Reference Architectures 2017

Deploying and Managing OpenShift Container Platform 3 on Google Cloud Platform

Last Updated: 2017-11-08

Reference Architectures 2017 Deploying and Managing OpenShift Container Platform 3 on Google Cloud Platform

Chris Murphy

Peter Schiffer
refarch-feedback@redhat.com

Legal Notice

Abstract

The purpose of this document is to provide guidelines and considerations for installing and configuring Red Hat OpenShift Container Platform on Google Cloud Platform.

Table of Contents

COMMENTS AND FEEDBACK

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architecture. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers. Feedback on the papers can be provided by emailing refarch-feedback@redhat.com. Please refer to the title within the email.

CHAPTER 1. EXECUTIVE SUMMARY

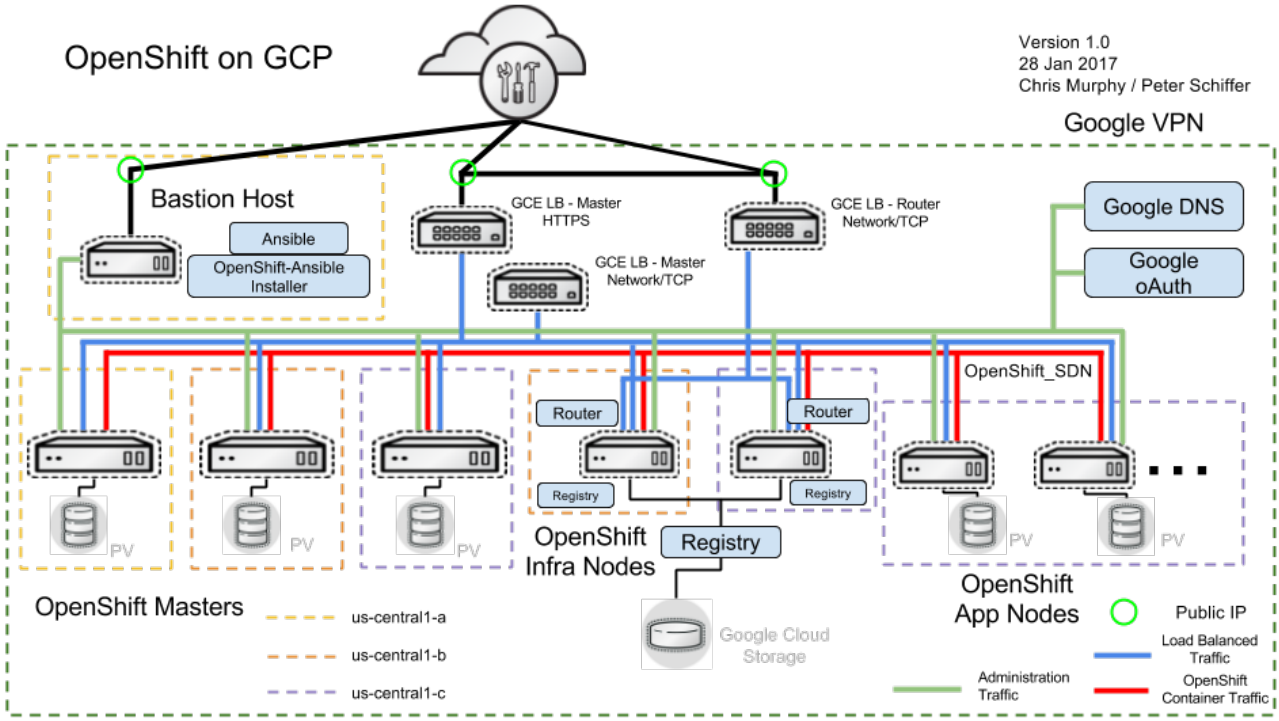
Red Hat® OpenShift Container Platform 3 is built around a core of application containers, with orchestration and management provided by Kubernetes, on a foundation of Atomic Host and Red Hat Enterprise Linux. OpenShift Origin is the upstream community project that brings it all together along with extensions, to accelerate application development and deployment.

This reference environment provides a comprehensive example demonstrating how OpenShift Container Platform 3 can be set up to take advantage of the native high availability capabilities of Kubernetes and Google Cloud Platform in order to create a highly available OpenShift Container Platform 3 environment. The configuration consists of three OpenShift Container Platform masters, three OpenShift Container Platform infrastructure nodes, three OpenShift Container Platform application nodes and native Google Cloud Platform integration in a multi zone environment. In addition to the configuration, operational management tasks are shown to demonstrate functionality.

CHAPTER 2. COMPONENTS AND CONSIDERATIONS

This chapter describes the highly available OpenShift Container Platform 3 reference architecture environment on Google Cloud Platform (GCP) that can be deployed.

The image below provides a high-level representation of the components within this reference architecture. Using Google Cloud Platform (GCP), resources are highly available using a combination of multiple **zones**, Load Balancers, and a Google Cloud Storage (GCS). Instances deployed are given specific roles to support OpenShift. The Bastion host limits the external access to internal servers by ensuring that all **SSH** traffic passes through the Bastion host. The master instances host the OpenShift master components such as etcd and the OpenShift API. The Application instances are for users to deploy their containers while the Infrastructure instances are used for the OpenShift router and registry. Authentication is managed by Google OAuth. OpenShift on **GCP** has two cloud native storage options; Solid-State Persistent disks are used for the filesystem of instances but can also be used for persistent storage in containers. The other storage option is Google Cloud Storage (GCS) which is object based storage. GCS is used for the persistent storage of the OpenShift registry. The network is configured to leverage three Load Balancers for access to the OpenShift API, OpenShift console, and the OpenShift routers. The first Load Balancer is for the OpenShift API and console access originating from outside of the cluster. The second Load Balancer is for API access within the cluster. The third Load Balancer is for accessing services deployed in the cluster that have been exposed through routes. Finally, the image shows that DNS is handled by Google's **`Cloud DNS`**.



This reference architecture breaks down the deployment into separate phases.

- Phase 1: Provision the infrastructure on **GCP**
- Phase 2: Provision OpenShift Container Platform on **GCP**
- Phase 3: Post deployment activities

For Phase 1, the provisioning of the environment is done using a series of Ansible playbooks that are provided in the openshift-ansible-contrib GitHub repo. Once the infrastructure is deployed the playbooks will flow automatically into Phase 2. Phase 2 is the installation of OpenShift Container Platform which is done via Ansible playbooks as well. These playbooks are installed by the

openshift-ansible-playbooks RPM package. The playbooks in **openshift-ansible-contrib** utilize the playbooks provided by the **openshift-ansible-playbooks** RPM package to perform the installation of OpenShift and also to configure **GCP** specific parameters. During Phase 2 the router and registry are deployed. The last phase, Phase 3, concludes the deployment by confirming the environment was deployed properly. This is done by running the systems engineering teams **validation** Ansible playbook and OpenShift client command line tools.

NOTE

The deployment scripts provided in the GitHub repo are not supported by Red Hat but the subsequent OpenShift deployment is supported by Red Hat. The scripts merely provide a mechanism that can be used to build out your own infrastructure.

2.1. GOOGLE CLOUD PLATFORM

This section will give a brief overview of some of the common Google Cloud Platform products that are used in this reference implementation.

2.1.1. Google Compute Engine

Per Google; "Google Compute Engine delivers virtual machines running in Google's innovative data centers and worldwide fiber network. Compute Engine's tooling and workflow support enable scaling from single instances to global, load-balanced cloud computing."

Within this reference environment, the instances are deployed in multiple **Regions and Zones** in the **us-central1-a** zone as the default. However, the default region can be changed during the deployment portion of the reference architecture to a zone closer to the user's geo-location. The master instances for the Container Platform environment are machine type **n1-standard-4**. The node instances are machine type **n1-standard-2** and are provided two extra disks, which are used for Docker storage and OpenShift Container Platform ephemeral volumes. The bastion host is deployed as machine type **g1-small**. Instance sizing can be changed in the variable file for the installer, which is covered in a later chapter.

For more on sizing and choosing zones and regions, see https://cloud.google.com/compute

2.1.2. Solid-state persistent disks

Google provides persistent block storage for your virtual machines as both standard (HDD) and solid-state (SSD) disks. This reference implementation will be utilizing **Solid-State Persistent disks** as they provide better performance for random input/output operations per second.

Google's persistent storage has the following attributes:

- Snapshotting capabilities
- Redundant, industry-standard data protection
- All data written to disk is encrypted on the fly and stored in encrypted form

Within this reference environment, the node instances for the OpenShift Container Platform environment are provisioned with an extra 25GB **Solid-State Persistent disk** for Docker storage and a 50GB disk of the same type used for OpenShift Container Platform ephemeral volumes. Disk sizing can be changed in the variable file for the installer, which is covered in a later chapter.

For more information see https://cloud.google.com/compute/docs/disks

2.1.3. Custom Images and Instance Templates

Google provides images of many popular Operating Systems including Red Hat and CentOS, which can be used to deploy infrastructure. They also provide the ability to upload and create custom images, which has the added appeal of being pre-configured with common packages and services common for every OpenShift Container Platform master and node instance. This feature speeds up the couple of initial steps needed for each instance, thus improving the overall deployment time. For this reference architecture a image provided by Red Hat will be uploaded and pre-configured for use as the base image for all OpenShift Container Platform instances.

This reference architecture will also take advantage of GCE's instance template feature. This allows for all the required information (such as a machine type, disk image, network, and zone) to be turned into templates for each the OpenShift Container Platform instance types. These custom instance templates are then used to successfully launch both OpenShift Container Platform masters and nodes.

For more information see https://cloud.google.com/compute/docs/creating-custom-image and https://cloud.google.com/compute/docs/instance-templates

2.1.4. Regions and Zones

GCP has a global infrastructure that covers 13 regions and 39 availability zones. Per Google; "Certain Compute Engine resources live in regions or zones. A region is a specific geographical location where you can run your resources. Each region has one or more zones. For example, the us-central1 region denotes a region in the Central United States that has zones us-central1-a, us-central1-b, us-central1-c and us-central1-f."

For this reference architecture, we will default to the "us-central1" region. Minimum requirement for the region is that it has to have at least 3 available zones. To check which region and zone your user is set to use, the following command will provide the answer:

```
$ gcloud config configurations list
NAME    IS_ACTIVE ACCOUNT             PROJECT     DEFAULT_ZONE
DEFAULT_REGION
default True      chmurphy@redhat.com ose-refarch us-central1-a us-
central1
```

In the next chapter, we will go deeper into regions and zones and discuss how to set them using the gcloud tool as well as in the variables file for the infrastructure setup.

For more information see https://cloud.google.com/compute/docs/zones

2.2. GOOGLE CLOUD PLATFORM NETWORKING

GCP Networking allows you to configure custom virtual networking components including IP address ranges, gateways, route tables and firewalls.

2.2.1. Load Balancing and Scaling

A Load Balancing service enables distribution of traffic across multiple instances in the **GCE** cloud. There are clear advantages to doing this:

- Keeps your applications available

- Supports Compute Engine Autoscaler, which allows users to perform autoscaling on groups of instances

GCP provides five kinds of Load Balancing, HTTP(S), SSL Proxy, TCP Proxy, Network based and Internal. This implementation guide will use HTTP(S) and Network Load Balancing.

For more information see https://cloud.google.com/compute/docs/load-balancing-and-autoscaling

GCE LB implements parts of the Section 2.12, “Routing Layer”.

2.2.1.1. Load Balancers Details

Three load balancers are used in the reference environment. The table below describes the load balancer **`Cloud DNS`** name, the instances in which the load balancer is attached, and the port(s) monitored by the **`health check`** attached to each load balancer to determine whether an instance is in or out of service.

Table 2.1. Load Balancers

Load Balancers	Assigned Instances	Port
master.gce.sysdeseng.com	ocp-master-*	443
internal-master.gce.sysdeseng.com	ocp-master-*	443
*.apps.gce.sysdeseng.com	ocp-infra-node-*	80 and 443

Both the **`internal-master`**, and the **`master`** load balancers utilize the OpenShift Container Platform Master API port for communication. The **`internal-master`** load balancer is used for internal cluster communication with the API. The **`master`** load balancer is used for external traffic to access the OpenShift Container Platform environment through the API or the web interface. The "*.apps" load balancer allows public traffic to reach the infrastructure nodes. The infrastructure nodes run the router pod, which then directs traffic directly from the outside world into OpenShift Container Platform pods with external routes defined.

2.2.1.2. Firewall Details

According to the specific needs of OpenShift services, only the following ports are open in the network firewall.

Table 2.2. ICMP and SSH Firewall Rules

Port	Protocol	Service	Source	Target
	ICMP		Internet	All nodes
22	TCP	SSH	Internet	Bastion
22	TCP	SSH	Bastion	All internal nodes

Table 2.3. Master Nodes Firewall Rules

Port	Protocol	Service	Source	Target
2379, 2380	TCP	etcd	Master nodes	Master nodes
8053	TCP, UDP	SkyDNS	All internal nodes	Master nodes
443	TCP	Web Console, API	Internet	Master nodes

Table 2.4. App and Infra Nodes Firewall Rules

Port	Protocol	Service	Source	Target
4789	UDP	SDN	Master, App and Infra Nodes	Master, App and Infra Nodes
10250	TCP	Kubelet	Master and Infra Nodes	Master, App and Infra Nodes
5000	TCP	Registry	App and Infra Nodes	Infra Nodes
80, 443	TCP	HTTP(S)	Internet	Infra Nodes

2.3. CLOUD DNS

DNS is an integral part of a successful OpenShift Container Platform deployment/environment. **GCP** has the Google Cloud DNS web service; per Google: "Google Cloud DNS is a scalable, reliable and managed authoritative Domain Name System (DNS) service running on the same infrastructure as Google. It has low latency, high availability and is a cost-effective way to make your applications and services available to your users. Cloud DNS translates requests for domain names like www.google.com into IP addresses like 74.125.29.101."

OpenShift Container Platform requires a properly configured wildcard **DNS** zone that resolves to the IP address of the OpenShift Container Platform router. For more information, please refer to the OpenShift Container Platform DNS. In this reference architecture Cloud DNS will manage **DNS** records for the OpenShift Container Platform environment.

For more information see https://cloud.google.com/dns

2.3.1. Public Zone

The Public Cloud DNS zone requires a domain name either purchased through Google's "Domains" service or through one of many 3rd party providers. Once the zone is created in **Cloud DNS**, the name servers provided by Google will need to be added to the registrar.

For more information see https://domains.google

2.4. CLOUD STORAGE AND CONTAINER REGISTRY

GCP provides object cloud storage which can be used by OpenShift Container Platform 3.

OpenShift Container Platform can build Docker images from your source code, deploy them, and manage their life-cycle. To enable this, OpenShift Container Platform provides an internal, integrated Docker registry that can be deployed in your OpenShift Container Platform environment to manage images.

The registry stores Docker images and metadata. For production environment, you should use persistent storage for the registry, otherwise any images anyone has built or pushed into the registry would disappear if the pod were to restart.

Using the installation methods described in this document, the registry is deployed using a Google Cloud Storage (GCS) bucket. The **GCS** bucket allows for multiple pods to be deployed at once for HA but also use the same persistent backend storage. **GCS** is object based storage, which does not get assigned to nodes in the same way that Persistent Disks are attached and assigned to a node. The bucket does not mount as block based storage to the node so commands like ***`fdisk`*** or ***`lsblk`*** will not show information in regards to the **GCS** bucket. The configuration for the **GCS** bucket and credentials to login to the bucket are stored as OpenShift Container Platform secrets and applied to the pod. The registry can be scaled to many pods and even have multiple instances of the registry running on the same host due to the use of **GCS**.

For more information see https://cloud.google.com/storage

2.5. CLOUD IDENTITY AND ACCESS MANAGEMENT

GCP provides **`IAM`** to securely control access to the GCP services and resources for your users. In this guide, IAM provides security for our admins creating the cluster.

The deployment of OpenShift Container Platform requires a user that has the proper permissions by the **GCP** IAM administrator. The user must be able to create service accounts, cloud storage, instances, images, templates, Cloud DNS entries, and deploy load balancers and health checks. It is helpful to have delete permissions in order to be able to redeploy the environment while testing.

For more information see https://cloud.google.com/iam

2.6. SOFTWARE VERSION DETAILS

The following tables provide the installed software versions for the different servers that make up Red Hat OpenShift Container Platform highly available reference environment.

Table 2.5. OpenShift Container Platform 3 Details

Software	Version
Red Hat Enterprise Linux 7.4 x86_64	kernel-3.10.0-x
Atomic-OpenShift{master/clients/node/sdn-ovs/utils}	3.6.x.x
Docker	1.12.x
Ansible	2.3.x

2.7. REQUIRED CHANNELS

A subscription to the following channels is required in order to deploy this reference environment's configuration.

