Table of Contents

* [HA Deployment Lab](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_ha_deployment_lab)
* [1. Provision Lab Environment](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_provision_lab_environment)
  + [1.1. Share Public Key with OPENTLC](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_share_public_key_with_opentlc)
  + [1.2. Start Environment After Shut Down](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_start_environment_after_shut_down)
  + [1.3. Test Server Connections](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_test_server_connections)
  + [1.4. Prepare GUID environment variable across all hosts](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_prepare_guid_environment_variable_across_all_hosts)
* [2. Configure Red Hat Enterprise Linux 7 Hosts for OpenShift Deployment](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#labexercises)
  + [2.1. Explore and Verify Infrastructure Deployment](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_explore_and_verify_infrastructure_deployment)
  + [2.2. Verify Installation and Configuration of Docker](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_verify_installation_and_configuration_of_docker)
  + [2.3. Verify Yum Repositories and NFS Shared Volumes on Hosts](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_verify_yum_repositories_and_nfs_shared_volumes_on_hosts)
* [3. Deploy Highly Available OpenShift Cluster](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_deploy_highly_available_openshift_cluster)
  + [3.1. Requirements for OpenShift Deployment](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_requirements_for_openshift_deployment)
  + [3.2. Basic Required Ansible Inventory Parameters](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_basic_required_ansible_inventory_parameters)
  + [3.3. Configure Authentication Against IPA (LDAP) Server](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_configure_authentication_against_ipa_ldap_server)
  + [3.4. Execute the **openshift-ansible** Deployer](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_execute_the_code_openshift_ansible_code_deployer)
  + [3.5. Uninstalling OpenShift](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_uninstalling_openshift)
* [4. Verify OpenShift Cluster](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_verify_openshift_cluster)
  + [4.1. Verify Configuration](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_verify_configuration)
  + [4.2. Verify Authentication Provider Configuration](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_verify_authentication_provider_configuration)
  + [4.3. Synchronize Groups from IPA Server to OpenShift Cluster](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_synchronize_groups_from_ipa_server_to_openshift_cluster)
  + [4.4. Test Run Group Synchronization [ Optional ]](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_test_run_group_synchronization_optional)
  + [4.5. Synchronize Groups](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_synchronize_groups)
* [5. Post installation configuration](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_post_installation_configuration)
  + [5.1. Create PVs for Users](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_create_pvs_for_users)
  + [5.2. Fix NFS Persistent Volume Recycling](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_fix_nfs_persistent_volume_recycling)
* [6. Smoke Test](https://www.opentlc.com/labs/ocp_advanced_deployment/02_1_HA_Deployment_Solution_Lab.html#_smoke_test)

HA Deployment Lab

In this lab’s scenario, you are a consultant assigned to MitziCom, a telecommunications company. MitziCom provides hosting and cloud services to a variety of clients, ranging from medium-sized companies to enterprise giants.

MitziCom has asked you to lead a 30- to 40-hour proof-of-concept (POC) using Red Hat OpenShift Container Platform. The purpose of the POC is to determine the feasibility of using Red Hat OpenShift Container Platform as a target for internal and client workloads.

The lab details a recommended process for meeting MitziCom’s requirements, as defined in the lab’s goals. Information about the infrastructure, DNS names, passwords, and more is provided throughout the lab.

|  |  |
| --- | --- |
|  | Because the lab does not include all of the required steps and commands, you will want to consult the [OpenShift Container Platform](https://docs.openshift.com/container-platform/3.9/welcome/index.html) product documentation. This link is especially useful for research and troubleshooting. Your instructor is also available to help you. |

**Goals**

In this lab, you deploy and configure OpenShift Container Platform on a group of servers to meet these requirements:

* Configure Red Hat Enterprise Linux 7 hosts for OpenShift deployment
* Deploy a highly available OpenShift Container Platform cluster
* Configure the OpenShift Container Platform cluster

**Provisioned Environment Hosts**

* Bastion host: **bastion.$GUID.example.opentlc.com**, **bastion.$GUID.internal**
* Load balancer: **loadbalancer.$GUID.example.opentlc.com**, **loadbalancer1.$GUID.internal**
* 3 OpenShift master nodes: **master{1,2,3}.$GUID.internal**
* 2 OpenShift infrastructure nodes: **infranode{1,2}.$GUID.example.opentlc.com**, **infranode{1,2}.$GUID.internal**
* 3 OpenShift worker nodes: **node{1-2}.$GUID.internal**
* IPA Server: **ipa.shared.example.opentlc.com** (shared resource for all students)
* NFS server: **support1.$GUID.internal**
* 3 GlusterFS servers: **support{1-3}.$GUID.internal**
  + These servers are initially disabled

2. Configure Red Hat Enterprise Linux 7 Hosts for OpenShift Deployment

In this section, you prepare the hosts for installation. You then make sure that the correct Docker version is installed and configured. You also configure yum repositories on the hosts.

2.1. Explore and Verify Infrastructure Deployment

Instances are already created for you, and an **/etc/ansible/hosts** file is populated with the hosts in your cluster.

