[pre-publication DRAFT]

# Debugging On Cluster Layering in OpenShift 4.20: A Practical Guide

On Cluster Layering (OCL) represents a powerful capability in OpenShift that allows customization of CoreOS-based nodes. While this feature provides unprecedented flexibility, it also introduces new layers of complexity when things go wrong.

This guide walks through common OCL debugging scenarios in OpenShift 4.20 and provides practical troubleshooting steps to get clusters back on track.

Let's dive into the tools and techniques that will make debugging OCL less mysterious and more manageable.

## Understanding the On Cluster Layering Process

When On Cluster Layering (OCL) is enabled in OpenShift, the workflow can be broken down into three stages. Each stage has distinct failure points. The stages are: MachineOSConfig creation, MachineOSBuild creation and execution, and application of the new image to the nodes.

### Stage 1: MachineOSConfig (MOSC) Creation

The process begins when a MachineOSConfig resource is created targeting a specific MachineConfigPool. This resource acts as the blueprint, defining how the custom OS image should be built and where it should be stored.

What to watch for:

- Validation errors: The resource may fail to create if required fields are missing or incorrectly configured.
- Secret references: Ensure all referenced pull/push secrets exist in the correct namespace (openshift-machine-config-operator).
- Registry specifications: Verify that renderedImagePushSpec points to a valid, accessible registry location.

At this stage, issues are typically configuration errors that prevent the resource from being created or accepted by the cluster.

## Success

If the MOSC resource was successfully created, the machine-os-builder pod should be healthy and running in the MCO namespace.

```
None

$ oc get pods -n openshift-machine-config-operator -l k8s-app=machine-os-builder

NAME                                READY   STATUS    RESTARTS   AGE

machine-os-builder-b8f48488f-lth94   1/1     Running   0          44s
```

If this pod is visible and running, debugging can proceed to the next step, which builds the image.

If this pod is not visible, it indicates that some errors have occurred.

## Error

If a forbidden value is used for any of the fields in MachineOSConfig, it should be printed in the create command output:

None

```
# Bad mosc resource

$ cat ./mosc.yaml

apiVersion: machineconfiguration.openshift.io/v1

kind: MachineOSConfig

metadata:

  name: worker

spec:

  machineConfigPool:

    name: infra

  currentImagePullSecret:

    name: current-image-pull

  imageBuilder:

    imageBuilderType: Job

  baseImagePullSecret:

    name: base-image-pull

  renderedImagePushSecret:

    name: rendered-image

  renderedImagePushSpec: "quay.io/sregidor/sregidor-os:mco_layering"
```

```
$ oc create -f ./mosc.yaml

The MachineOSConfig "worker" is invalid:

* spec.imageBuilder.imageBuilderType: Unsupported value: "job": supported values: "Job"
```

The example shows that "spec.imageBuilder.imageBuilderType" is set to "job" instead of the required "Job" (with a capital “J”).

Another example:

```
None

$ oc create -f ./mosc.yaml

The MachineOSConfig "worker" is invalid: <nil>: Invalid value: "object": MachineOSConfig name must match the referenced
MachineConfigPool name; can only have one MachineOSConfig per MachineConfigPool
```

If the configured values are not forbidden but nevertheless are causing problems, the information to detect those problems can be found in the machine-config-operator pod, in the triggered events, and in the machine-config ClusterOperator. The most detailed information can be found in the machine-config-operator pod.