Table 2.6. Required Channels - OpenShift Container Platform 3 Master and Node Instances

Channel	Repository Name
Red Hat Enterprise Linux 7 Server (RPMs)	rhel-7-server-rpms
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	rhel-7-server-extras-rpms
Red Hat Enterprise Linux Fast Datapath (RHEL 7 Server) (RPMs)	rhel-7-fast-datapath-rpms
Red Hat OpenShift Container Platform 3.6 (RPMs)	rhel-7-server-ose-3.6-rpms

2.8. DYNAMIC INVENTORY

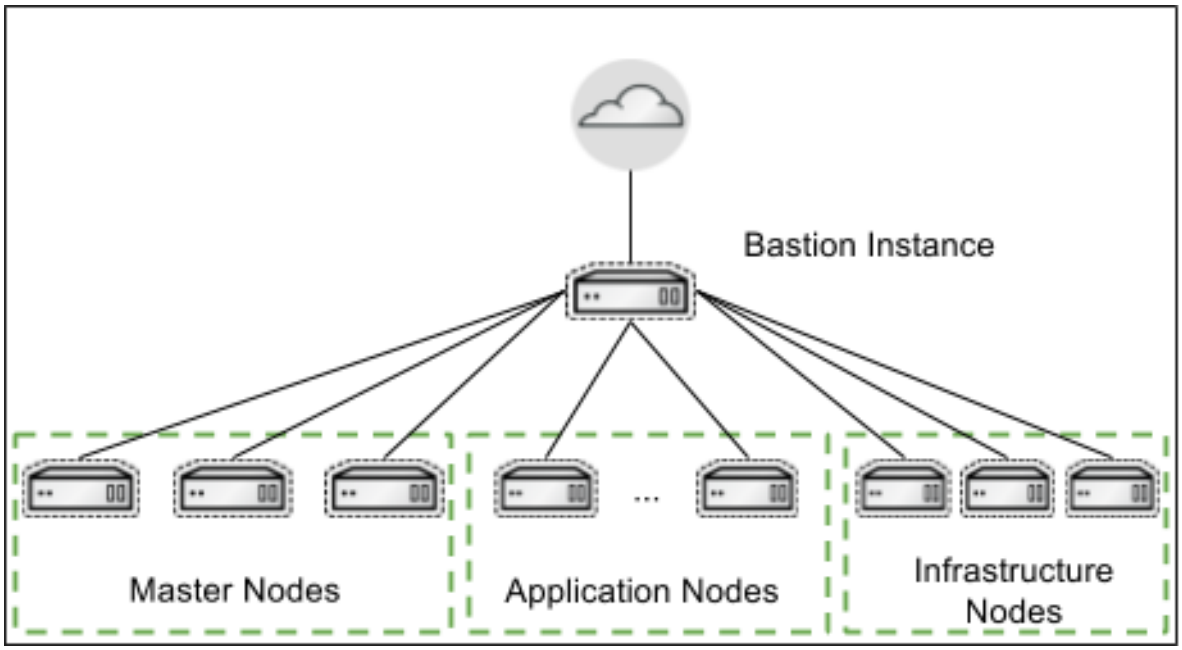
Ansible relies on inventory files and variables to perform playbook runs. As part of the reference architecture provided Ansible playbooks, the inventory is created automatically using a dynamic inventory script. The dynamic inventory script provided queries the Google Compute API to display information about **GCE** instances. The dynamic inventory script is also referred to as an Ansible Inventory script and the **GCE** specific script is written in python. The script can manually be executed to provide information about the environment but for this reference architecture, it is automatically called to generate the Ansible Inventory. For the OpenShift installation, the python script and the Ansible module **`add_host`** allow for instances to be grouped based on their purpose to be used in later playbooks. The reason the instances can be grouped is because during Phase 1 when the infrastructure was provisioned tags were applied to each instance. The masters were assigned the **`[prefix]-master`** tag, the infrastructure nodes were assigned the **`[prefix]-infra-node`** tag, and the application nodes were assigned the **`[prefix]-node`** tag.

For more information see http://docs.ansible.com/ansible/intro_dynamic_inventory.html

2.9. BASTION

As shown in the Figure 2.1, "Bastion Diagram" the **`bastion`** server in this reference architecture provides a secure way to limit **SSH** access to the **GCE** environment. The master and node firewall rules only allow for **SSH** connectivity between bastion and nodes inside of the local network while the bastion allows **SSH** access from everywhere. The bastion host is the only ingress point for **SSH** traffic in the cluster from external entities. When connecting to the OpenShift Container Platform infrastructure, the bastion forwards the request to the appropriate server. Connecting through the bastion server requires specific **SSH** configuration, which is configured by the **`ocp-on-gcp.sh`** installation script.

Figure 2.1. Bastion Diagram



2.10. NODES

Nodes are **GCE** instances that serve a specific purpose for OpenShift Container Platform. OpenShift Container Platform masters are also considered nodes. Nodes deployed on **GCE** can be vertically scaled up after the OpenShift Container Platform installation using the **ocp-on-gcp.sh** installation script. All OpenShift-specific nodes are assigned an IAM role, which allows for cloud-specific tasks to occur against the environment such as adding persistent volumes or removing a node from the OpenShift Container Platform cluster automatically. There are three types of nodes as described below.

2.10.1. Master nodes

The master nodes contain the master components, including the API server, controller manager server and ETCD. The master maintains the clusters configuration, manages nodes in its OpenShift Container Platform cluster. The master assigns pods to nodes and synchronizes pod information with service configuration. The master is used to define routes, services, and volume claims for pods deployed within the OpenShift Container Platform environment.

For a highly available OpenShift deployment deploy at least 3 masters behind a load balancer.

2.10.2. Infrastructure nodes

The infrastructure nodes are used for the router and registry pods. These nodes could be used if the optional components like Kibana and Hawkular metrics are required. The storage for the Docker registry that is deployed on the infrastructure nodes is backed by a **GCS** bucket, which allows for multiple pods to use the same storage. **GCS** storage is used because it is synchronized between the availability zones, providing data redundancy.

To ensure the least impact when performing upgrades of OpenShift deploy at least 3 infrastructure nodes. The router and registry components must be running on all infrastructure nodes to have the least chance of downtime during the upgrade.

2.10.3. Application Nodes

The Application nodes are the instances where non-infrastructure based containers run. Depending on the application, **GCE** specific storage can be applied such as a **`Block Storage`** which can be assigned using a **`Persistent Volume Claim`** for application data that needs to persist between container restarts. A configuration parameter is set on the master which ensures that OpenShift Container Platform user containers will be placed on the application nodes by default.

2.10.4. Node labels

All OpenShift Container Platform nodes are assigned a label. This allows certain pods to be deployed on specific nodes. For example, nodes labeled **`infra`** are Infrastructure nodes. These nodes run the router and registry pods. Nodes with the label **`app`** are nodes used for end user Application pods. The configuration parameter **`'defaultNodeSelector: "role=app"`** in **`/etc/origin/master/master-config.yaml`** ensures all projects automatically are deployed on Application nodes.

Node labels can also allow for segmentation of the OpenShift environment. If an organization would like to run development applications on a series of instances and production on another set of instances node labels can be applied to those instances to ensure that applications for development are launched on the development nodes and the production nodes only run those applications that have the label for production. This segmentation also shines with the multitennant SDN plugin which isolates pods and services unless explicitly specified.

2.11. OPENSHIFT CONTAINER PLATFORM PODS

OpenShift Container Platform leverages the Kubernetes concept of a pod, which is one or more containers deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed. For example, a pod could be just a single php application connecting to a database outside of the OpenShift Container Platform environment or a pod could be a php application that has an ephemeral database. OpenShift Container Platform pods have the ability to be scaled at runtime or at the time of launch using the OpenShift Container Platform console or the **`oc`** CLI tool. Any container running in the environment is considered to be a part of the pod. The pods containing the OpenShift Container Platform router and registry are required to be deployed in the OpenShift Container Platform environment.

2.12. ROUTING LAYER

Pods inside of an OpenShift Container Platform cluster are only reachable via their IP addresses on the cluster network. An edge load balancer can be used to accept traffic from outside networks and proxy the traffic to pods inside the OpenShift Container Platform cluster.

An OpenShift Container Platform administrator can deploy routers in an OpenShift Container Platform cluster. These enable routes created by developers to be used by external clients.

OpenShift Container Platform routers provide external hostname mapping and load balancing to services over protocols that pass distinguishing information directly to the router; the hostname must be present in the protocol in order for the router to determine where to send it. Routers support the following protocols:

- HTTP
- HTTPS (with SNI)
- WebSockets
- TLS with SNI

The router utilizes the wildcard zone specified during the installation and configuration of OpenShift Container Platform. This wildcard zone is used by the router to create routes for a service running within the OpenShift Container Platform environment to a publicly accessible URL. The wildcard zone itself is a wildcard entry in Cloud DNS that is linked using a CNAME to an **`Load Balancer`**, which performs a health checks for **`High Availability`** and forwards traffic to router pods on port 80 and 443.

2.13. OPENSHIFT SDN

OpenShift Container Platform uses a software-defined networking (SDN) approach to provide a unified cluster network that enables communication between pods across the OpenShift Container Platform cluster. The reference architecture scripts deploy the ovs-multitenant plug-in by default but the option exists to deploy the ovs-subnet plug-in.

- The ovs-multitenant plug-in provides OpenShift Container Platform project level isolation for pods and services. Each project receives a unique Virtual Network ID (VNID) that identifies traffic from pods assigned to the project. Pods from different projects cannot send packets to or receive packets from pods and services of a different project.

- The ovs-subnet plug-in is the original plug-in which provides a "flat" pod network where every pod can communicate with every other pod and service.

2.14. OPENSHIFT CLUSTER METRICS

OpenShift has the ability to gather metrics from kubelet and store the values in **`Heapster`**. OpenShift Cluster Metrics provide the ability to view CPU, memory, and network-based metrics and display the values in the user interface. These metrics can allow for the horizontal autoscaling of pods based on parameters provided by an OpenShift user. It is important to understand capacity planning when deploying metrics into an OpenShift environment.

When metrics are deployed, persistent storage should be use to allow for metrics to be preserved. Metrics data by default is stored for 7 days unless specified.

2.15. OPENSHIFT LOGGING

OpenShift allows the option to deploy aggregate logging for containers running in the OpenShift environment. OpenShift uses the EFK stack to collect logs from applications and present them to an OpenShift users. Cluster administrators can view all logs, but application developers can only view logs for projects they have permission to view. The EFK stack consists of the following components:

`Elasticsearch` - object store where logs are stored and provides the ability to search logs **`Fluentd`** - gathers and sends logs to Elasticsearch **`Kibana`** - a web interface to allow for easy interaction with Elasticsearch **`Curator`** - schedule Elasticsearch maintenance operations automatically

2.16. AUTHENTICATION

There are several options when it comes to authentication of users in OpenShift Container Platform. OpenShift Container Platform can leverage an existing identity provider within an organization such as **LDAP** or utilize an external identity providers like GitHub, Google, and GitLab. The configuration of identification providers occurs on the OpenShift Container Platform master instances. OpenShift Container Platform allows for multiple identity providers to be specified. This reference architecture uses Google as the authentication provider but any of the other mechanisms would be an acceptable choice. Roles can be added to user accounts to allow for extra privileges such as the ability to list nodes or assign persistent storage volumes to a project.

For more information on Google Oauth and other authentication methods see OpenShift Container Platform documentation.

CHAPTER 3. DEPLOYING OPENSHIFT

This chapter focuses on Phase 1 and 2 of the process. The prerequisites defined below are required for a successful deployment of infrastructure and the installation of OpenShift Container Platform 3.

3.1. PREREQUISITES FOR PROVISIONING

The script and playbooks provided within the openshift-ansible-contrib GitHub repository deploy infrastructure, install and configure OpenShift Container Platform, and perform post installation tasks such as scaling the router and registry. The playbooks create specific roles, policies, and users required for cloud provider configuration in OpenShift Container Platform.

Before beginning to provision resources and installing OpenShift Container Platform, there are few prerequisites that need to be taken care of that will be covered in the following sections.

3.1.1. Red Hat Subscription Manager

The installation of OpenShift Container Platform (OCP) requires a valid Red Hat subscription. During the installation of OpenShift the Ansible **`redhat_subscription`** module will attempt to register the instances. The script will fail if the OpenShift entitlements are exhausted. For the installation of **OCP** on **GCP** the following items are required:

Table 3.1. Red Hat Subscription Manager requirements

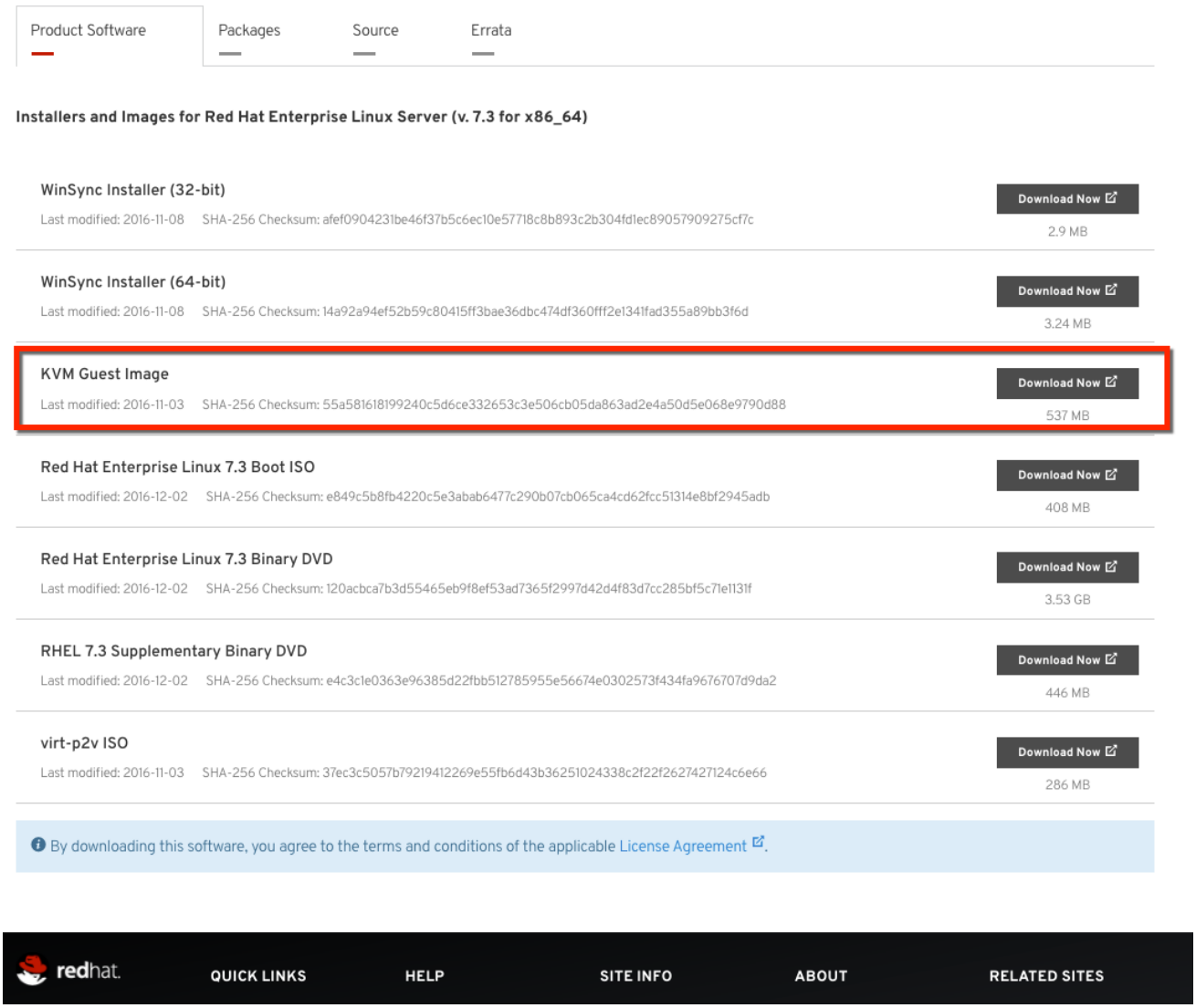
Requirement	Example User, Example Password, and Required Subscription
Red Hat Subscription Manager User	<Red Hat Username>
Red Hat Subscription Manager Password	<Red Hat Password>
Subscription Name or Pool ID	Red Hat OpenShift Container Platform, Standard, 2-Core

The items above are examples and should reflect subscriptions relevant to the account performing the installation. There are a few different variants of the OpenShift Container Platform Subscription Name. It is advised to visit https://access.redhat.com/management/subscriptions to find the specific Subscription Name as the value will be used below during the deployment.

3.1.2. Download Red Hat Enterprise Linux Image

Download the latest Red Hat Enterprise Linux 7.x KVM image from the Customer Portal.

Figure 3.1. Red Hat Enterprise Linux 7.x KVM Image Download



It is advised that the downloaded KVM image be placed in the `~/Downloads/` directory. If another directory is used to store the downloaded `.qcow2` file, write down the path to that directory for later use in variables file. The KVM image file that gets downloaded will later be used to create the Custom Image for use in **GCE**.

3.2. PREPARING THE GOOGLE CLOUD PLATFORM INFRASTRUCTURE

Preparing the Google Cloud Platform Environment.

