Arktos deployment without Mizar CNI (On Premise)

Date-15 Dec. 2021

This document is intended for new users to install the Arktos platform without Mizar as the underlying network technology.

Prepare a machine of 16 Gb RAM, 8 vCPUs, 128G Storage, Ubuntu 18.04 LTS.

1. Check the kernel version:

Command:

uname -a

Update the kernel if the kernel version is below `5.6.0-rc2`

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

sudo bash kernelupdate.sh

uname -a

Output:

```
root@node-d:~# uname -a
Linux node-d 5.6.0-rc2 #1 SMP Tue Feb 25 18:54:05 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
root@node-d:~#
```

2. Clone the Arktos repository and install the required dependencies:

git clone https://github.com/Click2Cloud-Centaurus/arktos.git ~/go/src/k8s.io/arktos

cd ~/go/src/k8s.io/arktos

git checkout cni-mizar

sudo bash ./hack/setup-dev-node.sh

Output:

```
root@node-d:~# git clone https://github.com/Click2Cloud-Centaurus/arktos.git  ~/go/src/k8s.io/arktos
Cloning into '/root/go/src/k8s.io/arktos'...
remote: Enumerating objects: 61743, done.
remote: Counting objects: 100% (1147/1147), done.
remote: Compressing objects: 100% (528/528), done.
remote: Total 61743 (delta 713), reused 908 (delta 599), pack-reused 60596
Receiving objects: 100% (61743/61743), 221.50 MiB | 10.98 MiB/s, done.
Resolving deltas: 100% (37873/37873), done.
Checking out files: 100% (20766/20766), done.
root@node-d:~# cd ~/go/src/k8s.io/arktos
root@node-d:~/go/src/k8s.io/arktos# git checkout cni-mizar
Branch 'cni-mizar' set up to track remote branch 'cni-mizar' from 'origin'.
Switched to a new branch 'cni-mizar'
```

Command:

echo export PATH=$PATH:/usr/local/go/bin\ >> ~/.profile

echo cd \$HOME/go/src/k8s.io/arktos >> ~/.profile

source ~/.profile

Output:

```
root@node-d:~/go/src/k8s.io/arktos# echo export PATH=$PATH:/usr/local/go/bin\ >> ~/.profile
root@node-d:~/go/src/k8s.io/arktos# echo cd \$HOME/go/src/k8s.io/arktos >> ~/.profile
root@node-d:~/go/src/k8s.io/arktos# source ~/.profile
```

3. Start Arktos cluster

Command:

`./hack/arktos-up.sh`

The terminal was stuck in this state.

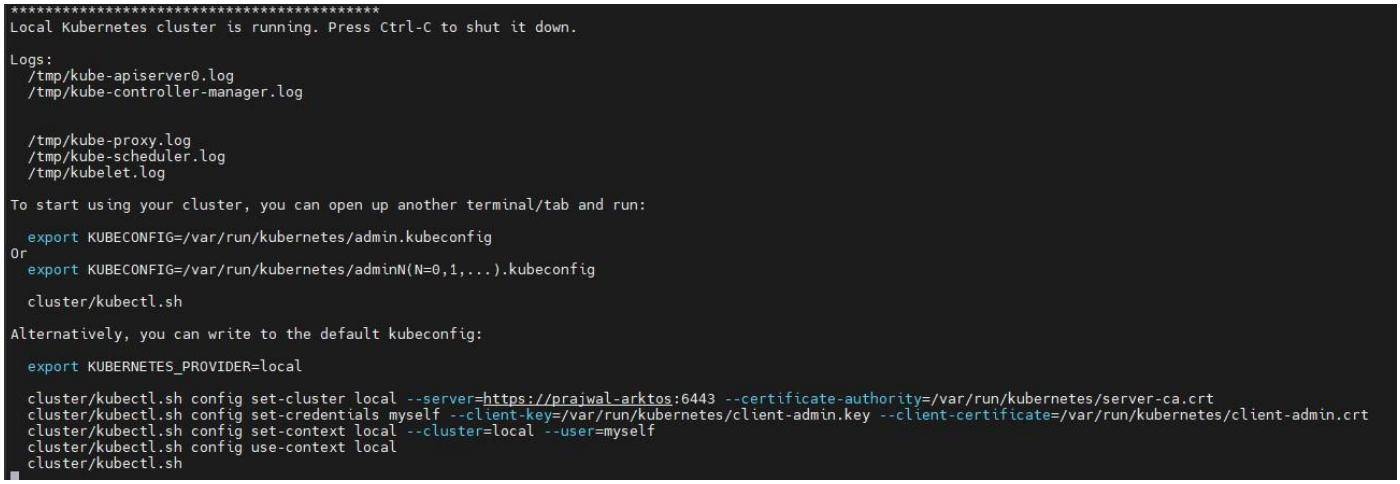
```
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
Waiting for node ready at api server
```

