Red Hat OpenShift Admin I (v3.9) DO280/EX280

1. Installation - Ansible inventory file & vars

```
[workstations]
[nfs]
[masters]
[etcd]
[nodes]
  openshift_node_labels                             # e.g. "{'region':'infra',
      'node-role.kubernetes.io/compute':'true'}"
[OSEv3:children]
[nodes:vars]
  # pre-installation vars
  registry_local                                    # registry.lab.example.com
  use_overlay_driver                                # true
  insecure_registry                                 # false
  run_docker_offline                                # true
  docker_storage_device                             # /dev/vdb
[OSEv3:vars]
  # general vars
  openshift_deployment_type                         # openshift-enterprise
  openshift_release                                 # v3.9
  openshift_image_tag                               # v3.9.14
  openshift_disable_check                           # disk_availability,docker_storage,memory_availability
  # networking
  os_firewall_use_firewalld                         # true
  openshift_master_api_port                         # 443
  openshift_master_console_port                     # 443
  openshift_master_default_subdomain                # apps.lab.example.com
  # authentication
  openshift_master_identity_providers               # [{'name':'htpasswd_auth', 'login':'true',
      'challenge':'true',
    'kind':'HTPasswdPasswordIdentityProvider','filename':'/etc/origin/master/htpasswd'}]
  openshift_master_htpasswd_users                   # {'user':'<<HASH>>'}
                                                    # openssl passwd -apr1 <PASSWORD> or htpasswd -nbm <USER>
                                                        <PASSWORD>
  # nfs
  openshift_enable_unsupported_configurations       # true
  openshift_hosted_registry_storage_kind            # nfs
  openshift_hosted_registry_storage_access_modes    # ReadWriteMany
  openshift_hosted_registry_storage_nfs_directory   # /exports
  openshift_hosted_registry_storage_nfs_options     # "*(rw,root_squash)"
  openshift_hosted_registry_storage_volume_name     # registry
  openshift_hosted_registry_storage_volume_size     # 40Gi
  # etcd
  openshift_hosted_etcd_storage_kind                # nfs
  openshift_hosted_etcd_storage_access_modes        # ["ReadWriteOnce"]
  openshift_hosted_etcd_storage_nfs_directory       # /exports
  openshift_hosted_etcd_storage_nfs_options         # "*(rw,root_squash,sync,no_wdelay)"
  openshift_hosted_etcd_storage_volume_name         # etcd-vol2
  openshift_hosted_etcd_storage_volume_size         # 1G
  openshift_hosted_etcd_storage_labels              # {'storage':'etcd'}
  # disconnected installation
  oreg_url                                          #
      registry.lab.example.com/openshift3/ose-${component}:${version}
  openshift_examples_modify_imagestreams            # true
  openshift_docker_additional_registries            # registry.lab.example.com
  openshift_docker_blocked_registries               # registry.lab.example.com,docker.io
  # image prefixes
  openshift_web_console_prefix                      # registry.lab.example.com/openshift3/ose-
  openshift_cockpit_deployer_prefix                 # 'registry.lab.example.com/openshift3'
```

```
  openshift_service_catalog_image_prefix          # registry.lab.example.com/openshift3/ose-
  openshift_service_broker_prefix                 # registry.lab.example.com/openshift3/ose-
  openshift_service_broker_image_prefix           # registry.lab.example.com/openshift3/ose-
  openshift_service_broker_etcd_image_prefix      # registry.lab.example.com/rhel7
  # metrics
  openshift_metrics_install_metrics               # true
```

2. Installation process

```
sudo yum install atomic-openshift-utils
# Prerequisites - FROM THE DIR WITH 'ansible.cfg'!
ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/prerequisites.yml
# Deploy - FROM THE DIR WITH 'ansible.cfg'!
ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/deplpoy_cluster.yml
```

3. Post-installation process

```
oc login -u <USER> -p <PASSWORD> --insecure-skip-tls-verify=true
oc get nodes --show labels
ssh master.lab.example.com
sudo -i
oc adm policy add-cluster-role-to-user cluster-admin <USER>
oc explain
```

4. Creating a route

a/ Generate private key

```
openssl genrsa -out <hello.apps.lab.example.com.key> 2048
```

b/ Generate CSR (request)

```
openssl req -new -key <hello.apps.lab.example.com.key> -out <hello.apps.lab.example.com.csr> \
  -subj "/C=US/ST=NC/L=Raileigh/O=RedHat/OU=RH/CN=hello.apps.lab.example.com"
```

c/ Generate certificate

```
openssl x509 -req -days 365 -in <hello.apps.lab.example.com.csr> -signkey <hello.apps.lab.example.com.key> \
  -out <hello.apps.lab.example.com.crt>
```

d/ Create secure edge-terminated route

```
oc create route edge --service=hello --hostname=hello.apps.lab.example.com --key=hello.apps.lab.example.com
    \
  --cert=hello.apps.lab.example.com.crt
oc types
oc get routes
oc get route/hello -o yaml
oc get pods -o wide
ssh node1 curl -vvv http://<IP>:8080              # IP from the previous command

# Troubleshooting:
oc describe svc hello-openshift [-n <NAMESPACE>]
oc describe pod <hello-openshift-1-abcd>
oc edit svc hello-openshift
oc edit route hello-openshift
```

