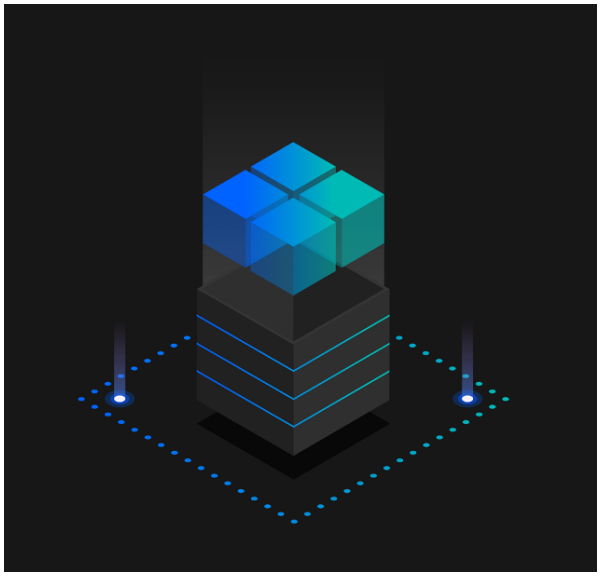
الشركة السعودية للكهرباء
Saudi Electricity Company



SEC CP4I v 2020.4.1
Production ~ Installation

Hakam Abdelqader (hakam.abdelqader@ibm.com)

Table of Contents

Openshift Production Configuration

In this section we want to segregate the infra and storage nodes so workloads won't schedule on those nodes. We will be labeling, adding taints, schedule relevant pods to schedule on those node and removing any workloads that should be used up by worker nodes.

Labeling nodes

1. Label **storage** nodes with `node-role.kubernetes.io/storage=''`

```
$ oc label node r.h-smoc-cloud-ocs3.ocpcluster.secsmoc.local node-role.kubernetes.io/storage=''
# node/rh-smoc-cloud-ocs3.ocpcluster.secsmoc.local labeled
$ oc label node rh-smoc-cloud-ocs2.ocpcluster.secsmoc.local node-role.kubernetes.io/storage=''
# node/rh-smoc-cloud-ocs2.ocpcluster.secsmoc.local labeled
$ oc label node rh-smoc-cloud-ocs1.ocpcluster.secsmoc.local node-role.kubernetes.io/storage=''
# node/rh-smoc-cloud-ocs1.ocpcluster.secsmoc.local labeled
```

2. Label **infra** nodes with `node-role.kubernetes.io/infra=''`

```
$ oc label node rh-smoc-cloud-infra1.ocpcluster.secsmoc.local node-role.kubernetes.io/infra=''
# node/rh-smoc-cloud-infra1.ocpcluster.secsmoc.local labeled
$ oc label node rh-smoc-cloud-infra2.ocpcluster.secsmoc.local node-role.kubernetes.io/infra=''
# node/rh-smoc-cloud-infra2.ocpcluster.secsmoc.local labeled
$ oc label node rh-smoc-cloud-infra3.ocpcluster.secsmoc.local node-role.kubernetes.io/infra=''
# node/rh-smoc-cloud-infra3.ocpcluster.secsmoc.local labeled
```

Taint Nodes

Adding a `NoSchedule` OpenShift Container Storage taint is also required so that the `infra` node will only schedule OpenShift Container Storage resources and repel any other non-OpenShift Container Storage workloads.

Storage

```
$ oc adm taint node <node> node.ocs.openshift.io/storage="true":NoSchedule
```

```
 $ oc adm taint node rh-smoc-cloud-ocs1.ocpcluster.secsmoc.local node.ocs.openshift.io/storage="true":NoSchedule
# node/rh-smoc-cloud-ocs1.ocpcluster.secsmoc.local tainted
 $ oc adm taint node rh-smoc-cloud-ocs2.ocpcluster.secsmoc.local node.ocs.openshift.io/storage="true":NoSchedule
# node/rh-smoc-cloud-ocs2.ocpcluster.secsmoc.local tainted
 $ oc adm taint node rh-smoc-cloud-ocs3.ocpcluster.secsmoc.local node.ocs.openshift.io/storage="true":NoSchedule
# node/rh-smoc-cloud-ocs3.ocpcluster.secsmoc.local tainted

```

Infra

```
$ oc adm taint node <node> node-role.kubernetes.io/infra=''
```

```
 $ oc adm taint node rh-smoc-cloud-infra1.ocpcluster.secsmoc.local node-role.kubernetes.io/infra='':NoSchedule
# node/rh-smoc-cloud-infra1.ocpcluster.secsmoc.local tainted
 $ oc adm taint node rh-smoc-cloud-infra2.ocpcluster.secsmoc.local node-role.kubernetes.io/infra='':NoSchedule
# node/rh-smoc-cloud-infra2.ocpcluster.secsmoc.local tainted
 $ oc adm taint node rh-smoc-cloud-infra3.ocpcluster.secsmoc.local node-role.kubernetes.io/infra='':NoSchedule
# node/rh-smoc-cloud-infra3.ocpcluster.secsmoc.local tainted

```

Troubleshoot `Pending` pods

Problem

When you have a pod stuck in `Pending` state check the events for the following message:

```
0/23 nodes are available: 1 node(s) had taint {node-role.kubernetes.io/infra:
}, that the pod didn't tolerate, 22 node(s) didn't match node selector.
```

Resolution

```
$ oc edit pod <pod-name> -n <namespace>
```

Add the following lines to the toleration section depending on what taint was described in the events:

```
spec:
  tolerations:
    ... #omitted (other tolerations configured)
  # add the lines below (toleration section should already exists)
    - key: node-role.kubernetes.io/infra # add key matching error in events
      operator: Exists
      effect: NoSchedule
```

Machine Config Pool and Machine Config Configuration

Create two yaml files to create **MachineConfigPools**

Infra

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: infra
spec:
  machineConfigSelector:
    matchExpressions:
      - {key: machineconfiguration.openshift.io/role, operator: In, values:
[worker,infra]}
  maxUnavailable: null
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/infra: ""
  paused: false
```

Storage

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: storage
spec:
  machineConfigSelector:
    matchExpressions:
      - {key: machineconfiguration.openshift.io/role, operator: In, values:
[storage]}
```

```
  maxUnavailable: 0
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/storage: ""
  paused: false

```

```
$ oc create -f mcp-infra.yaml -f mcp-storage.yaml
```

To check the status of machineconfigspool. **MachineCount** should equal **ReadyMachineCount**

```
$ oc get mcp # machineconfigpool
NAME      CONFIG                                             UPDATED
UPDATING   DEGRADED   MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
infra     rendered-infra-75dafd4d626a5109120a08a059e8158e     True      False
     False      3              3                   3                    0
                  14d
master    rendered-master-0cd9beec62ccc05897c12aa052328e83    True      False
     False      3              3                   3                    0
                  47d
storage   rendered-storage-75dafd4d626a5109120a08a059e8158e   True      False
     False      3              3                   3                    0
                  14d
worker    rendered-worker-75dafd4d626a5109120a08a059e8158e    True      False
     False      14             14                  14                   0
                  47d


$ oc get mc # machineconfig
```

Final output

```
   $  oc get nodes -l node-role.kubernetes.io/infra
   NAME                          STATUS   ROLES     AGE   VERSION

   rh-smoc-cloud-infra1.ocp...   Ready    infra     25d   v1.19.0+f173eb4
   rh-smoc-cloud-infra2.ocp...   Ready    infra     25d   v1.19.0+f173eb4
   rh-smoc-cloud-infra3.ocp...   Ready    infra     25d   v1.19.0+f173eb4

   $  oc get nodes -l node-role.kubernetes.io/storage
   rh-smoc-cloud-ocs1.ocp...     Ready    storage   25d   v1.19.0+f173eb4
   rh-smoc-cloud-ocs2.ocp...     Ready    storage   25d   v1.19.0+f173eb4
   rh-smoc-cloud-ocs3.ocp...     Ready    storage   25d   v1.19.0+f173eb4

```

Status of MCP

Troubleshoot any problem with Nodes in `Not Ready` state

```
$ oc get node -o custom-
columns=STATE:metadata.annotations.machineconfiguration\\.openshift\\.io/state
,DESIRED:metadata.annotations.machineconfiguration\\.openshift\\.io/desiredCon
fig,CURRENT:metadata.annotations.machineconfiguration\\.openshift\\.io/current
Config,REASON:metadata.annotations.machineconfiguration\\.openshift\\.io/reaso
n
```

```
$ oc get nodes -o jsonpath="{range .items[*]}{@.metadata.name}{'\t\t'}
{@.metadata.annotations.machineconfiguration\.openshift\.io/state}{'\n'}{end}"
| grep -v Done
```

Reference:

1. https://access.redhat.com/solutions/5055931

Drain nodes of all the workloads

```
$ oc cordon <node-name> # ScheduleDisabled

```

```
$ oc adm drain <node-name> --force --delete-local-data --ignore-daemonsets

```

```
evicting pod openshift-storage/rook-ceph-mon-a-6f7c6dbdb4-m5fnc
evicting pod openshift-storage/rook-ceph-osd-0-7f78c9dbbf-29vk9
evicting pod openshift-storage/rook-ceph-osd-4-695b975576-7srpb
error when evicting pod "rook-ceph-mon-a-6f7c6dbdb4-m5fnc" (will retry after
5s): Cannot evict pod as it would violate the pod's disruption budget.
error when evicting pod "rook-ceph-osd-0-7f78c9dbbf-29vk9" (will retry after
5s): Cannot evict pod as it would violate the pod's disruption budget.
error when evicting pod "rook-ceph-osd-4-695b975576-7srpb" (will retry after
5s): Cannot evict pod as it would violate the pod's disruption budget.
evicting pod openshift-storage/rook-ceph-osd-4-695b975576-7srpb
evicting pod openshift-storage/rook-ceph-mon-a-6f7c6dbdb4-m5fnc
evicting pod openshift-storage/rook-ceph-osd-0-7f78c9dbbf-29vk9
error when evicting pod "rook-ceph-mon-a-6f7c6dbdb4-m5fnc" (will retry after
5s): Cannot evict pod as it would violate the pod's disruption budget.
error when evicting pod "rook-ceph-osd-0-7f78c9dbbf-29vk9" (will retry after
5s): Cannot evict pod as it would violate the pod's disruption budget.
error when evicting pod "rook-ceph-osd-4-695b975576-7srpb" (will retry after
5s): Cannot evict pod as it would violate the pod's disruption budget.

```

```
$ oc uncordon <node-name> # Scheduleable


```

```
$ oc scale deploy <pod-name> --replicas=0 -n openshift-storage
```

References

1. https://access.redhat.com/solutions/5034771
2. https://github.com/openshift/machine-config-operator/blob/master/docs/custom-pools.md
3. https://docs.openshift.com/container-platform/4.6/machine_management/creating-infrastructure-machinesets.html#moving-resources-to-infrastructure-machinesets
4. https://www.redhat.com/en/blog/openshift-container-platform-4-how-does-machine-config-pool-work

Troubleshoot MachineConfigPool (MCP stuck in updating status)

MCP stuck in updating status when checked from ocp CLI:

```
[ocpuser@ocphost ~]$ oc get mcp
NAME     CONFIG              UPDATED   UPDATING   DEGRADED   MACHINECOUNT
READYMACHINECOUNT   UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE
master   rendered-master-xxxx   True      False      False      3
 3                   3                     0                    300d
worker   rendered-worker-xxxx   False     True       False      21
1                    1                     0                    300d
```

Trying to `rsh` into pod return below error:

```
[ocpuser@ocpnode ~]$ oc rsh <pod_name> -n <project_name>
error: unable to upgrade connection: Unauthorized
```

Resolution

Step 1 Find out the node where the MCP is stuck with below command:

```
$ oc get nodes -o jsonpath="{range .items[*]}{@.metadata.name}{'\t\t'}
{@.metadata.annotations.machineconfiguration\.openshift\.io/state}{'\n'}{end}"
| grep -v Done
```

Step 2 Login into the node using SSH and see the logs from `machine-config-daemon` on the node:

```
$ crictl ps | grep -i machine

$ crictl logs <container_id>
    =>Container ID from above command output
```

Below logs observed from `machine-config-daemon` container:

```
I0218 14:36:51.075295 2225251 update.go:99] error when evicting pod "mypdb-1"
(will retry after 5s): Cannot evict pod as it would violate the pod's
disruption budget.
I0218 14:36:56.081506 2225251 update.go:99] error when evicting pod "mypdb-1"
(will retry after 5s): Cannot evict pod as it would violate the pod's
disruption budget.
```

Step 3 Get the `PodDisruptionBudget` YAML definition and observe why it's not able to evict the pod.
Here, PDB `maxUnavailable` is set to 1 and none of the pod is healthy, so it is not able to evict the pod.

```
...
...
spec:
  maxUnavailable: 1
  selector:
    matchLabels:
      app.kubernetes.io/instance: mypdb
      app.kubernetes.io/name: mypdb
      component: server
status:
  currentHealthy: 0
  desiredHealthy: 2
  disruptionsAllowed: 0
  expectedPods: 3
  observedGeneration: 1
```

Step 4 Patch `maxUnavailable` parameter to an appropriate value:

```
   $ oc patch pdb <pdb_name> -n <project_name> --type=merge -p '{"spec":
{"maxUnavailable":3}}'
```

Step 5 Observe if the pod has been rescheduled on another node. If not, delete the pod forcefully. Now the pod should be running on another node.
Step 6 Wait for 15-20 minute, if `MachineConfigPool` isn't updating delete `machine-config-daemon` pod for the node (where MCP is stuck). Check `Step 1` for node name.
Step 7 After MCP has been applied successfully on all the desired nodes, revert the changes performed in `Step 4`.

