# OC Basic Cheat Sheet

Show console URL:

$ oc whoami --show-console

Show API url Copy:

$ oc whoami --show-server

Cluster infoCopy:

$ oc cluster-info

Cluster info DUMP Copy:

$ oc cluster-info dump

Get CRI-O settings Copy:

$ oc get containerruntimeconfig

Delete all 'Completed' pods:

$ oc delete pod --field-selector=status.phase==Succeeded --all-namespaces

$ oc get pods --all-namespaces |  awk '{if ($4 == "Completed") system ("oc delete pod " $2 " -n " $1 )}'

Additional methods to remove Failed, Pending, Evicted, and all 'Non-Running' pods:

$ oc delete pod --field-selector=status.phase==Failed --all-namespaces

$ oc delete pod --field-selector=status.phase==Pending --all-namespaces

$ oc delete pod --field-selector=status.phase==Evicted --all-namespaces

$ oc get pods --all-namespaces |  awk '{if ($4 != "Running") system ("oc delete pod " $2 " -n " $1 )}'

# STOP and START

From ocp bastion host    
$ export  KUBECONFIG=<PATH\_TO\_KUBECONFIG\_INSTALLATION>

or login via token from Console using LDAP credentials

$ oc get node

$ oc debug node/ocp-np-zpbhs-master-0

When access to node type

$ chroot /host

Before starting maintenance operations on a node it is necessary to exclude the node from the list of nodes.

If more master and / or worker nodes are to be excluded, it is necessary to work in rolling mode, giving priority to the masters

STOP (Draining) procedure:

Before working on the node, mark the node as unschedulable and move the pods

$ oc adm drain <node\_name> --force --delete-local-data --ignore-daemonsets

START (Uncordon) procedure:

$ oc adm uncordon <node\_name>

# Trouble shooting

## Analysis

POD Listing:

$ oc get pods --all-namespaces

Run a shell on a POD:

$ oc rsh <pod> -n <namespace>

Read POD logs:

$ oc logs <pod> -n <namespace>

List cluster nodes:

$ oc get nodes

Describe node content:

$ oc describe node <node>

## Health checks

The most important health checks to monitor is the MachineConfigPool and ClusturOperator status, that is monitoring via cli with the following command:

$ oc get co

# Describe Cluster Operator status  
NAME                                      AVAILABLE  PROGRESSING  DEGRADED

authentication                            True       False        False

baremetal                                 True       False        False

cloud-credential                          True       False        False

cluster-autoscaler                        True       False        False

config-operator                           True       False        False

console                                   True       False        False

csi-snapshot-controller                   True       False        False

dns                                       True       False        False

etcd                                      True       False        False

image-registry                            True       False        False

ingress                                   True       False        False

insights                                  True       False        False

kube-apiserver                            True       False        False

kube-controller-manager                   True       False        False

kube-scheduler                            True       False        False

kube-storage-version-migrator             True       False        False

machine-api                               True       False        False

machine-approver                          True       False        False

machine-config                            True       False        False

marketplace                               True       False        False

monitoring                                True       False        False

network                                   True       False        False

node-tuning                               True       False        False

openshift-apiserver                       True       False        False

openshift-controller-manager              True       False        False

openshift-samples                         True       False        False

operator-lifecycle-manager                True       False        False

operator-lifecycle-manager-catalog        True       False        False

operator-lifecycle-manager-packageserver  True       False        False

service-ca                                True       False        False

storage                                   True       False        False

Every CO must be in AVAILABLE status (true) and all Degread row must False

$ oc get mcp # Describe Machine Config Pool status

NAME    CONFIG                                            UPDATED  UPDATING  DEGRADED  MACHINECOUNT  READYMACHINECOUNT  UPDATEDMACHINECOUNT  DEGRADEDMACHINECOUNT

infra   rendered-infra-a0ca86477c1adc80702df03f35d5ec24   True     False     False     3             3                  3                    0

master  rendered-master-041999a65cb1e5b3d6f0447450f79070  True     False     False     3             3                  3                    0

worker  rendered-worker-a0ca86477c1adc80702df03f35d5ec24  True     False     False     3             3*3                    0*

## MachineConfigPool stuck in degraded

MachineConfigPool stuck in degraded after applying a modification  
  
$ oc get mcp | grep infra  
NAME    CONFIG                                            UPDATED  UPDATING  DEGRADED  MACHINECOUNT  READYMACHINECOUNT  UPDATEDMACHINECOUNT  DEGRADEDMACHINECOUNT

infra rendered-infra-a0ca86477c1adc80702df03f35d5ec24 False True True 1 0 0 1 17m

Check Red Hat KB [https://access.redhat.com/search/?q=\*&p=1&rows=10&documentKind=Solution](https://access.redhat.com/search/?q=*&p=1&rows=10&documentKind=Solution) for finding proper solution check the MachineConfigPool will switch to UPDATED true

$ oc get mcp | grep infra

NAME    CONFIG                                            UPDATED  UPDATING  DEGRADED  MACHINECOUNT  READYMACHINECOUNT  UPDATEDMACHINECOUNT  DEGRADEDMACHINECOUNT

infra   rendered-infra-a0ca86477c1adc80702df03f35d5ec24   True     False     False     3             3                  3                    0

## Image layer not known

The CRI-O container runtime v1.18.2 causes this problem. Here, the developer optimized the code for more performance. Unfortunately, this optimization, in some cases, does not flush file operations. As a result, image metadata is missing or corrupt. Accordingly, the underlying docker image becomes unusable.

