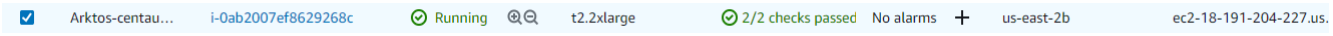
Test report -Deploying Arktos cluster with Mizar CNI on AWS

This document captures the steps to deploy an Arktos cluster lab with mizar cni. The machines in this lab used are AWS EC2 t2.2xlarge (8 CPUs, 32GB mem, 128 GiB storage) Ubuntu 18.04 LTS.

Date: 7.12.2021

Created an instance on AWS



SSH instance using credentials

NOTE: Attach elastic IP to the instance

Step-1: Update kernel version

- Check kernel version:

```
uname -a
```

Output

```
root@ip-172-31-6-218:~# uname -a
Linux ip-172-31-6-218 5.4.0-1058-aws #61~18.04.3-Ubuntu SMP Fri Oct 1 14:04:01 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
```

Here kernel version was `5.4.0-1045-aws` hence, to update kernel version to `5.6.0-rc2`, we used following steps :

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

```
sudo bash kernelupdate.sh
```

Output

```
root@ip-172-31-6-218:~# wget https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh
--2021-12-07 10:32:21--  https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 791 [text/plain]
Saving to: 'kernelupdate.sh'

kernelupdate.sh                   100%[=============================================================>]     791  --.-KB/s    in 0s

2021-12-07 10:32:21 (46.9 MB/s) - 'kernelupdate.sh' saved [791/791]

root@ip-172-31-6-218:~# sudo bash kernelupdate.sh -y
--2021-12-07 10:32:39--  https://mizar.s3.amazonaws.com/linux-5.6-rc2/linux-headers-5.6.0-rc2_5.6.0-rc2-1_amd64.deb
Resolving mizar.s3.amazonaws.com (mizar.s3.amazonaws.com)... 52.217.206.49
Connecting to mizar.s3.amazonaws.com (mizar.s3.amazonaws.com)|52.217.206.49|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7621020 (7.3M) []
Saving to: '../linux-5.6-rc2/linux-headers-5.6.0-rc2_5.6.0-rc2-1_amd64.deb'

linux-headers-5.6.0-rc2_5.6.0-rc2- 100%[=============================================================>]   7.27M  27.4MB/s    in 0.3s
```

Step-2: Install dependencies

Relogin the instance and run following steps to install dependencies required for arktos deployment:

- Clone the Arktos repository

```
git clone https://github.com/Click2Cloud-Centaurus/arktos.git
~/go/src/k8s.io/arktos -b default-cni-mizar
```

Output

```
ubuntu@ip-172-31-6-218:~$ git clone https://github.com/Click2Cloud-Centaurus/arktos.git ~/go/src/k8s.io/arktos -b default-cni-mizar
Cloning into '/home/ubuntu/go/src/k8s.io/arktos'...
remote: Enumerating objects: 61743, done.
remote: Counting objects: 100% (1147/1147), done.
remote: Compressing objects: 100% (529/529), done.
remote: Total 61743 (delta 712), reused 905 (delta 598), pack-reused 60596
Receiving objects: 100% (61743/61743), 221.50 MiB | 26.75 MiB/s, done.
Resolving deltas: 100% (37872/37872), done.
Checking out files: 100% (20767/20767), done.
ubuntu@ip-172-31-6-218:~$ sudo bash $HOME/go/src/k8s.io/arktos/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://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
28 packages can be upgraded. Run 'apt list --upgradable' to see them.
Install docker.
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Then installed prerequisites required for Arktos clusters using following command

```
sudo bash $HOME/go/src/k8s.io/arktos/hack/setup-dev-node.sh
```

Output

```
ubuntu@ip-172-31-6-218:~$ sudo bash $HOME/go/src/k8s.io/arktos/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://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
28 packages can be upgraded. Run 'apt list --upgradable' to see them.
Install docker.
Reading package lists... Done
Building dependency tree
```

and then run the following commands:

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

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

```
source ~/.profile
```

Output

```
ubuntu@ip-172-31-6-218:~$ echo export PATH=$PATH:/usr/local/go/bin\ >> ~/.profile
ubuntu@ip-172-31-6-218:~$ echo cd \$HOME/go/src/k8s.io/arktos >> ~/.profile
ubuntu@ip-172-31-6-218:~$ source ~/.profile
```

Step-3: Start Arktos cluster

Run following steps to deploy arktos cluster with Mizar as CNI

```
CNIPLUGIN=mizar ./hack/arktos-up.sh
```

Finally we got following output, which indicates that arktos cluster created successfully with Mizar as CNI

Output

