Test report - Deployment of Arktos Cluster without Mizar CNI on Premise

This document captures the steps to deploy an Arktos cluster lab without Mizar CNI. The machine in this lab used are **16 GB RAM, 8 vCPUs, 128 GB storage, and Ubuntu 18.04 LTS.**

Install golang 1.13.9

Date-20 Dec. 2021

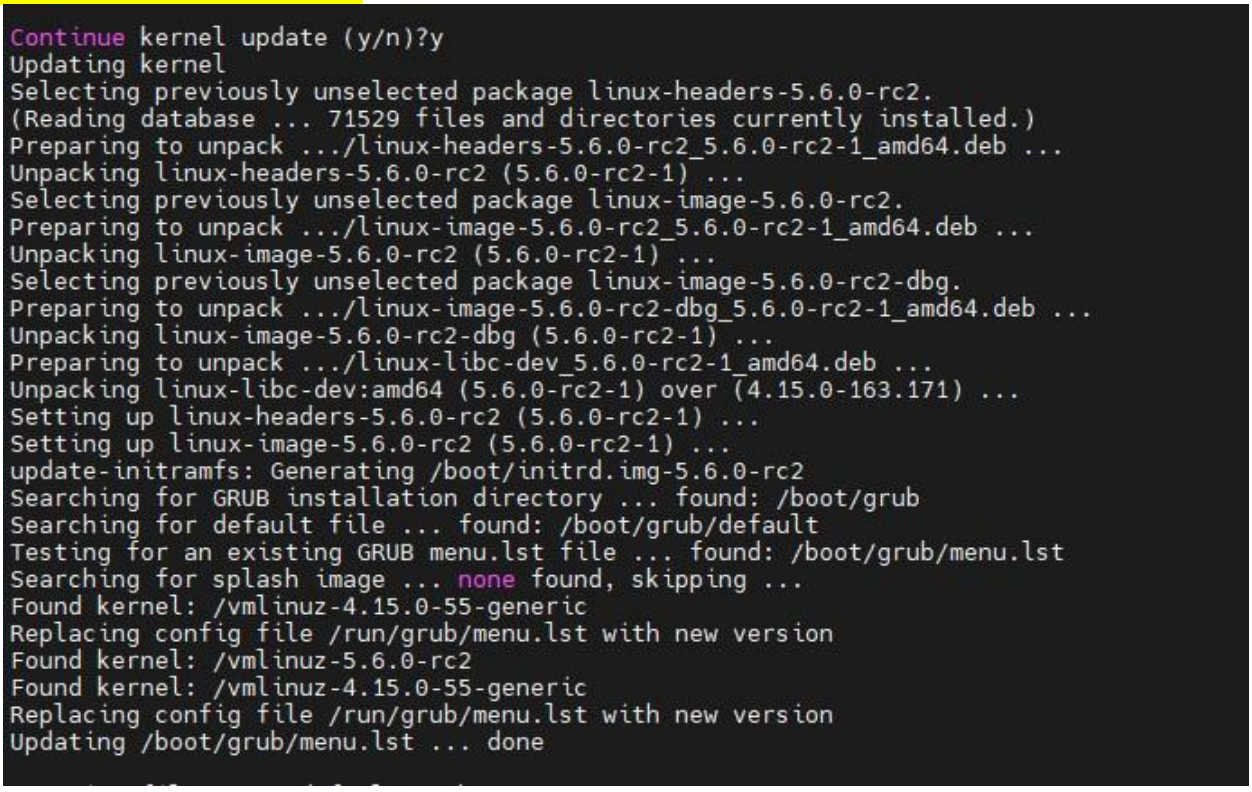
Step-1: Update kernel (If required)

To check kernel, run following command

```
uname -a
```

```
wget https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh

sudo bash kernelupdate.sh
```



