Arktos and Mizar Single Node Installation Guide (On-Prem)

Date: 4 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**. The recommended instance size is `16 vCPUs, 16 GB RAM`, 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

```
root@master:~# cd
root@master:~# 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 | 8.95 MiB/s, done.
Resolving deltas: 100% (4500/4500), done.
root@master:~# cd
root@master:~# git clone https://github.com/CentaurusInfra/mizar.git
fatal: destination path 'mizar' already exists and is not an empty directory.
root@master:~# cd mizar
root@master:~/mizar# chmod 755 setup-machine-arktos.sh
root@master:~/mizar# ./setup-machine-arktos.sh
Setup: Install go (currently limited to version 1.13.9)
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists...
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
```

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@master:~/mizar# 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

Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
Reading package lists... Done
```

cd ~/mizar

sudo su

Install grpcio tools:

python3 -m pip install --user grpcio-tools

make

```
root@master:~/mizar# python3 -m pip install --user grpcio-tools
Collecting grpcio-tools
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/55/7a/b6d5a5d69d6ab0df70a7ceed16f0e9a6c0bdc09376c92fa5638d088
    100% |████████████████████████████████| 2.2MB 283kB/s
Collecting grpcio>=1.43.0 (from grpcio-tools)
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/c6/6b/5f7cd38ff3ac80f47cbe56618fe45502f90b41a56f5d9e248ee574e
    100% |████████████████████████████████| 21.5MB 49kB/s
Collecting protobuf<4.0dev,>=3.5.0.post1 (from grpcio-tools)
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/c1/12/7479ece04931984162698bfaa05cbb2fc23d7f6ee1ab5146cfc6ade
    100% |████████████████████████████████| 163kB 3.6MB/s
Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (from grpcio-tools)
Requirement already satisfied: six>=1.5.2 in /usr/lib/python3/dist-packages (from grpcio>=1.43.0->grpcio-tools)
Building wheels for collected packages: grpcio-tools, grpcio
  Running setup.py bdist_wheel for grpcio-tools ... done
  Stored in directory: /root/.cache/pip/wheels/85/c7/a5/d41a749ff6bbb1e64684ebf5a6deb86bece79c3285278227f5
  Running setup.py bdist_wheel for grpcio ... done
  Stored in directory: /root/.cache/pip/wheels/b2/98/2d/7aac9507590b235d4ed0ca74e24c91d3c3704cc86416adcd31
Successfully built grpcio-tools grpcio
Installing collected packages: grpcio, protobuf, grpcio-tools
Successfully installed grpcio-1.43.0 grpcio-tools-1.43.0 protobuf-3.19.1
root@master:~/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
```

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@master:~/mizar# cd $HOME/go/src/k8s.io/arktos
root@master:~/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://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
142 packages can be upgraded. Run 'apt list --upgradable' to see them.
Install docker.
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker.io is already the newest version (20.10.7-0ubuntu5~18.04.3).
The following packages were automatically installed and are no longer required:
  accountsservice apport-symptoms bc command-not-found-data libaccountsservice0 python3-a
  python3-debian python3-distro-info python3-gdbm python3-httplib2 python3-hyperlink pyth
```

Also, please ensure the hostname and its ip address in /etc/hosts.

```
127.0.0.1 localhost
192.168.1.210 master

# 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
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Replace the Arktos containerd:

cd $HOME/mizar

sudo bash replace-containerd.sh

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

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

./hack/arktos-up.sh

```
root@master:~/go/src/k8s.io/arktos# cd $HOME/mizar
root@master:~/mizar# sudo bash replace-containerd.sh
root@master:~/mizar#
root@master:~/mizar# cd $HOME/go/src/k8s.io/arktos
root@master:~/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 /root/go/src/k8s.io/arktos/hack/testdata/default-flat-network.tmpl
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.
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/kubectl.sh

Alternatively, you can write to the default kubeconfig:

  export KUBERNETES_PROVIDER=local

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

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

cd $HOME/mizar

./deploy-mizar.sh

```
root@master:~/go/src/k8s.io/arktos# cd $HOME/mizar
root@master:~/mizar# ./deploy-mizar.sh
[common:check_cluster_ready] Checking cluster readyness 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:dns/proxy

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
```

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

```
root@master:~/go/src/k8s.io/arktos# kubectl get nodes
NAME      STATUS     ROLES    AGE      VERSION
master    NotReady   <none>   2m55s    v0.9.0
```

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**

```
root@master:~/mizar# sudo ls /etc/cni/net.d
10-mizarcni.conf  bridge.conf
root@master:~/mizar# sudo rm /etc/cni/net.d/bridge.conf
root@master:~/mizar# cd $HOME/go/src/k8s.io/arktos
root@master:~/go/src/k8s.io/arktos# ./_output/local/bin/linux/amd64/arktos-network-controller --kubeconfig=/var/run/kubernetes/admin
2.168.1.210
I0104 06:41:27.833929   11241 controller.go:92] starting flat network controller
I0104 06:41:27.934681   11241 event.go:278] Event(v1.ObjectReference{Kind:"Network", Namespace:"", Name:"default", UID:"fa56e342-2fc
ion:"arktos.futurewei.com/v1", ResourceVersion:"326", FieldPath:"", Tenant:"system"}): type: 'Normal' reason: 'SuccessfulProvision' s
tem/default
```

Open another terminal:

Deploy test pods:

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

kubectl get pods -A

```
root@master:~/go/src/k8s.io/arktos# kubectl apply -f   https://raw.githubusercontent.com/Click2Cloud-Centaurus/Documentation/m
pod/netpod1 created
pod/netpod2 created
root@master:~/go/src/k8s.io/arktos# kubectl get pods -A
NAMESPACE     NAME                                HASHKEY               READY   STATUS              RESTARTS   AGE
default       mizar-daemon-6k9w4                  4618648239353704249   1/1     Running             0          7m42s
default       mizar-operator-64d6f9fc8d-v7w57     626887559617227703    1/1     Running             0          3m46s
default       netpod1                             1912857985405033468   0/1     ContainerCreating   0          7s
default       netpod2                             6784119082314768547   0/1     ContainerCreating   0          7s
kube-system   coredns-default-54db4cd8c9-gj4j8    8369157905134481102   1/1     Running             0          16m
kube-system   kube-dns-554c5866fc-flvzm           5419777347410370378   3/3     Running             0          16m
kube-system   virtlet-9l6h9                       4525399306379366827   3/3     Running             0          9m59s
root@master:~/go/src/k8s.io/arktos# kubectl get pods -A
NAMESPACE     NAME                                HASHKEY               READY   STATUS              RESTARTS   AGE
default       mizar-daemon-6k9w4                  4618648239353704249   1/1     Running             0          8m36s
default       mizar-operator-64d6f9fc8d-v7w57     626887559617227703    1/1     Running             0          4m40s
default       netpod1                             1912857985405033468   0/1     ContainerCreating   0          61s
default       netpod2                             6784119082314768547   0/1     ContainerCreating   0          61s
kube-system   coredns-default-54db4cd8c9-gj4j8    8369157905134481102   1/1     Running             0          17m
kube-system   kube-dns-554c5866fc-flvzm           5419777347410370378   3/3     Running             0          17m
kube-system   virtlet-9l6h9                       4525399306379366827   3/3     Running             0          10m
root@master:~/go/src/k8s.io/arktos# kubectl get pods -A
```

Pods are getting stuck in ***ContainerCreating*** state