3Build Book for OpenShift 3.11 Platform Installation on GCP Loblaw projects

# Building Jumpbox

## GCP Preset

1. Create GCP OpenShift projects and their service accounts

There are four OpenShift platforms are built. Each platform is built on its own GCP *project*:

* Sandbox: lt-osp-sbx
* Lower: lt-osp-lwr
* Upper: lt-osp-upr
* DR: lt-osp-dr

GCP admins create GCP projects for OpenShift platforms and one *service account* on each project. Service account is assigned to role **Owner**. All VMs will use this service account in the same project, not use GCP default service account for each VM itself.

* Sandbox: sa-pro-lt-osp-sbx [sa-pro-lt-osp-sbx@lt-osp-sbx.iam.gserviceaccount.com](mailto:sa-pro-lt-osp-sbx@lt-osp-sbx.iam.gserviceaccount.com)
* Lower: sa-pro-lt-osp-lwr [sa-pro-lt-osp-lwr@lt-osp-lwr.iam.gserviceaccount.com](mailto:sa-pro-lt-osp-lwr@lt-osp-lwr.iam.gserviceaccount.com)
* Upper: sa-pro-lt-osp-upr [sa-pro-lt-osp-upr@lt-osp-upr.iam.gserviceaccount.com](mailto:sa-pro-lt-osp-upr@lt-osp-upr.iam.gserviceaccount.com)
* Dr: sa-pro-lt-osp-dr id-[sa-pro-lt-osp-dr@lt-osp-drr.iam.gserviceaccount.com](mailto:sa-pro-lt-osp-dr@lt-osp-drr.iam.gserviceaccount.com)

1. Create subnets and share them to OpenShift projects

GCP network admin creates three *subnets* (master, infra and worker) on network project and shared to OpenShift project. All network settings are created on network projects.

* Sandbox:

Subnet Network IP range

snet-gfsbx-osp-master-nane vpc-gf-pte 10.195.177.0/24

snet-gfsbx-osp-infra-nane vpc-gf-pte 10.195.179.128/25

snet-gfsbx-osp-worker-nane vpc-gf-pte 10.195.179.0/25

* Lower:

Subnet Network IP range

snet-gfpte-osp-lwr-master-nane vpc-gf-pte 10.195.176.0/24

snet-gfpte-osp-lwr-infra-nane vpc-gf-pte 10.195.178.128/25

snet-gfpte-osp-lwr-worker-nane vpc-gf-pte 10.195.178.0/25

* Upper:

Subnet Network IP range

snet-gfpre-osp-master-nane vpc-gf-pre 10.195.48.0/24

snet-gfpre-osp-infra-nane vpc-gf-pre 10.195.50.0/24

snet-gfpre-osp-worker-nane vpc-gf-pre 10.195.49.0/24

* DR:

Subnet Network IP range

snet-gfdr-osp-master-useast vpc-gf-pre 10.195.112.0/24

snet-gfdr-osp-infra-useast vpc-gf-pre 10.195.113.0/24

snet-gfdr-osp-worker-useast vpc-gf-pre 10.195.114.0/24

Sandbox, Lower and Upper platforms are built in GCP region northamerican-northeast1, so we set their subnet names with **nane**.

Dr platform is built in GCP region us-east1, so we set its subnet names with **useast**.

1. Assign service accounts to subnets and Golden Image.

After network admin created three subnets, GCP admin adds service account to three subnets and project **lt-images** in order to service account can fetch image **rhel-latest** as all VMs OS image.

## Create Jumpbox VM

Jumpbox (Red Hat calls it Bastion as well) is a non-OpenShift VM accessible from outside of the Red Hat OpenShift Container Platform environment, configured to allow remote access via secure shell (ssh). The system administrator first accesses to the Jumpbox, then “jumps” via another ssh connection to the intended OpenShift VM.

Moreover, jumpbox is an ideal CLI console for running administrative tasks such as the Red Hat OpenShift Container Platform installation playbooks and OCP management commands.

Each OpenShift project has its jumpbox. So we have below Jumpbox VMs:

* Sandbox: vlospsb001-j 10.195.177.253 northamerica-northeast1-a
* Lower: vlospdv001-j 10.195.176.253 northamerica-northeast1-a
* Upper: vlosppr001-j 10.195.48.253 northamerica-northeast1-a
* DR: vlosppr011-j 10.195.112.253 us-east1-d

Below is the example of creating Lower jumpbox VM detail:

Log on to GCP Lower project,

Name: *vlopsdv001-j*

Region: *northamerica-northeast1* (Montreal) DR Regionis

Zone: *northamerica-northeast1-a*

Machine type: *n1-standard-4*

Boot disk: Change > Custom images > Show image from lt-images > *rhel-latest*

Boot disk type: *Standard persistent disk*

Size (GB): *40*

Service account: *sa-pro-lt-osp-lwr*

Network tags: *allow-internet, ltosplwrjocp*

Hostname: *vlospdv001-j.ltosplwr.osp.ngco.com*

Network interface: Networks shared with me (from host project “lcl-net-svc”)

Shared subnetwork: *snet-gfpte-osp-lwr-master-nane*

Primary interface IP: Reserve a static internal IP address

Name: *vlospdv001-j*

Static IP address: Let me choose

Custom IP address: *10.195.176.253*

Click **Create** button.

Jumpbox vlospdv001-j is created.

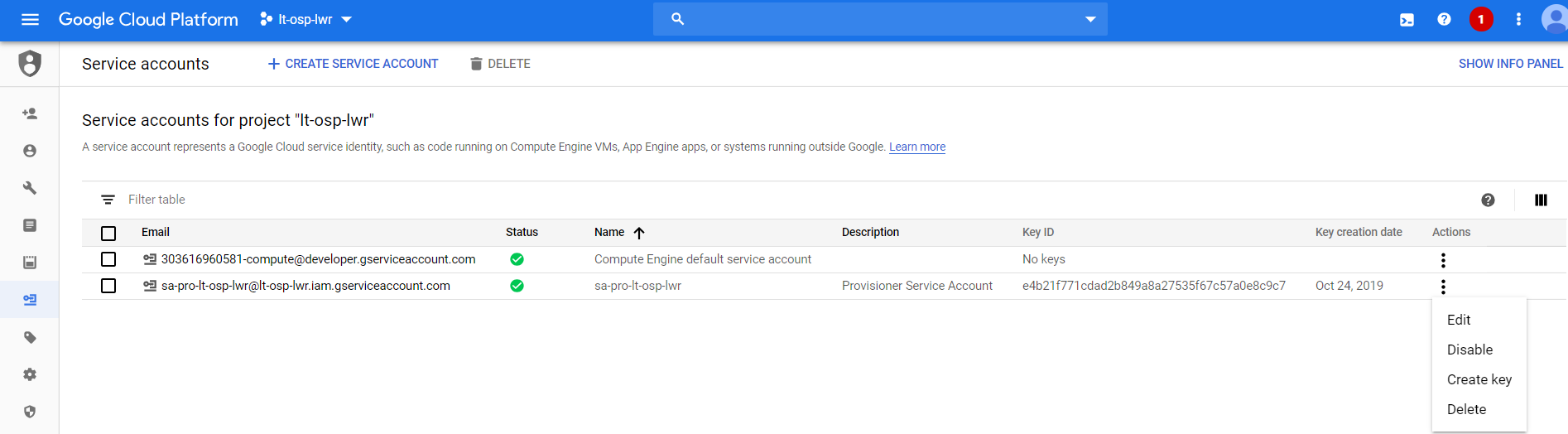
## Install Software on Jumpbox

1. Activate Service Account

On GCP console, select the OpenShift project, such as lt-osp-lwr.

IAM & admin > Service accounts

At the Action of the service account, select **Create key**,then save the key in a json file, such as **lt-osp-lwr-e4b21f771cda.json** on your laptop.



Log on to the new jumpbox with user **lcluser01**.

Mkdir two folders: /home/lcluser01/<openshift\_platform\_name>, and /home/lcluser01/<openshift\_platform\_name>/files.

Save service account key file in folder /home/lcluser01/<openshift\_platform\_name>/files.

<openshift\_platform\_name> for all OpenShift platforms are listed:

* Sandbox: ltospsbx
* Lower: ltosplwr
* Upper: ltospupr
* DR: ltospdr

Activate service account with below command:

$ *gcloud auth activate-service-account* ***<service account name>*** *--key-file=****<service account key>*** *--project=****<project>***

For example:

$*gcloud auth activate-service-account sa-pro-lt-osp-lwr@lt-osp-lwr.iam.gserviceaccount.com --key-file=/home/lcluser01/ltosplwr/files/ lt-osp-lwr-e4b21f771cda.json --project=lt-osp-lwr*

Activated service account credentials for: [sa-pro-lt-osp-lwr@lt-osp-lwr.iam.gserviceaccount.com]

To verify the service account:

[root@vlospdv001-j lcluser01]$ *gcloud auth list*

Credentialed Accounts

ACTIVE ACCOUNT

\* sa-pro-lt-osp-lwr@lt-osp-lwr.iam.gserviceaccount.com

To set the active account, run:

$ gcloud config set account `ACCOUNT`

To take a quick anonymous survey, run:

$ gcloud alpha survey

1. Generate ssh-key pair

We need to create a ssh-key pair for us from jumpbox to other OpenShift VMs in the future.

On jumpbox VM,

$*sudo –i*

#*ssh-keygen*

This will generate a ssh-key pair in the folder /root/.ssh. The pair files are id\_rsa and id\_rsa.pub.

**Id\_rsa.pub** will be distributed to other OpenShift VMs for admin to access to them from jumpbox.

1. Install software packages

In order to get Red Hat support and download software from Red Hat in the future, we need to setup Red Hat Subscription Manager on jumpbox.

# *subscription-manager register --org=2932735 --activationkey=ocp-master-infra*

The system has been registered with ID: ########-####-####-####-############

The registered system name is: vlospdv001-j

Installed Product Current Status:

Product Name: Red Hat Enterprise Linux Server

Status: Subscribed

# *subscription-manager repos --enable="rhel-7-server-rpms" --enable="rhel-7-server-extras-rpms" --enable="rhel-7-server-ose-3.11-rpms" --enable="rhel-7-server-ansible-2.6-rpms"*

Repository 'rhel-7-server-rpms' is enabled for this system.

Repository 'rhel-7-server-extras-rpms' is enabled for this system.

Repository 'rhel-7-server-ansible-2.6-rpms' is enabled for this system.

Repository 'rhel-7-server-ose-3.11-rpms' is enabled for this system.

Install the following packages on the jumpbox:

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

1. Setup Ansible Environment

Make the folders as below tree and copy the files from right BitBucket repos to the right folders.

[root@vlospdv001-j ltosplwr]# *tree*

.

├── **ansible.cfg**

├── files

│   ├── certs

│   │   ├── **ldap-tls-ca-bundle.pem**

│   │   ├── **digicert01.cer**

│   │   ├── **digicertroot.cer**

│   │   ├── **ltosplwr\_apps.cer**

│   │   ├── **ltosplwr\_apps.key**

│   │   **├── ltosplwr\_console.cer**

│   │   └── **ltosplwr\_console.key**

│   └── **lt-osp-lwr-e4b21f771cda.json**

├── inventory

│   ├── group\_vars

│   │   └── OSEv3

│   │   └── **oreg\_auth.yaml**

│   └── **hosts**

├── playbooks

│   ├── **gcp\_creation.yml**

│   ├── **master\_ldap.yml**

│   └── **setup\_golden\_image\_outside\_rhsm.yml**

├── scripts

│   ├── **ldap\_sync.yaml**

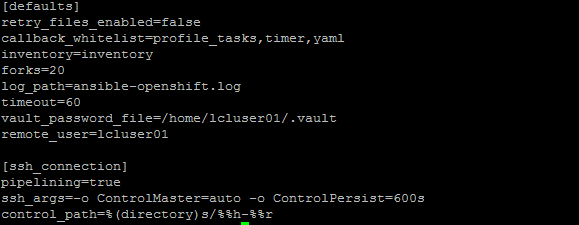
│   └── **white\_list**

└── templates

If tree command is not available, install it with command:

#*yum –y install tree*

Ansible environment configuration file is **/home/lcluser01/<Platform\_name>/ansible.cfg**. It shows below:



Notes: Make sure **vault\_password\_file** is in the right folder and exists.

# Create OpenShift Platform VMs

## Create Storage Bucket

Prepare storage bucket on GCP OpenShift project for OpenShift registry.

Enter GCP project CLI console and run below commands:

gsutil mb -c regional -l <Region> gs://<Bucket\_name>

cat <<EOF > labels.json

{

"ocp-cluster": "Label"

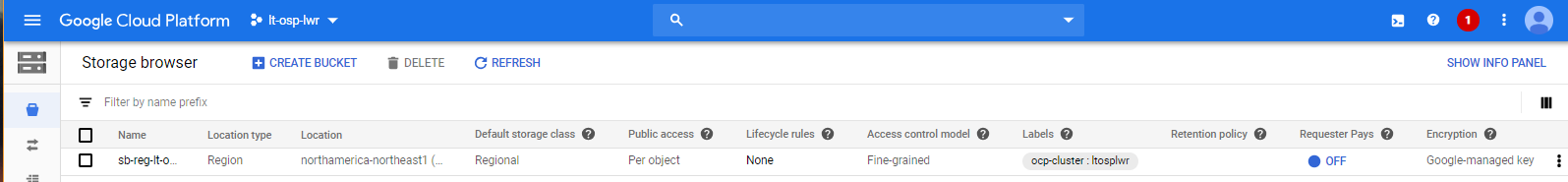
}

EOF

gsutil label set labels.json gs://<Bucket\_name>

rm -f labels.json

Storage bucket is created in the OpenShift project on GCP. Storage bucket name much be unique globally.



Platform Bucket\_name Region Label

* Sandbox: sb-reg-lt-osp-sbx nothamerica-northeast1 ocp-cluster:ltospsbx
* Lower: sb-reg-lt-osp-lwr nothamerica-northeast1 ocp-cluster:ltosplwr
* Upper: sb-reg-lt-osp-upr nothamerica-northeast1 ocp-cluster:ltospupr
* DR: sb-reg-lt-osp-dr us-east1 ocp-cluster:ltospdrocp

## Create OpenShift nodes (VMs on GCP)

We run Ansible playbook to create OpenShift nodes (VMs on GCP). The playbook is **/home/lcluser01/<openshift\_name>/playbooks/gcp\_creation.yml**. The VM creation playbooks have been saved to BitBucket fro This playbook defines all nodes’ configuration, and includes VM IP creation and VM creation commands.