Moving Workloads to infra

Ingress Controller

We need to edit it and modify the `spec` section to add a `NodeSelector` stanza in the following format

```
$ oc edit ingresscontroller default -n openshift-ingress-operator -o yaml
```

Replace `spec: {}` with the following - that references the `infra` label to the `spec` section, as shown:

```
spec:
    nodePlacement:
      nodeSelector:
        matchLabels:
          node-role.kubernetes.io/infra: ""
```

Confirm the correct nodes are in the right place by running

```
$ oc get pod -n openshift-ingress -o wide
```

Add tolerations for the pod configurations you want to schedule on the infra node, like router, registry, and monitoring workloads. Add the following code to the `deployment` object specification:

Router

```
$ oc get deployments -n openshift-ingress

$ oc edit deployment router-default  -n openshift-ingress
```

Place code under `spec.template.spec..tolerations`

```
spec:
  template:
    spec:
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/infra
          operator: Exists
```

Default Registry

View the `config/instance` object:

```
$ oc get configs.imageregistry.operator.openshift.io cluster -o yaml
```

To move the default registry, it is a similar pattern - first by editing the `config/instance` object

```
oc edit config/cluster
```

Add the following in the `spec`

```
nodeSelector:
    node-role.kubernetes.io/infra: ""
```

Verify the registry pod has been moved to the infrastructure node.

1. Run the following command to identify the node where the registry pod is located:

   ```
   $ oc get pods -o wide -n openshift-image-registry
   ```

2. Confirm the node has the label you specified:

   ```
   $ oc describe node <node_name>
   ```

   Review the command output and confirm that `node-role.kubernetes.io/infra` is in the `LABELS` list.

Monitoring

To move monitoring, we need to create a ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |+
    alertmanagerMain:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusK8s:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusOperator:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    grafana:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    k8sPrometheusAdapter:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    kubeStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    telemeterClient:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    openshiftStateMetrics:
```

```
30         nodeSelector:
31           node-role.kubernetes.io/infra: ""
```

Applying the above can be done ising `oc create`

```
$ oc create -f cluster-monitoring-configmap.yaml
```

After a few minutes, you can check that the pods are in the correct place with the following command

```
$ oc get pod -n openshift-monitoring -o wide
```

Apply the new config map:

```
$ oc create -f cluster-monitoring-configmap.yaml
```

Watch the monitoring pods move to the new machines:

```
$ watch 'oc get pod -n openshift-monitoring -o wide'
```

If a component has not moved to the `infra` node, delete the pod with this component:

```
$ oc delete pod -n openshift-monitoring <pod>
```

The component from the deleted pod is re-created on the `infra` node.

References

1. https://docs.openshift.com/container-platform/4.6/machine_management/creating-infrastructure-machinesets.html#machineset-creating_creating-infrastructure-machinesets
2. https://docs.openshift.com/container-platform/4.6/storage/persistent_storage/persistent-storage-ocs.html

Prerequisites (Mirroring CP4I operators and images)

Introduction

You can make the operators for IBM Cloud Pak for Integration available to a cluster that has no internet connectivity by using a *bastion host*, which is a machine that has connectivity both to the internet and to the cluster in the restricted environment. This process allows the necessary images to be mirrored directly to an internal image registry for use by the cluster.

- Install an OpenShift 4.6 cluster (see [operating-environment]).
- Prepare a Docker registry and make it available. For more information, see Prepare a Docker registry.
- Configure a bastion host. For more information, see Prepare a bastion host.
- `cloudctl` version 3.6.1-2002 or later is required.

Prepare a Docker registry

A local Docker registry is used to store all images in your restricted environment. You must create such a registry and ensure that it meets the following requirements:

- Supports Docker Manifest V2, Schema 2.
- Is accessible from both the bastion host and your OpenShift cluster nodes.
- Has the username and password of a user who can write to the target registry from the bastion host.
- Has the username and password of a user who can read from the target registry that is on the OpenShift cluster nodes.
- Allows path separators in the image name.

An example of a simple registry is included in Creating a mirror registry for installation in a restricted network in the OpenShift documentation.

The internal Red Hat OpenShift registry is not compliant with Docker Manifest V2, Schema 2, so it is not suitable for use as a private registry for restricted environments.

Verify that you:

- Have the credentials of a user who can write and create repositories. The bastion host uses these credentials.
- Have the credentials of a user who can read all repositories. The OpenShift cluster uses these credentials.

Prepare a bastion host

Prepare a bastion host that can access the OpenShift cluster, the local Docker registry, and the internet.

- The bastion host must be on a Linux x86_64 platform, or any operating system that the IBM Cloud Pak CLI and the OpenShift CLI supports.
- The bastion host `locale` must be set to English.

Complete these steps on your bastion node:

1. Install OpenSSL version 1.11.1 or higher.
2. Install Docker or Podman on the bastion node.
    - For example, to install Docker on a Red Hat machine, run these commands:

      ```
      yum check-update
      yum install docker
      ```

    - To install Podman, see Podman Installation Instructions.
3. Install the latest version of the IBM Cloud Pak CLI, using the binary file that is applicable for your platform. The minimum supported version of the binary file is 3.4.4. For a listing of available binaries, see cloud-pak-cli.
    1. Download the binary file.

       ```
       wget https://github.com/IBM/cloud-pak-cli/releases/latest/download/cloudctl-linux-amd64.tar.gz
       ```

    2. Run the following commands to modify and move the file.

       ```
       chmod 755 cloudctl-linux-amd64.tar.gz
       mv cloudctl-linux-amd64.tar.gz /usr/local/bin/cloudctl
       ```

    3. Confirm that `cloudctl` is installed:

       ```
       cloudctl --help
       ```

       The cloudctl usage is displayed.
4. Install the `oc` OpenShift CLI tool. For more information, see Getting started with the CLI in the Red Hat OpenShift documentation.
5. Install the `skopeo` CLI version `1.0.0` or higher. For more information, see Installing skopeo from packages.
6. Create a directory that serves as the offline store.

   Following is an example directory. This example is used in the subsequent steps.

   ```
   mkdir $HOME/offline
   ```

> This offline store must be persistent to avoid transferring data more than once. The persistence also helps to run the mirroring process multiple times or on a schedule.

Create environment variables for the installer and image inventory

Create the following environment variables with the installer image name and the image inventory.

```
export CASE_ARCHIVE=ibm-cp-integration-2.1.0.tgz
export CASE_INVENTORY_SETUP=operator
```

Using this CASE archive will mirror the whole Cloud Pak for Integration. **If does not work mirror individual components**

> Event Streams is deprecated and is not included in the CASE archive for Cloud Pak for Integration. Event Streams is still available as an individual component as described in the following section.

Mirroring individual components

To mirror part of the Cloud Pak, use the CASE archive and inventory item for an individual component, and repeat the process for each component that you need to be available in your restricted environment.

CASE files for IBM components can be found in the IBM CASE repository.

The CP4I component CASE files available for individual mirroring are:

IBM Cloud Pak for Integration Platform Navigator

```
  export CASE_ARCHIVE=ibm-integration-platform-navigator-1.1.0.tgz
  export CASE_INVENTORY_SETUP=platformNavigatorOperator
```

IBM API Connect

```
export CASE_ARCHIVE=ibm-apiconnect-2.0.1.tgz
export CASE_INVENTORY_SETUP=apiconnectOperatorSetup
```

IBM App Connect

```
  export CASE_ARCHIVE=ibm-appconnect-1.1.0.tgz
  export CASE_INVENTORY_SETUP=ibmAppconnect
```

IBM MQ

```
export CASE_ARCHIVE=ibm-mq-1.3.0.tgz
export CASE_INVENTORY_SETUP=ibmMQOperator
```

Download the installer and image inventory

Download the installer and image inventory to the bastion host.

This step downloads the selected CASE file and its dependencies to the local machine.
It also produces CSV files listing the images and helm charts included in each CASE file. The CP4I components do not include any helm charts.

```
$ cloudctl case save \
    --case https://github.com/IBM/cloud-pak/raw/master/repo/case/${CASE_ARCHIVE} \
    --outputdir $HOME/offline/

```

Log in to OpenShift cluster as a cluster administrator

Log in to the OpenShift cluster. For example:

```
$ oc login <cluster host:port> --username=<cluster admin user> --password=
<cluster admin password>
```

Create a Kubernetes namespace

Create an environment variable with a namespace to install, then create the namespace. For example:

```
export NAMESPACE=cp4i
oc create namespace ${NAMESPACE}

```

Mirror the images and configure the cluster

Complete these steps to mirror the images and configure your cluster:

1. Store authentication credentials for the IBM Entitled Registry.

   See [entitlement-key] for how to obtain your entitlement key.

   After obtaining your entitlement key, run the following command to configure credentials for the IBM Entitled Registry:

   ```
   cloudctl case launch \
       --case $HOME/offline/${CASE_ARCHIVE} \
       --inventory ${CASE_INVENTORY_SETUP} \
       --action configure-creds-airgap \
       --namespace ${NAMESPACE} \
       --args "--registry cp.icr.io --user cp --pass <entitlement key> --inputDir
   $HOME/offline"
   ```

   This command stores and caches the registry credentials in a file on your file system in the `$HOME/.airgap/secrets` location.

2. Create environment variables with the local Docker registry connection information.

   ```

   export LOCAL_DOCKER_REGISTRY_HOST=rh-smoc-cloud-inst.secsmoc.local
   export LOCAL_DOCKER_REGISTRY_PORT=5000
   export
   LOCAL_DOCKER_REGISTRY=$LOCAL_DOCKER_REGISTRY_HOST:$LOCAL_DOCKER_REGISTRY_PORT
   export LOCAL_DOCKER_USER=admin
   export LOCAL_DOCKER_PASSWORD=admin123

   ```

3. Configure a global image pull secret and `ImageContentSourcePolicy`.

   To enable your disconnected cluster to access images from your private registry, it must be configured to use your private registry as a mirror of the images hosted in the online registries, and to be able to access those images.

   This step configures an `ImageContentSourcePolicy` for the images listed in the component CASEs. See Configuring image registry repository mirroring in the Red Hat OpenShift documentation for more details.

   This step also configures the global cluster pull secret for the cluster to allow it to access the private registry. See Adding the registry to your pull secret in the Red Hat OpenShift documentation for more details.

   In OpenShift version 4.6, this step performs a rolling restart of all cluster nodes. The cluster resources might be unavailable until the new `ImageContentSourcePolicy` and global cluster pull secret is applied.

```
cloudctl case launch \
    --case $HOME/offline/${CASE_ARCHIVE} \
    --inventory ${CASE_INVENTORY_SETUP} \
    --action configure-cluster-airgap \
    --namespace ${NAMESPACE} \
    --args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --
pass ${LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

4. Verify that the ImageContentSourcePolicy resource is created.

```
oc get ImageContentSourcePolicy
```

5. Verify your cluster node status.

```
oc get nodes
```

After the `ImageContentsourcePolicy` and global image pull secret are applied, you might see the node status as `Ready`, `Scheduling`, or `Disabled`. Wait until all the nodes show a `Ready` status.

6. Configure an authentication secret for the local Docker registry.

This step needs to be done only one time.

```
cloudctl case launch \
    --case $HOME/offline/${CASE_ARCHIVE} \
    --inventory ${CASE_INVENTORY_SETUP} \
    --action configure-creds-airgap \
    --namespace ${NAMESPACE} \
    --args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --
pass ${LOCAL_DOCKER_PASSWORD}"
```

The command stores and caches the registry credentials in a file on your file system in the `$HOME/.airgap/secrets` location.

