Installing Kasten K10 on Openshift

Validated with K10 Version 5.0.8

The K10 data management platform provides enterprise operations teams an easy-to-use, scalable, and secure system for backup/restore, disaster recovery, and mobility of Red Hat OpenShift applications.

K10's application-centric approach and deep integrations with relational and NoSQL databases, Kubernetes distributions, Red Hat OpenShift versions, and all clouds provides teams the freedom of infrastructure choice without sacrificing operational simplicity. Policy-driven and extensible, K10 provides a native Kubernetes API and includes features such as full-spectrum consistency, database integrations, automatic application discovery, multi-cloud mobility, and a powerful web-based user interface.

Pre-Requisites

- Helm 3.x+
- Red Hat Openshift 4.6+
- Red Hat Advanced Cluster Management 4.11
- Nutanix CSI 2.5.x+
- Nutanix AOS 5.x+
- kubectl and oc client

Install Common Kasten K10 Pre-Reqs (per Cluster)

https://docs.kasten.io/latest/install/requirements.html#install-prereqs

- Create Default Nutanix CSI Driver VolumeSnapshotClass and Set as Default
- Set Kasten Annotations on Nutanix CSI VolumeSnapshotClass
- Setup Helm Repo
- Run Kasten K10 Pre-Requisites Validation Script

```
## 1. Create Default Nutanix CSI Driver VolumeSnapshotClass and Set as
Default
SECRET=$(kubectl get sc -o=jsonpath='{.items[?
(@.metadata.annotations.storageclass\.kubernetes\.io\/is-default-
class=="true")].parameters.csi\.storage\.k8s\.io\/provisioner-secret-
name}')

cat <<EOF | kubectl apply -f -
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: default-snapshotclass
driver: csi.nutanix.com
parameters:
  storageType: NutanixVolumes
  csi.storage.k8s.io/snapshotter-secret-name: $SECRET
  csi.storage.k8s.io/snapshotter-secret-namespace: kube-system
```

```
deletionPolicy: Delete
EOF

## 2. Set Kasten Annotations on Nutanix CSI VolumeSnapshotClass
kubectl annotate volumesnapshotclass default-snapshotclass
k10.kasten.io/is-snapshot-class=true

## 3. Add Kasten Helm Repo as it's required for pre-check
helm repo add kasten https://charts.kasten.io/
helm repo update

## 4. Run Kasten K10 Pre-Requisites Validation Script
curl https://docs.kasten.io/tools/k10_primer.sh | bash
```

Deploy Kasten k10 on ALL Openshift clusters (via Helm)

Kasten k10 will be deployed on all clusters (including Redhat ACM Hub) initially and subsequently reconfigured to support k10 Multi-Cluster Manager.

- Create Kasten Namespace
- Execute Helm Install and provide Openshift specific Options to configure SecurityContextConstraints for K10 ServiceAccounts, tokenAuth and Openshift Routes
- Verify Install
- Get Route Info and Navigate to k10 Application via Browser
- Generate Openshift OAuth Access Token to Access k10 Application via UI