**Sandbox ltospsbx**



**Lower ltosplwr**



**Upper ltospupr**

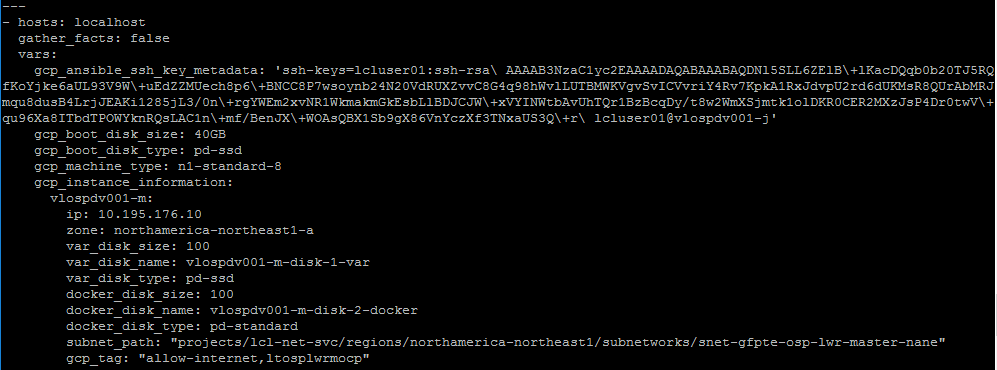


**Dr ltospdr**

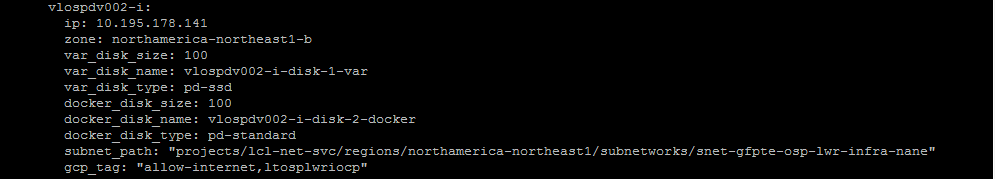


Different OpenShift platforms have different numbers of Master nodes, Infra nodes and worker nodes. Reference below tables for each OpenShift platform to set the variables in the playbook **/home/lcluser01/<openshift\_name>/playbooks/gcp\_creation.yml**.

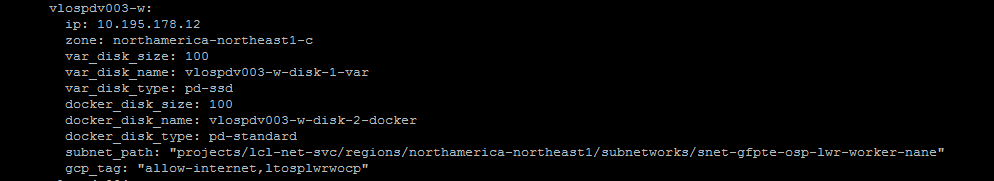
The playbook **gcp\_creation.yml** is listed below. We set example from ltosplwr.



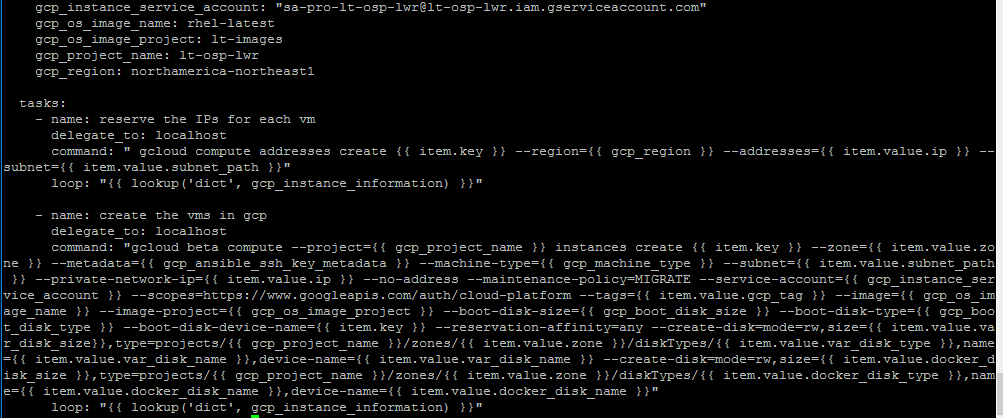
Skip list of Master nodes vlospdv002-m and vlospdv003-m.



Skip list of Infra nodes vlospdv001-i and vlospdv003-i.



Skip list Worker nodes vlospdv001-w, vlospdv002-w, vlospdv004-w, vlospdv005-w and vlospdv006-w.



Notes:

* The string following variable **gcp\_ansible\_ssh\_key\_metadata: 'ssh-keys=lcluser01:** is the content of ssh public key id\_rsa.pub generated at 1.3.b, only add “\” in front of characters “+” or “ “. Change root to be lcluser01. From jumpbox access to other nodes with user lcluser01, use command **ssh** [**lcluser01@vlosplwr001-w.ltosplwr.osp.ngco.com**](mailto:lcluser01@vlosplwr001-w.ltosplwr.osp.ngco.com) without user password.
* Master, Infra and Worker nodes have the same number of variables. According to the inventory sheet to set each variable. Pay attention to **IP, zone, subnet\_path and gcp\_tag**.
* In the tasks, the first command is reserve IPs for each VM and the second command is create the VMs in GCP.

Under /home/lcluser01/ltosplwr folder to run ansible playbook to create OpenShift nodes on GCP.

[root@vlospdv001-j ltosplwr]# *ansible-playbook ./playbooks/gcp\_creation.yml*

**Send network team a request to setup DNS for all new OpenShift nodes.**

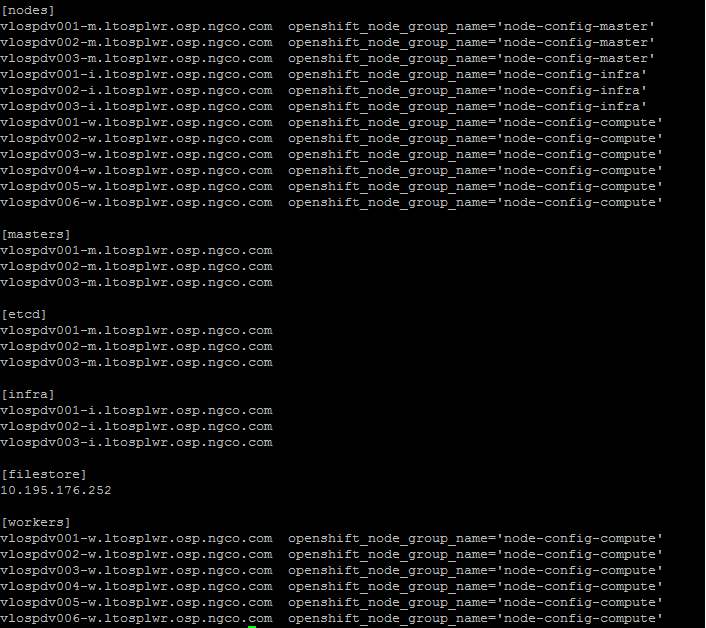
## Configure VMs

After all OpenShift nodes have been created on GCP, we need to configure the nodes as OpenShift requirement, such as:

* Create /var/lib/etcd file system for Master nodes
* Create /var/lib/origin/openshift.local.volumes file system.
* Create /var/log file system.
* Red Hat Subscription Manager and Repo
* Network Manager

The VMs’ configuration playbook is **/home/lcluser01/<openshift\_name>/playbooks/****setup\_golden\_image\_outside\_rhsm.yml*.*** Download it from OpenShift BitBucket.

Before we run the VMs’ configuration playbook, we need to list all VMs in **/home/lcluser01/<openShift\_name>/inventory/hosts** as below:



Notes:

* There are four OSEv3 groups in **hosts** file: **master, etcd, infra, and nodes**. Master and etcd groups have the same VMs. Group nodes covers all OpenShift nodes.
* Ignore filestore and workers here. They are not OSEv3 groups. Filestore is a backup data store VM.

Run below command on jumpbox to configure all new OpenShift nodes.

[root@vlospdv001-j ltosplwr]# *ansible-playbook ./playbooks/setup\_golden\_image\_outside\_rhsm.yml*

## Correct Golden Image Configuration

Because Golden Image is a usual purpose Linux image, two configuration files much be updated **on all OpenShfit VMs** to meet OpenShift nodes requirement.

1. **/etc/logrotate.d/syslog**

The file **syslog** should be the same as below. Three yellow lines are added. This will resolve file system **/var/log** full issue.

/var/log/cron  
/var/log/maillog  
/var/log/messages  
/var/log/secure  
/var/log/spooler  
 {  
  daily  
  rotate 7  
  compress  
  sharedscripts  
  missingok  
  postrotate  
    /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true  
  endscript  
  }

1. **/etc/sysconfig/docker**

On each OpenShift VM, run

#*systemctl stop docker*

Replace the OPTIONS line in /etc/sysconfig/docker file with below line:

*OPTIONS='--selinux-enabled --signature-verification=false --insecure-registry 10.189.129.7:8082 --log-opt max-size=50m --log-opt max-file=5'*

Run command

#systemctl start docker

This update can avoid **/var/lib/docker** file system full issue.

This part will be reviewed in Nexus.

# Building OpenShift Platform

## Firewall rules and Load Balancers

1. Firewall rules creation

Send request to network team to create firewall rules for the OpenShift platform.

Normally, two Ingress firewall rules will be created for each OpenShift platform: One is for tcp and another is for udp. Each firewall rule has **four targets** (tags: jumpbox, master, infra and worker) and **three IP ranges** (subnets: master, infra and worker).



For Lower and Upper OpenShift platforms, there are two Egress firewalls as well. That is for Netbackup. One is for tcp and another is for udp.

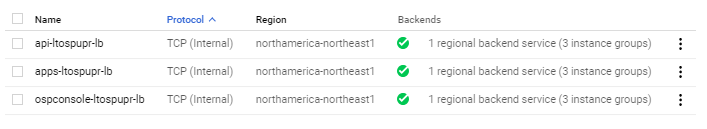


1. Load Balancers creation

Send a request to network admin to create three load balancers for each OpenShift platform.

* api-<platform\_name>-lb
* ospconsole-<platform\_name>-lb
* apps-<platform\_name>-lb

Each load balancer has three instance groups. api-<platform\_name>-lb and ospconsole-<platform\_name>-lb have master VMs instance groups as backends. apps-<platform\_name>-lb has three infra VMs instance groups as backends.



Make sure all backends should be green normally.

**Sandbox**



**Lower**



**Upper**



**DR**

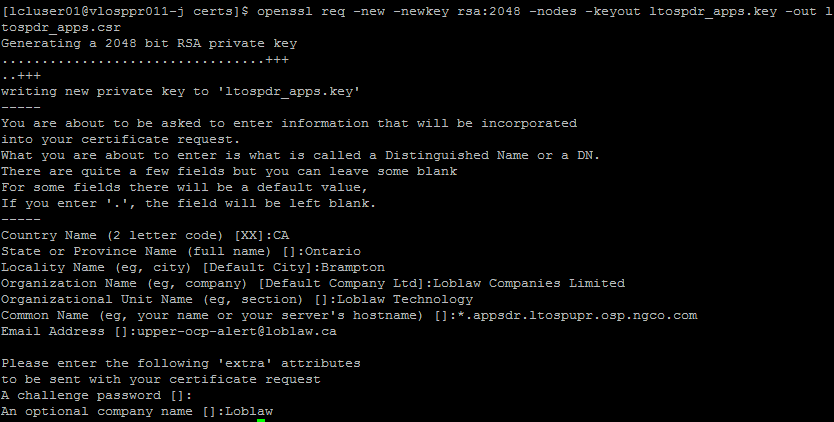


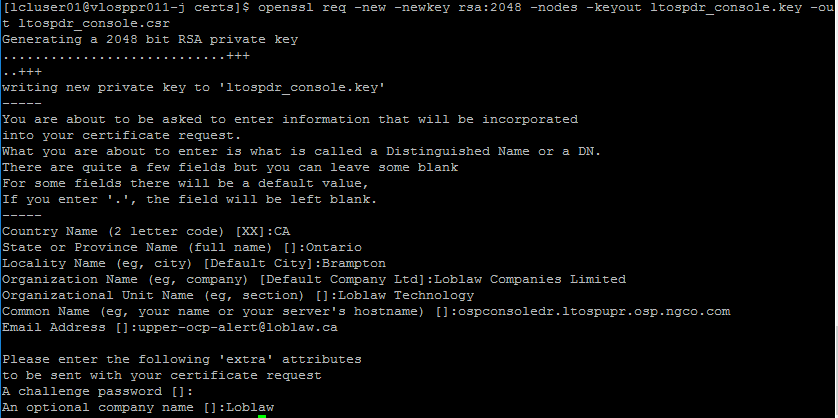
1. Certificates of Console and Apps

When we have load balancers information, we should apply certificates for two load balancers: **ospconsole.<platform>.osp.ngco.com** and **\*.apps.<platform>.osp.ngco.com**.

For each load balancer, we will create its .csr and .key files.

Below is an example with DR OpenShift platform. DR platform requires certificates \*appsdr.ltospupr.osp.ngco.com and ospconsoledr.ltospupr.osp.ngco.com.





Four files will be generated:

* **ltospdr\_apps.csr**
* **ltospdr\_apps.key**
* **ltospdr\_console.csr**
* **ltospdr\_console.key**

Send two .csr files to security team, and wait for a while, security team will send back two certificate .cer files.

* **ltospdr\_apps.cer**
* **ltospdr\_console.cer**

Copy two .cer files and two .key files to folder **/home/lcluser01/<platform>/files/certs**.

Download LDAP certificate file **ldap-tls-ca-bundle.pem** from BitBucket and save it to folder **/home/lcluser01/<platform>/files/certs** as well.

Edit file **/home/lcluser01/<platform>/inventory/hosts** and make sure it includes this section:

# Custom Certificates

openshift\_master\_overwrite\_named\_certificates=true

openshift\_master\_named\_certificates=[{"certfile": "/home/lcluser01/ltospdr/files/certs/**ltospdr\_console.cer**", "keyfile": "/home/lcluser01/ltospdr/files/certs/**ltospdr\_console.key**", "names": ["ospconsoledr.ltospupr.osp.ngco.com"], "cafile": "/home/lcluser01/ltospdr/files/certs/**ldap-tls-ca-bundle.pem**"}]

openshift\_hosted\_router\_certificate={"certfile": "/home/lcluser01/ltospdr/files/certs/**ltospdr\_apps.cer**", "keyfile": "/home/lcluser01/ltospdr/files/certs/**ltospdr\_apps.key**", "cafile": "/home/lcluser01/ltospdr/files/certs/**ldap-tls-ca-bundle.pem**"}

1. DNS for three load balancers

Send a request to network team for setting up DNS for three load balancers.

After get the confirmation from network team, make sure the DNS settings are correct with command **nslookup** six times:

* *nslookup ospconsole.<platform>.osp.ngco.com*
* *nslookup <IP of ospconsole.<platform>.osp.ngco.com>*
* *nslookup api.<platform>.osp.ngco.com*
* *nslookup <IP of api.<platform>.osp.ngco.com>*
* *nslookup \*.apps.<platform>.osp.ngco.com*
* *nslookup <IP of \*.apps.<platform>.osp.ngco.com>*

## OpenShift Prerequistes

1. Install OpenShift packages to all OpenShift nodes

On jumpbox, run below ansible command to install OpenShift prerequisite packages to all OpenShift nodes.

