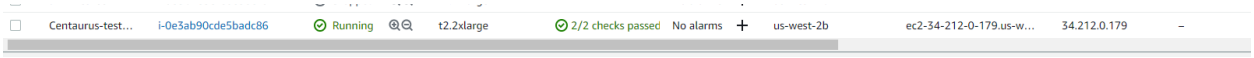
Test report -Deploying Arktos cluster with Mizar CNI on AWS

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), Ubuntu 18.04 LTS.

Date: 27.09.2021

Created an instance on AWS



SSH instance using credentials

Step-1: Update kernel version

- Check kernel version:

```
uname -a
```

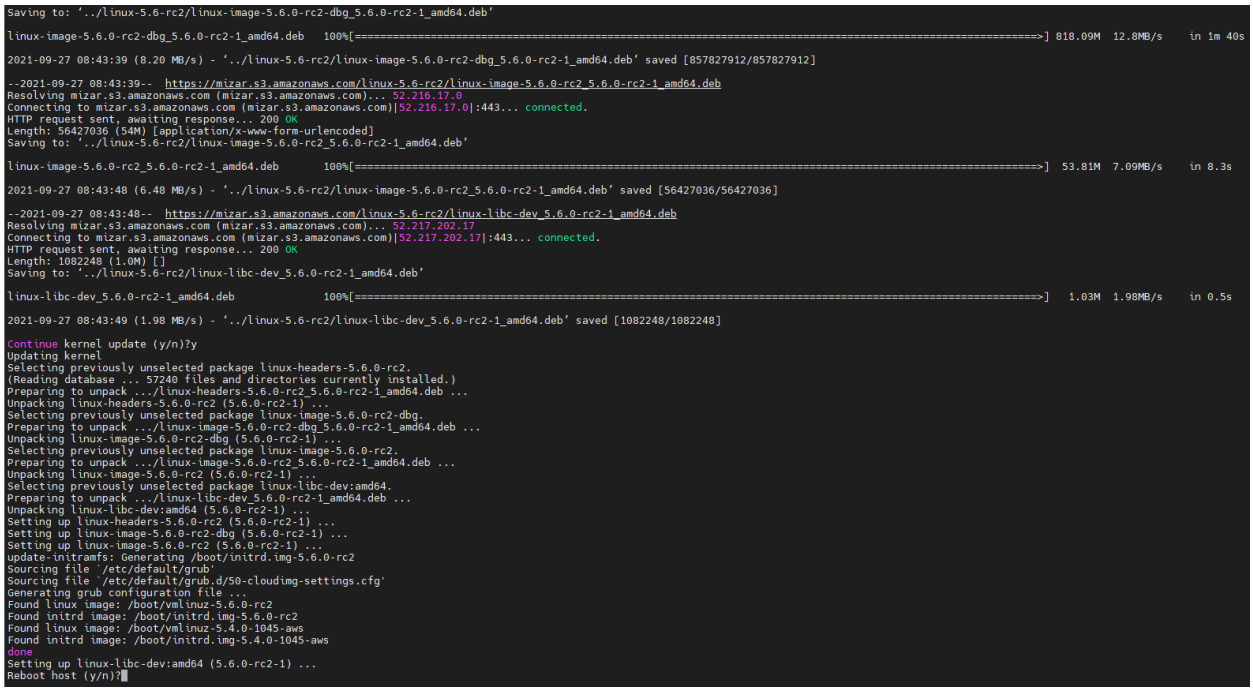
Output

```
ubuntu@ip-172-31-17-53:~$ uname -a
Linux ip-172-31-17-53 5.4.0-1045-aws #47~18.04.1-Ubuntu SMP Tue Apr 13 15:58:14 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
ubuntu@ip-172-31-17-53:~$
```

