Red Hat OpenShift Container Platform 4.13 on G42 Cloud

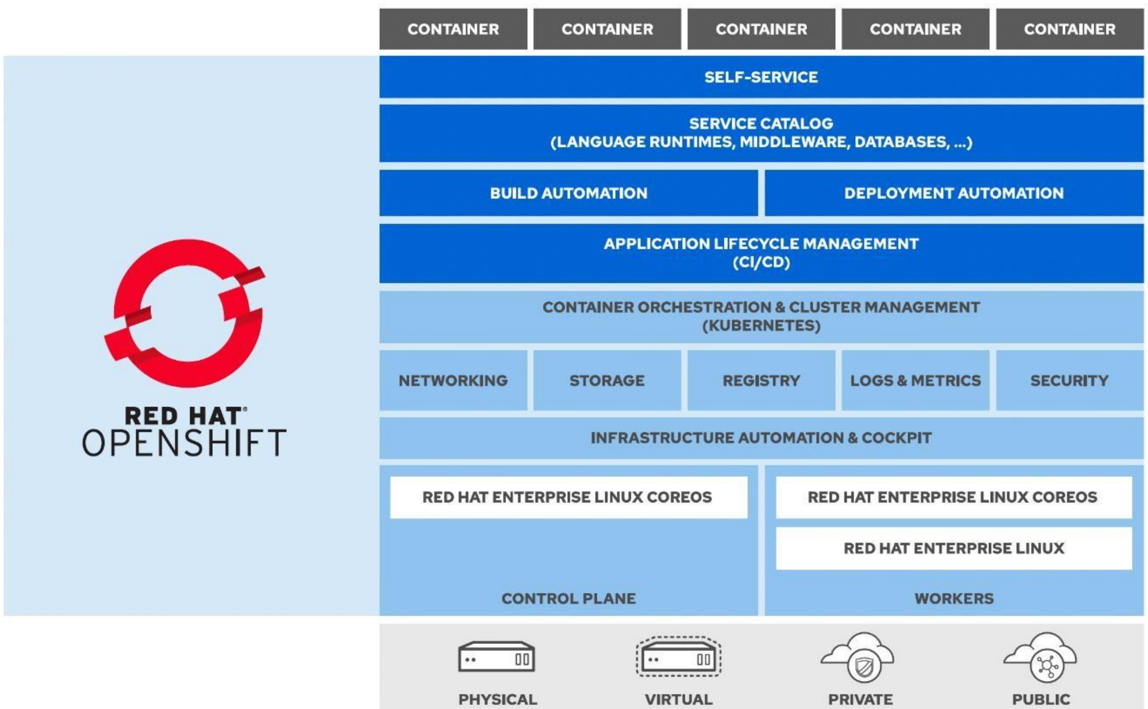
Deployment Guide

Date 15/05/2024

Table of Contents

1. Introduction

Red Hat OpenShift Container Platform is based on the Kubernetes open-source project and extends the platform with features that bring a robust, flexible, and scalable container platform to customer datacentres, enabling developers to run their workload in a high availability environment. It is designed to help developers and operations personnel to easily build, deploy, and manage applications, allowing the supported applications to expand from a small number of machines to thousands that can serve millions of clients. Red Hat OpenShift Container Platform provides a powerful and flexible platform to better manage application life cycles.



The Red Hat OpenShift product family integrates many components:

- The Red Hat Enterprise Linux CoreOS container-optimized, immutable operating system.
- The CRI-O engine, a small footprint, Open Container Initiative (OCI)-compliant container runtime engine with a reduced attack surface.
- Kubernetes, an open-source container orchestration platform.
- A few preinstalled application services, such as an internal container image registry and monitoring framework.
- A self-service web-console with comprehensive and modern interface.
- Certified container images for multiple programming language runtimes, databases, and other software packages.

This guide describes all steps that are required to manually deploy a cluster of Red Hat OpenShift Container Platform in G42 Cloud environment.

2. Introduction of G42 Cloud Resources

The following G42 Cloud services are used in a typical deployment of Red Hat OpenShift in G42 Cloud.

Table 2-1 - List of G42 Cloud Resources

Service	Description
VPC	Virtual Private Cloud (VPC) enables you to create private, isolated virtual networks. You can configure IP address ranges, subnets, and security groups, assign Elastic IP (EIP) addresses, and allocate bandwidth in a VPC. https://docs.vb.g42cloud.com/vpc/index.html
ECS	Elastic Cloud Server (ECS) provides secure, reliable, and scalable on-demand computing resources that enable you to deploy different workloads efficiently an flexibly. https://docs.vb.g42cloud.com/ecs/index.html
EVS	Elastic Volume Service (EVS) provides persistent block storage for services such as Elastic Cloud Server (ECS) and Bare Metal Server (BMS). With advanced data redundancy and cache acceleration capabilities, EVS offers high availability and durability with an extremely low latency. https://docs.vb.g42cloud.com/evs/index.html
IMS	Image Management Service (IMS) allows you to easily create and manage images. You can create a system disk image or data disk image from a disk or an external image file. You can also create a full-ECS image from an ECS with data disks or a backup of an ECS. https://docs.vb.g42cloud.com/ims/index.html
OBS	Object Storage Service (OBS) is a stable, secure, efficient, and easy-to-use cloud storage service that is scalable and compatible, allowing storage of any amount of unstructured data in any format. It also provides REST APIs, enabling internet access from anywhere. https://docs.vb.g42cloud.com/obs/index.html
NAT Gateway	NAT Gateway provides Source Network Address Translation (SNAT) and Destination Network Address Translation (DNAT) functions for Elastic Cloud Servers (ECSs) in a Virtual Private Cloud (VPC), making it easier for you to configure the ingress and egress for a VPC. https://docs.g42cloud.com/en-us/nat/index.html
ELB	Elastic Load Balance (ELB) automatically distributes incoming traffic across multiple backend servers based on a set of rules that you configure. ELB expands the service capabilities of applications and improves their availability by eliminating single points of failure (SPOFs). https://docs.vb.g42cloud.com/elb/index.html
DNS	Domain Name Service (DNS) provides highly available and scalable authoritative DNS resolution services and domain name management services for private domain. It translates domain names or application resources into IP addresses required for network connection. https://docs.g42cloud.com/en-us/dns/index.html

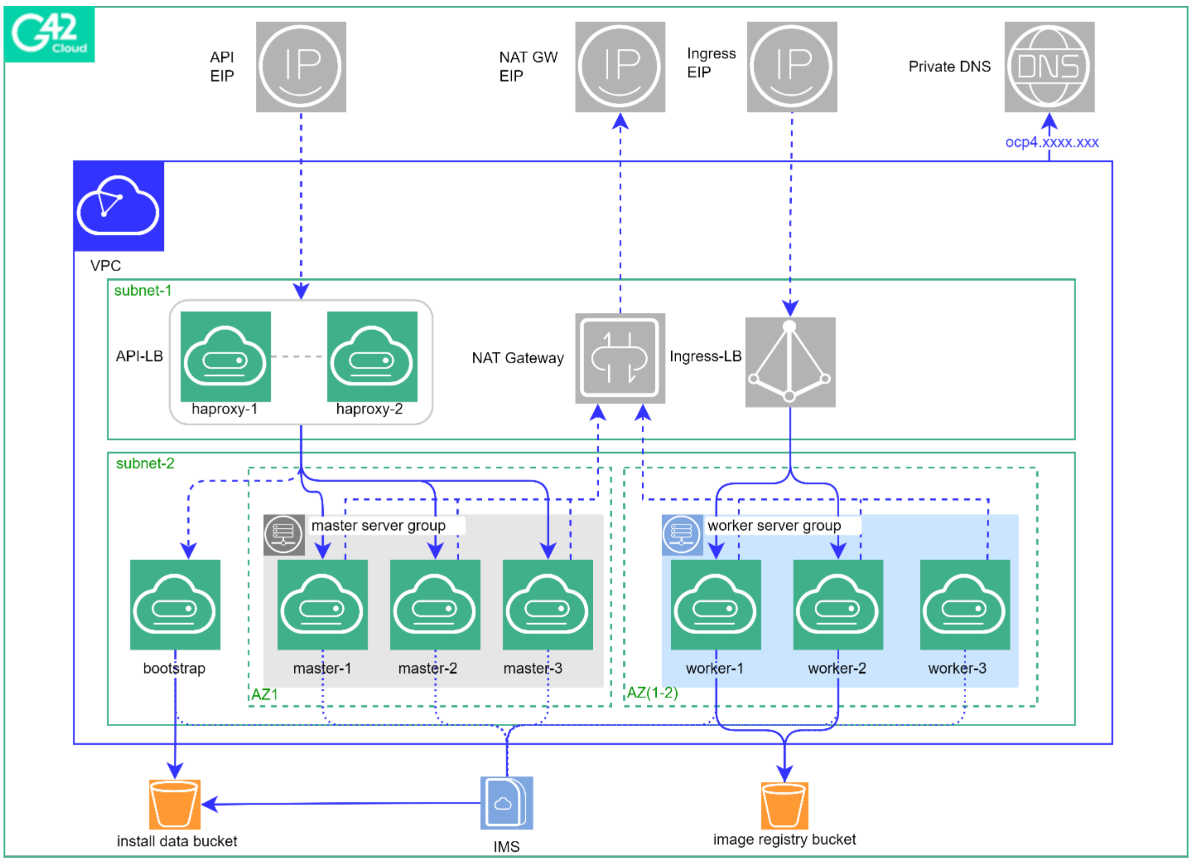
3. Architecture and Planning

This document will describe a deployment of a sized typical highly available Red Hat OpenShift cluster requires 3 control-plane nodes and at least 3 compute nodes of RHOCP

3.1. Architecture Diagram

Error! Reference source not found. represents a typical deployment architecture of OpenShift on G42 Cloud.

Figure 3-1 Deployment Architecture



3.2. Billing and Resource Consideration

3.2.1. Billing Considerations

The table below provides an overview of resources and billing considerations for resources.

Table 3-1 - Resource Requirement Charging Consideration

Charged by	Type	Remarks	Billing Mode
G42 Cloud	ECS	This resource is mandatory and will be used in every deployment.	Charged based on Hourly, Monthly or Annual Mode based on charging mode, flavor and size of ECS
	EVS	This resource is mandatory and will be used in every deployment.	Charged monthly based on EVS disks type and size
	NAT Gateway	This is required for providing the Internet access to the cluster	Charged based on NAT Gateway flavor and associate charges of EIP associated with NAT Gateway
	ELB	This resource is mandatory to distribute traffic between control plan and compute nodes	This service is free of Charge
	DNS	Private DNS service to setup a locally resolvable domain	This service is free of Charge
	EIP	This resource is Optional. One EIP can be used for access via the Internet	Charged based on no EIPs per month and data out transfer or bandwidth
	Direct Connect	This resource is Optional. Size of bandwidth depends on the use case and scenario	Charged based on carrier charges (including cross connect) and G42 Cloud port charges
Red Hat	OpenShift Subscription	OpenShift Cluster Subscription	Bring your own subscription to G42 Cloud

3.2.2. ECS and EVS sizing

A typical highly-available Red Hat OpenShift cluster requires 3 control-plane nodes and at least 3 compute nodes of RHOCP. These control and workers will be spawned in a single availability zone. The environment will consist of the standard and recommended machine layout.

Table 3-2 - ECS and EVS Sizing

Name	vCPU	RAM (GB)	ECS Flavor	System Disk	Disk Type	Max./Assured Bandwidth (Gbit/s)	Qty
Bootstrap Machine	8	64	d3.2xlarge.8	200	Ultra-High IO	5/5	1
Control Plane Machine	8	64	d3.2xlarge.8	200	Ultra-High IO	5/5	3
Compute Machines	8	64	d3.2xlarge.8	200	Ultra-High IO	5/5	3 <= worker
HA Proxy*	4	8	c6s.4xlarge.2	100	Ultra-High IO	2/2	1*

* Please refer to the [Link to HA Proxy Hardware Recommendation]
** Quantity will be 2 if redundancy is required

3.2.3. NTP Requirements

OpenShift Container Platform clusters are configured to use a public Network Time Protocol (NTP) servers by default. If you want to use a local enterprise NTP server, or if your cluster is being deployed in a disconnected network, you can configure the cluster to use a specific time server.

3.2.4. DNS Requirements

DNS name resolution is required for the following components:

- The Kubernetes API
- The OpenShift Container Platform application wildcard
- The bootstrap, control plane, and compute machine

3.2.5. Load Balancing Requirements

API and application ingress load balancing infrastructure is mandatory for OpenShift Cluster. In production scenarios, you can choose to deploy the API and application ingress load balancers separately so that you can scale the load balancer infrastructure independently and this is optional.

In this deployment, we are using ha-proxy for load balancing and HA-Proxy is recommended.

API Load Balancer provides a common endpoint for users, both human and machine, to interact with and configure the platform. We will use external load balancer configured on HA-Proxy on ECS instance.

Table 3-3 - API load balancer

Port	Pool Members	Internal	External	Description
6443	Bootstrap and Control Plane machines	✓	✓	OpenShift API
22623		✓	✗	Machine-Config

- **Application Ingress Load Balance** provides an ingress point for application traffic flowing in from outside the cluster. This load balancer will be based on a Dedicated Elastic Load Balancer (ELB)

Table 3-4 - Ingress load balancer

Port	Pool Members	Internal	External	Description
80	Compute machines	✓	✓	HTTP traffic
443		✓	✓	HTTPS traffic

3.3. Security Requirements

It is strongly recommended to follow security best practices, please refer official documentation of cloud security service in the G42 Cloud Help Centre.