For example:

```
None

# Error: The secrets were not created.

$ cat ./mosc.yaml

apiVersion: machineconfiguration.openshift.io/v1
```

```
kind: MachineOSConfig
metadata:
  name: infra
spec:
  machineConfigPool:
    name: infra
  currentImagePullSecret:
    name: current-image-pull
  imageBuilder:
    imageBuilderType: Job
  baseImagePullSecret:
    name: base-image-pull
  renderedImagePushSecret:
    name: rendered-image
  renderedImagePushSpec: "quay.io/sregidor/sregidor-os:mco_layering"

# The resource can be created
$ oc create -f ./mosc.yaml
machineosconfig.machineconfiguration.openshift.io/infra created
```

```
# The builder pod is not created

$ oc get pods -n openshift-machine-config-operator |grep build



# The error can be found in the machine-config-operator pod

$ oc logs -n openshift-machine-config-operator machine-config-operator-7498f4576b-h5vzj

...

E1017 08:56:53.431756       1 operator.go:467] "Unhandled Error" err="could not update Machine OS Builder deployment:
could not validate renderedImagePushSecret \"rendered-image\" for MachineOSConfig infra: secret rendered-image from
infra is not found. Did you use the right secret name?"

...



# And there are events reporting this error too

$ oc get events  -n openshift-machine-config-operator --sort-by metadata.creationTimestamp  |tail -3

34s         Warning   OperatorDegraded: MachineOSBuilderFailed   /machine-config
Failed to resync 4.20.0-0-2025-10-16-080835-test-ci-ln-bfn63jk-latest because: could not update Machine OS Builder
deployment: could not validate renderedImagePushSecret "rendered-image" for MachineOSConfig infra: secret
rendered-image from infra is not found. Did you use the right secret name?

11s         Warning   OperatorDegraded: MachineOSBuilderFailed   /machine-config
Failed to resync 4.20.0-0-2025-10-16-080835-test-ci-ln-bfn63jk-latest because: could not update Machine OS Builder
deployment: could not validate baseImagePullSecret "base-image-pull" for MachineOSConfig infra: secret base-image-pull
from infra is not found. Did you use the right secret name?

96s         Normal    ConfigMapUpdated                           deployment/machine-config-operator
Updated ConfigMap/kube-rbac-proxy -n openshift-machine-config-operator:...
```

```
# The machine-config ClusterOperator can be checked too

$ oc get co machine-config

NAME             VERSION                                                AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE

machine-config   4.20.0-0-2025-10-16-080835-test-ci-ln-bfn63jk-latest   True        False         True       76m
Failed to resync 4.20.0-0-2025-10-16-080835-test-ci-ln-bfn63jk-latest because: could not update Machine OS Builder
deployment: could not validate renderedImagePushSecret "rendered-image" for MachineOSConfig infra: secret
rendered-image from infra is not found. Did you use the right secret name?
```

## Stage 2: MachineOSBuild (MOSB) Creation and Image Build Process

Once the MachineOSConfig is successfully created and the machine-os-builder pod is running, MCO automatically generates a MachineOSBuild resource. The MachineOSBuild resource will control an actual image build job that pulls the base CoreOS image, applies the customizations (via Containerfile), and pushes the result to the specified registry.

In order to execute this process, several auxiliary secrets and configmaps will be created in the MCO namespace.

What to watch for:

- Build status: Monitor the MachineOSBuild resource for conditions showing Succeeded=True or Failed=True.
- Job failures: Check if the build job in openshift-machine-config-operator namespace completes successfully.
- Image pull errors: Authentication failures when pulling the base image indicate problems with baseImagePullSecret.

- Build errors: Containerfile syntax issues, missing packages, or failed RUN commands will cause build failures.
- Image push errors: Problems pushing to the registry suggest issues with renderedImagePushSecret or registry permissions.

This is where most OCL failures occur, as it involves pulling images, executing build steps, and pushing results, all of which depend on external resources and credentials.

The following output is displayed while the image is being built:

```
None

# The MachineOSBuild resource

$ oc -n openshift-machine-config-operator get machineosbuild

NAME                                    PREPARED   BUILDING   SUCCEEDED   INTERRUPTED   FAILED   AGE

infra-b1b93a87b88b18b3ad70e9fb2596b2cd   False      True       False       False         False    108s

# The job created in the MachineOSBuild execution

$ oc -n openshift-machine-config-operator get job

NAME                                          STATUS     COMPLETIONS   DURATION   AGE

build-infra-b1b93a87b88b18b3ad70e9fb2596b2cd   Running    0/1           105s       105s

# The pod controlled by the Job, that will execute the actual build process

$ oc -n openshift-machine-config-operator get pods

NAME                                                            READY   STATUS      RESTARTS       AGE

build-infra-b1b93a87b88b18b3ad70e9fb2596b2cd-q7tsb               0/1     Init:0/1    0              2m49s

...
```

NOTE: Only changes to kernel arguments, kernel type, OSImageURL, or extension bundles will create a new Job and trigger a new image build process. All other MachineConfig changes reuse the existing MOSB and do not trigger a new build.