*ansible all -m shell -a "yum install -y wget git net-tools bind-utils yum-utils iptables-services bridge-utils bash-completion kexec-tools sos psacct* *"*

1. Reboot all nodes

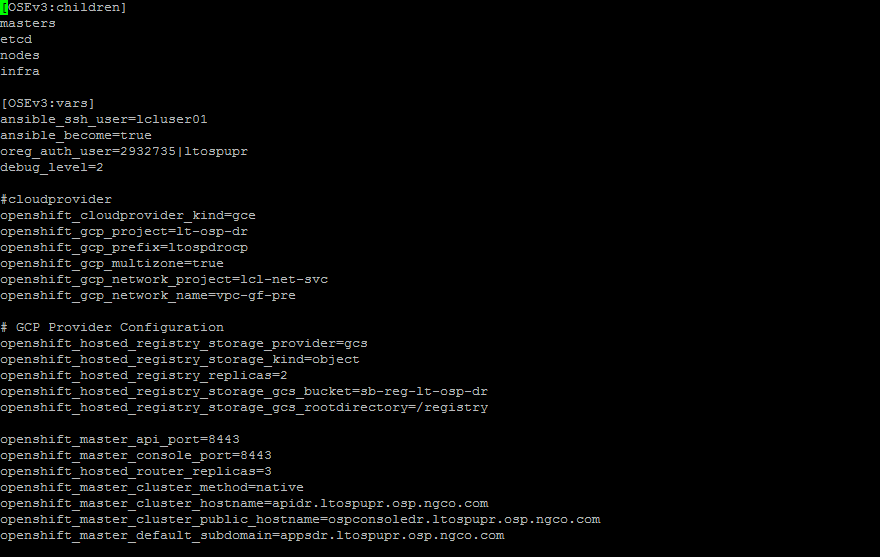
After above configuration change and software package installation on all OpenShift nodes, their status may be not the same or clean. Reboot them all is a best practices.

On jumpbox, run below ansible command to reboot all OpenShift nodes

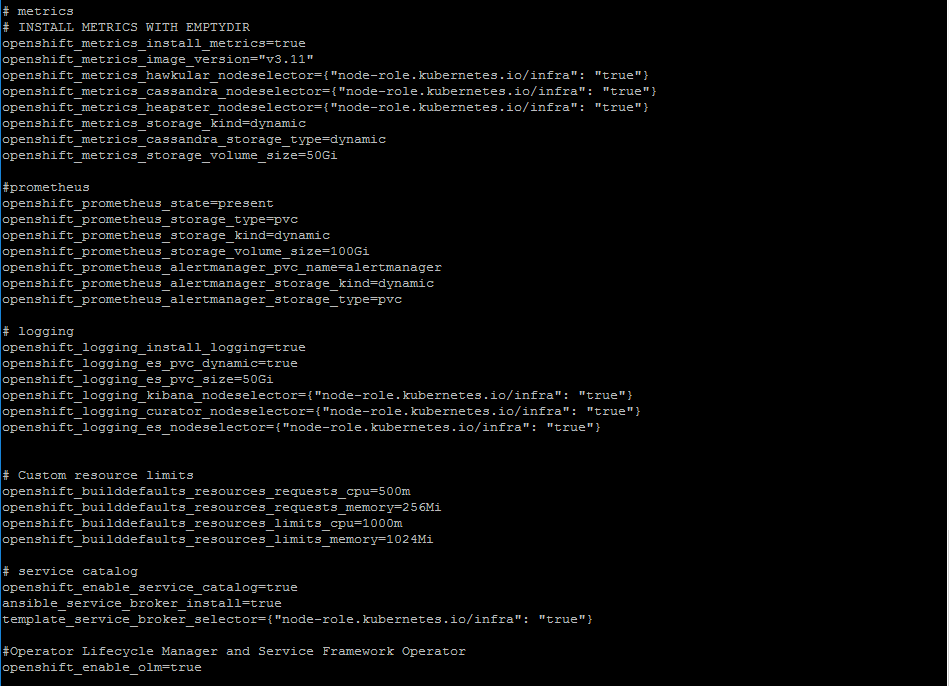
*ansible all –a 'reboot'*

1. File hosts

The file **/home/lcluser01/<platform>/inventory/hosts** is a key file in OpenShift platform prerequisites and deployment.







Notes:

* Sections **Nodes** and **Custom Certifications** in the **hosts** file has been introduced previously. We do not list here.
* Variable **oreg\_auth\_user**=2932735|XXXXXXX is Red Hat service account for an OpenShift platform. Will be introduced in next section.
* Variable **debug\_level** is set to 2. In higher debug\_level, OpenShift will generate more detail logs. Debug\_level is too high, nodes log folder will be full quickly.
* Variable **openshift\_gcp\_prefix**=ltospdrocp is the storage bucket label. It relates with other two variables: **openshift\_hosted\_registry\_storage\_gcs\_bucket**=**sb-reg-lt-osp-dr** and **openshift\_hosted\_registry\_storage\_gcs\_rootdirectory**=/**registry**. OpenShift registryuses storage space in storage bucket.
* Section **Deployment type and package versions** defines the OpenShift version and release. Normally, we will install the latest version image.
* Section **Docker Storage configuration** tells which physical disk will be used for OpenShift Docker on all nodes.
* For the best practices, sections **metrics, Prometheus and logging** will be commented out for the first platform deploy process because they are easy to be failed. We normally run more one time deploy process for these three sections.

1. Organization authority file oreg\_auth.yml

As in **hosts** notes said, **orge\_auth\_user** is Red Hat service account for OpenShift.

We have two Red Hat service accounts:

**2932735|ltospupr** is for Upper and Dr platforms **ltospupr** and **ltospdr**.

**2932735|ltosplwsb** is for Lower and Sandbox platforms **ltosplwr** and **ltospsbx**.

**2932735** is Loblaw organization number in Red Hat repository.

Manager applied a new Red Hat Service Account and got a password **string**. Save this string

Create file **oreg\_auth.yml** in folder **/home/lcluser01/<platform>/inventory/group\_vars/OSEv3**

#*ansible-vault view oreg\_auth.yml --vault-password-file=/root/.vault*

*oreg\_auth\_password:* ***<string\_content>***

Paste the password string content got from Red Hat after oreg\_auth\_password:

1. Prerequisites checking

After above all variables are set in configuration files, we will verify all of them are set correctly and not miss anything by running Ansible playbook **prerequistes.yml** provided by Red Hat for OpenShift build on jumpbox.

#*cd /home/lcluser01/<platform>*

#*ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/prerequisites.yml*

If there is anything fails, correct it and rerun prerequistes.yml till everything is successful.

## OpenShift Deploy\_cluster

1. First deploy\_cluster

After running prerequistes.yml successfully, comment out (not delete) three sections, **metrics, Prometheus and logging** in the file **/home/lcluser01/<platform>/inventory/hosts**. Then we can run the first Ansible playbook **deploy\_cluster.yml** on jumpbox.

#*ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/deploy\_cluster.yml*

If there are any errors, for some simple errors, we only need to update the file **/home/lcluser01/<platform>/inventory/hosts**, then rerun prerequisties.yml and deploy\_cluster.yml.

But for some serious errors, we must remove all OpenShift configuration in all OpenShift nodes. In this situation, we need to run **uninstall.yml** to do it on jumpbox.

#*ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml*

After removing the mistakes, we rerun prerequisties.yml and first time deploy\_cluster.yml.

1. Second deploy\_cluster

After running first time deploy \_cluster.yml successfully, remove the comment signs for three sections, **metrics, Prometheus and logging** in the file **/home/lcluser01/<platform>/inventory/hosts**. Then run the second time deploy\_cluster.yml on jumpbox.

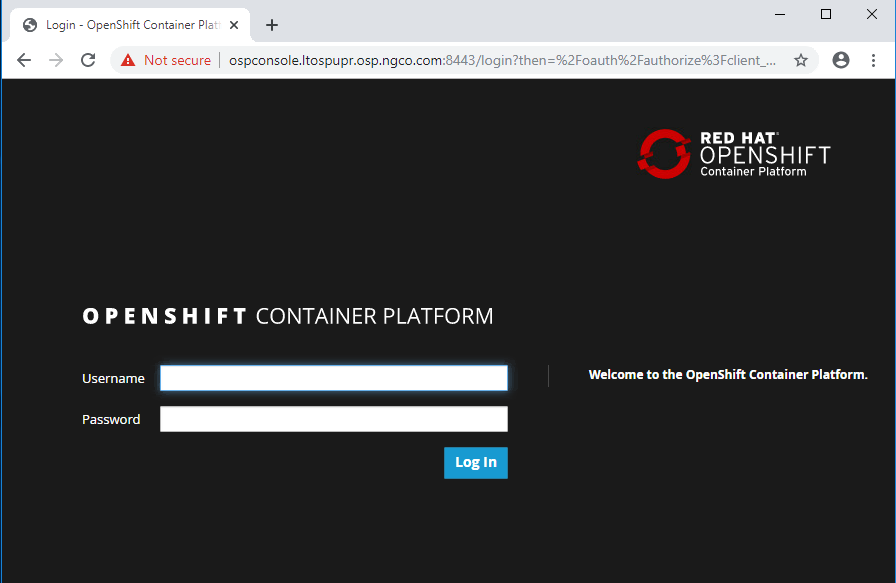
#*ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/deploy\_cluster.yml*

The second deploy\_cluster.yml process will spend much shorter time for only three sections deploy on all OpenShift nodes.

1. Verify the OpenShift Platform Build Is Successful

Log on to the Citrix and start a web browser, such as Google Chrome.

Open URL https://ospconsole.<platform>.osp.ngco.com:8443



Use admin and its default password to open OpenShift console GUI, and can change consoles among **Service Catalog, Application Console and Cluster Console**.

# Integration

## 4.1. LDAP & RBAC

1. admin account

In a new OpenShift platform user **admin** account does not exist. We must add it as a cluster-admin first.

Log on to one of the Master nodes in root. You will have permission to run OpenShift command **oc**. We cannot run oc command on Jumpbox with user admin now.

If you cannot run oc command on the Master node, please install it software package first

#*yum whatprovides oc*

#*yum install –y atomic-openshift-clients*

[root@vlosppr001-m ~]# *oc adm policy add-cluster-role-to-user cluster-admin admin*

cluster role "cluster-admin" added: "admin"

After user admin is added into platform on Master node, we are able to run oc command on Jumpbox to manage OpenShift platform with admin ID.

1. LDAP

Apply service accounts for accessing LDAP from security team. We can do this before platform installation.

Platform Service Account

Sandbox ltospsbx

Lower ltosplwr

Upper & Dr ltospupr

As 1.3.d, we can download file **master\_ldap.yml** to /home/lcluser01/<platform>/playbooks folder.

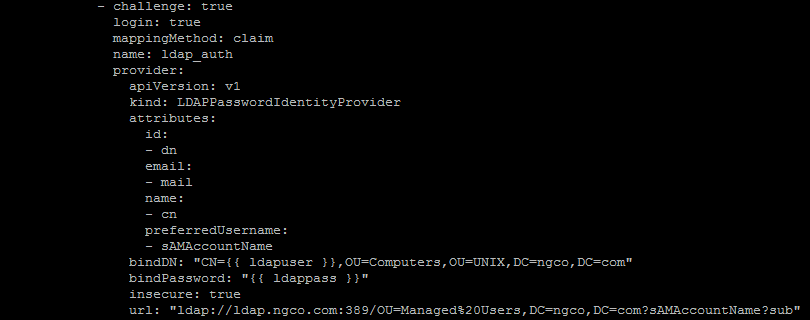
Update **ldapuser** to above service account in file master\_ldap.yml for different platforms:

There are two modes to communicate with LDAP system: Secure mode and Insecure mode. Because our LDAP environment limitation, we can only set Insecure mode now.

Insecure Mode, LDAP URL is "ldap://ldap.ngco.com:389/OU=Managed%20Users,DC=ngco,DC=com?sAMAccountName?sub"

Secure Mode, LDAP URL is "ldap://ldap.ngco.com:636/OU=Managed%20Users,DC=ngco,DC=com?sAMAccountName?sub"

This playbook actually copy below block after **identityProviders:** in file **/etc/origin/master/master-config.yaml** on each master node in the platform.



Run this command on jumpbox as root.

[root@vlosppr011-j ltospdr]# *ansible-playbook ./playbooks/master\_ldap.yml -e "ldappass=<PASSWORD>"*

1. RBAC

Install oc software to jumpbox like the Master node.

#*yum install –y atomic-openshift-clients*

Copy files **ldap\_sync.yaml** and **white\_list** to folder **/home/lcluser01/ltospdr/scripts.**

The **ldap\_sync.yaml** file content is listed below:

kind: LDAPSyncConfig

apiVersion: v1

url: ldap://ldap.ngco.com:389

insecure: true

bindDN: "CN=<platform>,OU=Computers,OU=UNIX,DC=ngco,DC=com"

bindPassword: <ldap\_password>

augmentedActiveDirectory:

groupsQuery:

baseDN: "OU=Cloud,OU=Groups,DC=ngco,DC=com"

scope: sub

derefAliases: never

pageSize: 0

groupUIDAttribute: dn

groupNameAttributes: [ dn ]

usersQuery:

baseDN: "OU=Managed Users,DC=ngco,DC=com"

scope: sub

derefAliases: never

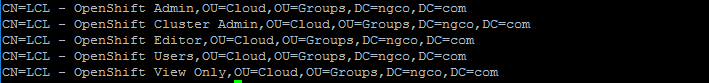
filter: (&(objectclass=user)(memberof=CN=LCL - OpenShift View Only,OU=Cloud,OU=Groups,DC=ngco,DC=com))

pageSize: 0

userNameAttributes: [ sAMAccountName ]

groupMembershipAttributes: [ memberOf ]

The **white\_list** file content is listed below:



Pay attention that filter variable in ldap\_sync.yaml must matches with group in file white\_list. Normally, only the group need to be updated will be in white\_list file. So when you add different app’s groups into OpenShift, in white\_list file, it is better only include the group need to be updated. ldap\_sync.yaml filter needs to be updated as well.

Change folder to /home/lcluser01/<platform>/scripts

[root@vlosppr011-j scripts]# *oc login https://ospconsoledr.<platform>.osp.ngco.com:8443*

[root@vlosppr011-j scripts]# *oc get groups*

No resources found.