3.3.1. Security Groups

Required network ports to be allowed for inbound and outbound connection should be based on an application scenario. G42 Cloud allows security groups and Network ACL to limit port access to VMs and VPC subnets. Please refer a documentation about Security Groups for more details.

Table 3-5- Ports used for all-machine to all-machine communications.

Protocol	Port	Description
ICMP	N/A	Used by network devices to diagnose network communication issues
TCP	1936	(Optional) Required to be open to access load balancer statistics and metrics.

	9000-9999	Host level services, including ports 9100 for Prometheus and the Cluster Version Operator on port 9099.
Protocol	**Port**	**Description**
	10250-10259	The default range of ports that are reserved by Kubernetes
	10256	OpenShift SDN
UDP	4789	Virtual Extensible LAN (VXLAN)
	6081	Network encapsulation protocol Geneve
	9000-9999	Host level services, including ports 9100 for Prometheus and the Cluster Version Operator on port 9099.
	500	IPSec IKE packets
	4500	IPSec NAT-T packets
TCP/UDP	30000-32767	Port range for Kubernetes NodePort service

Table 3-6 -Ports used for control plane communications

Protocol	Port	Description
TCP	6443	OpenShift API

Table 3-7 - Ports that are used for “control-plane” to “control-plane” communication.

Protocol	Port	Description
TCP	2379-2380	Etcd servers and peer hosts communication

3.3.2. Required IAM Permissions

In order to create all the necessary resources in G42 Cloud for the deployment of RHOCP cluster a “Tenant Administrator” role is required for a sub-user account. For more fine-grained access, the following IAM role permissions should be applied:

- ECS Administrator
- ELB Administrator
- VPC Administrator
- IMS Administrator
- OBS Administrator
- NAT Gateway Administrator
- DNS Administrator

3.4. High Availability

Server availability groups are used to achieve the anti-affinity rules for control nodes which can achieve host level resiliency within Availability Zone (AZ).

4. Setup Cluster disconnected installation

In this guide, You can setup cluster in disconnected installation, Upgrade of your cluster, ensure your clusters only use container images that satisfy your organizational controls on external content. Before you install a cluster on infrastructure that you provision in a restricted network, you must mirror the required container images into that environment.

4.1. Prerequisites

You must have a container image registry that supports Docker v2-2 in the location that will host the OpenShift Container Platform cluster, such as one of the following registries:

- Red Hat Quay
- JFrog Artifactory
- Sonatype Nexus Repository
- Harbor

Here the procedure how to configure Red Hat Quay

4.2. Create ECS:

We will install Red Hat Quay on ECS Instance to be used for Mirror registry.

Navigate to “Elastic Compute Server” service from the main page of G42 Cloud Console and click on a button “Create ECS” in the right top corner. Fill all the fields and submit the request.

Table 4.2 Single Node HAProxy ECS Parameters

Section	Example value
AZ	AZ1
Specification	s6.xlarge.4 (4 vCPU | 16GiB)
Public Image	CentOS 8 Stream (600GB)
System Disk	High IO (600 GB)
Quantity	1
Network	vpc-openshift/subnet-openshift
Security Group	sg-ocp4-api-lb
EIP	Do not use
ECS Name	ecs-ocp4-quay
Login Mode (Key pair)	*Choose your ECS Key pair*
Cloud Backup/Recovery	Do not use

Enterprise Project	Default

4.3. Register the IP in DNS

Go to DNS -> select privet zone example.com -> add IP for quay
Name : g42quay.example1.com.
IP: Red Hat Qauy Server IP (172.20.X.X)
Login to https://console.redhat.com/openshift/downloads#tool-mirror-registry and download

- Pull secret
- OpenShift Client (oc) mirror plugin
- mirror registry for Red Hat OpenShift.

4.4. Add the new mirror registry in OCP cluster :

Login to ECS and Run below commands:

```
$ ./mirror-registry install --quayHostname g42quay.example1.com --quayRoot /opt/quay/
#You can find the username password from above installation

#Install podman
$ yum install podman -y

# login local Mirror registry
$ podman login -u init   -p <password>   <host_example_com>:8443>   --tls-verify=false

# Encrypt your mirror registry password
$ echo -n 'init:<password>'  | base64 -w0

#Add New registry in pull secret :-
"auths": {
  "g42quay.example1.com:8443": {
    "auth":        "aW5pdDpKQXZOMjR4VWRnV2FzcmNvRDhTOTVtUGY2MTdlVFowMw==",
    "email": mail@g42cloud.ai
  },
..
$ oc get clusterversion
$ OCP_RELEASE=4.13.8
$ LOCAL_REGISTRY='g42quay.example1.com:8443'
$ LOCAL_REPOSITORY=ocp4/openshift4
$ PRODUCT_REPO='openshift-release-dev'
$ LOCAL_SECRET_JSON=/root/pullsecret (edited one)
$ RELEASE_NAME="ocp-release"
$ ARCHITECTURE=x86_64
$ REMOVABLE_MEDIA_PATH=/
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  --
from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE} --
to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE} --
dry-run

$ export REGISTRY_HOSTNAME=g42quay.example1.com
$ export REGISTRY_PORT=8443
$ echo "" | openssl s_client -showcerts -prexit -connect
"${REGISTRY_HOSTNAME}:${REGISTRY_PORT}" 2> /dev/null | sed -n -e '/BEGIN
CERTIFICATE/,/END CERTIFICATE/ p' > tmp.crt

Copy this tmp.crt to bastion server /etc/pki/ca-trust/source/anchors/ dir and run below command in bastion.
```

```
$ update-ca-trust

Check the image in quay.
$ du -sh /var/lib/containers/storage/volumes/quay-storage
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE} —dy-run
( to update in removable media)
```

Login to OCP cluster using “oc login” or “kubeconfig”, then run below command.

```
$ oc extract secret/pull-secret -n openshift-config --confirm --to=.
$ cat .dockerconfigjson

Add new pull secret with mirror registry created in step 7.5.

#oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=./quay.yaml
```

you can add below configuration and certificate at installation time of openshift by adding additionalTrustBundle, the second option is that add this certificate after installation as below:

```
#oc create configmap registry-config --from-file=quay.ocp4.example1.com..8443=./rootCA.pem -n openshift-
config
#oc patch image.config.openshift.io/cluster --patch '{"spec":{"additionalTrustedCA":{"name":"registry-
config"}}}' --type=merge

#cat <<EOF |oc create -f -
---
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: mirror-ocp
spec:
  repositoryDigestMirrors:
  - mirrors:
    - quay.ocp4.g42cloud.com:8443/ocp4
    source: quay.io/openshift-release-dev/ocp-release
  - mirrors:
    - quay.ocp4.g42cloud.com:8443/ocp4
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
EOF
```

4.5. Disable default Operator Hub in OCP cluster:

```
$ oc patch OperatorHub cluster --type json \ -p '[{"op": "add", "path": "/spec/disableAllDefaultSources",
"value": true}]'
```

5. Installation and Configuration

Red Hat offers two types of methodologies for deploying and maintaining the Red Hat OpenShift Container Platform clusters.

Installer-Provisioned Infrastructure (IPI)
A customer can use a special installation program to deploy a cluster on infrastructure that the installation program provisions, and the cluster maintains.

User-Provisioned Infrastructure (UPI).
A customer can deploy a cluster on infrastructure that the customer prepares and maintain on they own.

This guide uses the platform-agnostic approach described in Red Hat official documentation, which is a generic approach for deploying OpenShift 4 in UPI mode. It is also recommended to review the official documentation from Red Hat for more clarity.

5.1. Deployment Overview

Figure 5-1 - Deployment workflow

5.2. Tools and Prerequisites

The following are the prerequisite to before you start the deployment

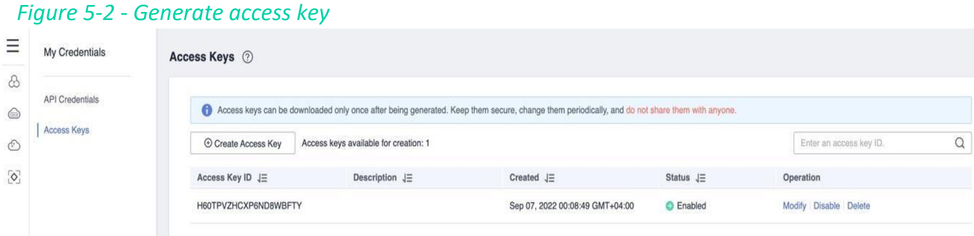
- G42 Cloud tenant administrator account or an IAM user with privileges as described in section 3.3.2
- Access to Red Hat Portal
- A workstation with internet browser and access to the Internet

5.3. Generating Access and Secret Keys

Navigate to "My Credential" section from the main page of the G42 cloud console and select "Access Key" tab on the left side.

Click on "Create Access Key", submit the request, and download an save the key on your workstation.

Figure 5-2 - Generate access key



Access Key and Secret Key (AK/SK) is needed for OBS/S3 bucket access by image repository storage, upload ignition config files and generating signed OBS URL. You can choose to create a new IAM user with limited privilege to OBS service and use the AK/SK from IAM account.

5.4. Creating Virtual Private Cloud and Subnet

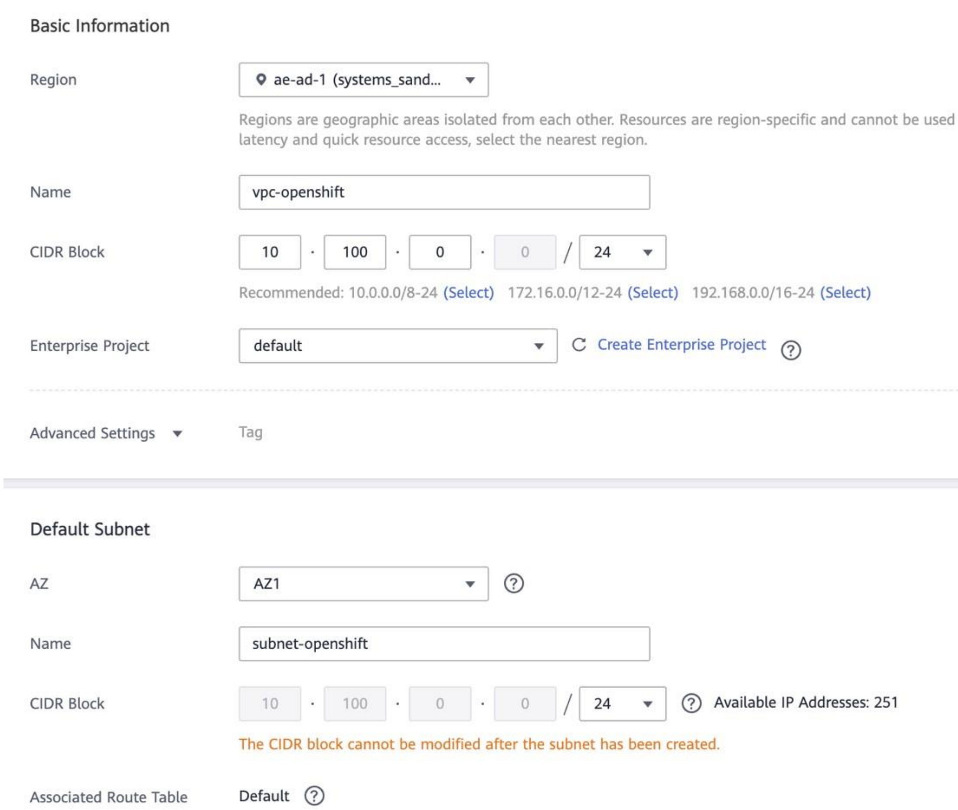
Login to G42 Cloud console with your account and navigate to "Virtual Private Cloud" service under Network. Click on a button "Create VPC" in the right top corner, Create VPC page will be displayed, fill the fill the fields as per table below.

Figure 5-3 - VPC parameter descriptions

Section	Example value
VPC Name	vpc-openshift
CIDR Block	10.100.0.0/24
Enterprise Project	default
AZ	AZ1
Subnet Name	subnet-openshift

Click on button “Create Now” and wait until the VPC and its subnet are created.

Figure 5-4 - Create VPC



You can also refer to the G42 Cloud official user guide.

5.5. Creating Security Groups

A security group is a collection of access control rules for cloud resources, the resources within same security group have the same security protection requirements and that are mutually trusted within a VPC. After a security group is created, you can create various access rules for the security group, these rules will apply to all cloud resources added to this security group.

Based on our deployment guide for OpenShift, it is required to create at-least five security groups for different group of resources. These cloud resource groups are:

- Load Balancer – API
- Load Balancer - Ingress
- Control plane nodes.
- Compute nodes.
- Bastion node

To create the security group, navigate to "Virtual Private Cloud" section under Network Category from the main page of G42 Cloud Console and select "Access Control" => "Security Groups" on the left side of navigation menu. Click on a "Create Security Group" and submit the request.

