Arktos deployment without Mizar CNI

This document is intended for new users to install the Arktos platform with Mizar as the underlying network technology.

Prepare lab machine, the preferred OS is **Ubuntu 18.04**. If you are using AWS, the recommended instance size is ```t2.2xlarge``` and the storage size is ```128GB``` or more.

1. Check the kernel version:

Command:

uname -a

Output:

```
Linux ip-172-31-28-71 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
```

Update the kernel if the kernel version is below `5.6.0-rc2`

Commands:

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

sudo bash kernelupdate.sh

uname -a

Output:

```
ubuntu@ip-172-31-28-71:~$ wget https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh
--2021-10-23 13:54:59--  https://raw.githubusercontent.com/CentaurusInfra/mizar/dev-next/kernelupdate.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|: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

2021-10-23 13:54:59 (38.4 MB/s) - 'kernelupdate.sh' saved [791/791]

ubuntu@ip-172-31-28-71:~$ sudo bash kernelupdate.sh
--2021-10-23 13:55:09--  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)... 52.217.203.233
```

```
ubuntu@ip-172-31-28-71:~$ sudo bash kernelupdate.sh
--2021-10-23 13:55:09--  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)... 52.217.203.233
Connecting to mizar.s3.amazonaws.com (mizar.s3.amazonaws.com)|52.217.203.233|: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  6.21MB/s    in 1.2s

2021-10-23 13:55:10 (6.21 MB/s) - '../linux-5.6-rc2/linux-headers-5.6.0-rc2_5.6.0-rc2-1_amd64.deb' saved [7621020/7621020]
```

```
Found linux image: /boot/vmlinuz-5.6.0-rc2
Found initrd image: /boot/initrd.img-5.6.0-rc2
Found linux image: /boot/vmlinuz-5.4.0-1045-aws
Found initrd image: /boot/initrd.img-5.4.0-1045-aws
done
Setting up linux-libc-dev:amd64 (5.6.0-rc2-1) ...
Reboot host (y/n)?y
Rebooting
```

```
ubuntu@ip-172-31-28-71:~$ uname -a
Linux ip-172-31-28-71 5.6.0-rc2 #1 SMP Tue Feb 25 18:54:05 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

2. Clone the Arktos repository and install the required dependencies:

Commands:

git clone https://github.com/Click2Cloud-Centaurus/arktos.git ~/go/src/k8s.io/arktos

cd ~/go/src/k8s.io/arktos

git checkout cni-mizar

sudo bash ./hack/setup-dev-node.sh

Output:

```
ubuntu@ip-172-31-28-71:~$ git clone https://github.com/Click2Cloud-Centaurus/arktos.git  ~/go/src/k8s.io/arktos
Cloning into '/home/ubuntu/go/src/k8s.io/arktos'...
remote: Enumerating objects: 61570, done.
remote: Counting objects: 100% (974/974), done.
remote: Compressing objects: 100% (567/567), done.
remote: Total 61570 (delta 554), reused 614 (delta 390), pack-reused 60596
Receiving objects: 100% (61570/61570), 221.37 MiB | 24.33 MiB/s, done.
Resolving deltas: 100% (37790/37790), done.
Checking out files: 100% (20761/20761), done.
ubuntu@ip-172-31-28-71:~$ cd ~/go/src/k8s.io/arktos
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ git checkout cni-mizar
error: pathspec 'cni-mizar' did not match any file(s) known to git.
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/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.
Update apt.
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [8570 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:6 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic/universe Translation-en [4941 kB]
Get:7 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [151 kB]
Get:8 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic/multiverse Translation-en [108 kB]
```

Command:

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

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

source ~/.profile

Output:

```
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ echo export PATH=$PATH:/usr/local/go/bin\ >> ~/.profile
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ echo cd \$HOME/go/src/k8s.io/arktos >> ~/.profile
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ source ~/.profile
```

3. Start Arktos cluster

Command:

`./hack/arktos-up.sh`

Output:

```
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-172-31-28-71:6443 --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
rnetes/client-admin.crt
  cluster/kubectl.sh config set-context local --cluster=local --user=myself
  cluster/kubectl.sh config use-context local
  cluster/kubectl.sh
```

4. Leave the "arktos-up.sh" terminal and open another terminal to the master node.

Check nodes:

Command:

```
./cluster/kubectl.sh get nodes
```

Output:

```
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh get nodes
NAME              STATUS   ROLES    AGE   VERSION
ip-172-31-28-71   Ready    <none>   21m   v0.9.0
```

Deploy test pods:

Command:

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

Output:

```
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh apply -f  https://raw.githubusercontent.com/Click2Cloud-Centaurus/Documentation/main/test-yamls/test_pods.yaml
pod/netpod1 created
pod/netpod2 created
```

Check deployed pods:

Command:

```
./cluster/kubectl.sh get pods
```

Output:

```
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh get pods
NAME      HASHKEY              READY   STATUS    RESTARTS   AGE
netpod1   8545252777166194393  1/1     Running   0          6m13s
netpod2   4754460051747511963  1/1     Running   0          6m13s
```

Check ping deployed pods:

Command:

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

Output:

```
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh get pods -o wide
NAME      HASHKEY              READY   STATUS    RESTARTS   AGE   IP          NODE              NOMINATED NODE   READINESS GATES
netpod1   8545252777166194393  1/1     Running   0          10m   10.88.0.4   ip-172-31-28-71   <none>           <none>
netpod2   4754460051747511963  1/1     Running   0          10m   10.88.0.5   ip-172-31-28-71   <none>           <none>
```

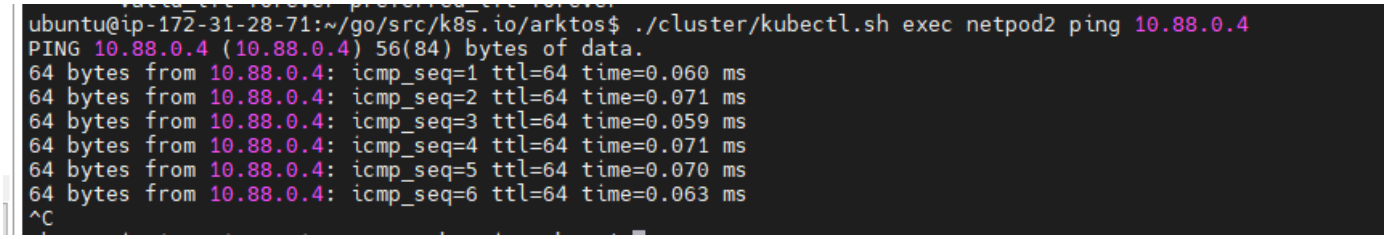
Command:

./cluster/kubectl.sh exec netpod1 ping 10.88.0.5

./cluster/kubectl.sh exec netpod2 ping 10.88.0.4

Output:

```
ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh exec 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=0.085 ms
64 bytes from 10.88.0.5: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 10.88.0.5: icmp_seq=3 ttl=64 time=0.059 ms
64 bytes from 10.88.0.5: icmp_seq=4 ttl=64 time=0.067 ms
64 bytes from 10.88.0.5: icmp_seq=5 ttl=64 time=0.064 ms
^C
```


ubuntu@ip-172-31-28-71:~/go/src/k8s.io/arktos$ ./cluster/kubectl.sh exec 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=0.060 ms
64 bytes from 10.88.0.4: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from 10.88.0.4: icmp_seq=3 ttl=64 time=0.059 ms
64 bytes from 10.88.0.4: icmp_seq=4 ttl=64 time=0.071 ms
64 bytes from 10.88.0.4: icmp_seq=5 ttl=64 time=0.070 ms
64 bytes from 10.88.0.4: icmp_seq=6 ttl=64 time=0.063 ms
^C