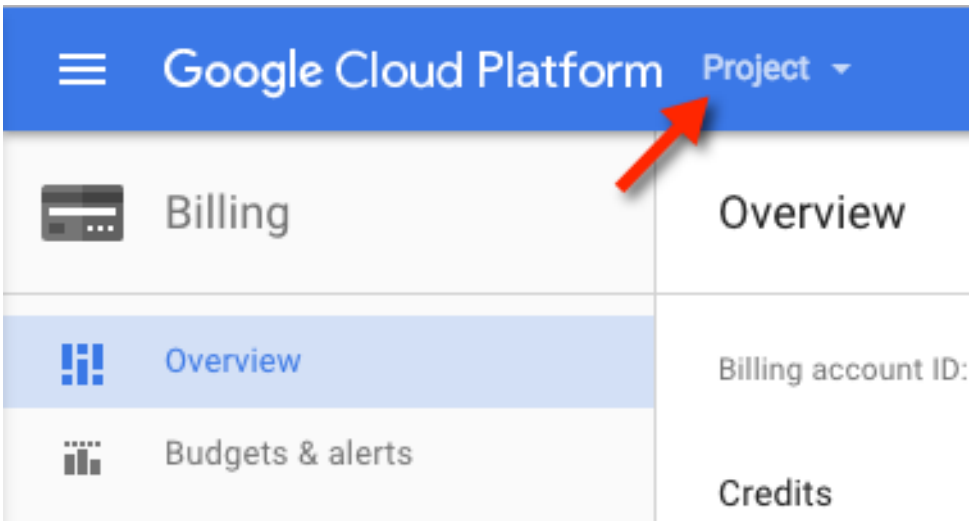
3.2.1. Sign in to the Google Cloud Platform Console

Documentation for the **GCP** and the Google Cloud Platform Console can be found on the following link: https://cloud.google.com/compute/docs. To access the Google Cloud Platform Console, a Google account is needed.

3.2.2. Create a new Google Cloud Platform Project

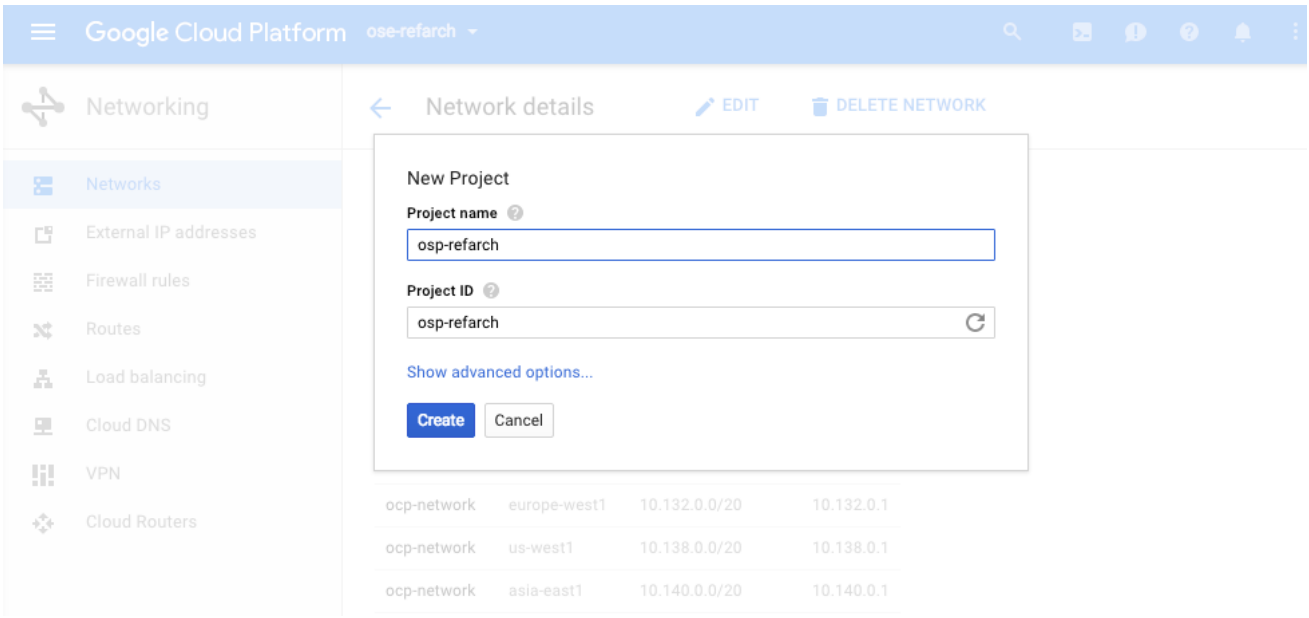
This reference architecture assumes that a "Greenfield" deployment is being done in a newly created **GCP** project. To set that up, log into the **GCP** console home and select the Project button:

Figure 3.2. Create New Project



Next, name your new project. The name `ocp-refarch` was used for this reference architecture, and can be used for your project also:

Figure 3.3. Name New Project



3.2.3. Set up Billing

The next step will be to set up billing for **GCP** so you are able to create new resources. Go to the Billing tab in the **GCP** Console and select Enable Billing. The new project can be linked to an existing project or financial information can be entered:

Figure 3.4. Enable Billing

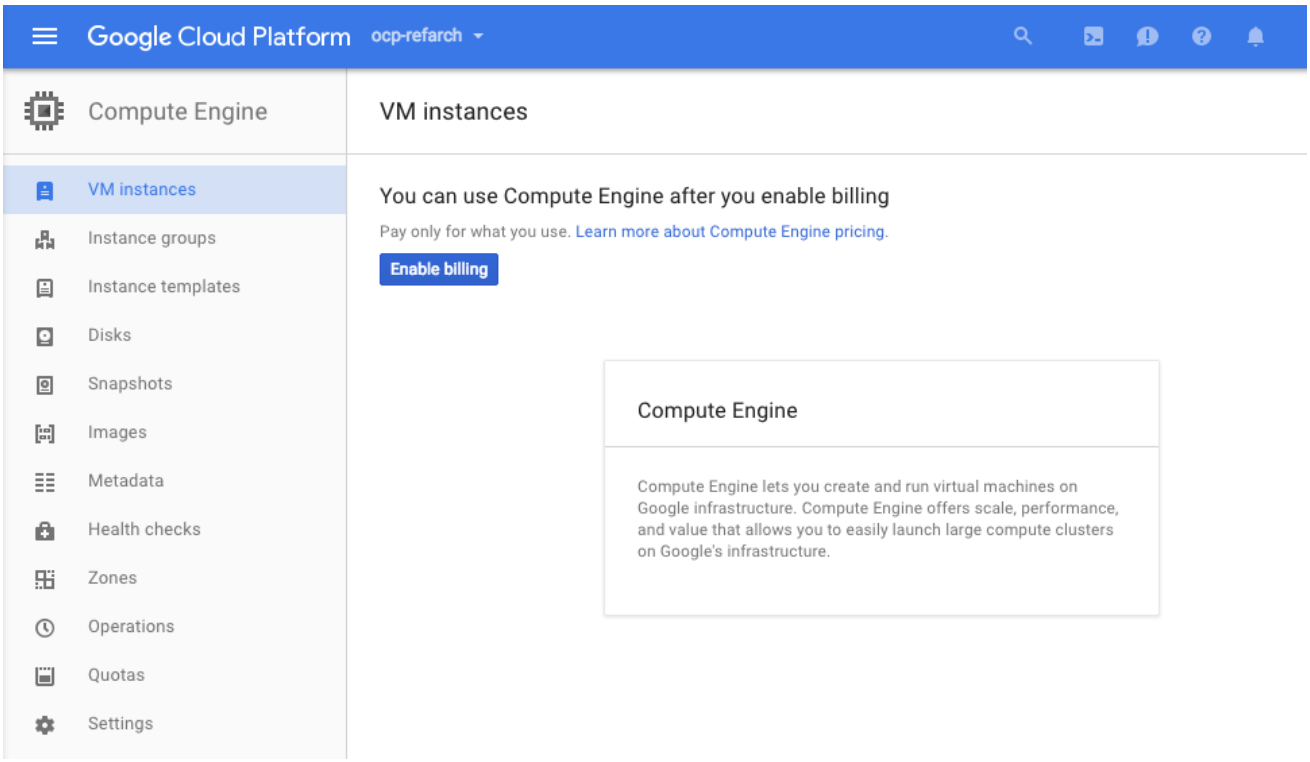
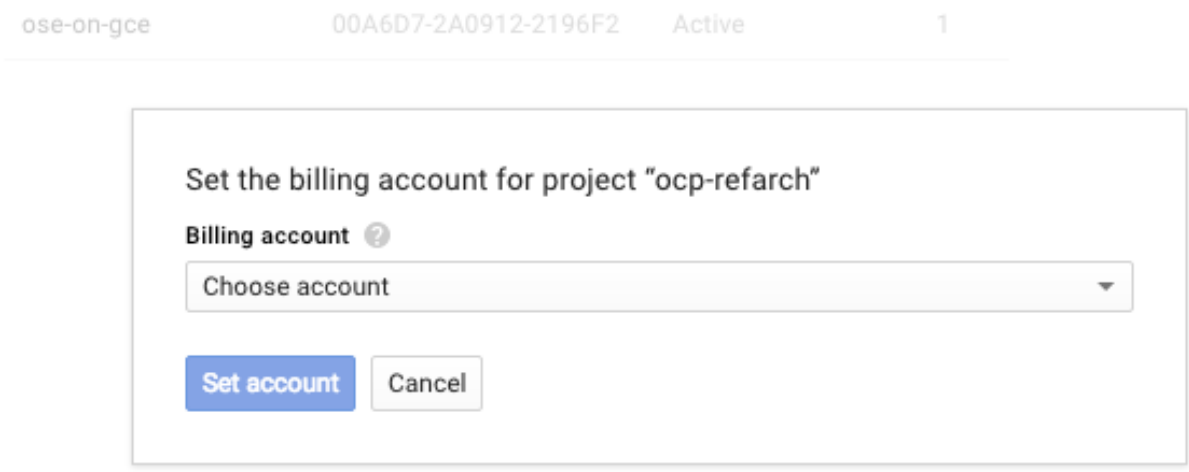


Figure 3.5. Link to existing account



3.2.4. Add a Cloud DNS Zone

In this reference implementation guide, the domain **sysdeseng.com** was purchased through an external provider and the subdomain **gce.sysdeseng.com** will be managed by **Cloud DNS**. For the example below, the domain **gce.sysdeseng.com** will represent the Publicly Hosted Domain Name used for the installation of OpenShift Container Platform, however, your Publicly Hosted Domain Name will need to be substituted for the actual installation. Follow the below instructions to add your Publicly Hosted Domain Name to **Cloud DNS**.

- From the main **GCP** dashboard, select the Network services section and click **Cloud DNS**
  - Click **Create Zone**

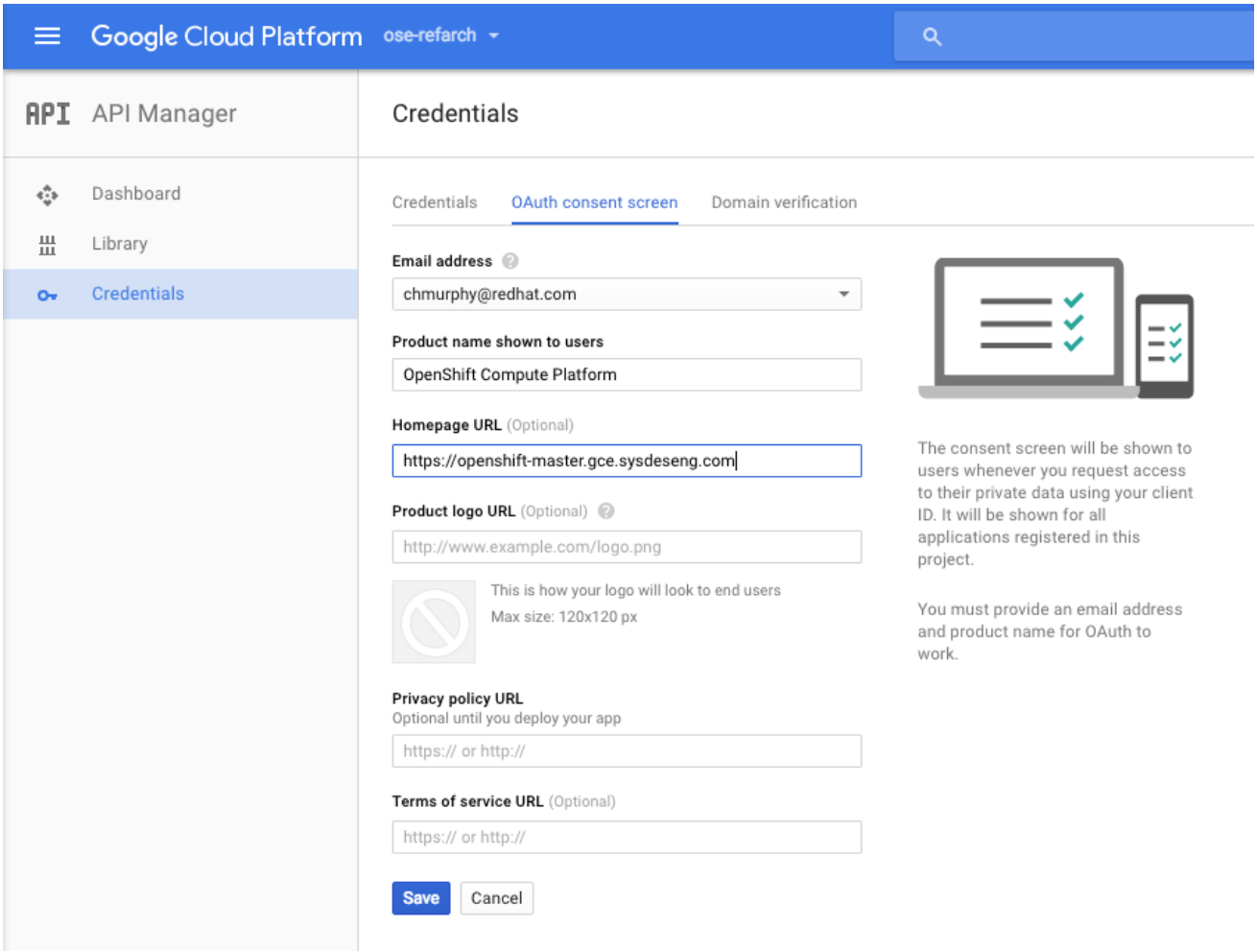
- Enter a name logical represent your Publicly Hosted Domain Name as the **Cloud DNS** Zone
  - the zone name must be separated by **-**,
  - it is recommended to just replace any **.'s with `-** from your Publicly Hosted Domain Name easier recognition of the zone: **gce-sysdeseng-com**
- Enter your Publicly Hosted Domain Name: **gce.sysdeseng.com**
- Enter a Description: Public Zone for OpenShift Container Platform 3 Reference Architecture
  - Click **Create**

Once the Pubic Zone is created, select the **gce-sysdeseng-com** domain and copy the Google Name Servers to add to your external registrar if applicable.

3.2.5. Create Google OAuth 2.0 Client IDs

In order to obtain new OAuth credentials in **GCP**, the OAuth consent screen must be configured. Browse to the API Manager screen in the **GCP** console and select Credentials. Select the "OAuth consent screen" tab as shown in the image below:

Figure 3.6. Google OAuth consent screen



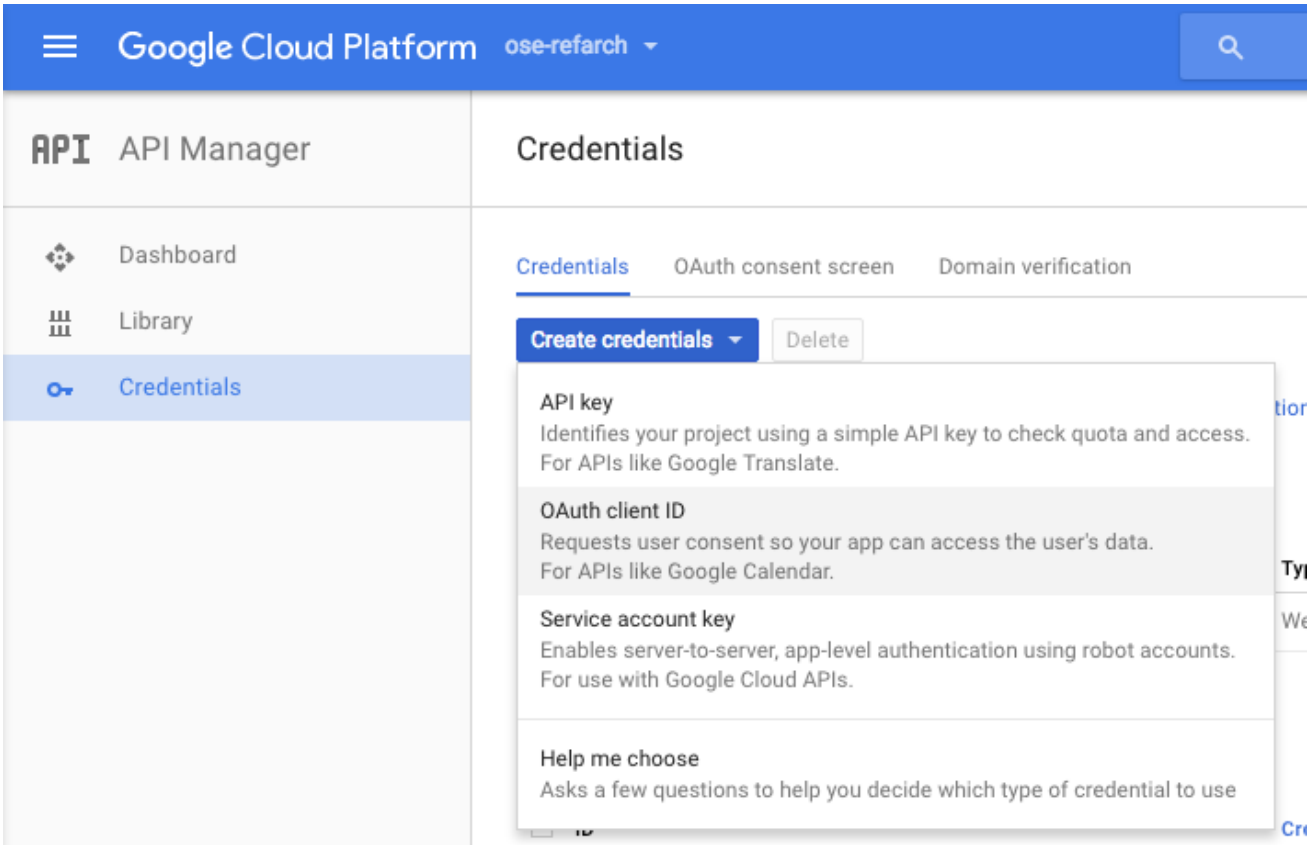
- Chose a email address from the dropdown

