K8s 集群部署（坑贼多）

Author: 申恒恒 Date：Dec 14th, 2018

http://blog.51cto.com/douya/1945382

http://www.cnblogs.com/burningTheStar/p/7865998.html

1. 环境描述：

采用 CentOS7.4 minimual，docker 1.13，kubeadm 1.10.0，etcd 3.0， k8s 1.10.0 我们这里选用两个个节点搭建一个实验环境。

```
192.168.59.133 k8smaster
192.168.59.150 k8snode1
```

2. 准备环境：

1.配置好各节点 hosts 文件

```
cat>> /etc/hosts   << EOF
192.168.59.133 k8smaster
192.168.59.150 k8snode1
EOF
```

2.关闭系统防火墙

```
systemctl stop firewalld && systemctl disable firewalld
```

3. 关闭 SElinux

```
setenforce 0
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
```

重启生效！

4.关闭 swap

```
swapoff -a
```

5.配置系统内核参数

使流过网桥的流量也进入 iptables/netfilter 框架中，在/etc/sysctl.conf 中添加以下配置：

```
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
sysctl -p
```

3. 使用 kubeadm 安装：

1.首先配置阿里 K8S YUM 源

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
EOF
yum -y install epel-release
yum clean all
yum makecache
```

2.安装 kubeadm 和相关工具包

```
yum -y install docker kubelet kubeadm kubectl kubernetes-cni
```

3.启动 Docker 与 kubelet 服务

```
systemctl enable docker && systemctl start docker
systemctl enable kubelet && systemctl start kubelet
```

提示：此时 kubelet 的服务运行状态是异常的，因为缺少主配置文件 kubelet.conf。但可以暂不处理，因为在完成 Master 节点的初始化后才会生成这个配置文件。

4. 下载 K8S 相关镜像

因为无法直接访问 gcr.io 下载镜像，所以需要配置一个国内的容器镜像加速器

配置一个阿里云的加速器：（可省略)

登录 https://cr.console.aliyun.com/

在页面中找到并点击镜像加速按钮，即可看到属于自己的专属加速链接，选择 Centos 版本后即可看到配置方法。

提示：在阿里云上使用 Docker 并配置阿里云镜像加速器，可能会遇到 daemon.json 导致 docker daemon 无法启动的问题，可以通过以下方法解决。

你需要的是编辑

```
vim /etc/sysconfig/docker
```

然后

```
OPTIONS='--selinux-enabled --log-driver=journald --registry-mirror=http://xxxx.mirror.aliyuncs.com'
```

registry-mirror 输入你的镜像地址

最后 service docker restart 重启 daemon

然后 ps aux | grep docker 然后你就会发现带有镜像的启动参数了。

5. 下载 K8S 相关镜像

OK，解决完加速器的问题之后，开始下载 k8s 相关镜像，下载后将镜像名改为 k8s.gcr.io/开头的名字，以便 kubeadm 识别使用。

```
#!/bin/bash
images=(kube-proxy-amd64:v1.10.0 kube-scheduler-amd64:v1.10.0 kube-controller-manager-amd64:v1.10.0 kube-apiserver-amd64:v1.10.0
etcd-amd64:3.1.12 pause-amd64:3.1 kubernetes-dashboard-amd64:v1.8.3 k8s-dns-sidecar-amd64:1.14.8 k8s-dns-kube-dns-amd64:1.14.8
k8s-dns-dnsmasq-nanny-amd64:1.14.8)
for imageName in ${images[@]} ; do
  docker pull keveon/$imageName
  docker tag keveon/$imageName k8s.gcr.io/$imageName
  docker rmi keveon/$imageName
```

```
done
```

上面的 shell 脚本主要做了 3 件事，下载各种需要用到的容器镜像、重新打标记为符合 k8s 命令规范的版本名称、清除旧的容器镜像。

提示：镜像版本一定要和 kubeadm 安装的版本一致，否则会出现 time out 问题。

6. 初始化安装 K8S Master

执行上述 shell 脚本，等待下载完成后，执行 kubeadm init

