Arktos deployment with Mizar CNI

Prepared On-Premises lab machine with below Configuration

- Processor: x86_64
- Cores: 8
- Memory: 16 GB RAM
- Hard Disk: 128 GB HDD
- Network: One network adapter with active Internet connection
- Operating System: Ubuntu 18.04 LTS 64-bit

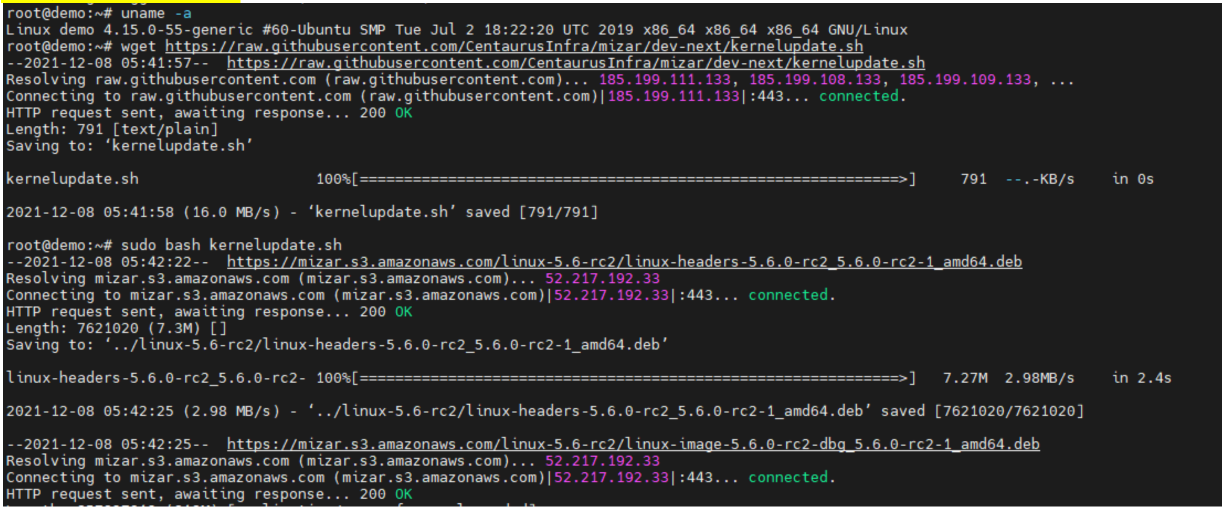
Step-1 Check the kernel version & update the kernel

uname -a

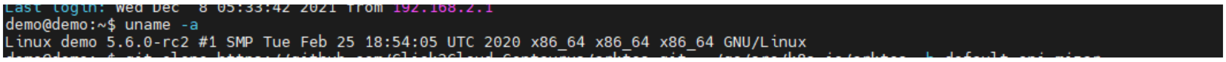
wget https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh sudo

bash kernelupdate.sh

Output



```
root@demo:~# uname -a
Linux demo 4.15.0-55-generic #60-Ubuntu SMP Tue Jul 2 18:22:20 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
root@demo:~# wget https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh
--2021-12-08 05:41:57--  https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|: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

2021-12-08 05:41:58 (16.0 MB/s) - ‘kernelupdate.sh’ saved [791/791]

root@demo:~# sudo bash kernelupdate.sh
--2021-12-08 05:42:22--  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)... 52.217.192.33
Connecting to mizar.s3.amazonaws.com (mizar.s3.amazonaws.com)|52.217.192.33|: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  2.98MB/s    in 2.4s

2021-12-08 05:42:25 (2.98 MB/s) - ‘../linux-5.6-rc2/linux-headers-5.6.0-rc2_5.6.0-rc2-1_amd64.deb’ saved [7621020/7621020]

--2021-12-08 05:42:25--  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)... 52.217.192.33
Connecting to mizar.s3.amazonaws.com (mizar.s3.amazonaws.com)|52.217.192.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
```