- Enter a Product Name
- (Optional, but recommended) Enter in a Homepage **URL**
  - The Homepage URL will be `https://.<your.domain.name>` (This will be the URL used when accessing OpenShift Container Platform 3)
- Optionally fill out the remaining fields, but they are not necessary for this reference architecture.
- Click **Save**

3.2.6. Fill in OAuth Request Form

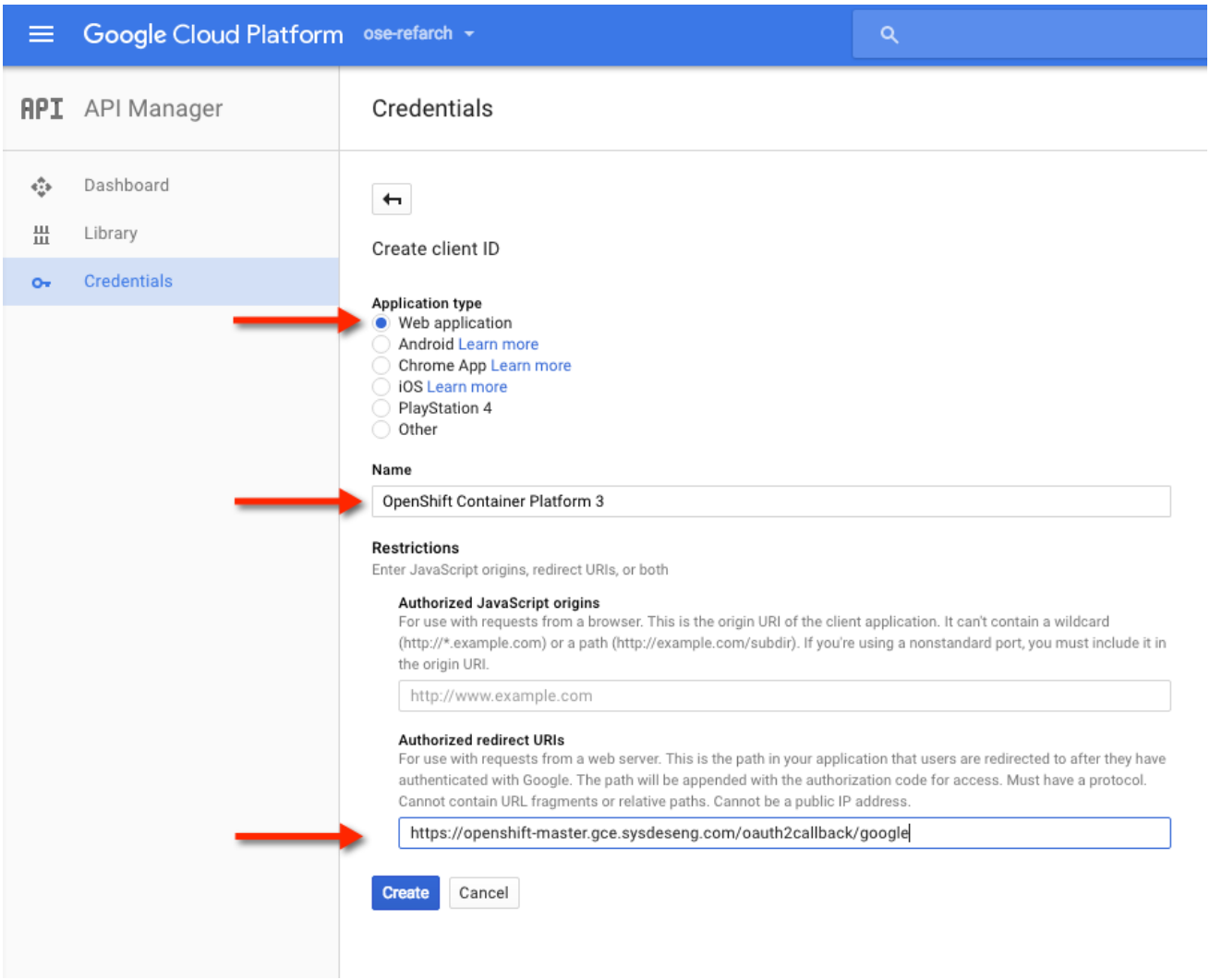
Clicking save will bring you back to the API Manager → Credentials tab. Next click on the **`Create Credentials`** dropdown and select **`OAuth Client ID`**

Figure 3.7. Select OAuth Client ID



On the the **`Create Client ID`** page:

Figure 3.8. New Client ID Page



Obtain both "Client ID" and "Client Secret"

- Click the **`Web Application`** radio button:
- Enter a Name
- Enter Authorization callback **URL**
  - The Authorization callback **URL** will be the Homepage **URL** + /oauth2callback/google (See example in the New Client ID Page image above)
- Click **`Create`**

Copy down the Client ID and Client Secret for use in the provisioning script.

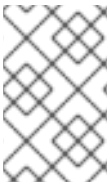
3.3. TOOLING PREREQUISITES

This section describes how the environment should be configured to use Ansible to provision the infrastructure, install OpenShift, and perform post installation tasks.

3.3.1. Required packages

The set of packages is required on the host running the deployment:

```
$ sudo yum update
$ sudo yum install curl python which tar qemu-img openssl git ansible
python-libcloud python2-jmespath java-1.8.0-openjdk-headless httpd-tools
python2-passlib
```



NOTE

You need to have GNU tar because the BSD version will not work. Also, it may be necessary to update qemu-img if the package is already installed. If the package is not updated, errors may occur when uploading the RHEL image to GCP.

3.3.2. OpenShift Ansible Installer

If you are running the deployment from RHEL 7 machine, enable required repositories and install the OpenShift Ansible Installer package:

```
$ sudo subscription-manager repos --enable rhel-7-server-rpms --enable
rhel-7-server-extras-rpms --enable rhel-7-fast-datapath-rpms --enable
rhel-7-server-ose-3.6-rpms
$ sudo yum install atomic-openshift-utils
```

NOTE

This step is not required. If the installer rpm package is not found on the system, it will be downloaded directly from the GitHub by the **ocp-on-gcp.sh** script.

3.3.3. Download and Set Up Google Cloud SDK

To manage your Google Cloud Platform resources, download and install the Google Cloud SDK with the gcloud command-line tool.

Google provides repository which can be used to install Google Cloud SDK on RHEL 7 or Fedora based OS:

```
$ sudo tee -a /etc/yum.repos.d/google-cloud-sdk.repo << EOM
[google-cloud-sdk]
name=Google Cloud SDK
baseurl=https://packages.cloud.google.com/yum/repos/cloud-sdk-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
       https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOM
```

Now it's easy to install the SDK and initialize the gcloud utility:

```
$ sudo yum install google-cloud-sdk
$ gcloud init
```

More information about the Google Cloud SDK can be found in the documentation.

3.3.3.1. Configure gcloud default project

This step is optional, as it was done as part of the **`gcloud init`** command, but if change is needed later, then:

```
$ gcloud config set project PROJECT-ID
```

3.3.3.2. List out available gcloud Zones and Regions

To find all regions and zones available to you, run:

```
$ gcloud compute zones list
NAME                     REGION                 STATUS  NEXT_MAINTENANCE
TURNDOWN_DATE
asia-east1-b             asia-east1             UP
asia-east1-c             asia-east1             UP
asia-east1-a             asia-east1             UP
asia-northeast1-b        asia-northeast1        UP
asia-northeast1-c        asia-northeast1        UP
asia-northeast1-a        asia-northeast1        UP
asia-south1-c            asia-south1            UP
asia-south1-a            asia-south1            UP
asia-south1-b            asia-south1            UP
asia-southeast1-a        asia-southeast1        UP
asia-southeast1-b        asia-southeast1        UP
australia-southeast1-a   australia-southeast1   UP
australia-southeast1-b   australia-southeast1   UP
australia-southeast1-c   australia-southeast1   UP
europe-west1-c           europe-west1           UP
europe-west1-b           europe-west1           UP
europe-west1-d           europe-west1           UP
europe-west2-a           europe-west2           UP
europe-west2-b           europe-west2           UP
europe-west2-c           europe-west2           UP
europe-west3-c           europe-west3           UP
europe-west3-b           europe-west3           UP
europe-west3-a           europe-west3           UP
southamerica-east1-a     southamerica-east1     UP
southamerica-east1-b     southamerica-east1     UP
southamerica-east1-c     southamerica-east1     UP
us-central1-f            us-central1            UP
us-central1-b            us-central1            UP
us-central1-a            us-central1            UP
us-central1-c            us-central1            UP
us-east1-c               us-east1               UP
us-east1-d               us-east1               UP
us-east1-b               us-east1               UP
us-east4-a               us-east4               UP
us-east4-b               us-east4               UP
us-east4-c               us-east4               UP
us-west1-c               us-west1               UP
us-west1-b               us-west1               UP
us-west1-a               us-west1               UP
```

This reference architecture uses the **us-central1-a** as the default zone. If you decide to use another zone, you can select from any of the zones within a region to be the default, there is no advantage to choosing one over the other.

3.4. PREPARE THE INSTALLATION SCRIPT

Once the **gcloud** utility is configured, deployment of the infrastructure and OpenShift Container Platform can be started. First, clone the **openshift-ansible-contrib** repository, switch to the **gcp** directory and copy the **config.yaml.example** file to **config.yaml**:

```
$ git clone https://github.com/openshift/openshift-ansible-contrib.git
$ cd openshift-ansible-contrib/reference-architecture/gcp
$ cp config.yaml.example config.yaml
```

Next edit the configuration variables in the **config.yaml** file to match those which you have been creating throughout this reference architecture. Below are the variables that can be set for the OpenShift Container Platform installation on Google Compute Engine. The Table is broken into two sections. Section A contains variables that must be changed in order for the installation to complete. The only case where these would not need to be changed would be if the default values were used during the set up sections in previous chapters. Links back to those chapters will be placed where possible. Section B contains variables that don't have to be changed. These variables can be set based on user preference including node sizing and node types, however, modifying any of these variables is beyond the scope this reference architecture and is therefore not recommended.

Table 3.2. OpenShift Container Platform config.yaml Installation Variables (Section A)

Variable Name	Defaults	Explanation/Purpose
prefix	'ocp'	Arbitrary prefix for every resource deployed
rhsm_user	'user@example.com'	Section 3.1.1, "Red Hat Subscription Manager"
rhsm_password	'xxx'	Section 3.1.1, "Red Hat Subscription Manager"
rhsm_pool	'OpenShift Enterprise Broker Infrastructure'	Section 3.1.1, "Red Hat Subscription Manager"
rhel_image_path	'~/Downloads/rhel-server-7.4-x86_64-kvm.qcow2'	Figure 3.1, "Red Hat Enterprise Linux 7.x KVM Image Download"
delete_image	false	Delete the uploaded image on teardown
delete_gold_image	false	Delete the gold image on teardown
gcloud_project	'project-1'	Figure 3.2, "Create New Project"

Variable Name	Defaults	Explanation/Purpose
gcloud_zone	'us-central1-a'	Section 3.3.3.2, "List out available gcloud Zones and Regions"
public_hosted_zone	'ocp.example.com'	Section 3.2.4, "Add a Cloud DNS Zone"
openshift_master_cluster_public_hostname	'master.{{ public_hosted_zone }}'	Public master hostname
openshift_master_cluster_hostname	'internal-master.{{ public_hosted_zone }}'	Internal master hostname
wildcard_zone	'apps.{{ public_hosted_zone }}'	Domain name for the OpenShift applications
master_https_key_file	''	
master_https_cert_file	''	
 **If SSL Certs signed by a CA are available for the Web Console, set these values to paths to the key and certificate files on the local file system. If you don't have SSL Certs, leave out the values empty and a SSL key and a self-signed certificate will be generated automatically during the provisioning process.**		
openshift_master_identity_providers	- name: 'google' kind: 'GoogleIdentityProvider' login: true challenge: false mapping_method: 'claim' clientID: **'xxx-yyy.apps.googleusercontent.com'** clientSecret: **'zzz'** hostedDomain: **'example.com'**	
 **This contains the YAML snippet that will set up Single-Sign-On as described in Obtain both "Client ID" and "Client Secret". Paste the "Client ID" and "Client Secret" in the appropriate places. The "hosted_domain" section is optional, but setting it will limit logins to users with the provided domain in their email address. The variable `openshift_master_identity_providers` is directly passed to the OpenShift Ansible Installer, so in case you need to use different identity provider, you can follow the documentation.**		

Table 3.3. OpenShift Container Platform `config.yaml` Installation Variables (Section B)

Variable Name	Defaults	Explanation/Purpose

Variable Name	Defaults	Explanation/Purpose
master_instance_group_size	3	**Minimum** Number of Master Nodes
infra_node_instance_group_size	3	**Minimum** Number of Infrastructure Nodes
node_instance_group_size	3	**Minimum** Number of Application Nodes
bastion_machine_type	'g1-small'	**Minimum** Machine Size for the Bastion Node
master_machine_type	'n1-standard-4'	**Minimum** Machine Size for Master Nodes
infra_machine_type	'n1-standard-2'	**Minimum** Machine Size for Infrastructure Nodes
node_machine_type	'n1-standard-2'	**Minimum** Machine Size for Application Nodes
openshift_deployment_type	'openshift-enterprise'	To deploy OpenShift Origin on CentOS, use 'origin'
containerized	false	To deploy OpenShift in containers, set to true
os_sdn_network_plugin_name	'redhat/openshift-ovs-multitenant'	OpenShift SDN
openshift_hosted_metrics_deploy	false	OpenShift Cluster Metrics
openshift_hosted_metrics_storage_volume_size	'20Gi'	Storage size for OpenShift Cluster Metrics
openshift_hosted_logging_deploy	false	OpenShift Logging
openshift_logging_es_pvc_size	'100Gi'	Storage size for OpenShift Logging
bastion_disk_size	20	**Minimum** Size of the Bastion disk in GB
master_boot_disk_size	45	**Minimum** Size of the Master Nodes boot disk in GB

Variable Name	Defaults	Explanation/Purpose
master_docker_disk_size	25	**Minimum** Size of the Master Nodes disk for Docker storage in GB, when containerized deployment is used (otherwise this disk is not created)
infra_boot_disk_size	25	**Minimum** Size of the Infrastructure Nodes boot disk in GB
infra_docker_disk_size	25	**Minimum** Size of the Infrastructure Nodes disk for Docker storage in GB
infra_openshift_disk_size	50	**Minimum** Size of the Infrastructure Nodes disk for OpenShift storage in GB
node_boot_disk_size	25	**Minimum** Size of the Application Nodes boot disk in GB
node_docker_disk_size	25	**Minimum** Size of the Application Nodes disk for Docker storage in GB
node_openshift_disk_size	50	**Minimum** Size of the Application Nodes disk for OpenShift storage in GB
gce_vpc_custom_subnet_cidr	''	Custom GCE VPC Subnet, example value: '10.160.0.0/20'. Default value is empty, when random subnet in form of 10.x.0.0/20 will be used

3.5. ADDITIONAL CONFIGURATION

Additional variables used to configure the OpenShift deployment can be found in the `ansible/playbooks/openshift-installer-common-vars.yaml` file. These variables can be overridden in the `config.yaml` if needed, but that is mostly out of scope of this reference architecture.

3.5.1. Containerized Deployment

The OCP installation playbooks allow for OpenShift to be installed in containers. If the containerized installation of OpenShift is preferred, set `containerized: true` in the `config.yaml` file.

3.5.2. OpenShift Metrics

OpenShift Cluster Metrics can be deployed during installation. To enable metrics deployment, set **`openshift_hosted_metrics_deploy: true`** in the **`config.yaml`**. This includes creating a PVC dynamically which is used for persistent metrics storage. The URL of the **`Hawkular Metrics`** endpoint is also set on the masters and defined by an OpenShift route. Metrics components are deployed on nodes with the label of "role=infra".

3.5.3. OpenShift Logging

OpenShift Cluster Logging can be deployed during installation. To enable metrics deployment, set **`openshift_hosted_logging_deploy: true`** in the **`config.yaml`**. This includes creating three PVCs dynamically which are used for storing logs in a redundant and durable way. Logging components are deployed on nodes with the label of "role=infra".

