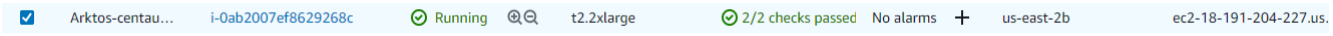
Test report -Deploying Arktos cluster with Mizar CNI on AWS

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

Step-1: Update kernel version

- Check kernel version:

```
uname -a
```

Output

```
root@arktos:~# uname -a
Linux arktos 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@arktos:~# wget https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh
--2021-12-07 05:32:24--  https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|: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 05:32:24 (28.8 MB/s) - 'kernelupdate.sh' saved [791/791]

root@arktos:~# sudo bash kernelupdate.sh
--2021-12-07 05:32:39--  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)... 52.216.92.171
Connecting to mizar.s3.amazonaws.com (mizar.s3.amazonaws.com)|52.216.92.171|: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  19.8MB/s    in 0.4s

2021-12-07 05:32:39 (19.8 MB/s) - '../linux-5.6-rc2/linux-headers-5.6.0-rc2_5.6.0-rc2-1_amd64.deb' saved [7621020/7621020]

--2021-12-07 05:32:39--  https://mizar.s3.amazonaws.com/linux-5.6-rc2/linux-image-5.6.0-rc2-dbg_5.6.0-rc2-1_amd64.deb
Resolving mizar.s3.amazonaws.com (mizar.s3.amazonaws.com)... 52.216.92.171
Connecting to mizar.s3.amazonaws.com (mizar.s3.amazonaws.com)|52.216.92.171|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 857827912 (818M) [application/x-www-form-urlencoded]
Saving to: '../linux-5.6-rc2/linux-image-5.6.0-rc2-dbg_5.6.0-rc2-1_amd64.deb'

linux-image-5.6.0-rc2-dbg_5.6.0-rc 100%[=============================================================>] 818.09M  45.0MB/s    in 26s

2021-12-07 05:33:06 (30.9 MB/s) - '../linux-5.6-rc2/linux-image-5.6.0-rc2-dbg_5.6.0-rc2-1_amd64.deb' saved [857827912/857827912]
```

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

```
root@arktos:~# git clone https://github.com/Click2Cloud-Centaurus/arktos.git ~/go/src/k8s.io/arktos -b default-cni-mizar
Cloning into '/root/go/src/k8s.io/arktos'...
remote: Enumerating objects: 61743, done.
remote: Counting objects: 100% (1147/1147), done.
remote: Compressing objects: 100% (527/527), done.
remote: Total 61743 (delta 712), reused 905 (delta 600), pack-reused 60596
Receiving objects: 100% (61743/61743), 221.39 MiB | 26.26 MiB/s, done.
Resolving deltas: 100% (37948/37948), done.
Checking out files: 100% (20767/20767), done.
```

Then installed prerequisites required for Arktos cluster suing following command

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

Output

```
root@arktos:~# 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
The following additional packages will be installed:
  bridge-utils containerd pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd docker.io pigz runc ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 28 not upgraded.
Need to get 74.2 MB of archives.
After this operation, 360 MB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 bridge-utils amd64 1.5-15ubuntu1 [30.1 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 runc amd64 1.0.1-0ubuntu2~18.04.1 [4155 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 containerd amd64 1.5.5-0ubuntu3~18.04.1 [33.0 MB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 docker.io amd64 20.10.7-0ubuntu5~18.04.3 [36.9 MB]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 ubuntu-fan all 0.12.10 [34.7 kB]
Fetched 74.2 MB in 2s (32.3 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 93723 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...
```

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

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

```
****************************************
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://arktos: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/kube
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

```
ubuntu@arktos:/root/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-RUNTIME
arktos   Ready    <none>   49m   v0.9.0    172.31.19.33   <none>        Ubuntu 18.04.6 LTS   5.6.0-rc2        containerd://1.4.0-beta.1-2
9-g70b0d3cf
```

2) Check pods status

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

Output

```
ubuntu@arktos:/root/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE     NAME                                HASHKEY              READY   STATUS    RESTARTS   AGE   IP             NODE     NOMINATE
D NODE   READINESS GATES
default       mizar-daemon-9zm46                  5447211320762283802   1/1     Running   0          56m   172.31.19.33   arktos   <none>
         <none>
default       mizar-operator-6b78d7ffc4-p979h     2737426404035080618   1/1     Running   0          56m   172.31.19.33   arktos   <none>
         <none>
kube-system   kube-dns-554c5866fc-2m9zz           7431288247843844650   0/3     Pending   0          56m   <none>         <none>   <none>
         <none>
```

3) Check vpc status

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

Output

```
ubuntu@arktos:/root/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          Init     2021-12-07T05:49:08.897243
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

4) Check subnets

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

Output

```
ubuntu@arktos:/root/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   Init     1          2021-12-07T05:49:08.965783
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

5) Check net

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

Output

```
ubuntu@arktos:/root/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get net -Ao wide
NAME      TYPE    VPC                    PHASE   DNS
default   mizar   system-default-network
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

6) Check dividers

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

Output

```
ubuntu@arktos:/root/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get dividers -Ao wide
No resources found.
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

7) Check bouncers

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

Output

```
ubuntu@arktos:/root/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get bouncers -Ao wide
No resources found.
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