7. Mirror the images to the local registry.

This command calls the `oc image mirror` command to mirror images from the online registry to the private registry:

```
cloudctl case launch \
    --case $HOME/offline/${CASE_ARCHIVE} \
    --inventory ${CASE_INVENTORY_SETUP} \
    --action mirror-images \
    --namespace ${NAMESPACE} \
    --args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --
pass ${LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

If you are using an insecure registry, you must also add the local registry to the `insecureRegistries` list for your cluster:

```
oc patch image.config.openshift.io/cluster --type=merge \
   -p '{"spec":{"registrySources":{"insecureRegistries":
["'${LOCAL_DOCKER_REGISTRY}'"]}}}'
```

Create the CatalogSource

CP4I can be installed by adding the `CatalogSource` for the mirrored operators to your cluster and using OLM to install the operators.

1. Create a catalog source.

This command adds the `CatalogSource` for the components to your cluster, so the cluster can access them from the private registry:

```
cloudctl case launch \
    --case $HOME/offline/${CASE_ARCHIVE} \
    --inventory ${CASE_INVENTORY_SETUP} \
    --action install-catalog \
    --namespace ${NAMESPACE} \
    --args "--registry ${LOCAL_DOCKER_REGISTRY} --inputDir $HOME/offline --recursive"
```

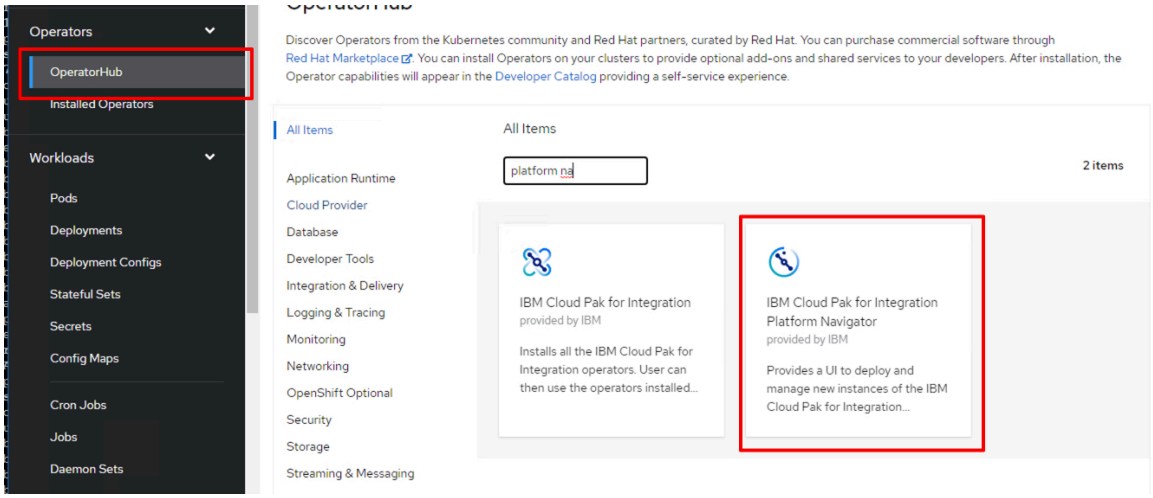
2. Verify that the `CatalogSource` for the Cloud Pak installer operator is created.