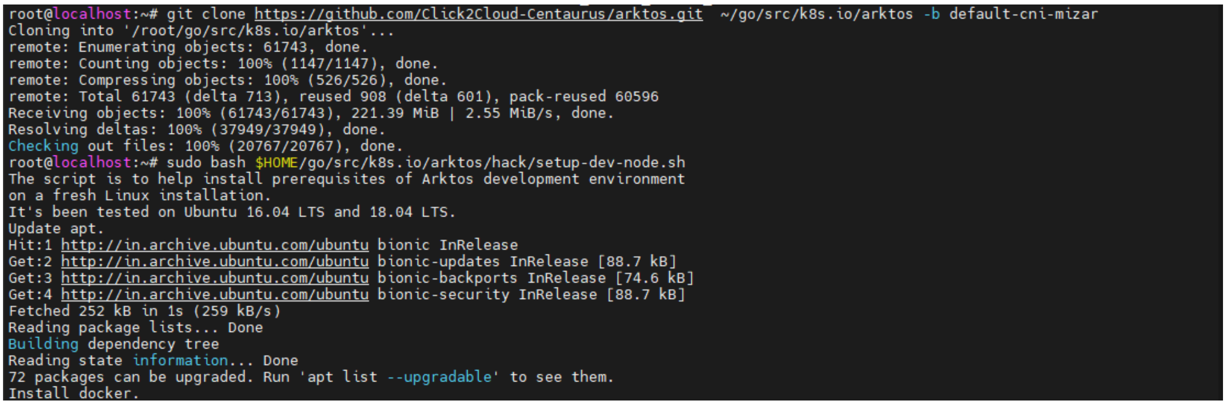
```
Last login: Wed Dec  8 05:33:42 2021 from 192.168.2.1
demo@demo:~$ uname -a
Linux demo 5.6.0-rc2 #1 SMP Tue Feb 25 18:54:05 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

Kernel updated successfully.

Step-2 Clone the Arktos repository and install the required dependencies:

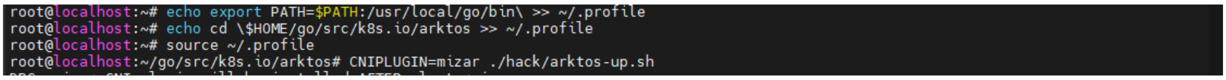
```
git clone https://github.com/Click2Cloud-Centaurus/arktos.git
~/go/src/k8s.io/arktos -b default-cni-mizar
sudo bash $HOME/go/src/k8s.io/arktos/hack/setup-dev-node.sh
```

Output



```
root@localhost:~# 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% (526/526), done.
remote: Total 61743 (delta 713), reused 908 (delta 601), pack-reused 60596
Receiving objects: 100% (61743/61743), 221.39 MiB | 2.55 MiB/s, done.
Resolving deltas: 100% (37949/37949), done.
Checking out files: 100% (20767/20767), done.
root@localhost:~# 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://in.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 252 kB in 1s (259 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
72 packages can be upgraded. Run 'apt list --upgradable' to see them.
Install docker.
```

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



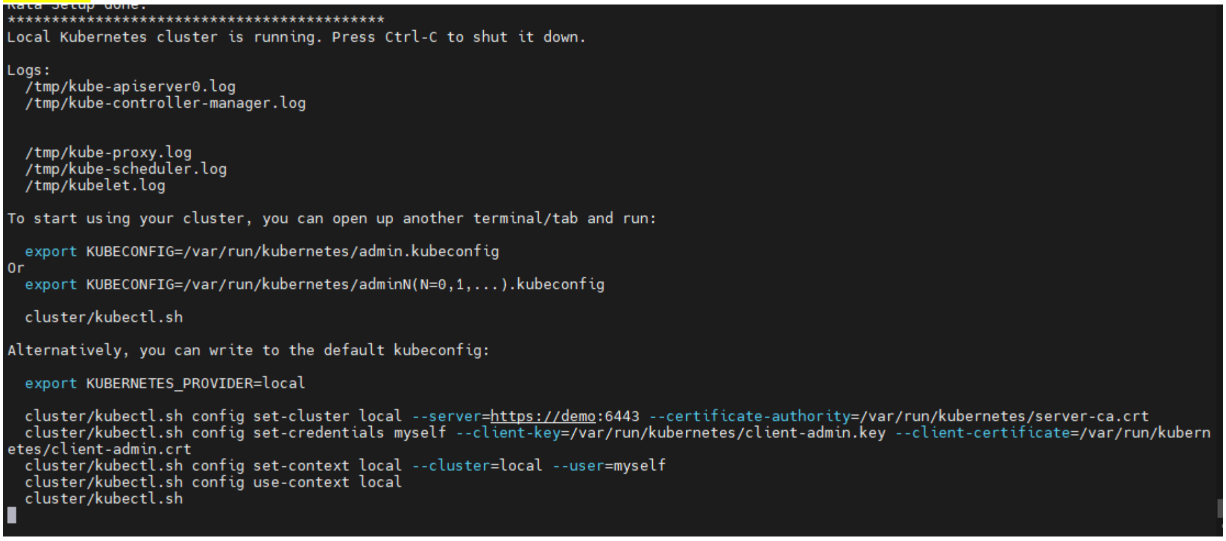
```
root@localhost:~# echo export PATH=$PATH:/usr/local/go/bin\ >> ~/.profile
root@localhost:~# echo cd \$HOME/go/src/k8s.io/arktos >> ~/.profile
root@localhost:~# source ~/.profile
root@localhost:~/go/src/k8s.io/arktos# CNIPLUGIN=mizar ./hack/arktos-up.sh
```

Step-3 Start Arktos cluster

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

Deployment Successfully done.

Output

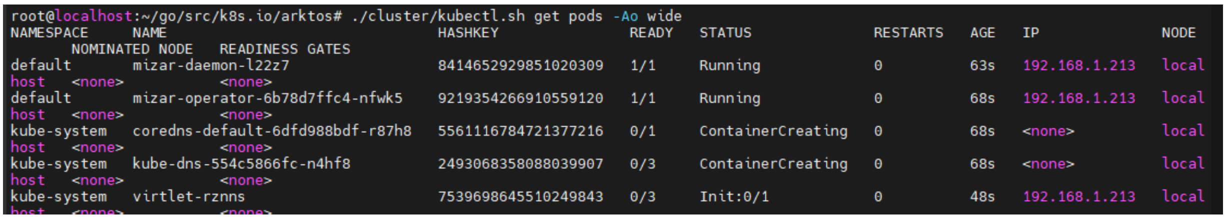


