

Arktos and Mizar Single Node Installation Guide

Date: 3 Jan 2022

Introduction

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

Installation Steps

- 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

```
cd
```

```
git clone https://github.com/CentaurusInfra/mizar.git
```

```
cd mizar
```

```
chmod 755 setup-machine-arktos.sh
```

```
./setup-machine-arktos.sh
```

```
ubuntu@ip-172-31-25-250:~$ cd
ubuntu@ip-172-31-25-250:~$ git clone https://github.com/CentaurusInfra/mizar.git
Cloning into 'mizar'...
remote: Enumerating objects: 6756, done.
remote: Counting objects: 100% (978/978), done.
remote: Compressing objects: 100% (567/567), done.
remote: Total 6756 (delta 575), reused 713 (delta 390), pack-reused 5778
Receiving objects: 100% (6756/6756), 11.53 MiB | 17.71 MiB/s, done.
Resolving deltas: 100% (4500/4500), done.
ubuntu@ip-172-31-25-250:~$ cd mizar
ubuntu@ip-172-31-25-250:~/mizar$ chmod 755 setup-machine-arktos.sh
ubuntu@ip-172-31-25-250:~/mizar$ ./setup-machine-arktos.sh
Setup: Install go (currently limited to version 1.13.9)
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists...
Reading package lists...
Building dependency tree...
Reading state information...
```

The lab machine will be rebooted once the above script is completed, you will be automatically logged out of the lab machine.

- Log onto your lab machine, then run `bootstrap.sh` script from the Mizar project folder to bootstrap your lab machine.
- Once bootstrap is completed, you can then compile Mizar. Make sure to run these in `sudo` mode:

```
root@ip-172-31-25-250:/home/ubuntu/mizar# sudo bash bootstrap.sh
NOTE: This script will reboot the system if you opt to allow kernel update.
      If reboot is not required, it will log you out and require re-login for new permissions to take effect.

Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python-scapy' instead of 'scapy'
build-essential is already the newest version (12.4ubuntu1)
```

```
cd ~/mizar
```

```
sudo su
```

```
make
```

```
root@ip-172-31-25-250:/home/ubuntu/mizar# make
LDFLAGS=-Llib/usr/lib64 -l:libbpf.a -l:libelf.a -lz -lnsl -static-liblsan -static-libubsan
mkdir -p core
mkdir -p cov
mkdir -p lcov/report
mkdir -p build/bin
mkdir -p build/tests
mkdir -p build/xdp
mkdir -p test/trn_func_tests/output
mkdir -p test/trn_perf_tests/output
git submodule update --init --recursive
DESTDIR=../../lib/ make install -C src/extern/libbpf/src
make[1]: Entering directory '/home/ubuntu/mizar/src/extern/libbpf/src'
if [ ! -d '../../lib/usr/include/bpf' ]; then install -d -m 755 '../../lib/usr/include/bpf'; fi; install
'../../lib/usr/include/bpf'
if [ ! -d '../../lib/usr/lib64/pkgconfig' ]; then install -d -m 755 '../../lib/usr/lib64/pkgconfig'; fi;
r/lib64/pkgconfig'
if [ ! -d '../../lib/usr/lib64' ]; then install -d -m 755 '../../lib/usr/lib64'; fi; cp -fpR ./libbpf.a
../../lib/usr/lib64'
make[1]: Leaving directory '/home/ubuntu/mizar/src/extern/libbpf/src'
python3 -m grpc_tools.protoc -I mizar/proto/ --python_out=. --grpc_python_out=. mizar/proto/mizar/proto/*.proto
/usr/bin/python3: Error while finding module specification for 'grpc_tools.protoc' (ModuleNotFoundError: No module na
mizar/module.mk:11: recipe for target 'proto' failed
make: *** [proto] Error 1
root@ip-172-31-25-250:/home/ubuntu/mizar# python3 -m pip install --user grpcio-tools
Collecting grpcio-tools
  Downloading https://files.pythonhosted.org/packages/55/7a/b6d5a5d69d6ab0df70a7ceed16f0e9a6c0bdc09376c92fa5638d08803
```