5.5.1. Security Group for API Load Balancer

Create a new security group as per the table below

Table 5-6 - Security group parameters for API load balancer

Section	Example value
Name	sg-ocp4-api-lb
Enterprise Project	default

After the security group is created, click on "Add Rule" button and add the rules as per table below

Table 5-7-Inbound rule parameter for API load balancer security group

Protocol & Port	Type	Source	Description
ICMP:All	IPv4	0.0.0.0/24	
TCP:6443	Ipv4	0.0.0.0/0	API traffic
TCP: 22623	Ipv4	10.100.0/24	Machine config server ports

5.5.2. Security Group for Ingres Load Balancer

Create a new security group as per the table below

Table 5-8 - Security group parameters for ingress load balancer

Section	Example value
Name	sg-ocp4-ingress-lb
Enterprise Project	default

After the security group is created, click on "Add Rule" button and add the rules as per table below

Table 5-9-Inbound rule parameter for ingress load balancer security group

Protocol & Port	Type	Source	Description
ICMP:All	IPv4	0.0.0.0/24	
TCP:80	IPv4	0.0.0.0/24	HTTP Traffic

TCP:443	IPv4	0.0.0.0/24	HTTPS Traffic

5.5.3. Security Group for Control Plane Nodes

Create a new security group as per the table below.

Table 5-10 - Security group parameters for ingress load balancer

Section	Example value
Name	sg-ocp4-control
Enterprise Project	default

After the security group is created, click on “Add Rule” button and add the rules as per table below.

Table 5-11 - Inbound rule parameter for control plane security group

Protocol & Port	Type	Source	Description
ICMP:All	IPv4	0.0.0.0/24	
TCP:22	IPv4	10.100.0/24	
TCP:6443	IPv4	sg-ocp4-api-lb	
TCP:22623	IPv4	sg-ocp4-api-lb	
TCP:19531	IPv4	10.100.0/24	
TCP: 2379-2380	IPv4	10.100.0/24	etcd
UDP:4789	IPv4	10.100.0/24	VxLAN Packets
TCP:9000-9999	IPv4	10.100.0/24	Internal connects
UDP:9000-9999	IPv4	10.100.0/24	Internal connects
TCP:10250-10259	IPv4	10.100.0/24	Kubelet, Scheduler
TCP:30000-32767	IPv4	0.0.0.0/24	Kubernetes Ingress
UDP:30000-32767	IPv4	0.0.0.0/24	Kubernetes Ingress

5.5.4. Security Group for Compute Nodes

Create a new security group as per the table below

Table 5-12 - Security group parameters for compute nodes

Section	Example value
Name	sg-ocp4-compute
Enterprise Project	default

After the security group is created, click on "Add Rule" button and add the rules as per table below.

Table 5-13 - Inbound rule parameter for compute nodes security group

Protocol & Port	Type	Source	Description
ICMP:All	IPv4	0.0.0.0/24	
TCP:22	IPv4	10.100.0/24	
TCP:80	IPv4	sg-ocp4-lb-api	HTTP Traffic
TCP:443	IPv4	sg-ocp4-lb-api	HTTPS Traffic
UDP:4789	IPv4	0.0.0.0/24	VxLAN Packets
TCP:9000-9999	IPv4	10.100.0/24	Internal connects
UDP:9000-9999	IPv4	10.100.0/24	Internal connects
TCP:10250	IPv4	10.100.0/24	Kubelet, Scheduler
TCP:30000-32767	IPv4	0.0.0.0/24	Kubernetes Ingress
UDP:30000-32767	IPv4	0.0.0.0/24	Kubernetes Ingress

5.5.5. Security Group for Bastion Nodes

Create a new security group as per the table below.

Table 5-14 - Security group parameters for Bastion/Helper node

Section	Example value
Name	sg-ocp4-bastion
Enterprise Project	default

After the security group is created, click on "Add Rule" button and add the rules as per table below.

Table 5-15-Inbound rule parameter for bastion/helper security group

Protocol & Port	Type	Source	Description
ICMP:All	IPv4	0.0.0.0/24	
TCP:22	Ipv4	0.0.0.0/24	

5.6. Creating NAT Gateway

NAT Gateway is required for the control plane and compute nodes to interact with the Internet in order register the cluster and download installation images and software.

Navigate to "NAT Gateway" section from the main page of G42 Cloud Console and click on a button "Create Public NAT Gateway". For optimal performance, it's crucial to match the NAT gateway specifications with the cluster size. Larger clusters require NAT gateways of extra-large size. Please consult the documentation at the following link for detailed specifications:
https://docs.vb.g42cloud.com/en-us/usermanual/nat/en-us_topic_0086739763.html

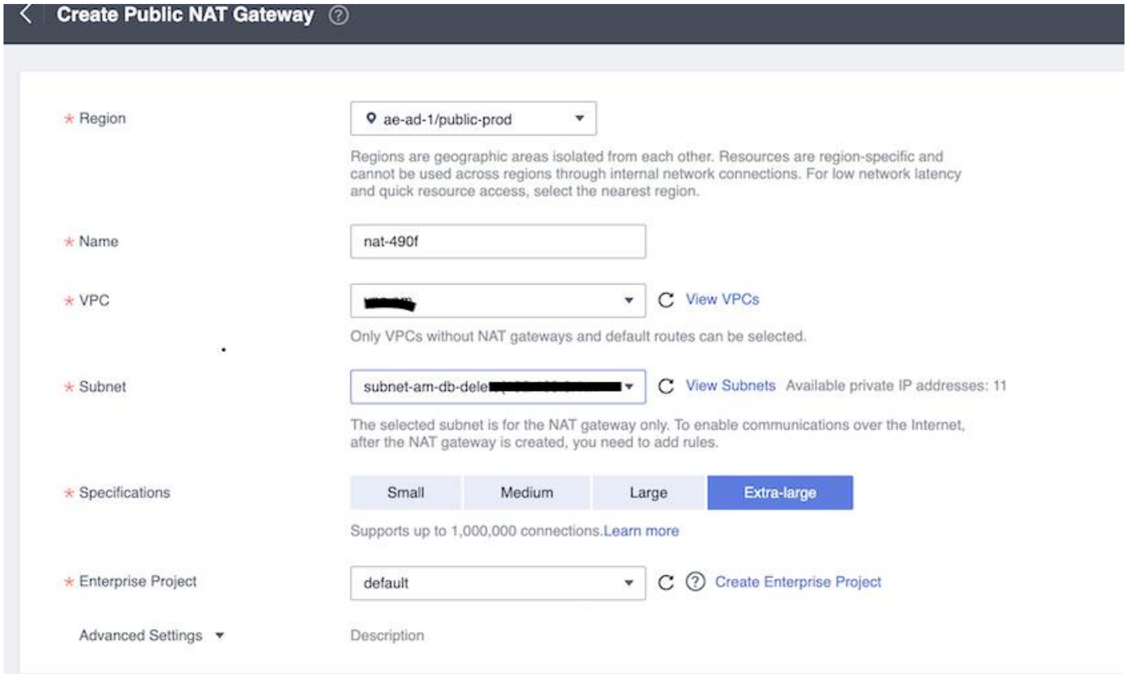
Table 5-16 – Parameter description for NAT Gateway

Section	Example value
Name	nat-openshift
VPC	vpc-openshift
Subnet	subnet-openshift (10.100.0.0/24)
Type	Small
Enterprise Project	default

Submit the request by clicking on "Create Now" button in the right bottom corner.

Figure 5-17 Create NAT Gateway

The next step is to add SNAT rule to allow our OpenShift nodes to send network traffic to the Internet.



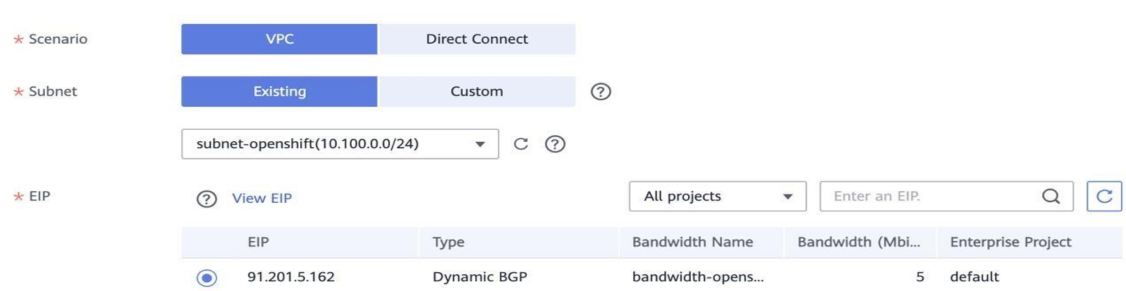
Select the recently created NAT Gateway and create a new SNAT rule. Click on a button “Add Rule” and fill the fields. (Optional : Add multiple EIP on NAT).

Table 5-18 - SNAT rule parameters

Section	Example value
Scenario	VPC
Subnet	Existing (subnet-openshift 10.100.0.0/24)
EIP	Select an already existed or create a new with high bandwidth.

Submit the rule by clicking on a button “OK”.

5-19 Create SNAT rule



5.7. Creating OBS Buckets

There are two OBS buckets are required for this deployment.

- A bucket to store the ignition configuration of Red Hat OpenShift Container Platform cluster and an image of Red Hat CoreOS.
- A bucket to store container images as part of OpenShift image repository.

Navigate to "Storage" > "Object Storage Service" to reach OBS console page from the main page of G42 Cloud Console and click on a button "Create Bucket". Fill the fields and submit by clicking on a button "Create Now".

Table 5-20 -Parameter description for buckets

Section	Bucket 1	Bucket 2
Name	obs-openshift	obs-image-registry
Bucket Policy	Private	Private
Default Encryption	Enabled	Enabled
KMS Key	obs/default	obs/default
Enterprise Project	default	default

In order to enable encryption at rest with your custom key you are required to create your own KMS key. In this guide default key is used.

Bucket names are unique in every region, you should select a unique name.

5.8. Downloading Software from Red Hat Portal

The table below gives the summary of tools and packages required to deploy OpenShift.

Table 5-21 Summary of items required from Red Hat portal

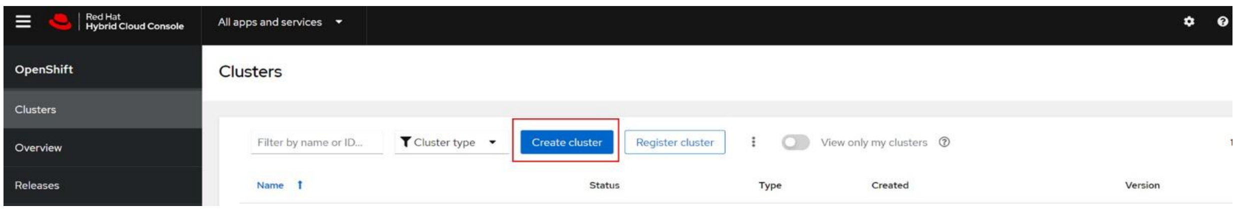
File	Description
OpenShift Installer	OpenShift installation program
Command-Line interface	CLI tool to manage the cluster
Pull-secret	Encrypted key that is required for the installation.
rhcos-openstack.x86_64.qcow2.gz	An OpenStack based image of Red Hat CoreOS operating system

5.8.1. Downloading Installer, CLI tool and Pull Secret

Login to the [Red Hat Cloud Portal] to download the required software packages of the latest version of Red Hat OpenShift Container Platform.

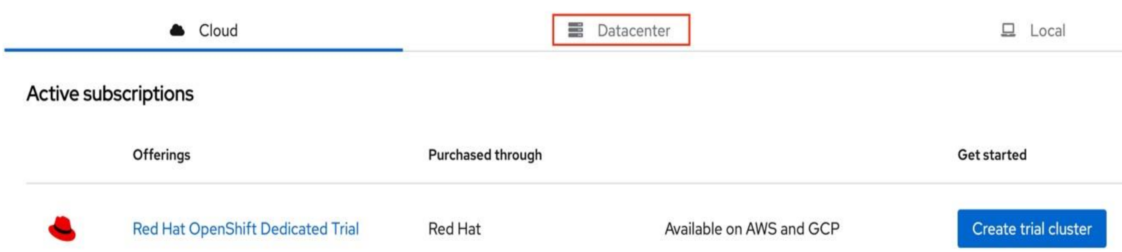
Navigate to "Clusters" item on left hand navigation plane and select create cluster tab on the navigation page.

Figure 5-22 - Create a new cluster



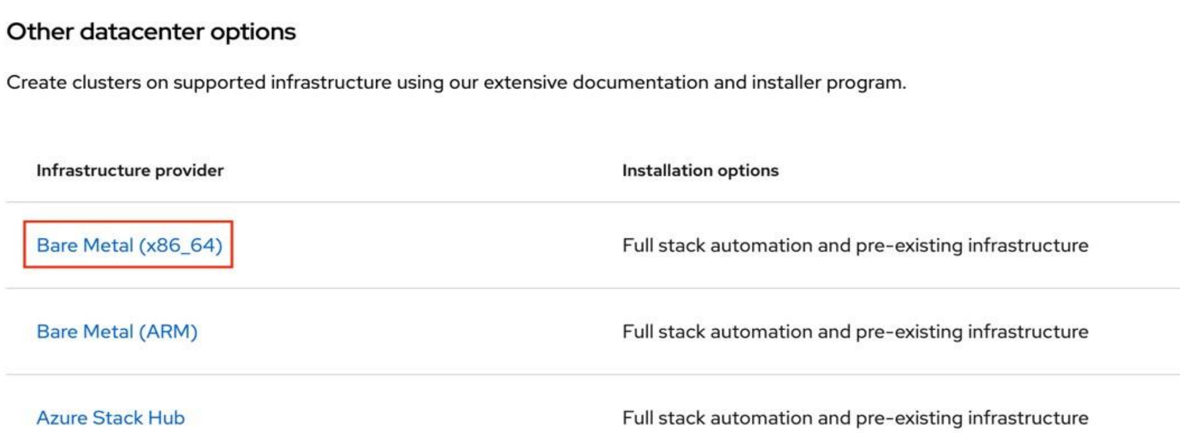
On the create page select "Datacenter" tab in the top navigation menu as shown in the figure below.