```
Kata setup done.
*******************************************
Local Kubernetes cluster is running. Press Ctrl-C to shut it down.

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://demo: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

1) Check pod status

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



```
root@localhost:~/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-l22z7                    8414652929851020309   1/1     Running             0          63s   192.168.1.213   local
host   <none>           <none>
default       mizar-operator-6b78d7ffc4-nfwk5       9219354266910559120   1/1     Running             0          68s   192.168.1.213   local
host   <none>           <none>
kube-system   coredns-default-6dfd988bdf-r87h8      5561116784721377216   0/1     ContainerCreating   0          68s   <none>          local
host   <none>           <none>
kube-system   kube-dns-554c5866fc-n4hf8             2493068358088039907   0/3     ContainerCreating   0          68s   <none>          local
host   <none>           <none>
kube-system   virtlet-rznns                         7539698645510249843   0/3     Init:0/1            0          48s   192.168.1.213   local
host   <none>           <none>
```

Pods kube-dns and coredns are in container creating state for long time

After re-running the script, we have below outputs

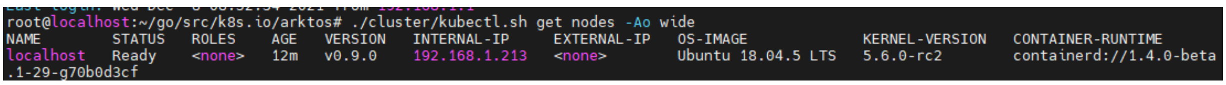
```
CNIPLUGIN=mizar ./hack/arktos-up.sh  -O
```

Step-4 Check Cluster health

Open new terminal for same instance and run following commands:

1) Check node status

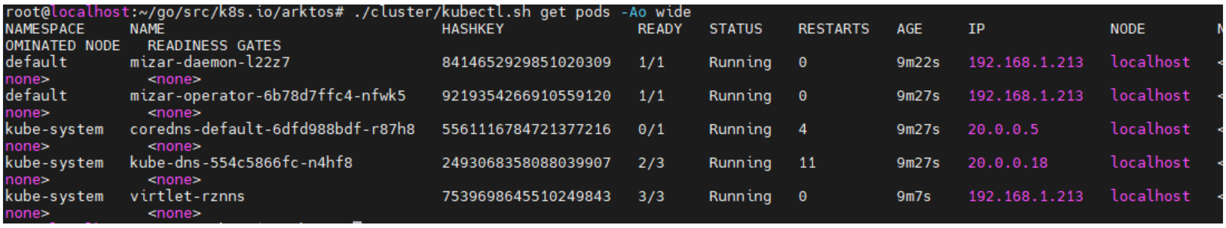
sudo ./cluster/kubectl.sh get nodes -Ao wide



```
root@localhost:~/go/src/k8s.io/arktos# ./cluster/kubectl.sh get nodes -Ao wide
NAME        STATUS   ROLES    AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
localhost   Ready    <none>   12m   v0.9.0    192.168.1.213   <none>        Ubuntu 18.04.5 LTS   5.6.0-rc2        containerd://1.4.0-beta
.1-29-g70b0d3cf
```