```
[root@k8smaster ~]# kubeadm init --token=102952.1a7dd4cc8d1f4cc5 --kubernetes-version
1.10.0
. . . . . .
. . . . . .
最后会出现这个信息，在这个信息会非常有用
kubeadm join 192.168.59.133:6443 --token 102952.1a7dd4cc8d1f4cc5
--discovery-token-ca-cert-hash
sha256:6f1a864c6f530351f9ec7a42f74404497fcbe91ad7bf726bffd8cb3e3c333a38
```

提示：选项–kubernetes-version=v1.10.0 是必须的，否则会因为访问 google 网站被墙而无法执行命令。这里使用 v1.10.0 版本，刚才前面也说到了下载的容器镜像版本必须与 K8S 版本一致否则会出现 time out。

上面的命令大约需要 1 分钟的过程，期间可以观察下 tail -f /var/log/message 日志文件的输出，掌握该配置过程和进度。上面最后一段的输出信息保存一份，后续添加工作节点还要用到。

7. 配置 kubectl 认证信息

```
# 对于非 root 用户
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
# 对于 root 用户
export KUBECONFIG=/etc/kubernetes/admin.conf
也可以直接放到~/.bash_profile
```

```
echo "export KUBECONFIG=/etc/kubernetes/admin.conf" >> ~/.bash_profile
```

8. 安装 flannel 网络

```
mkdir -p /etc/cni/net.d/
cat <<EOF> /etc/cni/net.d/10-flannel.conf
{
“name”: “cbr0”,
“type”: “flannel”,
“delegate”: {
“isDefaultGateway”: true
}
}
EOF
mkdir /usr/share/oci-umount/oci-umount.d -p
mkdir /run/flannel/
cat <<EOF> /run/flannel/subnet.env
FLANNEL_NETWORK=10.244.0.0/16
FLANNEL_SUBNET=10.244.1.0/24
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true
EOF
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/v0.9.1/Documentation/kube-flannel.yml
```

9. 让 node1 加入集群

在 node1 节点上分别执行 kubeadm join 命令，加入集群：

```
[root@k8snode1 ~]# kubeadm join 192.168.59.133:6443 --token 102952.1a7dd4cc8d1f4cc5 --discovery-token-ca-cert-hash sha256:6f1a864c6f530351f9ec7a42f74404497fcbe91ad7bf726bffd8cb3e3c333a38
[preflight] Running pre-flight checks.
[WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
```

```
        [WARNING FileExisting-crictl]: crictl not found in system path
    Suggestion: go get github.com/kubernetes-incubator/cri-tools/cmd/crictl
    [discovery] Trying to connect to API Server "10.0.100.202:6443"
    [discovery] Created cluster-info discovery client, requesting info from "https://10.0.100.202:6
443"
    [discovery] Requesting info from "https://10.0.100.202:6443" again to validate TLS against t
he pinned public key
    [discovery] Cluster info signature and contents are valid and TLS certificate validates against
pinned roots, will use API Server "10.0.100.202:6443"
    [discovery] Successfully established connection with API Server "10.0.100.202:6443"
    This node has joined the cluster:
    * Certificate signing request was sent to master and a response
      was received.
    * The Kubelet was informed of the new secure connection details.
    Run 'kubectl get nodes' on the master to see this node join the cluster.
```

提示：细心的童鞋应该会发现，这段命令其实就是前面 K8S Matser 安装成功后我让你们保存的那段命令。

默认情况下，Master 节点不参与工作负载，但如果希望安装出一个 All-In-One 的 k8s 环境，则可以执行以下命令，让 Master 节点也成为一个 Node 节点：

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

10. 验证 K8S Master 是否搭建成功

```
# 查看节点状态
kubectl get nodes
# 查看 pods 状态
kubectl get pods --all-namespaces
# 查看 K8S 集群状态
kubectl get cs
```

11 . 常见错误解析

```
[root@k8smaster ~]# kubectl get nodes
NAME        STATUS     ROLES     AGE       VERSION
k8smaster   NotReady   master    3m        v1.10.1
k8snode1    Ready      <none>    14s       v1.10.1
```

如果出现 NotReady，说明网络问题，需要创建一个网络服务，在这里需要先下载一个文件：

https://github.com/weaveworks/weave/releases

下载

wget

https://github.com/weaveworks/weave/releases/download/v2.3.0/weave-daemonset-k8s-1.7.yaml

然后执行

kubectl create -f weave-daemonset-k8s-1.7.yaml