After restarting the containerd we got the output:

systemctl restart containerd

```
root@node-d:~/go/src/k8s.io/arktos# systemctl restart containerd
root@node-d:~/go/src/k8s.io/arktos# systemctl status containerd
● containerd.service - containerd container runtime
   Loaded: loaded (/lib/systemd/system/containerd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2021-12-10 07:33:41 UTC; 9s ago
     Docs: https://containerd.io
  Process: 31699 ExecStartPre=/sbin/modprobe overlay (code=exited, status=0/SUCCESS)
 Main PID: 31706 (containerd)
    Tasks: 27
   CGroup: /system.slice/containerd.service
           └─31706 /usr/bin/containerd

Dec 10 07:33:49 node-d containerd[31706]: time="2021-12-10T07:33:49.621118864Z" level=info msg="No cni config template is specified, wait for other system compone
Dec 10 07:33:49 node-d containerd[31706]: time="2021-12-10T07:33:49.721994662Z" level=info msg="No cni config template is specified, wait for other system compone
Dec 10 07:33:49 node-d containerd[31706]: time="2021-12-10T07:33:49.822897670Z" level=info msg="No cni config template is specified, wait for other system compone
Dec 10 07:33:49 node-d containerd[31706]: time="2021-12-10T07:33:49.923930419Z" level=info msg="No cni config template is specified, wait for other system compone
Dec 10 07:33:50 node-d containerd[31706]: time="2021-12-10T07:33:50.024838447Z" level=info msg="No cni config template is specified, wait for other system compone
Dec 10 07:33:50 node-d containerd[31706]: time="2021-12-10T07:33:50.125881377Z" level=info msg="No cni config template is specified, wait for other system compone
Dec 10 07:33:50 node-d containerd[31706]: time="2021-12-10T07:33:50.227019115Z" level=info msg="No cni config template is specified, wait for other system compone
Dec 10 07:33:50 node-d containerd[31706]: time="2021-12-10T07:33:50.428371958Z" level=info msg="No cni config template is specified, wait for other system compone
Dec 10 07:33:50 node-d containerd[31706]: time="2021-12-10T07:33:50.529398482Z" level=info msg="No cni config template is specified, wait for other system compone
Dec 10 07:33:50 node-d containerd[31706]: time="2021-12-10T07:33:50.630468349Z" level=info msg="No cni config template is specified, wait for other system compone
root@node-d:~/go/src/k8s.io/arktos# systemctl status containerd
● containerd.service - containerd container runtime
   Loaded: loaded (/lib/systemd/system/containerd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2021-12-10 07:33:41 UTC; 39s ago
     Docs: https://containerd.io
  Process: 31699 ExecStartPre=/sbin/modprobe overlay (code=exited, status=0/SUCCESS)
 Main PID: 31706 (containerd)
    Tasks: 75
   CGroup: /system.slice/containerd.service
           ├─ 310 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id 4c70725e1ab35bbfded5566e32c0c474651ac01c3dacc8b134ba8882ed86275b -address /run/container
           ├─ 333 /pause
           ├─ 565 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id 71494a722b3f8fba4cd6243a6dfc65a38adbe744c9414662568dbc637f6089b0 -address /run/container
           ├─ 587 /pause
```