3.6. RUNNING THE OCP-ON-GCP.SH SCRIPT

OpenShift Container Platform `ocp-on-gcp.sh` Installation Script

```
$ ./ocp-on-gcp.sh
~/..../openshift-ansible-contrib/reference-architecture/gcp/ansible
~/..../openshift-ansible-contrib/reference-architecture/gcp

PLAY [configure ansible connection to the gcp and some basic stuff]
************

TASK [pre-flight-validation : check required packages]
*************************
included: /..../openshift-ansible-contrib/reference-
architecture/gcp/ansible/playbooks/roles/pre-flight-
validation/tasks/check-package.yaml for localhost
included: /..../openshift-ansible-contrib/reference-
architecture/gcp/ansible/playbooks/roles/pre-flight-
validation/tasks/check-package.yaml for localhost
included: /..../openshift-ansible-contrib/reference-
architecture/gcp/ansible/playbooks/roles/pre-flight-
validation/tasks/check-package.yaml for localhost
included: /..../openshift-ansible-contrib/reference-
architecture/gcp/ansible/playbooks/roles/pre-flight-
validation/tasks/check-package.yaml for localhost
included: /..../openshift-ansible-contrib/reference-
architecture/gcp/ansible/playbooks/roles/pre-flight-
validation/tasks/check-package.yaml for localhost
included: /..../openshift-ansible-contrib/reference-
architecture/gcp/ansible/playbooks/roles/pre-flight-
validation/tasks/check-package.yaml for localhost
included: /..../openshift-ansible-contrib/reference-
architecture/gcp/ansible/playbooks/roles/pre-flight-
validation/tasks/check-package.yaml for localhost

TASK [pre-flight-validation : check if package curl is installed]
**************
 [WARNING]: Consider using yum, dnf or zypper module rather than running
rpm

ok: [localhost]
```

```
TASK [pre-flight-validation : assert that package curl exists]
*****************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [pre-flight-validation : check if package python is installed]
************
ok: [localhost]

TASK [pre-flight-validation : assert that package python exists]
***************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [pre-flight-validation : check if package which is installed]
*************
ok: [localhost]

TASK [pre-flight-validation : assert that package which exists]
*****************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [pre-flight-validation : check if package openssl is installed]
***********
ok: [localhost]

TASK [pre-flight-validation : assert that package openssl exists]
**************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [pre-flight-validation : check if package git is installed]
***************
ok: [localhost]

TASK [pre-flight-validation : assert that package git exists]
******************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [pre-flight-validation : check if package python-libcloud is
installed] ***
ok: [localhost]

TASK [pre-flight-validation : assert that package python-libcloud exists]
```

```
******
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

.........................
.........................
.........................
```

Finished installation - OpenShift Container Platform `ocp-on-gcp.sh` Installation Script

```
.........................
.........................
.........................

TASK [validate-etcd : Validate etcd]
*****************************************
changed: [ocp-master-rscb]
changed: [ocp-master-rmtb]
changed: [ocp-master-97gr]

TASK [validate-etcd : ETCD Cluster is healthy]
*******************************
ok: [ocp-master-rscb] => {
    "msg": "Cluster is healthy"
}
ok: [ocp-master-97gr] => {
    "msg": "Cluster is healthy"
}
ok: [ocp-master-rmtb] => {
    "msg": "Cluster is healthy"
}

TASK [validate-etcd : ETCD Cluster is NOT healthy]
*****************************
skipping: [ocp-master-rscb]
skipping: [ocp-master-97gr]
skipping: [ocp-master-rmtb]

PLAY [single_master]
************************************************************

TASK [validate-app : Gather facts]
*****************************************
ok: [ocp-master-rscb]

TASK [validate-app : Create the validation project]
***************************
changed: [ocp-master-rscb]

TASK [validate-app : Create Hello world app]
*********************************
changed: [ocp-master-rscb]

TASK [validate-app : Wait for build to complete]
```

```
******************************
FAILED - RETRYING: Wait for build to complete (30 retries left).
FAILED - RETRYING: Wait for build to complete (29 retries left).
FAILED - RETRYING: Wait for build to complete (28 retries left).
FAILED - RETRYING: Wait for build to complete (27 retries left).
FAILED - RETRYING: Wait for build to complete (26 retries left).
FAILED - RETRYING: Wait for build to complete (25 retries left).
changed: [ocp-master-rscb]

TASK [validate-app : Wait for App to be running]
******************************
FAILED - RETRYING: Wait for App to be running (30 retries left).
changed: [ocp-master-rscb]

TASK [validate-app : Sleep to allow for route propegation]
*********************
Pausing for 10 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [ocp-master-rscb]

TASK [validate-app : check the status of the page]
*****************************
ok: [ocp-master-rscb]

TASK [validate-app : Delete the Project]
************************************
changed: [ocp-master-rscb]

PLAY [print message about the ocp console]
***********************************

TASK [print message about the ocp console url]
*********************************
ok: [localhost] => {
    "msg": "Deployment is complete. OpenShift Console can be found at
https://master.gce.sysdeseng.com/"
}

PLAY RECAP
*********************************************************************
localhost                  : ok=42   changed=17   unreachable=0
failed=0
ocp-bastion                : ok=10   changed=0    unreachable=0
failed=0
ocp-infra-node-4k4l        : ok=272  changed=74   unreachable=0
failed=0
ocp-infra-node-rwjl        : ok=272  changed=74   unreachable=0
failed=0
ocp-infra-node-stl6        : ok=272  changed=74   unreachable=0
failed=0
ocp-master-97gr            : ok=487  changed=143  unreachable=0
failed=0
ocp-master-rmtb            : ok=487  changed=143  unreachable=0
failed=0
ocp-master-rscb            : ok=704  changed=210  unreachable=0
failed=0
```

```
ocp-node-9xt0              : ok=264  changed=78   unreachable=0
failed=0
ocp-node-mh4w              : ok=264  changed=78   unreachable=0
failed=0
ocp-node-z8rv              : ok=264  changed=78   unreachable=0
failed=0

~/work/redhat/e2e/openshift-ansible-contrib/reference-architecture/gcp
```

NOTE

A successful deployment is shown above with the two important columns being "unreachable=0" and "failed=0"

NOTE

The last message of the installation script will show the address for the configured OpenShift Container Platform Console. In this example, the OpenShift Container Platform Console can now be reached at **https://master.gce.sysdeseng.com**.

3.7. REGISTRY CONSOLE SELECTOR (OPTIONAL)

The OpenShift Registry Console deployment is deployed on any Red Hat OpenShift Container Platform node by default, so the container may end up running on any of the application nodes.

From the first *master* instance(ocp-master-97gr), ensure the OpenShift Registry Console pod runs on the *infra* nodes by modifying the *nodeSelector* as follows:

```
$ oc patch dc registry-console \
    -n default \
    -p '{"spec":{"template":{"spec":{"nodeSelector":{"role":"infra"}}}}}'
```

NOTE

There is a bugzilla ID: 1425022 being investigated by Red Hat at the time of writing this paper to fix this issue.

CHAPTER 4. OPERATIONAL MANAGEMENT

With the successful deployment of OpenShift, the following section demonstrates how to confirm proper functionality of the Red Hat OpenShift Container Platform.

4.1. VALIDATE THE DEPLOYMENT

With the successful deployment of OpenShift, the following section demonstrates how to confirm proper functionality of the OpenShift environment. An Ansible script in the git repository will allow for an application to be deployed which will test the functionality of the master, nodes, registry, and router. The playbook will test the deployment and clean up any projects and pods created during the validation run.

The playbook will perform the following steps:

Environment Validation

- Validate the public OpenShift Container Platform Load Balancer **`ocp-master-https-lb-url-map`** address from the installation system
- Validate the public OpenShift Container Platform Load Balancer **`ocp-router-network-lb-pool`** address from the master nodes
- Validate the OpenShift Container Platform Load Balancer **`ocp-master-network-lb-pool`** address from the master nodes
- Validate the master local master address
- Validate the health of the **ETCD** cluster to ensure all **ETCD** nodes are healthy
- Create a project on OpenShift Container Platform called **`validate`**
- Create an OpenShift Container Platform Application
- Add a route for the Application
- Validate the URL returns a status code of 200 or healthy
- Delete the validation project

4.1.1. Running the Validation playbook

In case you need to re-validate the OpenShift deployment, run:

```
$ ./ocp-on-gcp.sh --validation
```

4.2. GATHERING HOSTNAMES

Because **GCE** will automatically append a random 4-digit string to instances created from an "Instance Template", it is impossible to know the final hostnames of the instances until after they are created. To view the final hostnames, browse to the **GCP** "Compute Engine" → "VM instances" dashboard. The following image shows an example of a successful installation.

Figure 4.1. Compute Engine - VM Instances

Name	Zone	Recommendation	Internal IP	External IP
ocp-bastion	us-central1-a		10.128.0.2	35.202.16
ocp-infra-node-4k4l	us-central1-b		10.128.0.7	35.193.22
ocp-infra-node-rwjl	us-central1-c		10.128.0.4	35.188.13.
ocp-infra-node-stl6	us-central1-f		10.128.0.6	35.194.13.
ocp-master-97gr	us-central1-c		10.128.0.3	35.188.93.
ocp-master-rmtb	us-central1-b		10.128.0.8	35.192.14
ocp-master-rscb	us-central1-f		10.128.0.10	35.194.63.
ocp-node-9xt0	us-central1-b		10.128.0.11	35.202.13
ocp-node-mh4w	us-central1-c		10.128.0.5	35.202.18
ocp-node-z8rv	us-central1-f		10.128.0.9	35.192.99.

To help facilitate `Operational Management` Chapter, the following hostnames based on the above image will be used with numbers replacing the random 4-digit strings. The numbers are not meant to suggest ordering or importance, they are simply used to make it easier to distinguish the hostnames.

- ocp-master-01.gce.sysdeseng.com
- ocp-master-02.gce.sysdeseng.com
- ocp-master-03.gce.sysdeseng.com
- ocp-infra-node-01.gce.sysdeseng.com
- ocp-infra-node-02.gce.sysdeseng.com
- ocp-infra-node-03.gce.sysdeseng.com
- ocp-node-01.gce.sysdeseng.com
- ocp-node-02.gce.sysdeseng.com
- ocp-node-03.gce.sysdeseng.com
- ocp-bastion.gce.sysdeseng.com

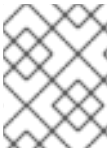
4.3. CHECKING THE HEALTH OF ETCD

This section focuses on the **ETCD** cluster. It describes the different commands to ensure the cluster is healthy. The internal `Cloud DNS` names of the nodes running **ETCD** must be used.

SSH into the first master node (ocp-master-01.gce.sysdeseng.com). Using the output of the command **hostname** issue the **etcdctl** command to confirm that the cluster is healthy.

```
$ ssh ocp-master-01
$ sudo -i
```

```
# ETCD_ENDPOINT=$(hostname) && etcdctl -C https://$ETCD_ENDPOINT:2379 --
ca-file /etc/etcd/ca.crt --cert-file=/etc/origin/master/master.etcd-
client.crt --key-file=/etc/origin/master/master.etcd-client.key cluster-
health
member 8499e5795394a44 is healthy: got healthy result from
https://10.128.0.8:2379
member 36d79a9d80d2baca is healthy: got healthy result from
https://10.128.0.10:2379
member abdd0db2d8da61c7 is healthy: got healthy result from
https://10.128.0.3:2379
cluster is healthy
```



NOTE

In this configuration the **ETCD** services are distributed among the OpenShift Container Platform master nodes.

4.4. DEFAULT NODE SELECTOR

As explained in Section 2.10.4, “Node labels”, node labels are an important part of the OpenShift Container Platform environment. By default of the reference architecture installation, the default node selector is set to **role=apps** in **/etc/origin/master/master-config.yaml** on all of the master nodes. This configuration parameter is set by the OpenShift installation playbooks on all masters and the master API service is restarted that is required when making any changes to the master configuration.

SSH into the first master node (ocp-master-01.gce.sysdeseng.com) to verify the **defaultNodeSelector** is defined.

```
# vi /etc/origin/master/master-config.yaml
...omitted...
projectConfig:
  defaultNodeSelector: "role=app"
  projectRequestMessage: ""
  projectRequestTemplate: ""
...omitted...
```

NOTE

If making any changes to the master configuration then the master API service must be restarted or the configuration change will not take place. Any changes and the subsequent restart must be done on all masters.

4.5. MANAGEMENT OF MAXIMUM POD SIZE

Quotas are set on ephemeral volumes within pods to prohibit a pod from becoming too large and

impacting the node. There are three places where sizing restrictions should be set. When persistent volume claims are not set a pod has the ability to grow as large as the underlying filesystem will allow. The required modifications are set using a combination of user-data and Ansible.

OpenShift Container Platform Volume Quota

At launch time user-data creates a xfs partition on the **/dev/disk/by-id/google-openshift** block device, adds an entry in fstab, and mounts the volume with the option of gquota. If gquota is not set the OpenShift node will not be able to start with the **perFSGroup** parameter defined below. This disk and configuration is done on the infrastructure and application nodes.

SSH into the first infrastructure node (ocp-infra-01.gce.sysdeseng.com) to verify the entry exists within fstab.

```
# vi /etc/fstab
/dev/disk/by-id/google-openshift  /var/lib/origin/openshift.local.volumes
xfs  defaults,gquota,comment=cloudconfig  0  0
```

Docker Storage Setup

Docker storage utilizes **overlay2** storage driver, which doesn't require any specific configuration. Only requirement for this driver is xfs filesystem. The reference architecture installation is providing additional disk for Docker storage **/dev/disk/by-id/google-docker**.

SSH into the first infrastructure node (ocp-infra-01.gce.sysdeseng.com) to verify the entry exists within fstab and that **/etc/sysconfig/docker-storage** matches the output below.

```
# vi /etc/fstab
/dev/disk/by-id/google-docker  /var/lib/docker  xfs
defaults,comment=cloudconfig  0  0
```

```
# vi /etc/sysconfig/docker-storage
DOCKER_STORAGE_OPTIONS='--storage-driver overlay2'
```

OpenShift Container Platform Emptydir Quota

The Ansible variable **node_local_quota_per_fsgroup** configures **perFSGroup** on all nodes. The **perFSGroup** setting restricts the ephemeral emptyDir volume from growing larger than 512Mi. This empty dir quota is done on the infrastructure and application nodes.

SSH into the first infrastructure node (ocp-infra-01.gce.sysdeseng.com) to verify **/etc/origin/node/node-config.yaml** matches the information below.

```
# vi /etc/origin/node/node-config.yaml
...omitted...
volumeConfig:
  localQuota:
     perFSGroup: 512Mi
```

4.6. YUM REPOSITORIES

In Section 2.7, "Required Channels" the specific repositories for a successful OpenShift Container Platform installation were defined. All systems except for the bastion host should have the same

subscriptions. To verify subscriptions match those defined in Required Channels perform the following. The repositories below are enabled during the rhsm-repos playbook during the installation. The installation will be unsuccessful if the repositories are missing from the system.

```
# yum repolist
repo id                                    repo name
status
google-cloud-compute                       Google Cloud Compute
6
rhel-7-fast-datapath-rpms/7Server/x86_64 Red Hat Enterprise Linux Fast
Datapath (RHEL 7 Server) (RP     38
rhel-7-server-extras-rpms/x86_64         Red Hat Enterprise Linux 7 Server
- Extras (RPMs)          619+20
rhel-7-server-ose-3.6-rpms/x86_64        Red Hat OpenShift Container
Platform 3.6 (RPMs)            503+20
rhel-7-server-rpms/7Server/x86_64        Red Hat Enterprise Linux 7 Server
(RPMs)                   17,188
repolist: 18,354
```

NOTE

All rhui repositories are disabled and only those repositories defined in the Ansible role **rhsm-repos** are enabled.

4.7. CONSOLE ACCESS

This section will cover logging into the OpenShift Container Platform management console via the GUI and the CLI. After logging in via one of these methods applications can then be deployed and managed.

4.7.1. Log into GUI console and deploy an application

Perform the following steps from your local workstation.

Open a browser and access https://master.gce.sysdeseng.com/console. When logging into the OpenShift Container Platform web interface the first time the page will redirect and prompt for Google OAuth credentials. Log into Google using an account that is a member of the account specified during the install. Next, Google will prompt to grant access to authorize the login. If Google access is not granted the account will not be able to login to the OpenShift Container Platform web console.

To deploy an application, click on the **New Project** button. Provide a **Name** and click **Create**. Next, deploy the **jenkins-ephemeral** instant app by clicking the corresponding box. Accept the defaults and click **Create**. Instructions along with a URL will be provided for how to access the application on the next screen. Click **Continue to Overview** and bring up the management page for the application. Click on the link provided and access the application to confirm functionality.

4.7.2. Log into CLI and Deploy an Application

Perform the following steps from your local workstation.

Install the **oc client** by visiting the public URL of the OpenShift Container Platform deployment. For example, https://master.gce.sysdeseng.com/console/command-line and click latest release. When directed to https://access.redhat.com, login with the valid Red Hat customer credentials and download the client relevant to the current workstation. Follow the instructions located on the production documentation site for getting started with the cli.