## Success

This stage can be considered successful if:

- The MachineOSBuild was created and is reporting Succeeded=True Failed=False.
- The Job is automatically removed by the machine-os-builder pod.

```
None

$ oc -n openshift-machine-config-operator get machineosbuild

NAME                                     PREPARED   BUILDING   SUCCEEDED   INTERRUPTED   FAILED   AGE

infra-f509ba5b2d76bcc5a113fd81de75ee99   False      False       True       False         False    4m14s
```

If the MachineOSBuild resource is not created or is not successful, it indicates that an error has occurred.

## Error

### The MachineOSBuild resource was not created

The process in charge of creating the MachineOSBuild resource is the machine-os-builder pod.

This error is not very common, but if it happens, the logs in this pod need to be read to find the causes:

```
None

$ oc -n openshift-machine-config-operator logs machine-os-builder-b8f48488f-nsdbk

....

I1017 09:42:42.524084       1 reconciler.go:634] New MachineOSBuild created: infra-f509ba5b2d76bcc5a113fd81de75ee99

....
```

### The MachineOSBuild was created but failed

The most common cause of a failed MachineOSBuild is that the Job building the image failed to build it

If the MachineOSBuild (MOSB) resource fails, the first step is to locate the associated Job.

### The Job was not created

The process in charge of creating/deleting this job is the machine-os-builder pod. If the job cannot be found, the logs in this pod should be read for further information.

```
None

  $ oc -n openshift-machine-config-operator logs machine-os-builder-b8f48488f-nsdbk

...
```

### Debugging a failed Job

For example, the following MOSC is created:

```
None

$ cat mosc.yaml

apiVersion: machineconfiguration.openshift.io/v1

kind: MachineOSConfig

metadata:

  name: infra
```

```
spec:
  machineConfigPool:
    name: infra
  currentImagePullSecret:
    name: current-image-pull
  imageBuilder:
    imageBuilderType: Job
  baseImagePullSecret:
    name: base-image-pull
  renderedImagePushSecret:
    name: rendered-image
  renderedImagePushSpec: "quay.io/sregidor/sregidor-os:mco_layering"
  containerFile:
      - content: |-
          RUN curl --fail -L https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64_wrong -o
/usr/bin/yq && chmod +x /usr/bin/yq


$ oc create -f mosc.yaml
machineosconfig.machineconfiguration.openshift.io/infra created
```

The MachineConfigPool shows as degraded:

```
None

$ oc get mcp infra

NAME    CONFIG                                            UPDATED   UPDATING   DEGRADED   MACHINECOUNT
READYMACHINECOUNT   UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE

infra   rendered-infra-6208c0db8119cfe2c9c4e42099617a43   False     False      True       1              0
0                     0                       158m


$ oc get mcp infra -oyaml

...

  - lastTransitionTime: "2025-10-17T10:55:00Z"

    message: 'Failed to build OS image for pool infra (MachineOSBuild: infra-32ef35dea3e553071277954842edb33a):

      Failed: Build Failed'

    reason: BuildFailed

    status: "True"

    type: ImageBuildDegraded

...
```

The MOSB resource shows as failing:

```
None

$ oc -n openshift-machine-config-operator get machineosbuild

NAME                                      PREPARED   BUILDING   SUCCEEDED   INTERRUPTED   FAILED   AGE

infra-32ef35dea3e553071277954842edb33a   False      False      False       False         True     31m
```

The job can be located as follows:

```
None

$ oc get job -l machineconfiguration.openshift.io/machine-os-config=infra

NAME                                            STATUS   COMPLETIONS   DURATION   AGE

build-infra-32ef35dea3e553071277954842edb33a   Failed   0/1           31m        31m


# These are the pods launched by the failed job

$ oc -n openshift-machine-config-operator get pods

NAME                                                          READY   STATUS       RESTARTS        AGE

build-infra-32ef35dea3e553071277954842edb33a-2jg2t            0/1     Init:Error   0               25m

build-infra-32ef35dea3e553071277954842edb33a-bzfcp            0/1     Init:Error   0               29m

build-infra-32ef35dea3e553071277954842edb33a-cndjm            0/1     Init:Error   0               32m

build-infra-32ef35dea3e553071277954842edb33a-lqlk9            0/1     Init:Error   0               22m
```

The failed pod's logs can be examined to determine the reason.

