift-service-ca.crt
storageClassName: ocs-storagecluster-cephfs
template:
  compactor:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  distributor:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  gateway:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  indexGateway:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  ingester:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  querier:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  queryFrontend:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  ruler:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
tenants:
  mode: openshift-logging

```

To install Cluster Logging Operator:

- In the OpenShift Container Platform web console, click Operators → OperatorHub.
- Choose Cluster Logging from the list of available Operators, and click Install.
- Ensure that the A specific namespace on the cluster is selected under Installation Mode.
- Ensure that Operator recommended namespace is openshift-logging under Installed Namespace.
- Select Enable operator recommended cluster monitoring on this namespace.
- This option sets the openshift.io/cluster-monitoring: "true" label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the openshift-logging namespace.
- Select stable as the Update Channel.

- Select an Approval Strategy. In this cluster, the "Automatic" option was chosen.
- Click Install.
- Verify that the Cluster Logging Operator installed by switching to the Operators → Installed Operators page.
- Ensure that Cluster Logging and Elasticsearch is listed in the openshift-logging project with a Status of Succeeded.

Create ClusterLogging instance

Listing 6. cluster-logging.yaml

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  collection:
    type: vector
  logStore:
    lokistack:
      name: lokistack-logging
    retentionPolicy:
      application:
        maxAge: 2d
      audit:
        maxAge: 2d
      infra:
        maxAge: 2d
    type: lokistack
  managementState: Managed
  visualization:
    ocpConsole:
      logsLimit: 15
    type: ocp-console
```

```
oc create -f cluster-logging.yaml
```

Verify that the fluentd daemonset for all nodes has been created.

```
oc get pods -n openshift-logging -o wide
```

Verify that pvc is created

```
oc get pvc -n openshift-logging
```

No logging forward configured for the NON Prod environment:

5.2.5.11. Alertmanager configuration

[See: Configuring alert receivers](#)

SMTP configuration will be enabled for Alert Manager component in the monitoring stack to send e-mail alerts.

Extract the existing alertmanager-main secret from the openshift-monitoring namespace to the /tmp/ directory.

```
oc extract secret/alertmanager-main --to /tmp/ -n openshift-monitoring --confirm
```

Modify the /tmp/alertmanager.yaml file.

Listing 7. alertmanager.yaml

```
global:
  resolve_timeout: 5m
inhibit_rules:
  - equal:
      - namespace
      - alertname
    source_matchers:
      - severity = critical
    target_matchers:
      - severity =~ warning|info
  - equal:
      - namespace
      - alertname
    source_matchers:
      - severity = warning
    target_matchers:
      - severity = info
receivers:
  - name: Critical
    email_configs:
      - to: nonprod.ocp.alert@vib.com.vn
        from: ocp.nonprod.notify@vib.com.vn
```



```

smarthost: 'mail.vib.com.vn:25'
require_tls: false
auth_username: ocp.nonprod.notify@vib.com.vn
auth_password: <removed>
- name: Default
- name: Watchdog
route:
  group_by:
    - namespace
  group_interval: 5m
  group_wait: 30s
  receiver: Default
  repeat_interval: 2h
  routes:
    - matchers:
        - alertname = Watchdog
      receiver: Watchdog
    - receiver: Critical
      matchers:
        - severity = critical

```

Apply the configuration

```
$ oc set data secret/alertmanager-main --from-file /tmp/alertmanager -n openshift-monitoring
```

5.2.5.12. Internal image registry configuration

Configuring the registry for bare metal Image registry removed during installation

By default OpenShift Image registry operator bootstraps itself as 'Removed' in the absence of a object storage. This is done to allow the installation to complete without errors. After installation we change the management state from 'Removed' to 'Managed'. Follow the steps below to change it.

Create a PVC for image registry to use.:

- In the OpenShift Web Console, click Storage → Persistent Volume Claims
- Set the Project to openshift-image-registry.
- Click Create Persistent Volume Claim.
 - From the list of available storage classes retrieved above, Choose ntnx-files-dynamic StorageClass
 - Specify the Persistent Volume Claim Name, registry-storage-pvc.
 - Specify an Access Mode of Shared Access (RWX).
 - Specify a Size of 512 GB.

- Click Create. Wait until the status of the new Persistent Volume Claim is listed as Bound.

Edit Imageregistry

Edit registry configuration

```
oc edit configs.imageregistry.operator.openshift.io
```

Assign registry-storage-pvc as pvc for Image Registry

```
storage:
  pvc:
    claim: registry-storage-pvc
```

Change managementState Image Registry Operator configuration from Removed to Managed.

```
oc patch configs.imageregistry.operator.openshift.io cluster \
--type merge --patch '{"spec":{"managementState":"Managed"}}'
```

Edit image registry configuration to arrange pod scheduling and rollout strategy

```
oc edit configs.imageregistry.operator.openshift.io
```

```
spec:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
  tolerations:
  - effect: NoSchedule
    key: node-role.kubernetes.io/infra
    value: reserved
```

If necessary, enable image registry default route

```
oc patch configs.imageregistry.operator.openshift.io/cluster --type merge -p '{"spec":{"defaultRoute":true}}'
```

Verify the configuration was successful by running the below command.

```
oc get pods -n openshift-image-registry
```

```
oc get pvc -n openshift-image-registry
```

Check if a new pod starting with "image-registry-*" is created. Check if the PVC is in bound state.

5.2.5.12.1. Enable ETCD encryption

etcd encryption is enabled to encrypt sensitive resources in the cluster.



It is not recommended to take a backup of etcd until the initial encryption process is complete. If the encryption process has not completed, the backup might be only partially encrypted.

Modify the API server object:

```
oc edit apiserver
```

Set the encryption field type to aescbc:

```
spec:
  encryption:
    type: aescbc
```

The aescbc type means that AES-CBC with PKCS#7 padding and a 32 byte key is used to perform the encryption

Save the file to apply the changes.

The encryption process starts. It can take 20 minutes or longer for this process to complete, depending on the size of your cluster.

Verify that etcd encryption was successful.