Figure 5-23 – Selecting model on Red Hat portal



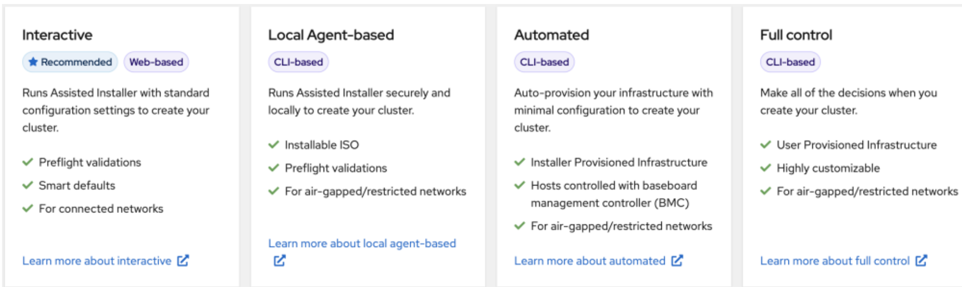
And click on "Bare Metal (x86_64)" link button.

5-24 Select installer architecture



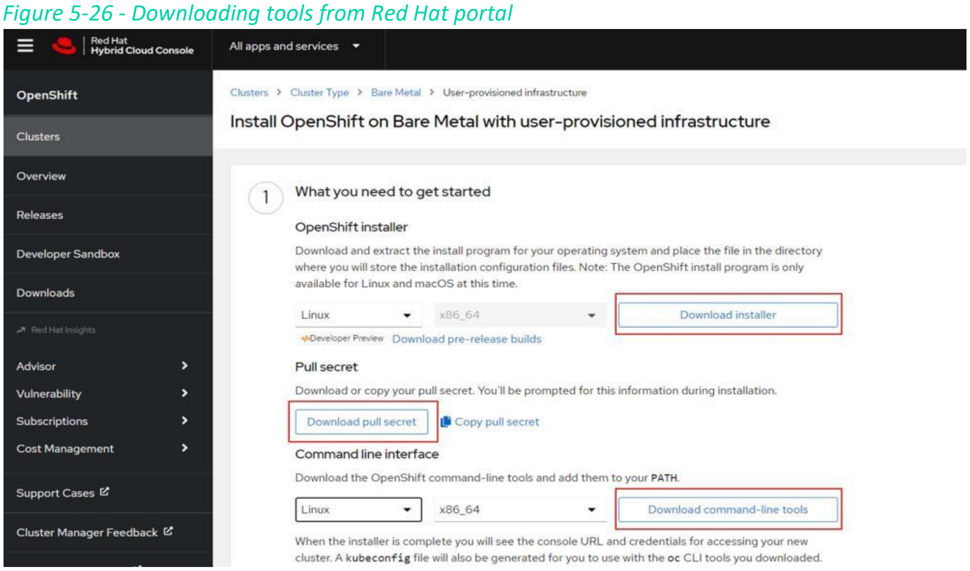
Select the installation type Full-Control (CLI based)

Figure 5-25 - Selecting installer type



Download installer, command-line tools and pull secret from the page as described in figure and table below.

Figure 5-26 - Downloading tools from Red Hat portal



Don't download the CoreOS image from this page as we will use openstack CoreOS image

Table 5-27 Description of Resources to be downloaded

File	Description	CPU Architecture
OpenShift Installer	OpenShift installation program	Linux x86
Command-Line interface	CLI tool to manage the cluster	Linux x86

Pull-secret	Encrypted key that is required for the installation.	N/A

5.8.2. Downloading CoreOS Image

We will need to download openstack CoreOS image from Red Hat portal. You should always select the latest release version for the version you wish to deploy.

Open a new page with the latest available minor version for 4.13 orm and download the following file "***rhcos-openstack.x86_64.qcow2.gz***"

Figure 5-28 - Download CoreOS Image

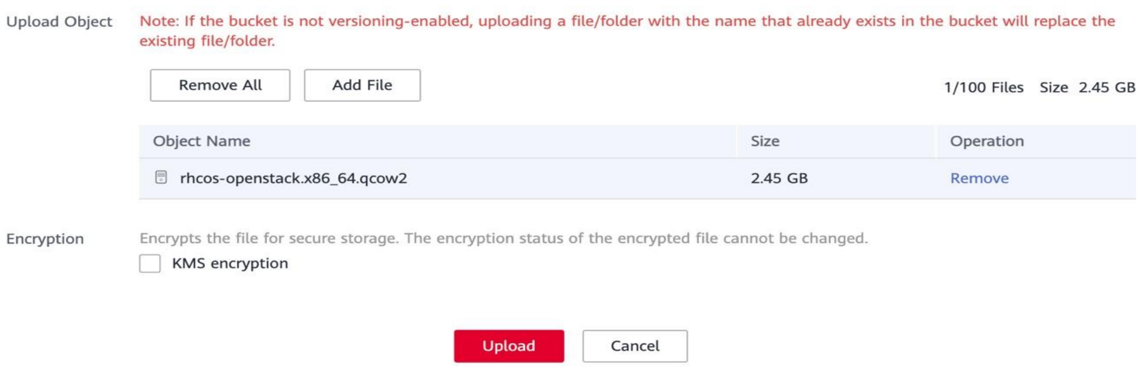
rhcos-metal4k.x86_64.raw.gz	1 GB	Thu Aug 18 17:18:03 2022
rhcos-nutanix.x86_64.qcow2	1 GB	Thu Aug 18 17:18:10 2022
rhcos-openstack.x86_64.qcow2.gz	1 GB	Thu Aug 18 17:18:13 2022
rhcos-ostree.x86_64.ociarchive	974 MB	Thu Aug 18 17:18:15 2022
rhcos-qemu.x86_64.qcow2.gz	1 GB	Thu Aug 18 17:18:25 2022
rhcos-vmware.x86_64.ova	1 GB	Thu Aug 18 17:18:25 2022
sha256sum.txt	3 KB	Thu Aug 18 17:18:34 2022

Unzip the downloaded file archive, the image file should have an extension qcow2.

5.9. Uploading RHCOS Image

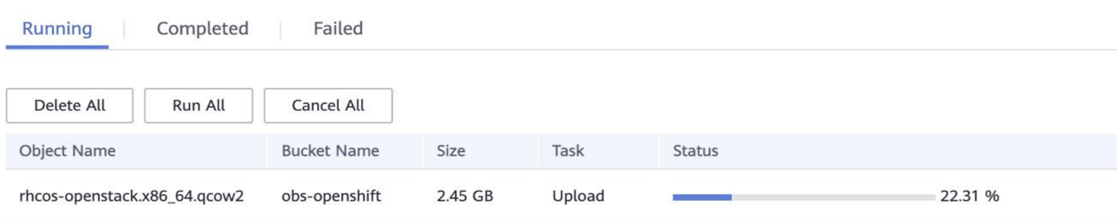
To upload the downloaded image of RHCOS, navigate to "Object Storage Service" under storage category from the main page of G42 Cloud Console and find the OBS bucket created in Section 4.7, we used the name "obs-openshift". Click to navigate to bucket page. In the bucket page, click on a button "Upload Object" and select the image from your local disk. Submit the upload by click on a button "Upload"

5-29 – Uploading the CoreOS qcow2 image to OBS bucket



The process can take up to 20-30 minutes depends on your network speed and bandwidth. You can monitor the progress as described in the figure below.

Figure 5-30 - Checking the upload status



5.10. Creating Private Image for RHCOS

After the image has been uploaded to the OBS, it is required to register it as an private image to be able to seamlessly create an ECS instance.

Navigate to “Image Management Service” under computing section from the main page of G42 Cloud Console and click on a button “Create Image”.

Table 5-31 - Parameters for creating private image

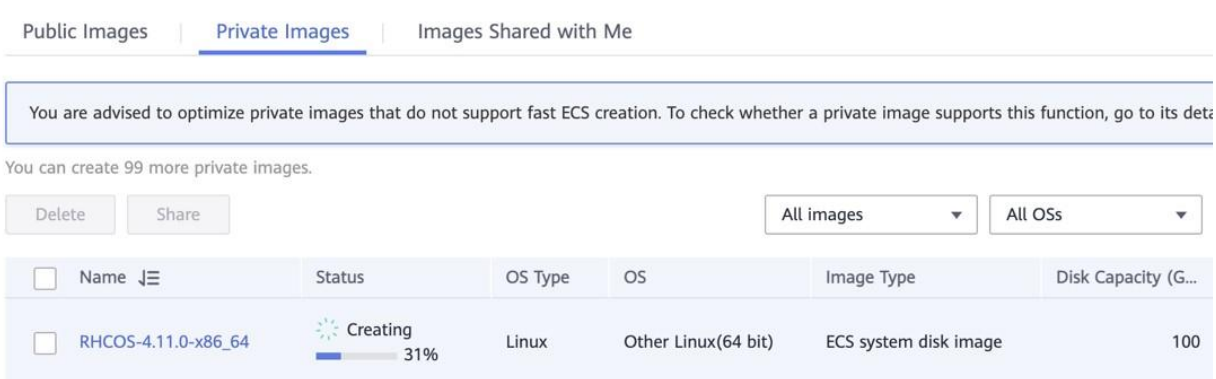
Section	Example value
Type	System disk image
Source	Image File
Bucket Name	obs-openshift

File Name	rhcos-openstack.x86_64.qcow2
Automatic Configuration	True

Boot Mode	BIOS
OS	Other Linux(64 bit)
System Disk (GB)	200
Image Name	RHCOS-4.13.0-x86_64

Submit the application by clicking on a button "Apply Now". The process of the image creation can take up to 15-20 minutes.

Figure 5-32 - Create private image



5.11. Create Helper Server

We will need to deploy a bastion server which will be used to assist in the deployment.

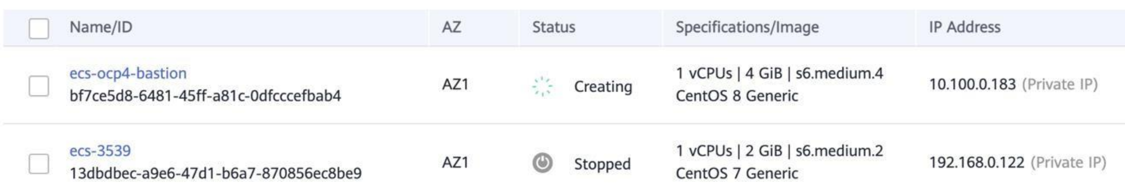
Navigate to "Elastic Compute Server" service from the main page of G42 Cloud Console and click on a button "Create ECS" in the right top corner. Fill all the fields and submit the request.

Table 5-33-Helper server parameters

Section	Example value
AZ	AZ1
Specification	s6.2xlarge.2 (8 vCPU | 16GiB)

Public Image	CentOS 8 Generic (40GB)
System Disk	High IO (40 GB)
Quantity	1
Network	vpc-openshift/subnet-openshift
Security Group	sg-ocp4-bastion
EIP	Do not use
ECS Name	ecs-ocp4-bastion
Login Mode (Key pair)	*Choose your ECS Key pair*
Cloud Backup/Recovery	Do not use
Enterprise Project	Default

Figure 5-34 - Helper server creation progress



In order to connect to this node using SSH you can either assign an EIP or can configure a DNAT rule in the NAT Gateway. In our guide we will use DNAT feature of NAT Gateway.

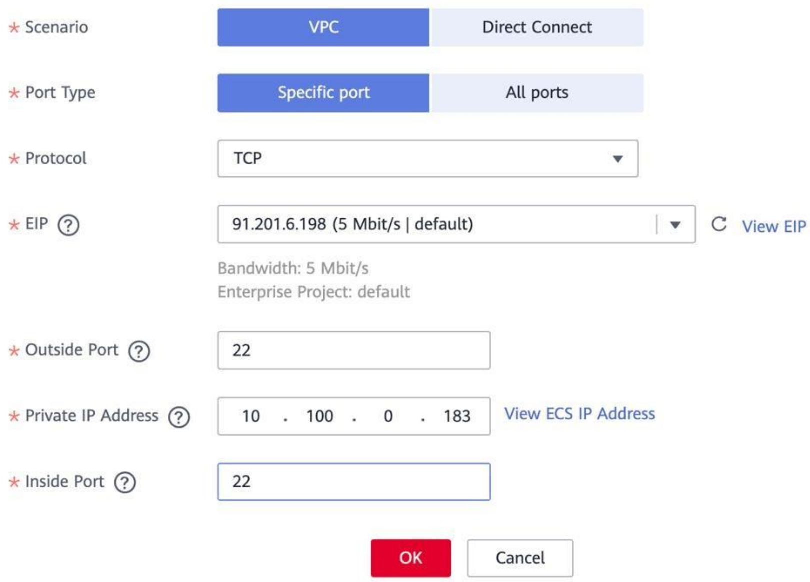
Navigate to “NAT Gateway” service from the main page of G42 Cloud Console and select “nat-openshift” gateway. Click on a button “Add DNAT Rule” and add the next rule.

Table 5-35 - SNAT parameters for NAT Gateway

Section	**Example value**
Scenario	VPC
Port Type	Specific Port
Protocol	TCP
EIP	*Select an already existed or create a new*
Outside Port	22
Private IP Address	10.100.0.183 (*IP address of Bastion ECS)*
Inside Port	22

You can refer to the below figure below as an example.