Output:



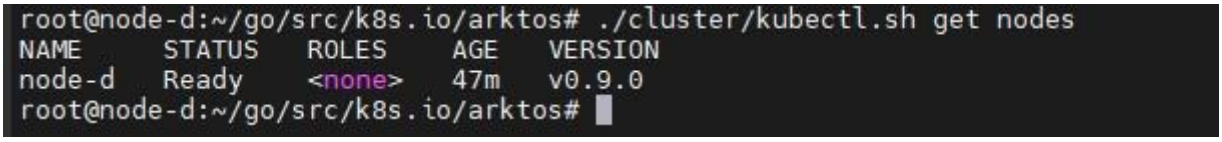
4. Leave the "arktos-up.sh" terminal and open another terminal to the master node.

Check nodes:

Command:

```
./cluster/kubectl.sh get nodes
```

Output:



Deploy kubernetes dashboard:

Link for yaml file of dashboard:

https://click2cloud-my.sharepoint.com/:u:/g/personal/amit_nagpure_click2cloud_net/EdmJx0itP0RGl8WqAVVplbwBurpuI2EhSi3_Uj-d8xy7zQ

vi kubernetes-dashboard.yaml

copy the link yaml content to

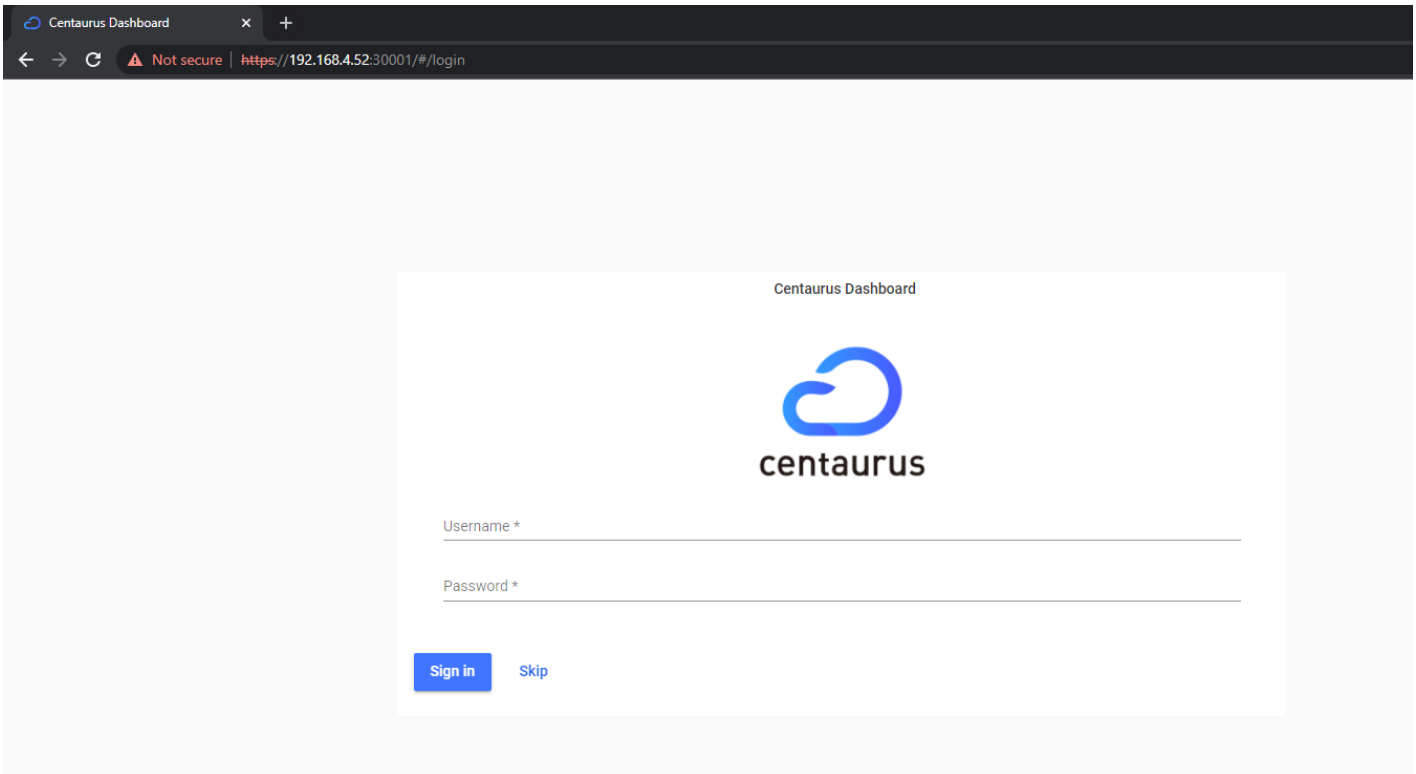
```
sudo sed -i '0,/RANDFILE/{s/RANDFILE/\#&/}' /etc/ssl/openssl.cnf
openssl genrsa -out dashboard.key 2048
openssl rsa -in dashboard.key -out dashboard.key
openssl req -sha256 -new -key dashboard.key -out dashboard.csr -subj "/CN=$(hostname -I | awk '{print $1}')"
openssl x509 -req -sha256 -days 365 -in dashboard.csr -signkey dashboard.key -out dashboard.crt

kubectl create namespace kubernetes-dashboard

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

kubectl create -f kubernetes-dashboard.yaml
```

```
root@node-d:~/go/src/k8s.io/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
```