A token is required to login using Google OAuth and OpenShift Container Platform. The token is presented on the https://master.gce.sysdeseng.com/console/command-line URL. Click the link to obtain a token and then perform the following on the workstation in which the oc client was installed.

```
$ oc login https://master.gce.sysdeseng.com --
token=fEAjn7LnZE6v5SOocCSRVmUWGBNIIEKbjD9h-Fv7p09
```

After the oc client is configured, create a new project and deploy an application.

```
$ oc new-project test-app

$ oc new-app https://github.com/openshift/cakephp-ex.git --name=php
--> Found image 0d2f8fa (5 weeks old) in image stream "openshift/php"
under tag "7.0" for "php"

    Apache 2.4 with PHP 7.0
    -----------------------
    PHP 7.0 available as docker container is a base platform for building
and running various PHP 7.0 applications and frameworks. PHP is an HTML-
embedded scripting language. PHP attempts to make it easy for developers
to write dynamically generated web pages. PHP also offers built-in database
integration for several commercial and non-commercial database management
systems, so writing a database-enabled webpage with PHP is fairly simple.
The most common use of PHP coding is probably as a replacement for CGI
scripts.

    Tags: builder, php, php70, rh-php70

    * The source repository appears to match: php
    * A source build using source code from
https://github.com/openshift/cakephp-ex.git will be created
      * The resulting image will be pushed to image stream "php:latest"
      * Use 'start-build' to trigger a new build
    * This image will be deployed in deployment config "php"
    * Port 8080/tcp will be load balanced by service "php"
      * Other containers can access this service through the hostname
"php"

--> Creating resources ...
    imagestream "php" created
    buildconfig "php" created
    deploymentconfig "php" created
    service "php" created
--> Success
    Build scheduled, use 'oc logs -f bc/php' to track its progress.
    Run 'oc status' to view your app.

$ oc expose service php
route "php" exposed
```

Display the status of the application.

```
$ oc status
In project test-app on server https://master.gce.sysdeseng.com:443
```

```
http://php-test-app.apps.gce.sysdeseng.com to pod port 8080-tcp (svc/php)
  dc/php deploys istag/php:latest <-
    bc/php source builds https://github.com/openshift/cakephp-ex.git on
openshift/php:7.0
    deployment #1 deployed about a minute ago - 1 pod

View details with 'oc describe <resource>/<name>' or list everything with
'oc get all'.
```

Access the application by accessing the URL provided by **`oc status`**. The CakePHP application should be visible now.

4.8. EXPLORE THE ENVIRONMENT

4.8.1. List Nodes and Set Permissions

If you try to run the following command, it should fail.

```
$ oc get nodes --show-labels
Error from server: User "user@example.com" cannot list all nodes in the
cluster
```

The reason it is failing is because the permissions for that user are incorrect. Get the username and configure the permissions.

```
$ oc whoami
```

Once the username has been established, log back into a master node and enable the appropriate permissions for your user. Perform the following step from the first master (ocp-master-01.gce.sysdeseng.com).

```
# oadm policy add-cluster-role-to-user cluster-admin user@example.com
```

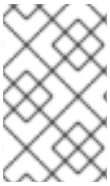
Attempt to list the nodes again and show the labels.

```
$ oc get nodes --show-labels
NAME                  STATUS                     AGE       VERSION
LABELS
ocp-infra-node-4k4l   Ready                      32m
v1.6.1+5115d708d7
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=n1-
standard-2,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=us-central1,failure-
domain.beta.kubernetes.io/zone=us-central1-b,kubernetes.io/hostname=ocp-
infra-node-4k4l,role=infra
ocp-infra-node-rwjl   Ready                      32m
v1.6.1+5115d708d7
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=n1-
standard-2,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=us-central1,failure-
domain.beta.kubernetes.io/zone=us-central1-c,kubernetes.io/hostname=ocp-
infra-node-rwjl,role=infra
ocp-infra-node-stl6   Ready                      32m
```

```
v1.6.1+5115d708d7
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=n1-
standard-2,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=us-central1,failure-
domain.beta.kubernetes.io/zone=us-central1-f,kubernetes.io/hostname=ocp-
infra-node-stl6,role=infra
ocp-master-97gr       Ready,SchedulingDisabled   32m
v1.6.1+5115d708d7
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=n1-
standard-2,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=us-central1,failure-
domain.beta.kubernetes.io/zone=us-central1-c,kubernetes.io/hostname=ocp-
master-97gr,role=master
ocp-master-rmtb       Ready,SchedulingDisabled   32m
v1.6.1+5115d708d7
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=n1-
standard-2,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=us-central1,failure-
domain.beta.kubernetes.io/zone=us-central1-b,kubernetes.io/hostname=ocp-
master-rmtb,role=master
ocp-master-rscb       Ready,SchedulingDisabled   32m
v1.6.1+5115d708d7
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=n1-
standard-2,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=us-central1,failure-
domain.beta.kubernetes.io/zone=us-central1-f,kubernetes.io/hostname=ocp-
master-rscb,role=master
ocp-node-9xt0         Ready                      32m
v1.6.1+5115d708d7
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=n1-
standard-2,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=us-central1,failure-
domain.beta.kubernetes.io/zone=us-central1-b,kubernetes.io/hostname=ocp-
node-9xt0,role=app
ocp-node-mh4w         Ready                      32m
v1.6.1+5115d708d7
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=n1-
standard-2,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=us-central1,failure-
domain.beta.kubernetes.io/zone=us-central1-c,kubernetes.io/hostname=ocp-
node-mh4w,role=app
ocp-node-z8rv         Ready                      32m
v1.6.1+5115d708d7
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=n1-
standard-2,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=us-central1,failure-
domain.beta.kubernetes.io/zone=us-central1-f,kubernetes.io/hostname=ocp-
node-z8rv,role=app
```

4.8.2. List Router and Registry

List the router and registry by changing to the **`default`** project.



NOTE

If the OpenShift account configured on the workstation has cluster-admin privileges perform the following. If the account does not have this privilege ssh to one of the OpenShift masters and perform the steps.

```
$ oc project default
$ oc get all
NAME                  DOCKER REPO
TAGS      UPDATED
is/registry-console   docker-registry.default.svc:5000/default/registry-
console   v3.6      23 minutes ago

NAME                  REVISION   DESIRED   CURRENT   TRIGGERED BY
dc/docker-registry    1          3         3         config
dc/registry-console   1          1         1         config
dc/router             1          3         3         config

NAME                    DESIRED   CURRENT   READY     AGE
rc/docker-registry-1    3         3         3         24m
rc/registry-console-1   1         1         1         23m
rc/router-1             3         3         3         27m

NAME                      HOST/PORT
PATH      SERVICES           PORT      TERMINATION   WILDCARD
routes/docker-registry    docker-registry-default.apps.gce.sysdeseng.com
docker-registry    <all>     passthrough   None
routes/registry-console   registry-console-default.apps.gce.sysdeseng.com
registry-console   <all>     passthrough   None

NAME                  CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
svc/docker-registry   172.30.139.160   <none>        5000/TCP
25m
svc/kubernetes        172.30.0.1       <none>
443/TCP,53/UDP,53/TCP     56m
svc/registry-console  172.30.41.250    <none>        9000/TCP
23m
svc/router            172.30.91.254    <none>
80/TCP,443/TCP,1936/TCP   27m

NAME                          READY     STATUS    RESTARTS   AGE
po/docker-registry-1-mxqc4    1/1       Running   0          24m
po/docker-registry-1-qp05k    1/1       Running   0          24m
po/docker-registry-1-s3kp9    1/1       Running   0          24m
po/registry-console-1-gn1kg   1/1       Running   0          22m
po/router-1-60q15             1/1       Running   0          27m
po/router-1-m4swp             1/1       Running   0          27m
po/router-1-qq86v             1/1       Running   0          27m
```

Observe the output of **`oc get all`**.

4.8.3. Explore the Registry

The OpenShift Ansible playbooks configure three infrastructure nodes that have three registries

running. In order to understand the configuration and mapping process of the registry pods, the command **oc describe** is used. The **oc describe** details how registries are configured and mapped to the **GCS** for storage. Using **oc describe** should help explain how HA works in this environment.

NOTE

If the OpenShift account configured on the workstation has cluster-admin privileges perform the following. If the account does not have this privilege ssh to one of the OpenShift masters and perform the steps.

```
$ oc describe svc/docker-registry
Name:              docker-registry
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          docker-registry=default
Type:              ClusterIP
IP:                172.30.139.160
Port:              5000-tcp  5000/TCP
Endpoints:         172.16.14.3:5000,172.16.16.2:5000,172.16.4.3:5000
Session Affinity:  ClientIP
Events:            <none>
```

Notice that the registry has three **endpoints** listed. Each of those **endpoints** represents a container. The **ClusterIP** listed is the actual ingress point for the registries.

The **oc** client allows similar functionality to the **docker** command. To find out more information about the registry storage perform the following.

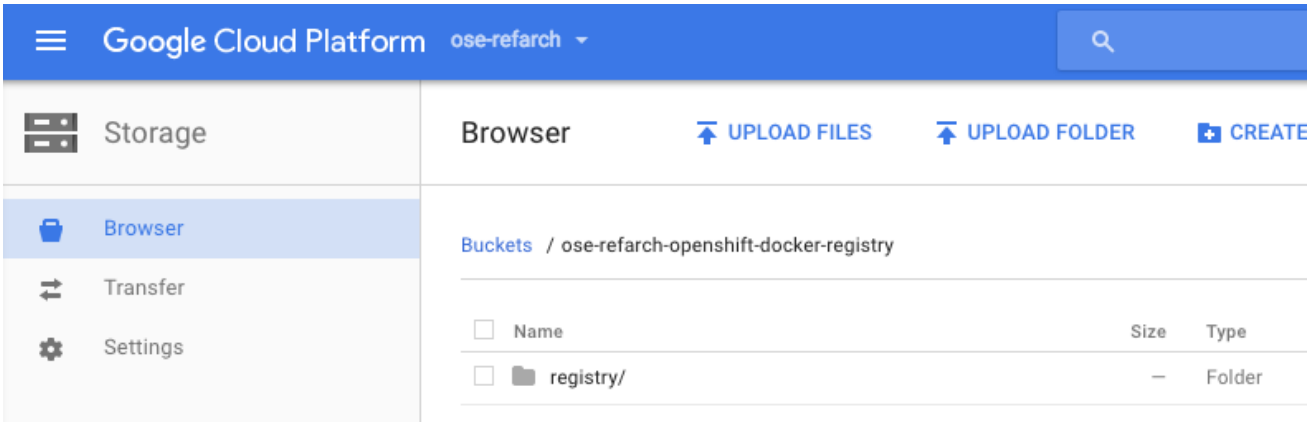
```
$ oc get pods
NAME                       READY     STATUS    RESTARTS   AGE
docker-registry-1-mxqc4    1/1       Running   0          27m
docker-registry-1-qp05k    1/1       Running   0          27m
docker-registry-1-s3kp9    1/1       Running   0          27m
registry-console-1-gn1kg   1/1       Running   0          25m
router-1-60q15             1/1       Running   0          29m
router-1-m4swp             1/1       Running   0          29m
router-1-qq86v             1/1       Running   0          29m
```

```
$ oc exec docker-registry-1-mxqc4 cat /etc/registry/config.yml
version: 0.1
log:
  level: debug
http:
  addr: :5000
storage:
  delete:
    enabled: true
  cache:
    blobdescriptor: inmemory
  gcs:
    bucket: ose-refarch-ocp-registry-bucket
    rootdirectory: /registry
auth:
```

```
  openshift:
    realm: openshift
middleware:
  registry:
  - name: openshift
  repository:
  - name: openshift
    options:
      pullthrough: True
      acceptschema2: True
      enforcequota: True
  storage:
  - name: openshift
```

Observe the **GCS** stanza. Confirm the bucket name is listed, and access the **GCP** console. Click on the "Storage" tab and locate the bucket. The bucket should contain the **`/registry`** content as seen in the image below.

Figure 4.2. Registry Bucket in GCS



Confirm that the same bucket is mounted to the other registries via the same steps.

4.8.4. Explore Docker Storage

This section will explore the Docker storage on an infrastructure node.

The example below can be performed on any node but for this example the infrastructure node(ocp-infra-01.gce.sysdeseng.com) is used.

The output below verifies docker storage is not using a loop back device.

```
# docker info
Containers: 4
 Running: 4
 Paused: 0
 Stopped: 0
Images: 3
Server Version: 1.12.6
Storage Driver: overlay2
 Backing Filesystem: xfs
Logging Driver: journald
Cgroup Driver: systemd
Plugins:
```

```
 Volume: local
 Network: host null bridge overlay
 Authorization: rhel-push-plugin
Swarm: inactive
Runtimes: docker-runc runc
Default Runtime: docker-runc
Security Options: seccomp selinux
Kernel Version: 3.10.0-693.1.1.el7.x86_64
Operating System: Employee SKU
OSType: linux
Architecture: x86_64
Number of Docker Hooks: 3
CPUs: 2
Total Memory: 7.147 GiB
Name: ocp-infra-node-4k4l
ID: FNH2:VZ2O:LQ4V:CDDY:LLYE:6YVN:VEHU:4735:X4EW:KK6F:KV7H:WLSN
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://registry.access.redhat.com/v1/
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Insecure Registries:
 127.0.0.0/8
Registries: registry.access.redhat.com (secure),
registry.access.redhat.com (secure), docker.io (secure)
```

Verify 3 disks are attached to the instance. The disk **/dev/sda** is used for the OS, **/dev/sdb** is used for docker storage, and **/dev/sdc** is used for emptyDir storage for containers that do not use a persistent volume.

```
# fdisk -l
Disk /dev/sda: 26.8 GB, 26843545600 bytes, 52428800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk label type: dos
Disk identifier: 0x000a73bb

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    52428766    26213359+  83  Linux

Disk /dev/sdc: 53.7 GB, 53687091200 bytes, 104857600 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/sdb: 26.8 GB, 26843545600 bytes, 52428800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
```

4.8.5. Explore Firewall Rules

As mentioned earlier in the document several firewall rules have been created. The purpose of this section is to encourage exploration of the firewall rules that were created.

NOTE

Perform the following steps from the **GCP** web console.

On the main **GCP** console, click on the left hand navigation panel select the **VPC network**. Select **Firewall rules** and check the rules that were created as part of the infrastructure provisioning. For example, notice how the **ocp-ssh-external** firewall rule only allows **SSH** traffic inbound. That can be further restricted to a specific network or host if required.

4.8.6. Explore the Load Balancers

As mentioned earlier in the document several **Load Balancers** have been created. The purpose of this section is to encourage exploration of the **Load Balancers** that were created.

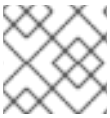
NOTE

Perform the following steps from the **GCP** web console.

On the main **GCP** console, click on the left hand navigation panel and select the **Network services** → **Load Balancing**. Select the **ocp-master-https-lb-url-map** load balancer and note the **Frontend** and how it is configured for port 443. That is for the OpenShift Container Platform web console traffic. On the same tab, notice that the **ocp-master-https-lb-cert** is offloaded at the load balancer level and therefore not terminated on the masters. On the **Backend**, there should be three healthy master instances running. Next check the **ocp-master-network-lb-pool** load balancer to see that it contains the three masters that make up the **Backend** from the **ocp-master-https-lb-url-map** load balancer. Further details of the configuration can be viewed by exploring the Ansible playbooks to see exactly what was configured.

4.8.7. Explore the VPC Network

As mentioned earlier in the document, a network was created especially for **OCP**. The purpose of this section is to encourage exploration of the network that was created.



NOTE

Perform the following steps from the **GCP** web console.

On the main **GCP** console, click on the left hand navigation panel select the **VPC network** → **VPC networks**. Select the **ocp-network** recently created and explore the **Subnets**, **Firewall Rules**, and **Routes**. More detail can be looked at with the configuration by exploring the Ansible playbooks to see exactly what was configured.

4.9. TESTING FAILURE

In this section, reactions to failure are explored. After a successful install and some of the smoke tests noted above have been completed, failure testing is executed.

4.9.1. Generate a Master Outage

NOTE

Perform the following steps from the **GCP** web console and the OpenShift Container Platform public URL.

On the main **GCP** console, click on the left hand navigation panel select the **`VM Instances`**. Locate your running ocp-master-02.gce.sysdeseng.com instance, select it, and click on **STOP**.

Ensure the console can still be accessed by opening a browser and accessing https://master.gce.sysdeseng.com/. At this point, the cluster is in a degraded state because only 2/3 master nodes are running, but complete functionality remains.

4.9.2. Observe the Behavior of ETCD with a Failed Master Node

SSH into the first master node (ocp-master-01.gce.sysdeseng.com). Using the output of the command **`hostname`** issue the **`etcdctl`** command to confirm that the cluster is healthy.

```
$ ssh ocp-master-01
$ sudo -i
```

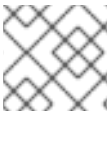
```
# hostname -i
10.128.0.8
# ETCD_ENDPOINT=$(hostname) && etcdctl -C https://$ETCD_ENDPOINT:2379 --
ca-file /etc/etcd/ca.crt --cert-file=/etc/origin/master/master.etcd-
client.crt --key-file=/etc/origin/master/master.etcd-client.key cluster-
health
member 8499e5795394a44 is healthy: got healthy result from
https://10.128.0.8:2379
failed to check the health of member 36d79a9d80d2baca on
https://10.128.0.10:2379: Get https://10.128.0.10:2379/health: dial tcp
10.128.0.10:2379: i/o timeout
member 36d79a9d80d2baca is unreachable: [https://10.128.0.10:2379] are all
unreachable
member abdd0db2d8da61c7 is healthy: got healthy result from
https://10.128.0.3:2379
cluster is healthy
```

Notice how one member of the **ETCD** cluster is now unreachable. Restart ocp-master-02.gce.sysdeseng.com by following the same steps in the **GCP** web console as noted above.

4.9.3. Generate an Infrastructure Node outage

This section shows what to expect when an infrastructure node fails or is brought down intentionally.

4.9.3.1. Confirm Application Accessibility



NOTE

Perform the following steps from the browser on a local workstation.