[root@vlosppr011-j scripts]#*oc adm groups sync --sync-config=ldap\_sync.yaml --whitelist=white\_list --confirm*

group/CN=LCL - OpenShift Admin,OU=Cloud,OU=Groups,DC=ngco,DC=com

group/CN=LCL - OpenShift Cluster Admin,OU=Cloud,OU=Groups,DC=ngco,DC=com

group/CN=LCL - OpenShift Editor,OU=Cloud,OU=Groups,DC=ngco,DC=com

group/CN=LCL - OpenShift Users,OU=Cloud,OU=Groups,DC=ngco,DC=com

group/CN=LCL - OpenShift View Only,OU=Cloud,OU=Groups,DC=ngco,DC=com

[root@vlosppr011-j scripts]# *oc get groups*

NAME USERS

CN=LCL - OpenShift Admin,OU=Cloud,OU=Groups,DC=ngco,DC=com hprajap, hlai, ajemath, amirazi

CN=LCL - OpenShift Cluster Admin,OU=Cloud,OU=Groups,DC=ngco,DC=com hprajap, hlai, ajemath, amirazi

CN=LCL - OpenShift Editor,OU=Cloud,OU=Groups,DC=ngco,DC=com hprajap, hlai, ajemath, amirazi

CN=LCL - OpenShift Users,OU=Cloud,OU=Groups,DC=ngco,DC=com hprajap, hlai, ajemath, amirazi

CN=LCL - OpenShift View Only,OU=Cloud,OU=Groups,DC=ngco,DC=com hprajap, hlai, ajemath, amirazi

Add groups to roles.

***#oc adm policy add-cluster-role-to-group cluster-admin "CN=LCL - OpenShift Cluster Admin,OU=Cloud,OU=Groups,DC=ngco,DC=com"***

cluster role "cluster-admin" added: "CN=LCL - OpenShift Cluster Admin,OU=Cloud,OU=Groups,DC=ngco,DC=com"

***#oc adm policy add-cluster-role-to-group edit "CN=LCL - OpenShift Editor,OU=Cloud,OU=Groups,DC=ngco,DC=com"***

cluster role "edit" added: "CN=LCL - OpenShift Editor,OU=Cloud,OU=Groups,DC=ngco,DC=com"

***#oc adm policy add-cluster-role-to-group view "CN=LCL - OpenShift View Only,OU=Cloud,OU=Groups,DC=ngco,DC=com"***

cluster role "view" added: "CN=LCL - OpenShift View Only,OU=Cloud,OU=Groups,DC=ngco,DC=com"

Three groups are set to three roles, admin, edit and view.

We can verify the group settings with command:

#*oc get clusterrolebindings |grep groupname*

Log on to OpenShift console your user ID to verify you have been added into OpenShift console GUI.

1. Self-provisioner role

Not allow the initial user create project.

#*oc adm policy remove-cluster-role-from-group self-provisioner system:authenticated:oauth*

Prevent OpenShift Updateds to re-add self-provisioner permissions

# *oc patch clusterrolebinding.rbac self-provisioners -p '{"subjects": null}'*

clusterrolebinding.rbac.authorization.k8s.io/self-provisioners patched

Ensure that the role binding does not auto-update, setting the self-provisioner permission back:

# *oc annotate clusterrolebinding.rbac self-provisioner 'rbac.authorization.kubernetes.io/autoupdate=false' --overwrite*

clusterrolebinding.rbac.authorization.k8s.io/self-provisioner annotated

## 4.2. SMTP

We update SMTP setting on OpenShift is in order to let OpenShift send out its alert emails to the right user group.

Log on to the platform jumpbox and change to folder /home/lcluser01/<platform>

$*oc login*

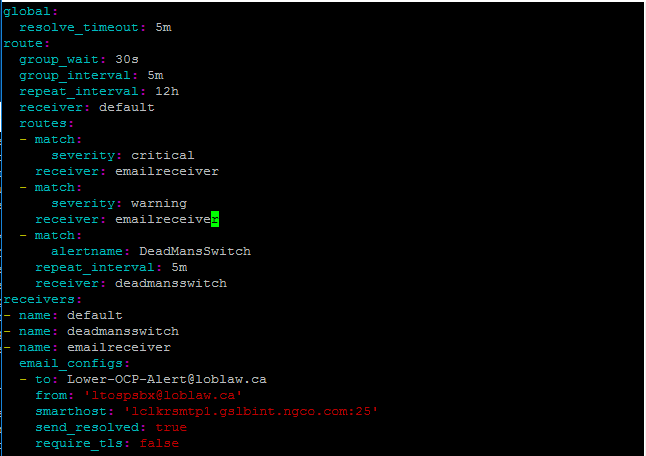
$*oc project default*

##Only get data.alertmanager.yaml secret in project openshift-monitoring and decode it to file alertmanager.yaml

$ *oc get secrets -n openshift-monitoring -o go-template='{{ index .data "alertmanager.yaml"}}' alertmanager-main | base64 -d > alertmanager.yaml*

##Add email configuration in alertmanager.yaml file.

$*vi alertmanager.yaml*



**ltospsbx** and **ltosplwr** alert emails will be sent to [**Lower-OCP-Alert@loblaw.ca**](mailto:Lower-OCP-Alert@loblaw.ca);

**ltospupr** and **ltospdr** alert emails will be sent to [**Upper-OCP-Alert@loblaw.ca**](mailto:Upper-OCP-Alert@loblaw.ca).

##base64-Encoded the yaml file

$ *base64 -w0 alertmanager.yaml*

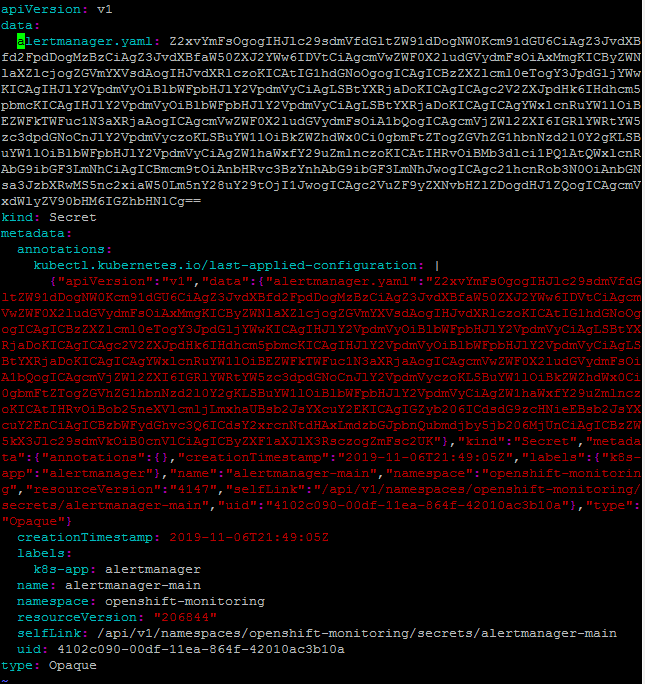


## Reget the alermanager-main secret to new-alertmanager.yaml

$ *oc get secrets -n openshift-monitoring -o yaml alertmanager-main > new-alertmanager.yaml*

##Replace the new alertmanager secret to data:alertmanager.yaml field in new-alertmanager.yaml

$*vi new-alertmanager.yaml*

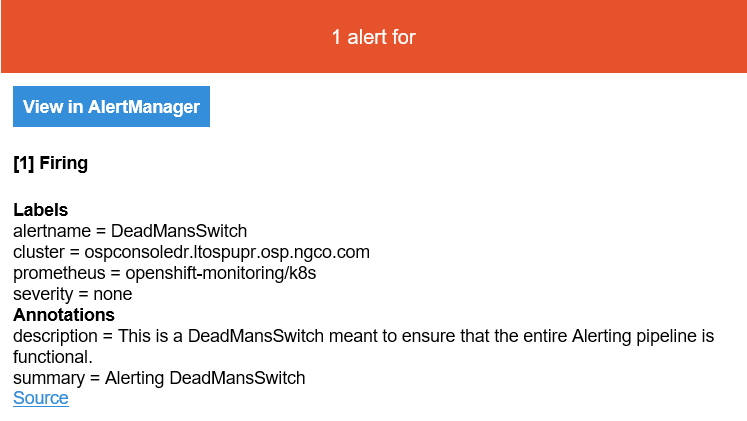


##Replace project openshift-monitoring config with new values

$ *oc replace -n openshift-monitoring -f new-alertmanager.yaml*

secret/alertmanager-main replaced

**DeadMansSwitch** is not an alert. It is to ensure that the entire Alerting pipeline is functional.



## BitBucket

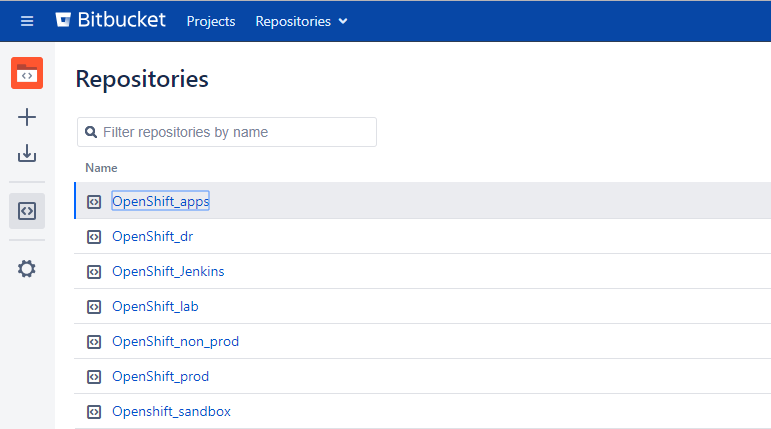
OpenShift platforms communicate with BitBucket for two purposes?

1. Save platform configuration scripts

Saving platform configuration scripts in BitBucket in case the need for fast recovery the platforms. We can download the scripts when we restore the platform configuration.

The BitBucket URL is <https://bitbucket.loblaw.ca/>

Under BitBucket OpenShift project, there are multiple repositories. Different platforms save their scripts to different repositories.



Platform Repository

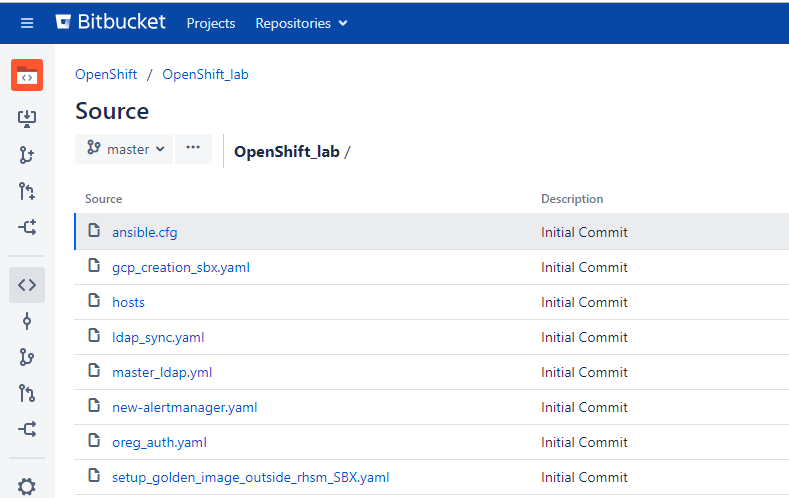
Sandbox **OpenShift\_lab**

Lower **OpenShift\_non\_prod**

Upper **OpenShift\_prod**

Dr **OpenShift\_dr**

In each platform’s repository, we store below OpenShift platform scripts and configuration files:



All these configuration or script files usage have been introduced in previous chapters.

We follow the below steps to upload these files to the BitBucket repository.

In platform jumpbox, issue below commands:

#*git config --global user.name "<platform>"*

#*git config --global user.email "<platform>@loblaw.ca"*

#*mkdir /home/lcluser01/<platform>/git*

#*cd /home/lcluser01/<platform>*

In order to stop SSL verification, issue command:

#*git config --global http.sslVerify false*

#*git clone* [*https://bitbucket.loblaw.ca/scm/opsh/openshift\_dr.git*](https://bitbucket.loblaw.ca/scm/opsh/openshift_dr.git)(for Dr platform)

Subfolder openshift\_dr will be created in folder /home/lcluser01/<platform>/git. Copy all necessary files to this folder.

#*cp ansible.cfg new-alertmanager.yaml git/openshift\_dr/.*

#*cd playbooks*

#*cp gcp\_creation.yml setup\_golden\_image\_outside\_rhsm.yml master\_ldap.yml ../git/openshift\_dr/.*

#*cd ../inventory*

#*cp hosts ../git/openshift\_dr/.*

#*cd group\_vars/OSEv3*

#*cp oreg\_auth.yml ../../../git/openshift\_dr/.*

#*cd ../../../scripts*

#*cp ldap\_sync.yaml ../git/openshift\_dr/.*

#*git add –all*

#*git status*

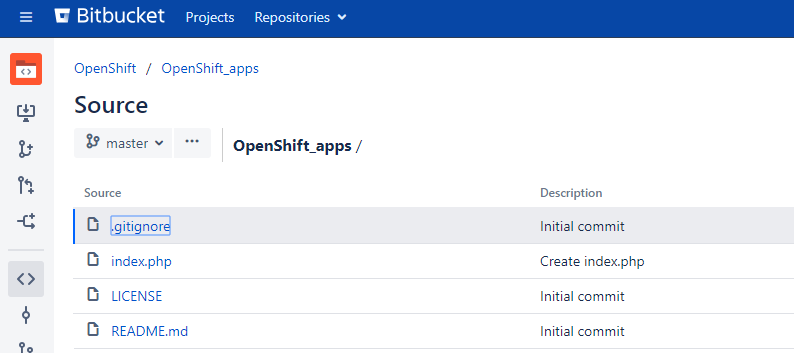
#*git commit -m "Initial Commit"*

#*git push -u origin master*

#*git status*

1. Apps resource

Repository **OpenShift\_apps** save a test app.



In jumpbox folder /home/lcluser01/<platform>/files/certs, generate file ca.crt

#*cat digicert01.cer digicertroot.cer bb\_ssl\_certificate.cer > ca.crt*

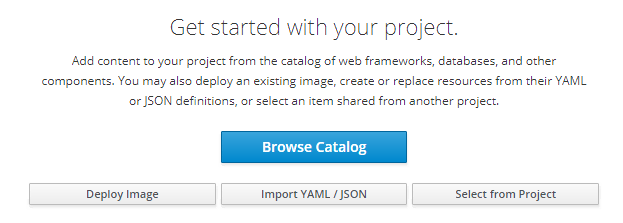
SCP **ca.crt** to your laptop folder.