./cluster/kubectl get pods -Ao wide

```
root@node-d:~/go/src/k8s.io/arktos# ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE              NAME                                        HASHKEY              READY   STATUS    RESTARTS   AGE     IP             NODE     NOMINATED NODE
  READINESS GATES
kube-system            coredns-default-6ff7d98dc8-9r66m            3790096273370637885  1/1     Running   0          11m     10.88.0.3      node-d   <none>
  <none>
kube-system            kube-dns-554c5866fc-2rght                   3515277765848946229  3/3     Running   0          11m     10.88.0.2      node-d   <none>
  <none>
kube-system            virtlet-65pcv                               6213951806386579682  3/3     Running   0          7m30s   192.168.4.52   node-d   <none>
  <none>
kubernetes-dashboard   dashboard-metrics-scraper-6fbf748d6d-qrrwf  7726004176900680265  1/1     Running   0          67s     10.88.0.4      node-d   <none>
  <none>
kubernetes-dashboard   kubernetes-dashboard-5745876c85-6w6hn       6811219126517615809  1/1     Running   0          68s     10.88.0.5      node-d   <none>
  <none>
kubernetes-dashboard   kubernetes-dashboard-5745876c85-vm4xb       2942170762646836114  1/1     Running   0          67s     10.88.0.6      node-d   <none>
  <none>
```