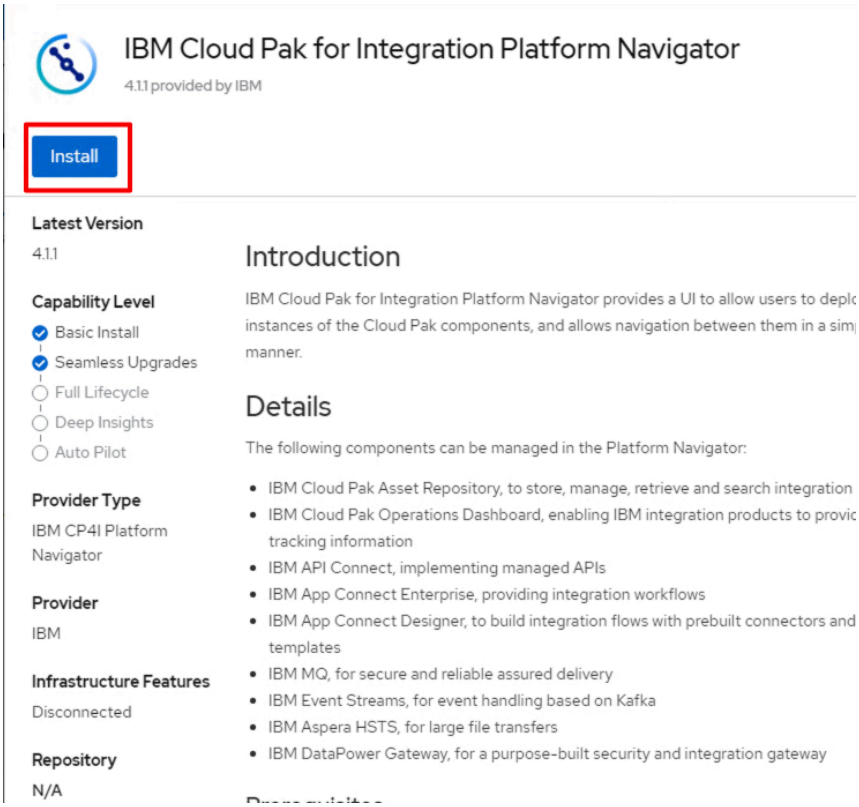
```
oc get pods -n openshift-marketplace
oc get catalogsource -n openshift-marketplace
```

Cloud Pak for Integration Platform Navigator

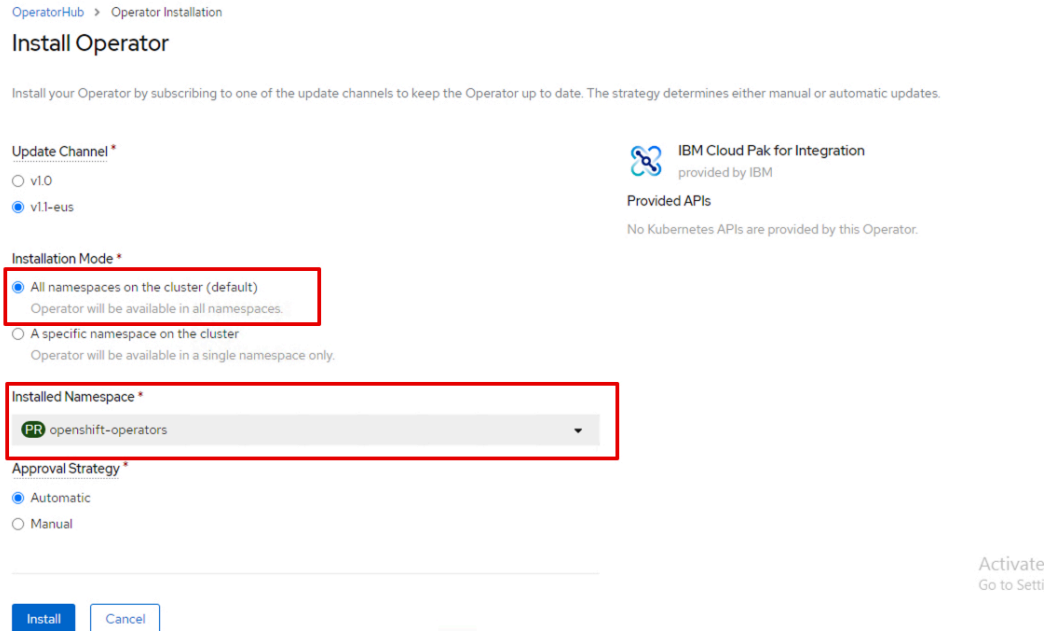
Now that we have mirrored the images we can open **Web console** to install all the operators

1. Install **Platform Navigator Operator** from **OperatorHub**

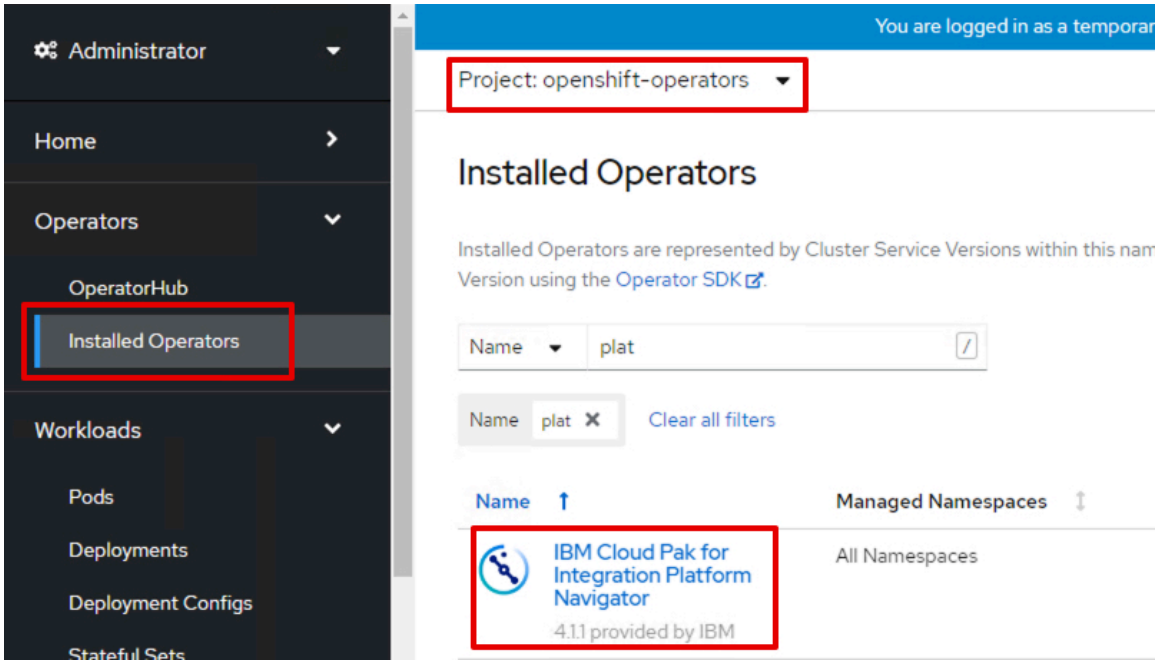




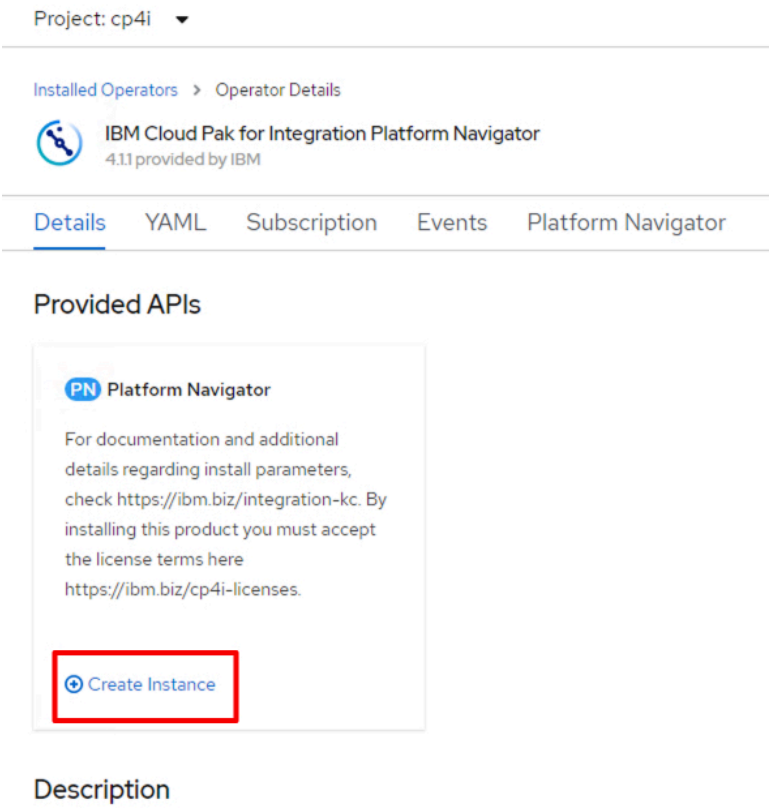
When installing operator select **All namespaces on the cluster**



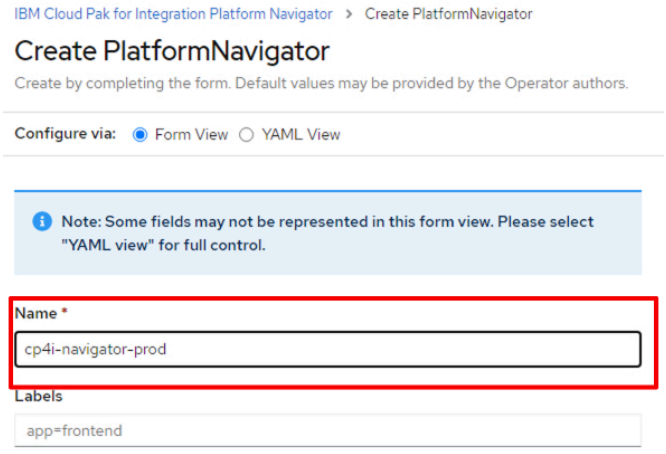
1. In the left navigation, click **Operators > Installed Operators**.
2. Click **IBM Cloud Pak for Integration Platform Navigator**.



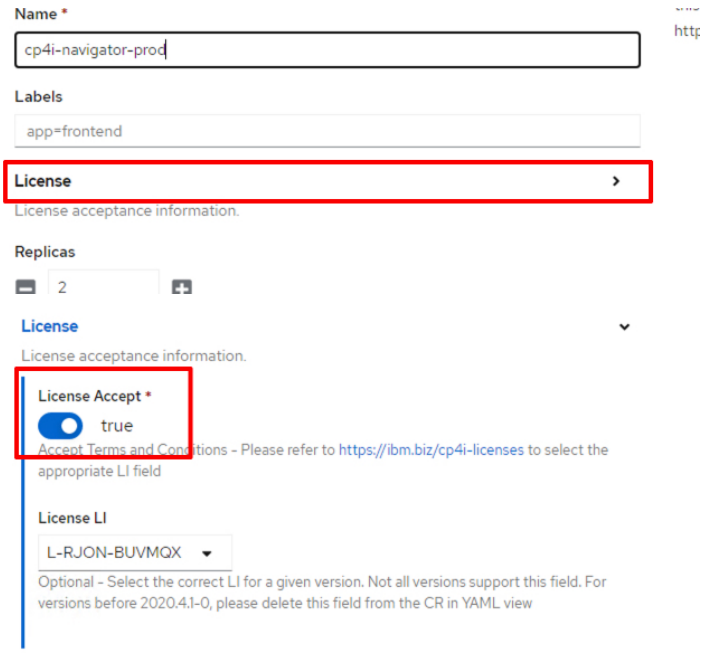
3. Click **Platform Navigator**, then click **Create Instance**



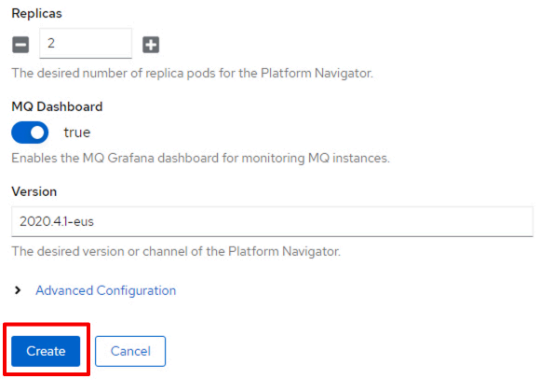
4. Change the **name** of Instance



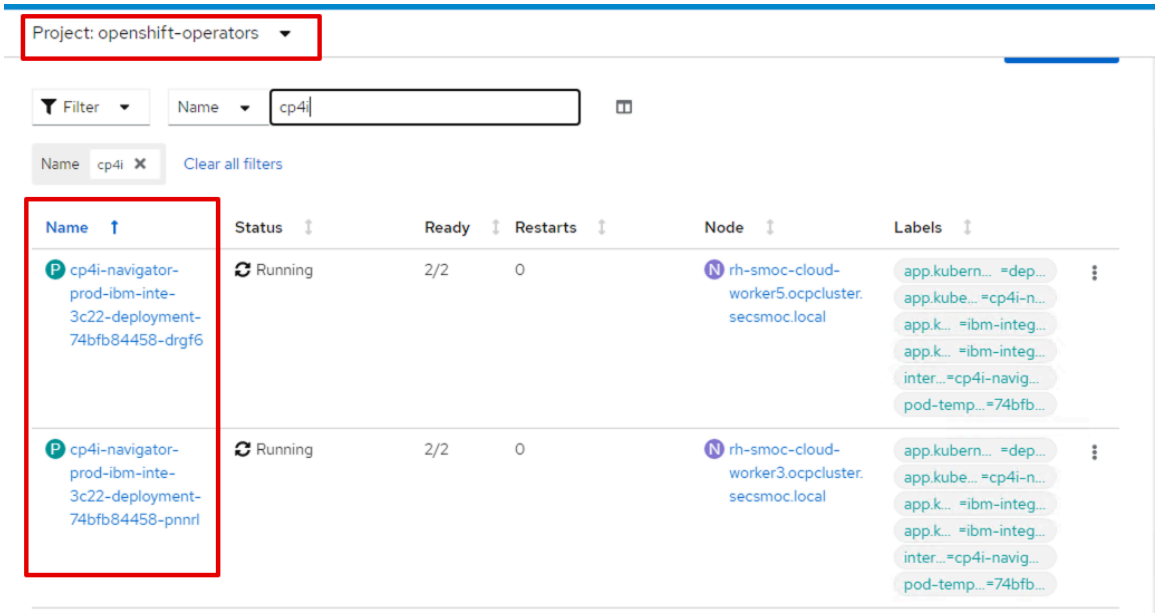
5. Accept the **License**



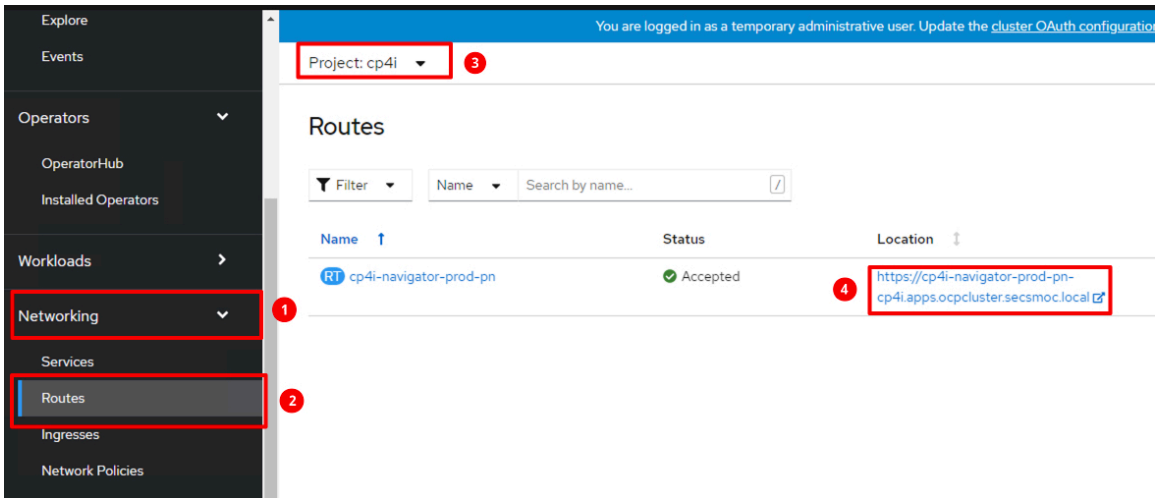
6. **Create** (Ensure version is set to 2020.4.1-eus)



7. Ensure all pods in `ibm-common-services` are running (should be around **65-67 pods**) and the two pods in Operator hub was created



8. After completed you can find the route here



Navigate to Platform Navigator login using **Openshift Authentication** or **Admin**:

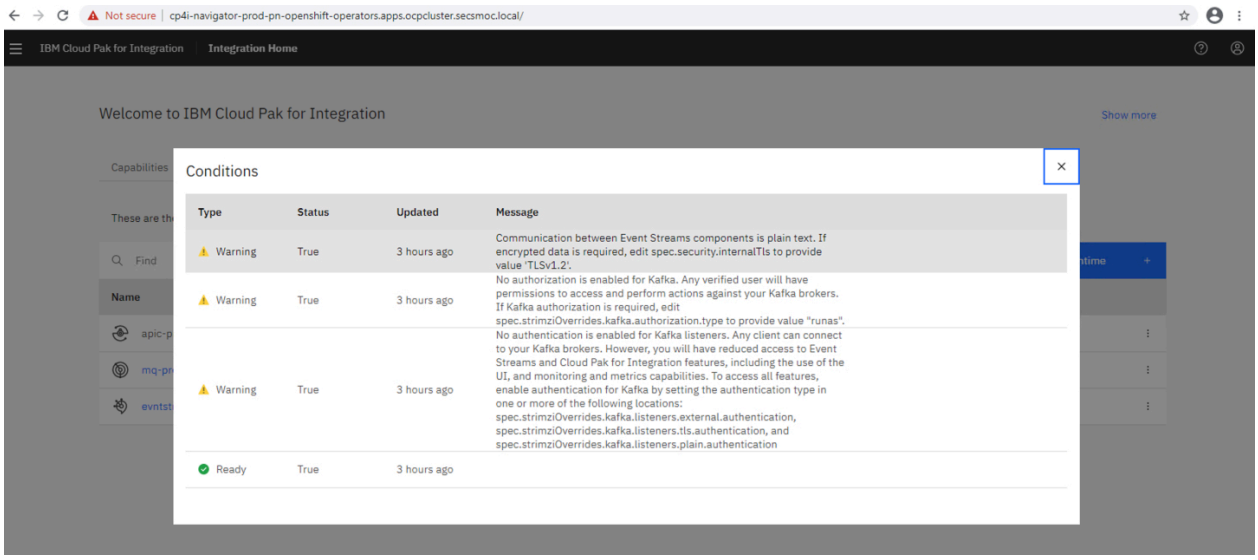
Login details can be found by running the following command:

```
$ oc extract secret/ibm-iam-bindinfo-platform-auth-idp-credentials -n ibm-common-services --to=-
```

Link

1. https://cp4i-navigator-prod-pn-openshift-operators.apps.ocpcluster.secsmoc.local/

EventStreams



If you find in the platform navigator event streams not ready and have a yellow warning sign do the following:

```
spec:
  security:
    internalTls: TLSv1.2 # change from NONE to TLSv1.2
  license:
    accept: true
    use: CloudPakForIntegrationProduction
  strimziOverrides:
    kafka:
      config:
        inter.broker.protocol.version: '2.6'
        interceptor.class.names:
com.ibm.eventstreams.interceptors.metrics.ProducerMetricsInterceptor
        log.message.format.version: '2.6'
        offsets.topic.replication.factor: 1
        transaction.state.log.min.isr: 1
        transaction.state.log.replication.factor: 1
      listeners:
        tls:
          authentication:
            type: tls # change plain: {} to the following
      metrics: {}
      # add these lines
      authorization:
        type: runas
       # end of addition
      replicas: 3
      storage:
        class: ocs-storagecluster-ceph-rbd
        size: 100Gi
        type: persistent-claim
    zookeeper:
      metrics: {}
      replicas: 1
      storage:
        class: ocs-storagecluster-ceph-rbd
        size: 100Gi
        type: persistent-claim
```

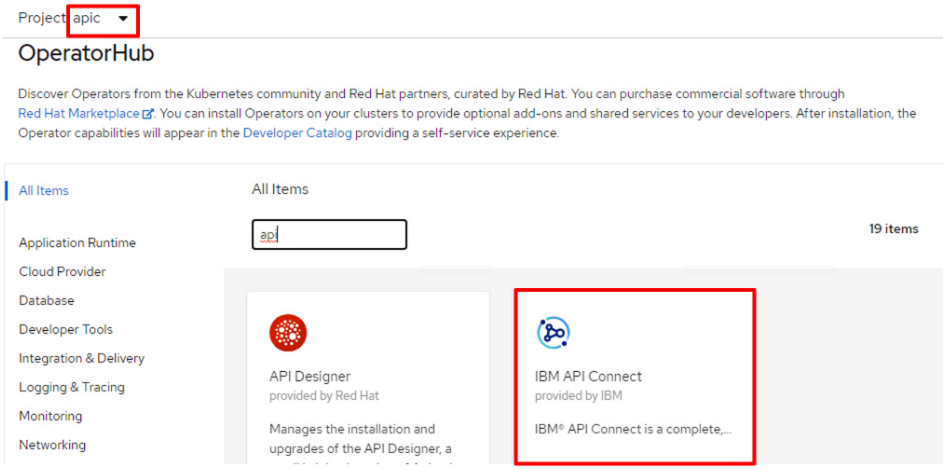
```
    adminUI: {}
    restProducer: {}
    apicurioRegistry: {}
    adminApi: {}
    collector: {}
    version: 10.2.0-eus
```

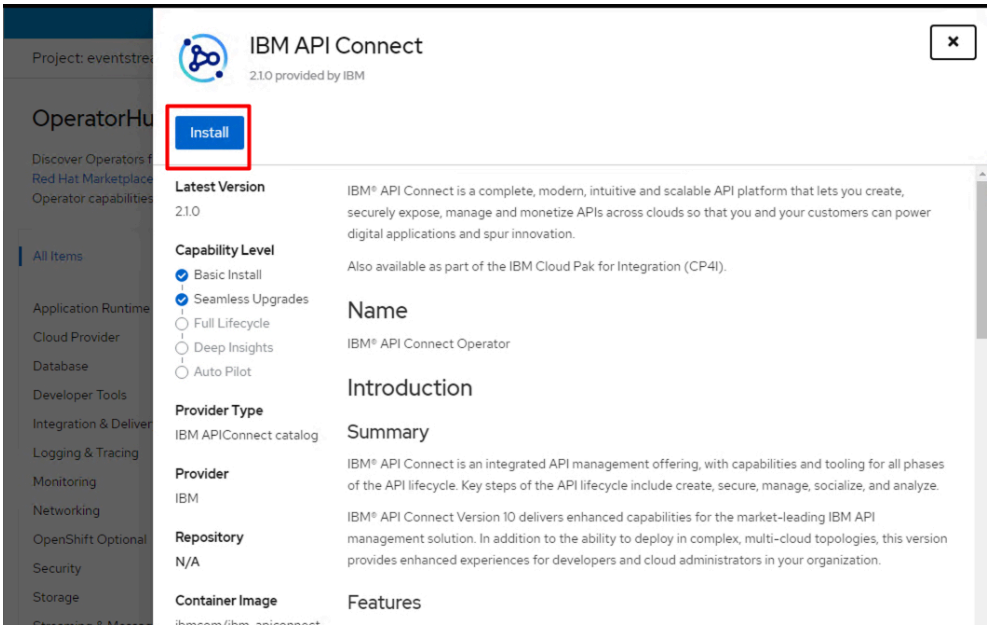
https://ibm.github.io/event-streams/installing/configuring/

API Connect

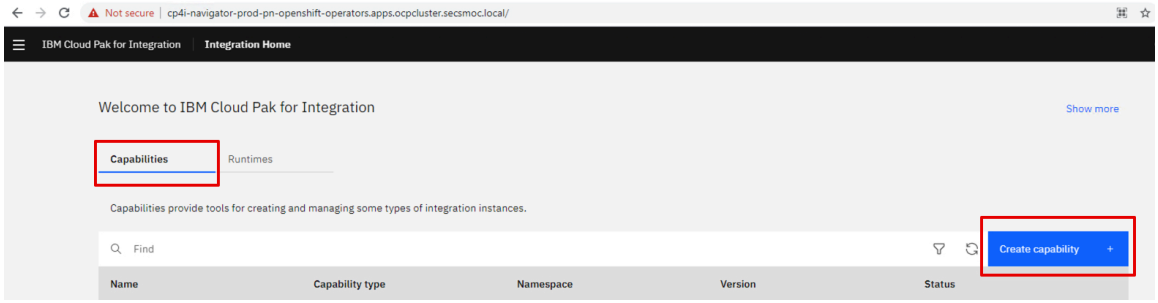
Install API Connect

1. Install **Operator** in `apic` project

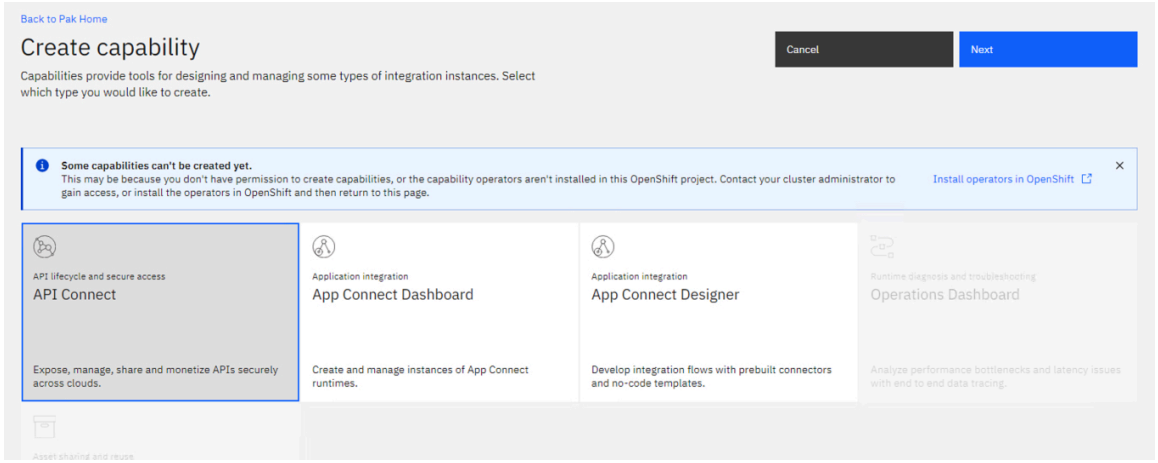




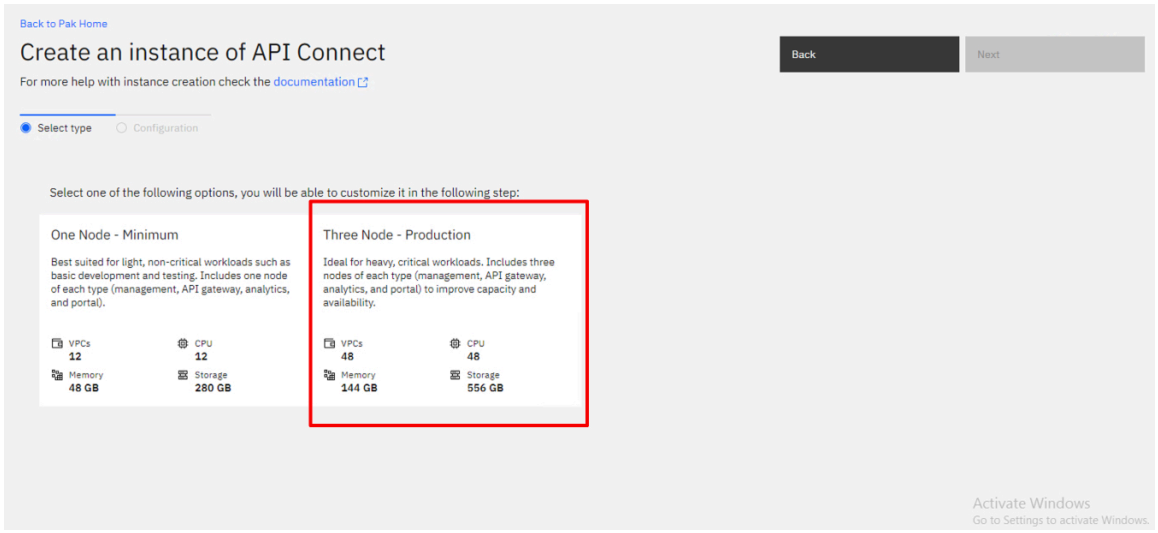
2. After the APIC operator installed Navigate to the **Platform Navigator** > **Create capability**
   https://cp4i-navigator-prod-pn-openshift-operators.apps.ocpcluster.secsmoc.local/



3. Click **API Connect** then click **Next**

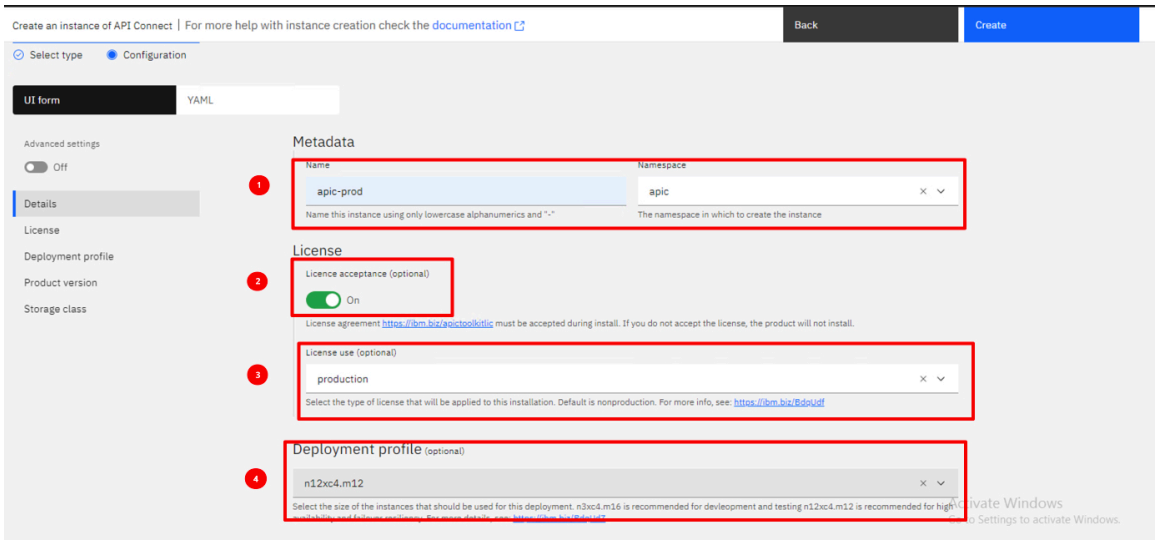


4. Select **Production** under types then click **Next**

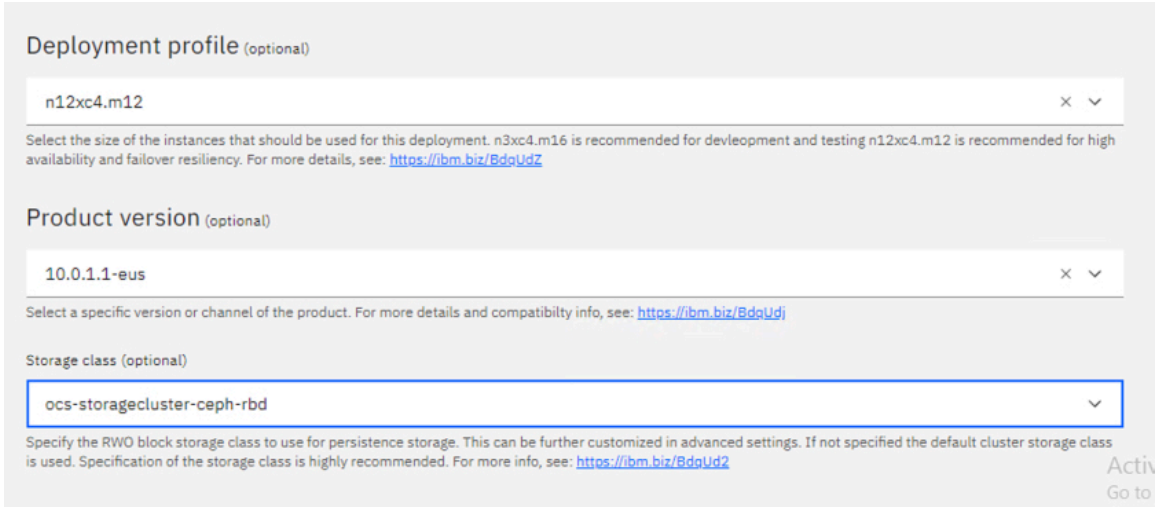


5. Fill in the details as below changing the following

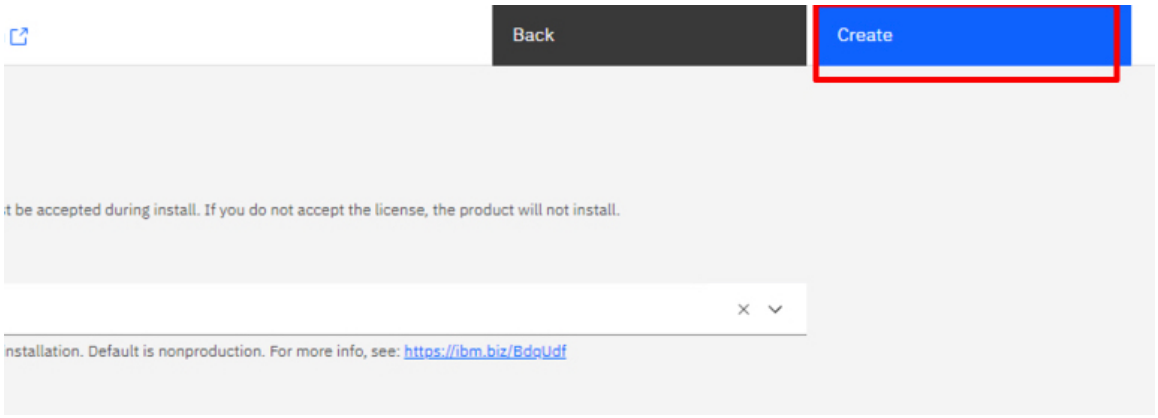
    - **Name**
    - **Accept** license,
    - Change License use to **Production**,
    - Select **Deployment Profile** = **n12xc4.m12**
    - **Storage Class** to match **ocs-storagecluster-cephfs**



6. Update the **Storage type** and **Version** to match below



7. Click **Create**



Make sure the deployment succeeded. (lots of pods with a subset below)

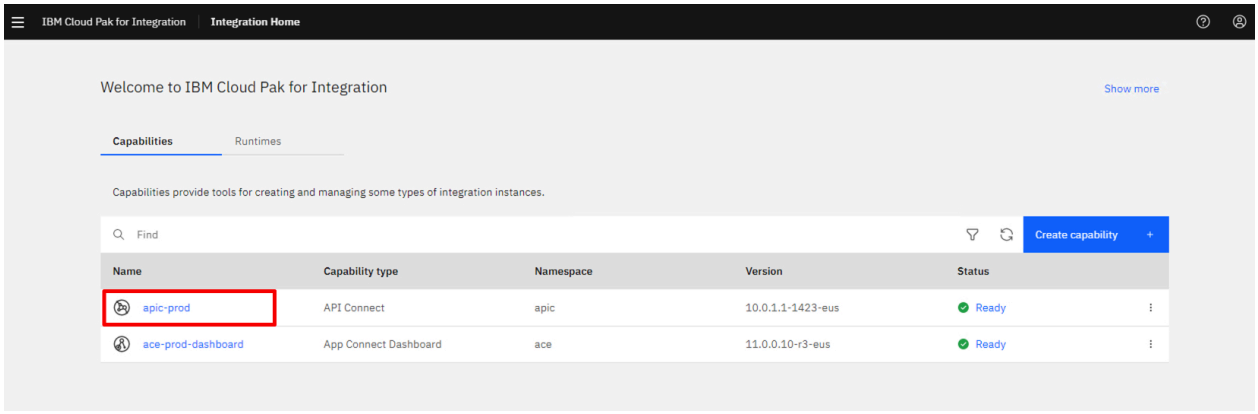
```
$ oc get pods -n apic | grep apic

apic-a7s-cj-rollover-1612818900-x2rnt                             0/1
Completed   0          5m20s
apic-a7s-client-79b58ccd5f-kd24h                                  1/1
Running     0          45m
```