5. ImageStreams

```
oc new-app --name=hello -i php:5.4 \              # -i = imagestream
  http://services/lab/example.com/php-helloworld  # git repository
oc describe is php -n openshift
oc get pods -o wide
oc logs hello-1-build
oc get events
ssh root@master oc get nodes
ssh root@node1 systemctl status atomic-openshift-node
ssh root@node1 systemctl status docker
oc describe is
```

6. Common problems

```
oc delete all -l app=<node-hello>
oc get all
oc describe pod <hello-1-deploy>
oc get events --sort-by='.metadata.creationTimestamp'
oc get dc <hello> -o yaml
sudo vi /etc/sysconfig/docker
oc rollout latest hellp
oc logs <hello-2-abcd>
pc expose service --hostname=hello.apps.lab.example.com <node-hello>
oc debug pod <PODNAME>
```

7. Secrets

```
oc create secret generic <mysql> --from-literal='database-user'='mysql' \
  --from-literal='database-password'='r3dh4t'
  --from-literal='database-root-password'='redhat'
oc get secret <mysql> -o yaml
oc new-app --file=mysql.yml
oc port-forward <pod> <local>:<on the pod>        # oc port-forward mysql-1-abcd 3306:3306
```

8. User accounts, access

```
ssh root@master htpasswd /etc/origin/master/htpasswd <USER>
```

a/ Remove capability to create projects for all regular users

```
oc login -u <admin> -p <redhat> <master>
oc adm policy remove-cluster-role-from-group self-provisioner system:authenticated
    system:authenticated:oauth
```

b/ Associate user with secure project

```
oc login -u <admin> -p <redhat>
oc new-project <secure>
oc project <secure>                               # you don't have to do this, if you then specify -n (last
    command)
oc policy add-role-to-user edit <user>
oc policy add-role-to-user edit <user> -n <secure># you don't have to do this, if you switched to the
    namespace already
```

c/ Pass environment variable to the new app

```
oc new-app --name=phpmyadmin --docker-image=registry.lab.example.com/phpmyadmin:4.7 -e PMA_HOST=mysql.secure-revi
```

d/ Failed deployment because of the default security

Enable container to run with root privileges:

```
oc login -u <admin> -p <redhat>
oc create serviceaccount <phpmyadmin-account>
oc adm policy add-scc-to-user anyuid -z <phpmyadmin-account>
```

e/ Use & update deployment with the new service account

`oc edit dc/phpmyadmin` # or this command:

```
oc patch dc/phpmyadmin --patch '{"spec":{"template":{"spec":{"serviceAccountName":"<phpmyadmin-account>"}}}}'
```

JSON representation of the above:

```
{
  "spec": {
    "template": {
      "spec": {
        "serviceAccountName": "<phpmyadmin-account>"
      }
    }
  }
}
```

9. Persistent volume

`cat mysqldb-volume.yml`

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysqldb-volume
spec:
  capacity:
    storage: 3Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: /var/export/dbvol
    server: services.lab.example.com
  persistentVolumeReclaimPolicy: Recycle
```

```
oc create -f <mysqldb-volume.yml>
oc get pv
oc status -v
oc describe pod <mysqldb>
oc set volume dc/<mysqldb> --add --overwrite --name=<mysqldb-volume-1> -t pvc
    --claim-name=<mysqldb-pvclaim> \
  --claim-size=<3Gi> --claim-mode=<'ReadWriteMany'>
oc get pvc
```

```
# Important knowledge about PV/PVC:
  - PV doesn't have a namespace
  - Allocated capacity of PVC may be bigger than requested capacity, imagine a scenario:
    1. Create 'review-pv' PV of 3Gi
    2. Create a new app from template with PVC called 'mysql-pvc' of 1Gi with 'review-pv' selector (step 1.)
    3. In the template, there is "container" in the "DeploymentConfig" using "volumeMounts" with the name of
       'mysql-data' mounting it to '/var/lib/mysql/data'
    4. In the template, there is "volumes" object 'mysql-data' using "persistenVolumeClaim" with
        "claimName" of mysql-pvc
    - What happens is following:
      - 'mysql-pvc' is bound to volume 'review-pv'
      - it has requested capacity of 1GiB, but allocated 3GiB
      - if the selector in PVC is not specified, it will automatically find the closest one
```

10. Controlling scheduling & scaling