1. Verify that the Ansible hosts file is populated.

*# cat /etc/ansible/hosts*

[OSEv3:vars]

###########################################################################

### Ansible Vars

###########################################################################

timeout=60

ansible\_become=yes

ansible\_ssh\_user=ec2-user

# disable memory check, as we are not a production environment

openshift\_disable\_check="memory\_availability"

[OSEv3:children]

lb

masters

etcd

nodes

nfs

[lb]

loadbalancer1.GUID.internal host\_zone=eu-central-1a

[masters]

master1.GUID.internal host\_zone=eu-central-1a

master2.GUID.internal host\_zone=eu-central-1a

master3.GUID.internal host\_zone=eu-central-1a

[etcd]

master1.GUID.internal host\_zone=eu-central-1a

master2.GUID.internal host\_zone=eu-central-1a

master3.GUID.internal host\_zone=eu-central-1a

[nodes]

## These are the masters

master1.GUID.internal openshift\_hostname=master1.GUID.internal openshift\_node\_labels="{'env': 'master', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

master2.GUID.internal openshift\_hostname=master2.GUID.internal openshift\_node\_labels="{'env': 'master', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

master3.GUID.internal openshift\_hostname=master3.GUID.internal openshift\_node\_labels="{'env': 'master', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

## These are infranodes

infranode1.GUID.internal openshift\_hostname=infranode1.GUID.internal openshift\_node\_labels="{'env':'infra', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

infranode2.GUID.internal openshift\_hostname=infranode2.GUID.internal openshift\_node\_labels="{'env':'infra', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

## These are regular nodes

node1.GUID.internal openshift\_hostname=node1.GUID.internal openshift\_node\_labels="{'env':'app', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

node2.GUID.internal openshift\_hostname=node2.GUID.internal openshift\_node\_labels="{'env':'app', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

node3.GUID.internal openshift\_hostname=node3.GUID.internal openshift\_node\_labels="{'env':'app', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

## These are CNS nodes

# support1.GUID.internal openshift\_hostname=support1.GUID.internal openshift\_node\_labels="{'env':'glusterfs', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

# support2.GUID.internal openshift\_hostname=support2.GUID.internal openshift\_node\_labels="{'env':'glusterfs', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

# support3.GUID.internal openshift\_hostname=support3.GUID.internal openshift\_node\_labels="{'env':'glusterfs', 'cluster': 'GUID', 'zone': 'eu-central-1a'}"

[nfs]

support1.GUID.internal openshift\_hostname=support1.GUID.internal

#[glusterfs]

# support1.GUID.internal glusterfs\_devices='[ "/dev/xvdd" ]'

# support2.GUID.internal glusterfs\_devices='[ "/dev/xvdd" ]'

# support3.GUID.internal glusterfs\_devices='[ "/dev/xvdd" ]'

1. Use the Ansible **--list-hosts** command line to list the masters, nodes, and all of the host groups.

ansible masters --list-hosts

**Sample Output**

hosts (3):

master1.GUID.internal

master2.GUID.internal

master3.GUID.internal

ansible nodes --list-hosts

**Sample Output**

hosts (8):

master1.GUID.internal

master2.GUID.internal

master3.GUID.internal

infranode1.GUID.internal

infranode2.GUID.internal

node1.GUID.internal

node2.GUID.internal

node3.GUID.internal

ansible all --list-hosts

**Sample Output**

hosts (10):

master1.GUID.internal

master2.GUID.internal

master3.GUID.internal

infranode1.GUID.internal

infranode2.GUID.internal

node1.GUID.internal

node2.GUID.internal

node3.GUID.internal

loadbalancer1.GUID.internal

support1.GUID.internal

1. Verify that all of your hosts are running:

ansible all -m ping

2.2. Verify Installation and Configuration of Docker

1. Verify that Docker is running on all of the nodes in the cluster.

ansible nodes -m shell -a"systemctl status docker | grep Active"

**Sample Output**

infranode1.GUID.internal | SUCCESS | rc=0 >>

Active: active (running) since Thu 2017-03-16 20:11:10 EDT; 23min ago

master2.GUID.internal | SUCCESS | rc=0 >>

Active: active (running) since Thu 2017-03-16 20:11:19 EDT; 23min ago

master3.GUID.internal | SUCCESS | rc=0 >>

Active: active (running) since Thu 2017-03-16 20:11:18 EDT; 23min ago

master1.GUID.internal | SUCCESS | rc=0 >>

Active: active (running) since Thu 2017-03-16 20:11:18 EDT; 23min ago

infranode2.GUID.internal | SUCCESS | rc=0 >>

Active: active (running) since Thu 2017-03-16 20:11:10 EDT; 23min ago

node1.GUID.internal | SUCCESS | rc=0 >>

Active: active (running) since Thu 2017-03-16 20:11:10 EDT; 23min ago

node2.GUID.internal | SUCCESS | rc=0 >>

Active: active (running) since Thu 2017-03-16 20:11:10 EDT; 23min ago

1. Make sure that the Docker version is correct for the desired OpenShift version

ansible nodes -m shell -a"docker version|grep Version"

**Sample Output**

master2.GUID.internal | SUCCESS | rc=0 >>

Version: 1.13.1

Version: 1.13.1

master1.GUID.internal | SUCCESS | rc=0 >>

Version: 1.13.1

Version: 1.13.1

master3.GUID.internal | SUCCESS | rc=0 >>

Version: 1.13.1

Version: 1.13.1

infranode2.GUID.internal | SUCCESS | rc=0 >>

Version: 1.13.1

Version: 1.13.1

infranode1.GUID.internal | SUCCESS | rc=0 >>

Version: 1.13.1

Version: 1.13.1

node2.GUID.internal | SUCCESS | rc=0 >>

Version: 1.13.1

Version: 1.13.1

node3.GUID.internal | SUCCESS | rc=0 >>

Version: 1.13.1

Version: 1.13.1

node1.GUID.internal | SUCCESS | rc=0 >>

Version: 1.13.1

Version: 1.13.1

2.3. Verify Yum Repositories and NFS Shared Volumes on Hosts

The required yum repositories and NFS shared volumes are already set up on the environment. In this section you verify that they are set up properly.