> Repeat all steps for each cluster

```
## 1. Create Kasten Namespace
kubectl create ns kasten-io

## 2. Execute Helm Install and provide Openshift specific Options
helm upgrade --install k10 kasten/k10 --namespace=kasten-io \
  --set eula.accept=true \
  --set eula.company=Nutanix \
  --set eula.email=no-reply@nutanix.com \
  --set global.persistence.storageClass=nutanix-volume \
  --set auth.tokenAuth.enabled=true \
  --set scc.create=true \
  --set route.enabled=true \
  --set route.tls.enabled=true

## 3. Verify Install
❯ kubectl get pods --namespace=kasten-io
NAME                                  READY   STATUS    RESTARTS   AGE
aggregatedapis-svc-69ffb8c56b-bsfvg   1/1     Running   0          90s
auth-svc-9c7f48b87-jbljt              1/1     Running   0          90s
catalog-svc-5d89797f96-zd6lx          2/2     Running   0          90s
controllermanager-svc-77485666fc-67mdf   1/1     Running   0          89s
crypto-svc-58c98cf4b8-7tcpw           3/3     Running   0          89s
dashboardbff-svc-f6f744bc7-8rgzb      1/1     Running   0          89s
```

```
executor-svc-7ff6f9d897-4gjkf            2/2     Running   0          89s
executor-svc-7ff6f9d897-5vqmp            2/2     Running   0          89s
executor-svc-7ff6f9d897-rm7cr            2/2     Running   0          89s
frontend-svc-764d849996-qdv9c            1/1     Running   0          90s
gateway-b57fd6849-99pnm                  1/1     Running   0          90s
jobs-svc-cd859ff76-c9cs2                 1/1     Running   0          90s
k10-grafana-5ddb99695f-nm7wg             1/1     Running   0          89s
kanister-svc-748cb4f464-r6n8g            1/1     Running   0          89s
logging-svc-5bbd67f7cc-qfjxv             1/1     Running   0          90s
metering-svc-7f747697c4-xwn82            1/1     Running   0          89s
prometheus-server-9f7769bbb-7cvdt        2/2     Running   0          89s
state-svc-5d89bbccbb-dgg24               2/2     Running   0          89s

## 4. Get Route Info and Navigate to k10 Application via Browser (e.g.,
https://k10-route-kasten-io.apps.ocp-hub.dachlab.net/k10/#).
❯ kubectl get route -n kasten-io
⎵⌋
NAME        HOST/PORT                                        PATH
SERVICES   PORT   TERMINATION     WILDCARD
k10-route   k10-route-kasten-io.apps.<oc-cluster>.<domain>   /k10/
gateway    http   edge/Redirect   None

## 5. Generate Openshift OAuth Access Token to Access k10 Application via
UI.  Can be retrieved via OC cli.
oc whoami --show-token
```

Deploy and Configure Kasten k10 Multi-Cluster Manager

The K10 multi-cluster manager simplifies K10 operations across multiple Kubernetes clusters.

Administrators define primary-secondary relationships between their K10 instances. Primary K10 instances provide a single entry point and dashboard for administrators to manage secondary instances.

K10 resources, like Policies and Profiles, are defined in the primary instance and distributed to secondary instances. Secondary instances enact their policies and the secondaries' actions and metrics are summarized in the primary instance.

With K10 already installed in each cluster, a simple bootstrapping process enables the K10 multi-cluster manager. Through the bootstrapping process, one cluster will be setup as the primary. Additional clusters are then bootstrapped as secondaries.

The `k10multicluster` tool is used to setup `primary cluster` as a K10 Multi-Cluster Manager, and subsequently used to add or remove `secondary` clusters.

Requirements

The primary requirement for setting up the K10 Multi-Cluster Manager is having K10 installed.

In addition, The Secondary Server's K10 ingress & API Server must be accessible by the primary.

Bootstrap Primary Kasten k10 Multi-Cluster Manager (target ACM Hub Cluster)

The cluster from which the K10 Multi-Cluster Manager will be accessed is designated as primary.

The primary cluster defines policies and other configuration centrally. Centrally defined policies and configuration can then be distributed to designated clusters to be enacted.

The primary cluster also aggregates metrics so that they may be reported centrally.

The `k10multicluster` tool uses `Kubernetes contexts` available on the local system. To see the contexts available, use the following command:

`kubectl config get-contexts`

- Install k10multicluster on linux if it doesn't already exists
- Configure the ACM Hub Cluster primary cluster via k10multicluster cli

```
## 1. install k10multicluster on linux if it doesn't already exists
KASTEN_VERSION=$(kubectl -n kasten-io get deployment kanister-svc -o
jsonpath='{.spec.template.spec.containers[?(@.name=="kanister-
svc")].image}' | cut -d : -f 2)
curl -sSL "https://github.com/kastenhq/external-
tools/releases/download/${KASTEN_VERSION}/k10multicluster_${KASTEN_VERSION
}_linux_amd64.tar.gz"  | tar xz -C /tmp && \
sudo mv /tmp/k10multicluster /usr/local/bin && \
sudo chmod +x /usr/local/bin/k10multicluster

## 2. Configure the ACM Hub Cluster primary cluster via k10multicluster
cli
k10multicluster setup-primary \
  --context=kasten-io/api-ocp-hub-dachlab-net:6443/kube:admin \
  --name=ocp-hub \
  --replace
```

Bootstrap Secondary Kasten k10 Workload Clusters

Non-primary clusters are designated as secondaries.

The secondary clusters receive policies and other configuration from the primary cluster. Once policies are distributed to a secondary, the local K10 installation enacts the policy. This ensures that the policy will continue to be enforced, even if disconnected from the primary.

- Configure ACM Managed Openshift Clusters as k10 secondary clusters via k10multicluster cli

```
## run on az1
k10multicluster bootstrap \
  --primary-context=kasten-io/api-ocp-hub-dachlab-net:6443/kube:admin \
  --primary-name=ocp-hub \
  --secondary-context=kasten-io/api-ocp-az1-dachlab-net:6443/kube:admin \
  --secondary-name=ocp-az1 \
  --secondary-cluster-ingress-tls-insecure=true \
  --secondary-cluster-ingress=https://k10-route-kasten-io.apps.ocp-
```

```
az1.dachlab.net/k10/ \
  --replace

## run on az2
k10multicluster bootstrap \
  --primary-context=default/api-ocp-hub-dachlab-net:6443/kube:admin \
  --primary-name=ocp-hub \
  --secondary-context=default/api-ocp-az2-dachlab-net:6443/kube:admin \
  --secondary-name=ocp-az2 \
  --secondary-cluster-ingress-tls-insecure=true \
  --secondary-cluster-ingress=https://k10-route-kasten-io.apps.ocp-
az2.dachlab.net/k10/ \
  --replace
```

https://k10-route-kasten-io.apps.ocp-az1.dachlab.net/k10/

Install via Operator

https://docs.kasten.io/latest/install/openshift/operator.html

```
oc new-project kasten-io \
  --description="Kubernetes data management platform" \
  --display-name="Kasten K10"
```

Appendix

Install with OpenShift Authentication - DEX OIDC Provider

References

- https://www.kasten.io/kubernetes/resources/blog/learn-the-best-way-to-install-kasten-k10-on-openshift
- https://docs.kasten.io/latest/access/authentication.html#openshift-authentication

Kasten K10 integrates with Dex, an identity service that uses OpenID Connect to drive authentication for other apps. It acts as a portal to other identity providers through "connectors". This lets Dex defer authentication to LDAP servers, SAML providers, or other identity providers like GitHub, Google, and Active Directory. Among those connectors is the OpenShift OAuth connector. Dex is acting as an OAuth client on behalf of Kasten K10. This picture depicts the relationship between Kasten K10, Dex, and the OpenShift OAuth server.

- Configure the OAuth Client to be registered with Openshift
- Configure Self-Signed CA Bundle ConfigMap needed for DEX to Ingress Router connectivity

