ZTP for Factory Workflow

Overview

IMPORTANT The ZTP for Factory Workflow images and code described in this document are for **Developer Preview** purposes and are **not supported** by Red Hat at this time.

ZTP for Factory Workflow provides a way for installing on top of OpenShift Container Platform the required pieces that will enable it to be used as a disconnected Hub Cluster and able to deploy Spoke Clusters that will be configured as the last step of the installation as disconnected too.

This repository contains the scripts and OpenShift Pipelines definitions used to configure a provided OpenShift cluster (reachable via `KUBECONFIG`) for use with the ZTP for Factory.

The pipeline will then cover several aspects:

- Create required components(ACM, Registry, etc…)
- Deploy and mirror a Registry with all required images and operators
- Configure ACM to provision the spokes (based on the `spokes.yaml` file) cluster and deploy all required components on it.
- Deploy Advanced Cluster Management (ACM) components
- etc…

The pipeline has two parts:

- One that deploys the HUB cluster configuration (based on existing requirements, like OCP deployed with ODF and volumes created)
- Another that deploys Spoke clusters based on the configuration `spokes.yaml` file using a HUB cluster configured with the previous pipeline.

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

Prerequisites

Installer-provisioned installation of OpenShift Container Platform requires:

- OpenShift Cluster with 3 masters
  1. All Cluster Operators in good health status
  2. Cluster reachable via a `KUBECONFIG` file
- PVC's defined for the HUB
- DNS entries configured
  1. `httpd-server.apps.CLUSTER.DOMAIN`
  2. `kubeframe-registry-kubeframe-registry.apps.CLUSTER.DOMAIN`
- `spokes.yaml` file with the configuration for the spokes
- A set of systems for Spoke usage with 3 masters and 1 worker

Of course, the requirements for the installation of OpenShift Container Platform are also to be satisfied on the hardware involved in the installation.

The Spokes YAML file

The `spokes.yaml` file contains all the configuration information required about the setup.

There's an example in the repo at https://raw.githubusercontent.com/rh-ecosystem-edge/ztp-pipeline-relocatable/main/examples/config.yaml

As you can check, it has two major sections `config` and `spokes` that will be explained in the next section.

Just keep in mind that the spokes section, can contain several `spoke-name` entries, one per spoke cluster to be deployed by the workflow.

Spokes.yaml walktrough

Check next table for a commented configuration file with links to the explanation to each relevant file section and configuration value.

```
config:
  clusterimageset: openshift-v4.9.0
  OC_OCP_VERSION: "4.9"
  OC_OCP_TAG: "4.9.0-x86_64"
  OC_RHCOS_RELEASE: "49.84.202110081407-0"
  OC_ACM_VERSION: "2.4"
  OC_OCS_VERSION: "4.8"

spokes:
  - spoke1-name:
      master0:
        nic_ext_dhcp: eno4
        nic_int_static: eno5
        mac_ext_dhcp: "aa:ss:dd:ee:b0:10"
        mac_int_static: "aa:ss:dd:ee:b1:10"
        bmc_url: "<url bmc>"
        bmc_user: "user-bmc"
        bmc_pass: "user-pass"
        storage_disk:
          - sdb
          - sdc
          - sde
          - sdd
      master1:
        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"
        storage_disk:
          - sdb
          - sdc
          - sde
          - sdd
      master2:
        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"
        storage_disk:
          - sdb
          - sdc
          - sde
          - sdd
    worker0:
        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"
  - spoke2-name:
      master0:
        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:
      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:
      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"
```

Table 1. Required parameters

Parameter/Section	Description
`config`	This section marks the cluster configuration values that will be used for installation or configuration in both Hub and Spokes.
`clusterimageset`	This setting defines the Cluster Image Set used for the HUB and the Spokes
`OC_OCP_VERSION`	Defines the OpenShift version to be used for the installation.
`OC_OCP_TAG`	This setting defines version tag to use
`OC_RHCOS_RELEASE`	This is the release to be used
`OC_ACM_VERSION`	Specifies which ACM version should be used for the deployment
`OC_OCS_VERSION`	This defines the OCS version to be used
`spokes`	This section is the one containing the configuration for each one of the Spoke Clusters
`spokename`	This option is configurable and will be the name to be used for the spoke cluster
`mastername`	This value must match `master0`, `master1` or `master2`.
`nic_ext_dhcp`	NIC connected to the external DHCP
`nic_int_static`	NIC interface name connected to the internal network
`mac_ext_dhcp`	MAC Address for the NIC connected to the external DHCP network
`mac_int_static`	MAC Address for the NIC connected to the internal static network
`bmc_url`	URL for the Baseboard Management Controller
`bmc_user`	Username for the BMC
`bmc_pass`	Password for the BMC
`storage_disk`	List of disk available in the node to be used for storage
`workername`	Hardcoded name as `worker0` for the worker node

The workflow

OpenShift Pipelines installation

First, we need to install OpenShift Pipelines Operator that will be used for running the pipeline, this is achieved by using a bootstrapping script that will install the Operator and the CR to initiate the deployment.