The build pods have 2 containers: image-build and create-digest-configmap:

- The **container image-build** will actually build the image and push it.
- The **container create-digest-configmap** will create an auxiliary configmap with the right digest so that it can be read and MCO can update the MOSB and MOSC resources.

To identify errors in the build process, the image-build container in the build pod should be examined:

```
None

$ oc -n openshift-machine-config-operator logs build-infra-32ef35dea3e553071277954842edb33a-2jg2t -c image-build

...

time="2025-10-17T10:51:32Z" level=debug msg="Running &exec.Cmd{Path:\"/bin/sh\", Args:[]string{\"/bin/sh\", \"-c\",
\"curl --fail -L https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64_wrong -o /usr/bin/yq && chmod
+x /usr/bin/yq\"}, Env:[]string{\"HTTP_PROXY=\", \"HTTPS_PROXY=\", \"NO_PROXY=\",
\"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin\", \"HOSTNAME=0430829320a1\", \"HOME=/root\"},
Dir:\"/\", Stdin:(*os.File)(0xc0001280a0), Stdout:(*os.File)(0xc0001280a8), Stderr:(*os.File)(0xc0001280b0),
ExtraFiles:[]*os.File(nil), SysProcAttr:(*syscall.SysProcAttr)(0xc00017c0c0), Process:(*os.Process)(nil),
ProcessState:(*os.ProcessState)(nil), ctx:context.Context(nil), Err:error(nil), Cancel:(func() error)(nil),
WaitDelay:0, childIOFiles:[]io.Closer(nil), parentIOPipes:[]io.Closer(nil), goroutine:[]func() error(nil),
goroutineErr:(<-chan error)(nil), ctxResult:(<-chan exec.ctxResult)(nil), createdByStack:[]uint8(nil),
lookPathErr:error(nil), cachedLookExtensions:struct { in string; out string }{in:\"\", out:\"\"}} (PATH = \"\")"

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current

                                 Dload  Upload   Total   Spent    Left  Speed

^M  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0^M  0     0    0     0    0     0
0      0 --:--:-- --:--:-- --:--:--     0

^M  0     9    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0

curl: (22) The requested URL returned error: 404
```

```
subprocess exited with status 22

subprocess exited with status 22

time="2025-10-17T10:51:32Z" level=debug msg="Error building at step {Env:[HTTP_PROXY= HTTPS_PROXY= NO_PROXY=
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin] Command:run Args:[curl --fail -L
https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64_wrong -o /usr/bin/yq && chmod +x /usr/bin/yq]
Flags:[] Attrs:map[] Message:RUN curl --fail -L
https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64_wrong -o /usr/bin/yq && chmod +x /usr/bin/yq
Heredocs:[] Original:RUN curl --fail -L https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64_wrong
-o /usr/bin/yq && chmod +x /usr/bin/yq}: exit status 22"

Error: building at STEP "RUN curl --fail -L
https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64_wrong -o /usr/bin/yq && chmod +x /usr/bin/yq":
exit status 22
```

The logs show that curl returned curl: (22) The requested URL returned error: 404 when attempting to reach https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64_wrong. This happens because there is a typo in the URL and the actual URL should be https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64.

Once the error has been detected, editing the MOSC resource and using the correct URL in the Containerfile section will trigger a new MOSB resource that will successfully build the image and apply the config.