`APPS_BASE_DOMAIN` which is apps + base domain portion appended to each route on OpenShift when you don't specify explicitly a host for instance : apps.myopenshiftcluster.com `API_BASE_DOMAIN` which is the OpenShift API FQDN for instance api.myopenshiftcluster.com

```
## 1. Configure the OAuth Client to be registered with Openshift
APPS_BASE_DOMAIN=apps.ocp-az1.dachlab.net
API_BASE_DOMAIN=api.ocp-az1.dachlab.net

## token has to be created and mounted manually starting with k8s 1.24
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: k10-dex-sa-token
  annotations:
    kubernetes.io/service-account.name: k10-dex-sa
type: kubernetes.io/service-account-token
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: k10-dex-sa
  namespace: kasten-io
  annotations:
    serviceaccounts.openshift.io/oauth-redirecturi.dex: "https://k10-
route-kasten-io.$( echo $APPS_BASE_DOMAIN )/k10/dex/callback"
secrets:
  - name: k10-dex-sa-token
EOF

DEX_TOKEN=$(kubectl get secrets k10-dex-sa-token -o
jsonpath='{.data.token}' | base64 -d)

## 2. Configure Self-Signed CA Bundle ConfigMap needed for DEX to Ingress
Router connectivity.
## This assumes that openshift router has been configured with internal
PKI of Openshift, so ingress-tls is most likely self-signed.
CA_CERT=$(kubectl get secret router-ca -n openshift-ingress-operator -o
jsonpath='{.data.tls\.crt}' | base64 -d)
kubectl create configmap custom-ca-bundle-store --from-literal=custom-ca-
bundle.pem=$CA_CERT --namespace kasten-io --dry-run=client -o yaml |
kubectl apply -f -

## quickly validate that cert is formatted correctly within configmap
using openssl
kubectl get configmap custom-ca-bundle-store --namespace kasten-io -o
jsonpath='{.data.custom-ca-bundle\.pem}' | openssl x509 -in - -noout -text

## validate variables are set correctly
echo $APPS_BASE_DOMAIN
echo $API_BASE_DOMAIN
echo $DEX_TOKEN

## 3. Install K10 with Helm
helm repo add kasten https://charts.kasten.io/
helm repo update
helm upgrade --install k10 kasten/k10 --namespace=kasten-io \
```

```
  --set eula.accept=true \
  --set eula.company=Nutanix \
  --set eula.email=no-reply@nutanix.com \
  --set global.persistence.storageClass=nutanix-volume \
  --set scc.create=true \
  --set route.enabled=true \
  --set route.tls.enabled=true \
  --set auth.openshift.enabled=true \
  --set auth.openshift.serviceAccount=k10-dex-sa \
  --set auth.openshift.clientSecret=${DEX_TOKEN} \
  --set auth.openshift.dashboardURL=https://k10-route-kasten-
io.${APPS_BASE_DOMAIN}/k10/ \
  --set auth.openshift.openshiftURL=https://${API_BASE_DOMAIN}:6443 \
  --set auth.openshift.insecureCA=true \
  --set cacertconfigmap.name=custom-ca-bundle-store \
  --reset-values

## TODO: seems like this option preps openshift environment and preps
resources, but fails due to manual token creation issue.
k10tools openshift prepare-install --insecure-ca --recreate-resources
```

Known Issues

1. Docker Rate Limits

2. SecurityContext Warnings during Pre-Flight Checks.

If you see the following warning, you can ignore as this should be addressed during helm install

```
W0922 18:22:53.272306       7 warnings.go:70] would violate PodSecurity
"restricted:latest": allowPrivilegeEscalation != false (container
"container" must set securityContext.allowPrivilegeEscalation=false),
unrestricted capabilities (container "container" must set
securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or
container "container" must set securityContext.runAsNonRoot=true),
seccompProfile (pod or container "container" must set
securityContext.seccompProfile.type to "RuntimeDefault" or "Localhost")
```

References

https://docs.kasten.io/latest/install/openshift/helm.html
https://docs.kasten.io/operating/support.html#support

Troubleshooting

OpenShift iscsid service not starting after installation

> If you're seeing an error similar to below, iscsid service most likely not started correctly.

```
Warning FailedMount 16s (x7 over 61s) kubelet MountVolume.SetUp failed for
volume "pvc-6a57b403-1b08-4424-9aeb-f171b25c9ba3" : rpc error: code =
Internal desc = iscsi/lvm failure, last err seen:
iscsi: failed to sendtargets to portal x.x.x.x:3260 output: Failed to
connect to bus: No data available
iscsiadm: can not connect to iSCSI daemon (111)!
iscsiadm: Cannot perform discovery. Initiatorname required.
iscsiadm: Could not perform SendTargets discovery: could not connect to
iscsid
, err exit status 20
```