Before bringing down an infrastructure node, check behavior and ensure things are working as expected. The goal of testing an infrastructure node outage is to see how the OpenShift Container

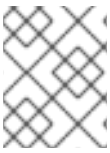
Platform routers and registries behave. Confirm the simple application deployed from before is still functional. If it is not, deploy a new version. Access the application to confirm connectivity. As a reminder, to find the required information the ensure the application is still running, list the projects, change to the project that the application is deployed in, get the status of the application which including the URL and access the application via that URL.

```
$ oc get projects
NAME               DISPLAY NAME   STATUS
default                           Active
kube-public                       Active
kube-system                       Active
logging                           Active
management-infra                  Active
openshift                         Active
openshift-infra                   Active
test-app                          Active

$ oc project test-app
Now using project "test-app" on server "https://master.gce.sysdeseng.com".

$ oc status
In project test-app on server https://master.gce.sysdeseng.com:443

http://php-test-app.apps.gce.sysdeseng.com to pod port 8080-tcp (svc/php)
  dc/php deploys istag/php:latest <-
    bc/php source builds https://github.com/openshift/cakephp-ex.git on
openshift/php:7.0
    deployment #1 deployed 3 minutes ago - 1 pod

View details with 'oc describe <resource>/<name>' or list everything with
'oc get all'.
```

Open a browser and ensure the application is still accessible.

4.9.3.2. Confirm Registry Functionality

This section is another step to take before initiating the outage of the infrastructure node to ensure that the registry is functioning properly. The goal is to push to the OpenShift Container Platform registry.



NOTE

Perform the following steps from a CLI on a local workstation and ensure that the oc client has been configured.

A token is needed so that the Docker registry can be logged into.

```
$ oc whoami -t
AP4y3J6IFGrqtZsliqpIRkncHKaOSV9gRuoIzXzBGtg
```

Pull a new docker image for the purposes of test pushing.

```
$ sudo docker pull fedora/apache
$ sudo docker images
```

Capture the registry endpoint. The **svc/docker-registry** shows the endpoint.

```
$ oc status
In project default on server https://internal-master.gce.sysdeseng.com:443

https://docker-registry-default.apps.gce.sysdeseng.com (passthrough)
(svc/docker-registry)
  dc/docker-registry deploys docker.io/openshift3/ose-docker-
registry:v3.6.173.0.21
    deployment #1 deployed 31 minutes ago - 3 pods

svc/kubernetes - 172.30.0.1 ports 443, 53->8053, 53->8053

https://registry-console-default.apps.gce.sysdeseng.com (passthrough)
(svc/registry-console)
  dc/registry-console deploys
registry.access.redhat.com/openshift3/registry-console:v3.6
    deployment #1 deployed 29 minutes ago - 1 pod

svc/router - 172.30.91.254 ports 80, 443, 1936
  dc/router deploys docker.io/openshift3/ose-haproxy-router:v3.6.173.0.21
    deployment #1 deployed 34 minutes ago - 3 pods

View details with 'oc describe <resource>/<name>' or list everything with
'oc get all'.
```

Tag the docker image with the endpoint from the previous step.

```
$ sudo docker tag docker.io/fedora/apache docker-registry-
default.apps.gce.sysdeseng.com/openshift/prodapache
```

Check the images and ensure the newly tagged image is available.

```
$ sudo docker images
```

Configure registry **docker-registry-default.apps.gce.sysdeseng.com** as insecure registry for local Docker daemon and issue a Docker login.

```
$ sudo docker login -u user@example.com -p
AP4y3J6IFGrqtZsliqpIRkncHKaOSV9gRuoIzXzBGtg docker-registry-
default.apps.gce.sysdeseng.com
Login Succeeded
```

Allow our user to push images to the OpenShift registry.

```
$ oadm policy add-role-to-user admin user@example.com -n openshift
$ oadm policy add-role-to-user system:registry user@example.com
$ oadm policy add-role-to-user system:image-builder user@example.com
```

Push the image to the OpenShift Container Platform registry now.

```
$ sudo docker push docker-registry-
default.apps.gce.sysdeseng.com/openshift/prodapache
The push refers to a repository [docker-registry-
```

```
default.apps.gce.sysdeseng.com/openshift/prodapache]
3a85ee80fd6c: Pushed
5b0548b012ca: Pushed
a89856341b3d: Pushed
a839f63448f5: Pushed
e4f86288aaf7: Pushed
latest: digest:
sha256:ec10a4b8f5acec40f08f7e04c68f19b31ac92af14b0baaadd0def17deb53a229
size: 1362
```

4.9.3.3. Get Location of Router and Registry.

NOTE

Perform the following steps from the CLI of a local workstation.

Change to the default OpenShift Container Platform project and check the router and registry pod locations.

```
$ oc project default
Now using project "default" on server "https://master.gce.sysdeseng.com".

$ oc get pods
NAME                       READY     STATUS    RESTARTS   AGE
docker-registry-1-mxqc4    1/1       Running   0          32m
docker-registry-1-qp05k    1/1       Running   0          32m
docker-registry-1-s3kp9    1/1       Running   0          32m
registry-console-1-gn1kg   1/1       Running   0          30m
router-1-60q15             1/1       Running   0          34m
router-1-m4swp             1/1       Running   0          34m
router-1-qq86v             1/1       Running   0          34m

$ oc describe pod docker-registry-1-mxqc4 | grep Node:
Node:    ocp-infra-node-stl6/10.128.0.6
$ oc describe pod docker-registry-1-qp05k | grep Node:
Node:    ocp-infra-node-4k4l/10.128.0.7
$ oc describe pod docker-registry-1-s3kp9 | grep Node:
Node:    ocp-infra-node-rwjl/10.128.0.4
$ oc describe pod router-1-60q15 | grep Node:
Node:    ocp-infra-node-4k4l/10.128.0.7
$ oc describe pod router-1-m4swp | grep Node:
Node:    ocp-infra-node-stl6/10.128.0.6
$ oc describe pod router-1-qq86v | grep Node:
Node:    ocp-infra-node-rwjl/10.128.0.4
```

4.9.3.4. Initiate the Failure and Confirm Functionality

NOTE

Perform the following steps from the **GCP** web console and a browser.

On the main **GCP** console, click on the left hand navigation panel and select the **`VM instances`**. Locate your running infra01 instance, select it, and click on **STOP**. Wait a minute or two for the registry

and pod to recreate on other instances. Check the registry locations and confirm that they are not on the infra01 node.

```
$ oc get pods
NAME                       READY     STATUS    RESTARTS   AGE
docker-registry-1-3897j    1/1       Running   0          11m
docker-registry-1-qp05k    1/1       Running   0          32m
docker-registry-1-s3kp9    1/1       Running   0          32m
registry-console-1-gn1kg   1/1       Running   0          30m
router-1-60q15             1/1       Running   0          34m
router-1-m4swp             1/1       Running   0          34m
router-2-l90pf             1/1       Running   0          11m

$ oc describe pod docker-registry-1-qp05k | grep Node:
Node:    ocp-infra-node-4k4l/10.128.0.7
$ oc describe pod docker-registry-1-s3kp9 | grep Node:
Node:    ocp-infra-node-rwjl/10.128.0.4
$ oc describe pod docker-registry-1-3897j | grep Node:
Node:    ocp-infra-node-4k4l/10.128.0.7
```

Follow the procedures above to ensure a Docker image can still be pushed to the registry now that infra01 is down.

4.10. GENERATING A STATIC INVENTORY

The need may arise for a static Ansible inventory to be used. The Ansible playbook below will query Google Cloud Platform to list the OpenShift instances. The playbook generates a file named **`static-inventory`**.

```
$ ./ocp-on-gcp.sh --static-inventory
```

This script will output the file **`static-inventory`** to the **`ansible`** directory. The static inventory can then be called by using the -i parameter with Ansible. Below is an example running the uninstall playbook with the newly generated static-inventory.

```
$ ansible-playbook -i ansible/static-inventory
/usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

4.11. UPDATING THE OPENSHIFT DEPLOYMENT

Playbooks are provided to upgrade the OpenShift deployment when minor releases occur.

4.11.1. Performing the Upgrade

Run the following to perform the minor upgrade against the deployed environment.

```
$ ./ocp-on-gcp.sh --minor-upgrade
```

4.11.2. Upgrading and Restarting the OpenShift Environment (Optional)

The minor upgrade playbook will not restart the instances after updating occurs. Restarting the nodes including the masters can be completed by using the following option.

```
$ ./ocp-on-gcp.sh --minor-upgrade -e
'openshift_rolling_restart_mode=system'
```

4.11.3. Specifying the OpenShift Version when Upgrading

The deployed OpenShift environment may not be the latest major version of OpenShift. The upgrade playbook allows for a variable to be passed to perform an upgrade on previous versions. Below is an example of performing the upgrade on a 3.5 non-containerized environment.

```
$ ./ocp-on-gcp.sh --minor-upgrade -e 'openshift_vers=v3_5'
```

CHAPTER 5. PERSISTENT STORAGE

Container storage by default is not persistent. For example, if a new container build occurs then data is lost because the storage is non-persistent. If a container terminates then of the all changes to it's local filesystem are lost. OpenShift offers many different types of persistent storage. Persistent storage ensures that data that should persist between builds and container migrations is available. The different storage options can be found at https://docs.openshift.com/container-platform/latest/architecture/additional_concepts/storage.html#types-of-persistent-volumes. When choosing a persistent storage backend ensure that the backend supports the scaling, speed, and redundancy that the project requires. This reference architecture will focus on cloud provider specific storage.

5.1. PERSISTENT VOLUMES

Container storage is defined by the concept of **`persistent volumes`** (pv) which are OpenShift objects that allow for storage to be defined and then used by pods to allow for data persistence. Requesting of **`persistent volumes`** is done by using a **`persistent volume claim`** (pvc). This claim, when successfully fulfilled by the system will also mount the persistent storage to a specific directory within a pod or multiple pods. This directory is referred to as the **`mountPath`** and facilitated using a concept known as **`bind-mount`**.

5.2. STORAGE CLASSES

The **`StorageClass`** resource object describes and classifies different types of storage that can be requested, as well as provides a means for passing parameters to the backend for dynamically provisioned storage on demand. **`StorageClass`** objects can also serve as a management mechanism for controlling different levels of storage and access to the storage. **`Cluster Administrators (cluster-admin)`** or **`Storage Administrators (storage-admin)`** define and create the **`StorageClass`** objects that users can use without needing any intimate knowledge about the underlying storage volume sources. Because of this the naming of the **`storage class`** defined in the **`StorageClass`** object should be useful in understanding the type of storage it maps to (pd-standard or pd-ssd).

5.3. CLOUD PROVIDER SPECIFIC STORAGE

Cloud provider specific storage is storage that is provided from **GCP**. This type of storage is presented as a **`persistent disk`** and can be mounted by one pod at a time. The **`persistent disk`** must exist in the zone as the pod that requires the storage. This is because **`persistent disks`** cannot be mounted by an instance outside of the zone that it was created. For **GCP** there are two types of persistent disks that can be utilized, standard and ssd, for cloud provider specific storage. Cloud provider storage can be created manually and assigned as a persistent volume or a persistent volume can be created dynamically using a **`StorageClass`** object. Note that **`persistent storage`** can only use the access mode of Read-Write-Once (RWO). The reference architecture creates two storage classes by default - the **`standard`** storage class which is set as the default, and the **`ssd`** storage class.

5.3.1. Observe Storage Classes

To observe available storage classes, run following commands.

```
$ oc get storageclass
NAME                 TYPE
ssd                  kubernetes.io/gce-pd
```

```
standard (default)   kubernetes.io/gce-pd

$ oc describe storageclass standard
Name:            standard
IsDefaultClass:  Yes
Annotations:     storageclass.beta.kubernetes.io/is-default-class=true
Provisioner:     kubernetes.io/gce-pd
Parameters:      type=pd-standard
Events:          <none>

$ oc describe storageclass ssd
Name:            ssd
IsDefaultClass:  No
Annotations:     storageclass.beta.kubernetes.io/is-default-class=false
Provisioner:     kubernetes.io/gce-pd
Parameters:      type=pd-ssd
Events:          <none>
```

5.3.2. Creating and using a Persistent Volumes Claim

The example below shows a dynamically provisioned volume being requested from the **StorageClass** named standard.

```
$ vi db-claim.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: db
 annotations:
   volume.beta.kubernetes.io/storage-class: standard
spec:
 accessModes:
  - ReadWriteOnce
 resources:
   requests:
     storage: 10Gi

$ oc create -f db-claim.yaml
persistentvolumeclaim "db" created
```

5.3.3. Deleting a PVC (Optional)

There may become a point in which a **PVC** is no longer necessary for a project. The following can be done to remove the **PVC**.

```
$ oc delete pvc db
persistentvolumeclaim "db" deleted
$ oc get pvc db
Error from server (NotFound): persistentvolumeclaims "db" not found
```

5.4. RWO PERSISTENT STORAGE (OPTIONAL)

Cloud provider storage can be used for persistent storage for an application using a **StorageClass** object. This example is for a **PHP** application which has requirements for a persistent volume mount point.

Create a test project for the demo application.

```
$ oc new-project rwo
```

Create the application using the following github link:

```
$ oc new-app openshift/php:7.0~https://github.com/christianh814/openshift-
php-upload-demo --name=demo
--> Found image b73997e (13 days old) in image stream "openshift/php"
under tag "7.0" for "openshift/php:7.0"

    Apache 2.4 with PHP 7.0
    -----------------------
    PHP 7.0 available as docker container is a base platform for building
and running various PHP 7.0 applications and frameworks. PHP is an HTML-
embedded scripting language. PHP attempts to make it easy for developers
to write dynamically generated web pages. PHP also offers built-in database
integration for several commercial and non-commercial database management
systems, so writing a database-enabled webpage with PHP is fairly simple.
The most common use of PHP coding is probably as a replacement for CGI
scripts.

    Tags: builder, php, php70, rh-php70

    * A source build using source code from
https://github.com/christianh814/openshift-php-upload-demo will be created
      * The resulting image will be pushed to image stream "demo:latest"
      * Use 'start-build' to trigger a new build
    * This image will be deployed in deployment config "demo"
    * Port 8080/tcp will be load balanced by service "demo"
      * Other containers can access this service through the hostname
"demo"

--> Creating resources ...
    imagestream "demo" created
    buildconfig "demo" created
    deploymentconfig "demo" created
    service "demo" created
--> Success
    Build scheduled, use 'oc logs -f bc/demo' to track its progress.
    Run 'oc status' to view your app.
```

Validate that the build is complete and the pods are running.

```
$ oc get pods
NAME           READY     STATUS      RESTARTS   AGE
demo-1-build   0/1       Completed   0          1m
demo-1-snkgt   1/1       Running     0          39s
```

The next step is to retrieve the name of the OpenShift **svc** which will be used to create a route.

```
$ oc get svc
NAME      CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
demo      172.30.174.114   <none>        8080/TCP   1m
```

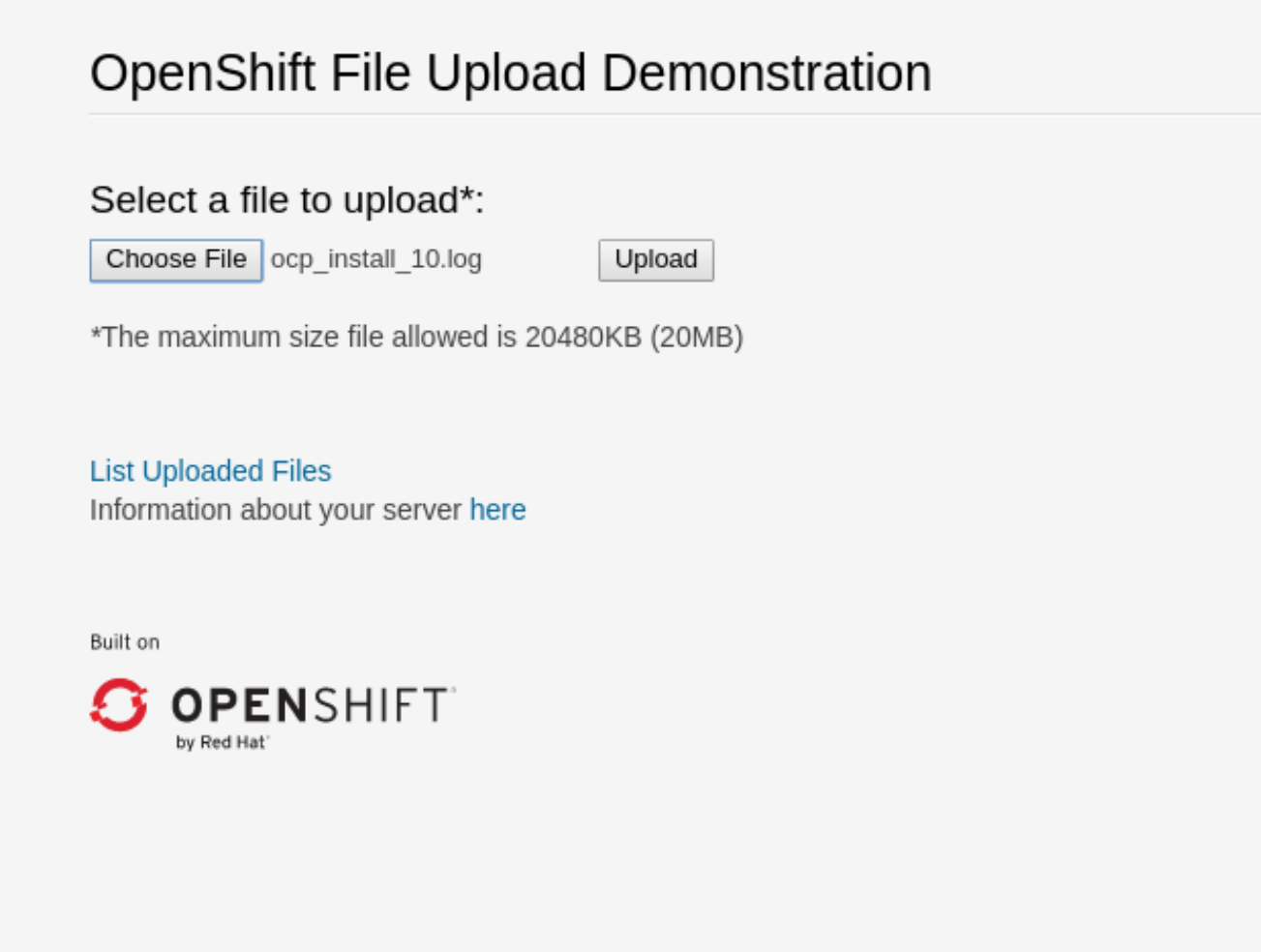
Expose the service as a public route by using the **oc expose** command.