Review the Encrypted status condition for the OpenShift API server to verify that its resources were successfully encrypted:

```
oc get openshiftapiserver -o=jsonpath='{range
.items[0].status.conditions[?(@.type=="Encrypted")]}{.reason}{"\n"}{.message}{"\n"}'
EncryptionCompleted
All resources encrypted: routes.route.openshift.io, oauthaccesstokens.oauth.openshift.io,
oauthauthorizetokens.oauth.openshift.io
```

Review the Encrypted status condition for the Kubernetes API server to verify that its resources were successfully encrypted:

```
oc get kubeapiserver -o=jsonpath='{range
.items[0].status.conditions[?(@.type=="Encrypted")]}{.reason}{"\n"}{.message}{"\n"}'
EncryptionCompleted
All resources encrypted: secrets, configmaps
```

If the output shows EncryptionInProgress, this means that encryption is still in progress. Wait a few minutes and try again.

5.2.5.13. Configuring ETCD backup script

On bastion node, create ETCD backup script on /backup/



Script will keep latest 5 version of ETCD backups

Listing 8. backup.sh

```
etcd_backup_path=/backup/etcd-backup
etcd_master_backup_script=/usr/local/bin/cluster-backup.sh

for master_node in master01 master02 master03
do
    ssh core@$master_node "sudo $etcd_master_backup_script ./assets/backup"
    ssh core@$master_node "sudo chmod 604 /var/home/core/assets/backup/snapshot_*.db"
    ssh core@$master_node "sudo chmod 604 /var/home/core/assets/backup/static_kubernetes_*.gz"
    scp core@$master_node:/var/home/core/assets/backup/snapshot_*.db $etcd_backup_path/$master_node/
    scp core@$master_node:/var/home/core/assets/backup/static_kubernetes_*.gz $etcd_backup_path/$master_node/
    ssh core@$master_node "sudo rm -rf ./assets"
    find $etcd_backup_path/$master_node/ -name "snapshot_*" -type f -daystart -mtime +7 -exec rm -f {} \;
    find $etcd_backup_path/$master_node/ -name "static_kubernetes*" -type f -daystart -mtime +7 -exec rm -f {} \;
done
```

Set ETCD backup crontab

```
#Backup OCP etcd
1 0 * * * /backup/scripts/backup.sh >> /backup/logs/etcd-backup.log
```

5.2.6. CI/CD Pipeline

5.2.6.1. CI/CD Pipeline Development for Sample Application using Azure DevOps

This documentation provides a step-by-step guide to set up a CI/CD pipeline for a sample Java Spring Boot application deployment to OpenShift using Azure DevOps Pipeline.

5.2.6.1.1. Prerequisites

1. **Azure DevOps Account:** Ensure you have an Azure DevOps account and an organization created.
2. **Repository:** Your application code must be hosted on Azure Repos Git repository. Access to the repository is required.
3. **Image Registry:** The application image built with the pipeline should be stored in an image registry (i.e. Quay). Access to the registry is required.
4. **Azure DevOps Agent:** Ensure that an Azure DevOps agent pool is configured and running.
5. **Target Environment (OpenShift):** Ensure the target environment is running and configured properly. Access to the target OpenShift clusters is required.
6. **Network Connectivity:** Verify that the agent has network access to Azure DevOps and any external services, such as your repository, artifact storage, or deployment targets (i.e. OpenShift clusters).
7. **Firewall Rules:** Update firewall rules or security groups to allow necessary traffic between the Azure DevOps agent and your infrastructure.

5.2.6.1.2. Repository Details

Table 8. Repository Details

No	Description	URL	Branch
1	Source Code	https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/backend-service-dummy	dev
2	Pipeline	https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/microservices-ci-templates/backend-service-dummy	main
3	Pipeline Variables	https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/variable-management/dev/ocp	main
4	Helm Chart	https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/aks-manifest-deploy/backend-service-dummy-helm-chart	main

No	Description	URL	Branch
5	ArgoCD Apps	https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/aks-manifest-deploy/argocd-apps	main
6	Quay Image Registry	https://registry-quay-quay.apps.dr-mgmt-01.ocp.vib/sharedrepo/backend-service-dummy	NA

5.2.6.1.3. CI/CD Pipeline Development

Before starting a CI pipeline development, perform the steps below to prepare the environment for the new pipeline.

1. Create a new Project in Azure DevOps (B01-IT-Container- Platform).
2. Create a new service connection for Azure Resource Manager, Quay, and OpenShift as deployment targets.
3. Set Up the Git repository for the **pipeline**, then clone the git repository into your local development machine.

```
> git clone https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/microservices-ci-templates/backend-service-dummy
```

4. Set Up the Git repository for the **pipeline variables**, then clone the git repository into your local development machine.

```
> git clone https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/variable-management/dev/ocp
```

5. Set Up the Git repository for the application **helm chart**, then clone the git repository into your local development machine.

```
> git clone https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/aks-manifest-deploy/backend-service-dummy-helm-chart
```



The sample pipeline reuses existing templates in this git repository: "https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/microservices-ci-templates/" branch "nonprod". However, there are minor changes introduced to make it work with the new pipeline and OpenShift target clusters. The copied templates and changes are stored in the main branch in the same repository, while the original remains

in the nonprod branch. More details [here](#).

5.2.6.1.3.1. Creating Continuous Integration Pipeline

1. Go to the **pipeline** directory in your local workstation. Create a new pipeline yaml file.

azure-pipeline-dev-onprem.yaml

```
pool: $(selfAgentVM)

variables:

# shared variables for all applications
- template: dev/ocp/var-shared-service-config-microservices-onpremise.yml@variables

# specific variables for dummy application
- template: dev/ocp/var-backend-service-dummy.yml@variables

parameters:

# image tag parameter for build
- name: imageTag
  default: "none"
  type: string

resources:
  repositories:

# application source code repo
- repository: backend-service-dummy
  type: git
  name: B01-IT-Container-Platform/_git/backend-service-dummy
  ref: refs/heads/main

# pipeline variables repo
- repository: variables
  type: git
  name: B01-IT-Container-Platform/_git/variable-management
  ref: refs/heads/main

# pipeline templates repo
- repository: variables
  type: git
  name: B01-IT-Container-Platform/_git/microservices-ci-templates
  ref: refs/heads/main
```

```
stages:
- template: templates/main-template.yml@ci-templates
  parameters:
    ENVIRONMENT: ${ variables['env.ENVIRONMENT'] }
    cloudProvider: ${ variables.cloudProvider }
    projectName: ${ variables.projectName }
    projectType: ${ variables.cloudProvider }
    programmingLang: ${ variables.programmingLang }
    imageTag: ${ variables.imageTag }
    deployOCPOnPrem: ${ variables.deployOCPOnPrem }
    helmBranchName: ${ variables.helmBranchName }
    useArgoRollout: ${ variables.useArgoRollout }
```