1. List the repositories on the **bastion** host:

yum repolist

**Sample Output**

Loaded plugins: amazon-id, rhui-lb, search-disabled-repos

repo id repo name status

rh-gluster-3-client-for-rhel-7-server-rpms Red Hat Gluster Client RPMS 163

rhel-7-fast-datapath-rpms Red Hat Enterprise Linux 7 Fast Datapath 76

rhel-7-server-ansible-2.4-rpms Red Hat Enterprise Linux 7 Ansible RPMS 19

rhel-7-server-extras-rpms Red Hat Enterprise Linux 7 Extras 838

rhel-7-server-optional-rpms Red Hat Enterprise Linux 7 Optional 15,105

rhel-7-server-ose-3.9-rpms Red Hat Enterprise Linux 7 OSE 3.9 610

rhel-7-server-rh-common-rpms Red Hat Enterprise Linux 7 Common 232

rhel-7-server-rpms Red Hat Enterprise Linux 7 20,490

repolist: 37,533

1. List the repositories on all the other hosts:

ansible all -m shell -a"yum repolist"

**Sample Output**

[WARNING]: Consider using yum module rather than running yum

infranode1.GUID.internal | SUCCESS | rc=0 >>

Loaded plugins: amazon-id, rhui-lb, search-disabled-repos

repo id repo name status

rh-gluster-3-client-for-rhel-7-server-rpms Red Hat Gluster Client RPMS 163

rhel-7-fast-datapath-rpms Red Hat Enterprise Linux 7 Fast Datapath 76

rhel-7-server-ansible-2.4-rpms Red Hat Enterprise Linux 7 Ansible RPMS 19

rhel-7-server-extras-rpms Red Hat Enterprise Linux 7 Extras 838

rhel-7-server-optional-rpms Red Hat Enterprise Linux 7 Optional 15,105

rhel-7-server-ose-3.9-rpms Red Hat Enterprise Linux 7 OSE 3.9 610

rhel-7-server-rh-common-rpms Red Hat Enterprise Linux 7 Common 232

rhel-7-server-rpms Red Hat Enterprise Linux 7 20,490

repolist: 37,533

infranode2.GUID.internal | SUCCESS | rc=0 >>

[...]

1. Examine the NFS server to see which NFS volumes are shared:

ansible nfs -m shell -a"exportfs"

**Sample Output**

support1.GUID.internal | SUCCESS | rc=0 >>

/srv/nfs <world>

3. Deploy Highly Available OpenShift Cluster

In this section, you use the Ansible advanced installer to deploy OpenShift as a clustered, highly available installation that includes a load balancer in front of the API servers. The environment includes three masters, two infra nodes, three worker nodes, and the load balancer.

1. On the **bastion** host, install the **atomic-openshift-utils** and **atomic-openshift-clients** package, which includes the installer and has Ansible and the playbooks as dependencies. You may find that this package is already installed.

yum -y install atomic-openshift-utils atomic-openshift-clients

3.1. Requirements for OpenShift Deployment

The **/etc/ansible/hosts** Ansible inventory file is used to define and configure the OpenShift cluster.

1. Your OpenShift cluster is expected to have the following characteristics as defined in the inventory file:
   * Three master hosts in the deployment
   * A load balancer to access the masters
   * Two infra nodes, each running a router
   * LDAP authentication (see below for details)
   * An integrated registry pod backed by a persistent volume (PV) storage (NFS)
   * Router pods deployed, configured, and running on each infranode in the cluster
   * Aggregated logging configured and working
   * Metrics collection configured and working
   * Worker nodes labeled as **env=app** and infra nodes labeled as **env=infra**
   * All infrastructure components (router, registry, logging, metrics, service brokers) running on infranodes
   * A **\*.apps.${GUID}.example.opentlc.com** wildcard DNS entry that points to the infra nodes
   * The load balancer configured with **loadbalancer.${GUID}.example.opentlc.com** as the external DNS entry

3.2. Basic Required Ansible Inventory Parameters

1. On the bastion, edit your **/etc/ansible/hosts** Ansible inventory file for the deployment. Use the following hints to set up your hosts file correctly:
   * Minimum Variable Settings:
     + **deployment\_type=** ???
     + **openshift\_master\_cluster\_method=** ???
     + **openshift\_master\_cluster\_hostname=** ???
     + **openshift\_master\_cluster\_public\_hostname=** ???
     + **openshift\_master\_default\_subdomain=** ???

3.3. Configure Authentication Against IPA (LDAP) Server

In this section, you mainly edit the Ansible inventory file to configure the OpenShift master API servers to authenticate against an existing IPA (LDAP) server. After installation by the **openshift-ansible** deployer is complete,, you will and synchronize or create OpenShift group objects to match the groups that are configured in IPA.

3.3.1. Distribute Identity Management (IdM) Certificate Authority Certificates

The IPA server sets up its own Certificate Authority (CA) and does not use the same CA as your RHOCP installation. In order for RHOCP to make secured LDAP requests to the IPA server, your master servers need the CA certificate.

1. On the **bastion** host, download the **ca.crt** file:
2. cd /root

wget http://ipa.shared.example.opentlc.com/ipa/config/ca.crt -O /root/ipa-ca.crt

3.3.2. Set Up IPA as Authentication Provider

1. Configure LDAP authentication using the Ansible installer. Use the following information to set up the authentication provider in the Ansible inventory file.:
   * **bindDN**: **uid=admin,cn=users,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com**
   * **bindPassword**: **r3dh4t1!**
   * **ca**: **/etc/origin/master/ipa-ca.crt**
   * **url**:

ldaps:*//ipa.shared.example.opentlc.com:636/cn=users,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com?uid?sub?(memberOf=cn=ocp-users,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com)*