Step-2: Install dependencies

Run the following steps to install dependencies required for arktos deployment:

mkdir -p $GOPATH/src/github.com

cd $GOPATH/src/github.com

git clone https://github.com/CentaurusInfra/arktos

cd arktos

sudo bash hack/setup-dev-node.sh

make

```
Done.
Please run and add 'export PATH=$PATH:/usr/local/go/bin' into your shell profile.
You can proceed to run arktos-up.sh if you want to launch a single-node cluster.
root@node-d:/src/github.com/arktos# export PATH=$PATH:/usr/local/go/bin
root@node-d:/src/github.com/arktos# make
+++ [1220 05:08:07] Building go targets for linux/amd64:
    ./vendor/k8s.io/code-generator/cmd/deepcopy-gen
+++ [1220 05:08:20] Building go targets for linux/amd64:
    ./vendor/k8s.io/code-generator/cmd/defaulter-gen
+++ [1220 05:08:33] Building go targets for linux/amd64:
    ./vendor/k8s.io/code-generator/cmd/conversion-gen
+++ [1220 05:08:50] Building go targets for linux/amd64:
    ./vendor/k8s.io/kube-openapi/cmd/openapi-gen
+++ [1220 05:09:07] Building go targets for linux/amd64:
    ./vendor/github.com/go-bindata/go-bindata/go-bindata
Running copyright check for repo: /src/github.com/arktos, logging to _output/ArktosCopyrightTool.log
/src/github.com/arktos /src/github.com/arktos
warning: inexact rename detection was skipped due to too many files.
warning: you may want to set your diff.renameLimit variable to at least 3067 and retry the command.
/src/github.com/arktos
/src/github.com/arktos /src/github.com/arktos
```

Run Arktos

The easiest way to run Arktos is to bring up a single-node cluster in your local development box:

cd $GOPATH/src/github.com/arktos

hack/arktos-up.sh

```
****************************************
Local Kubernetes cluster is running. Press Ctrl-C to shut it down.

Logs:
  /tmp/kube-apiserver0.log
  /tmp/kube-controller-manager.log


  /tmp/kube-proxy.log
  /tmp/kube-scheduler.log
  /tmp/kubelet.log

To start using your cluster, you can open up another terminal/tab and run:

  export KUBECONFIG=/var/run/kubernetes/admin.kubeconfig
Or
  export KUBECONFIG=/var/run/kubernetes/adminN(N=0,1,...).kubeconfig

  cluster/kubectl.sh

Alternatively, you can write to the default kubeconfig:

  export KUBERNETES_PROVIDER=local

  cluster/kubectl.sh config set-cluster local --server=https://node-b:6443 --certificate-authority=/var/run/kubernetes/server-ca.crt
  cluster/kubectl.sh config set-credentials myself --client-key=/var/run/kubernetes/client-admin.key --client-certificate=/var/run/kubernetes/client-admin.crt
  cluster/kubectl.sh config set-context local --cluster=local --user=myself
  cluster/kubectl.sh config use-context local
  cluster/kubectl.sh
```

1) Check nodes status:

./cluster/kubectl.sh get nodes

```
root@node-d:/src/github.com/arktos# ./cluster/kubectl.sh get nodes
NAME      STATUS   ROLES    AGE   VERSION
node-d    Ready    <none>   45s   v0.9.0
root@node-d:/src/github.com/arktos#
```

2) Check pods status:

./cluster/kubectl.sh get pods -Ao wide

```
root@node-d:/src/github.com/arktos# ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE     NAME                              HASHKEY              READY   STATUS    RESTARTS   AGE     IP              NODE     N
kube-system   coredns-default-fc74854f6-5czrd   2848153121546700097  1/1     Running   0          4m14s   10.88.0.4       node-d   <
kube-system   kube-dns-554c5866fc-vb4jx         4465680067321967175  3/3     Running   0          4m15s   10.88.0.3       node-d   <
kube-system   virtlet-2t9gb                     1716316480427753843  1/3     Running   0          3m57s   192.168.1.213   node-d   <
root@node-d:/src/github.com/arktos#
```

Deployment of Centaurus dashboard:

Link for YAML file of the dashboard:

https://click2cloud-my.sharepoint.com/personal/amit_nagpure_click2cloud_net/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Famit%5Fnagpure%5Fclick2cloud%5Fnet%2FDocuments%2FMicrosoft%20Teams%20Chat%20Files%2Fkubernetes%2Ddashboard%2Eyaml&parent=%2Fpersonal%2Famit%5Fnagpure%5Fclick2cloud%5Fnet%2FDocuments%2FMicrosoft%20Teams%20Chat%20Files