8) Pod deployment:

Output

```
NAMESPACE     NAME                              HASHKEY              READY   STATUS    RESTARTS   AGE   IP             NODE     NOMINATE
D NODE   READINESS GATES
default       mizar-daemon-9zm46                5447211320762283802  1/1     Running   0          66m   172.31.19.33   arktos   <none>
        <none>
default       mizar-operator-6b78d7ffc4-p979h   2737426404035080618  1/1     Running   0          66m   172.31.19.33   arktos   <none>
        <none>
kube-system   kube-dns-554c5866fc-2m9zz         7431288247843844650  0/3     Pending   0          66m   <none>         <none>   <none>
        <none>
```

*Pod getting stuck in **Pending** state.*

After re-running the arktos cluster the outputs are as follows :

1) Check pods status

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

Output

```
ubuntu@arktos:/root/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE     NAME                              HASHKEY              READY   STATUS     RESTARTS   AGE     IP             NODE     NOMIN
ATED NODE   READINESS GATES
default       mizar-daemon-c5jrs                5433553103926506877  1/1     Running    0          4m47s   172.31.19.33   arktos   <none
>           <none>
default       mizar-operator-6b78d7ffc4-tkx5m   4252175553422586693  1/1     Running    0          4m47s   172.31.19.33   arktos   <none
>           <none>
kube-system   kube-dns-554c5866fc-dgwpx         1606520876123553529  3/3     Running    0          4m47s   20.0.0.2       arktos   <none
>           <none>
kube-system   virtlet-g97z2                     2296479715811486546  0/3     Init:0/1   0          4m47s   172.31.19.33   arktos   <none
>           <none>
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

2) Check vpc status

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

Output

```
ubuntu@arktos:/root/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   2021-12-07T08:46:54.119013   41.164764
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

3) Check subnets

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

Output

```
ubuntu@arktos:/root/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   2021-12-07T08:46:54.119013   41.164764
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

4) Check net

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

Output

```
NAME      TYPE    VPC                     PHASE   DNS
default   mizar   system-default-network
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

5) Check dividers

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

Output

```
ubuntu@arktos:/root/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get dividers -Ao wide
NAMESPACE   NAME                                    VPC    IP             MAC                DROPLET   STATUS        CREATETIME
            PROVISIONDELAY
default     vpc0-d-c2c16f11-1758-44fe-8dd1-bac7096437ad   vpc0   172.31.19.33   06:c5:e6:0f:54:9e   arktos    Provisioned   2021-12-07T08
:47:35.278403   0.240979
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

6) Check bouncers

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

Output

```
ubuntu@arktos:/root/go/src/k8s.io/arktos$ sudo ./cluster/kubectl.sh get bouncers -Ao wide
NAMESPACE   NAME                                    VPC    NET    IP             MAC                DROPLET   STATUS        CREATE
TIME                  PROVISIONDELAY
default     net0-b-ab61a092-5b45-4f9f-82e7-69e23bc0135e   vpc0   net0   172.31.19.33   06:c5:e6:0f:54:9e   arktos    Provisioned   2021-1
2-07T08:47:55.141759   1.05588
ubuntu@arktos:/root/go/src/k8s.io/arktos$
```

7) Pod deployment:

Output

```
root@aws-arktos:~/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
default       mizar-daemon-mrlht                  6336367786187412136   1/1     Running             0          24m   172.31.19.33   aws-ark
tos   <none>           <none>
default       mizar-operator-6b78d7ffc4-srqzn     4103094056190627943   1/1     Running             0          24m   172.31.19.33   aws-ark
tos   <none>           <none>
default       netpod1                             4245008424301658704   0/1     ContainerCreating   0          80s   <none>         aws-ark
tos   <none>           <none>
default       netpod2                             7416432928224090983   0/1     ContainerCreating   0          80s   <none>         aws-ark
tos   <none>           <none>
kube-system   kube-dns-554c5866fc-5dp84           2024609569699318849   3/3     Running             0          24m   20.0.0.22      aws-ark
tos   <none>           <none>
root@aws-arktos:~/go/src/k8s.io/arktos#
```

*Pod getting stuck in **ContainerCreating** state*

8) Error Logs

Output

```
Alternatively, you can write to the default kubeconfig:

  export KUBERNETES_PROVIDER=local

  cluster/kubectl.sh config set-cluster local --server=https://aws-arktos: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
W1207 09:43:37]: arktos network controller terminated unexpectedly, see /tmp/arktos-network-controller.log
```

Due to multiple ip generation pods are getting stuck in the *ContainerCreating* state.

```
root@aws-arktos:~/go/src/k8s.io/arktos# hostname -i
172.31.19.33 172.17.0.1 172.31.19.33 fe80::4c5:e6ff:fe0f:549e fe80::e8d0:aff:fe40:fc0 fe80::500a:fdff:fe6f:f93e fe80::7cdb:39ff:fe77:98dd
root@aws-arktos:~/go/src/k8s.io/arktos#
```

```
root@aws-arktos:~/go/src/k8s.io/arktos# cat /tmp/arktos-network-controller.log
F1207 09:43:31.868325    5221 network-controller.go:62] --kube-apiserver-ip must be the valid ip address of kube-apiserver.
root@aws-arktos:~/go/src/k8s.io/arktos#
```