1. Use the **openshift\_master\_ldap\_ca\_file** variable to copy the certificate to the masters.
2. Make sure that your **/etc/ansible/hosts** file contains the following in the **[OSEv3:vars]** section. Make sure the file does not contain any other values for **openshift\_master\_identity\_provider**:
3. openshift\_master\_identity\_providers=[{'name': 'ldap', 'challenge': 'true', 'login': 'true', 'kind': 'LDAPPasswordIdentityProvider','attributes': {'id': ['dn'], 'email': ['mail'], 'name': ['cn'], 'preferredUsername': ['uid']}, 'bindDN': 'uid=admin,cn=users,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com', 'bindPassword': 'r3dh4t1!', 'ca': '/etc/origin/master/ipa-ca.crt','insecure': 'false', 'url': 'ldaps://ipa.shared.example.opentlc.com:636/cn=users,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com?uid?sub?(memberOf=cn=ocp-users,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com)'}]

openshift\_master\_ldap\_ca\_file=/root/ipa-ca.crt

3.3.3. OpenShift Hosted Components Requirements

1. Continue to edit your Ansible inventory file. Now address supporting hosted components.
   * **openshift\_hosted\_infra\_selector=** ???
     + Default node selector for infra components (**ONLY** for router/registry/template service broker/ansible service broker)

|  |  |
| --- | --- |
|  | Watch for Node Selectors! Because you are deploying a cluster with custom node labels (**env=infra**) rather than using the default (**region=infra**) **all** components need explicit Node Selectors! |

* + Logging Components: Elasticsearch, Kibana, Logging Curator
    - Find the several proper parameters for logging and its associated storage
  + Metrics Components: Cassandra, Hawkular, Heapster
    - Find the several proper parameters for metrics and its associated storage
  + Serivce Catalog Components: API server, Template Service Broker, Ansible Broker
    - Find the several proper parameters for the service catalog and its associated storage
  + Prometheus components if you chose to install them

|  |  |
| --- | --- |
|  | Do **NOT** enable any of the **[glusterfs]** nodes. Enabling Container Native Storage will be part of a later exercise. \* When configuring storage for any components (Logging, Metrics, etc.) make sure that the storage is created in **/srv/nfs** on the NFS server because this directory is backed by a separate volume group. |
|  | There is a fully populated and working solution hosts file in **/var/preserve/hosts** available on your bastion to consult. |

3.4. Execute the **openshift-ansible** Deployer

1. Run **ansible-playbook** to check that all prerequisites are met. This should take less than 5 minues.

|  |  |
| --- | --- |
|  | Use **tmux** **before** any ansible-playbook runs: [https://tmuxcheatsheet.com](https://tmuxcheatsheet.com/). If you want to be able to scroll in tmux window use Ctrl-B [ and Ctrl-B q to quit scroll mode. On a Mac use iTerm2 ([https://www.iterm2.com](https://www.iterm2.com/)) and then you can use "tmux -CC" to create a secondary tab with full scrollback etc. |

1. ansible-playbook -f 20 /usr/share/ansible/openshift-ansible/playbooks/prerequisites.yml
2. **Sample Output**
3. [...]
4. PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
5. infranode1.GUID.internal : ok=61 changed=13 unreachable=0 failed=0
6. infranode2.GUID.internal : ok=61 changed=13 unreachable=0 failed=0
7. loadbalancer1.GUID.internal: ok=37 changed=4 unreachable=0 failed=0
8. localhost : ok=13 changed=0 unreachable=0 failed=0
9. master1.GUID.internal : ok=62 changed=13 unreachable=0 failed=0
10. master2.GUID.internal : ok=62 changed=13 unreachable=0 failed=0
11. master3.GUID.internal : ok=68 changed=14 unreachable=0 failed=0
12. node1.GUID.internal : ok=61 changed=13 unreachable=0 failed=0
13. node2.GUID.internal : ok=61 changed=13 unreachable=0 failed=0
14. node3.GUID.internal : ok=61 changed=13 unreachable=0 failed=0
15. support1.GUID.internal : ok=35 changed=4 unreachable=0 failed=0
16. INSTALLER STATUS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
17. Initialization : Complete (0:00:27)
18. Run **ansible-playbook** to deploy your cluster. This will take about 30 minutes.

ansible-playbook -f 20 /usr/share/ansible/openshift-ansible/playbooks/deploy\_cluster.yml

**Sample Output**

[...]

PLAY RECAP

infranode1.GUID.internal : ok=135 changed=52 unreachable=0 failed=0

infranode2.GUID.internal : ok=135 changed=52 unreachable=0 failed=0

loadbalancer1.GUID.internal: ok=63 changed=10 unreachable=0 failed=0

localhost : ok=14 changed=0 unreachable=0 failed=0

master1.GUID.internal : ok=339 changed=137 unreachable=0 failed=0

master2.GUID.internal : ok=339 changed=137 unreachable=0 failed=0

master3.GUID.internal : ok=1056 changed=415 unreachable=0 failed=0

node1.GUID.internal : ok=135 changed=52 unreachable=0 failed=0

node2.GUID.internal : ok=135 changed=52 unreachable=0 failed=0

node3.GUID.internal : ok=135 changed=52 unreachable=0 failed=0

support1.GUID.internal : ok=33 changed=7 unreachable=0 failed=0

INSTALLER STATUS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Initialization : Complete (0:00:26)

Health Check : Complete (0:00:08)

etcd Install : Complete (0:01:05)

NFS Install : Complete (0:00:14)

Load balancer Install : Complete (0:00:15)

Master Install : Complete (0:09:32)

Master Additional Install : Complete (0:00:52)

Node Install : Complete (0:04:11)

Hosted Install : Complete (0:01:14)

Web Console Install : Complete (0:00:24)

Metrics Install : Complete (0:02:02)

Logging Install : Complete (0:03:50)

Prometheus Install : Complete (0:00:51)

Service Catalog Install : Complete (0:01:54)

|  |  |
| --- | --- |
|  | If your installation fails you will need to uninstall OpenShift using the uninstall playbook and start the installation from the beginning including checking the prerequisites! See <https://docs.openshift.com/container-platform/3.9/install_config/install/advanced_install.html#installer-known-issues> |

3.5. Uninstalling OpenShift

In case you need to uninstall OpenShift follow the instructions at <https://docs.openshift.com/container-platform/3.9/install_config/install/advanced_install.html#uninstalling-advanced>.