```
apic-a7s-ingestion-669f4d4bb4-nwfwx                               1/1
Running     0          45m
apic-a7s-mtls-gw-c44d4df6f-2fptt                                  1/1
Running     0          43m
apic-a7s-storage-coord-56965cf6f7-kxsk5                           1/1
Running     0          45m
apic-a7s-storage-data-0                                           1/1
Running     0          45m
apic-a7s-storage-master-0                                         1/1
Running     0          45m
apic-configurator-wrbdw                                           0/1
Completed   0          37m
apic-gw-0                                                         3/3
Running     0          41m
apic-gw-od-registration-job-6nmh5                                 0/1
Completed   0          45m
apic-mgmt-08730b55-postgres-765d98c78f-msqb2                      1/1
Running     0          58m
apic-mgmt-08730b55-postgres-backrest-shared-repo-5796c7887nzjfk   1/1
Running     0          60m
apic-mgmt-08730b55-postgres-pgbouncer-6f48bb576-tvldx             1/1
Running     0          56m
apic-mgmt-08730b55-postgres-stanza-create-hgmgq                   0/1
Completed   0          56m
apic-mgmt-analytics-proxy-598b786df5-frzzz                        1/1
Running     0          50m
apic-mgmt-apim-647f46f7b4-c9hmq                                   1/1
Running     0          47m
apic-mgmt-billing-798667ccdd-48rwp                                1/1
Running     0          50m
apic-mgmt-client-downloads-server-7f7ff89858-rhsnw                1/1
Running     0          55m
apic-mgmt-hub-b98b4f5b6-fwqgg                                     1/1
Running     0          55m
apic-mgmt-juhu-6cc6556549-48bd5                                   1/1
Running     0          48m
apic-mgmt-ldap-5d99986776-lpbqv                                   1/1
Running     0          55m
apic-mgmt-lur-7995cc77b9-vd9pq                                    1/1
Running     0          47m
apic-mgmt-nats-operator-6747c9b5f4-h5bw9                          1/1
Running     0          55m
apic-mgmt-nats-streaming-operator-6dc6794f46-b957q                1/1
Running     0          55m
apic-mgmt-natscluster-1                                           1/1
Running     0          55m
apic-mgmt-portal-proxy-64ccd9494c-prsls                           1/1
Running     0          50m
apic-mgmt-stancluster-1                                           1/1
Running     0          54m
apic-mgmt-taskmanager-5b85ccbd4f-dhv7j                            1/1
Running     0          50m
apic-mgmt-turnstile-55f79cbf8d-kvrv2                              1/1
Running     0          50m
apic-mgmt-ui-7bb65c78fd-kghpf                                     1/1
Running     0          55m
```

```
33  apic-mgmt-up-apim-data-populate-0-to-6-216243c7-dlk5d                0/1
    Completed   0          53m
34  apic-mgmt-up-apim-schema-0-to-6-216243c7-v2cjm                       0/1
    Completed   0          55m
35  apic-mgmt-up-lur-data-populate-0-to-2-216243c7-4h65w                 0/1
    Completed   0          54m
36  apic-mgmt-up-lur-schema-0-to-2-216243c7-77zww                        0/1
    Completed   0          55m
```

To access your APIC instance you can navigate to the **Platform navigator**

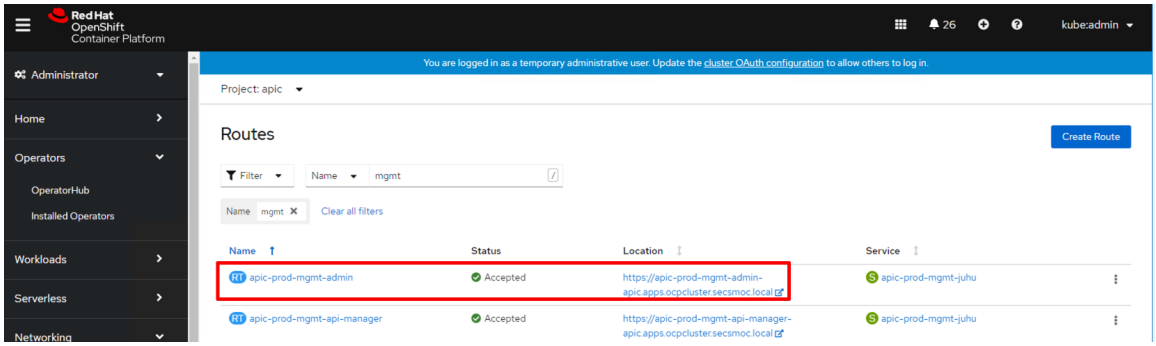


Configure API Connect

Configure a dummy email client

1. Navigate to Cloud Manager

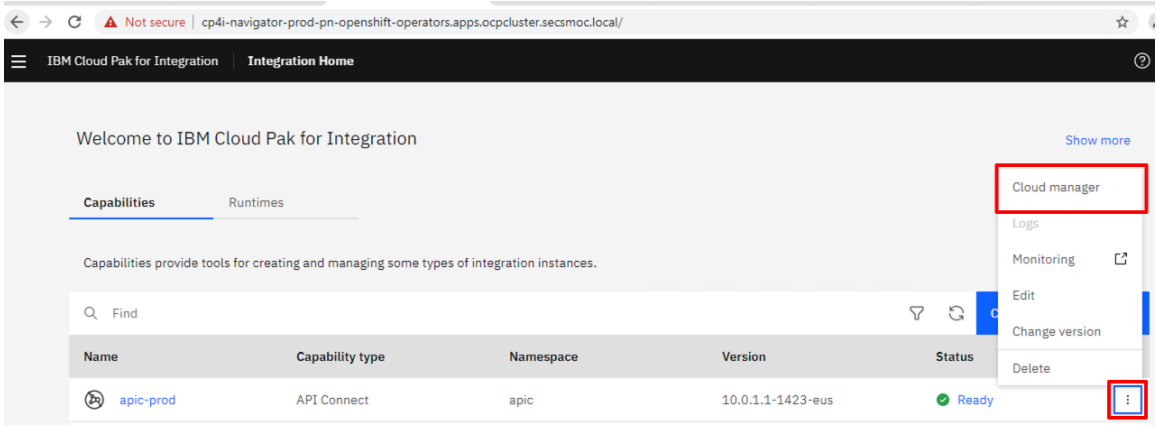
   Cloud Manager access route



   Password

```
1  oc get secrets -n apic apic-prod-mgmt-admin-pass -ojsonpath='{.data.password}'
   | base64 --decode && echo ""
```

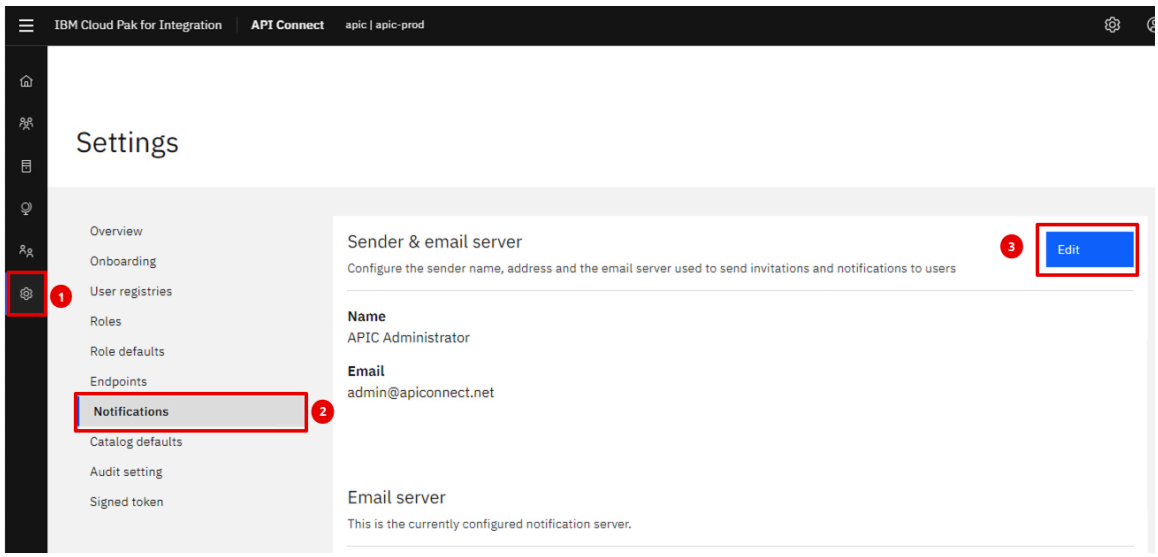
2.



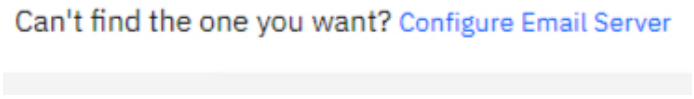
3. Login using credentials