2. Push the pipeline yaml file to the pipeline git repository.

```
> git commit -am "Create new pipeline" && git push
```

3. Go to the **pipeline variables** directory in your local workstation. Create a new variable yaml file.

var-backend-service-dummy.yaml

```
variables:
- name: serviceName
  value: 'backend-service-dummy'
- name: app.name
  value: 'backend-service-dummy'
- name: subPath
  value: 'backend-service-dummy'
- name: helmBranchName
  value: 'backend-service-dummy'
- name: REPO_NAME
  value: 'backend-service-dummy'
- name: app.port
  value: '8080'
- name: app.deployment_replicas
  value: '1'
- name: projectType
  value: 'java'

# add more variables here
# - name: <variable name>
#   value: <variable value>
```

4. Push the pipeline variables yaml file to the pipeline variables git repository.


```
> git commit -am "Create new pipeline variables" && git push
```

5. Go to the **helm chart** directory in your local workstation. Create a new helm chart yaml files.

Chart.yaml

```
apiVersion: v2
name: backend-service-dummy
description: A Helm chart for backend-service-dummy
type: application

# This is the chart version
version: 1.0.0

# This is the version number of the application being deployed
appVersion: "1.0.0"
```

values.yaml

```
deployment:
  image_repo: registry-quay-quay.apps.dr-mgmt-01.ocp.vib/sharedrepo/backend-service-dummy
  image_tag: xxxxx
  imagePullSecrets:
    - name: quay-registry-secret
service:
  name: backend-service-dummy
  namespace: cicd-sample-dev
  ports:
    - port: 8080
  healthcheck: /actuator/health
  healthcheck_port: 8080
  type: clusterIP
serviceAccount:
  name: backend-service-dummy-sa
route:
  expose: true
  targetPort: 8080
argocd:
  project: default
```

6. Push the helm chart yaml file to the helm chart git repository.

```
> git commit -am "Create new helm chart" && git push
```

5.2.6.1.3.2. Run and Test the Pipeline

1. Navigate to **Pipelines > New Pipeline**.
2. Select the repository and choose **Existing Azure Pipelines YAML file**.
3. Choose the **azure-pipelines.yml** file created earlier.
4. Save and run the pipeline.
5. Monitor the logs. Look for any error messages, then fix the pipeline accordingly.
6. A successful pipeline should look like this:

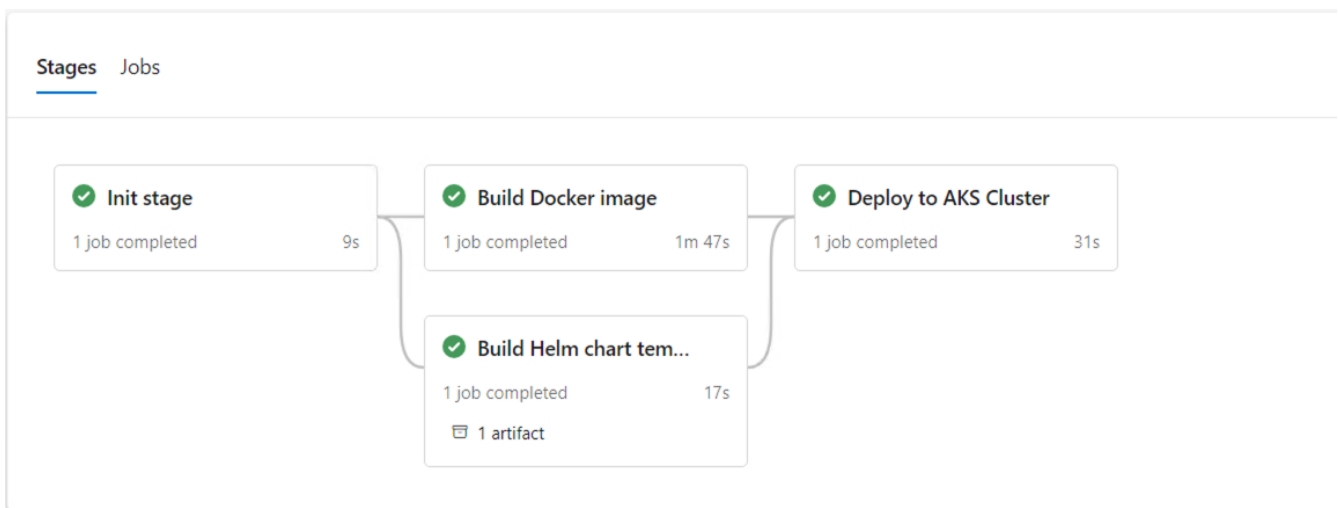


Figure 20. Azure Pipeline

5.2.6.1.3.3. Set Up Continuous Deployment with ArgoCD

Before creating a CD pipeline, perform the steps below to prepare the environment.

1. Deploy the OpenShift GitOps Operator. You need a cluster admin user to perform this.
2. Open the OpenShift web console and go to **Operators → OperatorHub**.
3. Search for "OpenShift GitOps", click the **Red Hat OpenShift GitOps** tile, and then click **Install**.
4. On the **Install Operator** page:
 - Select an **Update channel**.
 - Select a GitOps **Version** to install.
 - Choose an **Installed Namespace**. The default installation namespace is "openshift-gitops-operator". Click **Install** to make the GitOps Operator available on the OpenShift Container Platform cluster.
5. Verify that the Red Hat OpenShift GitOps Operator is listed in **Operators → Installed Operators**.

The **Status** should resolve to **Succeeded**.