Run the uninstall playbook:

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

After the uninstallation completes it is also usually necessary to delete all left over content (certificates etc.) in the /etc/origin directories on all masters and nodes.

ansible nodes -a "rm -rf /etc/origin"

It is also necessary to delete all data from the NFS Server.

ansible nfs -a "rm -rf /srv/nfs/\*"

4. Verify OpenShift Cluster

In this section, you will ensure that the proper configuration and components have been deployed in accord your ansible Inventory file.

1. Once the installation is complete copy the .kube directory from master1 to your bastion. That way you can run oc commands as **system:admin** on the bastion and don’t have to ssh to a master:

ansible masters[0] -b -m fetch -a "src=/root/.kube/config dest=/root/.kube/config flat=yes"

1. Verify that you are now **system:admin**:

oc whoami

**Sample Output**

system:admin

4.1. Verify Configuration

1. Run **oc get nodes** to display the nodes:

oc get nodes --show-labels

**Sample Output**

NAME STATUS ROLES AGE VERSION LABELS

infranode1.GUID.internal Ready <none> 12m v1.9.1+a0ce1bc657 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,cluster=GUID,env=infra,kubernetes.io/hostname=infranode1.GUID.internal,logging-infra-fluentd=true,zone=eu-central-1a

infranode2.GUID.internal Ready <none> 12m v1.9.1+a0ce1bc657 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,cluster=GUID,env=infra,kubernetes.io/hostname=infranode2.GUID.internal,logging-infra-fluentd=true,zone=eu-central-1a

master1.GUID.internal Ready master 12m v1.9.1+a0ce1bc657 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,cluster=GUID,env=master,kubernetes.io/hostname=master1.GUID.internal,logging-infra-fluentd=true,node-role.kubernetes.io/master=true,openshift-infra=apiserver,zone=eu-central-1a

master2.GUID.internal Ready master 12m v1.9.1+a0ce1bc657 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,cluster=GUID,env=master,kubernetes.io/hostname=master2.GUID.internal,logging-infra-fluentd=true,node-role.kubernetes.io/master=true,zone=eu-central-1a

master3.GUID.internal Ready master 12m v1.9.1+a0ce1bc657 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,cluster=GUID,env=master,kubernetes.io/hostname=master3.GUID.internal,logging-infra-fluentd=true,node-role.kubernetes.io/master=true,zone=eu-central-1a

node1.GUID.internal Ready compute 12m v1.9.1+a0ce1bc657 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,cluster=GUID,env=app,kubernetes.io/hostname=node1.GUID.internal,logging-infra-fluentd=true,node-role.kubernetes.io/compute=true,zone=eu-central-1a

node2.GUID.internal Ready compute 12m v1.9.1+a0ce1bc657 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,cluster=GUID,env=app,kubernetes.io/hostname=node2.GUID.internal,logging-infra-fluentd=true,node-role.kubernetes.io/compute=true,zone=eu-central-1a

node3.GUID.internal Ready compute 12m v1.9.1+a0ce1bc657 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,cluster=GUID,env=app,kubernetes.io/hostname=node3.GUID.internal,logging-infra-fluentd=true,node-role.kubernetes.io/compute=true,zone=eu-central-1a

1. Validate that all pods are running (none are pending) and on the correct nodes.

oc get pod --all-namespaces -o wide

**Sample Output for a fully configured OpenShift cluster**

NAMESPACE NAME READY STATUS RESTARTS AGE IP NODE

default docker-registry-1-9tfnm 1/1 Running 0 10m 10.128.0.3 infranode1.GUID.internal

default registry-console-1-4thqf 1/1 Running 0 9m 10.130.0.2 node3.GUID.internal

default router-1-jjz8n 1/1 Running 0 10m 192.199.0.188 infranode1.GUID.internal

default router-1-tnwlx 1/1 Running 0 10m 192.199.0.221 infranode2.GUID.internal

kube-service-catalog apiserver-jxcl2 1/1 Running 0 2m 10.131.0.5 master1.GUID.internal

kube-service-catalog controller-manager-22qrk 1/1 Running 0 2m 10.131.0.4 master1.GUID.internal

logging logging-curator-1-lkknp 1/1 Running 0 4m 10.128.0.7 infranode1.GUID.internal

logging logging-es-data-master-rx1fda95-1-66l6k 2/2 Running 0 3m 10.128.0.9 infranode1.GUID.internal

logging logging-fluentd-4mjx4 1/1 Running 0 4m 10.130.0.3 node3.GUID.internal

logging logging-fluentd-5qlcn 1/1 Running 0 4m 10.129.0.4 master2.GUID.internal

logging logging-fluentd-8wbx9 1/1 Running 0 4m 10.131.0.3 master1.GUID.internal

logging logging-fluentd-czjrn 1/1 Running 0 4m 10.131.2.2 node1.GUID.internal

logging logging-fluentd-nbdmh 1/1 Running 0 4m 10.128.0.8 infranode1.GUID.internal

logging logging-fluentd-r876d 1/1 Running 0 4m 10.129.2.3 master3.GUID.internal

logging logging-fluentd-t5gf8 1/1 Running 0 4m 10.130.2.8 infranode2.GUID.internal

logging logging-fluentd-z45b7 1/1 Running 0 4m 10.128.2.2 node2.GUID.internal

logging logging-kibana-1-gfwpr 2/2 Running 0 4m 10.130.2.6 infranode2.GUID.internal