Figure 5-36-SNAT configuration page



Now you can connect to bastion/helper node by using SSH to EIP assigned of the NAT Gateway's SNAT service on port 22 from your local workstation.

```
% ssh -l root@91.201.6.198 -i Key-Pair-openshift
The authenticity of host '91.201.6.198 (91.201.6.198)' can't be established.
ECDSA key fingerprint is SHA256:U4xWUq9LC2dcr0JNBK4tI2t42UL5bIVO+q5gVmjepjE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
[centos@ecs-ocp4-bastion ~]$
```

Also, you can check SNAT rule that we created in a section 4.6 by ICMP request of some external portal.

```
[centos@ecs-ocp4-bastion ~]$ ping google.com -c 4
PING google.com (142.250.181.110) 56(84) bytes of data.
64 bytes from f14.1e100.net (142.250.181.110): icmp_seq=1 ttl=53 time=10.7 ms
64 bytes from f14.1e100.net (142.250.181.110): icmp_seq=2 ttl=53 time=9.15 ms
64 bytes from f14.1e100.net (142.250.181.110): icmp_seq=3 ttl=53 time=9.32 ms
64 bytes from f14.1e100.net (142.250.181.110): icmp_seq=4 ttl=53 time=9.04 ms ---
google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 7ms rtt
min/avg/max/mdev = 9.043/9.552/10.694/0.670 ms
```

5.12. Creating Ignition Configurations

In order to start the installation, we will need to create the installation configuration for OpenShift. Login to the installation helper server created in section 4.11.

Create an install dir.

```
[centos@ecs-ocp4-bastion ~]$ mkdir ~/ocp4
[centos@ecs-ocp4-bastion ~]$ ls -l
total 0
drwxrwxr-x. 2 centos centos 6 Sep 13 09:47 ocp4
```

Copy the pull-secret and software from your local workstation to the installation server.

```
[centos@ecs-ocp4-bastion ~]$ ls -l total 387832
drwxrwxr-x. 2 centos centos         6 Sep 13 09:47
ocp4
-rw-rw-r--. 1 centos centos  52549905 Aug 30 17:47 openshift-client-linux.tar.gz
-rw-rw-r--. 1 centos centos 344580540 Aug 30 17:47 openshift-install-linux.tar.gz
-rw-rw-r--. 1 centos centos      2799 Sep 13 09:52 pull-secret
```

Unpack and install the software to be executable.

```
[centos@ecs-ocp4-bastion ~]$ tar zxvf openshift-client-linux.tar.gz
[centos@ecs-ocp4-bastion ~]$ tar zxvf openshift-install-linux.tar.gz
[centos@ecs-ocp4-bastion ~]$ sudo cp oc /bin/
[centos@ecs-ocp4-bastion ~]$ sudo cp kubectl /bin/
[centos@ecs-ocp4-bastion ~]$ sudo cp openshift-install /bin/
```

As the next step, generate an SSH-key that will be distributed across OpenShift nodes. You will login to them from the installation server by using this key.

```
[centos@ecs-ocp4-bastion ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/centos/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/centos/.ssh/id_rsa.
Your public key has been saved in /home/centos/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:2+ieRXD2VRvlheGitzFIXAdaXNrjjo5dygWtm/mLiiQ centos@ecs-ocp4-bastion.novalocal
The key's randomart image is:
+---[RSA 3072]----+
|           .+o++=|
|         . +.=.o+|
|        . * o =..|
|         = + = . |
|        S + * o  |
|         = . O   |
|      E + o + +  |
|       + + = O   |
|       .= o.X.o. |
+----[SHA256]-----+
```

Next, create an “install-config.yaml” file. Add a content of pull-secret and ssh public key in the end of the file.

```
[centos@ecs-ocp4-bastion ~]$ vi ~/ocp4/install-config.yaml
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  replicas: 2
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: ocp4
networking:
  clusterNetworks:
  - cidr: 10.254.0.0/16
    hostPrefix: 24
  networkType: OVNKubernetes
  serviceNetwork:
```

```
  - 172.30.0.0/16
platform:
 none: {}
pullSecret: '<pull secret>'
sshKey: '<ssh-key from bastion user>'
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
  ……
  -----END CERTIFICATE-----
imageContentSources:
- mirrors:
  - quay.ocp4.g42cloud.com:8443/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - quay.ocp4.g42cloud.com:8443/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

Optional: imageContentSources and imageContentSources are optional. We can add internal Quay certificate if required for disconnected environments and Ingress CA-Bundle certificate.

Create the manifest configuration.

```
[centos@ecs-ocp4-bastion ~]$ cd ~/ocp4
[centos@ecs-ocp4-bastion ocp4]$ openshift-install create manifests
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster
settings
INFO Manifests created in: manifests and openshift
```

Edit the manifests/cluster-scheduler-02-config.yml Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines by setting "mastersSchedulable" to false.

```
[centos@ecs-ocp4-bastion ocp4]$ sed -i 's/mastersSchedulable: true/mastersSchedulable: false/g'
manifests/cluster-scheduler-02-config.yml
```

Next, generate the ignition configs.

```
[centos@ecs-ocp4-bastion ocp4]$ openshift-install create ignition-configs
INFO Consuming OpenShift Install (Manifests) from target directory
INFO Consuming Common Manifests from target directory
INFO Consuming Worker Machines from target directory
INFO Consuming Master Machines from target directory
INFO Consuming Openshift Manifests from target directory
INFO Ignition-Configs created in: . and auth
```

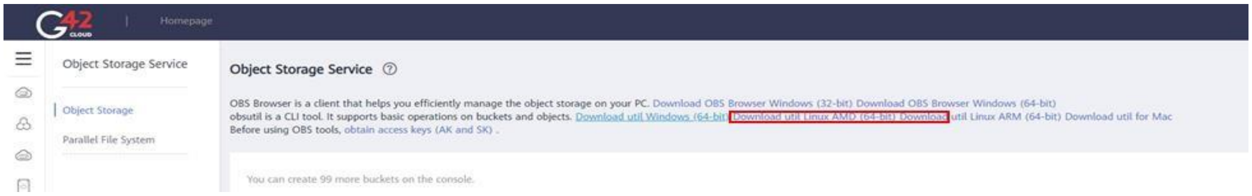
5.13. Uploading Ignition Configuration

We will use obsutil a command-line utility to upload the files to OBS bucket and generate the pre-signed URL for ignition file.

OBS console provides you with the download links of obsutil for different operating systems.

From G42 Cloud home page select the "Object Storage Service" under storage to navigate to OBS console page. From the home page, download the obsutil from the link as shown below.

Figure 4-18 – Get obsutil from OBS console home page



Copy the link and run the command as shown below to download

```
[centos@ecs-ocp4-bastion ocp4]$ wget https://obs-community.obs.ae-
ad1.g42cloud.com/obsutil/obsutil_hcso_linux_amd64_5.3.4.tar.gz
[centos@ecs-ocp4-bastion ocp4]$ cd
[centos@ecs-ocp4-bastion ocp4]$ tar -xzvf obsutil_hcso_linux_amd64_5.3.4.tar.gz
[centos@ecs-ocp4-bastion ocp4]$ cd obsutil_linux_amd64_5.3.4
[centos@ecs-ocp4-bastion ocp4]$ chmod 755 obsutil
```

Configure obsutil, use the AK/SK created in section 4.3

```
[centos@ecs-ocp4-bastion ocp4]$./obsutil config -interactive Please
input your ak:
xxxxxxxxxxxxxxxxxxxxxxxxx
Please input your sk:
xxxxxxxxxxxxxxxxxxxxxxxxx Please
input your endpoint:
xxxxxxxxxxxxxxxxxxxxxxxxx Please
input your token:

Config file url:
  /root/.obsutilconfig

Update config file successfully!
```

After successful setup you should be able to access the bucket via cli. To test you can run using the command shown below to ensure that obsutil is configured properly.

```
[centos@ecs-ocp4-bastion ocp4]$./obsutil ls

Bucket                      CreationDate                Location        BucketType    obs://bucket001
2018-09-03T01:53:02Z        example        OBJECT    obs://bucket002              2018-11-01T01:40:01Z
example        OBJECT       obs://bucket003             2018-10-25T11:45:45Z          example        OBJECT
obs://bucket004             2018-10-26T02:33:09Z        example         OBJECT        obs://bucket005
2018-10-26T02:34:50Z        example        OBJECT    :
```

In case of any issue you can refer to obsutil user guide on G42 Cloud portal to troubleshoot the configuration.

5.13.1. Uploading Files to OBS Bucket

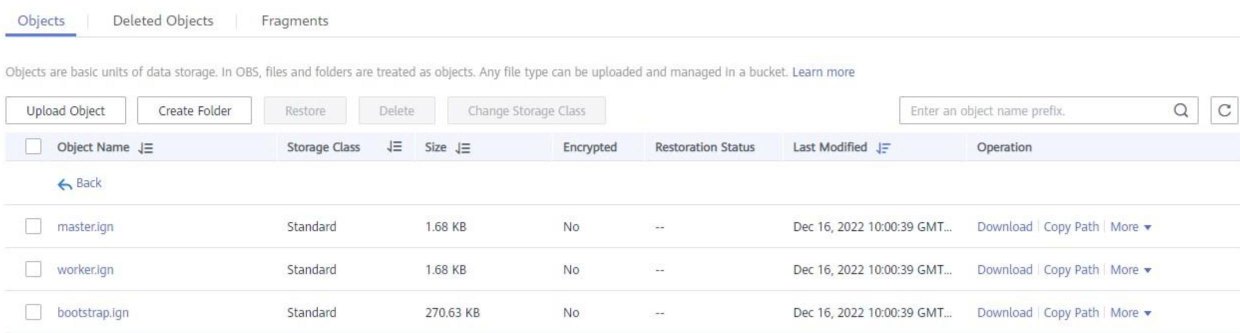
Use obsutil to upload the configuration files to the bucket created in section 4.7

```
[centos@ecs-ocp4-bastion ocp4]$./obsutil cp ~/ocp4/master.ign obs://obs-openshift/
[centos@ecs-ocp4-bastion ocp4]$./obsutil cp ~/ocp4/worker.ign obs://obs-openshift/
[centos@ecs-ocp4-bastion ocp4]$./obsutil cp ~/ocp4/bootstrap.ign obs://obs-openshift/
```

⚠ "obs-openshift" is the name of bucket we created in our demo, you should have your own bucket unique name and replace it the command

5.13.2. Downloading Compute and Control Ignition Files to Workstation

You should download the control.ign and compute.ign file to your local workstation from OBS console, by navigating to the bucket, these files will be needed later while creating ecs for control/control and compute/compute nodes.



5.13.3. Generating Pre -Signed file download URL

You can use obsutil sign command to generate the download link of a specified object in a bucket. We will use this to create signed url for bootstrap ignition file.

```
[centos@ecs-ocp4-bastion ocp4]$./obsutil sign obs://obs-openshift/bootstrap.ign Download
url of [obs://obs-openshift/bootstrap.ign] is:
http:// obs.ae-ad-1.g42cloud.com/bucket-test/test.txt?AccessKeyId=xxxx&Expires=1552548758&Signature=xxxx
```

"obs-openshift" is the name of bucket we created in our demo, you should have your own bucket unique name and replace it the command

Copy the pre-signed URL of the file. This link will be required later to deploy the bootstrap server

5.14. Creating and Configuring Domain Name Service (DNS)

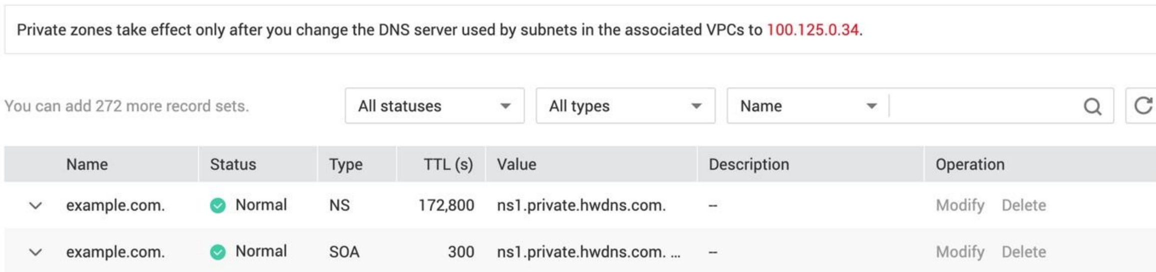
Navigate to "Domain Name Service" in the main page of G42 Cloud Console and click on "Create Private Zone" button in the right top corner. Follow the below table to fill the required fields and submit the request.

Table 5-37 - DNS parameters

Section	Example value
Name	example.com
VPC	vpc-openshift
Email	*Leave your corporate email address*

This new zone will be used by our OpenShift cluster, in later section we will add DNS records to this private zone.

Figure 5-38 - DNS Zone



5.15. Create Server Group for master and worker.

G42 Cloud offers VM affinity through server groups, which are recommended for organizing compute and control nodes. By utilizing server groups, ECS instances can be distributed across separate hosts within the same Availability Zone (AZ), thereby enhancing host-level resilience.

When creating ECS instances, you'll encounter the 'ECS Group (Optional)' option, allowing you to define ECS groups. It's important to establish two groups: one for master nodes and another for worker nodes. Master nodes should be assigned to the master ECS group, while worker nodes should belong to the worker ECS group.

- Navigate to the ECS page.
- In the left corner, locate the "ECS group" option and click on it.
- Create a new ECS group by providing a name, such as "master" or "worker."
- Choose the "Anti-Affinity" policy for the group.

5.16. Deploying a Bootstrap Server

Navigate to “Elastic Cloud Server” in the main page of G42 Cloud Console and click on “Create ECS” button in the right top corner. Follow the below table to fill the required fields. Submit the request.

The recommended ECS flavour is d3.2xlarge.8 which ensuring the Max/Assured Bandwidth is 5/5 Gbit/s.
https://docs.g42cloud.com/usermanual/ecs/en-us_topic_0264030168.html

Table 5-39 - Bootstrap server parameters

Section	Example value
AZ	AZ1
Specification	d3.2xlarge.8 (8 vCPU | 64GiB)
Private Image	RHCOS-4.13.0-x86_64(100GB)

System Disk	Ultra-High IO (200GB)
Quantity	1
Network	vpc-openshift / subnet-openshift
Security Group	sg-ocp4-control
EIP	Do not use
ECS Name	esc-ocp4-bootstrap
Login Mode (Key pair)	*Choose your ECS Key pair*
Cloud Backup/Restore	Do not use
Advanced Option	Configure Now (As Text)

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<pre-signed URL created in section 4.13.3>"
        }
      ]
    },
    "version": "3.2.0"
  }
}
```

Enterprise Project	default
Max./Assured Bandwidth (Gbit/s)	5/5

This ignition configuration as a link is required due to limitation to upload user-data size limitation of 32K. The size of bootstrap ignition file is more than 250K. This workaround will help to overcome the limitation and proceed with ECS deployment of the Bootstrap node. The link to the ignition file is a pre-signed URL from section 4.13.2

5.17. Deploying Control Nodes

Navigate to “Elastic Cloud Server” in the main page of G42 Cloud Console and click on “Create ECS” button in the right top corner. Follow the below table to fill the required fields. Submit the request.