6. After the OpenShift GitOps Operator is installed, a new link should appear under the custom link menu. Click **Cluster Argo CD** to open the OpenShift GitOps console.
7. Go back to the OpenShift console and open the **Secrets** menu. Open the **argocd-secret** secret and copy the **admin.password** value, it is the password of the **admin** user for the OpenShift GitOps.

5.2.6.1.3.4. Creating ArgoCD Application

1. Go to the application **helm chart** root directory of your local development machine, then create a new folder (e.g. "argocd-apps") to store the ArgoCD applications helm charts.
2. Create a new helm chart.

Chart.yaml

```
apiVersion: v2
name: app-of-apps
description: A Helm chart for ArgoCD applications
type: application

# This is the chart version
version: 1.0.0

# This is the version number of the application being deployed
appVersion: "1.0.0"
```

3. Create a new folder called **templates** and a new ArgoCD application yaml file for the sample application in this folder.

application-deployment.yaml

```
{{- range $apps := .Values.apps.deployments }}
---
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: {{ $apps.name }}-application
spec:
  destination:
    namespace: {{ $apps.namespace }}
    server: https://kubernetes.default.svc
  project: default
  source:
    directory:
      recurse: true
    path: helm-chart/helm-deployment
```

```
repoURL: https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/aks-manifest-deploy
targetRevision: main
helm:
  {{- with $apps.valueFiles }}
  valueFiles: {{- toYaml . | nindent 8 }}
  {{- end }}
{{- end }}
```

4. Create a **values.yaml** file for the ArgoCD application.

```
apps:
  deployments:
    - name: backend-service-dummy-dev
      namespace: cicd-sample-dev
      valueFiles:
        - ../../backend-service-dummy-helm-chart/values-dev.yaml
```

5. Commit and push the new files into the git repository.

```
> git commit -am "Create new ArgoCD applications helm chart" && git push
```

6. Go to the ArgoCD admin console and login using the OpenShift GitOps admin user.

7. In the Argo CD dashboard, click **NEW APP** to add a new Argo CD application.

8. Create an application with the following configurations:

Application Name: app-of-apps

Project: default

Sync Policy: Manual (or Automatic for auto-sync)

Repository URL: https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/aks-manifest-deploy

Revision: main

Path: helm-chart/helm-argocd-apps

Destination: <https://kubernetes.default.svc/>

Namespace: openshift-gitops

9. Click **CREATE** to create your application.

10. The sample application should be created and shown under the **Applications** menu in the ArgoCD web console.
11. Go to the sample application page (**backend-service-dummy-application**), then click **SYNC** to begin the application deployment to OpenShift. Monitor the status, if it is shown as **Synced**, then the deployment is finished successfully.

5.2.6.1.3.5. Creating ArgoCD Rollout for Blue/Green Deployment

1. Go to the application **helm chart/argocd-apps/templates** directory of your local development machine.
2. Create a new ArgoCD application yaml file for the sample application rollout in this folder.

application-rollout.yaml

```
{{- range $apps := .Values.apps.rollouts }}
---
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: {{ $apps.name }}-application
spec:
  destination:
    namespace: {{ $apps.namespace }}
    server: https://kubernetes.default.svc
  project: default
  source:
    directory:
      recurse: true
    path: helm-chart/helm-rollout
    repoURL: https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/aks-manifest-deploy
    targetRevision: main
  helm:
    {{- with $apps.valueFiles }}
    valueFiles: {{- toYaml . | nindent 8 }}
    {{- end }}
  {{- end }}
```

3. Edit the **values.yaml** file for the sample application rollout.

```
apps:
  rollouts:
  - name: backend-service-dummy-dev-rollout
    namespace: cicd-sample-dev
    valueFiles:
    - ../../backend-service-dummy-helm-chart/values-dev-rollout.yaml
```

4. Go to the sample application helm chart directory (`../../backend-service-dummy-helm-chart`). Create a **values.yaml** file for the sample application rollout.

```
rollout:
  image_repo: registry-quay-quay.apps.dr-mgmt-01.ocp.vib/sharedrepo/backend-service-dummy
  image_tag: xxxxx
  imagePullSecrets:
    - name: quay-registry-secret
service:
  name: backend-service-dummy
  namespace: cicd-sample-dev
  ports:
    - port: 8080
  healthcheck: /actuator/health
  healthcheck_port: 8080
  type: clusterIP
serviceAccount:
  name: backend-service-dummy-sa
route:
  expose: true
  targetPort: 8080
  activeWeight: 100
  previewWeight: 0
argocd:
  project: default
```

5. Commit and push the new files into the git repository.

```
> git commit -am "Add new ArgoCD application rollout" && git push
```

6. Go to the ArgoCD admin console, and trigger the synchronization of the **app-of-apps** application.
7. Open the newly created sample rollout application (`backend-service-dummy-dev-rollout`) and monitor the status.
8. Start the Rollout deployment by clicking on the **SYNC** button. Wait until the status is updated to **Synced**.
9. The Rollout will create a new ReplicaSet every time a new change is introduced in the git repository. To test this, make a change to the helm chart values of the sample rollout application (`backend-service-dummy-helm-chart/values-dev-rollout.yaml`). For example, change the image tag variable.
10. Commit and push the changes to the git repository.
11. Go back to the ArgoCD web console, then trigger the rollout application sync.
12. A new replicaSet will be created with a new pod. Monitor the progress of the pod creation. Once finished, the new pod will be associated with the **preview** service of the Rollout, while the old pod is

still associated with the **active** service.

13. You can test the new version by accessing the preview service directly from inside the OpenShift cluster, or through the route associated with the preview service.
14. If the **activeWeight** and **previewWeight** variables are set in the rollout values.yaml in the helm chart, a single route will be created. It connects to both the active and preview services with the respective weights. To test the new version, flip the weight values, i.e. **previewWeight: 100** and **activeWeight: 0**.
15. Once you are done with the testing, you can promote the new version permanently by clicking the 3-dots icon next to the rollout, then select **Promote-Full**. The new pod will be pointed to the **active** service and the old pod will be deleted.
16. If you want to rollback to the previous versions, click on the **HISTORY AND ROLLBACK** button, then select the version that you want to rollback to, then click on the 3-dots icon then click **Rollback**. The old version will be redeployed back to the OpenShift cluster.

