Test report - Deployment of Arktos Cluster without Mizar CNI on Premise (Community code)

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.**

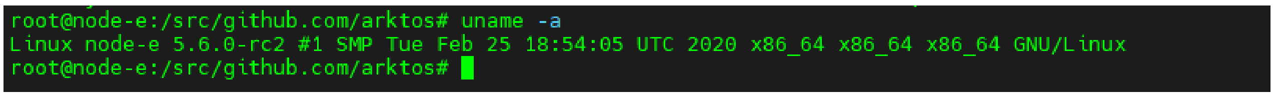
Date-30 Dec. 2021

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
```

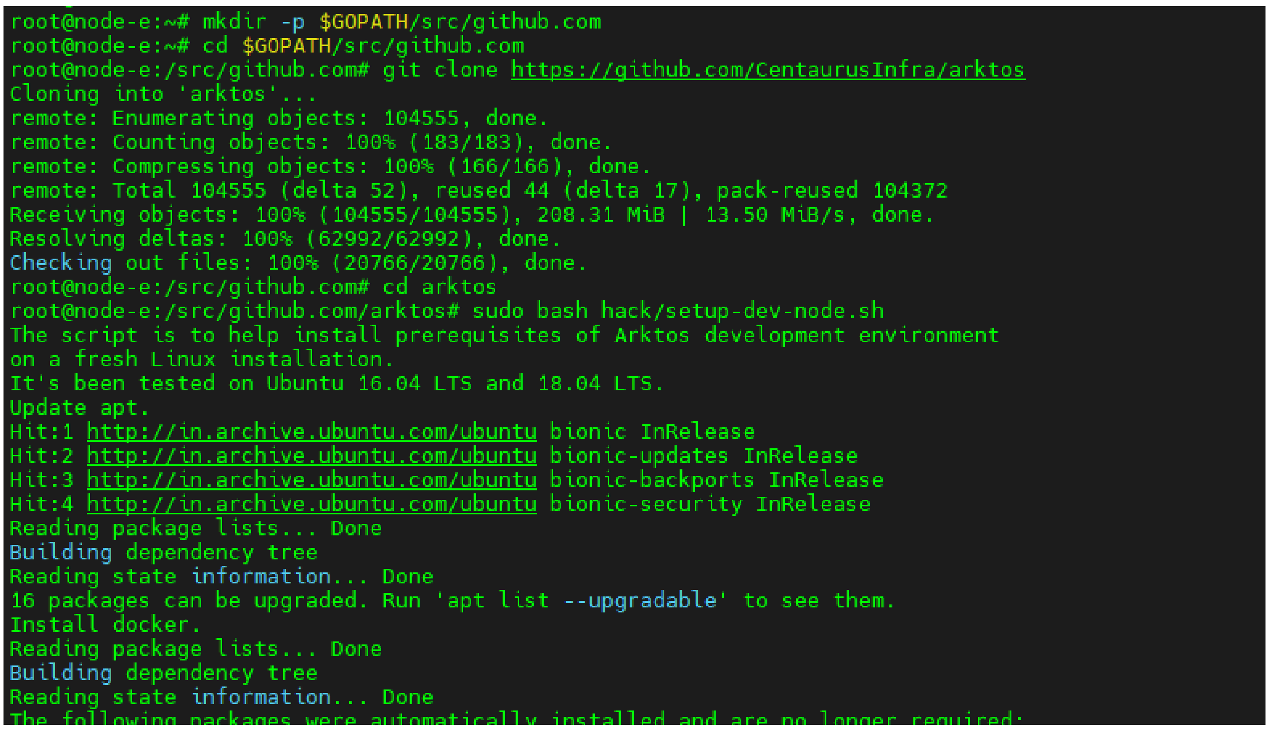


```
root@node-e:/src/github.com/arktos# uname -a
Linux node-e 5.6.0-rc2 #1 SMP Tue Feb 25 18:54:05 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
root@node-e:/src/github.com/arktos#
```

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
```