openshift-ansible-service-broker asb-1-zz2mg 1/1 Running 1 1m 10.131.2.3 node1.GUID.internal

openshift-ansible-service-broker asb-etcd-1-9xz4g 1/1 Running 0 1m 10.128.2.4 node2.GUID.internal

openshift-infra hawkular-cassandra-1-5l22f 1/1 Running 0 7m 10.128.0.5 infranode1.GUID.internal

openshift-infra hawkular-metrics-flbs5 1/1 Running 0 7m 10.130.2.4 infranode2.GUID.internal

openshift-infra heapster-r97pm 1/1 Running 0 7m 10.128.0.6 infranode1.GUID.internal

openshift-metrics prometheus-0 6/6 Running 0 3m 10.130.2.10 infranode2.GUID.internal

openshift-metrics prometheus-node-exporter-2c554 1/1 Running 0 3m 192.199.0.221 infranode2.GUID.internal

openshift-metrics prometheus-node-exporter-2c9dq 1/1 Running 0 3m 192.199.0.114 node3.GUID.internal

openshift-metrics prometheus-node-exporter-47gfh 1/1 Running 0 3m 192.199.0.128 node2.GUID.internal

openshift-metrics prometheus-node-exporter-5nf98 1/1 Running 0 3m 192.199.0.212 master2.GUID.internal

openshift-metrics prometheus-node-exporter-8xhlk 1/1 Running 0 3m 192.199.0.8 master3.GUID.internal

openshift-metrics prometheus-node-exporter-lxbfm 1/1 Running 0 3m 192.199.0.126 master1.GUID.internal

openshift-metrics prometheus-node-exporter-wlx2d 1/1 Running 0 3m 192.199.0.188 infranode1.GUID.internal

openshift-metrics prometheus-node-exporter-wxvss 1/1 Running 0 3m 192.199.0.142 node1.GUID.internal

openshift-template-service-broker apiserver-7wtwn 1/1 Running 0 1m 10.130.2.11 infranode2.GUID.internal

openshift-template-service-broker apiserver-hbqpb 1/1 Running 0 1m 10.128.0.10 infranode1.GUID.internal

openshift-web-console webconsole-c9cf7f469-7phd8 1/1 Running 2 9m 10.129.2.2 master3.GUID.internal

openshift-web-console webconsole-c9cf7f469-j27pc 1/1 Running 2 9m 10.131.0.2 master1.GUID.internal

openshift-web-console webconsole-c9cf7f469-xmfqb 1/1 Running 2 9m 10.129.0.3 master2.GUID.internal

4.2. Verify Authentication Provider Configuration

In this section, you verify the configuration of the authentication provider by attempting to log in to the master web console.

1. Navigate to the OpenShift Container Platform web console.
2. Log in using **payment1** as the username and **r3dh4t1!** as the password.
3. If you are unable to authenticate successfully, double-check your configuration.

4.3. Synchronize Groups from IPA Server to OpenShift Cluster

1. Synchronize the following groups from the IPA server to your OpenShift cluster:
   * **group/portalapp**
   * **group/paymentapp**
   * **group/ocp-production**
   * **group/ocp-platform**
     1. Use the following hints:
        + **url**: **ldap://ipa.shared.example.opentlc.com** or **ldaps://ipa.shared.example.opentlc.com:636**
        + **ca**: **/etc/origin/master/ipa-ca.crt**
        + **bindDN**: **uid=admin,cn=users,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com**
        + **bindPassword**: **r3dh4t1!**
        + **baseDN** for **groupsQuery**: **cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com**
        + **baseDN** for **usersQuery**: **cn=users,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com**
        + **filter**:

(&(!(objectClass=mepManagedEntry))(!(cn=trust admins))(!(cn=groups))(!(cn=admins))(!(cn=ipausers))(!(cn=editors))(!(cn=ocp-users))(!(cn=evmgroup\*))(!(cn=ipac\*)))

* + - * LDAP groups are referenced like this: **cn=portalapp,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com**

|  |  |
| --- | --- |
|  | The OPENTLC IPA server has a lot of groups defined. To limit the number of groups that are synced, set up a whitelist. |

* + 1. On the **master1** host, create the **/etc/origin/master/groupsync.yaml** file:
    2. ssh master1.$GUID.internal
    3. sudo -i
    4. cat << EOF > /etc/origin/master/groupsync.yaml
    5. kind: LDAPSyncConfig
    6. apiVersion: v1
    7. url: "ldap://ipa.shared.example.opentlc.com"
    8. insecure: false
    9. ca: "/etc/origin/master/ipa-ca.crt"
    10. bindDN: "uid=admin,cn=users,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com"
    11. bindPassword: "r3dh4t1!"
    12. rfc2307:
    13. groupsQuery:
    14. baseDN: "cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com"
    15. scope: sub
    16. derefAliases: never
    17. filter: (&(!(objectClass=mepManagedEntry))(!(cn=trust admins))(!(cn=groups))(!(cn=admins))(!(cn=ipausers))(!(cn=editors))(!(cn=ocp-users))(!(cn=evmgroup\*))(!(cn=ipac\*)))
    18. groupUIDAttribute: dn
    19. groupNameAttributes: [ cn ]
    20. groupMembershipAttributes: [ member ]
    21. usersQuery:
    22. baseDN: "cn=users,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com"
    23. scope: sub
    24. derefAliases: never
    25. userUIDAttribute: dn
    26. userNameAttributes: [ uid ]

EOF

* + 1. Map LDAP groups to specific names in RHOCP by adding this section to the file **/etc/origin/master/groupsync.yaml**:
    2. echo '
    3. groupUIDNameMapping:
    4. "cn=portalapp,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com": "portalapp"
    5. "cn=paymentapp,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com": "paymentapp"
    6. "cn=ocp-production,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com": "ocp-production"
    7. "cn=ocp-platform,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com": "ocp-platform"