2) Check pods status

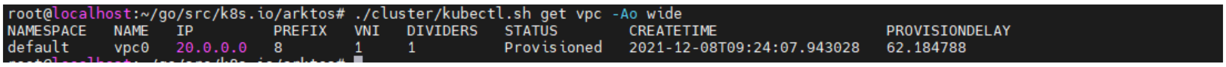
sudo ./cluster/kubectl.sh get pods -Ao wide



```
root@localhost:~/go/src/k8s.io/arktos# ./cluster/kubectl.sh get pods -Ao wide
NAMESPACE     NAME                                  HASHKEY               READY   STATUS    RESTARTS   AGE     IP              NODE        N
OMINATED NODE   READINESS GATES
default       mizar-daemon-l22z7                    8414652929851020309   1/1     Running   0          9m22s   192.168.1.213   localhost   <
none>             <none>
default       mizar-operator-6b78d7ffc4-nfwk5       9219354266910559120   1/1     Running   0          9m27s   192.168.1.213   localhost   <
none>             <none>
kube-system   coredns-default-6dfd988bdf-r87h8      5561116784721377216   0/1     Running   4          9m27s   20.0.0.5        localhost   <
none>             <none>
kube-system   kube-dns-554c5866fc-n4hf8             2493068358088039907   2/3     Running   11         9m27s   20.0.0.18       localhost   <
none>             <none>
kube-system   virtlet-rznns                         7539698645510249843   3/3     Running   0          9m7s    192.168.1.213   localhost   <
none>             <none>
```

3)check vpc status

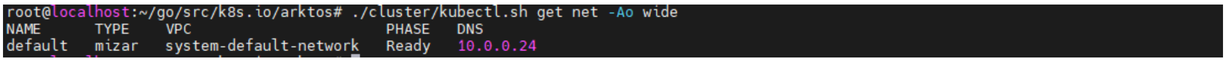
sudo ./cluster/kubectl.sh get vpc -Ao wide



```
root@localhost:~/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          Provisioned   2021-12-08T09:24:07.943028   62.184788
```

4)Check net status

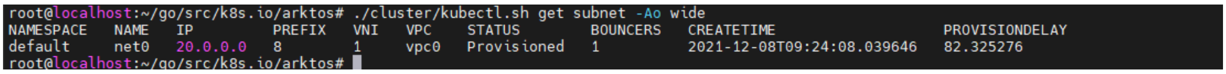
./cluster/kubectl.sh get net -Ao wide



```
root@localhost:~/go/src/k8s.io/arktos# ./cluster/kubectl.sh get net -Ao wide
NAME      TYPE    VPC                     PHASE   DNS
default   mizar   system-default-network  Ready   10.0.0.24
```

5)Check subnet status

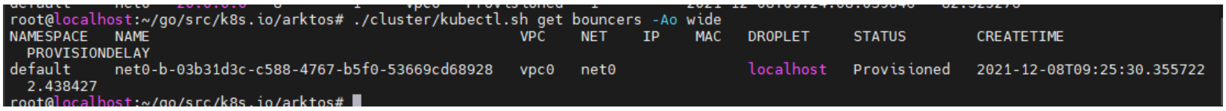
./cluster/kubectl.sh get subnet -Ao wide



```
root@localhost:~/go/src/k8s.io/arktos# ./cluster/kubectl.sh get subnet -Ao wide
NAMESPACE   NAME   IP         PREFIX   VNI   VPC    STATUS        BOUNCERS   CREATETIME                   PROVISIONDELAY
default     net0   20.0.0.0   8        1     vpc0   Provisioned   1          2021-12-08T09:24:08.039646   82.325276
root@localhost:~/go/src/k8s.io/arktos#
```

6)Check bouncers status

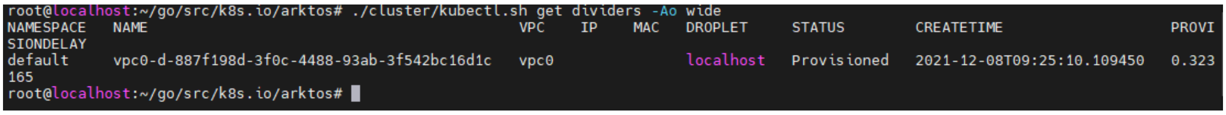
./cluster/kubectl.sh get bouncers -Ao wide



```
root@localhost:~/go/src/k8s.io/arktos# ./cluster/kubectl.sh get bouncers -Ao wide
NAMESPACE   NAME                                           VPC    NET    IP    MAC   DROPLET     STATUS        CREATETIME
  PROVISIONDELAY
default     net0-b-03b31d3c-c588-4767-b5f0-53669cd68928    vpc0   net0               localhost   Provisioned   2021-12-08T09:25:30.355722
  2.438427
root@localhost:~/go/src/k8s.io/arktos#
```

7)Check dividers status

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



```
root@localhost:~/go/src/k8s.io/arktos# ./cluster/kubectl.sh get dividers -Ao wide
NAMESPACE   NAME                                          VPC    IP    MAC   DROPLET     STATUS       CREATETIME                    PROVI
SIONDELAY
default     vpc0-d-887f198d-3f0c-4488-93ab-3f542bc16d1c   vpc0               localhost   Provisioned  2021-12-08T09:25:10.109450    0.323
165
root@localhost:~/go/src/k8s.io/arktos#
```