```
$ oc extract secret/ibm-iam-bindinfo-platform-auth-idp-credentials -n ibm-common-services --to=-
```

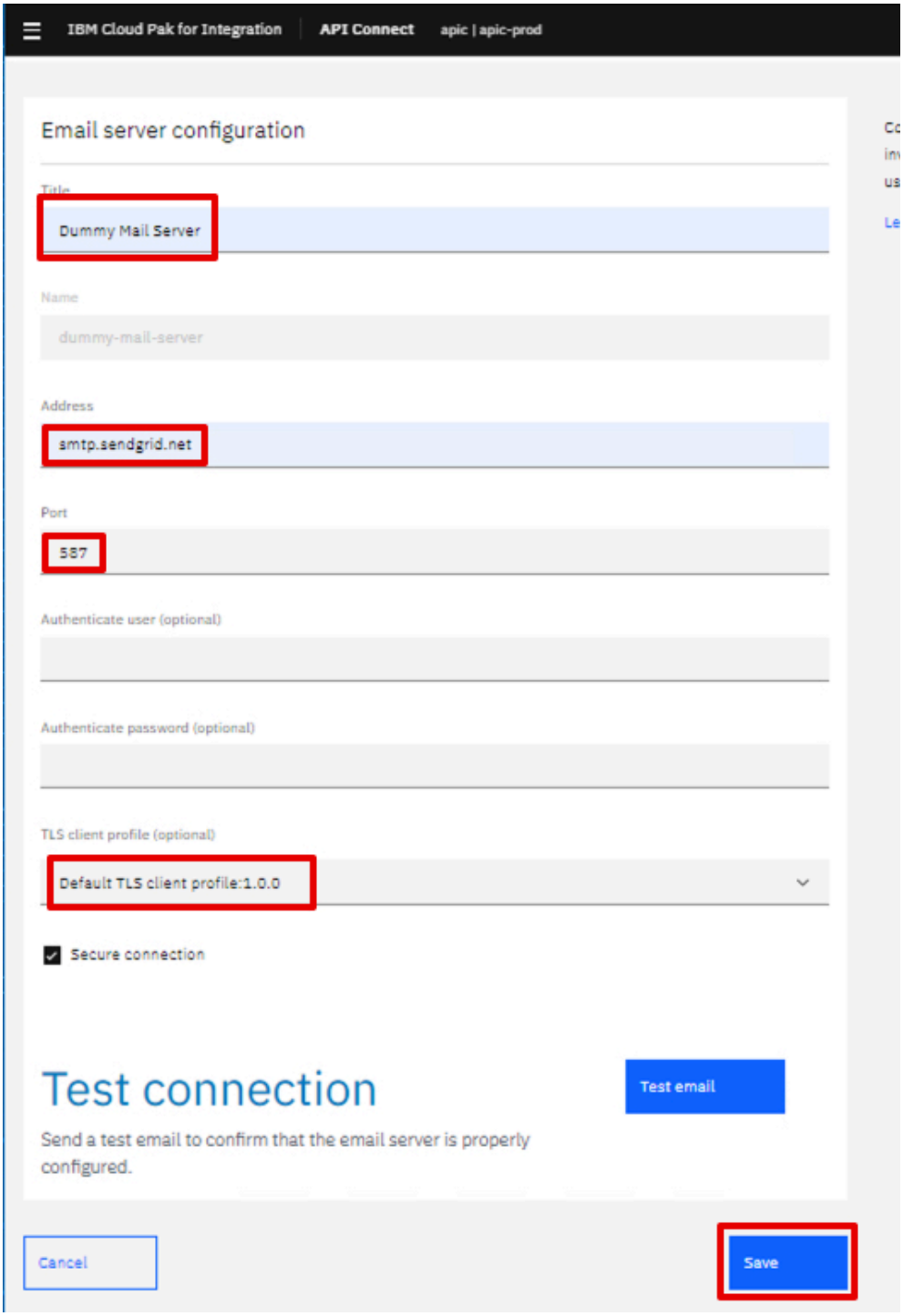
4. Navigate to **Settings** > **Notifications** > **Edit**



5. **Configure Email Server**



6. Enter the following and click **Save**



Reference

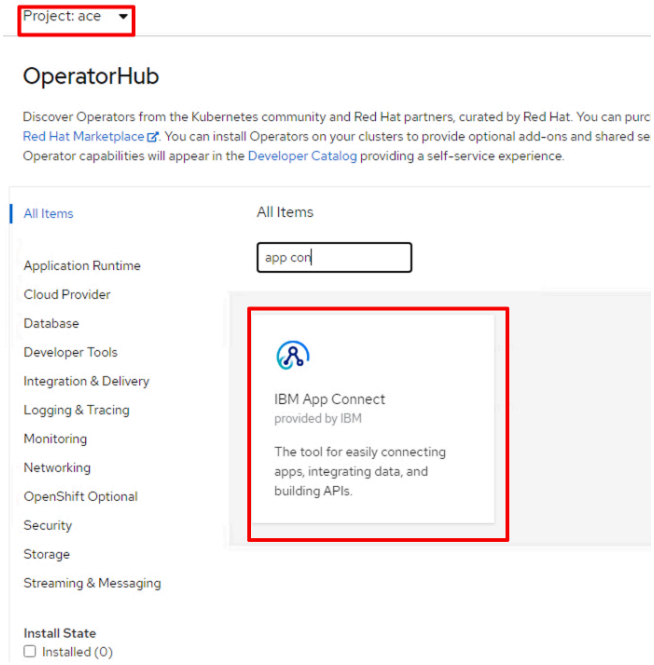
1. https://www.ibm.com/docs/en/api-connect/10.0.1.x?topic=configuration-basic-settings

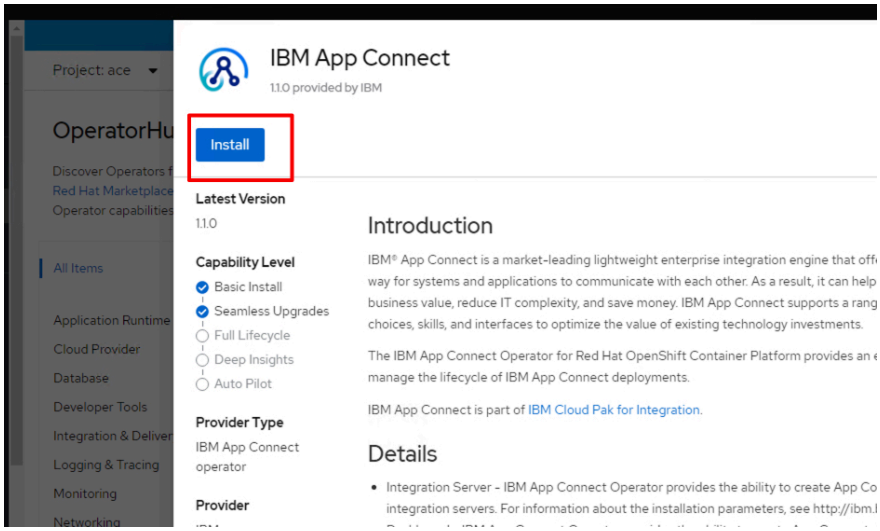
Link

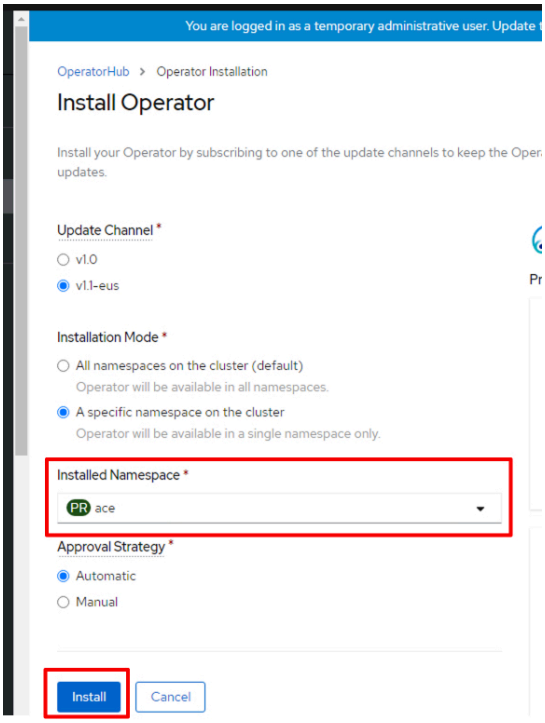
1. https://apic-prod-mgmt-admin-apic.apps.ocpcluster.secsmoc.local/admin/
2. https://apic-prod-mgmt-api-manager-apic.apps.ocpcluster.secsmoc.local/manager

App Connect

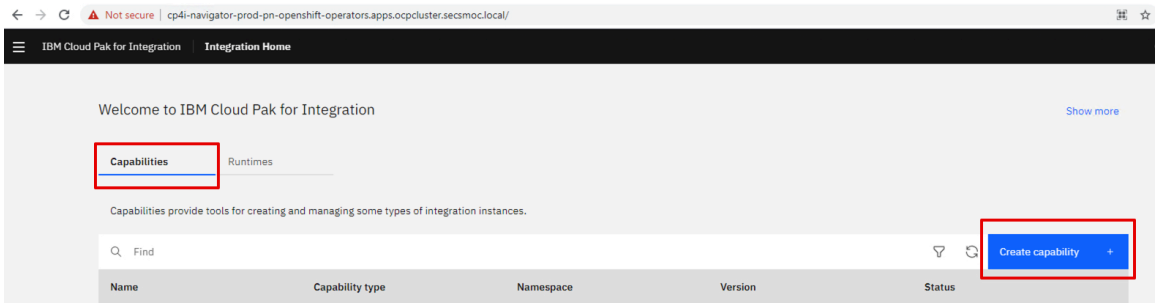
1. Install **Operator** in `ace` project



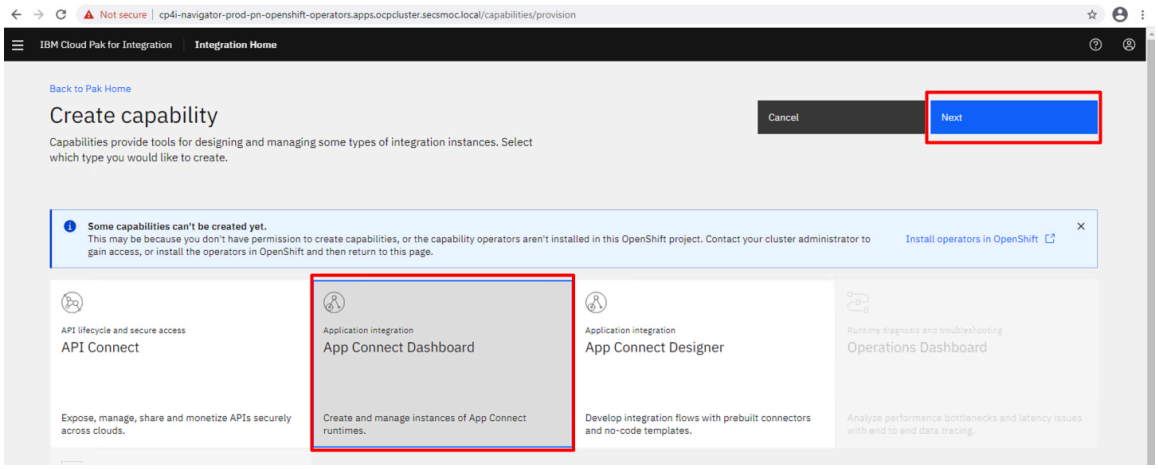




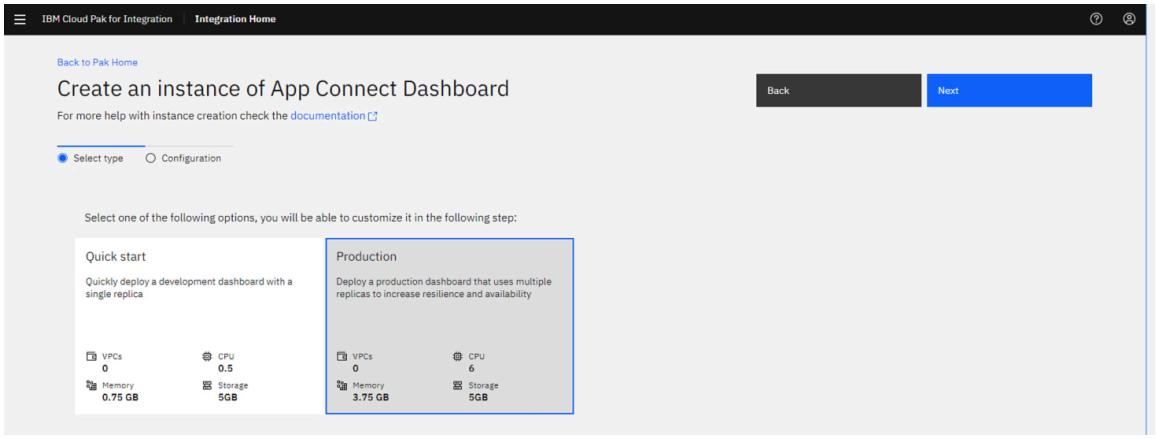
2. After the ACE operator installed Navigate to the **Platform Navigator** > **Create capability**
https://cp4i-navigator-prod-pn-openshift-operators.apps.ocpcluster.secsmoc.local/



3. Click **Ace Connect Dashboard** then click **Next**

