ZTP for Factory Workflow

Overview

Zero touch provisioning for factory workflows (ZTPFW) accelerates the deployment of OpenShift Container Platform with pre-certified hardware and software for rapid edge deployments.

ZTP for factory workflows enables original equipment manufacturer (OEM) partners to pre-install OpenShift Container Platform at their factory and build turnkey solutions on their hardware. This approach is well suited to a range of different industries including:

- healthcare
- manufacturing
- aerospace
- media
- entertainment
- retail
- telecommunications

ZTP for factory workflows installs the components that enable you to use OpenShift Container Platform as a disconnected hub cluster. This hub cluster is then able to deploy edge clusters that can be shipped off site for final configuration.

At the factory, the OEM partner first deploys a hub OpenShift Container Platform cluster and then uses the hub cluster to deploy one or more edge clusters at scale.

The hub cluster can be a single-node OpenShift cluster deploying multiple single-node OpenShift edge clusters or a compact cluster deploying 3 control plane and 1 worker node edge clusters at scale.

NOTE The hub cluster is also known as the factory cluster.

The following are the possible combinations of hub and edgecluster cluster topologies:

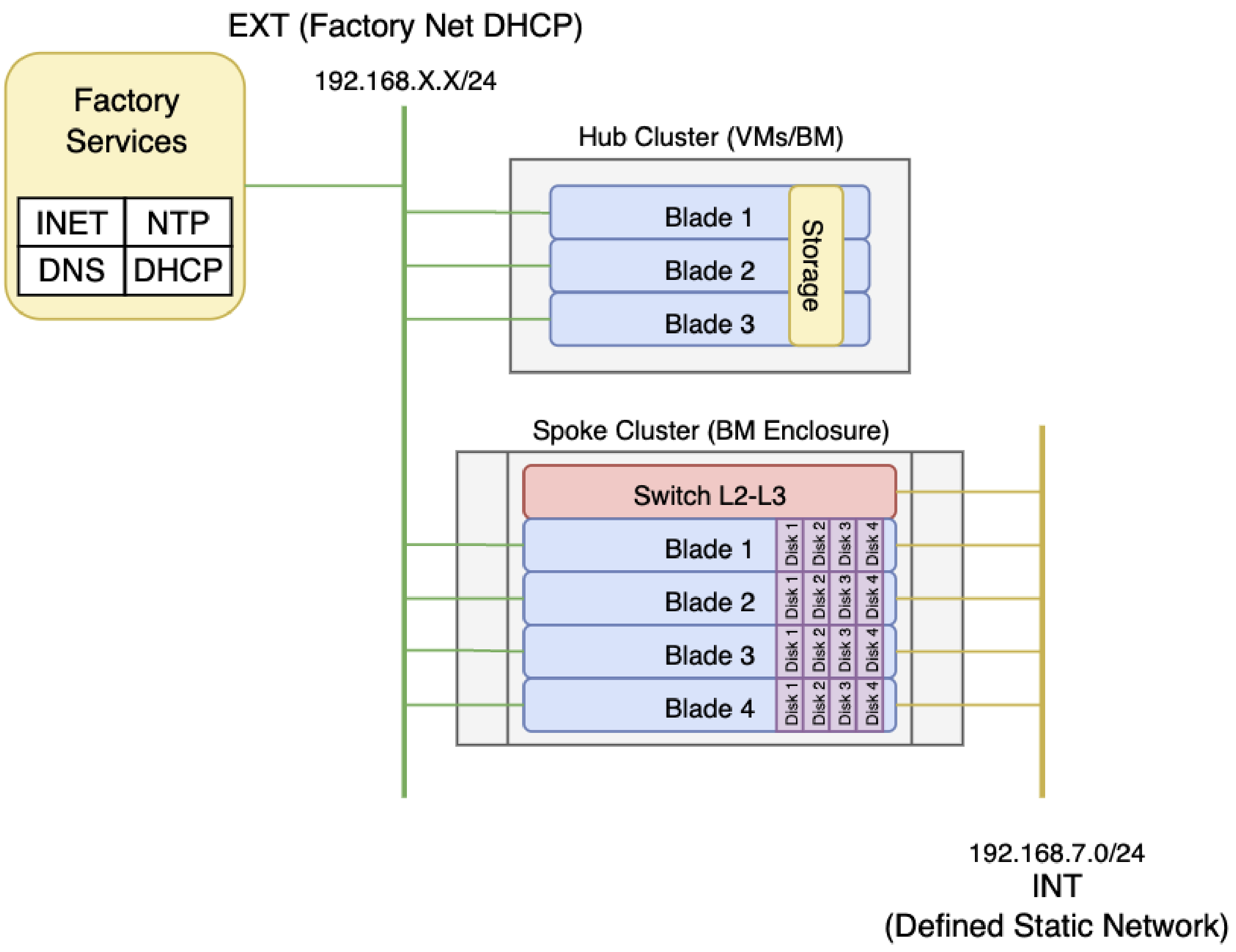
Table 1. Cluster topologies

Hub	Edge
Compact (3 control plane nodes also able to act as worker nodes)	3 + 1 (Compact and 1 worker node) Compact Single-node OpenShift
Single-node OpenShift (Control plane and worker node on a single node)	3 + 1 Compact Single-node OpenShift

Whatever the topology, the hub cluster uses Red Hat Advanced Cluster Management (RHACM) and the Assisted Installer (AI) to install edge clusters at scale by using zero touch provisioning (ZTP).

When the edge cluster is installed, it can be shipped to the customer onsite locations and there the end customer unboxes it, and configures the edge cluster making it fully operational.

The actual workflow and its details can be checked at the files inside the `pipelines` folder.



Hub and edge cluster architecture

After running all workflows in the hub and edge cluster pipelines, the architecture for a compact hub and 3 plus 1 edge cluster may resemble the following:

> **NOTE**
>
> In the documentation and particularly with reference to the various scripts invoked you might see the term edgecluster cluster or edgecluster clusters used. The preferred term to use in relation to ZTPFWs is edge cluster or edge clusters and they effectively mean the same thing.