Create YAML file naming 'kubernetes-dashboard.yaml' change image c2c/…..0.6.3

and in args input '—authentication-mode=basic'

```
spec:
  containers:
    - name: kubernetes-dashboard
      image: c2cengg20190034/dashboard:0.6.3
      imagePullPolicy: Always
      ports:
        - containerPort: 8443
          protocol: TCP
      args:
        - --auto-generate-certificates
        - --enable-skip-login
        - --authentication-mode=basic
        - --disable-settings-authorizer
        - --enable-insecure-login
        - --insecure-bind-address=0.0.0.0
        - --namespace=kubernetes-dashboard
        # Uncomment the following line to manually specify Kubernetes API server Host
        # If not specified, Dashboard will attempt to auto discover the API server and connect
        # to it. Uncomment only if the default does not work.
        # - --apiserver-host=http://my-address:port
      volumeMounts:
```

Input the following commands before deploying the dashboard:

sudo sed -i '0,/RANDFILE/{s/RANDFILE/\#&/}' /etc/ssl/openssl.cnf
openssl genrsa -out dashboard.key 2048

openssl rsa -in dashboard.key -out dashboard.key

openssl req -sha256 -new -key dashboard.key -out dashboard.csr -subj "/CN=$(hostname -I | awk '{print $1}')"

openssl x509 -req -sha256 -days 365 -in dashboard.csr -signkey dashboard.key -out dashboard.crt

./cluster/kubectl.sh create namespace kubernetes-dashboard

./cluster/kubectl.sh create secret generic kubernetes-dashboard-certs --from-file=$HOME/dashboard.key --from-file=$HOME/dashboard.crt -n kubernetes-dashboard

./cluster/kubectl.sh create -f kubernetes-dashboard.yaml

```
root@node-d:/src/github.com/arktos# ./cluster/kubectl.sh create -f kubernetes-dashboard.yaml
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
root@node-d:/src/github.com/arktos# ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE              NAME                                        HASHKEY              READY   STATUS    RESTARTS   AGE   IP
 READINESS GATES
kube-system            coredns-default-fc74854f6-5czrd             2848153121546700097  1/1     Running   0          16m   10.88.0.4
 <none>
kube-system            kube-dns-554c5866fc-vb4jx                   4465680067321967175  3/3     Running   0          16m   10.88.0.3
 <none>
kube-system            virtlet-2t9gb                               1716316480427753843  3/3     Running   0          16m   192.168.1.213
 <none>
kubernetes-dashboard   dashboard-metrics-scraper-5c577c86cf-gxp9n   2527889656649936796  1/1     Running   0          20s   10.88.0.7
 <none>
kubernetes-dashboard   kubernetes-dashboard-587589d554-h9lhh       6467292087516159583  1/1     Running   0          20s   10.88.0.6
 <none>
kubernetes-dashboard   kubernetes-dashboard-587589d554-zrdnq       5122227841876677148  1/1     Running   0          20s   10.88.0.5
```

Create the Kubernetes Dashboard password file:

mkdir /etc/kubernetes/auth -p

vi /etc/kubernetes/auth/auth.csv

Here is the file content:

adminpass,admin,admin,system:masters

we need to configure while deploying the arktos the following entry in ‘common.sh’

vi /hack/lib/common.sh

350 - --basic-auth-file=/etc/kubernetes/auth/auth.csv

```
APISERVER_LOG=${LOG_DIR}/$apiserverlog
${CONTROLPLANE_SUDO} "${GO_OUT}/hyperkube" kube-apiserver "${authorizer_arg}" "${priv_arg}" ${runtime_config} \
  ${cloud_config_arg} \
  "${advertise_address}" \
  "${node_port_range}" \
  --v="${LOG_LEVEL}" \
  --vmodule="${LOG_SPEC}" \
  --audit-policy-file="${AUDIT_POLICY_FILE}" \
  --audit-log-path="${LOG_DIR}/$apiserverauditlog" \
  --basic-auth-file="/etc/kubernetes/auth/auth.csv" \
  --cert-dir="${CERT_DIR}" \
  --client-ca-file="${CERT_DIR}/client-ca.crt" \
  --kubelet-client-certificate="${CERT_DIR}/client-kube-apiserver.crt" \
  --kubelet-client-key="${CERT_DIR}/client-kube-apiserver.key" \
  --service-account-key-file="${SERVICE_ACCOUNT_KEY}" \
  --service-account-lookup="${SERVICE_ACCOUNT_LOOKUP}" \
  --enable-admission-plugins="${ENABLE_ADMISSION_PLUGINS}" \
  --disable-admission-plugins="${DISABLE_ADMISSION_PLUGINS}" \
  --admission-control-config-file="${ADMISSION_CONTROL_CONFIG_FILE}" \
  --bind-address="${API_BIND_ADDR}" \
  --secure-port=$secureport \
```