```
[root@k8smaster ~]# kubectl get nodes
NAME        STATUS    ROLES     AGE       VERSION
k8smaster   Ready     master    11m       v1.10.1
k8snode1    Ready     <none>    8m        v1.10.1
[root@k8smaster ~]#
```

现在可以了！

安装时候最常见的就是 time out，因为 K8S 镜像在国外，所以我们在前面就说到了提前把他下载下来，可以用一个国外机器采用 habor 搭建一个私有仓库把镜像都 download 下来。

```
[root@k8smaster ~]# kubeadm init
[init] Using Kubernetes version: v1.10.0
[init] Using Authorization modes: [Node RBAC]
[preflight] Running pre-flight checks.
   [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
   [WARNING FileExisting-crictl]: crictl not found in system path
Suggestion: go get github.com/kubernetes-incubator/cri-tools/cmd/crictl
[preflight] Starting the kubelet service
[certificates] Generated ca certificate and key.
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names [k8smaster kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.0.100.202]
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] Generated etcd/ca certificate and key.
[certificates] Generated etcd/server certificate and key.
[certificates] etcd/server serving cert is signed for DNS names [localhost] and IPs [127.0.0.1]
[certificates] Generated etcd/peer certificate and key.
[certificates] etcd/peer serving cert is signed for DNS names [k8smaster] and IPs [10.0.100.202]
[certificates] Generated etcd/healthcheck-client certificate and key.
[certificates] Generated apiserver-etcd-client certificate and key.
[certificates] Generated sa key and public key.
[certificates] Generated front-proxy-ca certificate and key.
[certificates] Generated front-proxy-client certificate and key.
[certificates] Valid certificates and keys now exist in "/etc/kubernetes/pki"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager.conf"
```

```
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
[controlplane] Wrote Static Pod manifest for component kube-apiserver to "/etc/kubernetes/manifests/kube-apiserver.yaml"
[controlplane] Wrote Static Pod manifest for component kube-controller-manager to "/etc/kubernetes/manifests/kube-controller-manager.yaml"
[controlplane] Wrote Static Pod manifest for component kube-scheduler to "/etc/kubernetes/manifests/kube-scheduler.yaml"
[etcd] Wrote Static Pod manifest for a local etcd instance to "/etc/kubernetes/manifests/etcd.yaml"
[init] Waiting for the kubelet to boot up the control plane as Static Pods from directory "/etc/kubernetes/manifests".
[init] This might take a minute or longer if the control plane images have to be pulled.
Unfortunately, an error has occurred:
    timed out waiting for the condition
This error is likely caused by:
    - The kubelet is not running
    - The kubelet is unhealthy due to a misconfiguration of the node in some way (required cgroups disabled)
    - Either there is no internet connection, or imagePullPolicy is set to "Never",
      so the kubelet cannot pull or find the following control plane images:
        - k8s.gcr.io/kube-apiserver-amd64:v1.10.0
        - k8s.gcr.io/kube-controller-manager-amd64:v1.10.0
        - k8s.gcr.io/kube-scheduler-amd64:v1.10.0
        - k8s.gcr.io/etcd-amd64:3.1.12 (only if no external etcd endpoints are configured)
If you are on a systemd-powered system, you can try to troubleshoot the error with the following commands:
    - 'systemctl status kubelet'
    - 'journalctl -xeu kubelet'
couldn't initialize a Kubernetes cluster
```

那出现这个问题大部分原因是因为安装的K8S版本和依赖的K8S相关镜像版本不符导致的，关于这部分排错可以查看/var/log/message 我们在文章开始安装的时候也提到了要多看日志。

还有些童鞋可能会说，那我安装失败了，怎么清理环境重新安装啊？下面教大家一条命令：

```
kubeadm reset
```

好了，至此就完成了 K8S 三节点集群的安装部署

The connection to the server localhost:8080 was refused - did you specify the right host or port?