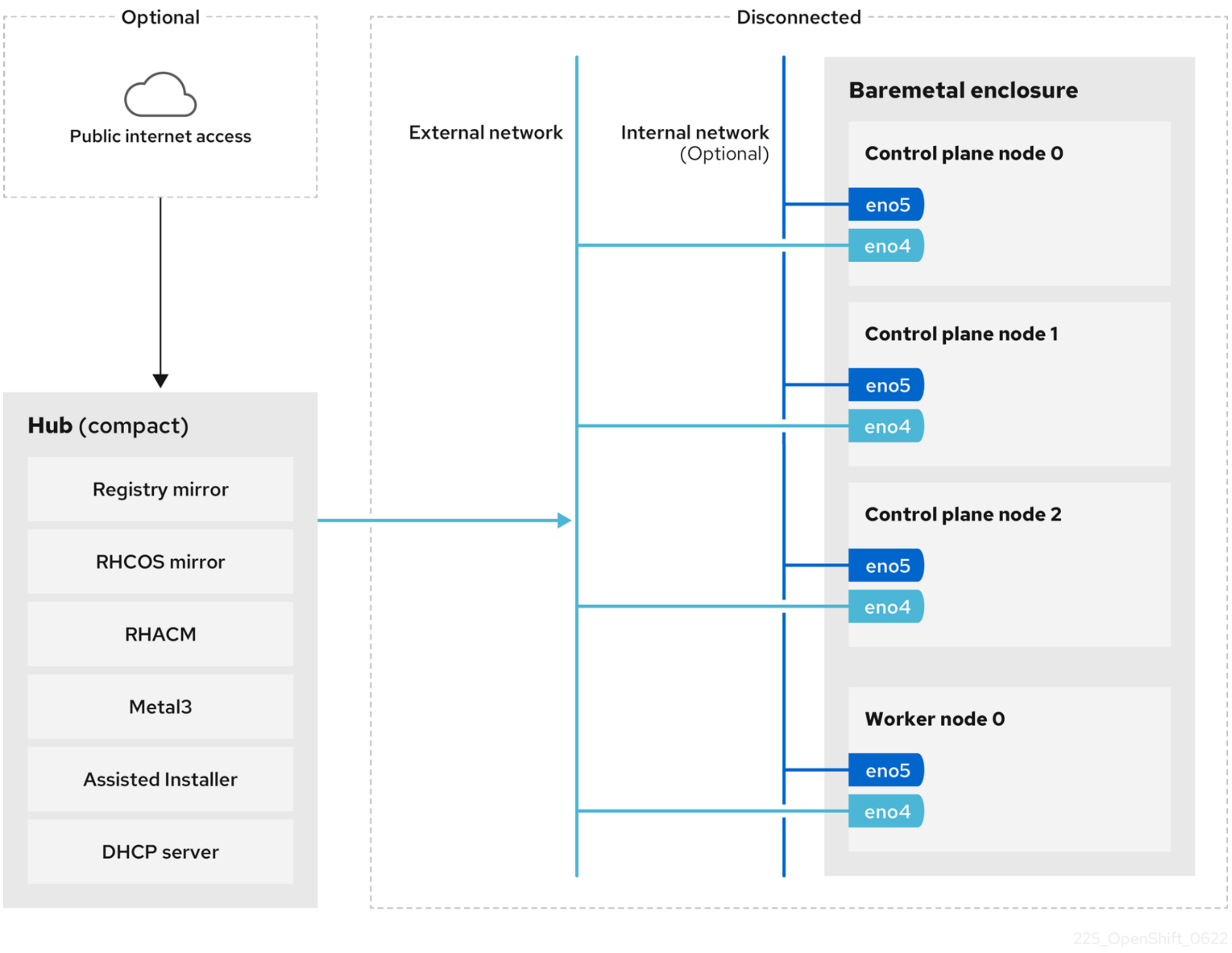


Figure 1. Compact hub and 3 + 1 edge cluster architecture

Every blade in the chassis has access to multiple NICs, which are connected to internal switches. Switches and NICs are referred to as networks using the name of the interface. The `eno4` and `eno5` networks are 10gbs networks with enough bandwidth to support the internal and external traffic of the cluster. The `eno4` network is used as the external network. It will be configured by DHCP to make it easier for the factory to configure and interact with it. This also simplifies the on site customer configuration. The `eno5` network is the internal network. It is only to be accessible from within the blades (isolated). This network is configured with static IPs and is expected to be used for the internal traffic of the cluster. The client also connects to this network and uses it to reconfigure the external connection. This interface (eno5) is optional. A vlan on eno4 will be created if eno5 is not specified. The switch access port should allow to pass vlan tag traffic using trunking, in this case.

NOTE The public internet access is initially required when working on the hub and can be disconnected later after everything is synced. The network interface names `eno4` and `eno5` are configurable in the `edgeclusters.yaml` file.

Preparing the factory install environment

Base prerequisites

- Deploy the OpenShift Container Platform cluster with three control plane nodes following the guidance in the section Deploying installer-provisioned clusters on bare metal.
    - Alternatively you can use the technology preview Assisted Installer from cloud.redhat.com to create the cluster.
- To install single-node OpenShift follow the guidance in Installing on a single node in the OpenShift Container Platform documentation.
- All cluster Operators are available.
- Cluster is reachable using a `KUBECONFIG` file.
- The dns names for `api.<hub-clustername>.<baseDomain>`, `api-int.<hub-clustername>.<baseDomain>` and `*.apps.<hub-clustername>.<baseDomain>` should be deployed on edge cluster on the DHCP external network.
- Metal³ has to be available in the hub cluster.

Storage prerequisites

- Storage can be provided by installing the Local Storage Operator and by using local volumes or by using OpenShift Data Foundation (ODF).

NOTE If the cluster is greater than 3 nodes, the recommendation is to use OpenShift Data Foundation. If it is a single-node OpenShift cluster, use the Local Storage Operator.

- Create the following persistent volumes with at least 200GB of storage (NVMe or SSD) for:
    - 2 for Assisted Installer.
    - 1 for the hub internal registry that is for the mirror of the images. At least 200GB is required on the hub, more may be required if ODF is installed.
    - 1 for HTTPD that hosts the Red Hat Enterprise Linux CoreOS (RHCOS) images.
    - 1 for zero touch provisioning factory workflows (ZTPFW).
    - 1 for Red Hat Advanced Cluster Manager (RHACM)

Networking prerequisites

The hub cluster requires internet connectivity and should be installed on a private network with customer configured DNS and DHCP services. Configure DNS for the API on the ingress of the hub to reach some routes on the hub cluster. Configure enough DNS entries for the number of edge clusters you intend to deploy in parallel.

You need enough DHCP addresses to host the number of edge clusters you intend to deploy. Each OpenShift Container Platform node in the cluster must have access to an NTP server. OpenShift

Container Platform nodes use NTP to synchronize their clocks. For example, cluster nodes use SSL certificates that require validation, which might fail if the date and time between the nodes are not in sync.

Specific requirements are:

- DNS entries need to be configured and resolvable from the external network, with DNS on the DHCP external network.
- Hub
  - `api.<hub-clustername>.<baseDomain>` and `api-int.<hub-clustername>.<baseDomain>` entries to the same IP address.
  - ingress (`*.apps.<hub-clustername>.<baseDomain>`).
- Edge
  - `api.<edge-cluster-name>.<baseDomain>` and `api-int.<edge-cluster-name>.<baseDomain>` entries to the same IP address.
  - ingress (`*.apps.<edge-cluster-name>.<baseDomain>`).

NOTE When deploying a single-node OpenShift cluster, the `api.<edge-cluster-name>.<baseDomain>` and `*.apps.<edge-cluster-name>.<baseDomain>` must be configured with different IP addresses.

- External DHCP with enough free IPs on the factory network to provide access to the edge cluster by using the external network interface.
- Every edge cluster needs at least 6 IPs from this external network (without the broadcast and network IP).
  - 1 per node.
  - 1 for API.
  - 1 for API-INT.
  - 1 for the Ingress entry (`*.apps.<edge-cluster-name>.<baseDomain>`).

About the factory install pipeline

The factory install pipeline builds out your factory environment (hub and edge clusters) in readiness for the edge cluster to be shipped off site. Red Hat has created a set of community scripts to help you get started with this task.

A GitHub repository contains all the relevant scripts and YAML files you need to provision the hub cluster and edge clusters.

The edge cluster installation uses a zero touch provisioning (ZTP) approach facilitated by Red Hat Advanced Cluster Management (RHACM) using the Assisted Installer (AI) installed as part of running the factory install pipeline.

With ZTP and AI, you can provision many OpenShift Container Platform edge clusters in a factory-type setting. RHACM manages clusters in a hub and edge architecture, where a single hub cluster manages many edge clusters. A hub cluster running RHACM provisions and deploys the edge clusters using ZTP and AI. AI provisions OpenShift Container Platform on the bare-metal edge clusters.

Factory install workflow

The factory install pipeline builds out your factory environment in readiness for the edge cluster to be shipped off site.

The following diagram provides a high level overview of the pipelines used to prepare the edge clusters:

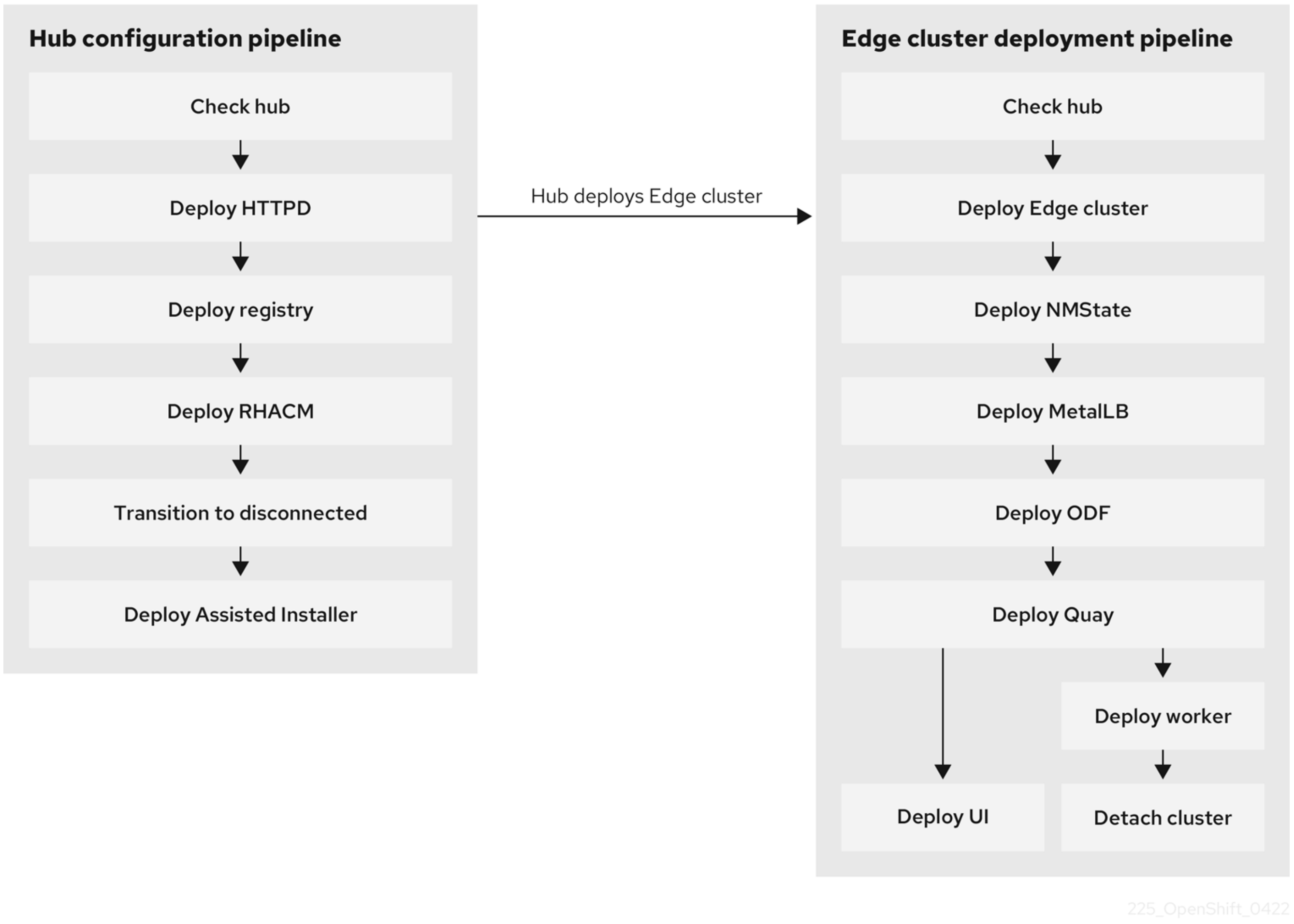


Figure 2. Hub and edge pipelines

NOTE There are no tasks that run in parallel.

- **Hub deployment**: This first part deploys the hub cluster configuration. The assumption being OpenShift Container Platform and optionally OpenShift Data Foundation is installed with persistent volumes created with supporting DHCP and DNS configuration.
- **Edge deployment**: This second part deploys relocatable edge clusters on the preferred hardware in parallel. When the deployment completes, the hardware where the edge cluster is installed is shipped to the end customer. The end customer runs some on site configuration steps and then has a fully operational OpenShift Container Platform cluster.

Hub factory pipeline

The hub configuration pipeline stage prepares the hub cluster to deploy multiple edge clusters for the end customer.

The flow associated with deploying the hub cluster is:

Check hub

The initial stages in the hub pipeline downloads the various tools needed. It downloads `jq`, `oc`, `opm` and `kubectl`. It also proceeds to verify that various hub install prerequisites exist before proceeding, for example it checks the:

- OpenShift Container Platform version.
- Nodes are ready.
- Cluster Operators are ready.
- Metal3 pods are ready.
- Persistent volumes are created.
- DNS requirements are satisfied.

Deploy HTTPD

This step deploys and configures an HTTP server on the hub cluster. It obtains the Red Hat Enterprise Linux CoreOS (RHCOS) ISO and RootFS images from mirror.openshift.com and ensures these are hosted on the deployed HTTPD server. These are then available to install on the edge cluster.

Deploy registry

This step deploys a registry on the hub cluster. The substeps involved in this process are as follows:

- Deploy the registry on the hub.
- Sync the OpenShift Container Platform and Operator Lifecycle Manager (OLM) images from Quay and Red Hat registries to the internal registry.
- Update the pull secret globally.

Deploy RHACM

This step installs the Red Hat Advanced Cluster Manager and Assisted Installer on the OpenShift Container Platform hub cluster. Red Hat Advanced Cluster Manager manages the deployment on many managed edge clusters.

Transition to disconnected

This step deploys the ImageContentSourcePolicy (ISCP) and the Catalog sources for the hub to point to itself as a source of the images and operator. From this step forward, the hub cluster is no longer connected to the Internet.

Deploy Assisted Installer

This step ensures the Assisted Installer service supports installing the edge clusters. This step configures the way the edge cluster is deployed, the certificates, the image sources, the cluster details, and so on.

At this stage, the hub cluster is ready to install the edge cluster.

The edge factory pipeline

This stage deploys and configures the edge clusters. When this pipeline is completed, the edge clusters are ready for use when the enclosure gets relocated to the end customer's remote site.

The flow associated with deploying the edge cluster is:

Check hub

This step installs the various tools on the edge cluster that are needed. It downloads `jq`, `oc`, `opm` and `kubectl`. It proceeds to verify that various hub install prerequisites exist before proceeding, for example it checks the:

- OpenShift Container Platform version.
- Nodes are ready.
- Cluster Operators are ready.
- Metal3 pods are ready.
- Persistent volumes are created.
- DNS requirements are satisfied.

Deploy edge

This step starts with the edge cluster provisioning. This process ends with pushing a notification from the edge cluster to the hub and answering with an ACK.

Deploy NMState and MetalLB

This step deploys the NMState and the MetalLB Operators. NMState creates one profile per node to obtain an IP from the external network's DHCP. Then the MetalLB creates a resource called an AddressPool to build the relationship between the internal and external interface using a LoadBalancer interface. Finally it creates a service for the API and the ingress. Without this step you will not be able to access the API or ingress by using the external address.

Deploy OpenShift Data Foundation

This step deploys the Local Storage Operator and also OpenShift Data Foundation (ODF). ODF and the Local Storage Operator uses disks defined in the `storage_disk` section of the `edgeclusters.yaml` configuration file to create persistent volumes. ODF generates the storage classes and dynamically provisions the persistent volumes. This provides the storage necessary to host the disconnected registry images (Quay).

Deploy Quay

This step deploys the Quay Operator and components of Quay, because the end customer needs a fully supported solution in the edge and the factory is expected to have their own internal registry. This Quay deployment has a small footprint enabling only the features needed to host an internal registry with basic functions.

Deploy workers

This step deploys the worker nodes, and adds these to the edge cluster.

Deploy UI

The deploy UI stage helps to simplify the configuration of the edge cluster after it is relocated to the customer’s site.

Detach cluster

This step ensures that everything is correctly configured, it sets the NodeNetworkConfigurationPolicy (NNCP), and ensures the detached edge cluster will work on site. During the edge deployment phase the `kubeconfig` and `kubeadmin` password are saved in the hub. The `SSH-RSA` gets saved in the hub and edge cluster and the newly created edge gets deleted in RHACM. This information is communicated to the end customer and used to complete the edge cluster configuration on site.

Verifying the hub cluster is ready to run the factory install pipeline

Run the following steps to ensure the hub cluster is ready to run the factory install pipeline.

Prerequisites

- An installed OpenShift Container Platform hub cluster.
- Access to the cluster as a user with the `cluster-admin` role.

Procedure

1. Verify the status of the nodes:

   ```
   $ oc get nodes
   ```

   Example output

   ```
   NAME            STATUS          ROLES                   AGE             VERSION
   test-master-0   READY           master,worker           154m
   v1.23.5+9ce5071
   test-master-1   READY           master,worker           154m
   v1.23.5+9ce5071
   test-master-2   READY           master,worker           154m
   v1.23.5+9ce5071
   ```