5.2.6.1.3.6. Existing CICD Pipeline Templates Customizations

Repository Details

Table 9. Repository Details

Description	URL	Branch
Original Pipeline Templates	https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/microservices-ci-templates/templates	nonprod
Customized Pipeline Templates	https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/microservices-ci-templates/templates	main
Original Helm Templates	https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/aks-manifest-deploy/helm-chart	nonprod
Customized Helm Templates	https://dev.azure.com/vib-lz-devops/B01-IT-Container-Platform/_git/aks-manifest-deploy/helm-chart	main

Pipeline Templates

Below are the list of changes done to the pipeline templates:

1. Stage Build Image

templates/stages/stage-build-image.yml

```
stages:
- stage: BuildImage
  jobs:
  - job: BuildImage
    steps:

    # Add new step: login to Quay registry
    - ${{ if parameters.deployOCPOnPrem }}:
      - template: ../steps/step-login-quay-ocp.yml
        parameters:
          ${{ each parameter in parameters }}:
            ${{ parameter.Key }}: ${{ parameter.Value }}
    - ${{ else }}:
      .....
```

2. Stage Helm Chart

templates/stages/stage-helm-chart.yml

```
stages:
- stage: BuildTemplate
  .....

  jobs:
  - job: BuildTemplate
    .....

    steps:
    .....

    - bash: |
      .....

      # Add new parameter: helmBranchName to parameterize the helm branch
      if [ -n "${{ parameters.helmBranchName }}" ]; then
        git clone -b ${{ parameters.helmBranchName }} https://$ADO_TOKEN@dev.azure.com/vib-lz-devops/$(projectName)/_git/aks-manifest-deploy
      else
        .....

        # Remove argocd rollout steps
        # if [[ -n "${{ parameters.useArgoRollout }}" ]]; then
        #
        #
        # fi
        .....
```


3. Stage Deploy

templates/stages/stage-deploy.yml

```
stages:
- stage: Deploy
  .....

# Add dependency for OCP deployment
${{ elseif parameters.deployOCPOnPrem }}:
  dependsOn:
  - BuildTemplate
  - BuildImage
```

4. Step Login to Quay Registry

templates/steps/step-login-quay-ocp.yml

```
steps:
- bash: |
    docker login $(quayUrl) -u $(quayUsername) -p $(quayPassword)
  displayName: Login to shared repo
```

5. Step Helm Upgrade

templates/steps/step-helm-upgrade.yml

```
steps:
# Add OCP namespace checking
- ${{ if parameters.deployOCPOnPrem }}:
  - bash: |
      if oc get namespace "$(aksNamespace)" > /dev/null 2>&1; then
        echo "Namespace $(aksNamespace) exists."
      else
        echo "Namespace $(aksNamespace) does not exist. Creating it."
        if oc create namespace "$(aksNamespace)"; then
          echo "Namespace $(aksNamespace) created successfully."
        else
          echo "Failed to create namespace $(aksNamespace)."

```

```
# Add ArgoCD sync trigger to deploy the application on OCP
- ${{ if parameters.deployOCPOnPrem }}:
  - bash: |
      ADMIN_PASSWD=$(oc get secret openshift-gitops-cluster -n openshift-gitops -o jsonpath='{.data.admin\.password}' |
base64 -d)
      SERVER_URL=$(oc get routes openshift-gitops-server -n openshift-gitops -o jsonpath='{.status.ingress[0].host}')
      argocd login --username admin --password ${ADMIN_PASSWD} ${SERVER_URL}
      argocd app sync $(app.name)
      displayName: Deploy with ArgoCD
.....
```

6. Step Push Image to Quay Registry

templates/steps/step-push-image-to-registry.yml

```
steps:
.....

# Add step to push to Quay
- ${{ if eq(parameters.cloudProvider, 'ocp') }}:
  - bash: |
      docker login -u $(quayUsername) -p $(quayPassword) $(getRegistryName.REGISTRY_NAME)
      docker push $(app.REPOSITORY_URI_SERVICE):latest
      docker push $(app.REPOSITORY_URI_SERVICE):$(IMAGE_TAG)
      displayName: Push Image OCP
.....
```

Helm Chart Templates

Below are the list of changes done to the helm chart templates:

1. ConfigMap Template

helm-chart/helm-deployment/configmap.yaml

```
{{- if .Values.configs }}
kind: ConfigMap
apiVersion: v1
metadata:
  name: {{ .Values.service.name }}-config
  namespace: {{ .Values.service.namespace }}
{{- with .Values.configs }}
data: {{- toYaml . | nindent 2 }}
{{- end }}
```

```
{{- end }}
```

2. Route Template

helm-chart/helm-deployment/route.yaml

```
{{- if .Values.route.expose -}}
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    meta.helm.sh/release-name: {{ .Values.service.name }}
    meta.helm.sh/release-namespace: {{ .Values.service.namespace }}
  name: {{ .Values.service.name }}
  namespace: {{ .Values.service.namespace }}
  labels:
    app: {{ .Values.service.name }}
    app.kubernetes.io/managed-by: Helm
spec:
  port:
    targetPort: {{ .Values.route.targetPort }}
  to:
    kind: Service
    name: {{ .Values.service.name }}
  tls:
    insecureEdgeTerminationPolicy: Redirect
    termination: edge
{{- end }}
```

3. Route Rollout Template

helm-chart/helm-rollout/route.yaml

```
{{- if .Values.route.expose -}}
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    meta.helm.sh/release-name: {{ .Values.service.name }}
    meta.helm.sh/release-namespace: {{ .Values.service.namespace }}
  name: {{ .Values.service.name }}
  namespace: {{ .Values.service.namespace }}
```

```
labels:
  app: {{ .Values.service.name }}
  app.kubernetes.io/managed-by: Helm
spec:
  port:
    targetPort: {{ .Values.route.targetPort }}
  to:
    kind: Service
    name: {{ .Values.service.name }}-active-svc
    weight: {{ .Values.route.activeWeight }}
  alternateBackends:
    - kind: Service
      name: {{ .Values.service.name }}-preview-svc
      weight: {{ .Values.route.previewWeight }}
  tls:
    insecureEdgeTerminationPolicy: Redirect
    termination: edge
{{- end }}
```

5.3. Challenges, Resolutions, and Recommendations

5.3.1. Failed ODF Installation

5.3.1.1. Challenge

The initial deployment of OpenShift Data Foundation failed with `slow_mon` shows up in the installation log

5.3.1.2. Immediate Resolution

VIB has increased the CPU count in worker. After increasing the CPUs, ODF installed successfully

5.3.1.3. Future Recommendation

We recommend to put the storage node into a different server separated from workload, for example a `infra-odf`. This can help in segregating workload and also save the subscriptions used.