Install grpcio tools:

```
python3 -m pip install --user grpcio-tools
```

```
make
```

```

root@ip-172-31-25-250:/home/ubuntu/mizar# make
LDFLAGS=-Llib/usr/lib64 -l:libbpf.a -l:libelf.a -lz -lnsl -static-liblsan -static-libubsan
mkdir -p core
mkdir -p cov
mkdir -p lcov/report
mkdir -p build/bin
mkdir -p build/tests
mkdir -p build/xdp
mkdir -p test/trn_func_tests/output
mkdir -p test/trn_perf_tests/output
git submodule update --init --recursive
DESTDIR=../../lib/ make install -C src/extern/libbpf/src
make[1]: Entering directory '/home/ubuntu/mizar/src/extern/libbpf/src'
if [ ! -d '../../lib//usr/include/bpf' ]; then install -d -m 755 '../../lib//usr/include/bpf'; fi; in
../../lib//usr/include/bpf'
if [ ! -d '../../lib//usr/lib64/pkgconfig' ]; then install -d -m 755 '../../lib//usr/lib64/pkgconfig'
r/lib64/pkgconfig'
if [ ! -d '../../lib//usr/lib64' ]; then install -d -m 755 '../../lib//usr/lib64'; fi; cp -fpR ./libb
../../lib//usr/lib64'
make[1]: Leaving directory '/home/ubuntu/mizar/src/extern/libbpf/src'
python3 -m grpc_tools.protoc -I mizar/proto/ --python_out=. --grpc_python_out=. mizar/proto/mizar/proto/*.proto
protoc --go_out=. --go-grpc_out=. mizar/proto/mizar/proto/interface.proto
G0111MODULE="on" go build cmd/mizarcni/mizarcni.go; mv mizarcni build/bin

```

Build arktos-network-controller (as it is not part of arktos-up.sh yet)

```
cd $HOME/go/src/k8s.io/arktos
```

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

```
make all WHAT=cmd/arktos-network-controller
```

```

root@ip-172-31-25-250:/home/ubuntu/mizar# cd $HOME/go/src/k8s.io/arktos
bash: cd: /root/go/src/k8s.io/arktos: No such file or directory
root@ip-172-31-25-250:/home/ubuntu/mizar# sudo ./hack/setup-dev-node.sh
sudo: ./hack/setup-dev-node.sh: command not found
root@ip-172-31-25-250:/home/ubuntu/mizar# sudo su ubuntu
ubuntu@ip-172-31-25-250:~/mizar$ cd $HOME/go/src/k8s.io/arktos
ubuntu@ip-172-31-25-250:~/go/src/k8s.io/arktos$ sudo ./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
Hit:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-west-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
7 packages can be upgraded. Run 'apt list --upgradable' to see them.
Install docker.
Reading package lists... Done
Building dependency tree

```

Also, please ensure the hostname and its ip address in `/etc/hosts`. For instance, if the hostname is ip-172-31-25-250, ip address is 172.31.25.250:

```

127.0.0.1 localhost
172.31.25.250 ip-172-31-25-250

```

```

!27.0.0.1 localhost
172.31.25.250 ip-172-31-25-250

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
~
~
~
~
~

```

Replace the Arktos containerd:

```
cd $HOME/mizar
```

```
sudo bash replace-containerd.sh
```

```

ubuntu@ip-172-31-25-250:~/go/src/k8s.io/arktos$ cd $HOME/mizar
ubuntu@ip-172-31-25-250:~/mizar$ ./replace-containerd.sh
bash: ./replace-containerd.sh: Permission denied
ubuntu@ip-172-31-25-250:~/mizar$ sudo ./replace-containerd.sh
sudo: ./replace-containerd.sh: command not found
ubuntu@ip-172-31-25-250:~/mizar$ ls
CODE_OF_CONDUCT.md  README.md      cli            cov            go.mod         kernelupdate.sh
CONTRIBUTING.md    appspec.yml    cluster.yaml   deploy-mizar.sh go.sum         kind-setup.sh
LICENSE             bootstrap.sh    cmd           docs           install        lcov
Makefile            build          core          etc            k8s           lib
ubuntu@ip-172-31-25-250:~/mizar$ sudo bash replace-containerd.sh
ubuntu@ip-172-31-25-250:~/mizar$

```

Before deploying Mizar, you will need first start up Arktos API server:

```
cd $HOME/go/src/k8s.io/arktos
```

```
./hack/arktos-up.sh
```

```

ubuntu@ip-172-31-25-250:~/mizar$ cd $HOME/go/src/k8s.io/arktos
ubuntu@ip-172-31-25-250:~/go/src/k8s.io/arktos$ ./hack/arktos-up.sh
DBG: Flannel CNI plugin will be installed AFTER cluster is up
DBG: effective feature gates AllAlpha=false,WorkloadInfoDefaulting=true,QPSDoubleGCController=true,QPSDoubleRSController=true
DBG: effective disabling admission plugins
DBG: effective default network template file is /home/ubuntu/go/src/k8s.io/arktos/hack/testdata/default-flat-network.tm
DBG: kubelet arg RESOLV_CONF is /run/systemd/resolve/resolv.conf
WARNING : The kubelet is configured to not fail even if swap is enabled; production deployments should disable swap.
WARNING : This script MAY be run as root for docker socket / iptables functionality; if failures occur, retry as root.
cni plugin is bridge; arktos will use bridge to provision pod network
Ensuring firewall to allow traffic forward by default
-P FORWARD DROP
-P FORWARD ACCEPT
Ensuring minimum cni plugin installation...
installing cni plugin binaries