2. Verify the status of the Cluster Operators:

   ```
   $ oc get co
   ```

   Example output

   ```
   NAME                                        VERSION  AVAILABLE   PROGRESSING
   DEGRADED   SINCE   MESSAGE
   authentication                              4.10.9   True        False
   False      110m
   baremetal                                   4.10.9   True        False
   False      178m
   cloud-controller-manager                    4.10.9   True        False
   False      3h
   cloud-credential                            4.10.9   True        False
   False      179m
   cluster-autoscaler                          4.10.9   True        False
   False      178m
   config-operator                             4.10.9   True        False
   False      3h
   console                                     4.10.9   True        False
   ```

```
False      168m
csi-snapshot-controller                    4.10.9   True        False
False      178m
dns                                        4.10.9   True        False
False      178m
etcd                                       4.10.9   True        False
False      177m
image-registry                             4.10.9   True        False
False      172m
ingress                                    4.10.9   True        False
False      173m
insights                                   4.10.9   True        False
False      172m
kube-apiserver                             4.10.9   True        False
False      175m
kube-controller-manager                    4.10.9   True        False
False      176m
kube-scheduler                             4.10.9   True        False
False      175m
kube-storage-version-migrator              4.10.9   True        False
False      179m
machine-api                                4.10.9   True        False
False      175m
machine-approver                           4.10.9   True        False
False      179m
machine-config                             4.10.9   True        False
False      102m
marketplace                                4.10.9   True        False
False      178m
monitoring                                 4.10.9   True        False
False      93m
network                                    4.10.9   True        False
False      3h
node-tuning                                4.10.9   True        False
False      178m
openshift-apiserver                        4.10.9   True        False
False      173m
openshift-controller-manager               4.10.9   True        False
False      174m
openshift-samples                          4.10.9   True        False
False      172m
operator-lifecycle-manager                 4.10.9   True        False
False      179m
operator-lifecycle-manager-catalog         4.10.9   True        False
False      178m
operator-lifecycle-manager-packageserver   4.10.9   True        False
False      173m
service-ca                                 4.10.9   True        False
False      179m
storage                                    4.10.9   True        Flase
```

```
False      179m
```

3. Verify that enough persistent volumes exist and are available:

```
$ oc get pv
```

Example output

```
NAME    CAPACITY    ACCESS-MODES    RECLAIM POLICY      STATUS      CLAIM
STORAGECLASS  REASON    AGE
pv001   200Gi       RWO             Recycle             Available
137m
pv002   200Gi       RWO             Recycle             Available
137m
pv003   200Gi       RWO             Recycle             Available
137m
pv004   200Gi       RWO             Recycle             Available
137m
pv005   200Gi       RWO             Recycle             Available
137m
pv006   200Gi       RWO             Recycle             Available
137m
pv007   200Gi       RWO             Recycle             Available
137m
pv008   200Gi       RWO             Recycle             Available
137m
pv009   200Gi       RWO             Recycle             Available
137m
pv010   200Gi       RWO             Recycle             Available
137m
pv011   200Gi       RWX             Recycle             Available
137m
pv012   200Gi       RWX             Recycle             Available
137m
pv013   200Gi       RWX             Recycle             Available
137m
pv014   200Gi       RWX             Recycle             Available
137m
pv015   200Gi       RWX             Recycle             Available
137m
pv016   200Gi       RWX             Recycle             Available
137m
pv017   200Gi       RWX             Recycle             Available
137m
pv018   200Gi       RWX             Recycle             Available
137m
pv019   200Gi       RWX             Recycle             Available
137m
pv020   200Gi       RWX             Recycle             Available
```

```
137m
```

Installing the OpenShift Pipelines Operator

Follow this guidance to install the OpenShift Pipelines Operator that is used to run the pipeline.

Prerequisites

- An installed OpenShift Container Platform hub cluster.
- Install the OpenShift CLI (`oc`).
- Access to the cluster as a user with the `cluster-admin` role.
- Install `git`. For guidance on installing `git`, see Install Git.

Procedure

1. Export the `KUBECONFIG` environment variable:

   ```
   $ export KUBECONFIG=<path_to_kubeconfig>/kubeconfig
   ```

2. Run the following bash script `bootstrap.sh` with the `KUBECONFIG` as a parameter to install the OpenShift Pipelines Operator:

   ```
   $ curl -sLK https://raw.githubusercontent.com/rh-ecosystem-edge/ztp-pipeline-
   relocatable/main/pipelines/bootstrap.sh | bash -s -- ${KUBECONFIG}
   ```

   This script:

   - Installs the `tkn` CLI. This tool manages OpenShift Container Platform pipelines from a terminal.
   - Clones the ztp-pipeline-relocatable pipeline repository.
   - Checks that the correct permissions are set on the hub cluster.
   - Deploys the OpenShift Pipelines Operator from the Operator Lifecycle Manager (OLM) catalog.
   - Deploys the ZTP pipeline and associated tasks.

3. Monitor the progress in the terminal window and in the web console.

   a. In the terminal window you are expected to see the following:

   ```
   >>>> Creating NS edgecluster-deployer and giving permissions to SA edgecluster-
   deployer
   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   >>>>>>>
   namespace/edgecluster-deployer configured
   serviceaccount/edgecluster-deployer configured
   clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-0 configured

   >>>> Cloning Repository into your local folder
   ```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Cloning into 'ztp-pipeline-relocatable'...
remote: Enumerating objects: 3824, done.
remote: Counting objects: 100% (1581/1581), done.
remote: Compressing objects: 100% (963/963), done.
remote: Total 3824 (delta 963), reused 1163 (delta 589), pack-reused 2243
Receiving objects: 100% (3824/3824), 702.12 KiB | 8.46 MiB/s, done.
Resolving deltas: 100% (2182/2182), done.

>>>> Deploying Openshift Pipelines
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
subscription.operators.coreos.com/openshift-pipelines-operator-rh unchanged
>>>> Waiting for: Openshift Pipelines
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>> Deploying ZTPFW Pipelines and tasks
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
pipeline.tekton.dev/deploy-ztp-hub configured
pipeline.tekton.dev/deploy-ztp-edgeclusters configured
task.tekton.dev/common-pre-flight configured
task.tekton.dev/hub-deploy-acm configured
task.tekton.dev/hub-deploy-disconnected-registry configured
task.tekton.dev/hub-deploy-httpd-server configured
task.tekton.dev/hub-deploy-hub-config configured
task.tekton.dev/hub-deploy-icsp-hub configured
task.tekton.dev/hub-save-config configured
task.tekton.dev/edgecluster-deploy-disconnected-registry-edgeclusters configured
task.tekton.dev/edgecluster-deploy-icsp-edgeclusters-post configured
task.tekton.dev/edgecluster-deploy-icsp-edgeclusters-pre configured
task.tekton.dev/edgecluster-deploy-metallb configured
task.tekton.dev/edgecluster-deploy-ocs configured
task.tekton.dev/edgecluster-deploy-edgecluster configured
task.tekton.dev/edgecluster-deploy-workers configured
task.tekton.dev/edgecluster-detach-cluster configured
task.tekton.dev/edgecluster-restore-hub-config configured
```

b. Log in to the OpenShift Container Platform web console.

   i. Navigate to **Pipelines** → **Pipelines**.

   ii. Select the project **edgecluster-deployer**.

NOTE Stored in the `edgecluster-deployer` namespace are all the artifacts for the successful execution of the pipelines. Monitor the progress of the pipelines in this window.

Running the hub cluster factory install pipeline

Run the following steps to run the hub factory install pipeline.

Prerequisites

- An installed OpenShift Container Platform hub cluster.
- Access to the cluster as a user with the `cluster-admin` role.

Procedure

1. Create a file `edgeclusters.yaml` with sample details as shown. A sample configuration file is present in `examples/config.yaml`.

   NOTE At this stage you only need to build out the `config` section. The `config` section specifies the cluster configuration values used to install and configure the hub and edge cluster.

   ```
   config:
     OC_OCP_VERSION: "4.10.9" ①
     OC_ACM_VERSION: "2.4" ②
     OC_OCS_VERSION: "4.9" ③
   ```

   ① OpenShift Container Platform version of the edge cluster.

   ② Red Hat Advanced Cluster Management (RHACM) version.

   ③ The OpenShift Data Foundation (ODF) version.

2. Start the hub cluster pipeline from the command line:

   ```
   $ tkn pipeline start -n edgecluster-deployer edgeclusters-config="$(cat /path-to-
   edgecluster.yaml/edgeclusters.yaml)" -p kubeconfig=${KUBECONFIG}
   -w=ztp,claimName=ztp-pvc --timeout 5h --use-param-defaults deploy-ztp-hub
   ```

   NOTE This command starts the pipeline in the namespace `edgecluster-deployer` with the defined edge cluster configuration and the `kube` configuration in the workspace `ztp` with the previously configured persistent storage claim `ztp-pvc`. A timeout of 5 hours is set for the execution of the `deploy-ztp-hub` with all other parameters set to the default.

   Example output

   ```
   PipelineRun started: deploy-ztp-hub-run-2h44k