This error can occurr  in few node when scheduling a pod:  
  
Failed to create pod sandbox: rpc error: code = Unknown desc = error creating pod sandbox with name "XXXX": layer not known

## Manually fixing corrupted images

A suitable workaround to fix the problem is to delete images from the node. After that, the container platform can load the missing docker images from the docker registry. In effect, the pods find valid docker images and start.  
Enter to node affected to this issue and type the following command:

$ sudo systemctl stop kubelet

$ sudo systemctl stop crio

$ sudo rm -rf /var/lib/containers/

$ sudo systemctl start crio

$ sudo systemctl start kubelet

## Troubleshooting DNS

Check the cluster operator to see if it is available:

*$ oc get clusteroperator dns*

Check the logs of dns-operator pod  
*$ oc logs pod/`oc get pods -o=jsonpath="{.items[0].metadata.name}" -n openshift-dns-operator` -n openshift-dns-operator 2>/dev/null || oc logs pod/`oc get pods -o=jsonpath="{.items[0].metadata.name}" -n openshift-dns-operator` -c dns-operator -n openshift-dns-operator*

Test resolving kubernetes service hostname to and from every DNS pod  
*$ oc get pods -n openshift-dns -o custom-columns="Pod Name:.metadata.name,Pod IP:.status.podIP,Node IP:.status.hostIP,Status:.status.phase"*

*$ DST\_HOST=kubernetes.default.svc.cluster.local; for dnspod in `oc get pods -n openshift-dns -o name --no-headers`; do for dnsip in `oc get pods -n openshift-dns -o go-template='{{ range .items }} {{index .status.podIP }} {{end}}'`; do echo -ne "$dnspod\tquerying $DST\_HOST to $dnsip ->\t"; oc exec -n openshift-dns $dnspod -- dig @$dnsip $DST\_HOST -p 5353 +short 2>/dev/null ; done; done*

## Restore

This procedure assumes that you copied the backup directory containing the etcd snapshot and the resources for the static pods to the /home/core/ directory of your recovery control plane host.

The official procedure to restore cluster on previous state is describe here:

<https://docs.openshift.com/container-platform/4.7/backup_and_restore/control_plane_backup_and_restore/disaster_recovery/scenario-2-restoring-cluster-state.html#dr-scenario-2-restoring-cluster-state_dr-restoring-cluster-state>

# Performance tuning

In case of performance issues, enter a list of parameters or activities to be done to fix it (e.g. configuration parameters changes, script optimization, ...)

## Check etcd Performance

Starting with OpenShift 4.5 there is a new recommended way to test the performance of etcd. The output reports whether the disk is fast enough to host etcd by comparing the 99th percentile of the fsync metric captured from the run to see if it is less than 10ms.

Run the follow command from a master node.

$ podman run --volume /var/lib/etcd:/var/lib/etcd:Z quay.io/openshift-scale/etcd-perf

## Collecting a tcpdump from a pod

* Identify the namespace and pod that you wish to trace.
* Determine the tcpdump filter and/or parameters you wish to use.
* Modify the script below to match the desired namespace, pod, and tcpdump parameters to use.

TRACE\_NS=openshift-dns

TRACE\_POD=dns-default-4mqsd

run\_on\_host() {

    oc debug node/$NODE -- chroot /host $@

}

run\_in\_toolbox() {

    oc debug node/$NODE -- $@

}

NODE=`oc get pod -n $TRACE\_NS $TRACE\_POD -o=jsonpath='{.spec.nodeName}'`

CID=`oc get pods -n $TRACE\_NS $TRACE\_POD --template '{{printf "%.21s" (index .status.containerStatuses 0).containerID}}{{"\n"}}' | sed -e 's|cri-o://||'`

PID=$(run\_on\_host crictl inspect -o yaml $CID | grep 'pid:' | awk '{print $2}')

run\_in\_toolbox nsenter -n -t $PID -- tcpdump -nn port 53 | tee tcpdump.log

# Scale in/scale out

Enter a step-by-step description of the activities necessary to add or remove hosts.