' >>/etc/origin/master/groupsync.yaml

* + 1. Create the **/etc/origin/master/whitelist.yaml** file:
    2. cat << EOF > /etc/origin/master/whitelist.yaml
    3. cn=portalapp,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com
    4. cn=paymentapp,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com
    5. cn=ocp-platform,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com
    6. cn=ocp-production,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com

EOF

4.4. Test Run Group Synchronization [ Optional ]

1. Still on the **master1** host, verify user IDs:

oc get users

1. If there are users with the **htpasswd\_auth** identity, delete them as shown in this example:

oc delete user andrew

1. Delete any **htpasswd** identities as shown in this example:

oc delete identity htpassword:uid=andrew

1. As a test, run the synchronization:

oc adm groups sync --sync-config=/etc/origin/master/groupsync.yaml --whitelist=/etc/origin/master/whitelist.yaml

**Sample Output**

apiVersion: v1

items:

- apiVersion: v1

kind: Group

metadata:

annotations:

openshift.io/ldap.sync-time: 2018-04-18T20:32:48Z

openshift.io/ldap.uid: cn=portalapp,cn=groups,cn=accounts,dc=shared,dc=example,dc=opentlc,dc=com

openshift.io/ldap.url: ipa.shared.example.opentlc.com:389

creationTimestamp: null

labels:

openshift.io/ldap.host: ipa.shared.example.opentlc.com

name: portalapp

users:

- andrew

- portal1

- portal2

- chenzhen

[...]

4.5. Synchronize Groups

1. Run the same **oc adm groups sync** command, but add **--confirm** to create the groups:

oc adm groups sync --sync-config=/etc/origin/master/groupsync.yaml --whitelist=/etc/origin/master/whitelist.yaml --confirm

**Sample Output**

group/portalapp

group/paymentapp

group/ocp-production

group/ocp-platform

1. Verify that the groups are created:

oc get groups

**Sample Output**

NAME USERS

ocp-platform david, admin1, admin2, luochen, loren, chenzhen

ocp-production karla, prod1, prod2, luochen, loren, chenzhen

paymentapp marina, payment1, payment2, chenzhen

portalapp andrew, portal1, portal2, chenzhen

1. Disconnect from the **master1** host, and return to the **bastion** host.

5. Post installation configuration

In this section, you create persistent volumes (PVs) for users to consume.

* Create 25 PVs with these parameters:
  + Size: 5 GB
  + PV access: **ReadWriteOnce**
  + ReclaimPolicy: **Recycle**
* Create 25 PVs with these parameters:
  + Size: 10 GB
  + PV access: **ReadWriteMany**
  + ReclaimPolicy: **Retain**

5.1. Create PVs for Users

1. Create directories on the **support1.$GUID.internal** NFS server to be used as PVs in the OpenShift cluster. These directories should be under **/srv/nfs** because this directory is backed by a separate volume group.

**Paste the following into your bastion host command line as root**

export GUID=`hostname|awk -F. '{print $2}'`

echo "export GUID=$GUID" >> ~/.bashrc

echo $GUID

ssh support1.$GUID.internal

sudo -i

mkdir -p /srv/nfs/user-vols/pv{1..200}

for pvnum in {1..50} ; do

echo /srv/nfs/user-vols/pv${pvnum} \*(rw,root\_squash) >> /etc/exports.d/openshift-uservols.exports

chown -R nfsnobody.nfsnobody /srv/nfs

chmod -R 777 /srv/nfs

done

systemctl restart nfs-server

exit

exit

1. On your bastion create 25 definition files for PVs **pv1** to **pv25** with a size of 5 GB and **ReadWriteOnce** access mode.
2. export GUID=`hostname|awk -F. '{print $2}'`
3. export volsize="5Gi"
4. mkdir /root/pvs
5. for volume in pv{1..25} ; do
6. cat << EOF > /root/pvs/${volume}
7. {
8. "apiVersion": "v1",
9. "kind": "PersistentVolume",
10. "metadata": {
11. "name": "${volume}"
12. },
13. "spec": {
14. "capacity": {
15. "storage": "${volsize}"
16. },
17. "accessModes": [ "ReadWriteOnce" ],
18. "nfs": {
19. "path": "/srv/nfs/user-vols/${volume}",
20. "server": "support1.${GUID}.internal"
21. },
22. "persistentVolumeReclaimPolicy": "Recycle"
23. }
24. }
25. EOF
26. echo "Created def file for ${volume}";

done;

1. On your bastion create 25 definition files for PVs **pv26** to **pv50** with a size of 10 GB and **ReadWriteMany** access mode.
2. export GUID=`hostname|awk -F. '{print $2}'`
3. export volsize="10Gi"
4. for volume in pv{26..50} ; do
5. cat << EOF > /root/pvs/${volume}
6. {
7. "apiVersion": "v1",
8. "kind": "PersistentVolume",
9. "metadata": {
10. "name": "${volume}"
11. },
12. "spec": {
13. "capacity": {
14. "storage": "${volsize}"
15. },
16. "accessModes": [ "ReadWriteMany" ],
17. "nfs": {
18. "path": "/srv/nfs/user-vols/${volume}",
19. "server": "support1.${GUID}.internal"
20. },
21. "persistentVolumeReclaimPolicy": "Retain"
22. }
23. }
24. EOF
25. echo "Created def file for ${volume}";

done;

1. Use **oc create** to create all of the PVs you defined.

cat /root/pvs/\* | oc create -f -

1. Double check your PVs.

oc get pv

**Sample Output**

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

etcd-asb-volume 10G RWO Retain Bound openshift-ansible-service-broker/etcd 22m

logging-volume 10Gi RWO Retain Bound logging/logging-es-0 22m

metrics-volume 10Gi RWO Retain Bound openshift-infra/metrics-cassandra-1 22m

prometheus-alertbuffer-volume 10Gi RWO Retain Bound openshift-metrics/prometheus-alertbuffer 22m

prometheus-alertmanager-volume 10Gi RWO Retain Bound openshift-metrics/prometheus-alertmanager 22m

prometheus-volume 10Gi RWO Retain Bound openshift-metrics/prometheus 22m

pv1 5Gi RWO Recycle Available 3s

pv10 5Gi RWO Recycle Available 3s

pv11 5Gi RWO Recycle Available 3s

pv12 5Gi RWO Recycle Available 3s

pv13 5Gi RWO Recycle Available 3s

[...]