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



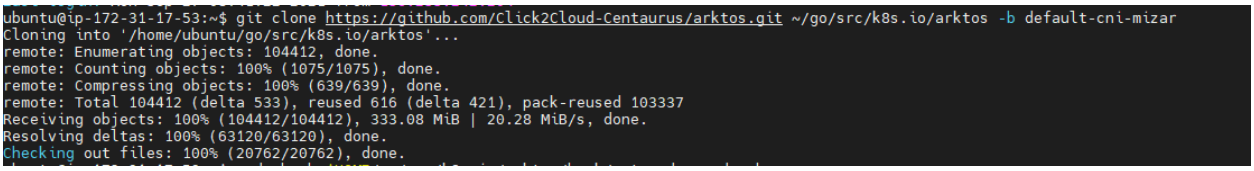
Step-2:Install dependencies

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
```

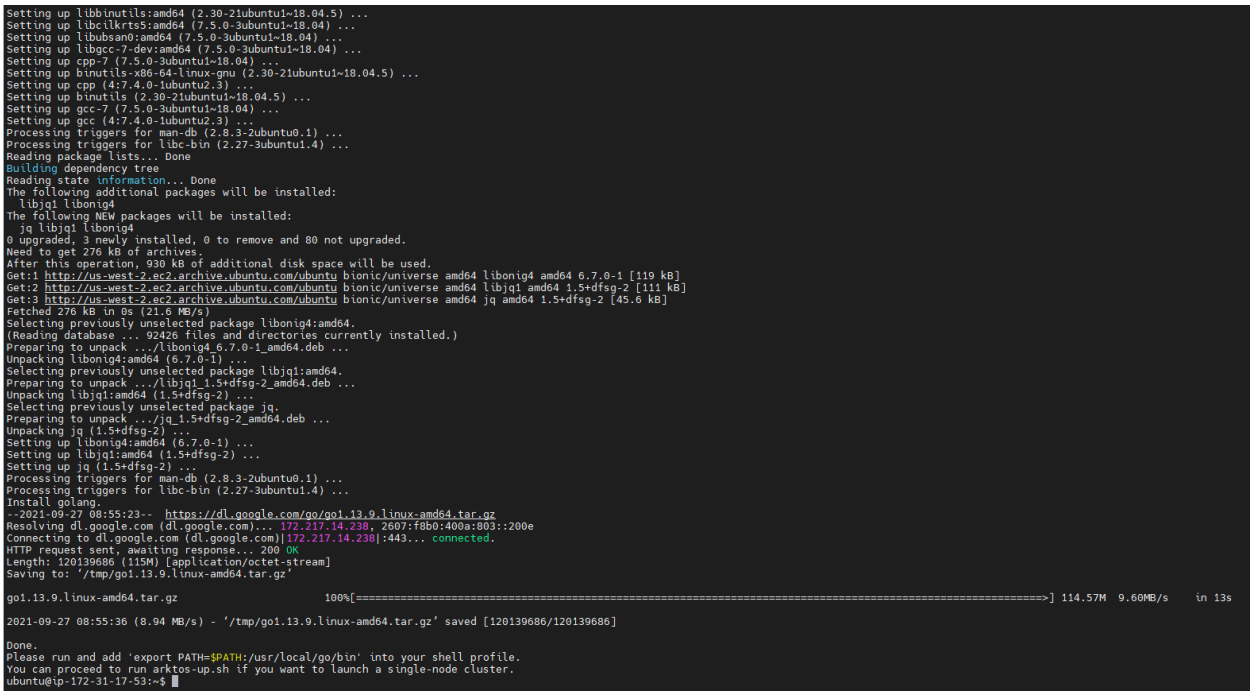
Output



Then installed prerequisites required for Arktos cluster suing following command

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

Output



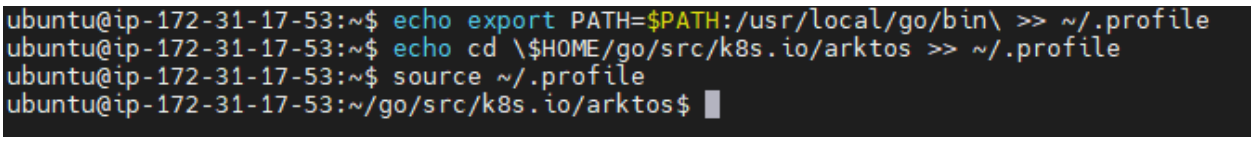
and then run the following commands:

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

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