Now re-run the arktos script:

hack/arktos-up.sh

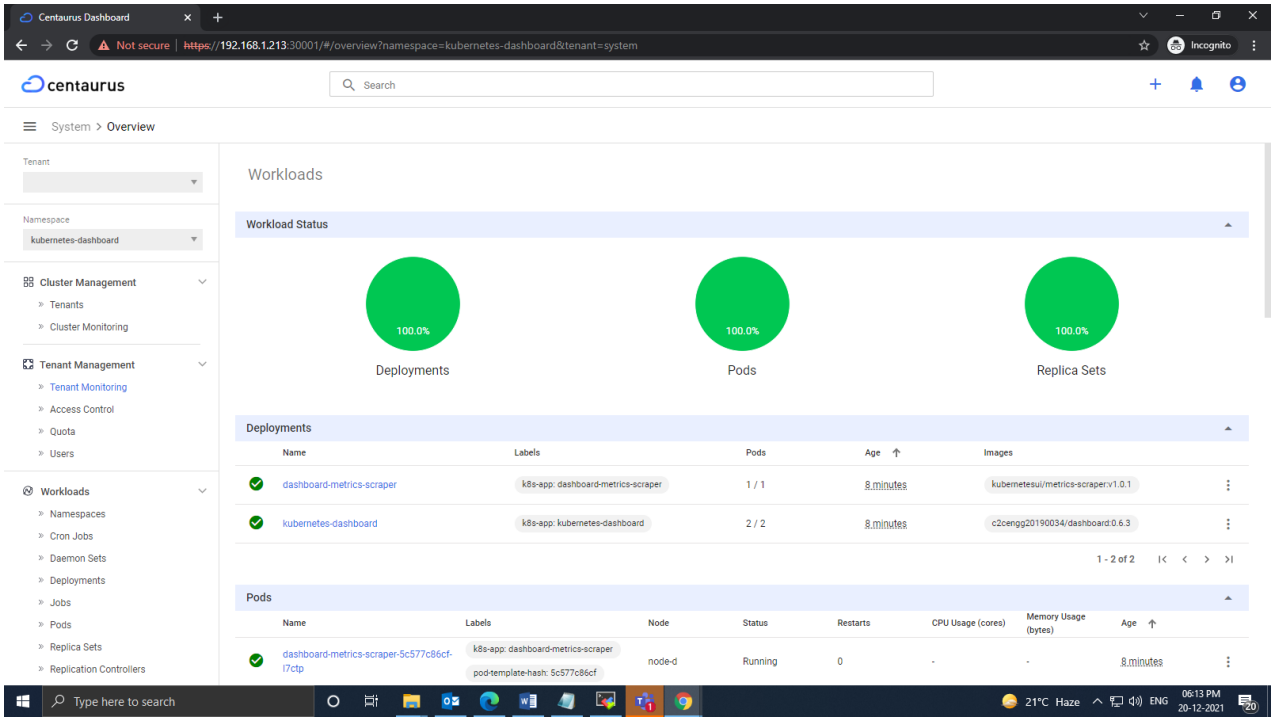
```
  cluster/kubectl.sh
^CCleaning up...
Killing the following apiserver running processes
987 has been killed
hack/arktos-up.sh: line 167:  1414 Terminated              ${CONTROLPLANE_SUDO} "${GO_OUT}/hyperkube" kube-controller-manager --v="${LOG_LEVEL}"
="${KUBE_CONTROLLER_MANAGER_ALLOCATE_NODE_CIDR}" --cluster-cidr="${KUBE_CONTROLLER_MANAGER_CLUSTER_CIDR}" --vmodule="${LOG_SPEC}" --service-acco
${SERVICE_ACCOUNT_KEY}" --root-ca-file="${ROOT_CA_FILE}" --cluster-signing-cert-file="${CLUSTER_SIGNING_CERT_FILE}" --cluster-signing-key-file="
FILE}" --enable-hostpath-provisioner="${ENABLE_HOSTPATH_PROVISIONER}" ${node_cidr_args[@]+"${node_cidr_args[@]}"} --pvclaimbinder-sync-period="$
IOD}" --feature-gates="${FEATURE_GATES}" "${cloud_config_arg[@]}" --kubeconfig "${kubeconfigfilepaths}" --use-service-account-credentials --cont
LLERS}" --leader-elect=false --cert-dir="${CERT_DIR}" --default-network-template-path="${ARKTOS_NETWORK_TEMPLATE}" > "${CTLRMGR_LOG}" 2>&1
hack/arktos-up.sh: line 167:  1416 Terminated              ${CONTROLPLANE_SUDO} "${GO_OUT}/arktos-network-controller" --v="${LOG_LEVEL}" --maste
80" --kube-apiserver-ip "${API_HOST_IP_EXTERNAL}" --kube-apiserver-port=6443 --resource-name-salt="${SALT}" > "${ARKTOS_NETWORK_CONTROLLER_LOG}"
hack/arktos-up.sh: line 167:  1519 Terminated              sudo "${GO_OUT}/hyperkube" kube-proxy --v="${LOG_LEVEL}" --config=/tmp/kube-proxy.yam
}://${API_HOST}:${port}" > "${PROXY_LOG}" 2>&1
Cleanup runtime metadata folder
hack/arktos-up.sh: line 167:  1521 Terminated              ${CONTROLPLANE_SUDO} "${GO_OUT}/hyperkube" kube-scheduler --v="${LOG_LEVEL}" --leader
fig "${kubeconfigfilepaths}" --feature-gates="${FEATURE_GATES}" > "${SCHEDULER_LOG}" 2>&1
Cleanup runtime log folder

root@node-d:/src/github.com/arktos# hack/arktos-up.sh
DBG: Flannel CNI plugin will be installed AFTER cluster is up
DBG: effective feature gates AllAlpha=false,WorkloadInfoDefaulting=true,QPSDoubleGCController=true,QPSDoubleRSController=true,MandatoryArktosNet
DBG: effective disabling admission plugins
DBG: effective default network template file is /src/github.com/arktos/hack/testdata/default-flat-network.tmpl
DBG: kubelet arg RESOLV_CONF is /run/systemd/resolve/resolv.conf
WARNING : The kubelet is configured to not fail even if swap is enabled; production deployments should disable swap.
cni plugin is bridge; arktos will use bridge to provision pod network
Ensuring firewall to allow traffic forward by default
-P FORWARD ACCEPT
Ensuring minimum cni plugin installation...
found bridge, host-local, loopback
setting up cni conf file...
/etc/cni/net.d/bridge.conf already exists; keep it.
```