#*oc adm new-project smoke-test2 --admin=<user\_name>*

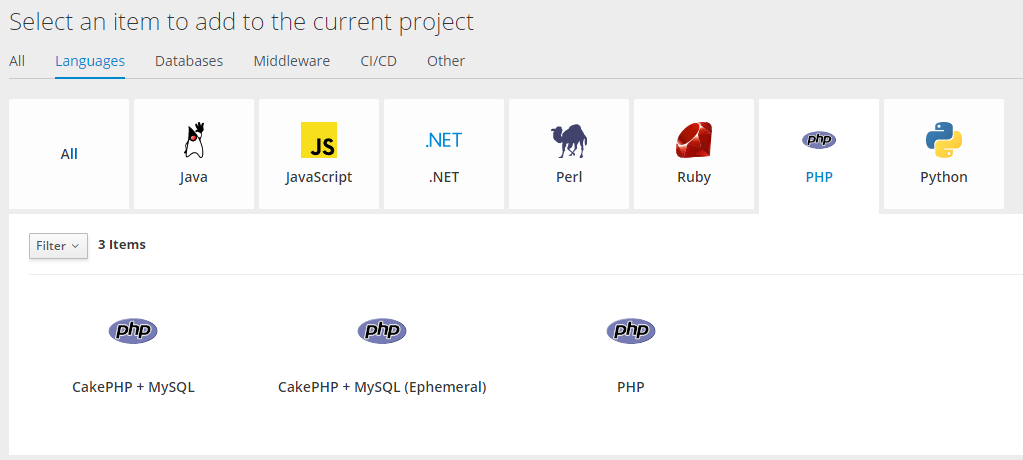
Created project smoke-test2

#*oc project smoke-test2*

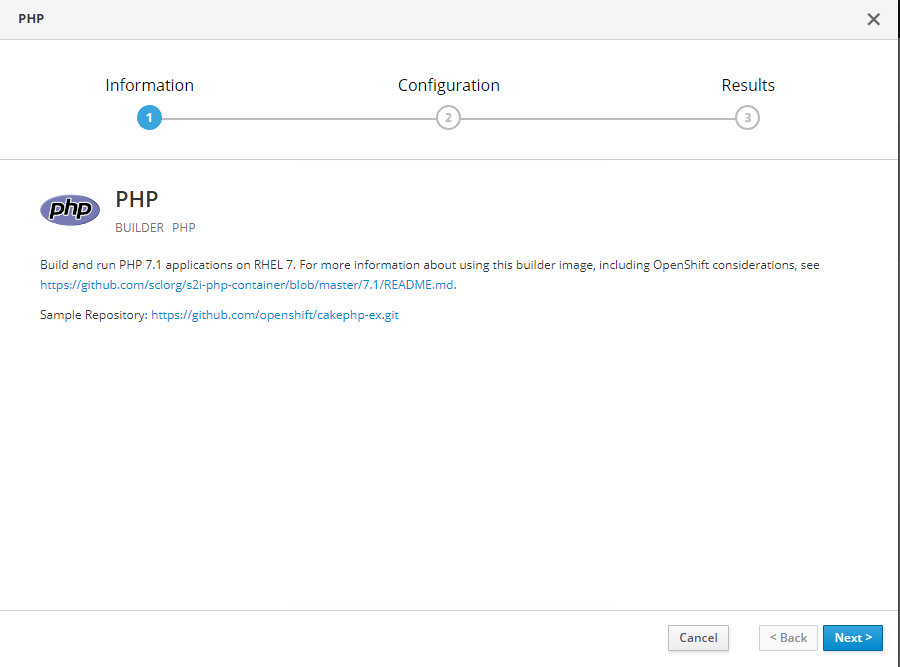
Log on to the platform console, https://ospconsole.<platform>.osp.ngco.com:8443, select the project smoke-test2.



In the **Overview**, Click **Browse Catalog** button.

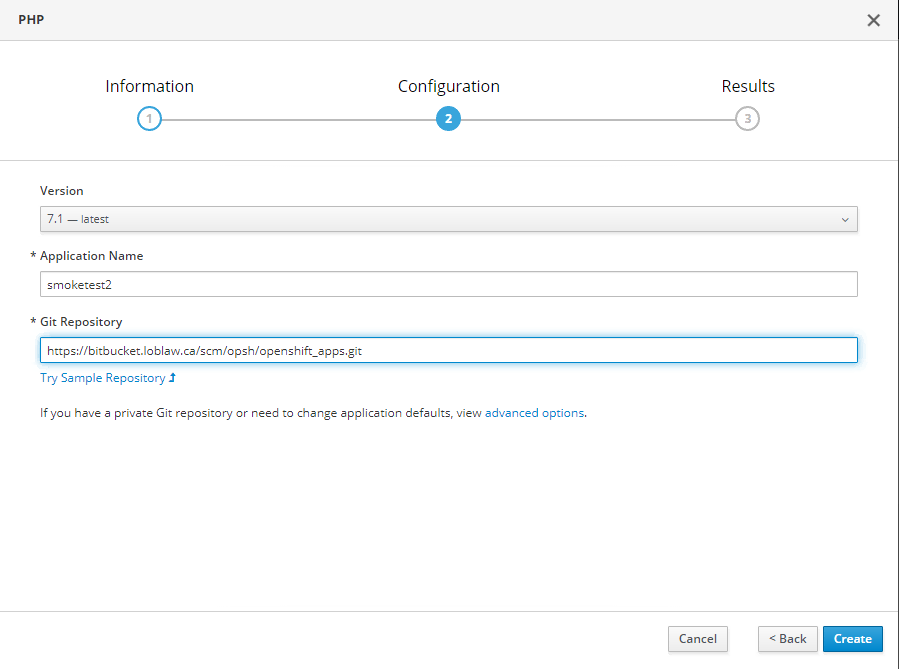


Select **Languages**, and select **PHP** > **PHP**.

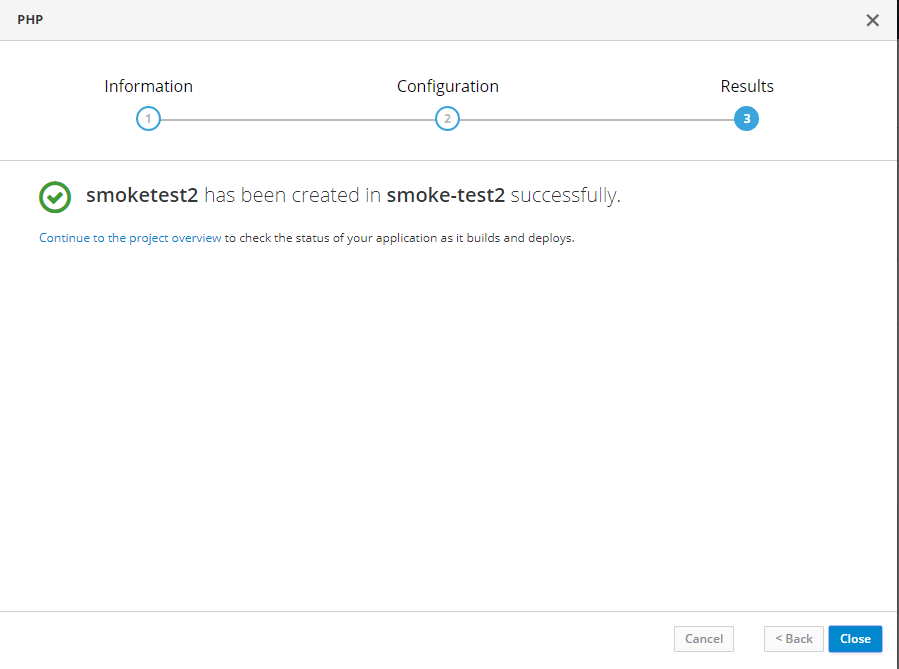


Then **Next** button.

Fill in the blanks as below, set **Git Repository** to the right repository access link on BitBucket as below:

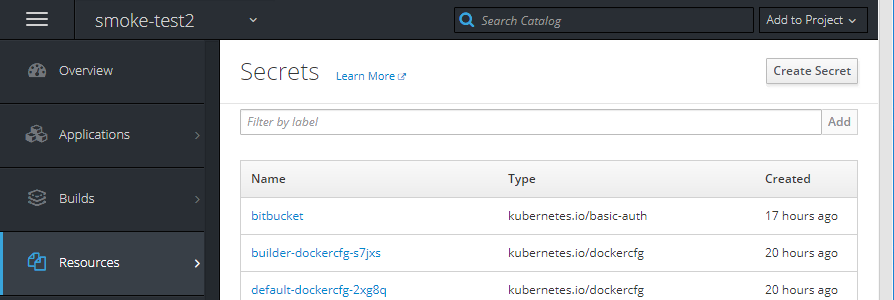


Click **Create** button.

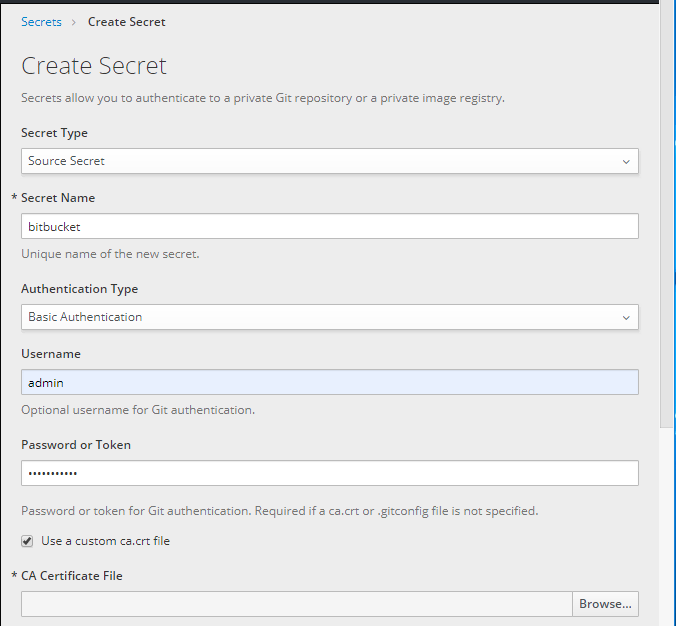


Click **Close** button.

Select **Resources** > **Secrets**. Click **Create Secret** button.

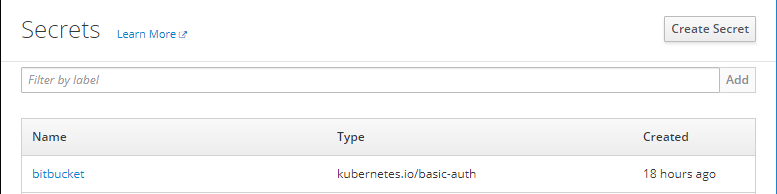


Then fill in the content on Create Secret GUI.

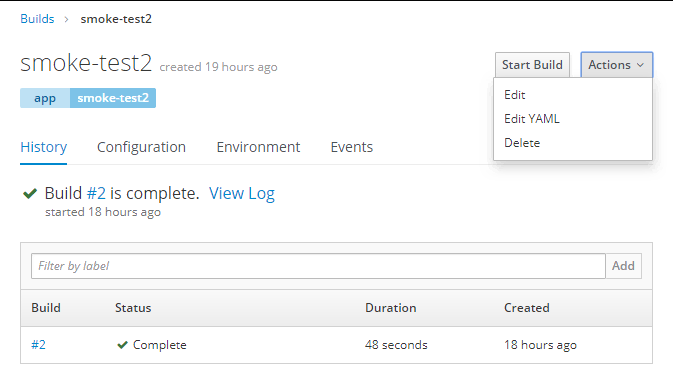


Click **Browse** button to select the **ca.crt** file, then click **Create** button at the bottom.

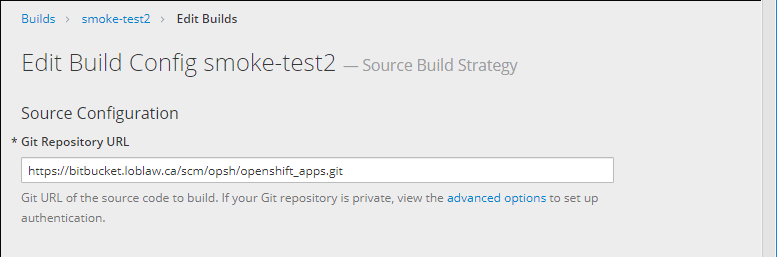
You will see the new secret in **kubernetes.io/basic-auth** type.



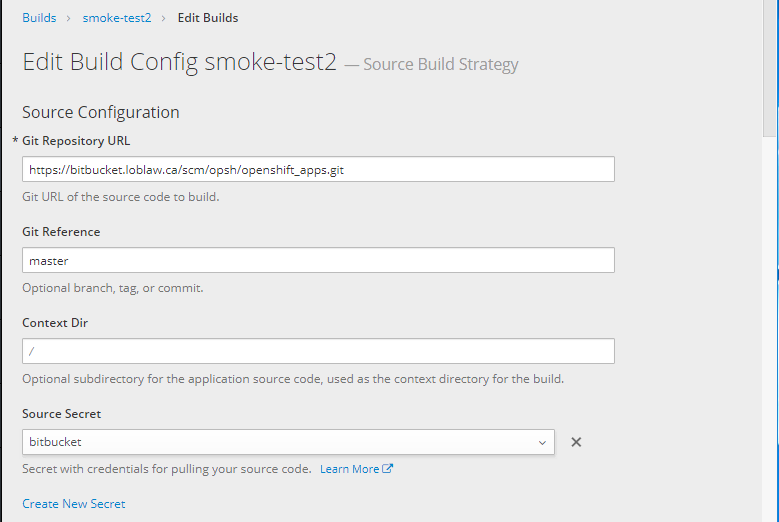
Select **Builds** > **Builds**, then select the name of application which you want to update.



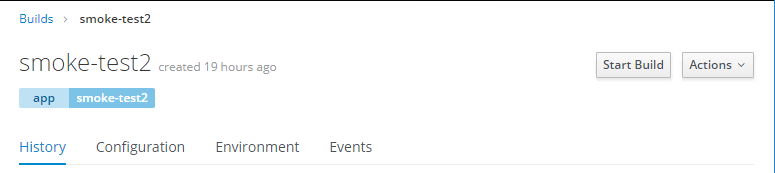
Select **Actions** > **Edit**.



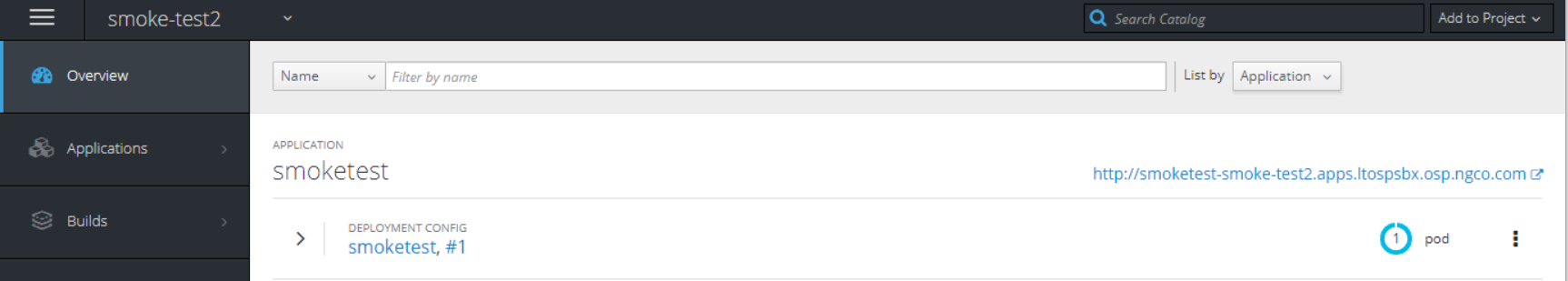
Select **advanced options**. Then select the Source Secret to the secret you have just created. Then click **Save** button at the bottom.