```
# Scaling:
oc new-app -o yaml -i php:7.0 http://registry.lab.example.com/scaling > scaling.yml
oc describe dc <scaling> | grep 'Replicas'
oc scale --replicas=5 dc <scaling>
```

```
oc get nodes -L region
oc label node <node2.lab.example.com> region=<apps> --overwrite
oc get dc/hello -o yaml > <hello.yml>
```

hello.yml

```
nodeSelector:
  region: apps
```

```
oc apply -f <hello.yml>
oc label node node1.lab.example.com region=apps --overwrite
```

a/ Disable scheduling on node2

```
oc adm manage-nmode --schedulable=false <node2.lab.example.com>
```

b/ Delete/drain all pods on node2

```
oc adm drain <node2.lab.example.com> --delete-local-data
```

c/ Load Docker image locally

```
docker load -i <phpmyadmin-latest.tar>
```

d/ Tag local image ID

```
docker tag <123abcdef> <docker-registry-default.apps.lab.example.com/phpmyadmin:4>
docker images
```

e/ Login to OpenShift internal image registry

```
TOKEN=$(oc whoami -t)
```

```
docker login -n developer -p ${TOKEN} docker-registry-default.apps.lab.example.com

# Certificate signed by unknown authority:
scp registry.crt root@master:/etc/origin/master/registry.crt
/etc/pki/ca-trust/source/anchors/docker-registry-default.apps.lab.example.com.crt
update-ca-trust
systemctl restart docker
<<RUN DOCKER LOGIN AGAIN>>>
```

11. Metrics subsystem

a/ Verify images required by metrics

```
docker-registry-cli <registry.lab.example.com> search <metrics-cassandra> ssl
```

```
# Output:
openshift3/ose-metrics-hawkular-metrics:v3.9
openshift3/ose-metrics-heapster:v3.9
openshift3/ose-metrics-cassandra:v3.9
openshift3/ose-metrics-recycler:v3.9
```

b/ Check NFS

```
ssh root@services cat /etc/exports.d/openshift-ansible.exports
```

c/ Create PV for NFS share

```
cat metrics-pv.yml
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: metrics
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce                             # Must have this!
  nfs:
    path: /exports/metrics
    server: services.lab.example.com
  persistenVolumeReclaimPolicy: Recycle
```

```
oc get pv
```

d/ Add to Ansible inventory file

```
[OSEv3:vars]
  openshift_metrics_install_metrics               # true
  openshift_metrics_image_prefix                  # registry.lab.example.com/openshift3/ose-
  openshift_metrics_image_version                 # v3.9
  openshift_metrics_heapster_request_memory       # 300M
  openshift_metrics_hawkular_request_memory       # 750M
  openshift_metrics_cassandra_request_memory      # 750M
  openshift_metrics_cassandra_storage_type        # pv
  openshift_metrics_cassandra_pvc_size            # 5Gi
  openshift_metrics_cassandra_pvc_prefix          # metrics
```

e/ Run Ansible, verify if it's OK

```
oc get pvc -n openshift-infra
oc get pod -n openshift-infra
oc adm diagnostics MetricsApiProxy
```

f/ Top command as admin

```
oc adm top node --heapster-namespace=openshift-infra --heapster-scheme=https
```

12. Limits

```
oc describe node <node1.lab.example.com>
oc describe node <node2.lab.example.com>
# Look for allocated resources (| grep -A 4 Allocated)
# After you deploy new app, allocated resources do NOT change
```

```
cat limits.yml
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: project-limits
spec:
```

```
  limits:
    - type: container
      default:
        cpu: 250m
```

```
oc describe limits
```

```
cat quota.yml
```

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: project-quota
spec:
  hard:
    cpu: 900m
```

```
# Same as:
oc quota project-quota --hard=cpu=900m
oc describe quota
# After you deploy an app, it will consume the project quota
```

```
oc describe pod <hello-1-abcdef> | grep -A 2 Requests
# When you scal up and get over the quota, resources will not be created
oc get resourcequota --list-all-quotas
oc get events | grep -i error
oc set resources dc hello --requests=memory=256Mi
# Memory request is not counted against the project quota
```

13. Readiness/liveness

```
oc status
curl http://probe.apps.lab.example.com/health
curl http://probe.apps.lab.example.com/ready
<<CREATE PROBES IN WEB GUI>>
oc get events --sort-by='.metadata.CreationTimestamp' | grep 'probe failed'
```

```
readinessProbe:
  httpGet:                                    # TCP Socket
    path: /health
    port: 8888
  initialDelaySeconds: 15
  timeoutSeconds: 1
```

```
livenessProbe:
  exec:
    command:
      - cat
      - /tmp/health
  initialDelaySeconds: 15
  timeoutSeconds: 1
```