```
$ oc expose svc/demo
route "demo" exposed
```

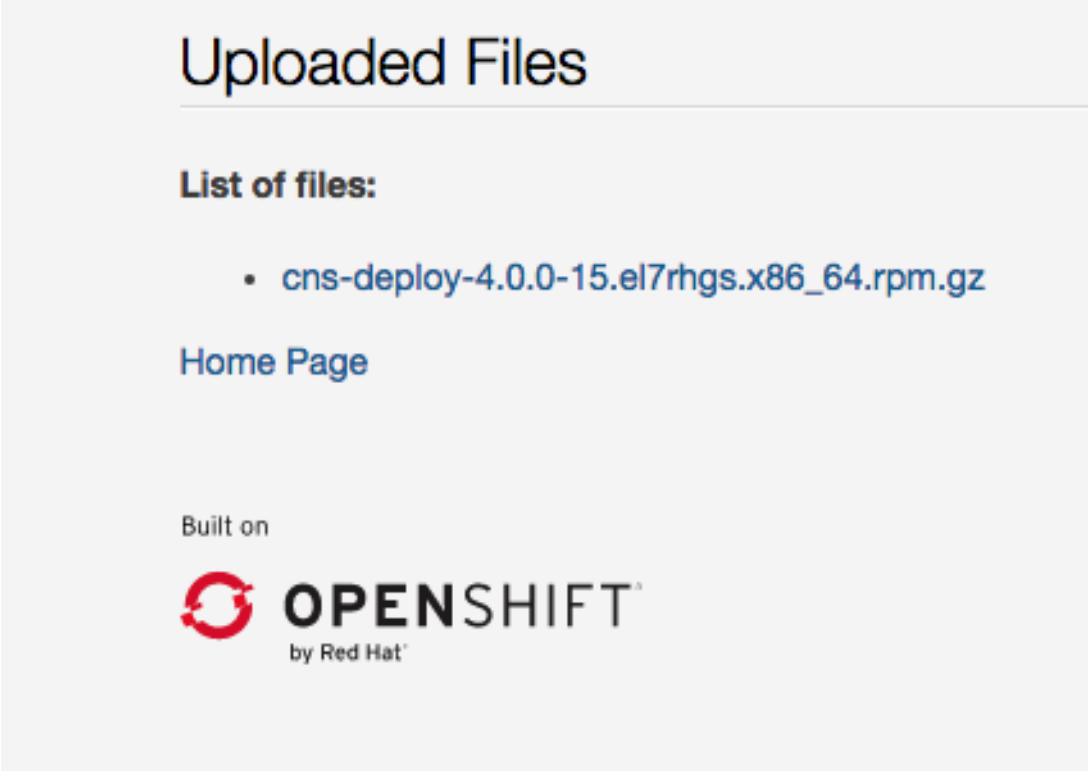
OpenShift will create a route based on the application name, project, and wildcard zone. This will be the **URL** that can be accessed by browser.

```
$ oc get route
NAME      HOST/PORT                          PATH      SERVICES   PORT
TERMINATION   WILDCARD
demo      demo-rwo.apps.gce.sysdeseng.com             demo       8080-tcp
None
```

Using a web browser validate the application (example http://demo-rwo.apps.gce.sysdeseng.com/) using the route defined in the previous step.



Upload a file using the web UI.



Connect to the **demo-1-snkgt** and verify the file exists.

```
$ oc get pods
NAME           READY     STATUS      RESTARTS   AGE
demo-1-build   0/1       Completed   0          2m
demo-1-snkgt   1/1       Running     0          2m

$ oc rsh demo-1-snkgt
sh-4.2$ cd uploaded/
sh-4.2$ pwd
/opt/app-root/src/uploaded
sh-4.2$ ls -lh
total 16K
-rw-r--r--. 1 1000080000 root 16K Sep 12 16:07 cns-deploy-4.0.0-
15.el7rhgs.x86_64.rpm.gz
```

First, add a persistent volume claim to the project. The existing **OCP StorageClass** object (standard) will be used to create a **PVC** with the access mode of **RWO**.

The first step is to create the **app-claim.yaml** file.

```
$ vi app-claim.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: app
 annotations:
   volume.beta.kubernetes.io/storage-class: standard
spec:
 accessModes:
  - ReadWriteOnce
```

```
  resources:
    requests:
      storage: 10Gi
```

Using the **app-claim.yaml** file use the OpenShift client to create the **PVC**.

```
$ oc create -f app-claim.yaml
persistentvolumeclaim "app" created
```

Verify the **PVC** was created.

```
$ oc get pvc app
NAME      STATUS    VOLUME                                     CAPACITY
ACCESSMODES   STORAGECLASS   AGE
app       Bound     pvc-d987c02c-97d4-11e7-8508-42010a80000a   10Gi
RWO           standard       6s
```

Now that the **PVC** exists tie the claim to the deployment configuration using the existing mount path **/opt/app-root/src/uploaded** for **demo** pods.

```
$ oc volume dc/demo --add --name=persistent-volume --
type=persistentVolumeClaim --claim-name=app --mount-path=/opt/app-
root/src/uploaded
```

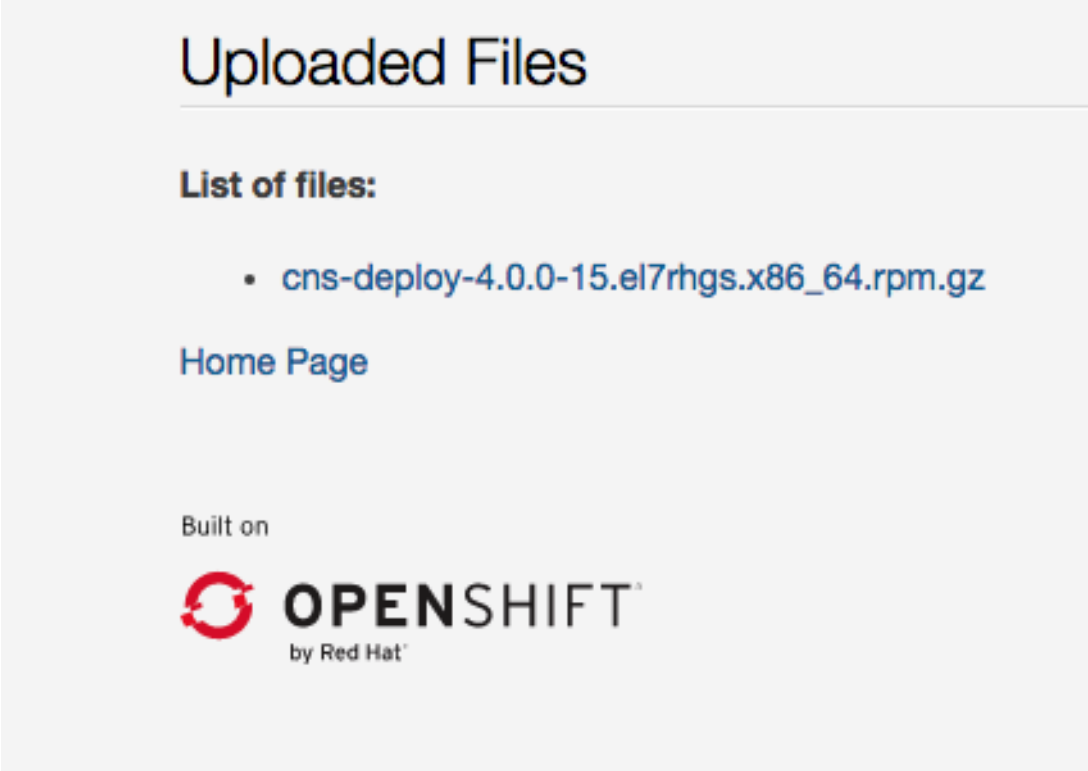
A new deployment is created using the **PVC** and there is a new **demo** pod.

```
$ oc get pods
NAME           READY     STATUS      RESTARTS   AGE
demo-1-build   0/1       Completed   0          6m
demo-2-8366g   1/1       Running     0          35s
```

Now there is a persistent volume allocated using the standard storage class and mounted at **/opt/app-root/src/uploaded** on the **demo-2** pod.

```
$ oc volumes dc demo
deploymentconfigs/demo
  pvc/app (allocated 10GiB) as persistent-volume
    mounted at /opt/app-root/src/uploaded
```

Using the route for the **demo-2** deployment upload a new file (example http://demo-rwo.apps.gce.sysdeseng.com/).



Now login to the pod and validate that the newly uploaded file exists.

On the pod perform the following.

```
$ oc rsh demo-2-8366g
sh-4.2$ df -h
Filesystem      Size  Used Avail Use% Mounted on
...omitted...
/dev/sdd        9.8G   42M  9.2G   1% /opt/app-root/src/uploaded
sh-4.2$ cd /opt/app-root/src/uploaded
sh-4.2$ ls -lh
total 5.6M
-rw-r--r--. 1 1000080000 1000080000 5.6M Sep 12 16:13 heketi-client-4.0.0-
7.el7rhgs.x86_64.rpm.gz
drwxrwS---. 2 root       1000080000  16K Sep 12 16:11 lost+found
```

Because of the use of persistent storage the **demo-2-8366g** pod can be deleted and a new pod will be spun up using the same **PVC**.

CHAPTER 6. EXTENDING THE CLUSTER

By default, the reference architecture playbooks are configured to deploy 3 master, 3 infrastructure, and 3 application nodes. This cluster size provides enough resources to get started with deploying a few test applications or a Continuous Integration Workflow example. However, as the cluster begins to be utilized by more teams and projects, it will be become necessary to provision more application or infrastructure nodes to support the expanding environment. To facilitate easily growing the cluster, the **--scaleup** parameter is available in **ocp-on-gcp.sh** script. It allows scaling up of masters, infrastructure, or application nodes.

6.1. PREREQUISITES FOR ADDING NODES

Verify the quantity and type of the nodes in the cluster by using the **oc get nodes** command. The output below is an example of a complete OpenShift environment after the deployment of the reference architecture environment.

```
$ oc get nodes
NAME                 STATUS                     AGE       VERSION
ocp-infra-node-9kwt  Ready                      11h
v1.6.1+5115d708d7
ocp-infra-node-b0hd  Ready                      11h
v1.6.1+5115d708d7
ocp-infra-node-l2b7  Ready                      11h
v1.6.1+5115d708d7
ocp-master-7k0h      Ready,SchedulingDisabled   11h
v1.6.1+5115d708d7
ocp-master-dpr5      Ready,SchedulingDisabled   11h
v1.6.1+5115d708d7
ocp-master-l9b6      Ready,SchedulingDisabled   11h
v1.6.1+5115d708d7
ocp-node-7d0h        Ready                      11h
v1.6.1+5115d708d7
ocp-node-btcd        Ready                      11h
v1.6.1+5115d708d7
ocp-node-nv8r        Ready                      11h
v1.6.1+5115d708d7
```

6.2. ADDING NODES

To add a node, modify the **config.yaml** file and update the instance group size to the required number of nodes. Multiple nodes of more than one type can be added at once. For example, the default configuration:

```
# How many instances should be created for each group
master_instance_group_size: 3
infra_node_instance_group_size: 3
node_instance_group_size: 3
```

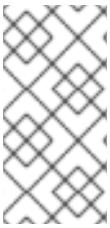
can be updated directly to:

```
# How many instances should be created for each group
master_instance_group_size: 3
infra_node_instance_group_size: 4
```

```
node_instance_group_size: 5
```

Afterwards, running the **ocp-on-gcp.sh** script with the **--scaleup** parameter is required:

```
$ ./ocp-on-gcp.sh --scaleup
```



NOTE

Scaling of pods (router, registry..) running on infrastructure nodes is not performed during the scale up. It is expected that the administrator will perform the scale up once the new nodes are deployed. For example, to scale router run: **oc scale dc/router --replicas=4** from any master node.

6.3. VALIDATING NEWLY PROVISIONED NODES

To verify newly provisioned nodes that have been added to the existing environment, use the **oc get nodes** command:

```
$ oc get nodes
NAME                  STATUS                     AGE       VERSION
ocp-infra-node-87fd   Ready                      42m
v1.6.1+5115d708d7
ocp-infra-node-9kwt   Ready                      13h
v1.6.1+5115d708d7
ocp-infra-node-b0hd   Ready                      13h
v1.6.1+5115d708d7
ocp-infra-node-l2b7   Ready                      13h
v1.6.1+5115d708d7
ocp-master-7k0h       Ready,SchedulingDisabled   13h
v1.6.1+5115d708d7
ocp-master-dpr5       Ready,SchedulingDisabled   13h
v1.6.1+5115d708d7
ocp-master-l9b6       Ready,SchedulingDisabled   13h
v1.6.1+5115d708d7
ocp-node-62h8         Ready                      42m
v1.6.1+5115d708d7
ocp-node-7d0h         Ready                      13h
v1.6.1+5115d708d7
ocp-node-btcd         Ready                      13h
v1.6.1+5115d708d7
ocp-node-d1t3         Ready                      42m
v1.6.1+5115d708d7
ocp-node-nv8r         Ready                      13h
v1.6.1+5115d708d7
```

CHAPTER 7. MULTIPLE OPENSHIFT DEPLOYMENTS

7.1. PREREQUISITES

The prerequisites described in Section 3.1, “Prerequisites for Provisioning” are required when deploying another **OCP** environment into **GCP**. Below is a checklist to perform to prepare for the deployment of another **OCP** cluster.

- Create subdomain
- Map subdomain NS records to root domain
- Configure authentication

7.2. DEPLOYING THE ENVIRONMENT

Using the **ocp-on-gcp.sh** script to deploy another **OCP** cluster is almost exactly the same as defined in Chapter 3, *Deploying OpenShift*. The important difference is to use different **prefix** configuration option in the **config.yaml** file. The **ocp-on-gcp.sh** script has **-c | --config FILE** parameter, which can be used to define different configuration file instead of the default one. Following commands can be used to deploy two OpenShift clusters with different settings in two configuration files.

```
$ ./ocp-on-gcp.sh -c ../config-dev.yaml

$ ./ocp-on-gcp.sh -c ../config-prod.yaml
```

CHAPTER 8. CONCLUSION

Red Hat solutions involving the OpenShift Container Platform are created to deliver a production-ready foundation that simplifies the deployment process, shares the latest best practices, and provides a stable highly available environment on which to run your production applications.

This reference architecture covered the following topics:

- A completely provisioned infrastructure on **GCP**
- OpenShift Container Platform Masters in multiple **Zones**
- OpenShift Container Platform Infrastructure nodes in multiple **Zones** with Router and Registry pods scaled accordingly
- Native integration with **GCP** services like **GCE**, **GCS**, **IAM**, **Regions and Zones**, **Load Balancing and Scaling**, **Google DNS**, **Google OAuth**, and **Custom Images**.
  - **Load Balancing** coupled with **Health Checks** to provide highly availability for Master and Infrastructure instances both internally and externally
  - OpenShift Container Platform console authentication managed by **Google OAuth**
  - **GCS** storage for persistent storage of container images
  - **Solid-State Persistent disks** storage for /var/lib/docker on each node
- Creation of applications
- Validating the environment
- Testing failover

For any questions or concerns, please email refarch-feedback@redhat.com and ensure to visit the Red Hat Reference Architecture page to find about all of our Red Hat solution offerings.

APPENDIX A. CONTRIBUTORS

- Jason DeTiberus, content provider
- Matthew Farrellee, content provider
- Tim St. Clair, content provider
- Rob Rati, content provider
- Andrew Beekhof, review
- David Critch, review
- Eric Jacobs, review
- Scott Collier, review

APPENDIX B. REVISION HISTORY

Revision 3.6.1-0 **Nov 8, 2017** **Peter Schiffer**

- **OpenShift Logging**

Revision 3.6.0-0 **Sep 19, 2017** **Peter Schiffer**

- **OpenShift Release 3.6**

Revision 3.5.2-0 **May 25, 2017** **Peter Schiffer**

- **OpenShift Metrics**
- **Change of demo project name**

Revision 3.5.1-0 **May 10, 2017** **Ryan Cook**

- **Storage Class and persistent storage rewrite**

Revision 3.5.0-0 **May 2, 2017** **Peter Schiffer**

- **OpenShift Release 3.5**
- **Architecture image update**

Revision 3.4.0-0 **Jan 24, 2017** **Chris Murphy, Peter Schiffer**

- **OpenShift Release 3.4**