In this deployment of RHCOP - single Availability Zone (AZ) for all control plane and compute nodes is used.

Table 5-40 - Control Plane ECS Parameters

Section	Example value

AZ	AZ1
Specification	d3.2xlarge.8 (8 vCPU | 64GiB)
Private Image	RHCOS-4.13.0-x86_64(100GB)
System Disk	Ultra-High IO (200GB)
Quantity	3
Network	vpc-openshift / subnet-openshift
Security Group	sg-ocp4-control
EIP	Do not use
ECS Name	esc-ocp4-control
Login Mode (Key pair)	*Choose your ECS Key pair*
Cloud Backup/Restore	Do not use
Advanced Option	Configure Now (As a File) Select and upload “control.ign” file you saved in your local workstation. This was downloaded in section 4.13.2
Max./Assured Bandwidth (Gbit/s)	5/5
ECS Group (Optional)	Master

5.18. Deploying Worker Nodes

Navigate to “Elastic Cloud Server” in the main page of G42 Cloud Console and click on “Create ECS” button in the right top corner. Follow the below table to fill the required fields. Submit the request.

Section	Example value

AZ	AZ1
Specification	d3.2xlarge.8 (8 vCPU | 64GiB)
Private Image	RHCOS-4.13.0-x86_64(100GB)
System Disk	Ultra-High IO (200GB)
Quantity	3 (minimum)
Network	vpc-openshift / subnet-openshift
Security Group	sg-ocp4-control
EIP	Do not use
ECS Name	esc-ocp4-control
Login Mode (Key pair)	*Choose your ECS Key pair*
Cloud Backup/Restore	Do not use
Advanced Option	Configure Now (As a File) Select and upload "control.ign" file you saved in your local workstation. This was downloaded in section 4.13.2
Max./Assured Bandwidth (Gbit/s)	5/5
ECS Group (Optional)	Worker

5.19. Configuring HAProxy on a Single ECS Node

We will install HAProxy on ECS Instance to be used as an external API load balancer. This section provides steps to create a single node HAProxy, if you wish to create a HAProxy with High Availability refer to next section.

Navigate to "Elastic Compute Server" service from the main page of G42 Cloud Console and click on a button "Create ECS" in the right top corner. Fill all the fields and submit the request.

Table 5-41 Single Node HAProxy ECS Parameters

Section	Example value
AZ	AZ1
Specification	c6s.4xlarge.2 (4 vCPU | 8GiB)
Public Image	Centos (100GB)
System Disk	Ultra-High IO (100 GB)
Quantity	1
Network	vpc-openshift/subnet-openshift
Security Group	sg-ocp4-api-lb
EIP	Do not use
ECS Name	ecs-ocp4-haproxy
Login Mode (Key pair)	*Choose your ECS Key pair*
Cloud Backup/Recovery	Do not use
Enterprise Project	Default
Max./Assured Bandwidth (Gbit/s)	2/2

- Login to the HAProxy ECS node and install HAProxy software

```
[centos@ecs-ocp4-haproxy]$ yum install haproxy -y
[centos@ecs-ocp4-haproxy]$ systemctl enable haproxy
```

- Adjust the following SELinux setting and restart HAProxy

```
[centos@ecs-ocp4-haproxy]$ setsebool -P haproxy_connect_any=1
[centos@ecs-ocp4-haproxy]$ systemctl restart haproxy
```

- Configure HAProxy, you need to add DNS names and IP addresses of the bootstrap and control plane (control) nodes.

  Configure api service frontend and backend on port 6443/tcp and HTTPS health check for /readyz path.

  Configure machine configuration service frontend and backend on port 22623/tcp and HTTPS health check for /healthz path.

```
[root@ecs-ocp4-haproxy]$ more /etc/haproxy/haproxy.cfg
...
global
   chroot      /var/lib/haproxy
   pidfile     /var/run/haproxy.pid
   maxconn     4000
   user        haproxy
   group       haproxy
   daemon

   # turn on stats unix socket
   stats socket /var/lib/haproxy/stats

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
   mode                    http
   log                     global
   option                  httplog
   option                  dontlognull
   option http-server-close
   option forwardfor       except 127.0.0.0/8
   option                  redispatch
   retries                 3
   timeout http-request    10s
   timeout queue           1m
   timeout connect         10s
   timeout client          1m
   timeout server          1m
   timeout http-keep-alive 10s
   timeout check           10s
   maxconn                 3000
--
#---------------------------------------------------------------------
# Kube API Server
#---------------------------------------------------------------------
frontend k8s_api_frontend
      bind :6443
      default_backend k8s_api_backend
      mode tcp
backend k8s_api_backend
      mode tcp
      balance roundrobin
      option httpchk GET /readyz HTTP/1.0
      server bootstrap.ocp4.g42cloud.com 172.20.0.76:6443 check check-ssl inter 1s fall 2 rise 3 verify none
      server master1.ocp4.g42cloud.com 172.20.0.250:6443 check check-ssl inter 1s fall 2 rise 3 verify none
      server master2.ocp4.g42cloud.com 172.20.0.150:6443 check check-ssl inter 1s fall 2 rise 3 verify none
      server master3.ocp4.g42cloud.com 172.20.0.219:6443 check check-ssl inter 1s fall 2 rise 3 verify none

#---------------------------------------------------------------------
# OCP Machine Config Server
#---------------------------------------------------------------------
frontend ocp_machine_config_server_frontend
      mode tcp
      bind :22623
      default_backend ocp_machine_config_server_backend
backend ocp_machine_config_server_backend
      mode tcp
      balance roundrobin
      option httpchk GET /healthz HTTP/1.0
      server bootstrap.ocp4.g42cloud.com 172.20.0.76:22623 check check-ssl inter 1s fall 2 rise 3 verify none
      server master1.ocp4.g42cloud.com 172.20.0.250:22623 check check-ssl inter 1s fall 2 rise 3 verify none
      server master2.ocp4.g42cloud.com 172.20.0.150:22623 check check-ssl inter 1s fall 2 rise 3 verify none
      server master3.ocp4.g42cloud.com 172.20.0.219:22623 check check-ssl inter 1s fall 2 rise 3 verify none
```

```
frontend ocp_apps_80_server_frontend
    mode tcp
    bind :80
    default_backend ocp_machine_apps_80_backend
backend ocp_machine_apps_80_backend
    mode tcp
    balance roundrobin
    server worker1.ocp4.g42cloud.com 172.20.0.170:80 check inter 1s fall 2 rise 3 verify none
    server worker2.ocp4.g42cloud.com 172.20.0.87:80 check inter 1s fall 2 rise 3 verify none
    server worker3.ocp4.g42cloud.com 172.20.0.175:80 check inter 1s fall 2 rise 3 verify none
    server worker4.ocp4.g42cloud.com 172.20.0.136:80 check inter 1s fall 2 rise 3 verify none

frontend ocp_apps_443_server_frontend
    mode tcp
    bind :443
    default_backend ocp_machine_apps_443_backend
backend ocp_machine_apps_443_backend
    mode tcp
    balance roundrobin
    server worker1.ocp4.g42cloud.com 172.20.0.170:443 check check-ssl inter 1s fall 2 rise 3 verify none
    server worker2.ocp4.g42cloud.com 172.20.0.87:443 check check-ssl inter 1s fall 2 rise 3 verify none
    server worker3.ocp4.g42cloud.com 172.20.0.175:443 check check-ssl inter 1s fall 2 rise 3 verify none

    server worker4.ocp4.g42cloud.com 172.20.0.136:443 check
```

Notes: -

- check-ssl not required for port 80.
- Need to update the proper server IP

- Restart the HAProxy service on the HAProxy ECS node.

```
[centos@ecs-ocp4-bastion]$ systemctl restart haproxy
```

5.20. Configuring HAProxy on Two ECS Nodes with High Availability

This section provides steps to create HAproxy with High Availability. Navigate to “Elastic Compute Server” service from the main page of G42 Cloud Console and click on a button “Create ECS” in the right top corner. Fill all the fields and submit the request.

If configuring high availability for HAProxy nodes, create an ECS group named "haproxy" and add both HAProxy nodes to this group.

Table 5-42 -High availability HAProxy ECS parameters

Section	Example value
AZ	AZ1
Specification	c6s.4xlarge.2 (4 vCPU | 8GiB)
Public Image	CentOS 7.6 (100GB)
System Disk	Ultra-High IO (100 GB)
Quantity	2

Network	vpc-openshift/subnet-openshift
Security Group	sg-ocp4-api-lb
EIP	Do not use
ECS Name	ecs-ocp4-haproxy
Login Mode (Key pair)	*Choose your ECS Key pair*
Cloud Backup/Recovery	Do not use
Enterprise Project	default
Max./Assured Bandwidth (Gbit/s)	2/2
ECS Group (Optional)	Proxy

- Login to the HAProxy ECS nodes and install HAProxy software on both ECS nodes

```
[centos@ecs-ocp4-haproxy1]$ yum install haproxy -y
[centos@ecs-ocp4-haproxy1]$ systemctl enable –now haproxy
```

- Adjust the following SELinux setting and restart HAProxy

```
[centos@ecs-ocp4-haproxy1]$ setsebool -P haproxy_connect_any=1
[centos@ecs-ocp4-haproxy1]$ systemctl restart –now haproxy
```

- Configure HAProxy, you need to add DNS names and IP addresses of the bootstrap and control plane (control) nodes.

  Configure api service frontend and backend on port 6443/tcp and HTTPS health check for /readyz path.

  Configure machine configuration service frontend and backend on port 22623/tcp and HTTPS health check for /healthz path.

```
[centos@ecs-ocp4-haproxy1]$ vi /etc/haproxy/haproxy.cfg ...
#---------------------------------------------------------------------
# Kube API Server
#---------------------------------------------------------------------
frontend k8s_api_frontend
        bind :6443
        default_backend k8s_api_backend
        mode tcp
backend k8s_api_backend
        mode tcp
        balance     roundrobin
        option      httpchk GET /readyz HTTP/1.0
        server      bootstrap.ocp4.example.com 10.100.0.69:6443 check check-ssl inter 1s fall 2 rise 3 verify none
        server      master-1.ocp4.example.com 10.100.0.68:6443 check check-ssl inter 1s fall 2 rise 3 verify none
        server      master-2.ocp4.example.com 10.100.0.162:6443 check check-ssl inter 1s fall 2 rise 3 verify none
        server      master-3.ocp4.example.com 10.100.0.169:6443 check check-ssl inter 1s fall 2 rise 3 verify none
#---------------------------------------------------------------------
# OCP Machine Config Server
#---------------------------------------------------------------------
frontend ocp_machine_config_server_frontend
        bind :22623
        default_backend ocp_machine_config_server_backend
        mode tcp
backend ocp_machine_config_server_backend
        mode tcp
        balance     roundrobin
        option      httpchk GET /healthz  HTTP/1.0
        server      bootstrap.ocp4.example.com 10.100.0.69:22623 check check-ssl inter 1s fall 2 rise 3 verify none
        server      master-1.ocp4.example.com 10.100.0.68:22623 check check-ssl inter 1s fall 2 rise 3 verify none
        server      master-2.ocp4.example.com 10.100.0.162:22623 check check-ssl inter 1s fall 2 rise 3 verify none
        server      master-3.ocp4.example.com 10.100.0.169:22623 check check-ssl inter 1s fall 2 rise 3 verify none


#---------------------------------------------------------------------
# OCP Machine APPS80 Server
#---------------------------------------------------------------------

frontend ocp_apps_80_server_frontend
          mode tcp
          bind :80
          default_backend ocp_machine_apps_80_backend
backend ocp_machine_apps_80_backend
          mode tcp
          balance roundrobin
   server worker-1.ocp4.example.com 172.20.0.119:80 check inter 1s fall 2 rise 3 verify none
   server worker-2.ocp4.example.com 172.20.0.16:80 check  inter 1s fall 2 rise 3 verify none
   server worker-3.ocp4.example.com 172.20.0.202:80 check inter 1s fall 2 rise 3 verify none
#---------------------------------------------------------------------
# OCP Machine APPS443 Server
#---------------------------------------------------------------------

frontend ocp_apps_443_server_frontend
          mode tcp
          bind :443
          default_backend ocp_machine_apps_443_backend
backend ocp_machine_apps_443_backend
          mode tcp
          balance roundrobin
   server worker-1.ocp4.example.com 172.20.0.119:443 check check-ssl inter 1s fall 2 rise 3 verify none
   server worker-2.ocp4.example.com 172.20.0.16:443 check check-ssl inter 1s fall 2 rise 3 verify none
   server worker-3.ocp4.example.com 172.20.0.202:443 check check-ssl inter 1s fall 2 rise 3 verify none
...
```