This script, will also create the required pipeline definitions and tasks.

Bootstrapping OpenShift Pipelines

- Execute the bootstrap script file `pipelines/bootstrap.sh ${KUBECONFIG}` you can do that using this command:

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

- An output similar to this one, will be shown:

```
>>>> Creating NS spoke-deployer and giving permissions to SA spoke-deployer
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>
namespace/spoke-deployer configured
serviceaccount/spoke-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 Kubeframe Pipelines and tasks
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
pipeline.tekton.dev/deploy-ztp-hub configured
pipeline.tekton.dev/deploy-ztp-spokes 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/spoke-deploy-disconnected-registry-spokes configured
task.tekton.dev/spoke-deploy-icsp-spokes-post configured
task.tekton.dev/spoke-deploy-icsp-spokes-pre configured
task.tekton.dev/spoke-deploy-metallb configured
task.tekton.dev/spoke-deploy-ocs configured
task.tekton.dev/spoke-deploy-spoke configured
task.tekton.dev/spoke-deploy-workers configured
task.tekton.dev/spoke-detach-cluster configured
task.tekton.dev/spoke-restore-hub-config configured
```

This script has deployed Openshift-Pipelines and enabled the Tasks and Pipelines into the Hub cluster. We can now continue the flow using the command line or the UI to interact with OpenShift Pipelines, so it is recommended to install the Tekton CLI `tkn` to interact with OpenShift Pipelines from this link.

Hub pipeline

The Hub pipeline is the pipeline that will be used to deploy the infrastructure for the HUB to be ready to deploy spokes:

```
tkn pipeline start -n spoke-deployer -p git-revision=main -p spokes-config="$(cat
/root/amorgant/ztp-pipeline-relocatable/hack/deploy-hub-local/spokes.yaml)" -p
kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param
-defaults deploy-ztp-hub
```

Spoke pipeline

```
tkn pipeline start -n spoke-deployer -p git-revision=tekton -p spokes-config="$(cat
/root/jparrill/ztp-pipeline-relocatable/hack/deploy-hub-local/spokes.yaml)" -p
kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param
-defaults deploy-ztp-spokes
```

Post-Installation Configuration

After successfully deploying an installer-provisioned cluster, consider the following post-installation procedures.

Troubleshooting

Troubleshooting the installer workflow

This process doesn’t fail… if it does.. you can keep the broken pieces!

```
tkn XXXXX
```

```
[root@flaper87-baremetal02 ~]# oc get pod -n spoke-deployer
NAME READY STATUS RESTARTS AGE
deploy-ztp-hub-run-96tnl-deploy-disconnected-registry-4m2-5ts85 2/4 NotReady 0 6m32s
deploy-ztp-hub-run-96tnl-deploy-httpd-server-rlrwq-pod-wsh5k 0/1 Completed 0 6m41s
deploy-ztp-hub-run-96tnl-fetch-from-git-zl7m5-pod-fck69 0/1 Completed 0 6m59s
deploy-ztp-hub-run-96tnl-pre-flight-rgdtr-pod-2gmh6 0/1 Completed 0 6m50s
```

```
[root@flaper87-baremetal02 ~]# oc debug pod/deploy-ztp-hub-run-96tnl-deploy-
disconnected-registry-4m2-5ts85 -n spoke-deployer
Defaulting container name to step-deploy-disconnected-registry.
Use 'oc describe pod/deploy-ztp-hub-run-96tnl-deploy-disconnected-registry-4m2-5ts85-
debug -n spoke-deployer' to see all of the containers in this pod.

Starting pod/deploy-ztp-hub-run-96tnl-deploy-disconnected-registry-4m2-5ts85-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-w6xlw
-entrypoint /tekton/scripts/script-0-x6mfw --
Pod IP: 10.134.0.60
If you don't see a command prompt, try pressing enter.
sh-4.4# cd /workspace/ztp/
```

Lab testing

This section will cover two aspects of the testing, the HUB and the Spokes.

The scripts described here are just used for testing in a laboratory and leverage the use of KCLI tool for vm creation and DNS setup on a `libvirt`-capable machine.

Hub

We internally use the files in the folder `hack/deploy` to virtually setup an environment to test the pipeline.

The first step is the `build-hub.sh` which builds the lab deployment with all the requirements (hub, DNS, PVC's, `spokes.yaml`, etc.) that will be later used with OpenShift Pipelines to perform all the tests.

Spokes

For the spokes we use the script `build-spokes.sh`

Usage

Pipeline execution Hub

```
tkn XXXXX
```

Pipeline execution Spoke

Monitoring

You can follow the pipeline execution via XXXXX