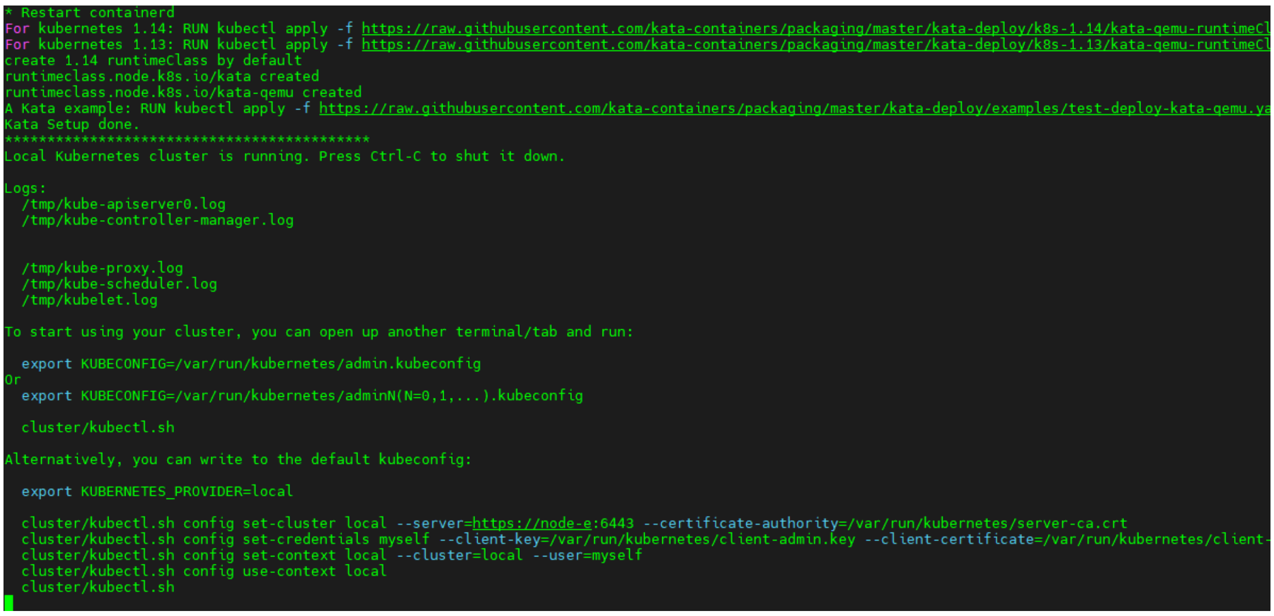
```
root@node-e:~# mkdir -p $GOPATH/src/github.com
root@node-e:~# cd $GOPATH/src/github.com
root@node-e:/src/github.com# git clone https://github.com/CentaurusInfra/arktos
Cloning into 'arktos'...
remote: Enumerating objects: 104555, done.
remote: Counting objects: 100% (183/183), done.
remote: Compressing objects: 100% (166/166), done.
remote: Total 104555 (delta 52), reused 44 (delta 17), pack-reused 104372
Receiving objects: 100% (104555/104555), 208.31 MiB | 13.50 MiB/s, done.
Resolving deltas: 100% (62992/62992), done.
Checking out files: 100% (20766/20766), done.
root@node-e:/src/github.com# cd arktos
root@node-e:/src/github.com/arktos# sudo bash hack/setup-dev-node.sh
The script is to help install prerequisites of Arktos development environment
on a fresh Linux installation.
It's been tested on Ubuntu 16.04 LTS and 18.04 LTS.
Update apt.
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
16 packages can be upgraded. Run 'apt list --upgradable' to see them.
Install docker.
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
```

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

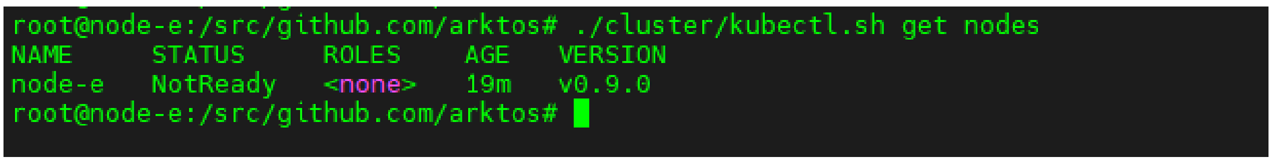
hack/arktos-up.sh



```
* Restart containerd
For kubernetes 1.14: RUN kubectl apply -f https://raw.githubusercontent.com/kata-containers/packaging/master/kata-deploy/k8s-1.14/kata-qemu-runtimeCl
For kubernetes 1.13: RUN kubectl apply -f https://raw.githubusercontent.com/kata-containers/packaging/master/kata-deploy/k8s-1.13/kata-qemu-runtimeCl
create 1.14 runtimeClass by default
runtimeclass.node.k8s.io/kata created
runtimeclass.node.k8s.io/kata-qemu created
A Kata example: RUN kubectl apply -f https://raw.githubusercontent.com/kata-containers/packaging/master/kata-deploy/examples/test-deploy-kata-qemu.ya
Kata Setup done.
*******************************************
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-e: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-
  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:

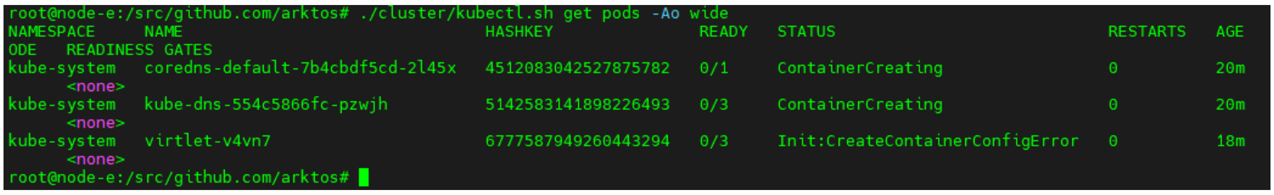
./cluster/kubectl.sh get nodes



```
root@node-e:/src/github.com/arktos# ./cluster/kubectl.sh get nodes
NAME      STATUS     ROLES    AGE   VERSION
node-e    NotReady   <none>   19m   v0.9.0
root@node-e:/src/github.com/arktos#
```

2) Check pods status:

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



```
root@node-e:/src/github.com/arktos# ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE     NAME                                 HASHKEY               READY   STATUS                           RESTARTS   AGE
ODE    READINESS GATES
kube-system   coredns-default-7b4cbdf5cd-2l45x   4512083042527875782   0/1     ContainerCreating                0          20m
      <none>
kube-system   kube-dns-554c5866fc-pzwjh          5142583141898226493   0/3     ContainerCreating                0          20m
      <none>
kube-system   virtlet-v4vn7                      6777587949260443294   0/3     Init:CreateContainerConfigError  0          18m
      <none>
root@node-e:/src/github.com/arktos#
```

Deployment failed