   In order to track the PipelineRun progress run:
   ```

```
tkn pipelinerun logs deploy-ztp-hub-run-2h44k -f -n edgecluster-deployer
```

Monitoring the progress of the hub cluster factory install pipeline

You can watch the progress of the pipeline by using the OpenShift Container Platform web console and using the deployment log file.

Procedure

1. Examine the logs to watch the progress of the `deploy-ztp-hub`:

   ```
   $ tkn pipeline logs deploy-ztp-hub-run-2h44k -f -n edgecluster-deployer
   ```

2. Log in to the OpenShift Container Platform web console.
3. Navigate to **Pipelines** → **Pipelines** and select the Project **edgecluster-deployer**.

   NOTE The `edgecluster-deployer` pipeline stores all the artifacts for OpenShift Container Platform Pipelines.

4. Select **PipelineRuns** to drill down into detail on the pipeline runs.
5. The stages of the pipeline are clearly shown and you can select each in turn to view the logs associated with that stage of the deployment.

Post hub factory pipeline verification checks

Perform the following steps after completion of the hub factory pipeline run.

Prerequisites

- An OpenShift Container Platform hub cluster.
- Log in as a user with `cluster-admin` privileges.

Procedure

1. Verify RHACM is successfully installed:

   ```
   $ oc get pod -n open-cluster-management
   ```

   Example output

   ```
   NAME                                                        READY   STATUS
   RESTART         AGE
   application-chart-ee7d2-applicastionui-7d99756554-jrs24     1/1     RUNNING    0
   6m31s
   application-chart-ee7d2-applicastionui-7d99756554-jrs24     1/1     RUNNING    0
   6m31s
   ```

```
application-chart-ee7d2-applicastionui-7d99756554-jrs24     1/1     RUNNING     0
6m31s
application-chart-ee7d2-applicastionui-7d99756554-jrs24     1/1     RUNNING     0
6m31s
assisted-image-service-67489b657b-68qtg                     1/1     RUNNING     0
2m30s
assisted-service-5b8874ffd9-rjrg                            2/2     RUNNING     1
(2m19s ago)   2m30s
```

2. Verify the HTTPD server is successfully running:

```
$ oc get pod -n default
```

Example output

```
NAME                       READY   STATUS      RESTART         AGE
httpd-5479bfd6cb-2p1d4     1/1     RUNNING     0               150m
```

3. Verify the internal registry is running:

```
$ oc get pod -n ztpfw-registry
```

Example output

```
NAME                             READY   STATUS      RESTART         AGE
ztpfw-registry-77ff664d47     1/1     RUNNING     0               151m
```

4. Review the pipeline run and verify the steps that were executed:

NOTE This shows the duration of every step and the parameters supplied to the pipeline. It also highlights any issues during the execution of the pipeline.

```
$ tkn pr describe -n edgecluster-deployer
```

Example output

```
Name:              deploy-ztp-hub-run-tjqp5
Namespace:         edgecluster-deployer
Pipeline Ref:      deploy-ztp-hub
Service Account:   pipeline
Timeout:           5h0m0s
Labels:
 tekton.dev/pipeline=deploy-ztp-hub

🌡️  Status
```

```
STARTED      DURATION     STATUS
1 week ago   21 minutes   Succeeded

▯ Resources

 No resources

▯ Params

 NAME                VALUE
 ▯ kubeconfig      /root/.kcli/clusters/test-ci/auth/kubeconfig
 ▯ edgeclusters-config   config:
  OC_OCP_VERSION: '4.10.9'
  OC_ACM_VERSION: '2.4'
  OC_OCS_VERSION: '4.9'
edgeclusters:
 ▯ ztp-container-image   quay.io/ztpfw/pipeline:latest

▯ Results

 No results

▯ Workspaces

 NAME    SUB PATH   WORKSPACE BINDING
 ▯ ztp   ---        PersistentVolumeClaim (claimName=ztp-pvc)

▯  Taskruns

 NAME                                                            TASK NAME
STARTED      DURATION     STATUS
 ▯ deploy-ztp-hub-run-tjqp5-deploy-hub-config-26pp5             deploy-hub-config
1 week ago   42 seconds   Succeeded
 ▯ deploy-ztp-hub-run-tjqp5-deploy-icsp-hub-5ctsr               deploy-icsp-hub
1 week ago   16 seconds   Succeeded
 ▯ deploy-ztp-hub-run-tjqp5-deploy-acm-76b6c                    deploy-acm
1 week ago   9 minutes    Succeeded
 ▯ deploy-ztp-hub-run-tjqp5-deploy-disconnected-registry-7b9rw   deploy-
disconnected-registry   1 week ago   11 minutes   Succeeded
 ▯ deploy-ztp-hub-run-tjqp5-deploy-httpd-server-9mfcn           deploy-httpd-
server            1 week ago   8 seconds    Succeeded
 ▯ deploy-ztp-hub-run-tjqp5-pre-flight-pk5bp                    pre-flight
1 week ago   9 seconds    Succeeded

▯▯  Skipped Tasks

 No Skipped Tasks
```

Running the edge cluster factory install pipeline

Run the following steps to run the edge factory install pipeline.

Prerequisites

- Enough DHCP IPs in the external network to hold the edge cluster.
- The following API, API-INT and ingress entries are available:
  - `api.<edge-cluster-name>.<network-domain>`
  - `api-int.<edge-cluster-name>.<network-domain>`
  - `*.apps.<edge-cluster-name>.<network-domain>`

NOTE When deploying a single-node OpenShift cluster, the `api.<edge-cluster-name>.<baseDomain>` and `*.apps.<edge-cluster-name>.<baseDomain>` must be configured with different IP addresses.

- Clean disks for the OpenShift Data Foundation Storage cluster.
- DNS Resolution between the edge and the hub API and ingress entries.
- An OpenShift Container Platform hub cluster.
- Log in as a user with `cluster-admin` privileges.

Procedure

1. Edit the `edgeclusters.yaml` with sample details as shown. A sample configuration file is present in `examples/config.yaml`.

   NOTE At this stage you are populating the `edgeclusters` section.

```
config:
  OC_OCP_VERSION: "4.10.9"
  OC_ACM_VERSION: "2.4"
  OC_OCS_VERSION: "4.9"

edgeclusters:
  - edgecluster1-name: ①
      master0: ②
        ignore_ifaces: eno1,eno2 ③
        nic_ext_dhcp: eno4 ④
        nic_int_static: eno5 ⑤
        mac_ext_dhcp: "aa:ss:dd:ee:b0:10" ⑥
        mac_int_static: "aa:ss:dd:ee:b1:10" ⑦
        bmc_url: "<url bmc>" ⑧
        bmc_user: "user-bmc" ⑨
        bmc_pass: "user-pass" ⑩
        root_disk: sda ⑪
      storage_disk: ⑫
        - sdb
        - sdc
        - sde
        - sdd
    master1:
      ignore_ifaces: eno1 eno2
      nic_ext_dhcp: eno4
      nic_int_static: eno5
      mac_ext_dhcp: "aa:ss:dd:ee:b0:11"
      mac_int_static: "aa:ss:dd:ee:b1:11"
      bmc_url: "<url bmc>"
      bmc_user: "user-bmc"
      bmc_pass: "user-pass"
      root_disk: sda
      storage_disk:
        - sdb
        - sdc
        - sde
        - sdd
    master2:
      ignore_ifaces: eno1 eno2
      nic_ext_dhcp: eno4
      nic_int_static: eno5
      mac_ext_dhcp: "aa:ss:dd:ee:b0:12"
      mac_int_static: "aa:ss:dd:ee:b1:12"
      bmc_url: "<url bmc>"
      bmc_user: "user-bmc"
      bmc_pass: "user-pass"
      root_disk: sda
      storage_disk:
        - sdb
        - sdc
        - sde
        - sdd
  worker0: ⑬
      nic_ext_dhcp: eno4
      nic_int_static: eno5
      mac_ext_dhcp: "aa:ss:dd:ee:b0:19"
      mac_int_static: "aa:ss:dd:ee:b1:19"
      bmc_url: "<url bmc>"
      bmc_user: "user-bmc"
      bmc_pass: "user-pass"
- edgecluster2-name:
    master0:
      ignore_ifaces: eno1 eno2
      nic_ext_dhcp: eno4
      nic_int_static:  eno5
      mac_ext_dhcp: "aa:ss:dd:ee:b0:20"
      mac_int_static: "aa:ss:dd:ee:b1:20"
      bmc_url: "<url bmc>"
      bmc_user: "user-bmc"
      bmc_pass: "user-pass"
      storage_disk:
        - sdb
        - sdc
        - sde
        - sdd
    master1:
      ignore_ifaces: eno1 eno2
      nic_ext_dhcp: eno4
      nic_int_static:  eno5
      mac_ext_dhcp: "aa:ss:dd:ee:b0:21"
      mac_int_static: "aa:ss:dd:ee:b1:21"
      bmc_url: "<url bmc>"
      bmc_user: "user-bmc"
      bmc_pass: "user-pass"
      storage_disk:
        - sdb
        - sdc
        - sde
        - sdd
    master2:
      ignore_ifaces: eno1 eno2
      nic_ext_dhcp: eno4
      nic_int_static:  eno5
      mac_ext_dhcp: "aa:ss:dd:ee:b0:22"
      mac_int_static: "aa:ss:dd:ee:b1:22"
      bmc_url: "<url bmc>"
      bmc_user: "user-bmc"
      bmc_pass: "user-pass"
      storage_disk:
        - sdb
        - sdc
        - sde
        - sdd
    worker0:
      nic_ext_dhcp: eno4
      nic_int_static:  eno5
      mac_ext_dhcp: "aa:ss:dd:ee:b0:29"
      mac_int_static: "aa:ss:dd:ee:b1:29"
      bmc_url: "<url bmc>"
      bmc_user: "user-bmc"
      bmc_pass: "user-pass"
```

① This option is configurable and sets the name of the edge cluster.

② This value must match `master0`, `master1` or `master2`.

③ Optional: Interfaces to ignore in the host.

④ NIC connected to the external DHCP.

⑤ NIC connected to the internal network (This interface is optional).

⑥ MAC address for the NIC connected to the external DHCP network.

⑦ MAC address for the NIC connected to the internal network (This MAC address is optional if we're using only 1 interface nic in <5>).

⑧ URL for the Baseboard Management Controller (BMC).

⑨ The BMC username.

⑩ The BMC password.

⑪ Mandatory: Disk device to be used for operating system installation.

⑫ List of disk available in the node to be used for storage.

⑬ Hardcoded name set as `worker0` for the worker node.

2. Set the following environment variable:

```
$ export KUBECONFIG=<path_to_kubeconfig>/kubeconfig-file
```

3. Start the edge cluster pipeline from the command line:

```
$ tkn pipeline start -n edgecluster-deployer edgeclusters-config="$(cat /path-to-
edgecluster-yaml/edgeclusters.yaml)" -p kubeconfig=${KUBECONFIG}
-w=ztp,claimName=ztp-pvc --timeout 5h --use-param-defaults deploy-ztp-edgeclusters
```

NOTE This command starts the pipeline in the namespace `edgecluster-deployer` with the defined configuration and the `kube` configuration in the workspace ztp with the previously configured persistent storage claim `ztp-pvc`. A timeout of 5 hours is set for the execution of the `deploy-ztp-hub` with all other parameters set to the default.

Example output

```
PipelineRun started: deploy-ztp-edgecluster-run-2rklt

In order to track the PipelineRun progress run:
tkn pipeline logs deploy-ztp-edgecluster-run-2rklt -f -n edgecluster-deployer
```

Monitoring the progress of the edge cluster factory install pipeline

You can watch the progress of the pipelines by using the OpenShift Container Platform web console and by using the deployment log file.

Procedure

1. Examine the logs to watch the progress of the `deploy-ztp-edgeclusters`.

```
$ tkn pipeline logs deploy-ztp-edgecluster-run-2rklt -f -n edgecluster-deployer
```

2. Log in to the OpenShift Container Platform web console.
3. Navigate to **Pipelines** → **Pipelines** and select the Project **edgecluster-deployer**.

NOTE The `edgecluster-deployer` pipeline stores all the artefacts for OpenShift Container Platform Pipelines.

4. Select **PipelineRuns** to drill down into the details of the pipeline runs.
5. The stages of the pipeline are clearly shown and you can select each in turn to view the logs associated with that stage of the deployment.

Post edge cluster factory pipeline verification checks

Perform the following steps after completion of the edge cluster factory pipeline run.

Prerequisites

- An OpenShift Container Platform hub cluster.
- Log in as a user with `cluster-admin` privileges.

Procedure

1. Verify MetalLB is successfully installed:

```
$ oc get addresspool -A
```

Example output

```
NAMESPACE        NAME                AGE
metallb          api-public-ip       10m
metallb          ingress-public-ip   10m
```

2. Confirm that the `NodeNetworkConfigurationPolicy` has been applied to the cluster:

```
$ oc get nncp -A
```

Example output

```
NAME                                         STATUS
kubeframe-edgecluster-0-master-0-nccp              Available
kubeframe-edgecluster-0-master-1-nccp              Available
kubeframe-edgecluster-0-master-2-nccp              Available
```

3. Verify the internal registry is running:

```
$ oc get pod -n ztpfw-registry
```

Expected output

```
NAME                          READY   STATUS     RESTART        AGE
ztpfw-registry-77ff664d47     1/1     RUNNING    0              151m
```

4. Run the following command to review the pipeline run and verify the steps that were executed:

NOTE This shows the duration of every step, the parameters supplied to the pipeline. It also highlights any issues during the execution of the pipeline.

```
$ tkn pr describe deploy-ztp-edgecluster-run-2rklt -n edgecluster-deployer
```

Troubleshooting a pipeline run

Perform the following steps to debug a pipeline run.

Procedure

1. Export the `KUBECONFIG` as follows:

   ```
   $ export KUBECONFIG=<path_to_kubeconfig>/kubeconfig
   ```

2. List the executed pipeline runs:

   ```
   $ tkn pr ls -A
   ```

   Example output

   ```
   NAMESPACE        NAME                              STARTED       DURATION     STATUS
   edgecluster-deployer   deploy-ztp-edgeclusters-run-sp8hm   1 hour ago    1 hour
   Cancelled(PipelineRunCancelled)
   edgecluster-deployer   deploy-ztp-hub-run-rwh4j      2 hours ago   35 minutes
   Succeeded
   edgecluster-deployer   deploy-ztp-hub-run-vgwz6      3 hours ago   2 minutes
   Failed
   ```

3. Run the following command against the failed pipeline run name and identify the failed task:

   ```
   $ tkn pr describe deploy-ztp-hub-run-vgwz6 -n edgecluster-deployer
   ```

   Example output

   ```
   Name:              deploy-ztp-hub-run-vgwz6
   Namespace:         edgecluster-deployer
   Pipeline Ref:      deploy-ztp-hub
   Service Account:   pipeline
   Timeout:           5h0m0s
   Labels:
    tekton.dev/pipeline=deploy-ztp-hub

   □□  Status

   STARTED       DURATION    STATUS
   3 hours ago   2 minutes   Failed

   □ Message

   Tasks Completed: 3 (Failed: 1, Cancelled 0), Skipped: 3 ("step-mirror-olm" exited
   with code 255 (image:
   ```

```
"quay.io/ztpfw/pipeline@sha256:d86d567f0ee76efdd5ea168fac3cbd5e8e7e479ddcea0be6aaf9
e890de9566b3"); for logs run: kubectl -n edgecluster-deployer logs deploy-ztp-hub-
run-vgwz6-deploy-disconnected-registry-xqz-kltxr -c step-mirror-olm
)

 Resources

 No resources

 Params

 NAME              VALUE
  kubeconfig      /root/.kcli/clusters/test-ci/auth/kubeconfig
  edgeclusters-config   config:
  OC_OCP_VERSION: '4.10.9'
  OC_ACM_VERSION: '2.4'
  OC_OCS_VERSION: '4.9'
edgeclusters:
  ztp-container-image   quay.io/ztpfw/pipeline:latest

 Results

 No results

 Workspaces

 NAME    SUB PATH   WORKSPACE BINDING
  ztp   ---        PersistentVolumeClaim (claimName=ztp-pvc)

  Taskruns

 NAME                                                            TASK NAME
STARTED       DURATION     STATUS
  deploy-ztp-hub-run-vgwz6-deploy-disconnected-registry-xqzz5   deploy-
disconnected-registry   3 hours ago   4 minutes    Failed
  deploy-ztp-hub-run-vgwz6-deploy-httpd-server-6n47b            deploy-httpd-
server            3 hours ago   56 seconds   Succeeded
  deploy-ztp-hub-run-vgwz6-pre-flight-slvkv                     pre-flight
3 hours ago   36 seconds   Succeeded

  Skipped Tasks

 NAME
  deploy-acm
  deploy-icsp-hub
  deploy-hub-config
```

4. Run the following command against the failed `taskrun` name to find the reason for the failure:

```
$ tkn tr describe deploy-ztp-hub-run-vgwz6-deploy-disconnected-registry-xqzz5 -n
```