Chapter 6. Recommendations

6.1. Technical Next Steps

6.1.1. MachineConfigPool separation on Worker and Infra node

6.1.1.1. Indication

During the test upgrade on Non Production environment the node upgrade for non control plane nodes running in serial, causing the upgrade process take more times

6.1.1.2. Recommendation

We recommends segregating infra node on a different MachineConfigPool. Any changes on nodes level can run independently between infra and worker nodes

6.1.2. Setup VM Latency Sensitivity to High on Control Plane

6.1.2.1. Indication

We observe the CPU_WAIT time is high on Control Plane nodes VM, this means this VM needs more time to get physical CPU resources which can impact etcd performance

6.1.2.2. Recommendation

We recommends to setup Control Plane VM with high latency sensitivity so it can get the CPU resources it needed.

6.1.3. Increase numbers and CPU capacity on infra node

6.1.3.1. Indication

Currently the infra node host the workload for router, monitoring, and logging. Logging requires high CPU counts as per documentation: https://docs.openshift.com/container-platform/4.16/observability/logging/log_storage/installing-log-storage.html.

6.1.3.2. Recommendation

We recommends to have a 3 infra node with minimum 16 vCPU each. When one infra node is powered off, logging, router, and monitoring will still have total usable 32 vCPUs.

Appendix A: Glossary

Table 10. General Terminology

Term	Definition
BZ	Bugzilla. This context refers to the specific instance of Bugzilla operated by Red Hat to track bugs, RFEs, and release information. Also, it may refer to a specific numbered Bugzilla bug when given with a definite article or number.
Case	Support Case, RH Case, A Red Hat support case, requested and prioritized by the client, then responded to directly by GSS staff. This case number is distinct from a BZ bug number (most RH support cases do not have a corresponding BZ and vice-versa).
DNS	Domain Name Service
Downstream	Refers toward “productized” deployable, supported artifact (signed binary) end of open source software development process and related contracted errata, event-based support artifacts, product engineering, and product-specific documentation.
FQDN	Fully Qualified Domain Name
GSS (a.k.a. CEE)	Red Hat Global Support Services (“support”) aka “Customer Experience and Engagement”. Able to resolve bugs, configuration problems, and shepherd RFEs.
KB	Knowledge Base article, a short and specific how-to guide available in the customer portal.
RFE	Request for Enhancement. These are requests from clients or RH engineering for new product functionality or behavior. Generally tracked internally as a BZ.
RHC	Red Hat Consulting. Able to provide on-site and remote engineering and analysis for solution creation.
RHEL	Red Hat Enterprise Linux
RPM	RPM Package Manager (a recursive acronym)
TAM	Technical Account Manager. A site-designated support resource manager from GSS that maintains site-specific knowledge, awareness, and drives resolution.
Upstream	Refers toward source-code origin end of open source software development process and related open source community collaboration, documentation, and testing.

Appendix B: Red Hat Subscriptions Overview

Three factors play an important part in selecting the right RHEL subscription:

- The (Service Level Agreement) SLA
- The number of (physical) Sockets and,
- The number of Virtual Machines.

Red Hat Enterprise Linux can have layered products, like clustering. These require a separate subscription.

B.1. Service Level Agreement

An important part of a subscription is access to Red Hat support. The SLA associated with a subscription plays an important part for the customer here as this defines response times.

	Self-support ¹	Standard	Premium	
Hours of coverage	N/A	Standard business hours	Standard business hours (24x7 for Severity 1 and 2) ²	
Support channel	None	Web and phone	Web and phone	
Number of cases	N/A	Unlimited	Unlimited	
Response times	Initial and ongoing response	Initial and ongoing response	Initial response	Ongoing response
Severity 1	N/A	1 business hour	1 hour	1 hour or as agreed
Severity 2	N/A	4 business hours	2 hours	4 hours or as agreed
Severity 3	N/A	1 business day	4 business hours	8 business hours or as agreed
Severity 4	N/A	2 business day	8 business hours	2 business days or as agreed

Figure 21. Service level agreement response times

B.2. Socket Pairs

A conventional RHEL subscription is based upon socket-pairs. A socket meant here is the *physical* CPU socket.

Any subset of one or two sockets counts for one pair, so a node with 16 cores and two sockets requires one subscription to be compliant. A node with four sockets requires two single socket-pair subscriptions to be compliant.

A node with one socket will then require one subscription to be compliant, and a node with three sockets will require two subscriptions.

Of course, exceptions rule, so a node with four physical sockets of which only two have a seated CPU, will require just one subscription to be compliant.

B.3. Virtualization

A conventional RHEL subscription can alternatively be used to subscribe RHEL virtual machines. The number of Virtual machines included with a subscription is two.

One subscription which would apply to a *physical* dual socket machine can also be used for two Virtual Machines. The number of virtual sockets or vCPUs assigned to the two Virtual Machines being subscribed by a conventional RHEL subscription are not relevant.

B.3.1. Virtual Datacenter Subscriptions

Virtual Datacenter (VDC) Subscriptions differ from conventional RHEL subscriptions in that they subscribe VMs per hypervisor rather than individually.

VDC subscriptions use a parent-child relationship to validate a VM's subscription: the parent (the hypervisor) consumes a subscription while the children (the VMs running on that particular hypervisor) "piggy back" on that subscription. Any supported RHEL VMs running on a hypervisor with a valid VDC subscription have a valid subscription.