The dashboard will expose on https:<ip>:30001



We need to set username and password for the dashboard

mkdir /etc/kubernetes/auth -p

vi /etc/kubernetes/auth/auth.csv

inside the auth.csv paste the following input

adminpass123,admin,admin,system:masters

Then go to file common.sh

vi ./hack/lib/common.sh

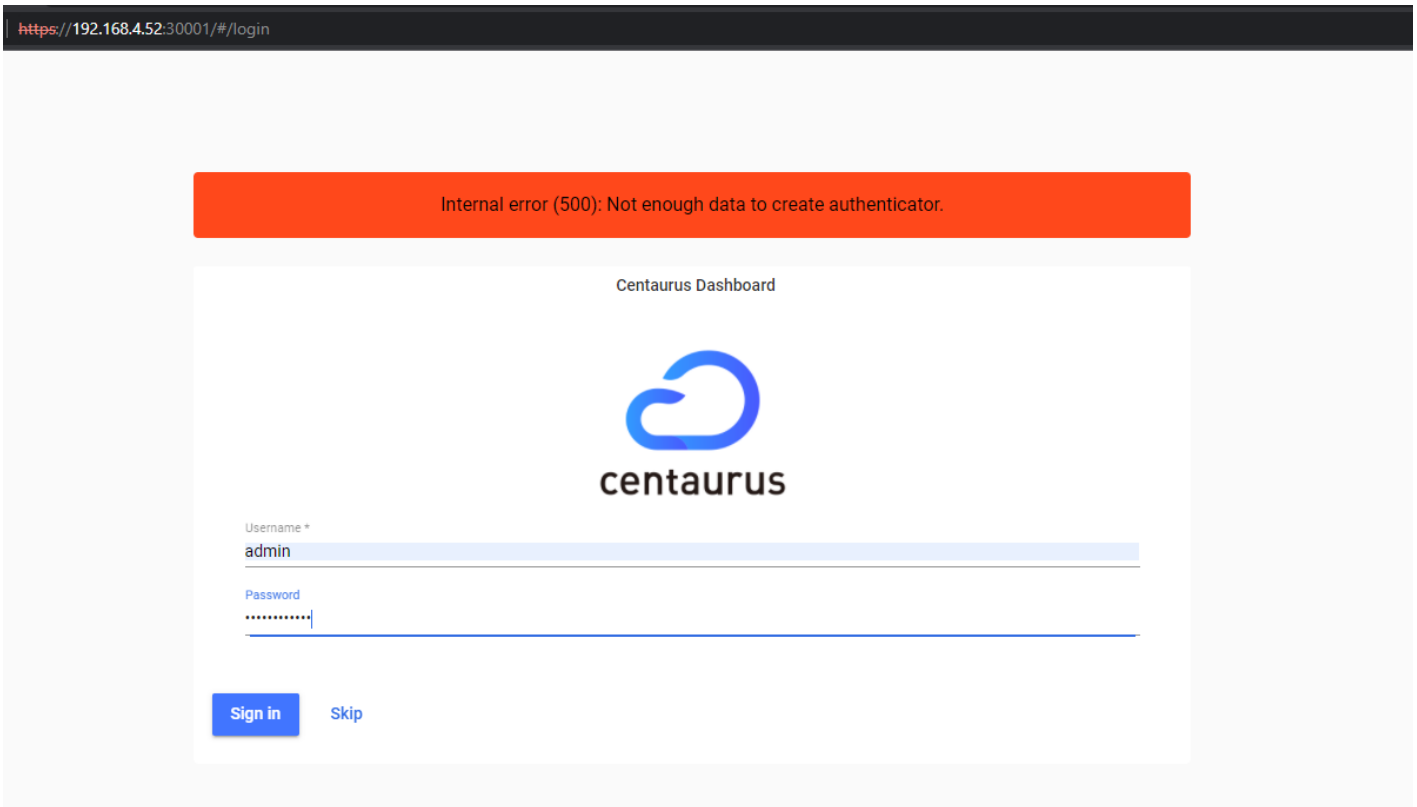
```
root@node-d:~/go/src/k8s.io/arktos# vi ./hack/lib/common.sh
```

after line.no 349 add the below entry

```
EOF
      AUDIT_POLICY_FILE="/tmp/kube-audit-policy-file$i"
    fi

    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}" \
-- INSERT --                                                                    350,7
```

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

hit the dashboard link again and try to login



ERROR :

Internal error (500)