- Create virtual IP, bind both the HAProxy ECS nodes NICs to virtual IP

Action	Official Guide Link
Create virtual IP	[Create Virtual IP Link]()
Bind virtual IP to ECS NIC	[Bind Virtual IP to ECS NIC Link]()

- Install keepalived on both HAProxy nodes

```
[root@ecs-ocp4-haproxy1]$ yum install -y keepalived
```

- Configure keepalived on both the HAProxy nodes
  You will need to configure one HAproxy ECS as primary and other node as backup along with different priority settings for each.
  On HAProxy node 1 use vi editor to configure keepalived with configuration as shown below

```
[root@ecs-ocp4-haproxy1]$ vi /etc/keepalived/keepalived.conf
vrrp_script
chk_haproxy {
        script \"killall -0 haproxy\" # check the haproxy process
        interval 2 # every 2 seconds weight 2 # add 2 points if
        OK
        }
vrrp_instance VI_1 {
        interface eth0 # interface to monitor
        state CONTROL # CONTROL on haproxy1, BACKUP on haproxy2
        virtual_router_id 51
        priority 101 # 101 on haproxy1, 100 on haproxy2
        virtual_ipaddress {<virtual IP address>  # virtual ip address  }
track_script {
        chk_haproxy
        }
}
[root@ecs-ocp4-haproxy1]$ systemctl enable keepalived
[root@ecs-ocp4-haproxy1]$ systemctl start keepalived
```

On HAProxy node 2 use vi editor to configure keepalived with configuration as shown below

```
[root@ecs-ocp4-haproxy2]$ vi /etc/keepalived/keepalived.conf
vrrp_script chk_haproxy {
  script \"killall -0 haproxy\" # check the haproxy
process   interval 2 # every 2 seconds   weight 2 # add
2 points if OK
}
vrrp_instance VI_1 {
  interface eth0 # interface to monitor
  state BACKUP # CONTROL on haproxy1, BACKUP on haproxy2
virtual_router_id 51
  priority 100 # 101 on haproxy1, 100 on haproxy2
virtual_ipaddress {
```

```
    <virtual IP address>  # virtual ip address
  }
```

systemctl start

keepalived

- Restart the HAProxy service on both HAProxy nodes.

```
[root@ecs-ocp4-haproxy1]$ systemctl restart haproxy
```

```
[root@ecs-ocp4-haproxy2]$ systemctl restart haproxy
```

5.21. Add Control Plane DNS Records

Navigate to “Domain Name Service” in the main page of G42 Cloud Console and select a private zone “example.com” that was created in section 4.14. Click on “+ Add RecordSet” button in the right top corner. Follow the below table to fill the required fields and submit the request.

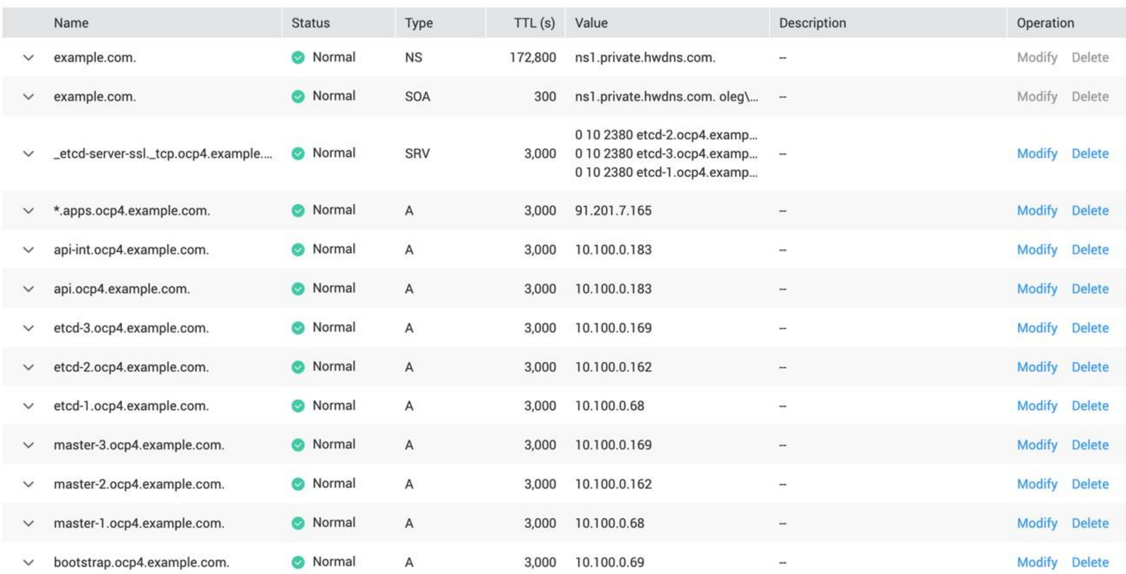
Table 5-43 - DNS record table

Name	Type	TTL	Value
bootstrap.ocp4	A	3000	IP Address of Bootstrap node
control-1.ocp4	A	3000	IP Address of Control-1 node
control-2.ocp4	A	3000	IP Address of Control-2 node
control-3.ocp4	A	3000	IP Address of Control-3 node
etcd-1.ocp4	A	3000	IP Address of Control-1 node
etcd-2.ocp4	A	3000	IP Address of Control-2 node
etcd-3.ocp4	A	3000	IP Address of Control-3 node
api.ocp4	A	3000	IP Address of HAProxy Node or API Virtual IP
api-int.ocp4	A	3000	IP Address of HAProxy Node or API Virtual IP

*.apps.ocp4	A	3000	IP Address of ELB (Internal)
_etcd-server-ssl._tcp.ocp4	SRV	3000	0 10 2380 etcd-2.ocp4.example.com. 0 10 2380 etcd-3.ocp4.example.com. 0 10 2380 etcd-1.ocp4.example.com.

The private DNS configuration is as shown in the figure below.

Figure 5-44 - DNS private zone configuration page



Name	Status	Type	TTL (s)	Value	Description	Operation
example.com.	Normal	NS	172,800	ns1.private.hwdns.com.	--	Modify Delete
example.com.	Normal	SOA	300	ns1.private.hwdns.com. oleg\...	--	Modify Delete
_etcd-server-ssl._tcp.ocp4.example....	Normal	SRV	3,000	0 10 2380 etcd-2.ocp4.examp... 0 10 2380 etcd-3.ocp4.examp... 0 10 2380 etcd-1.ocp4.examp...	--	Modify Delete
*.apps.ocp4.example.com.	Normal	A	3,000	91.201.7.165	--	Modify Delete
api-int.ocp4.example.com.	Normal	A	3,000	10.100.0.183	--	Modify Delete
api.ocp4.example.com.	Normal	A	3,000	10.100.0.183	--	Modify Delete
etcd-3.ocp4.example.com.	Normal	A	3,000	10.100.0.169	--	Modify Delete
etcd-2.ocp4.example.com.	Normal	A	3,000	10.100.0.162	--	Modify Delete
etcd-1.ocp4.example.com.	Normal	A	3,000	10.100.0.68	--	Modify Delete
master-3.ocp4.example.com.	Normal	A	3,000	10.100.0.169	--	Modify Delete
master-2.ocp4.example.com.	Normal	A	3,000	10.100.0.162	--	Modify Delete
master-1.ocp4.example.com.	Normal	A	3,000	10.100.0.68	--	Modify Delete
bootstrap.ocp4.example.com.	Normal	A	3,000	10.100.0.69	--	Modify Delete

5.22. Monitoring OpenShift Installation

Login to the installation/helper server and run the command as described below. Bootstrap machine boots by using an Ignition config pulled from the obs bucket (we uploaded the config to obs, generated presigned url and provided it as a user data to bootstrap ECS). The bootstrap machine assists in bringing up the control plane.

We will use openshift-install utility to monitor the installation process.

```
[root@ecs-ocp4-bastion ocp4]$ cd ~/ocp4
[root@ecs-ocp4-bastion ocp4]$ openshift-install wait-for bootstrap-complete
INFO Waiting up to 20m0s (until 12:00PM) for the Kubernetes API at
https://api.ocp4.g42cloud.com:6443...
INFO API v1.26.11+7dfc52e up
INFO Waiting up to 30m0s (until 12:10PM) for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 0s
```

If DNS entries and api load balancer are configured properly you would see API up info message.

You can also login to the nodes from the installation server via SSH by using the key that was generated in the previous steps and monitor the status of the pods.

5.23. Adding DNS Record for Compute Nodes

Navigate to "Domain Name Service" in the main page of G42 Cloud Console and select a private zone "example.com" that was created in section 4.14. Click on "+ Add RecordSet" button in the right top corner. Follow the below table to fill the required fields and submit the request.

Name	Type	TTL	Value
compute-1.ocp4	A	3000	IP Address of Compute-1 node
compute-2.ocp4	A	3000	IP Address of Compute-2 node
compute-3.ocp4	A	3000	IP Address of Compute-3 node
compute-4.ocp4	A	3000	IP Address of Compute-4 node

5.24. Monitoring the Bootstrapping Process

Login to the Bootstrap node from the bastion/helper node and perform the next command to monitor current activities of the bootstrapping.

```
$ journalctl -b -f -u release-image.service -u bootkube.service
```

5.25. Removing Bootstrap Node

As soon the installation node received the next message from the monitoring process that we started in a 4.21– we can remove the Bootstrap node from the installation process.

```
$ journalctl -b -f -u release-image.service -u bootkube.service
DEBUG Bootstrap status: complete
INFO It is now safe to remove the bootstrap resources DEBUG
Time elapsed per stage:
DEBUG Bootstrap Complete: 22m48s
DEBUG                API: 5m55s
INFO Time elapsed: 22m48s
```

5.25.1. Removing Bootstrap Node from HAProxy

Login to the HAProxy ECS node/s and comment the lines with a bootstrap node in haproxy config file at /etc/haproxy/haproxy.cfg. If you are using two nodes of HAProxy you will need to repeat this on the second ECS node.

Console 1 -HAProxy configuration

```
[centos@ecs-ocp4-haproxy1]$ vi
/etc/haproxy/haproxy.cfg ...
#-----------------------------------------------------------------
--- # Kube API Server
#---------------------------------------------------------------
----- frontend k8s_api_frontend     bind :6443
default_backend k8s_api_backend     mode tcp
 backend k8s_api_backend     mode tcp
balance     roundrobin     option
httpchk GET /readyz HTTP/1.0
#   server      bootstrap.ocp4.example.com 10.100.0.69:6443 check check-ssl inter 1s fall 2 rise 3
verify none     server      master-1.ocp4.example.com 10.100.0.68:6443 check check-ssl inter 1s fall
2 rise 3 verify none     server      master-2.ocp4.example.com 10.100.0.162:6443 check check-ssl
inter 1s fall 2 rise 3 verify none server      master-3.ocp4.example.com 10.100.0.169:6443 check
check-ssl inter 1s fall 2 rise 3 verify
none
#-----------------------------------------------------------------
--- # OCP Machine Config Server
#---------------------------------------------------------------
----- frontend ocp_machine_config_server_frontend     bind :22623
default_backend ocp_machine_config_server_backend     mode tcp
 backend
ocp_machine_config_server_backend
mode tcp     balance     roundrobin
option      httpchk GET /healthz
HTTP/1.0
#    server       bootstrap.ocp4.example.com 10.100.0.69:22623 check check-ssl inter 1s fall 2 rise 3
verify none     server      master-1.ocp4.example.com 10.100.0.68:22623 check check-ssl inter 1s fall
2 rise 3 verify none     server      master-2.ocp4.example.com 10.100.0.162:22623 check check-ssl inter
1s fall 2 rise 3 verify none     server      master-3.ocp4.example.com 10.100.0.169:22623 check check-
ssl inter 1s fall 2 rise 3 verify none
...
```

Restart HAProxy service

```
[root@ecs-ocp4-bastion]$ sudo systemctl restart haproxy
```

5.25.2. Removing Bootstrap DNS Record

Navigate to “Domain Name Service” in the main page of G42 Cloud Console and select the private zone “example.com” that was created in section 4.14. Find the record “bootstrap.ocp4.example.com” and click on “Delete” button opposite the record on the right side of the window.

5.25.3. Removing Bootstrap Node

Navigate to “Elastic Compute Server” in the main page of G42 Cloud Console and find the Bootstrap ECS that was created in section 0. Click on “Delete” button opposite the server on the right side of the window and submit.