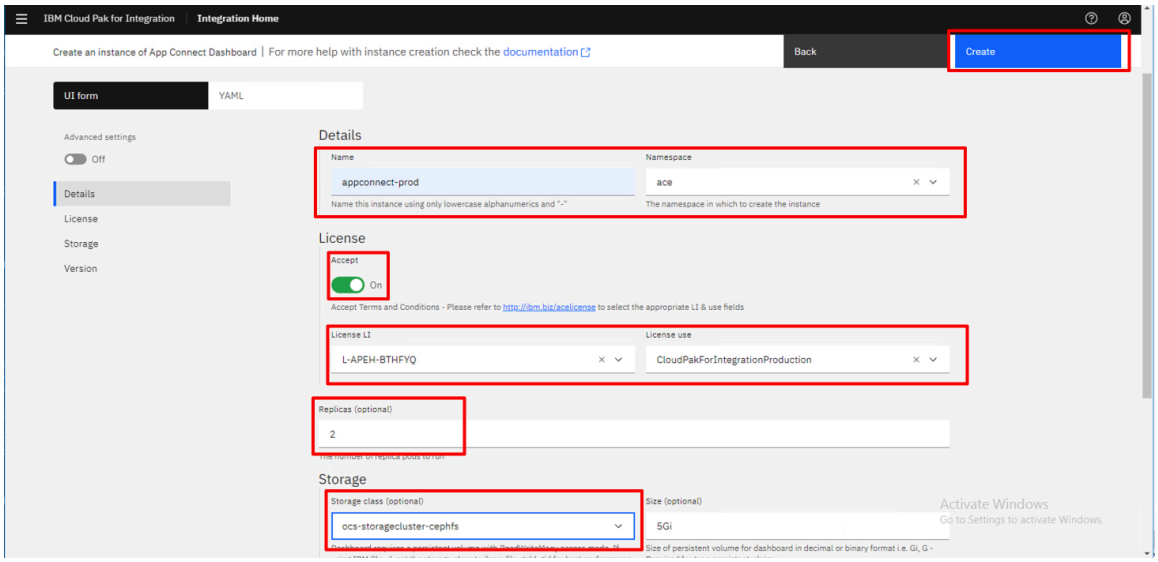


4. Select **Production** under types then click **Next**

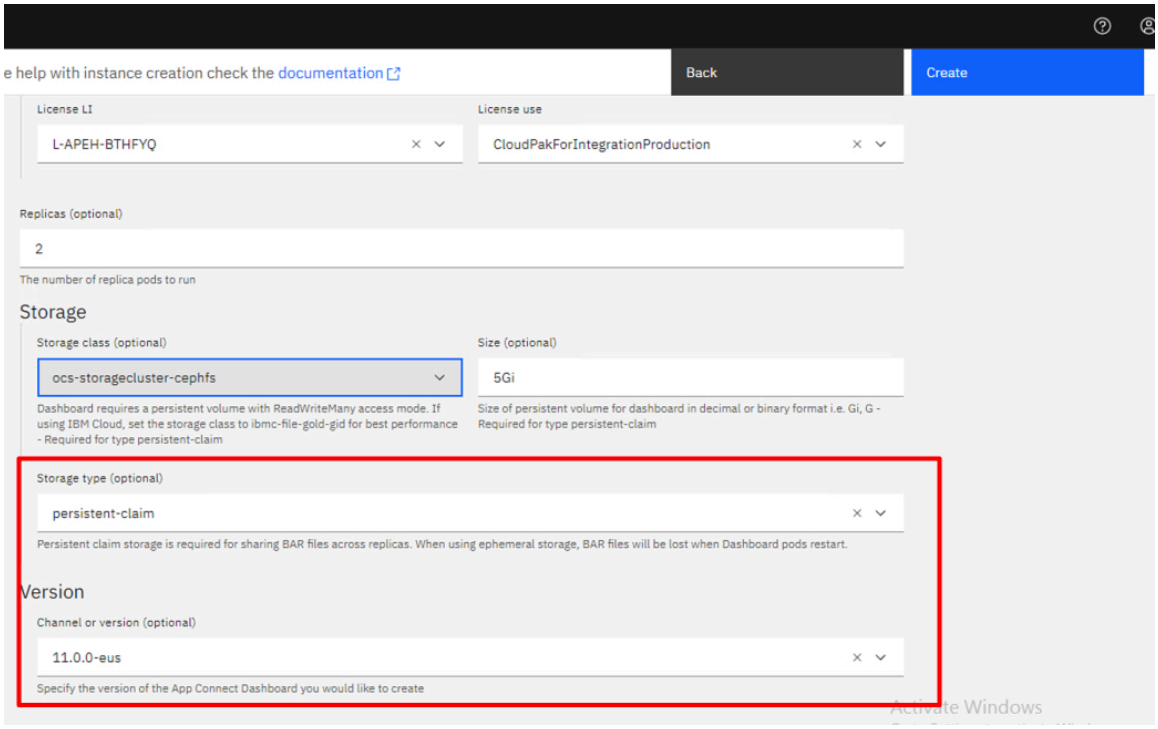


5. Fill in the details as below changing the following

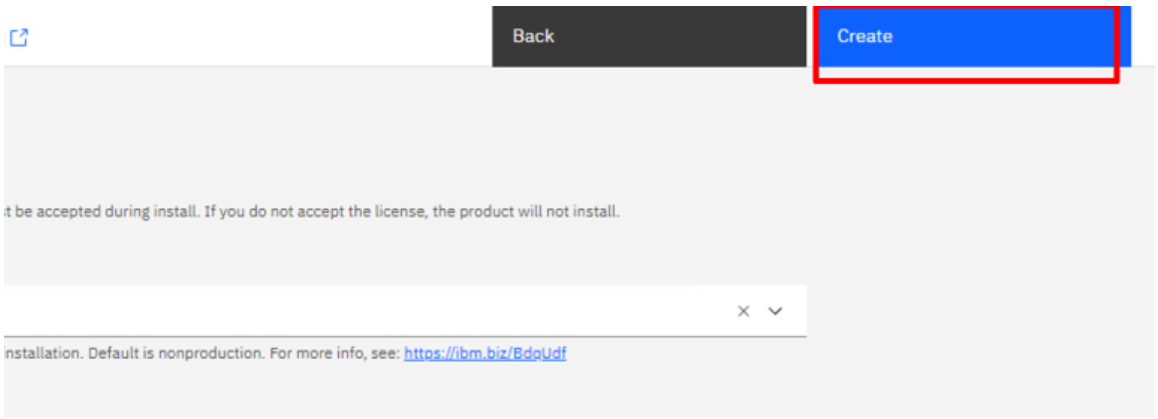
    - **Name**
    - **Accept** license,
    - Change License use to **Cloud Pak for Integration Production**,
    - **Replicas** = 2
    - **Storage Class** to match **ocs-storagecluster-cephfs**



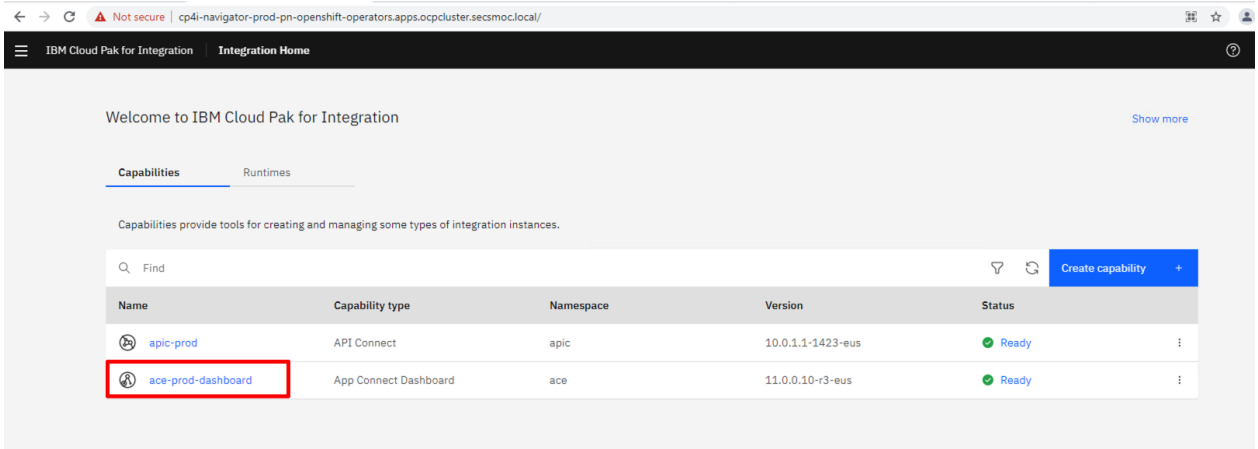
6. Update the **Storage type** and **Version** to match below



7. Click **Create**



To access your ACE instance you can navigate to the **Platform navigator**

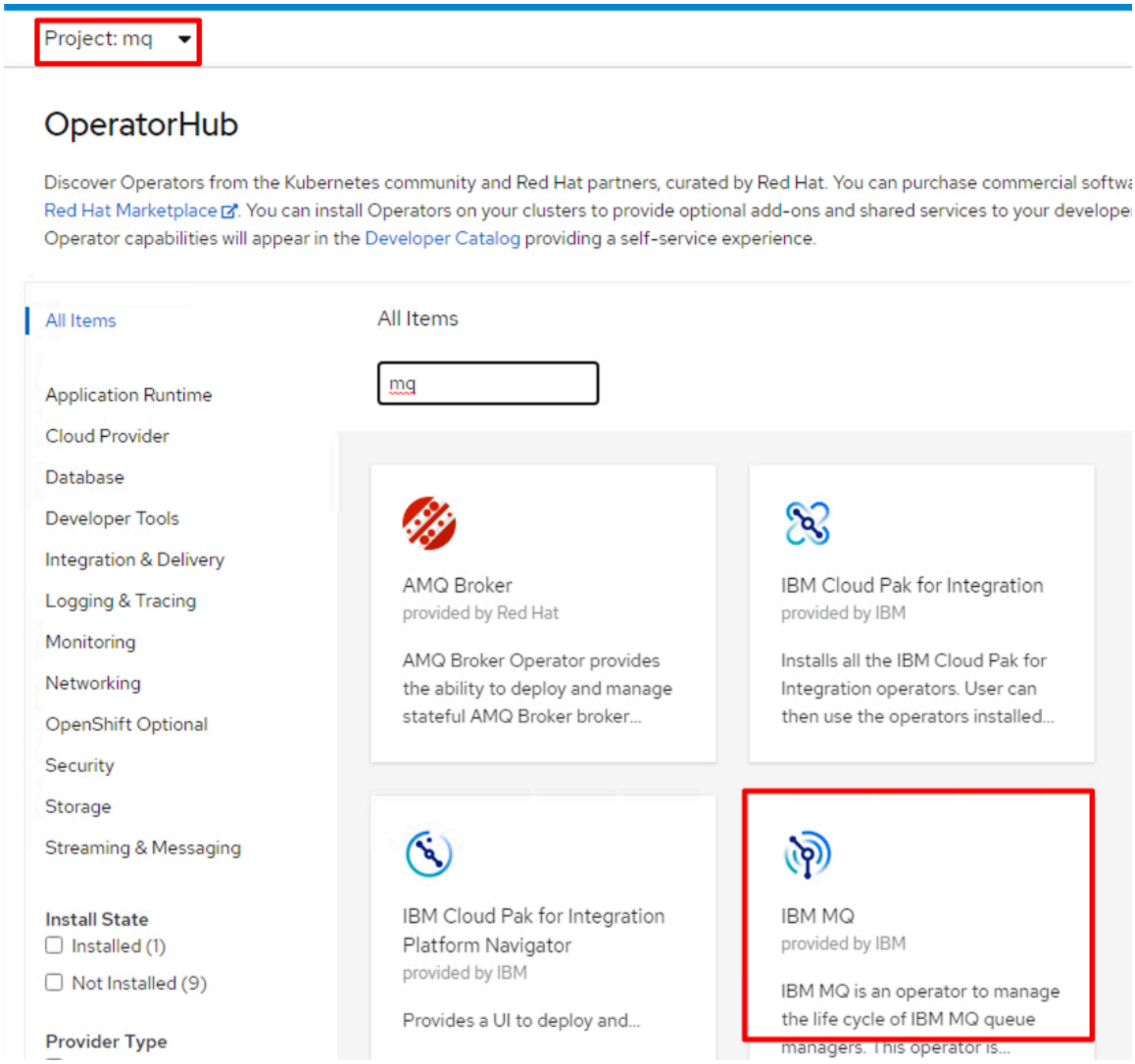


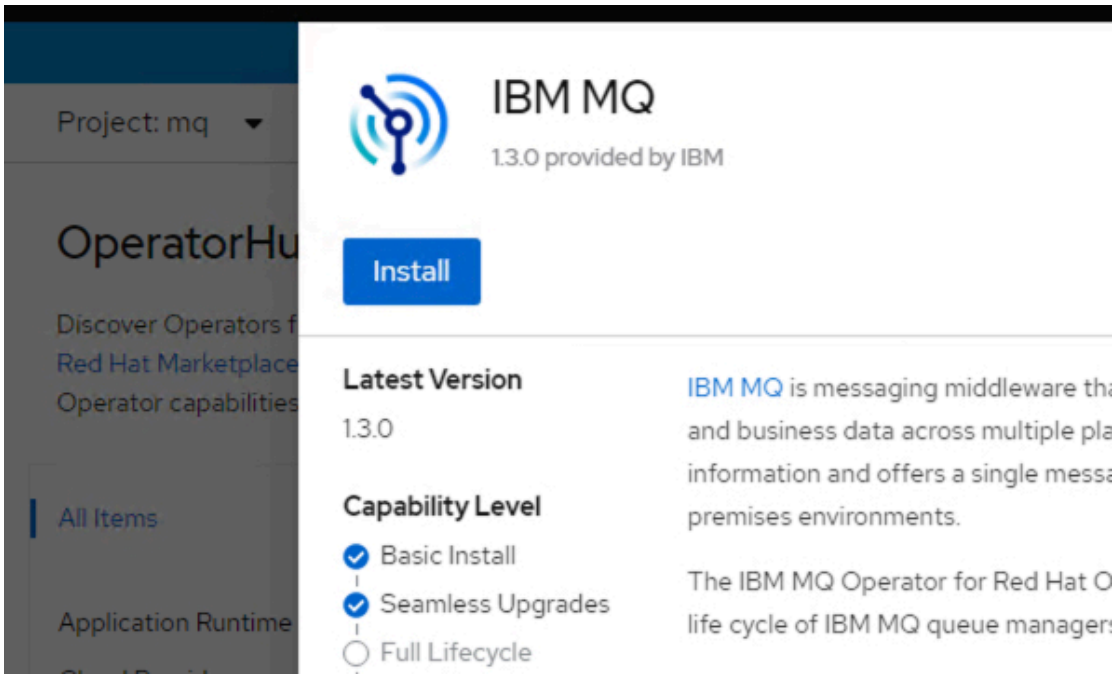
Link

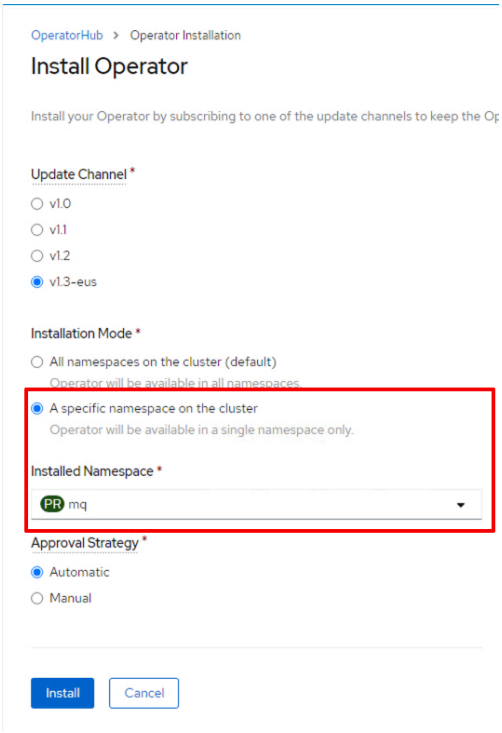
1. https://ace-prod-dashboard-ui-ace.apps.ocpcluster.secsmoc.local/home

MQ

1. Create MQ Operator







References

1. https://www.ibm.com/docs/en/ibm-mq/9.2?topic=dqmorhocpc-deploying-queue-manager-using-cloud-pak-integration-platform-navigator
2. https://www.ibm.com/docs/en/ibm-mq/9.2?topic=operator-example-configuring-multi-instance-queue-manage