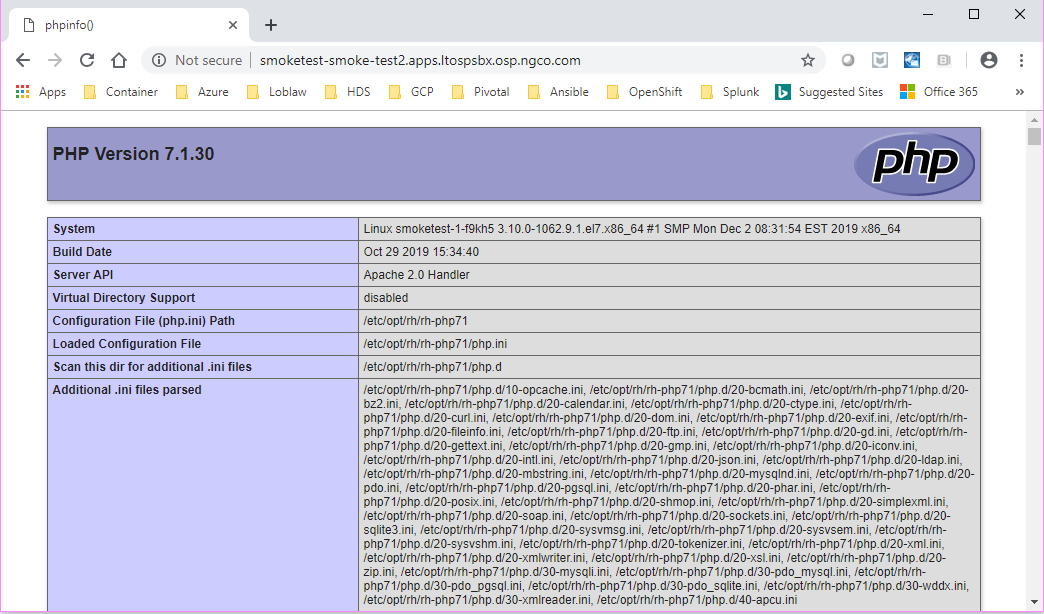
Return to Builds > Builds > Application. Click **Start Build** button.



After you complete setup the application on OpenShift platform, you will see:



Use the right top URL to open the web browser to open the app web GUI.



## Jenkins

1. Project & SA for Jenkins

In order to let OpenShift platform communicate with Jenkins, we will create a project ci and a service account in project ci.

On platform jumpbox, cd to /home/lcluser0/<platform> folder.

[root@vlosppr011-j ltospdr]# *oc new-project ci*

Now using project "ci" on server "https://ospconsoledr.ltospupr.osp.ngco.com:8443".

You can add applications to this project with the 'new-app' command. For example, try:

oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git

to build a new example application in Ruby.

Enter project ci.

[root@vlosppr011-j ltospdr]# *oc project ci*

Now using project "ci" on server "https://ospconsoledr.ltospupr.osp.ngco.com:8443".

Create service account, Jenkins, in project ci.

[root@vlosppr011-j ltospdr]# *oc create serviceaccount jenkins*

serviceaccount/jenkins created

[root@vlosppr011-j ltospdr]# *oc get sa*

NAME SECRETS AGE

builder 2 2m

default 2 2m

deployer 2 2m

jenkins 2 4s

Assign SA Jenkins to role edit.

[root@vlosppr011-j ltospdr]# *oc adm policy add-cluster-role-to-user edit system:serviceaccount:ci:jenkins*

cluster role "edit" added: "system:serviceaccount:ci:jenkins"

Get SA Jenkins token. Jenkins will use SA token to connect to OpenShift platform.

[root@vlosppr011-j ltospdr]# *oc serviceaccounts get-token jenkins -n ci*

eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJjaSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJqZW5raW5zLXRva2VuLWM4ZjJxIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImplbmtpbnMiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiI2YTM0ODIzOC0wYzg5LTExZWEtOTUwNS00MjAxMGFjMzcwMGEiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6Y2k6amVua2lucyJ9.SVegn5\_rihOeYYiX19AUEi6AmgvekRP5fSfCQwJ65c1mBtdWOXiJfaNcoWZDiCzwXn8EYN8hsDHzy7tRlE1A3AWqmeE8f6yN5bJSfmOYicG8GG6MNQwDuUwxw9PPQg35wmHD8EI9eO7tBJ9wdRjogxSwof1fOa8A1OT-LJFWiRTjCx-2YAD3Zw5CpELyei9njHL3ubX83WCr8ybD7\_1Ed4SYRnqdoism\_dYM1HbtgbDnEM2PhLhWWTnAQouSD4mWggBESKJvo4HCN7HKt3R94EHNnnvFrMk6BlVJe0yQZl9yZyZb4cipplb1uHVWmyocT0sJjCg0BkxCcbJPvhAZDg

Save the token to notepad.

Log in to OpenShift platform with Jenkins token.

[root@vlosppr011-j ltospdr]# *oc login --token=eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJjaSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJqZW5raW5zLXRva2VuLWM4ZjJxIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImplbmtpbnMiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiI2YTM0ODIzOC0wYzg5LTExZWEtOTUwNS00MjAxMGFjMzcwMGEiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6Y2k6amVua2lucyJ9.SVegn5\_rihOeYYiX19AUEi6AmgvekRP5fSfCQwJ65c1mBtdWOXiJfaNcoWZDiCzwXn8EYN8hsDHzy7tRlE1A3AWqmeE8f6yN5bJSfmOYicG8GG6MNQwDuUwxw9PPQg35wmHD8EI9eO7tBJ9wdRjogxSwof1fOa8A1OT-LJFWiRTjCx-2YAD3Zw5CpELyei9njHL3ubX83WCr8ybD7\_1Ed4SYRnqdoism\_dYM1HbtgbDnEM2PhLhWWTnAQouSD4mWggBESKJvo4HCN7HKt3R94EHNnnvFrMk6BlVJe0yQZl9yZyZb4cipplb1uHVWmyocT0sJjCg0BkxCcbJPvhAZDg*

Logged into "https://ospconsoledr.ltospupr.osp.ngco.com:8443" as "system:serviceaccount:ci:jenkins" using the token provided.

You have access to the following projects and can switch between them with 'oc project <projectname>':

\* ci

default

kube-public

kube-service-catalog

kube-system

management-infra

openshift

openshift-ansible-service-broker

openshift-console

openshift-infra

openshift-logging

openshift-metrics-server

openshift-monitoring

openshift-node

openshift-sdn

openshift-template-service-broker

openshift-web-console

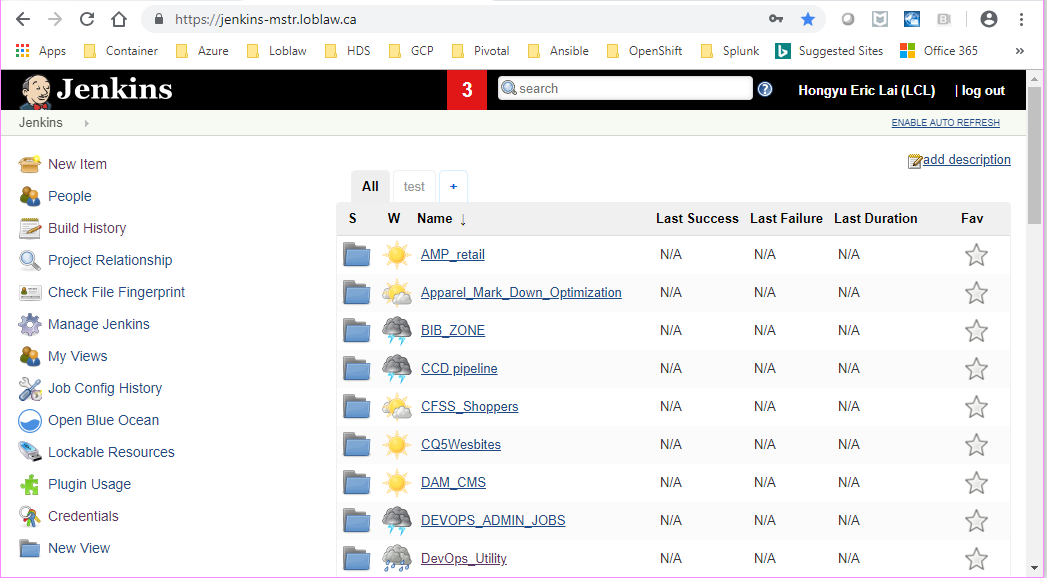
operator-lifecycle-manager

[root@vlosppr011-j ltospdr]# *oc whoami*

system:serviceaccount:ci:jenkins

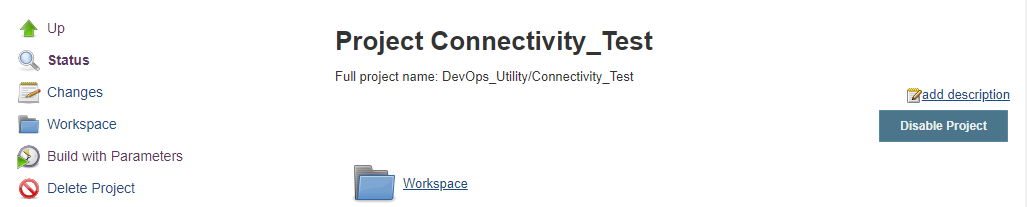
1. Connectivity Test

Open Jenkins GUI with URL [https://jenkins-mstr.loblaw.ca](https://jenkins-mstr.loblaw.ca/).

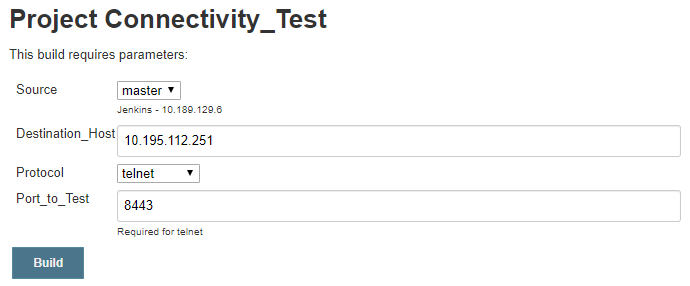


Select folder **DevOps\_Utility**, then select project **Connectivity\_Test**.

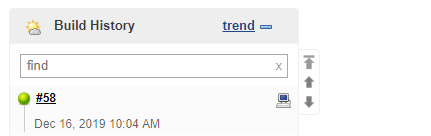




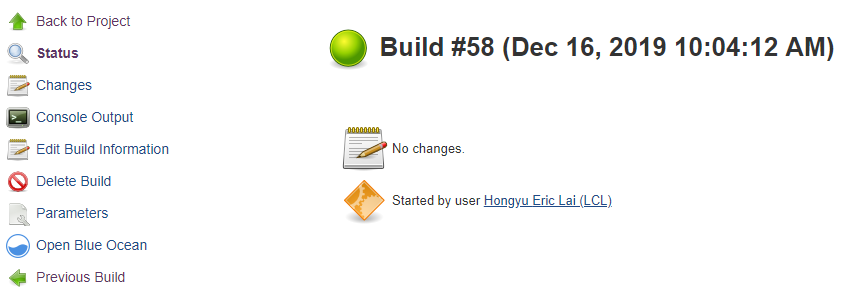
Select **Build with Parameters**.



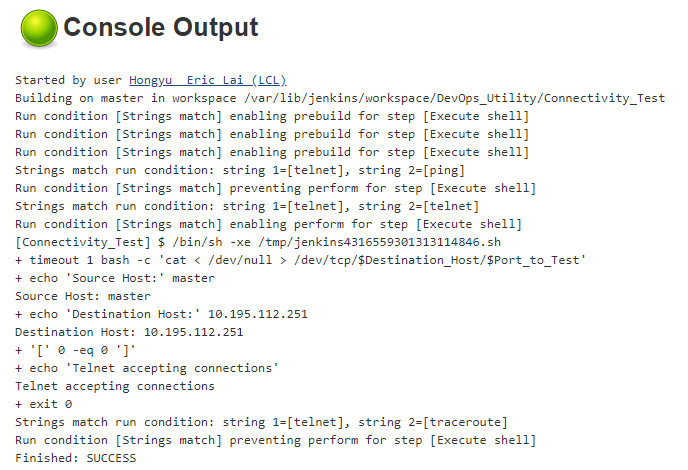
Click **Build** button.



Green means connectivity test works fine. We can check the detail for the build if you click the build link.



Click **Console Output**.



1. Jenkins Credentials

We need to create add a credential for each OpenShift platform on Jenkins.



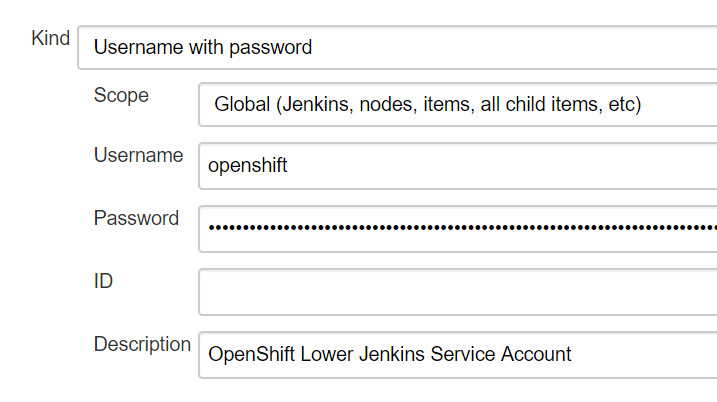
Select **Credentials**.



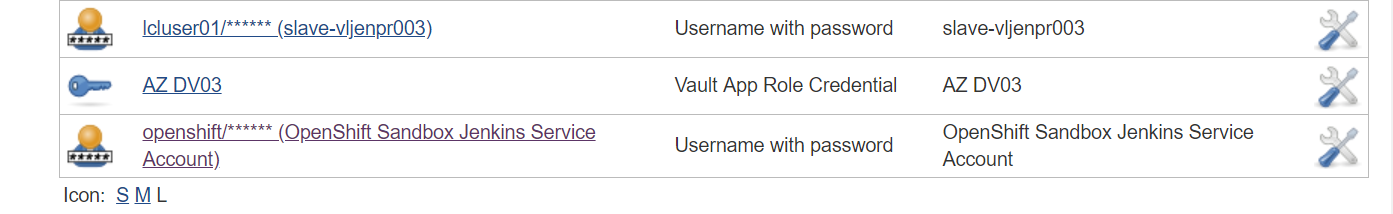
Select **global**.



Select **Add Credentials**.