14. FAQs

a/ Import the template into OpenShift

```
oc apply -n openshift -f <template.yml>
```

b/ Import the Docker image to OpenShift

```
oc import-image <stream> --from=registry.lab.example.com/todoapp/todoui --confirm -n <todoapp>
```

c/ Turn service into NodePort

```
oc edit svc <hello>
```

```
. . .
  ports:
    - name: 8080-tcp
      . . .
      nodePort: 30800
  type: NodePort
. . .
```

d/ Access shell inside the pod

```
oc rsh <hello-1-abcdef>
```

e/ Export resource to YAML

```
oc export pod <hello-1-abcdef> > pod.yml
# As template:
oc export svc,dc hello --as-template=docker-hello > template.yml
```

f/ Configure router to handle wildcard routes

```
oc scale dc/router --replicas=0
oc set env dc/router ROUTER_ALLOW_WILDCARD_ROUTES=true
oc scale dc/router --replicas=3
oc expose svc test --wildcard-policy-subdomain --hostname='www.lab.example.com'
```

g/ Autocomplete

```
source /etc/bash_completion.d/oc
```

h/ Troubleshooting policies

```
oc describe clusterPolicyBindings :default
oc describe policyBindings :default
```

i/ Security Context Constraints (SCCs)

```
oc get scc
oc create serviceaccount <account>
```

```
# SCCs:
- anyuid
- hostaccess
- hostmount-anyuid
- nouroot
- privileged
- restricted
```

```
# Default SELinux policies do not allow containers to access NFS shares!
setsebool -p virt_use_nfs=true
setsebool -p virt_sandbox_use_nfs=true
```

j/ ConfigMap

```
oc create configmap <special-config> --from-literal=serverAddress=172.20.30.40
```

```
# ConfigMaps:
  --from-literal=KEY=VALUE
  --from-file=directory/file
  --from-file=directory/
  --from-file=KEY=directory/file
# Consuming using "configMapKeyRef"

# List all ENV:
oc env dc/printenv --list
```

k/ RBAC table

Name of the role	Permissions
cluster-admin	superuser
cluster-status	read-only
edit	no admin, no quota, no access mgmt
basic-user	read account
self-provisioner	cluster role to create new project(s)
admin	anything

l/ Autoscale pods

```
oc autoscale dc/myapp --min 1 --max 5 --cpu-percent=80
oc get hpa/frontend
```

m/ Tag images

```
oc tag <ruby:latest> <ruby:2.0>
# Options:
#   --alias=true
#   --scheduled=true
#   --reference-policy=local
```

n/ Docker import vs Docker load

```
# Docker import
  - Create an empty filesystem image and import the contents of the tarball into it.
# Docker load
  - Load an image from a file or STDIN. Restores both images & tags. Write image names or IDs imported into
      STDOUT.
```

o/ OpenShift output vs export

```
oc get <RES> -o yaml
oc export <RES>
# Export will show object definition without any runtime specifics
```

p/ A/B routing

```
oc set route-backends <ROUTE> <svc1=weight>
oc set route-backends cotd cotd1=50 cotd2=50
```

q/ Link secret with service account

```
oc secret link <service-account> <secret-name>
```

r/ Process template into a list of resources

```
oc process -f <TEMPLATE> | oc create -f -         # examines template, generates parameters. To override
    params, add -v
```

s/ Examine pod contents

```
/usr/local/s2i/run
/var/run/secrets/kubernetes.io/serviceaccount
/root/buildinfo
```

t/ Delete environment variable

```
oc set env dc/d1 ENV1- ENV2- ENV3-
```

u/ Using secrets with ENV & Volumes

```
oc env dc/printenv --from=secret/printsecret
```

```
oc env dc/printenv --from=secret/printsecret --prefix=DB_
```

```
oc set volume dc/printenv --add --overwrite --name=db-conf-volume --mount-path /debconf/ --secret-name=printenv-s
```

v/ Turn off automatic triggers

```
oc set triggers dc <NAME> --manual
```

w/ Allow Jenkins to build & deploy the app

```
oc policy add-role-to-user edit system
  serviceaccount:<PROJECT_NAME>:jenkins -n <NAMESPACE>
```

Because Jenkins is in a different project than the application

Jenkins container has to be deployed first: Service Catalog > CI/CD > Jenkins (persistent)

x/ Generate values in templates

```
parameters:
  - name: PASSWORD
    description: "Random password"
    generate: expression
    from: "[a-zA-Z0-9]{12}"
```

```
# Create an application from template:
oc new-app --template=ruby-hello --param=A=B
```

To make template available accross the cluster, cluster admin must add it to the `openshift` namespace

List all parameters from mysql template: `oc process --parameters=true -n openshift mysql`