Similar to conventional subscriptions, VDC subscriptions are applied to hypervisors per socket-pair. See [Socket Pairs](#).



A VDC subscription is only valid for the virtual machines running on a hypervisor, not for the hypervisor itself (the hypervisor must have a separate subscription for its own operating system).

B.4. Layered products

Layered products require the same SLA and number of socket-pairs as the RHEL OS these products are deployed upon.

B.5. Red Hat Enterprise Linux life cycle

Red Hat Enterprise Linux Life Cycle

Description	Full Support	Maintenance Support 1 (RHEL 7) ¹²	Maintenance Support 2 (RHEL 7) ¹²	Extended Life Phase ⁷	Extended Life Cycle Support (ELS) Add-On ⁸	Extended Update Support (EUS) Add-On ⁸ Enhanced Extended Update Support (Enhanced EUS) Add-On ⁸
Access to Previously Released Content through the Red Hat Customer Portal	Yes	Yes	Yes	Yes	Yes	Yes
Self-help through the Red Hat Customer Portal	Yes	Yes	Yes	Yes	Yes	Yes
Technical Support ¹	Unlimited	Unlimited	Unlimited	Limited ⁹	Unlimited	Unlimited
Asynchronous Security Errata (RHSA) ^{10 11}	Yes	Yes	Yes	No	Yes ⁸	Yes ⁸
Asynchronous Bug Fix Errata (RHBA) ^{2 11}	Yes	Yes	Yes	No	Yes	Yes
Minor Releases	Yes	Yes	Yes (RHEL 7); Not applicable (RHEL 8 & 9)	No	No	No
Refreshed Hardware Enablement ³	Native	Limited ⁴ Native	Using Virtualization	Using Virtualization	Using Virtualization	Using Virtualization
Software Enhancements ⁵	Yes ⁶	No	No	No	No	No
Updated Installation Images	Yes	Yes	Yes ¹⁴	No	No	No

Figure 22. life cycle phases

The Life-cycle has different phases with different types of support. Depending on the applications, the deployment of nodes could be aligned with the different phases; deploy in Full Support, maintain in Maintenance Support 1 and replace in Maintenance Support 2 would be an ideal model.

Life-cycle Dates

All future dates mentioned for "End of Full Support" and "End of Maintenance Support 1" are close approximations, non definitive, and subject to change.

Red Hat Enterprise Linux Life-cycle Dates

Version	General availability	Full support ends	Maintenance Support 1 ends	Maintenance Support or Maintenance Support 2 ends	Extended life cycle support (ELS) add-on ends	Extended life phase ends	Final minor release
Full Support							
9	May 18, 2022	May 31, 2027	Not Applicable	May 31, 2032	May 31, 2035	Ongoing	9.10
8	May 7, 2019	May 31, 2024	Not Applicable	May 31, 2029	May 31, 2032	Ongoing	8.10
Maintenance Support							
7	June 10, 2014	August 6, 2019	August 6, 2020	June 30, 2024	June 30, 2028	Ongoing	7.9
End of maintenance							
7 (System z (Structure A))	April 10, 2018	August 6, 2019	August 6, 2020	May 31, 2021	Not Applicable	Ongoing	7.6
7 (ARM)	November 13, 2017	August 6, 2019	August 6, 2020	November 30, 2020	Not Applicable	Ongoing	7.6
7 (POWER9)	November 13, 2017	August 6, 2019	August 6, 2020	May 31, 2021	Not Applicable	Ongoing	7.6
6	November 10, 2010	May 10, 2016	May 10, 2017	November 30, 2020	June 30, 2024	Ongoing	6.10
5	March 15, 2007	January 8, 2013	January 31, 2014	March 31, 2017	November 30, 2020	Ongoing	5.11

All future dates mentioned are close approximations, non definitive, and subject to change.

Figure 23. Life-cycle dates

The RHEL lifecycle can be found here:

<https://access.redhat.com/support/policy/updates/errata>

Appendix C: Engaging Red Hat Support and Customer Service

Red Hat Support helps you get the most from your Red Hat subscriptions. This section includes information about the Red Hat Customer Portal, Red Hat phone support, and how to contact sales and customer service.

Red Hat Support is always available to assist you. If at any time you are unable to reach your personal Red Hat Support contacts, you can find [escalation contact information](#) below for the global support leadership team.

In order to assist you in the most efficient way, Red Hat provides some recommendations in the [Reference Guide for engaging with Red Hat Support](#). Some [tips](#) are also included below.

Opening a support case online can make it easier to share technical data, error messages, and system information with your Red Hat Support representative. Following the online submission with a phone call can reduce response time as well as potential errors in the capture of information. Red Hat recommends that you follow any [Severity 1 and 2](#) online support case submissions with a phone call to your local support center.

C.1. Red Hat Customer Portal

The [Red Hat Customer Portal](#) is an award-winning digital platform that delivers enterprise product knowledge, subscription resources, and technical expertise. Use the Customer Portal to access our Knowledgebase of technical expertise, collaborate with peers and Red Hat experts, create, track, and manage your case activity, and much more.

Use of the Customer Portal requires a login/password (linked to your Red Hat account). If you have any questions about your login and/or password, please contact your account representative, or Technical Account Manager/Customer Success Manager (if you have one assigned).

C.2. Red Hat phone support

Red Hat technical support engineers are available through phone support to quickly troubleshoot and resolve problems. Further information about regional contacts and business hours can be found at the [Technical Support contact information page](#).

Table 11. Red Hat Production Support phone contacts

Region	Contact details	Availability
North America	+888-GO-REDHAT (+888-467-3342)	24x7x365 (Premium support) Depending on severity, as the first point of contact for Red Hat support.
Europe, Middle East, Africa	+800-GO-REDHAT (00800 4673 3428)	

C.3. Escalation contacts

If you feel a support case requires extra attention, Red Hat provides [Escalation contacts](#).

C.4. Customer Service

In addition to Red Hat Support, Red Hat sales and [Customer Service](#) are available to assist you. If you have questions regarding sales, future product needs, or entitlements, please contact your regular account team, or use the contact information provided below.