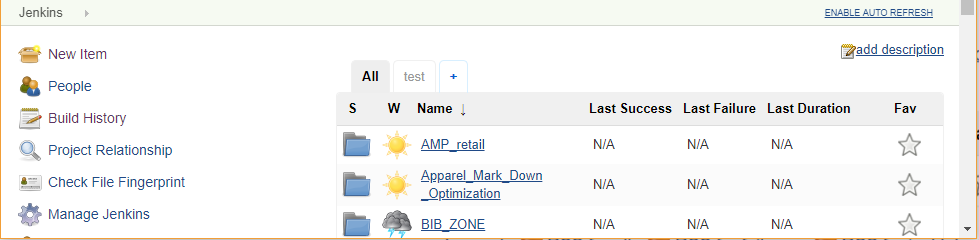
The **Password** is the token we saved.



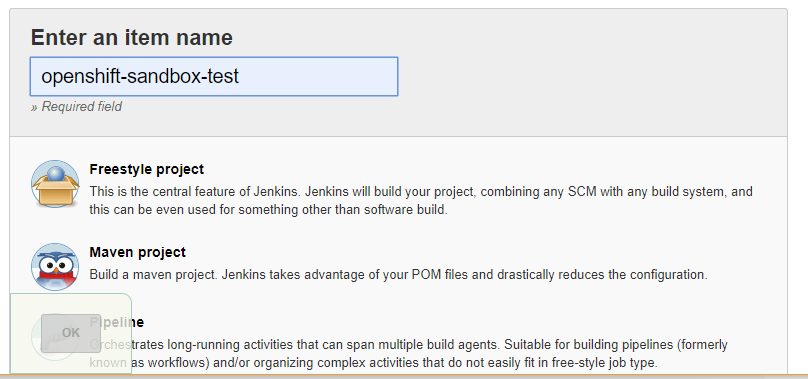
You will see the new credential is created. Later, you will need to use this credential to setup the test.

1. Jenkins project Test

Return to Jenkins main GUI.



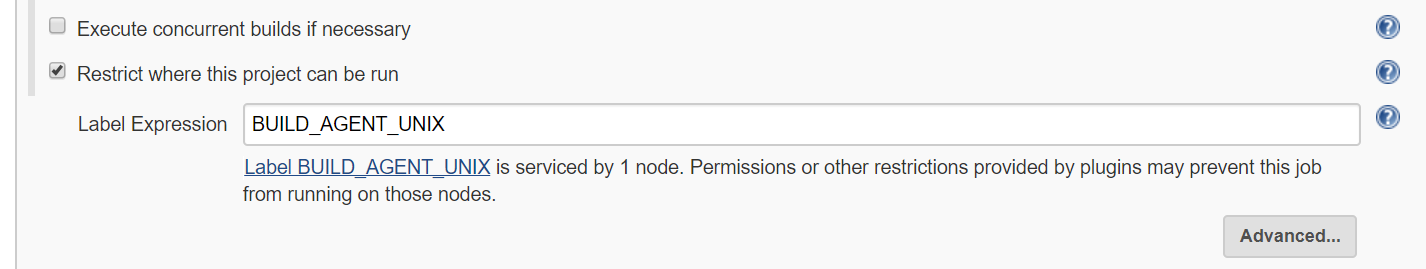
Select **New Item**,



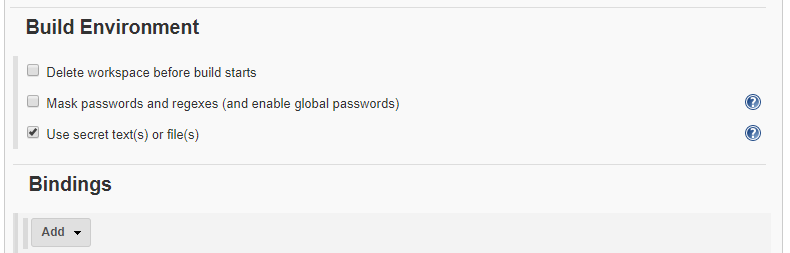
**Enter an item name**, such as *openshift-sandbox-test*, select **Freestyle project**, then click **OK** button.

In **General** tab, select **Restrict where this project can be run**.

In **Label Expression**, select **BUILD\_AGENT\_UNIX**.

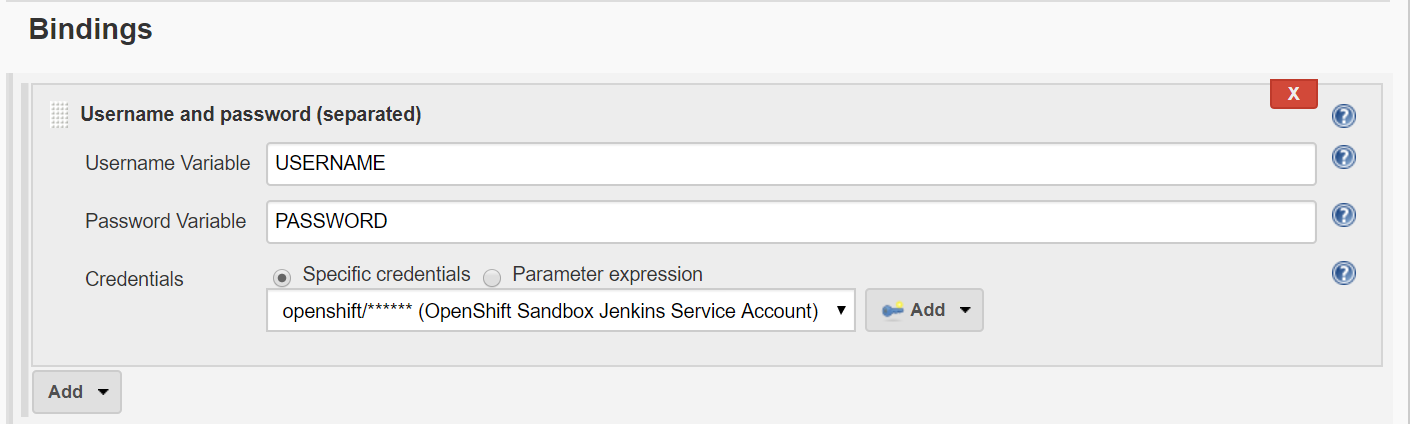


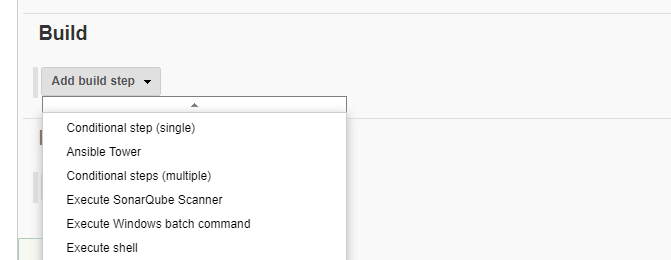
Move down to **Build Environment**, select **Use secret text(s) or file(s)**.



Click **Add** menu button, select **Username and password (separated)**.

In Bindings part, fill in the content as below. Select the Credentials we have just created.



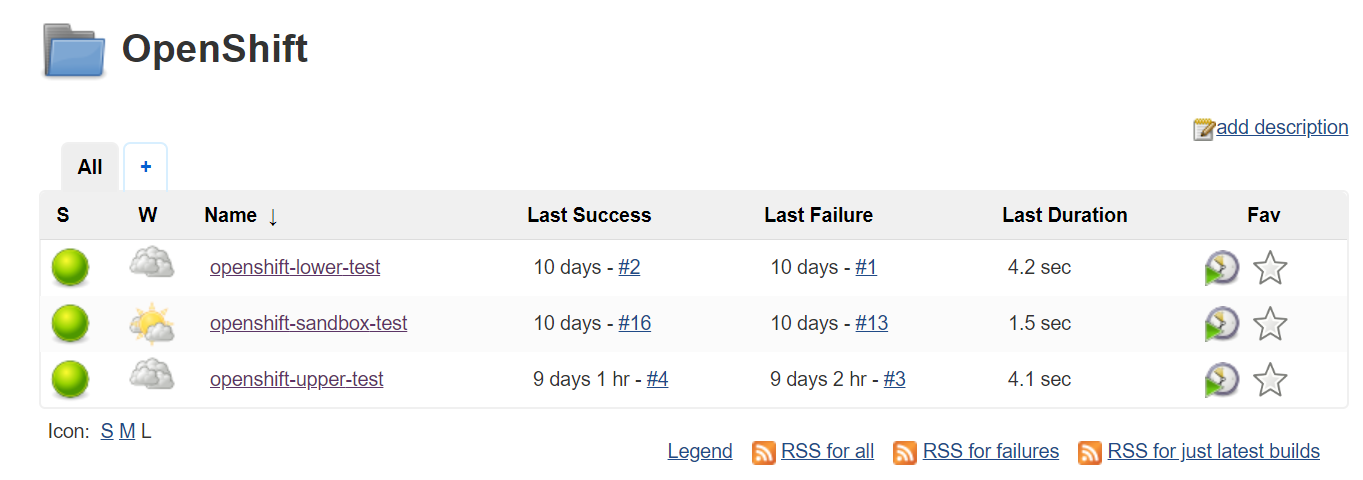


In the **Build** part, click **Add build step** menu button and select **Execute shell**.

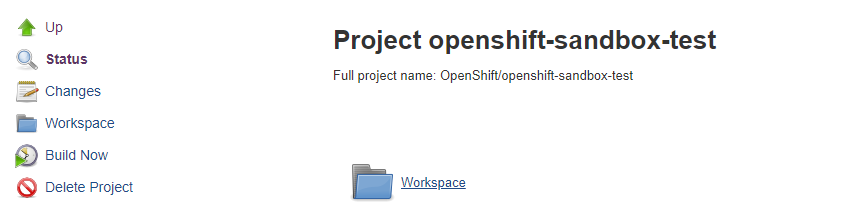
Then fill in commands in the blank field as below.



Then click **Save** button.



New test project will be created in OpenShift folder. Select it, and click **Build Now** on left.



Like Connectivity Test, select the Build link in the **Build History**, then Select **Console Output**, you will find the detail about the build. This is very useful if you want to do the troubleshooting.

Log on to platform jumpbox, change to /home/lcluser01/<platform>

[lcluser01@vlospdv001-j ~]$*oc project ci*

Now using project "ci" on server "https://ospconsole.ltospsbx.osp.ngco.com:8443".

[lcluser01@vlospdv001-j ~]$ *oc get pods*

NAME READY STATUS RESTARTS AGE

ruby-hello-world-1-build 0/1 Completed 0 2m

ruby-hello-world-1-cqgvb 1/1 Running 0 58s

There are pods in the project ci. One is Completed and another on is Running. This is the correct test result.

[lcluser01@vlospdv001-j ~]$ *oc get route*

No resources found.

[lcluser01@vlospdv001-j ~]$ *oc get svc*

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

ruby-hello-world ClusterIP 172.30.136.158 <none> 8080/TCP 3m

[lcluser01@vlospdv001-j ~]$ *oc expose svc/ruby-hello-world*

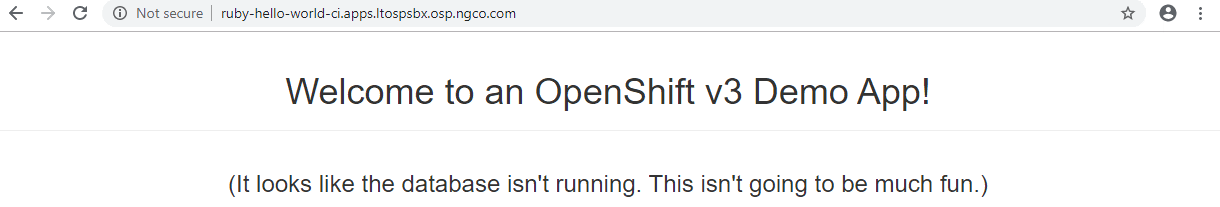
route.route.openshift.io/ruby-hello-world exposed

[lcluser01@vlospdv001-j ~]$ *oc get route*

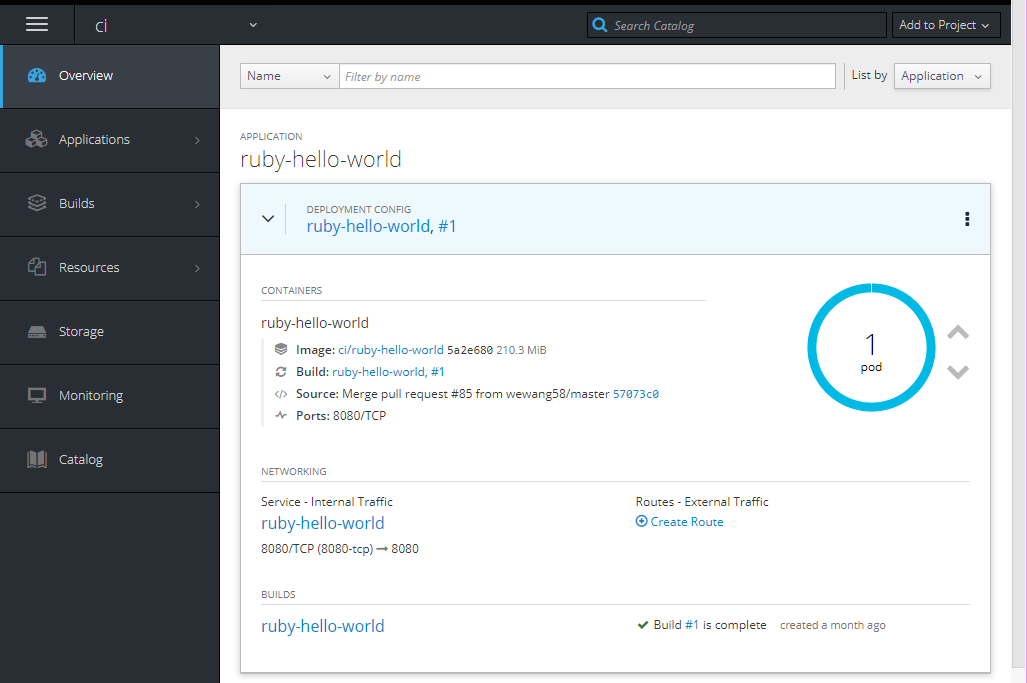
NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD

ruby-hello-world ruby-hello-world-ci.apps.ltospsbx.osp.ngco.com ruby-hello-world 8080-tcp None

Use web browser to open above link.



If you log on to OpenShift console, select project ci, you will see the application ruby-hello-world is there.



## Nexus

Nexus is a docker place which stores apps resource, but not like BitBucket, it is accessed by **docker** command, not **git** command.

Please review 2.4.b **/etc/sysconfig/docker**.

After confirming /etc/sysconfig/docker file has been updated as 2.4.b, please do the following test to verify OpenShift platform can access to Nexus platform correctly.