$ oc get machinesets -n openshift-machine-api  
NAME               DESIRED   CURRENT   READY   AVAILABLE   AGE

ocp-nsmwk-infra    3         3         3       3           6d3h

ocp-nsmwk-worker   3         3         3       3           14d

To scale down one infra node:

$ oc scale --replicas=2 machineset ocp-nsmwk-infra -n openshift-machine-api

To scale up one infra node:

$ oc scale --replicas=3 machineset ocp-nsmwk-infra -n openshift-machine-api

# Patching

Lot of patches to the RHCOS must be set via MachineConfig

## Apply sysctl tweaks to nodes

In order to modify sysctl parameters is recommended to create machine configs to add those parameters in the /etc/sysctl.d/ directory.

In this example, the vm.max\_map\_count parameter will be increased to 262144 in the masters hosts:

cat << EOF | oc create -f -

apiVersion: machineconfiguration.openshift.io/v1

kind: MachineConfig

metadata:

 labels:

   machineconfiguration.openshift.io/role: master

 name: 99-sysctl-elastic

spec:

 config:

   ignition:

     version: 2.2.0

   storage:

     files:

     - contents:

         # vm.max\_map\_count=262144

         source: data:text/plain;charset=utf-8;base64,dm0ubWF4X21hcF9jb3VudD0yNjIxNDQ=

       filesystem: root

       mode: 0644

       path: /etc/sysctl.d/99-elasticsearch.conf

EOF

## Disable auto rebooting with the machine-config-operator

Every change performed by the machine-config-operator triggers a reboot in the hosts where the change needs to be performed.

In the event of having a few changes to apply (such as modify NTP, registries, etc.) and specially for baremetal scenarios, the auto reboot feature can be paused by setting the spec.paused field in the machineconfigpool to true:

$ oc patch --type=merge --patch='{"spec":{"paused":true}}' machineconfigpool/master

## Modify vSphere Credentials

<https://access.redhat.com/solutions/4618011>A full explanation is available in the solution linked above, this article also covers changing the default datastore.

You will need cluster-admin to edit these credentials

*$ oc project kube-system*

*$ oc edit secret vsphere-creds -n kube-system*

These credentials are base64 encoded. Here's a quick reminder of how to encode and decode base64.

## Switch cluster version via update channels and Upgrade

Switch clusterversion channel:

$ oc patch \

  --patch='{"spec": {"channel": "stable-4.8"}}' \

  --type=merge \

  clusterversion/version

View the available updates and note the version number of the update that you want to apply:

$ oc adm upgrade

Apply an update:

$ oc adm upgrade --to-latest=true

$ oc adm upgrade --to=<version>

Unmanage operators - Create the patch yaml file and apply it

$ cat <<EOF >version-patch.yaml

- op: add

  path: /spec/overrides/-

  value:

  - kind: Deployment

    group: apps

    name: network-operator

    namespace: openshift-network-operator

    unmanaged: true

EOF

$ oc patch clusterversion version --type json -p "$(cat version-patch.yaml)"

You can verify the update with:

*$* oc get -o json clusterversion version | jq .spec.overrides

## Install package to RHCOS for debugging

For debugging purposes, install a package that is not installed in RHCOS nor toolbox by default.

# spin up a debug pod and change to root

$ oc debug node/<node-name>

$ chroot /host

# modify toolbox to use the support tools container image

$ vi ~/.toolboxrc

> REGISTRY=registry.redhat.io

> IMAGE=rhel7/support-tools:latest

# now launch toolbox, and as an example, install conntrack-tools

$ toolbox

$ yum install conntrack-tools

You may want to consider cleaning up the ~/.toolboxrc file afterwards, to revert back to the default toolbox image.

## Create static routes post cluster installation for a specific worker pool

While a rhcos node comes up the Network Manager configures the interface using the networkmanager keyfiles and the default interface details are locationed at /etc/NetworkManager/system-connections/default\_connection.nmconnection

Configure the following machine-config to run the ip route add and nmcli dev modify as a one-shot service which executes after network-online.target is up and running.

$ vi staticroute.yaml

apiVersion: machineconfiguration.openshift.io/v1

kind: MachineConfig

metadata:

  labels:

    machineconfiguration.openshift.io/role: worker

  name: worker-custom-route-configuration

spec:

  config:

    ignition:

      config: {}

      security:

        tls: {}

      timeouts: {}

      version: 3.1.0

    networkd: {}

    passwd: {}

    storage: {}

    systemd:

      units:

      - contents: |

             [Unit]

             Description=nmcli-dev-modify

             After=network-online.target

             [Service]

             Type=oneshot

             ExecStart=nmcli dev modify <ifname> +ipv4.routes "10.xx.xx.0/24 172.x.0.1"

             [Install]

             WantedBy=multi-user.target

        enabled: true

        name: nmcli-dev-modify.service

  osImageURL: ""

Once the file is created on the node, the object could be created on the cluster using the following command:

$ oc create -f staticroute.yaml