Now re-deploy the kubernetes-dashboard file:

```
root@node-d:~/go/src/k8s.io/arktos# ./cluster/kubectl.sh apply -f kubernetes-dashboard.yaml
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

The Dashboard will be accessible at `https://<host_machine_ip>:30001` and you can log in using `username` & `password` used in `auth.csv`


Centaurus Dashboard
Not secure | https://192.168.1.213:30001/#/overview?namespace=kubernetes-dashboard&tenant=system
Incognito
centaurus
Search
System > Overview
Tenant
Namespace
kubernetes-dashboard
Cluster Management
Tenants
Cluster Monitoring
Tenant Management
Tenant Monitoring
Access Control
Quota
Users
Workloads
Namespaces
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Workloads
Workload Status
100.0%
Deployments
100.0%
Pods
100.0%
Replica Sets
Deployments
Name
Labels
Pods
Age
Images
dashboard-metrics-scraper
k8s-app: dashboard-metrics-scraper
1 / 1
8 minutes
kubernetesui/metrics-scraper:v1.0.1
kubernetes-dashboard
k8s-app: kubernetes-dashboard
2 / 2
8 minutes
c2cengg20190034/dashboard:0.6.3
1 - 2 of 2
Pods
Name
Labels
Node
Status
Restarts
CPU Usage (cores)
Memory Usage (bytes)
Age
dashboard-metrics-scraper-5c577c86cf-l7ctp
k8s-app: dashboard-metrics-scraper
pod-template-hash: 5c577c86cf
node-d
Running
0
-
-
8 minutes
Type here to search
21°C Haze
ENG
06:13 PM
20-12-2021