5.26. Approving Pending Certificate Signing Requests

In order to allow compute nodes to join the cluster we will need to manually approve the certificate signing request (CSR).

Login to the installation/helper server and approve all current CSRs that in a pending state.

```
[centos@ecs-ocp4-bastion ocp4]$ export KUBECONFIG=~/ocp4/auth/kubeconfig
[centos@ecs-ocp4-bastion ocp4]$ oc get csr
NAME        AGE   SIGNERNAME                            CONDITION csr-6bgt4   1s    kubernetes.io/kubelet-serving
system:node:host-10-100-0-225  Pending                        csr-g57m8     5s      kubernetes.io/kubelet-serving
system:node:host-10-100-0-134 Pending                            csr-sxx2f    3s     kubernetes.io/kubelet-serving
system:node:host-10-100-0-249 Pending                            csr-zp9fv    2s     kubernetes.io/kubelet-serving
system:node:host-10-100-0-140 Pending
[centos@ecs-ocp4-bastion ocp4]$ oc get csr -o go-template='{{range .items}}{{if not
.status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
certificatesigningrequest.certificates.k8s.io/csr-6bgt4 approved
certificatesigningrequest.certificates.k8s.io/csr-g57m8 approved
certificatesigningrequest.certificates.k8s.io/csr-sxx2f approved
certificatesigningrequest.certificates.k8s.io/csr-zp9fv approved
```

You might need to do it several times as new CSRs might come in seconds.

5.27. Finishing Installation of OpenShift

To finish the installation process, run the following command on the installation/helper server.

```
[centos@ecs-ocp4-bastion ocp4]$ openshift-install wait-for install-complete
INFO Waiting up to 40m0s (until 9:48PM) for the cluster at https://api.ocp4.example.com:6443 to
initialize...
```

This process will take approximately up to 30 minutes. After the installation completes you should receive the following message.

```
INFO Waiting up to 10m0s (until 9:21PM) for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run INFO
export KUBECONFIG=/home/centos/ocp4/auth/kubeconfig
INFO Access the OpenShift web-console here:
INFO https://console-openshift-console.apps.ocp4.example.com
INFO Login to the console with user: "kubeadmin", and password: "r45jd-9qnWy-avcQI-PkpgQ" INFO
Time elapsed: 3m1s
```

Congratulations! your installation of Red Hat OpenShift Container Platform is now complete.

```
[centos@ecs-ocp4-bastion ~]$ export KUBECONFIG=/home/root/ocp4/auth/kubeconfig
[centos@ecs-ocp4-bastion ~]$ oc get nodes
NAME                STATUS   ROLES    AGE   VERSION host-10-
100-0-134   Ready    worker   13m   v1.24.0+4f0dd4d host-10-
100-0-140   Ready    worker   13m   v1.24.0+4f0dd4d host-10-
100-0-162   Ready    master   31m   v1.24.0+4f0dd4d host-10-
100-0-169   Ready    master   30m   v1.24.0+4f0dd4d host-10-
100-0-225   Ready    worker   13m   v1.24.0+4f0dd4d host-10-
100-0-249   Ready    worker   13m   v1.24.0+4f0dd4d host-10-
100-0-68     Ready    master   31m   v1.24.0+4f0dd4d
```

Now, you can login to the cluster via a web-user interface which will be available at the link bellow.
https://console-openshift-console.apps.ocp4.example.com/

This domain must be accessible from your workstation. There are two options

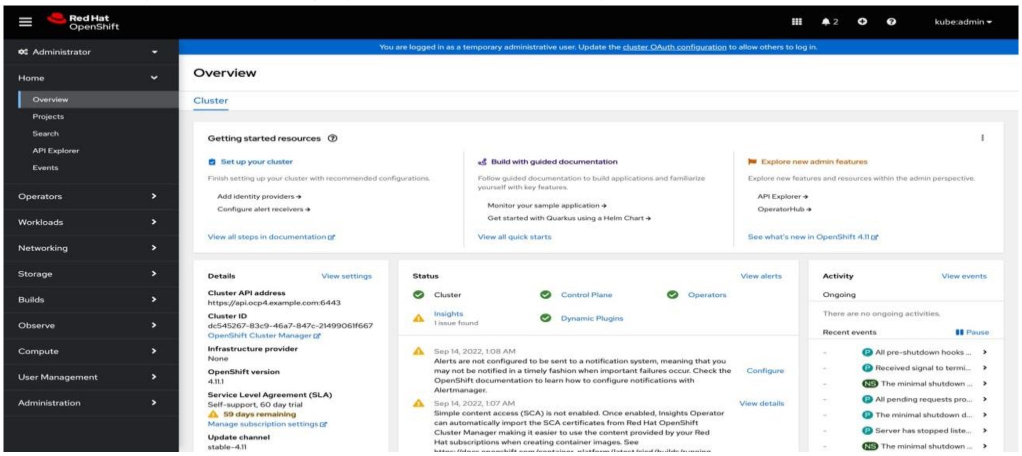
1. Contact your System Administrators who is responsible for corporate Domain Services and ask them to add the following record.

Domain name	IP address
*apps.ocp4.example.com	91.201.7.30

2. Add the next records into "/etc/hosts" file if you use a Linux/MacOS.

```
91.201.7.30 api.ocp4.example.com             console-openshift-
console.apps.ocp4.example.com              oauth-
openshift.apps.ocp4.example.com              downloads-openshift-
console.apps.ocp4.example.com              alertmanager-main-openshift-
monitoring.apps.ocp4.example.com              grafana-openshift-
monitoring.apps.ocp4.example.com              prometheus-k8s-openshift-
monitoring.apps.ocp4.example.com              thanos-querier-openshift-
monitoring.apps.ocp4.example.com
```

As a result, you can manage your cluster via the OpenShift Web-Console.



5.28. Configuring OBS Storage for Image Registry

As an additional step in this guide, it is required to configure storage for our Red Hat OpenShift Container Platform to store images and update the cluster. By default, a Bare Metal cluster does not provide with storage feature. In this deployment, OBS bucket "obs-image-registry" (see section 4.7) will be used as a primary storage for Image Registry.

Login to the installation/helper node and use AK/SK credentials created in section 4.3 to create a secret for OBS bucket authentication.

```
[centos@ecs-ocp4-bastion ~]$ export ACCESS="replace with your AK"
[centos@ecs-ocp4-bastion ~]$ export SECRET="replace with your SK"
[centos@ecs-ocp4-bastion ~]$ oc create secret generic image-registry-private-configuration-user --from-literal REGISTRY_STORAGE_S3_ACCESSKEY=$ACCESS --from-literal REGISTRY_STORAGE_S3_SECRETKEY=$SECRET --namespace openshift-image-registry
secret/image-registry-private-configuration-user created
```

Add a new storage configuration for Image Registry operator. In this configuration we will specify the parameters as described in table below

Table 5-45 - Values for image registry operator

Section	Example value
Bucket	obs-image-registry
Region Endpoint	https://obs.ae-ad-1.vb.g42cloud.com
Region	ae-ad-1

```
[centos@ecs-ocp4-bastion ~]$oc patch configs.imageregistry.operator.openshift.io/cluster --type='json' --patch='[{"op": "remove", "path": "/spec/storage" },{"op": "add", "path":"/spec/storage","value":{"s3":{"bucket":"obs-image-registry","regionEndpoint": "https://obs.ae-ad-1.vb.g42cloud.com","encrypt": false,"region": "ae-ad-1"}}}]'
config.imageregistry.operator.openshift.io/cluster patched
```

Update the Image Registry operator config by changing the management state to Managed

```
[centos@ecs-ocp4-bastion ocp4]$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"managementState":"Managed"}}'
config.imageregistry.operator.openshift.io/cluster patched
```

Check secret created and able to encrypted password.

```
[centos@ecs-ocp4-bastion ocp4]$ oc get secret image-registry-private-configuration-user  -o yaml |head -5
apiVersion: v1
data:
  REGISTRY_STORAGE_S3_ACCESSKEY: XXX=
  REGISTRY_STORAGE_S3_SECRETKEY: XXXXXX==
```

5.29. Upgrading Cluster to Latest Version

Check the current version of the cluster.

```
[centos@ecs-ocp4-bastion ~]$ oc get clusterversion
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.13.0    True        False         68m     Cluster version is 4.13.0
```

Check for any available updates.

```
[centos@ecs-ocp4-bastion ~]$ oc adm upgrade
Cluster version is 4.13.0

Upstream is unset, so the cluster will use an appropriate default.
Channel: stable-4.13

Recommended updates:

  VERSION     IMAGE
  4.13.2      quay.io/openshift-release-dev/ocp-
release@sha256:cf62f95558b5e555a32d5c40f4fbcea6c800624188c52657d13d0f4ef121d07d
  4.13.1      quay.io/openshift-release-dev/ocp-
release@sha256:e8f7c211321ec4cd4098f6cb9b69abb915da107a587dbb5a7b12aac14ec0b2bc

Additional updates which are not recommended based on your cluster configuration are available, to view
those re-run the command with --include-not-recommended.
```

Update the cluster to the latest available version of Red Hat OpenShift Container Platform.

```
[centos@ecs-ocp4-bastion ~]$ oc adm upgrade --to 4.13.2
Requesting update to 4.1
[centos@ecs-ocp4-bastion ~]$ oc get clusterversion
NAME    VERSION  AVAILABLE  PROGRESSING  SINCE  STATUS
version  4.13.0   True       True        2s     Working towards 4.13.2 14 of 803 done (1% complete)
```

6. Add ingress certificate (Optional)

By default, OpenShift Container Platform uses the Ingress Operator to create an internal CA and issue a wildcard certificate that is valid for applications under the .apps sub-domain. Both the web console and CLI use this certificate as well.
This is the steps explaining that how to add the ingress certificate in openshift cluster for *.apps.<clusternae>.domain.

Prerequisites: -

- You must have a wildcard certificate for the fully qualified .apps subdomain and its corresponding private key. Each should be in a separate PEM format file.
- The private key must be unencrypted. If your key is encrypted, decrypt it before importing it into OpenShift Container Platform.
- The certificate must include the subjectAltName extension showing *.apps.<clustername>.<domain>.
- The certificate file can contain one or more certificates in a chain. The wildcard certificate must be the first certificate in the file. It can then be followed with any intermediate certificates, and the file should end with the root CA certificate.
- Copy the root CA certificate into an additional PEM format file.

6.1. Generate a self-signed certificate and obtain signature from a certificate authority (CA):

Access the bastion host and create a self-signed certificate. Then, proceed with obtaining a signature from a certificate authority after completing the following steps.

```
openssl req -new -key ingress.key -out ingress.csr
openssl x509 -req -days 365 -in ingress.csr -signkey ingress.key -out ingress.crt
openssl x509 -in ingress.crt -text -noout
```

6.2. Update the certificate in OCP cluster:

After sigh the certificate, the CA-Bundle need to be added in additional TrustBundle at the time of installation, or after installation as per below :

Create a config map that includes only the root CA certificate used to sign the wildcard certificate and Update the cluster-wide proxy configuration with the newly created config map (if any quay certificate already using the cluster, then need to add this certificate after the quay.)

```
oc create configmap custom-ca --from-file=ca-bundle.crt=</path/to/example-ca.crt>  -n openshift-config
oc patch proxy/cluster --type=merge --patch='{"spec":{"trustedCA":{"name":"custom-ca"}}}'
```

Create a secret that contains the wildcard certificate chain and key and Create a secret that contains the wildcard certificate chain and key.

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

7. Summary

In this guide, we have deployed and configured a basic Red Hat OpenShift Container Platform cluster with a standard and recommended layout which consists of three control planes and compute nodes. All the described deployment procedure was manual and requires some time to prepare and complete. This installation successfully uses a variety of native G42 Cloud services like ECS, EVS, IMS, VPC, ELB, NAT, DNS, OBS and can be integrated with many others as per requirements. In order to effective use of the Red Hat and G42 Cloud products please refer to the official documentation pages.

8. Frequently Asked Questions (FAQs)

Q. Who will deploy and manage Red Hat OpenShift?

Ans. The deployment and management can be done by customer or their partners using the deployment guide as a guiding framework.

Q. Why do we need HAProxy for API load-balancer?

Ans. G42 native dedicated ELB cannot be used as API load-balancer due to certain ELB feature and its impact on API load-balancing. G42 Dedicated ELB can be used for ingress load-balancer, this deployment guide uses HAproxy

Q. Can HAProxy be configured in HA?

Ans. Yes, HA proxy can be configured in HA using virtual IP and using a tool like keepalived for heartbeat between instances. You can refer the section 4.19 for details.

Q. How is HA achieved on control-nodes?

Ans. G42 Cloud natively provides VM affinity using server groups, it is strongly advised to user server groups for compute nodes and control nodes to ensure that ECS are placed in separate host within same AZ to provide host level resiliency.

Q. Why is the cloud instance size important during the G42 Deployment and how is it correlated to its network configuration and OpenStack QoS?

Ans. The size of the cloud instance determines the allocated network bandwidth. For example, General Proposed VMs start with a default configuration where NICs are assigned to QoS configurations with Assured and Max Bandwidth limits. If the Assured bandwidth is less than the Max bandwidth, there's a higher chance of OpenStack QoS throttling the network flow, causing delays and latency. Conversely, when Assured equals Max bandwidth, as with Disk Intensive ECS, there's less chance of throttling, reducing the risk of issues. For instance, the current d3.2xlarge.8 instance type has a configured Max and Assured Bandwidth of 5 Gbit.