5.2. Fix NFS Persistent Volume Recycling

For persistent volumes with type **recycling** OpenShift does no longer provide the recycler pod automatically since it is deprecated. This pod however is needed to properly re-use persistent volumes.

The container image still exists with tag **latest** in the Red Hat registry. It only needs to be pulled to all the nodes and made available for use by tagging the image with the exact version installed (note that the version can change in the future):

ansible nodes -m shell -a "docker pull registry.access.redhat.com/openshift3/ose-recycler:latest"

ansible nodes -m shell -a "docker tag registry.access.redhat.com/openshift3/ose-recycler:latest registry.access.redhat.com/openshift3/ose-recycler:v3.9.30"

6. Smoke Test

In order to verify that your cluster is set up correctly it is always a good idea to execute a smoke test. This should verify that your PVs are working, you can build an application, the application image can be pushed to the registry, your pod will be running and the routers can route traffic to the application.

A simple smoke test can be done using the **nodejs-mongo-persistent** template. This template creates a MongoDB database with persistent storage. It also builds a Node.JS application from source code and pushes it into the registry. Finally when the application is running the route can be used to validate that the routers are working correctly.

Make sure to create a new project for this smoke test - it is generally a rather bad idea to deploy any applications into the **default**project on the OpenShift cluster.

1. Create a new project:

oc new-project smoke-test

**Sample Output**

Now using project "smoke-test" on server "https://loadbalancer1.GUID.internal:443".

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

oc new-app centos/ruby-22-centos7~https://github.com/openshift/ruby-ex.git

to build a new example application in Ruby.

1. Create the Node.JS Application:

oc new-app nodejs-mongo-persistent

**Sample Output**

--> Deploying template "openshift/nodejs-mongo-persistent" to project smoke-test

Node.js + MongoDB

---------

An example Node.js application with a MongoDB database. For more information about using this template, including OpenShift considerations, see https:*//github.com/openshift/nodejs-ex/blob/master/README.md.*

The following service(s) have been created in your project: nodejs-mongo-persistent, mongodb.

For more information about using this template, including OpenShift considerations, see https:*//github.com/openshift/nodejs-ex/blob/master/README.md.*

\* With parameters:

\* Name=nodejs-mongo-persistent

\* Namespace=openshift

\* Memory Limit=512Mi

\* Memory Limit (MongoDB)=512Mi

\* Volume Capacity=1Gi

\* Git Repository URL=https:*//github.com/openshift/nodejs-ex.git*

\* Git Reference=

\* Context Directory=

\* Application Hostname=

\* GitHub Webhook Secret=XdnlCWtW1p1KkBJBHwQMftofg1QUvWwvJtwIyvYf # generated

\* Generic Webhook Secret=t1gQT4xLApVDiGqmTLin6cXUapGP5TDG6OIXATBq # generated

\* Database Service Name=mongodb

\* MongoDB Username=user5UB # generated

\* MongoDB Password=8RjsBR14VKpWrWeb # generated

\* Database Name=sampledb

\* Database Administrator Password=7sxiK0lJNTN0rkmM # generated

\* Custom NPM Mirror URL=

--> Creating resources ...

secret "nodejs-mongo-persistent" created

service "nodejs-mongo-persistent" created

route "nodejs-mongo-persistent" created

imagestream "nodejs-mongo-persistent" created

buildconfig "nodejs-mongo-persistent" created

deploymentconfig "nodejs-mongo-persistent" created

persistentvolumeclaim "mongodb" created

service "mongodb" created

deploymentconfig "mongodb" created

--> Success

Access your application via route 'nodejs-mongo-persistent-smoke-test.apps.GUID.example.opentlc.com'

Build scheduled, use 'oc logs -f bc/nodejs-mongo-persistent' to track its progress.

Run 'oc status' to view your app.

1. Watch the build and mongo pods:

watch oc get pod

**Sample Output**

Every 2.0s: oc get pod Wed Apr 18 16:08:29 2018

NAME READY STATUS RESTARTS AGE

mongodb-1-deploy 1/1 Running 0 41s

mongodb-1-dx2kx 0/1 Running 0 38s

nodejs-mongo-persistent-1-build 1/1 Running 0 43s

1. When the build has finished and both the **mongodb** as well as the **nodejs-mongo-persistent** pod are running (showing **1/1** in the **READY** column) retrieve the route:

oc get route

**Sample Output**

NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD

nodejs-mongo-persistent nodejs-mongo-persistent-smoke-test.apps.GUID.example.opentlc.com nodejs-mongo-persistent <all> None

1. In a web browser navigate to the route (replacing **GUID** with your specific GUID). Your application should be running and showing a database connection at the bottom right of the page.
2. Delete your smoke test project:

oc delete project smoke-test

You can be reasonably certain that most aspects of your cluster work satisfactory.

Build Version: c3147ce9f77191e30b447cc423f2f68a0c40fc03 : Last updated 2018-07-31 01:29:02 EDT