Other kinds of errors can be found in the pod, like the ones regarding the lack of permissions to pull or push the images. For example, the pod can be seen reporting that the configured secret doesn't have permissions to push the image:

```
None

$ oc logs build-infra-5e0c7aaf3cf26e8fab9dd111bb336342-czzjb -c image-build

....
```

```
Copying blob sha256:29f46dbdbc11454d191cd70ebbd18aec36bc2afc72757d38f2ad473b6dba1c75

Copying blob sha256:d0a1fe72e3dceadb214f96787144ef31672f2b2a429a3798717d739a55a9b574

Error: pushing image "quay.io/sregidor/sregidor-os:infra-5e0c7aaf3cf26e8fab9dd111bb336342" to
"docker://quay.io/sregidor/sregidor-os:infra-5e0c7aaf3cf26e8fab9dd111bb336342": writing blob: initiating layer upload
to /v2/sregidor/sregidor-os/blobs/uploads/ in quay.io: unauthorized: access to the requested resource is not authorized
```

In this case it was a problem in the build, but if the build process is not failing and nevertheless the build pod fails, the create-digest-configmap container can be examined to see if there was any problem creating the configmap with the digest info.

## Auxiliary Resources

In order to build the image, MCO uses several auxiliary resources that are temporarily stored in the MCO namespace. These resources are only present during the build process. However, if the build fails, they remain available for debugging purposes.

Those auxiliary resources are mounted in the build pod, so that it can use them.

They can be found like this:

```
None

$ oc get cm -n openshift-machine-config-operator --sort-by metadata.creationTimestamp

...
```

```
additionaltrustbundle-infra-32ef35dea3e553071277954842edb33a   1      47m
etc-policy-infra-32ef35dea3e553071277954842edb33a              1      47m
mc-infra-32ef35dea3e553071277954842edb33a                      1      47m
containerfile-infra-32ef35dea3e553071277954842edb33a           1      47m
etc-registries-infra-32ef35dea3e553071277954842edb33a          1      47m

$ oc get secret -n openshift-machine-config-operator --sort-by metadata.creationTimestamp
NAME                                               TYPE                                  DATA   AGE
...
global-pull-secret-copy                            kubernetes.io/dockerconfigjson        1      48m
final-infra-32ef35dea3e553071277954842edb33a   kubernetes.io/dockerconfigjson        1      48m
base-infra-32ef35dea3e553071277954842edb33a    kubernetes.io/dockerconfigjson        1      48m
```

These resources can be described as follows:

- The additional trust bundle configmap: additionaltrustbundle-infra-32ef35dea3e553071277954842edb33a will store the necessary bundles to use RHEL packages in the Containerfile. It should be taken from a copy of the etc-pki-entitlement secret in the openshift-config-managed namespace. If the build is having problems using RHEL packages, this resource should be checked to ensure it stores the correct bundles.

- The current machine config configmap: mc-infra-32ef35dea3e553071277954842edb33a will store the MachineConfig resource that needs to be applied to the nodes in this MachineConfigPool. The next command can be executed to see its content:

```
None

$ oc get cm -n openshift-machine-config-operator mc-infra-32ef35dea3e553071277954842edb33a -o
jsonpath='{.data.machineconfig\.json\.gz}' | base64 -d | gunzip | jq | less
```

- The container file configmap containerfile-infra-32ef35dea3e553071277954842edb33a will store the full container file used to build the image. The next command can be executed to see its content:

```
None

$ oc get cm -oyaml containerfile-infra-32ef35dea3e553071277954842edb33a -o jsonpath='{.data.Containerfile}'
```

- The etc registries and policies configmaps (etc-registries-infra-32ef35dea3e553071277954842edb33a and etc-policy-infra-32ef35dea3e553071277954842edb33a) contain the registry configuration (registries.conf) and the policies (policy.json) used in the cluster, so that they can be used in the build process as well. Those resources can be checked if there are problems with the container registries.

```
None

$ oc -n openshift-machine-config-operator get cm -oyaml etc-registries-infra-32ef35dea3e553071277954842edb33a

apiVersion: v1

data:

  registries.conf: |
    unqualified-search-registries = ['registry.access.r.com', 'docker.io']
...
```

- The secrets are the ones configured in the MOSC resource and contain the credentials to pull/push the necessary images.

As mentioned above, if the MOSB fails, these auxiliary resources are not removed so that they can be used for further debugging.

