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

This document captures the steps to deploy an Arktos cluster lab without Mizar CNI. The machine in this lab used are **t2.2xlarge, 128 GB storage and Ubuntu 18.04 LTS.**

Date-31 Dec. 2021

Step-1: Update kernel (If required)

To check kernel, run following command

```
uname -a
```

```
ubuntu@arktos:~$ uname -a
Linux arktos 5.4.0-1060-aws #63~18.04.1-Ubuntu SMP Mon Nov 15 14:31:31 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
ubuntu@arktos:~$
```

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

sudo bash kernelupdate.sh
```

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

Step-2: Install dependencies

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

```
sudo mkdir -p $GOPATH/src/github.com

cd $GOPATH/src/github.com

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

cd arktos

sudo bash hack/setup-dev-node.sh

export PATH=$PATH:/usr/local/go/bin

sudo -i

cd $GOPATH/src/github.com/arktos

make
```

```
ubuntu@arktos:~$ sudo mkdir -p $GOPATH/src/github.com
ubuntu@arktos:~$ cd $GOPATH/src/github.com
ubuntu@arktos:/src/github.com$ sudo git clone https://github.com/CentaurusInfra/arktos
Cloning into 'arktos'...
remote: Enumerating objects: 104576, done.
remote: Counting objects: 100% (204/204), done.
remote: Compressing objects: 100% (176/176), done.
remote: Total 104576 (delta 69), reused 59 (delta 28), pack-reused 104372
Receiving objects: 100% (104576/104576), 208.32 MiB | 23.79 MiB/s, done.
Resolving deltas: 100% (63009/63009), done.
Checking out files: 100% (20766/20766), done.
ubuntu@arktos:/src/github.com$ cd arktos
ubuntu@arktos:/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.
```

Run Arktos

The easiest way to run Arktos is to bring up a single-node cluster in your local development box: 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@arktos:/src/github.com/arktos# ./cluster/kubectl.sh get nodes
NAME      STATUS   ROLES    AGE     VERSION
arktos    Ready    <none>   4m14s   v0.9.0
root@arktos:/src/github.com/arktos#
```

2) Check pods status:

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

```
root@arktos:/src/github.com/arktos# ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE     NAME                                HASHKEY             READY   STATUS    RESTARTS   AGE     IP
kube-system   coredns-default-689575ccc5-9g5mr    3354589342012628897   1/1     Running   0          4m19s   10.88.0.3
kube-system   kube-dns-554c5866fc-24k5z           4631335625920737646   3/3     Running   0          4m19s   10.88.0.2
kube-system   virtlet-wz88f                       3666647795961894360   3/3     Running   0          2m41s   172.31.22.165
root@arktos:/src/github.com/arktos#
```

Deploy test pods:

Command:

./cluster/kubectl.sh apply -f https://github.com/Click2Cloud-Centaurus/Documentation/blob/main/test-yamls/test_pods.yaml

Check deployed pods:

Command:

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

Output

```
root@arktos:/src/github.com/arktos# ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE     NAME                                HASHKEY             READY   STATUS    RESTARTS   AGE   IP              NODE
default       netpod1                             8374296334242631856   1/1     Running   0          15s   10.88.0.4       arktos
default       netpod2                             3985853951120055144   1/1     Running   0          15s   10.88.0.5       arktos
kube-system   coredns-default-689575ccc5-9g5mr    3354589342012628897   1/1     Running   0          15m   10.88.0.3       arktos
kube-system   kube-dns-554c5866fc-24k5z           4631335625920737646   3/3     Running   0          15m   10.88.0.2       arktos
kube-system   virtlet-wz88f                       3666647795961894360   3/3     Running   0          13m   172.31.22.165   arktos
root@arktos:/src/github.com/arktos#
```

Check ping deployed pods:

Command:

./cluster/kubectl.sh exec -it netpod1 ping 10.88.0.5

./cluster/kubectl.sh exec -it netpod2 ping 10.88.0.4

Output:

```
root@arktos:/src/github.com/arktos# ./cluster/kubectl.sh exec -it netpod1 ping 10.88.0.5
PING 10.88.0.5 (10.88.0.5) 56(84) bytes of data.
64 bytes from 10.88.0.5: icmp_seq=1 ttl=64 time=0.077 ms
64 bytes from 10.88.0.5: icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from 10.88.0.5: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 10.88.0.5: icmp_seq=4 ttl=64 time=0.040 ms
^C
--- 10.88.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 63ms
rtt min/avg/max/mdev = 0.039/0.054/0.077/0.018 ms
root@arktos:/src/github.com/arktos# ./cluster/kubectl.sh exec -it netpod2 ping 10.88.0.4
PING 10.88.0.4 (10.88.0.4) 56(84) bytes of data.
64 bytes from 10.88.0.4: icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from 10.88.0.4: icmp_seq=2 ttl=64 time=0.061 ms
64 bytes from 10.88.0.4: icmp_seq=3 ttl=64 time=0.034 ms
64 bytes from 10.88.0.4: icmp_seq=4 ttl=64 time=0.044 ms
^C
--- 10.88.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 78ms
rtt min/avg/max/mdev = 0.034/0.045/0.061/0.011 ms
root@arktos:/src/github.com/arktos#
```