- Update MachineConfigs on Master and Worker Nodes to ensure that systemctl enables and starts up iscsi daemon

```
## configure master and worker node machine configs
cat <<EOF | kubectl apply -f -
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 99-worker-ntnx-csi-enable-iscsid
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
      - enabled: true
        name: iscsid.service
EOF

cat <<EOF | kubectl apply -f -
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-master-ntnx-csi-enable-iscsid
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
      - enabled: true
        name: iscsid.service
EOF
```

Failed to create OIDC Provider when configuring Openshift Auth with Dex

This error implies the CA is not trusted on ingress router, so make to follow steps in configuring ca-bundle.pem

```
logging-svc-5c9556c744-md2mh logging-svc [1] ...
"Message"=>"Failed to create OIDC provider",
...
"cause"=>{"message"=>"Get "https://k10-route-kasten-io.apps.ocp-
az2.dachlab.net/k10/dex/.well-known/openid-configuration": x509:
certificate signed by unknown authority"}}},
...
```

> leverage k10tools to debug ca-certificate

The k10tools debug ca-certificate command can be used to check if the CA certificate is installed properly in K10.

```
❯ k10tools debug ca-certificate
⌙
CA Certificate Checker:
  Fetching configmap which contains CA Certificate information : custom-
ca-bundle-store
  Certificate exists in configmap  -  OK
  Found container aggregatedapis-svc to extract certificate
  Certificate exists in container at/etc/ssl/certs/custom-ca-bundle.pem
  Certificates matched successfully  -  OK

❯ k10tools debug auth -d openshift
⌙
Verify OpenShift OAuth Server Connection:
  Openshift URL - https://api.ocp-az1.dachlab.net:6443/.well-known/oauth-
authorization-server
  Trying to connect to Openshift without TLS (insecureSkipVerify=false)
  Connection failed, testing other options
  Trying to connect to Openshift with TLS but verification disabled
(insecureSkipVerify=true)
  Connection succeeded  -  OK

Verify OpenShift Service Account Token:
  Initiating token verification
  Fetched ConfigMap - k10-dex
  Service Account for OpenShift authentication - k10-dex-sa
  Service account fetched
  Secret - k10-dex-sa-token retrieved
  Token retrieved from Service Account secrets
  Token retrieved from ConfigMap
  Token matched  -  OK

Get Service Account Error Events:
```

```
  Searching for events with error in Service Account - k10-dex-sa
  No error event found in service account - k10-dex-sa  -  OK
```

```
openssl s_client -host k10-route-kasten-io.apps.ocp-hub.dachlab.net -port 443 -showcerts
```

Kubernetes 1.24 no longer automounts service account token secrets

https://kubernetes.io/docs/concepts/configuration/secret/#service-account-token-secrets

Some of the Kasten documentation for configuring OIDC Provider with service account token doesn't note supported kubernetes version and possible issues...i.e., service account token doesn't automount

Creating Storage Class(es) as needed

Install Nutanix CSI Operator from Hub and Create Instance as documented.

Subsequently Define Storage Classes for Volumes

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
data:
  key: <pass>
kind: Secret
metadata:
  name: ntnx-secret
  namespace: ntnx-system
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: nutanix-volume
parameters:
  csi.storage.k8s.io/controller-expand-secret-name: ntnx-secret
  csi.storage.k8s.io/controller-expand-secret-namespace: ntnx-system
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/node-publish-secret-name: ntnx-secret
  csi.storage.k8s.io/node-publish-secret-namespace: ntnx-system
  csi.storage.k8s.io/provisioner-secret-name: ntnx-secret
  csi.storage.k8s.io/provisioner-secret-namespace: ntnx-system
  storageContainer: Default
  storageType: NutanixVolumes
provisioner: csi.nutanix.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
EOF
```