## Stage 3: Image Applied to Nodes

After a successful build, the MCO will roll out the new image updating the machineconfiguration.openshift.io/desiredImage annotation in the nodes and the MachineConfigDaemon pods will apply the image.

What to watch for:

- Pool update status: The MachineConfigPool should show Updating=True as nodes begin updating
- Image pull failures: Nodes may fail to download the image if currentImagePullSecret is incorrect
- Network connectivity: Nodes must be able to reach the registry where the image is stored
- Node degradation: Nodes stuck in degraded state due to failed updates should be checked
- Reboot issues: Nodes should successfully reboot into the new OS image
- Stalled updates: If the pool remains in Updating state too long, individual node statuses should be investigated

In this final stage, issues typically relate to nodes' ability to access and apply the layered image.

## Success

The MCP should report an updated status:

```
None

$ oc get mcp infra

NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED   MACHINECOUNT
READYMACHINECOUNT   UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE

infra   rendered-infra-f47b79e31d1479ed9dfc5662e1d1ae74   True      False      False      3              3
3                     0                       61m
```

The proper application of the image on the nodes can be verified:

```
None

$ oc debug -q node/ip-10-0-10-154.us-east-2.compute.internal -- chroot /host rpm-ostree status

State: idle

Deployments:

ostree-unverified-registry:quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f557
02c9ef13

        Digest: sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f55702c9ef13
```

```
      Version: 9.6.20251013-1 (2025-10-17T12:09:08Z)


$ oc debug -q node/ip-10-0-10-154.us-east-2.compute.internal -- chroot /host which yq /usr/bin/yq


$ oc debug -q node/ip-10-0-10-154.us-east-2.compute.internal -- chroot /host yq -h

yq is a portable command-line data file processor (https://github.com/mikefarah/yq/)

See https://mikefarah.gitbook.io/yq/ for detailed documentation and examples.


Usage:

  yq [flags]

  yq [command]

...
```

## Error

At this point the debugging process is very similar to the one followed when applying a new MachineConfig. Focus should be placed on checking the MachineConfigPool status, the information in the MachineConfigNodes resources and the logs of the machine-config-daemon pods.

In case of error, the MCP will show as degraded:

```
None

$ oc get mcp infra

NAME    CONFIG                                            UPDATED   UPDATING   DEGRADED   MACHINECOUNT
READYMACHINECOUNT   UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE

infra   rendered-infra-6208c0db8119cfe2c9c4e42099617a43   False     False      True       3              0
0                     1                       3h51m



$ oc get mcp infra -oyaml

...

  - lastTransitionTime: "2025-10-17T12:23:48Z"

    message: 'Node ip-10-0-75-69.us-east-2.compute.internal is reporting: "Node
ip-10-0-75-69.us-east-2.compute.internal

      upgrade failure. Failed to update OS to
quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f55702c9ef13

      after retries: timed out waiting for the condition", Node ip-10-0-75-69.us-east-2.compute.internal

      is reporting: "Failed to update OS to
quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f55702c9ef13

      after retries: timed out waiting for the condition"'

    reason: 1 nodes are reporting degraded status on sync

    status: "True"

    type: NodeDegraded
```

And the detailed information can be found in the machine-config-daemon pod logs:

```
None

$ oc logs -n openshift-machine-config-operator $(oc get pods -n openshift-machine-config-operator -l
"k8s-app=machine-config-daemon" --field-selector "spec.nodeName=ip-10-0-75-69.us-east-2.compute.internal" -o
jsonpath="{.items[0].metadata.name}") -c machine-config-daemon

...

I1017 12:26:52.042570    2750 update.go:2546] Updating OS to layered image
"quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f55702c9ef13"

I1017 12:26:52.042590    2750 image_manager_helper.go:92] Running captured: rpm-ostree --version

I1017 12:26:52.055729    2750 image_manager_helper.go:194] Linking rpm-ostree authfile to
/etc/mco/internal-registry-pull-secret.json

I1017 12:26:52.055759    2750 rpm-ostree.go:183] Executing rebase to
quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f55702c9ef13

I1017 12:26:52.055764    2750 update.go:2630] Running: rpm-ostree rebase --experimental
ostree-unverified-registry:quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f557
02c9ef13

Pulling manifest:
ostree-unverified-registry:quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f557
02c9ef13

W1017 12:26:52.427068    2750 update.go:2591] Failed to update OS to
quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f55702c9ef13 (will retry):
error running rpm-ostree rebase --experimental
ostree-unverified-registry:quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f557
02c9ef13: error: Creating importer: failed to invoke method OpenImage: failed to invoke method OpenImage: reading
manifest sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f55702c9ef13 in quay.io/sregidor/sregidor-os:
manifest unknown
```