Table 12. Customer Service contacts

Region	Telephone	E-mail
North America	888-467-3342	customerservice@redhat.com
Germany	0800 1828065	customerservice-emea@redhat.com
France (Hors Dom-Tom)	0800 907101	
France (Dom-Tom)	+353 212303445	
UK	0800 032 9515	
Italy	800 979 269	
Spain	900 811 831	
Portugal	+34 900 811 831	
All other countries	+353 21 2303 445	

C.5. Production support terms of service

The table below summarises Red Hat's [Production Support terms of service](#).

	Self-support	Standard	Premium
Hours of coverage	N/A	Standard business hours	Standard business hours (24x7 for Severity 1 and 2)
Support channel	None	Web and phone	Web and phone

	Self-support	Standard	Premium	
Number of cases	N/A	Unlimited	Unlimited	
Response times	Initial and ongoing response	Initial and ongoing response	Initial response	Ongoing response
Severity 1	N/A	1 business hour	1 hour	1 hour or as agreed
Severity 2	N/A	4 business hours	2 hours	4 hours or as agreed
Severity 3	N/A	1 business day	4 business hours	8 business hours or as agreed
Severity 4	N/A	2 business day	8 business hours	2 business days or as agreed

C.6. Red Hat Support severity level definitions

Red Hat Support uses the following [severity level definitions](#) to classify issues:

C.6.1. Severity 1 (urgent)

A problem that severely impacts your use of the software in a production environment (such as loss of production data or in which your production systems are not functioning). The situation halts your business operations and no procedural workaround exists.

C.6.2. Severity 2 (high)

A problem where the software is functioning but your use in a production environment is severely reduced. The situation is causing a high impact to portions of your business operations and no procedural workaround exists.

C.6.3. Severity 3 (medium)

A problem that involves partial, non-critical loss of use of the software in a production environment or development environment. For production environments, there is a medium-to-low impact on your business, but your business continues to function, including by using a procedural workaround. For development environments, where the situation is causing your project to no longer continue or migrate into production.

C.6.4. Severity 4 (low)

A general usage question, reporting of a documentation error, or recommendation for a future product enhancement or modification. For production environments, there is low-to-no impact on your business or the performance or functionality of your system. For development environments, there is a medium-to-low impact on your business, but your business continues to function, including by using a procedural workaround.

C.7. Tips for creating a good support case

- Fill out the account number and your name. It is recommended to use individual accounts rather than global/generic accounts.
- Select the product that is affected.
- Specify the version of the product.
- The problem statement should be as precise as possible, to help identify if there are any Knowledgebase articles that may help with resolution. Please read any suggested Knowledgebase articles before opening a case.
- Use the name of the component that you are having issues with if possible (for example "Ceph", "Nova" etc.) Keywords are very important.
- Explain your issue with as much detail as possible, and add the commands that you are using with outputs.
- Problem statements should be descriptive; for example: "[Component][Region] Issue description". Examples might include:
[Nova][NL Region] Nova fails on one compute node
[Ceph][Singapore Region] Ceph without OSD

OPEN A SUPPORT CASE

Account

[★ Find My Account Number](#)

Owner

Product

Product Version

Problem Statement

What problem/issue/behavior are you having trouble with? What do you expect to see?

Based on your description, here are some possible solutions

How can I test the impact CRUSH map tunable modifications will have on my pg distribution across OSD... [↗](#)

that the osdmap has the newly edited `crushmap` embedded in it. The `pg` placement can be tested with:
`~~~ # osdmaptool osdmap--test-map-pgs`
`editedmap_output- This command` can also be used with option `[--pool poolid]` `~~~~~test-map-pgs [--pool poolid]`- Will print out the mappings from placement groups to OSDs [↗](#)

Changing pg_num on a pool that uses a custom CRUSH ruleset, doesn't change the total PGs in the clus... [↗](#)

~~~ # `ceph` osd dump | grep new\_pool pool 9 'new\_pool' replicated size 3 min\_size 2 crush\_ruleset 50 object\_hash rjenkins pg\_num 64 pgp\_num 64 last\_change 1574 flags hashpspool stripe\_width 0 ~~~  
 4.increase the PG and PGP number on the new pool ~~~  
 # `ceph` osd pool set pg\_num 128 set pool 9 pg\_num to 128 [↗](#)

#### Marking OSD Out caused pgs to become active+remapped [↗](#)

the crush tunables optimal may also help. ~~~ `ceph` osd crush reweight osd. 0 ~~~ ~~~ `ceph` osd crush tunables optimal ~~~ Marking OSD Out caused `pgs` to become

Figure 24. Portal: case

- It is important to specify the environment that it is affected (for example pre-production or production), and which region is affected (e.g. London, New York)
- Keep in mind the [Red Hat Support severity level definitions](#) and assign appropriately
- [Attach sosreports](#) of the servers/components affected, to speed up case resolution
- Select level of support; add Premium any time that you have the option.



**When does the behavior occur? Frequently? Repeatedly? At certain times?**

**What information can you provide around timeframes and urgency?**

**Get faster results.** Attaching logs or other diagnostics files typically results in faster resolution.

sosreports.rar (14.4 MB)

×

SOSreports of the servers affected

Attach Another File

**Support Level** ?

Figure 25. Portal: case details

- Select severity level
- Add notifications to an individual or distribution list.
- If available, select a case group to notify other members, for example "Cloud" to alert all members of the "Cloud" group from your team so that they can work on the case too.

Severity

| Severity Level                                      | Standard         | Premium          |
|-----------------------------------------------------|------------------|------------------|
| <input checked="" type="radio"/> 4 Severity 4 (Low) | 2 business days  | 8 business hours |
| <input type="radio"/> 3 Severity 3 (Normal)         | 1 business day   | 4 business hours |
| <input type="radio"/> 2 Severity 2 (High)           | 4 business hours | 2 hours          |
| <input type="radio"/> 1 Severity 1 (Urgent)         | 1 business hour  | 1 hour           |

For more details about initial and ongoing response times, visit the [Production Support Service Level Agreement](#)

Send Email Notifications to

Select a User

Case Group (Optional)

Cloud

Submit

Cancel

Figure 26. Portal: set severity and add notification