```
edgecluster-deployer
```

Example output

```
Name:              deploy-ztp-hub-run-vgwz6-deploy-disconnected-registry-xqzz5
Namespace:         edgecluster-deployer
Task Ref:          hub-deploy-disconnected-registry
Service Account:   pipeline
Timeout:           5h0m0s
Labels:
 app.kubernetes.io/managed-by=tekton-pipelines
 tekton.dev/memberOf=tasks
 tekton.dev/pipeline=deploy-ztp-hub
 tekton.dev/pipelineRun=deploy-ztp-hub-run-vgwz6
 tekton.dev/pipelineTask=deploy-disconnected-registry
 tekton.dev/task=hub-deploy-disconnected-registry

▯▯  Status

STARTED       DURATION    STATUS
3 hours ago   4 minutes   Failed

Message

"step-mirror-olm" exited with code 255 (image:
"quay.io/ztpfw/pipeline@sha256:d86d567f0ee76efdd5ea168fac3cbd5e8e7e479ddcea0be6aaf9
e890de9566b3"); for logs run: kubectl -n edgecluster-deployer logs deploy-ztp-hub-
run-vgwz6-deploy-disconnected-registry-xqz-kltxr -c step-mirror-olm


▯ Input Resources

 No input resources

▯ Output Resources

 No output resources

▯ Params

 NAME               VALUE
 ▯ edgeclusters-config   config:
  OC_OCP_VERSION: '4.10.9'
  OC_ACM_VERSION: '2.4'
  OC_OCS_VERSION: '4.9'
edgeclusters:
 ▯ kubeconfig            /root/.kcli/clusters/test-ci/auth/kubeconfig
 ▯ ztp-container-image   quay.io/ztpfw/pipeline:latest
 ▯ mock                  false
```

```
□ Results

 No results

□ Workspaces

 NAME     SUB PATH   WORKSPACE BINDING
 □ ztp    ---        PersistentVolumeClaim (claimName=ztp-pvc)

□ Steps

 NAME                              STATUS
 □ update-global-pullsecret        Error
 □ deploy-disconnected-registry    Completed
 □ mirror-ocp                      Completed
 □ mirror-olm                      Error

□ Sidecars

No sidecars
```

5. Debug a task execution from the container in the cluster as follows:

    a. Get all pods in the `edgecluster-deployer` namespace:

    ```
    $ oc get pod -n edgecluster-deployer
    ```

    Example output

    ```
    NAME                                                               READY   STATUS
    RESTARTS   AGE
    deploy-ztp-hub-run-rwh4j-deploy-acm-k92kf-pod-85n7t                0/1
    Completed   0           159m
    deploy-ztp-hub-run-rwh4j-deploy-disconnected-registry-8j9-rk469    0/4
    Completed   0           3h2m
    deploy-ztp-hub-run-rwh4j-deploy-httpd-server-fw49r-pod-lhkxf       0/1
    Completed   0           3h2m
    deploy-ztp-hub-run-rwh4j-deploy-hub-config-vmgf2-pod-cjg72         0/1
    Completed   0           149m
    deploy-ztp-hub-run-rwh4j-deploy-icsp-hub-c7tg7-pod-ntmqp           0/1
    Completed   0           149m
    deploy-ztp-hub-run-rwh4j-pre-flight-865p2-pod-6wmj4                0/1
    Completed   0           3h3m
    deploy-ztp-edgeclusters-run-sp8hm-deploy-icsp-edgeclusters-pre-76thd--2pg7t
    0/1      Completed   0           97m
    deploy-ztp-edgeclusters-run-sp8hm-deploy-metallb-d7cnj-pod-rmbcg         0/1
    Completed   0           94m
    deploy-ztp-edgeclusters-run-sp8hm-deploy-ocs-k7hf9-pod-7rwwq             0/1
    Completed   0           92m
    deploy-ztp-edgeclusters-run-sp8hm-deploy-edgeclusters-pmbnz-pod-kp5fc
    ```

```
0/2     Completed   0          123m
deploy-ztp-edgeclusters-run-sp8hm-pre-flight-zwdsn-pod-l2v7h            0/1
Completed   0          123m
edgecluster-deploy-disconnected-registry-edgeclusters-run-t6k2d-pod-cnm5t
4/4     NotReady    0          34s
```

b. Log in to the pod in `NotReady` state:

```
$ oc debug pod/edgecluster-deploy-disconnected-registry-edgeclusters-run-t6k2d-
pod-cnm5t -n edgecluster-deployer
```

Example output

```
Defaulting container name to step-deploy-disconnected-registry.
Use 'oc describe pod/edgecluster-deploy-disconnected-registry-edgeclusters-run-
t6k2d-pod-cnm5t-debug -n edgecluster-deployer' to see all of the containers in
this pod.

Starting pod/edgecluster-deploy-disconnected-registry-edgeclusters-run-t6k2d-
pod-cnm5t-debug, command was: /tekton/tools/entrypoint -wait_file
/tekton/downward/ready -wait_file_content -post_file /tekton/tools/0
-termination_path /tekton/termination -step_metadata_dir /tekton/steps/step-
deploy-disconnected-registry -step_metadata_dir_link /tekton/steps/0 -docker
-cfg=pipeline-dockercfg-t6ccl -entrypoint /tekton/scripts/script-0-mm64m --
Pod IP: 10.134.0.53
If you don't see a command prompt, try pressing enter.
sh-4.4#
```

Common and expected errors

A common issue that may occur during the ZTP pipelines run is a failure during the check hub stage.

Another expected error occurs during the running of **deploy registry** stage of the hub cluster pipeline `kubelet` is restarted and access to the Kubernetes API is temporarily interrupted. This is normal behavior and an error message similar to the following is printed.

```
[deploy-disconnected-registry : deploy-disconnected-registry]
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
[deploy-disconnected-registry : deploy-disconnected-registry] Creating
/workspace/ztp/build/edgeclusters.yaml from SPOKES_CONFIG
[deploy-disconnected-registry : deploy-disconnected-registry] Waiting for deployment
of ztpfw-registry in namespace ztpfw-registry with a timeout 1000 seconds
[deploy-disconnected-registry : deploy-disconnected-registry] Expected generation for
deployment ztpfw-registry: 1
[deploy-disconnected-registry : deploy-disconnected-registry] Observed expected
generation: 1
[deploy-disconnected-registry : deploy-disconnected-registry] Specified replicas: 1
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry]
current/updated/available replicas: 1/1/, waiting
[deploy-disconnected-registry : deploy-disconnected-registry] Deployment ztpfw-
registry successful. All 1 replicas are ready.
[deploy-disconnected-registry : deploy-disconnected-registry]
machineconfig.machineconfiguration.openshift.io/update-localregistry-ca-certs created
```

```
[deploy-disconnected-registry : deploy-disconnected-registry] Mode: hub
[deploy-disconnected-registry : deploy-disconnected-registry] >> Waiting for the MCO
to grab the new MachineConfig for the certificate...

failed to get logs for task deploy-disconnected-registry : error in getting logs for
step mirror-ocp: error getting logs for pod deploy-ztp-hub-run-wt5kr-deploy-
disconnected-registry-kxm-585tz(step-mirror-ocp) : Get
"https://192.168.150.190:10250/containerLogs/edgecluster-deployer/deploy-ztp-hub-run-
wt5kr-deploy-disconnected-registry-kxm-585tz/step-mirror-ocp?follow=true": dial tcp
192.168.150.190:10250: connect: connection refused
failed to get logs for task deploy-disconnected-registry : error in getting logs for
step mirror-olm: error getting logs for pod deploy-ztp-hub-run-wt5kr-deploy-
disconnected-registry-kxm-585tz(step-mirror-olm) : Get
"https://192.168.150.190:10250/containerLogs/edgecluster-deployer/deploy-ztp-hub-run-
wt5kr-deploy-disconnected-registry-kxm-585tz/step-mirror-olm?follow=true": dial tcp
192.168.150.190:10250: connect: connection refused
```

Configuring the edge cluster at the remote location

Configure the edge cluster by using the custom user interface.

NOTE Some of the commands need `root` access to run. You can either log in as the `root` user and proceed with the step or add `sudo` before every command.

Prerequisites

- `kubeadmin` password as supplied by vendor.
- URL of the custom user interface.

Procedure

1. Unbox and turn on the cluster.
2. Log in at the command line to `master-0` of the cluster:
3. Configure DNS on `master-0` of the hub cluster:
   a. Edit `resolv.conf` on `master-0` and add the IP address of `master-0`.

   ```
   $ vi /etc/resolv.conf
   ```

   b. Add the line.

   ```
   nameserver 192.168.7.10
   ```

4. Configure a static IP on the connected laptop:
   a. Determine the name of the laptop’s network interface card (NIC) as follows.

   ```
   $ ip addr
   ```

   NOTE Look for the NIC name starting with the letter `e`.

   b. Edit the NIC’s network configuration.

   ```
   $ vi /etc/sysconfig/network-scripts/ifcfg-eth0
   ```

   NOTE Here `eth0` is the network card name, and it can be different for different computers.

   c. Add or modify the configuration below:

```
BOOTPROTO=static
IPADDR=192.168.7.21
NETMASK=255.255.255.0
GATEWAY=192.168.0.1
DNS1=8.8.8.8
DNS2=8.8.4.4
```

> **NOTE** Use any IP in the range `192.168.7.20 - 192.168.7.150`.

5. Restart the network services:

```
$ systemctl restart NetworkManager
```

6. Open a browser and log in to the edge cluster configuration user interface at the following URL with the supplied `kubeadmin` `username` and `password`:

```
https://edge-cluster-setup.example-edge-cluster.domain.com
```

> **NOTE** This `kubeadmin` username and password was created at the factory and should have been supplied to you. Only one user is initially created.

7. Click **Continue**.
8. Step through the screens to complete the initial setup.
   a. In the first two screens create a new user account by entering a `username` and `password` when prompted.

   > **NOTE** This new user account is granted `cluster-admin` privileges and should be used rather than the factory created `kubeadmin` account.

   b. In the **API** screen assign the IP address that will be used for API traffic. A default value is assigned but you are free to update this.
   c. In the **Ingress** screen assign the IP address that will be used for new routes and traffic managed by the ingress controller. A default value is assigned but you are free to update this.
   d. Optional: Under Domain create unique URLs for the setup and console URLs for your edge cluster.
   e. Click **Download** in the **SSH** screen and download the edge cluster private SSH key.

   > **NOTE** You need this to access the nodes of the edge cluster.

   f. Click **Finish setup**.
   g. Selecting **OpenShift console** brings you direct to the web console.

h. Under **Settings** you have the option to change the values of the **API address**, **Ingress address** and the previously configured **Domain**.

NOTE At this stage you will not be prompted for a username and password as you are already logged in as `kubeadmin`.

i. Click **Log out** in the top right hand corner.

9. Log in to the cluster again.
10. Select the newly created identity provider `ztpfw-htpasswd-idp`.
11. In the cluster log in screen enter the `username` and `password` created in step 8a.
12. After you access the cluster, register your cluster subscription with the following steps:
    a. Log in to the console to register the disconnected OpenShift cluster. See How to register disconnected OpenShift Container Platform 4 cluster on cloud.redhat.com for details.
    b. Obtain the pull secret from Pull secret, which can be found under `Tokens`.
    c. Change the global pull secret. Follow the guidance in How to change the global pull secret in OCP 4 to do that.

Your cluster is now registered to Red Hat OpenShift Cluster Manager and entitled to Red Hat subscription management.

ZTP factory install pipelines flags and arguments

The pipeline arguments and flags are described in the following tables.

Table 2. Pipeline flags

Flag	Description
`-n`	OpenShift Container Platform namespace where the resources are located.
`-p`	Pipeline parameter.
`--timeout`	Pipeline general timeout.
`--use-param-defaults`	Specifies that apart from the parameters provided, the remainder use the default options.
`-w`	The workspace parameter sets where OpenShift Container Platform pipelines hold the files during every step. Do not use EmptyDir. The best choice is `name=ztp,claimName=ztp-pvc`. The persistent volume claim is created during the `bootstrap.sh` execution. It does not need more than 5Gb.

Table 3. Pipeline arguments

Flag	Description	Required
`Namespace`	This is a namespace where all the Tasks and Pipelines will be deployed.	Yes
`edgeclusters-config`	This `edgeclusters.yaml` file has the configuration for all the clusters you want to deploy at the same time. Run it with the `cat` command.	Yes
`kubeconfig`	This is the hub `kubeconfig` that is used during the pipeline execution. You can point to the file or just use the `KUBECONFIG` variable.	Yes
`-w name=ztp,claimName=ztp-pvc`	It is mandatory to use this argument exactly as it's shown here to have a successful run. This declaration instructs Tekton to use the workspace `ztp` and that the content should be placed in the `ztp-pvc` persistent volume.	Yes
`Pipeline Name`	In the command examples, this is the last argument. This flag instructs Tekton to run the pipeline with the particular name. You can examine the executed pipelines and tasks with `tkn pr ls` and `tkn task ls` respectively.	Yes

Development

NOTE

This documentation it’s mostly for the developers/qes etc… working in the project.

Deploying the environment in Virtual

This is a very expensive option to work with all nodes in virtual, which means, you will need a big boy to make this work:

Hardware requirements

Hardware Reqs for the Hub (3 Nodes):

- CPUs: 48 (16 each)
- RAM: 54 Gbs (18 each)
- Storage: 300 Gbs (each)

Hardware Reqs for the Edge Cluster (3 Master + 1 Worker Nodes):

Master Nodes:

- CPUs: 72 (24 each)
- RAM: 192 (64 each)
- Storage: 4 extra disks with 200Gb each one

Worker Node: - CPUs: 12 - RAM: 16 - Storage: 4 extra disks with 200Gb each one

Software requirements

- Libvirtd/Qemu/KVM
- Kcli for the scripts.
- Some binaries oc, kubectl, tkn, yq, jq and ketall (for debugging)

Deploying the Base Hub

Deploys the Hub cluster with an NFS as a Base Storage for the requirements

```
git clone git@github.com:rh-ecosystem-edge/ztp-pipeline-relocatable.git
cd ztp-pipeline-relocatable/hack/deploy-hub-local
./build-hub.sh ${HOME}/openshift_pull.json 1
```

Bootstraping OpenShift Pipelines

Installs the necessary things to start executing the Pipelines

```
export KUBECONFIG=/root/.kcli/clusters/test-ci/auth/kubeconfig
curl -sLk https://raw.githubusercontent.com/rh-ecosystem-edge/ztp-pipeline-
relocatable/main/pipelines/bootstrap.sh | bash -s
```

Executing the Hub Pipeline

You can customize the parameter `git-revision=<BRANCH>` to point to your own branch

```
export KUBECONFIG=/root/.kcli/clusters/test-ci/auth/kubeconfig
tkn pipeline start -n edgecluster-deployer -p ztp-container-
image="quay.io/ztpfw/pipeline:main" -p edgeclusters-config="$(cat /root/amorgant/ztp-
pipeline-relocatable/hack/deploy-hub-local/edgeclusters.yaml)" -p
kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param
-defaults deploy-ztp-hub
```

Creating the Edge Cluster VMs

Creates 4 VMs and the proper DNS entries for the involved network

```
./build-edgecluster.sh ${HOME}/openshift_pull.json 1
```

Executing the Edge Cluster Pipeline

You can customize the parameter `git-revision=<BRANCH>` to point to your own branch

```
export KUBECONFIG=/root/.kcli/clusters/test-ci/auth/kubeconfig
tkn pipeline start -n edgecluster-deployer -p ztp-container-
image="quay.io/ztpfw/pipeline:main" -p edgeclusters-config="$(cat /root/amorgant/ztp-
pipeline-relocatable/hack/deploy-hub-local/edgeclusters.yaml)" -p
kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param
-defaults deploy-ztp-edgeclusters
```

Build Images

You will need first access to the Quay Organization called **ZTPFW**, just ask whoever people involved in the project.

You have some targets already in the Makefile, and today you just need to execute:

```
make
```

this will change in the future to add functionality to the Image building

Executing a Pipeline Step

Imagine you have an environment already deployed and you need to test the step you are working on, think on for example the UI. For that you just need to:

- First step you updates the code in the PVC (This can change in the futurte when we embed the code in the Container Image)

```
tkn task start -n edgecluster-deployer -p git-revision=<YOUR BRANCH> -p edgeclusters-config="$(cat /root/jparrill/ztp-pipeline-relocatable/hack/deploy-hub-local/edgeclusters.yaml)" -p kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param-defaults fetch-from-git
```

- This second one executes the Pipeline Step

```
tkn task start -n edgecluster-deployer -p git-revision=<YOUR BRANCH> -p edgeclusters-config="$(cat /root/jparrill/ztp-pipeline-relocatable/hack/deploy-hub-local/edgeclusters.yaml)" -p kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param-defaults edgecluster-deploy-ui
```