The information reported by the MachineConfigNode resources should also be checked. This is especially important because in future versions of OpenShift more information regarding the OCL process will be added to those resources in order to make the debugging process easier.

```
None

$ oc get machineconfignode -o wide

NAME                                        POOLNAME   DESIREDCONFIG                                         CURRENTCONFIG
UPDATED   AGE     UPDATEPREPARED   UPDATEEXECUTED   UPDATEPOSTACTIONCOMPLETE   UPDATECOMPLETE   RESUMED
UPDATEDFILESANDOS   CORDONEDNODE   DRAINEDNODE   REBOOTEDNODE   UNCORDONEDNODE

ip-10-0-10-154.us-east-2.compute.internal   infra      rendered-infra-6208c0db8119cfe2c9c4e42099617a43
rendered-infra-6208c0db8119cfe2c9c4e42099617a43    True      4h34m   False            False            False
False            False     False               False          False         False          False

ip-10-0-22-152.us-east-2.compute.internal   master     rendered-master-93a022e91aa2bf815e4efed220ac97ea
rendered-master-93a022e91aa2bf815e4efed220ac97ea   True      4h44m   False            False            False
False            False     False               False          False         False          False

ip-10-0-41-78.us-east-2.compute.internal    infra      rendered-infra-6208c0db8119cfe2c9c4e42099617a43
rendered-infra-6208c0db8119cfe2c9c4e42099617a43    True      4h34m   False            False            False
False            False     False               False          False         False          False

ip-10-0-60-66.us-east-2.compute.internal    master     rendered-master-93a022e91aa2bf815e4efed220ac97ea
rendered-master-93a022e91aa2bf815e4efed220ac97ea   True      4h44m   False            False            False
False            False     False               False          False         False          False

ip-10-0-65-176.us-east-2.compute.internal   master     rendered-master-93a022e91aa2bf815e4efed220ac97ea
rendered-master-93a022e91aa2bf815e4efed220ac97ea   True      4h44m   False            False            False
False            False     False               False          False         False          False

ip-10-0-75-69.us-east-2.compute.internal    infra      rendered-infra-6208c0db8119cfe2c9c4e42099617a43
rendered-worker-6208c0db8119cfe2c9c4e42099617a43   False     4h40m   True             Unknown          False
False            False     Unknown             True           True          False          False
```

```
$ oc get machineconfignode ip-10-0-75-69.us-east-2.compute.internal -oyaml

...

  - lastTransitionTime: "2025-10-17T12:22:13Z"

    message: 'Node ip-10-0-75-69.us-east-2.compute.internal upgrade failure. Failed

      to update OS to
quay.io/sregidor/sregidor-os@sha256:8761d4273f3213f2f9c9b4aa9dbe33aa758f17d691f0f53d2b20f55702c9ef13

      after retries: timed out waiting for the condition'

    reason: NodeDegraded

    status: "True"
```

## Conclusion

Debugging On Cluster Layering doesn't have to be a black box operation. By understanding the three distinct stages (MachineOSConfig validation, MachineOSBuild execution, and image deployment to nodes), failures can be systematically narrowed down to identify where they occur and their root causes. The key is knowing where to look: the machine-config-operator pod logs for MOSC issues, the build job pod logs for image build failures, and the machine-config-daemon pod logs for node-level problems.

OCL failures usually happen during the build stage, often caused by pull secret authentication issues, Containerfile errors, or registry permission problems. The debugging techniques in this guide enable effective troubleshooting and successful deployment of customized node images.