```

```

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/kubect1.sh

Alternatively, you can write to the default kubeconfig:

export KUBERNETES_PROVIDER=local
cluster/kubect1.sh config set-cluster local --server=https://ip-172-31-25-250:6443 --certificate-authority=/var/run/kubernet
cluster/kubect1.sh config set-credentials myself --client-key=/var/run/kubernetes/client-admin.key --client-certificate=/var
cluster/kubect1.sh config set-context local --cluster=local --user=myself
cluster/kubect1.sh config use-context local
cluster/kubect1.sh

```

Deploy Mizar. Open a new terminal window, and run:

```
cd $HOME/mizar
```

```
./deploy-mizar.sh
```

```

ubuntu@ip-172-31-25-250:~/go/src/k8s.io/arktos$ cd $HOME/mizar
ubuntu@ip-172-31-25-250:~/mizar$ ./deploy-mizar.sh
[common:check_cluster_ready] Checking cluster readiness by getting node status.
Kubernetes master is running at http://localhost:8080
KubeDNS is running at http://localhost:8080/api/v1/tenants/system/namespaces/kube-system/services/kube-dns
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
customresourcedefinition.apiextensions.k8s.io/bouncers.mizar.com created
customresourcedefinition.apiextensions.k8s.io/dividers.mizar.com created
customresourcedefinition.apiextensions.k8s.io/droplets.mizar.com created
customresourcedefinition.apiextensions.k8s.io/endpoints.mizar.com created
customresourcedefinition.apiextensions.k8s.io/subnets.mizar.com created
customresourcedefinition.apiextensions.k8s.io/vpcs.mizar.com created
configmap/system-source created
clusterrolebinding.rbac.authorization.k8s.io/mizar-operator created
serviceaccount/mizar-operator created
daemonset.apps/mizar-daemon created
Waiting for daemon pod running. It may cost up to 30 minutes because it needs to setup pip3 modules such

```

Once your arktoS server and Mizar are running. To verify, you can open a new terminal and run `kubect1 get nodes`, you should see a node running with the name starts with "IP" followed by the private IP address of your lab machine.

```

ubuntu@ip-172-31-25-250:~/mizar$ kubectl get nodes
NAME                 STATUS    ROLES    AGE     VERSION
ip-172-31-25-250    Ready    <none>   8m30s   v0.9.0
ubuntu@ip-172-31-25-250:~/mizar$

```

You also want make sure the default kubernetes bridge network configuration file is deleted:

```
sudo ls /etc/cni/net.d
```

```
sudo rm /etc/cni/net.d/bridge.conf
```

Start Arktos network controller. From a new terminal window, run:

```
cd $HOME/go/src/k8s.io/arktos
```

```
./_output/local/bin/linux/amd64/arktos-network-controller --  
kubeconfig=/var/run/kubernetes/admin.kubeconfig --kube-apiserver-  
ip=xxx.xxx.xxx.xxx
```

where the `kube-apiserver-ip` is your lab machine's **private ip address**

```
ubuntu@ip-172-31-25-250:~/go/src/k8s.io/arktos$ ./_output/local/bin/linux/amd64/arktos-network-controller --kubeconfig=/var/run/kubernetes/admin.kubeconfig --k  
server-ip=172.31.25.250  
I0103 06:35:40.466832 3447 controller.go:92] starting flat network controller  
I0103 06:35:40.567199 3447 event.go:278] Event(v1.ObjectReference{Kind:"Network", Namespace:"", Name:"default", UID:"7ce8f8a3-bdd9-42a0-8e03-7ac92d3bf90b",  
on:"arktos.futurewei.com/v1", ResourceVersion:"318", FieldPath:""}, Tenant:"system"}): type: 'Normal' reason: 'SuccessfulProvision' successfully provision netw  
tem/default
```

Deploy test pods:

```
kubectctl apply -f https://raw.githubusercontent.com/Click2Cloud-Centaurus/Documentation/main/test-yamls/test\_pods.yaml
```

```
kubectl get pods -A
```

```
ubuntu@ip-172-31-25-250:~/go/src/k8s.io/arktos$ kubectl get pods -A
```

NAMESPACE	NAME	HASHKEY	READY	STATUS	RESTARTS	AGE
default	mizar-daemon-tvn7q	7406420459181898399	1/1	Running	0	36m
default	mizar-operator-64d6f9fc8d-d6g4q	7002844693710336710	1/1	Running	0	34m
default	netpod1	5576472854481464880	0/1	ContainerCreating	0	5m44s
default	netpod2	4686223812115657567	0/1	ContainerCreating	0	5m44s
kube-system	coredns-default-77b95bbf88-64tqk	8987197993154071725	1/1	Running	0	40m
kube-system	kube-dns-554c5866fc-6q4pq	1560218419041683253	3/3	Running	0	40m
kube-system	virtlet-6w875	400997503982034547	3/3	Running	0	40m

```
ubuntu@ip-172-31-25-250:~/go/src/k8s.io/arktos$
```

Pods are getting stuck in **ContainerCreating** state