```
source ~/.profile
```

Output



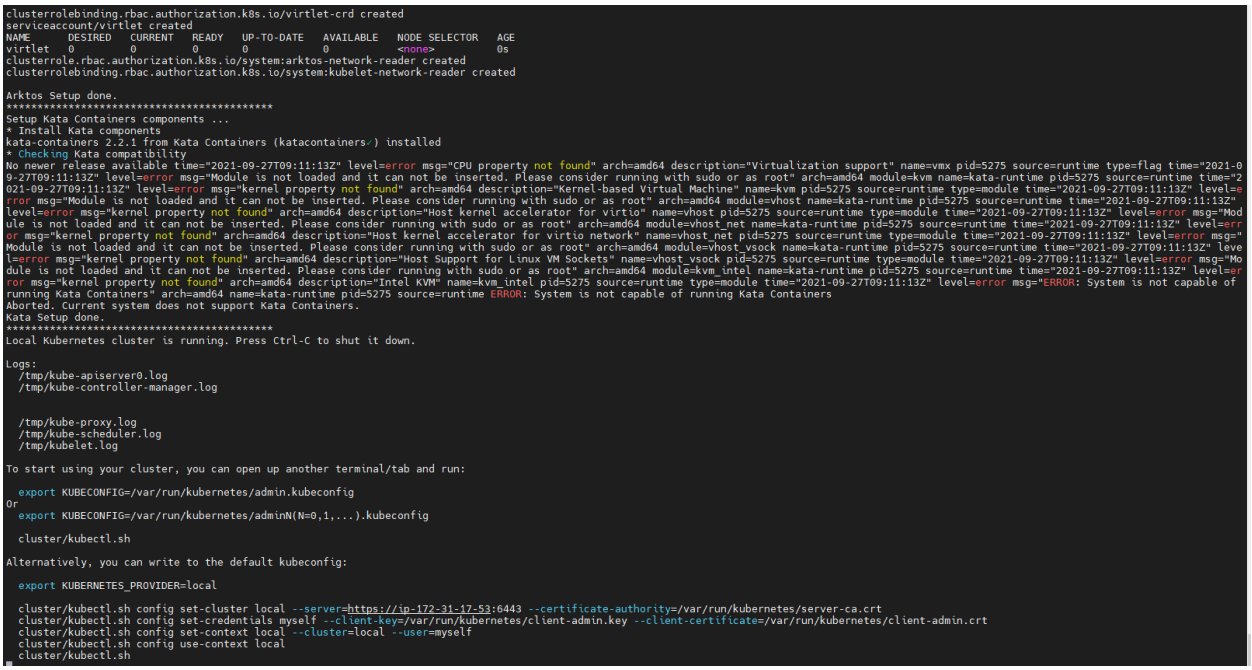
Step-3: Start Arktos cluster

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



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

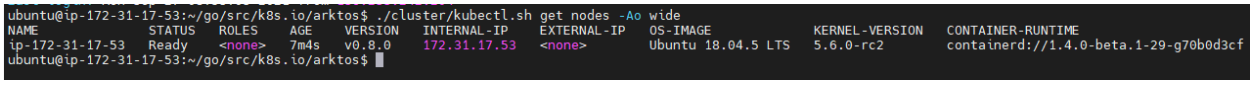
Step-4 Check Cluster health

Open new terminal for same instance and run following commands:

1) Check node status

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

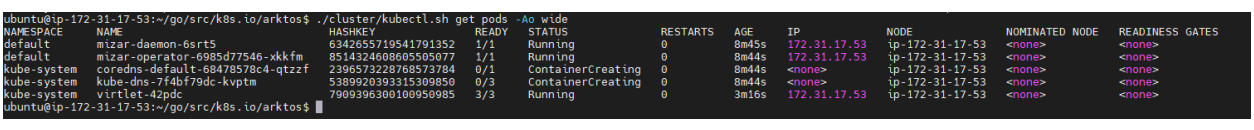
Output



2) Check pods status

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

Output



3) Check vpc status

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

Output

```
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$ ./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     2021-09-27T09:05:55.551525
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$
```

4) Check subnets

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

Output

```
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$ ./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          2021-09-27T09:05:55.618266
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$
```

5) Check net

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

Output

```
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh get net -Ao wide
NAME      TYPE     VPC                        PHASE   DNS
default   mizar    system-default-network     Ready   10.0.0.67
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$
```

6) Check dividers

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

Output

```
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh get dividers -Ao wide
No resources found.
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$
```

7) Check bouncers

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

Output

```
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh get bouncers -Ao wide
No resources found.
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$
```

8) Pod deployment:

Output

```
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh run ubuntu --image=ubuntu
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/ubuntu created
ubuntu@ip-172-31-17-53:~/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-6srt5                  6342655719541791352  1/1     Running             0          32m     172.31.17.53   ip-172-31-17-53   <none>           <none>
default       mizar-operator-6985d77546-xkkfm     8514324608605505077  1/1     Running             0          32m     172.31.17.53   ip-172-31-17-53   <none>           <none>
default       ubuntu-58f49b45d7-xr6xp             7930265081919824615  0/1     ContainerCreating   0          3m31s   <none>         ip-172-31-17-53   <none>           <none>
kube-system   coredns-default-68478578c4-qtzzf    2396573228768573784  0/1     ContainerCreating   0          32m     <none>         ip-172-31-17-53   <none>           <none>
kube-system   kube-dns-7f4bf79dc-kvptm            5389920393315309850  0/3     ContainerCreating   0          32m     <none>         ip-172-31-17-53   <none>           <none>
kube-system   virtlet-42pdc                       7909396300100950985  3/3     Running             0          27m     172.31.17.53   ip-172-31-17-53   <none>           <none>
ubuntu@ip-172-31-17-53:~/go/src/k8s.io/arktos$
```

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