On OpenShift jumpbox,

#*cd /etc/sysconfig*

#*systemctl status docker*

#*docker login 10.189.129.7:8082* or #*docket login* [*http://10.189.129.7:8082*](http://10.189.129.7:8082)

Username: *usera*

Password:

Login Succeeded

#*docker pull hello-world* or #*docker pull 10.189.129.7:8082/hello-world*

#*docker images*

#*sudo docker logout 10.189.129.7:8082*

Removing login credentials for 10.189.129.7:8082

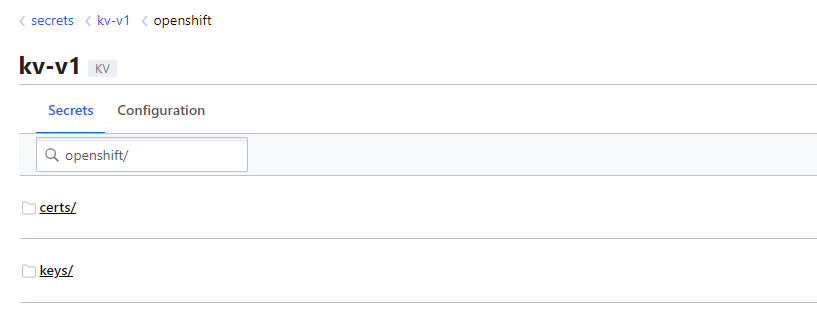
Notes:

* 10.189.129.7 is Nexus platform IP
* 8082 is the opened port for OpenShift platforms access to Nexus
* hello-world is a test app saving in Nexus.

## 4.6. Vault

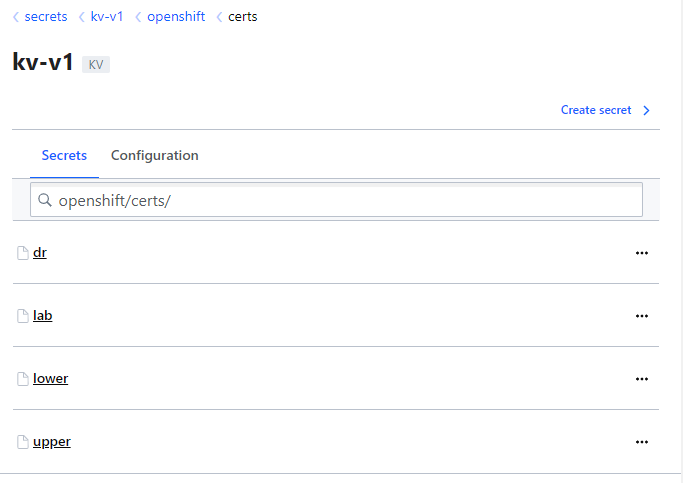
We store passwords, certificates and tokens in Vault system.

The Vault access link is <https://hashivault.loblaw.ca/ui/vault/secrets/kv-v1/list/openshift/>

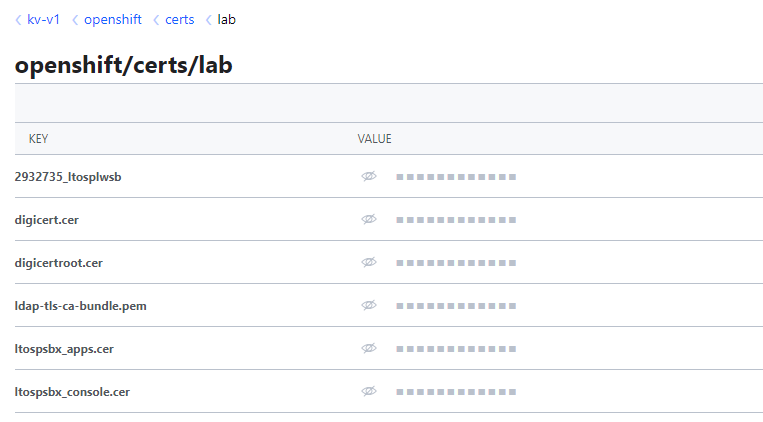


We divide **certs** path and **keys** path for OpenShift.

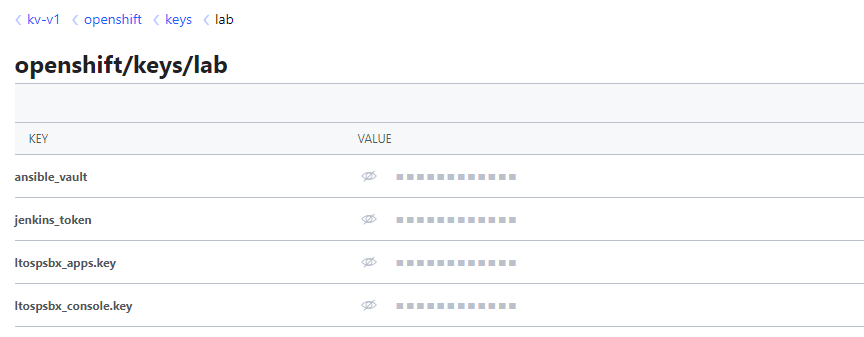
No matter certs or keys path, based on OpenShift platforms, we set one path for each platform.



For each OpenShift platform, in its certs path, we store its Red Hat Service Account for this platform, BitBucket access certificates, ldap systems access certificates, openshift console and apps load balancers certificates.



In platform certs path, we store its ansible vault, Jenkins access token, and platform console and apps certificates’ keys.



## 4.7. Splunk

a. HELM installation

We only installed HELM on Sandbox jumpbox. We reuse this HELM to create other OpenShift platform configuration files on Sandbox, and then copy them to other OpenShift platform jumpbox. We do not install HELM on all OpenShift platform jumpboxs.

On Sandbox jumpbox, as user lcluser01 in its home folder.

$*curl -s* [*https://storage.googleapis.com/kubernetes-helm/helm-v2.16.0-linux-amd64.tar.gz*](https://storage.googleapis.com/kubernetes-helm/helm-v2.16.0-linux-amd64.tar.gz) *| tar xz*  
$*cd linux-amd64/*$*./helm init --client-only*  
$*./helm version*

1. Create Splunk Client for each Platform

Here, the example is from ltospdr. HELM is ready on Sandbox.

On Sandbox Jumpbox, at folder /home/lcluser01

$*mkdir splunk\_ltospdr*

$*cd splunk\_ltospdr*

$*git clone* [*https://github.com/splunk/splunk-connect-for-kubernetes.git*](https://github.com/splunk/splunk-connect-for-kubernetes.git)

## This git command will create folder **splunk-connect-for-kubernetes**.

$*cd splunk-connect-for-kubernetes*

$*make*

$*cd ..*

$*mkdir render*

##Copy ltospupr kubernetes\_connect.yaml file to splunk\_ltospdr folder because we use the same token in ltospupr to build platform ltospdr splunk

$*cp ../splunk\_ltospupr/kubernetes\_connect.yaml .*

$*vi* *kubernetes\_connect.yaml*

Each platform has its own Splunk access token and access to different Splunk system.

Platform Splunk Token

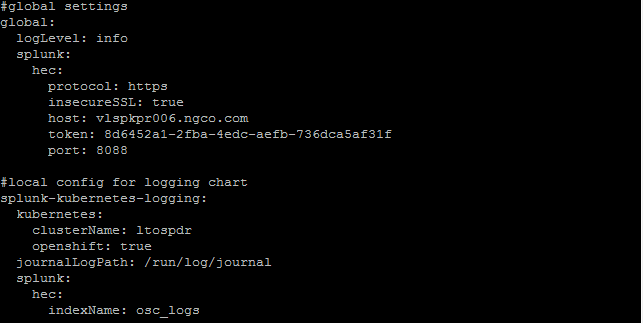
ltospsbx vlspkdv002 261c8854-d6d6-499f-9a86-950986dce1c8

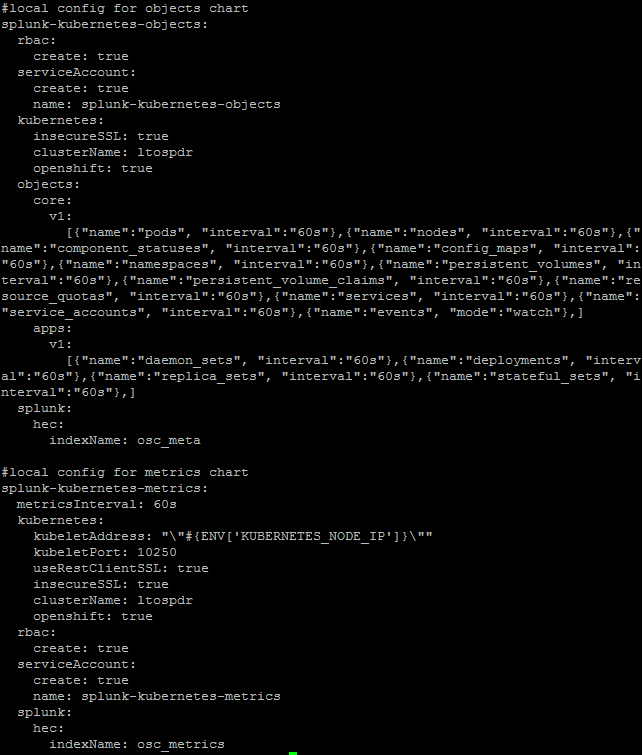
ltosplwr vlspkdv002 28ef68a8-9f65-496d-9e69-915d807ba130

ltospupr vlspkpr006 8d6452a1-2fba-4edc-aefb-736dca5af31f

ltospdr vlspkpr006 8d6452a1-2fba-4edc-aefb-736dca5af31f

Update the file **kubernetes\_connect.yaml** to change platform name from ltospupr to ltospdr. Keep other options. The file kubernetes\_connect.yaml content is listed as below:



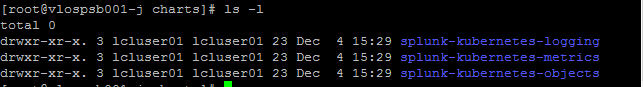


We will find there are three indexNames: **osc\_logs, osc\_meta and osc\_metrics**. All clusterNames will be changed to **ltospdr**. The **token** will be the same as ltopsupr token. Platform ltospsbx and ltosplwr have their own tokens.

HELM must be run with root. Use HELM create ltospdr Splunk configuration files on Sandbox jumpbox.

$*helm template --name=sck-rendered --namespace=splunk-connect --values kubernetes\_connect.yaml --output-dir ./render ./splunk-connect-for-kubernetes/build/splunk-connect-for-kubernetes-1.3.0.tgz*

From above command, we can find three folders are created under folder ./render/ splunk-connect-for-kubernetes/charts. Each folder represents one Splunk index.



Copy whole folder /home/lcluser0/splunk\_ltospdr from Sandbox jumpbox to ltospdr jumpbox /home/lcluser01 folder with SCP.

Log on to ltospdr jumpbox and sudo to root. Run oc command to make sure you are in project default.

#*cd /home/lcluser01/splunk\_ltospdr*

#*oc adm new-project splunk-connect --node-selector=""*

#*oc project splunk-connect*

#*oc adm policy add-scc-to-user privileged system:serviceaccount:splunk-connect:sck-rendered-splunk-kubernetes-logging*

scc "privileged" added to: ["system:serviceaccount:splunk-connect:sck-rendered-splunk-kubernetes-logging"]

#*oc adm policy add-scc-to-user privileged system:serviceaccount:splunk-connect:splunk-kubernetes-objects*

scc "privileged" added to: ["system:serviceaccount:splunk-connect:splunk-kubernetes-objects"]

#*oc adm policy add-scc-to-user privileged system:serviceaccount:splunk-connect:splunk-kubernetes-metrics*

scc "privileged" added to: ["system:serviceaccount:splunk-connect:splunk-kubernetes-metrics"]

#*cd render/*

#*cd splunk-connect-for-kubernetes/*

#*cd charts/*

#*cd splunk-kubernetes-logging/*

#*oc apply -f templates*

configmap/sck-rendered-splunk-kubernetes-logging created

daemonset.apps/sck-rendered-splunk-kubernetes-logging created

secret/splunk-kubernetes-logging created

serviceaccount/sck-rendered-splunk-kubernetes-logging created

#*cd /home/lcluser01/ltosplwr/splunk\_ltosplwr/render/splunk-connect-for-kubernetes/charts/splunk-kubernetes-metrics/*

#*oc apply -f templates*

clusterrole.rbac.authorization.k8s.io/kubelet-summary-api-read created

clusterrole.rbac.authorization.k8s.io/kube-api-aggregator created

clusterrolebinding.rbac.authorization.k8s.io/sck-rendered-splunk-kubernetes-metrics created

clusterrolebinding.rbac.authorization.k8s.io/sck-rendered-splunk-kubernetes-metrics-aggregator created

configmap/sck-rendered-splunk-kubernetes-metrics created

configmap/sck-rendered-splunk-kubernetes-metrics-aggregator created

daemonset.apps/sck-rendered-splunk-kubernetes-metrics created

deployment.apps/sck-rendered-splunk-kubernetes-metrics-agg created

secret/splunk-kubernetes-metrics created

serviceaccount/splunk-kubernetes-metrics created

#*cd /home/lcluser01/ltosplwr/splunk\_ltosplwr/render/splunk-connect-for-kubernetes/charts/splunk-kubernetes-objects/*

#*oc apply -f templates*

clusterrole.rbac.authorization.k8s.io/sck-rendered-splunk-kubernetes-objects created

clusterrolebinding.rbac.authorization.k8s.io/sck-rendered-splunk-kubernetes-objects created

configmap/sck-rendered-splunk-kubernetes-objects created

deployment.apps/sck-rendered-splunk-kubernetes-objects created

secret/splunk-kubernetes-objects created

serviceaccount/splunk-kubernetes-objects created

[root@vlosppr011-j splunk-kubernetes-objects]# *oc get pods*

NAME READY STATUS RESTARTS AGE

sck-rendered-splunk-kubernetes-logging-4zr9g 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-64r8p 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-8rcbl 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-b5vg9 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-d6gkz 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-h2p7p 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-ltb66 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-mlkwp 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-qs8bc 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-sw72n 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-vdw6q 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-logging-wrp6h 1/1 Running 0 4m

sck-rendered-splunk-kubernetes-metrics-26rrw 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-4d4kv 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-4kmqh 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-6wft6 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-agg-6bbf6f7cb4-xb8fd 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-dpmpv 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-prbqb 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-pvmfs 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-qgspq 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-r4rm5 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-smm68 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-vfp6t 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-metrics-zkmzj 1/1 Running 0 1m

sck-rendered-splunk-kubernetes-objects-77f68d76fb-dgjd5 1/1 Running 0 53s

From above result, we can see multiple splunk-kubernetes-logging and splunk-kubernetes-metrics pods and one splunk-kubernetes-objects pod are running

## 4.8. Backup/Restore

We built backup environment for different OpenShift platforms in different ways:

Platform FileStore VM Note

ltospsbx vlospsb001-f Store backup data but no NetBackup

ltosplwr vlospdv001-f NetBackup

ltospupr vlosppr001-f Share NetBackup with ltospdr

ltospdr vlosppr001-f Share NetBackup with ltospupr

For the backup / Restore detail, we have separating doc, **Backup and Restore Process**, in the team sharepoint. Please refer to that document.