```
root" arch=amd64 module=kvm_intel name=kata-runtime pid=31689 source=runtime time="2021-12-07T10:57:25Z" level=error msg="kernel propert
y kvm_intel not found" arch=amd64 description="Intel KVM" name=kvm_intel pid=31689 source=runtime type=module time="2021-12-07T10:57:25Z"
level=error msg="ERROR: System is not capable of running Kata Containers" arch=amd64 name=kata-runtime pid=31689 source=runtime ERROR: S
ystem is not capable of running Kata Containers
Aborted. Current system does not support Kata Containers.
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://ip-172-31-6-218:6443 --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/kubern
etes/client-admin.crt
  cluster/kubectl.sh config set-context local --cluster=local --user=myself
  cluster/kubectl.sh config use-context local
  cluster/kubectl.sh
```

Leave this terminal here as it is (do not close the terminal) and open new terminal of same instance

Step-4 Check Cluster health

Open new terminal for same instance and run following commands:

1) Check node status

```
sudo ./cluster/kubectl.sh get nodes -Ao wide
```

Output

```
8   <none>           <none>
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get nodes -Ao wide
NAME              STATUS   ROLES    AGE     VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIM
E
ip-172-31-6-218   Ready    <none>   5m32s   v0.9.0    172.31.6.218   <none>        Ubuntu 18.04.6 LTS   5.6.0-rc2        containerd://1.4
.0-beta.1-29-g70b0d3cf
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$
```

2) Check pods status

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

Output

```
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE     NAME                              HASHKEY               READY   STATUS    RESTARTS   AGE     IP             NODE
    NOMINATED NODE   READINESS GATES
default       mizar-daemon-2j945                1334722514597826759   1/1     Running   0          2m35s   172.31.6.218   ip-172-31-6-21
8   <none>           <none>
default       mizar-operator-6b78d7ffc4-8xn79   6880898832392603202   1/1     Running   0          2m35s   172.31.6.218   ip-172-31-6-21
8   <none>           <none>
default       netpod1                           1182152794275396464   1/1     Running   0          71s     20.0.0.17      ip-172-31-6-21
8   <none>           <none>
default       netpod2                           7281071563371397426   1/1     Running   0          71s     20.0.0.37      ip-172-31-6-21
8   <none>           <none>
kube-system   coredns-default-79cf8cb96c-f9fh2   2279686948564416783   1/1     Running   0          2m35s   20.0.0.22      ip-172-31-6-21
8   <none>           <none>
kube-system   kube-dns-554c5866fc-6x4ml          1027154578229379342   3/3     Running   0          2m35s   20.0.0.18      ip-172-31-6-21
8   <none>           <none>
kube-system   virtlet-c95fx                     1009090319510796233   3/3     Running   0          2m35s   172.31.6.218   ip-172-31-6-21
8   <none>           <none>
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$
```

3) Check vpc status

```
sudo ./cluster/kubectl.sh get vpc -Ao wide
```

Output

```
.0-beta.1-29-g7d0d3c1
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get vpc -Ao wide
NAMESPACE   NAME   IP         PREFIX   VNI   DIVIDERS   STATUS        CREATETIME                   PROVISIONDELAY
default     vpc0   20.0.0.0   8        1     1          Provisioned   2021-12-07T10:57:43.055420   41.020747
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$
```

4) Check subnets

```
sudo ./cluster/kubectl.sh get subnets -Ao wide
```

Output

```
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get subnets -Ao wide
NAMESPACE   NAME   IP         PREFIX   VNI   VPC    STATUS        BOUNCERS   CREATETIME                   PROVISIONDELAY
default     net0   20.0.0.0   8        1     vpc0   Provisioned   1          2021-12-07T10:57:43.122338   61.1043
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$
```

5) Check net

```
sudo ./cluster/kubectl.sh get net -Ao wide
```

Output

```
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get net -Ao wide
NAME      TYPE    VPC                      PHASE   DNS
default   mizar   system-default-network   Ready   10.0.0.157
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$
```

6) Check dividers

```
sudo ./cluster/kubectl.sh get dividers -Ao wide
```

Output

```
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get dividers -Ao wide
NAMESPACE   NAME                                        VPC    IP             MAC                DROPLET           STATUS        CREAT
ETIME                   PROVISIONDELAY
default     vpc0-d-2f11e233-d586-479e-956d-69723fc7345d   vpc0   172.31.6.218   02:b1:9b:ca:42:ec   ip-172-31-6-218   Provisioned   2021-
12-07T10:58:24.069159   0.420354
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$
```

7) Check bouncers

```
sudo ./cluster/kubectl.sh get bouncers -Ao wide
```

Output

```
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get bouncers -Ao wide
NAMESPACE   NAME                                        VPC    NET    IP             MAC                DROPLET           STATUS
  CREATETIME                  PROVISIONDELAY
default     net0-b-ae82f2b0-a9a3-4bb6-aa84-7e0fae4009e9   vpc0   net0   172.31.6.218   02:b1:9b:ca:42:ec   ip-172-31-6-218   Provisioned
  2021-12-07T10:58:44.221976   1.075692
ubuntu@ip-172-31-6-218:~/go/src/k8s.io/arktos$
```