```
sudo cp /etc/kubernetes/admin.conf $HOME/
sudo chown $(id -u):$(id -g) $HOME/admin.conf
export KUBECONFIG=$HOME/admin.conf
```

出现验证错误

```
[root@k8smaster guestbook]# export KUBECONFIG=$HOME/admin.conf
[root@k8smaster guestbook]# kubectl delete pods,service --all
Unable to connect to the server: x509: certificate signed by unknown authority (possibly becaus
 to verify candidate authority certificate "kubernetes")
Unable to connect to the server: x509: certificate signed by unknown authority (possibly becaus
 to verify candidate authority certificate "kubernetes")
[root@k8smaster guestbook]# kube
```

Unsure if this helps, but I had the same and realised I was using the old setup guide, copying /etc/kubernetes/admin.conf into ~/.kube/admin.conf and setting $KUBECONFIG=$HOME/.kube/admin.conf. I cleared the environment variable and kubectl defaults back to using ~/.kube/config.

使用 kubeadm 安装 Kubernetes v1.10 以及常见问题解答上篇文章中介绍了用 kubeadm 安装 Kubernetes1.10，本篇我们来一睹 Kubernetes Dashboard 风采。

老规矩，先介绍下环境：

Kubernetes1.10，Dashboard1.8.3

在 k8s 中 dashboard 可以有两种访问方式：kubeconfig（HTTPS）和 token（http）本篇先来介绍下 Token 方式的访问。

Token 访问是无登录密码的，简单方便

1.下载官方的 dashboard YAML 文件或者下载我的 YAML（无坑版）

```
# 官网版
https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashb
```

```
# 修改版
https://github.com/gh-Devin/kubernetes-dashboard/blob/master/kubernetes-dashboard.yaml
```

```
# 你下载完之后开始修改 YAML 文件，修改内容如下
image: k8s.gcr.io/kubernetes-dashboard-amd64:v1.8.3
```

修改文件里面的镜像为自己可用的镜像，也就是上篇文章使用 kubeadm 安装 Kubernetes v1.10 以及常见问题解答中让你提前下载到本地的 K8S 相关的镜像，这里也要注意镜像版本要一致，不然也会报错。

提示：下载官网版 YAML 需要梯子。

为了避免不必要的报错和坑，建议大家采用我的修改版，里面有 heapster 插件 YAML 和 RBAC YAML。

```
https://github.com/gh-Devin/kubernetes-dashboard
```

2.创建 pod

下载完上面的 YAML 文件之后，开始创建 pod

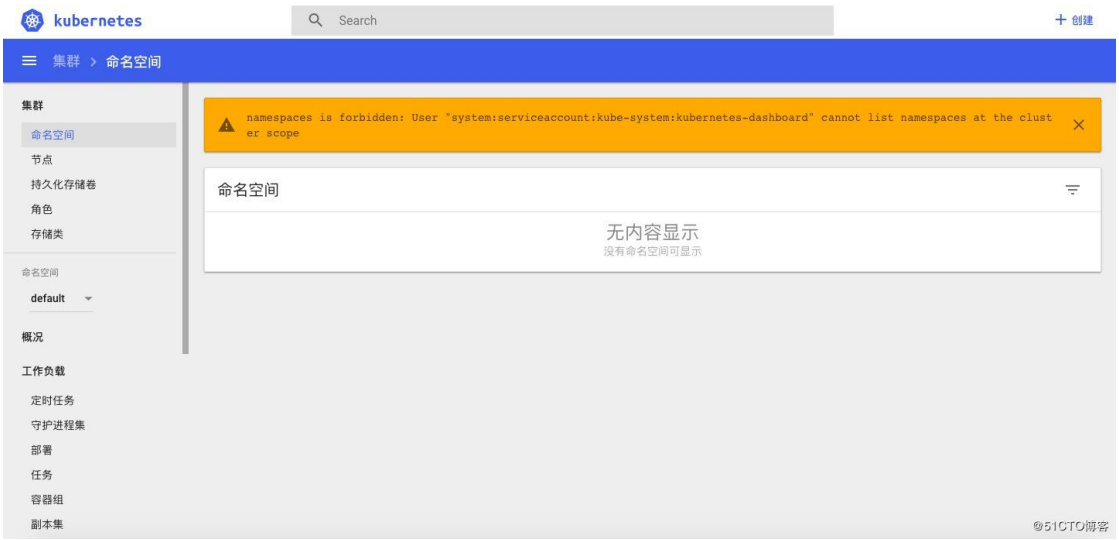
```
[root@k8smaster devin]# ls
heapster-rbac.yaml  heapster.yaml  kubernetes-dashboard-admin.rbac.yaml  kubernetes-dashboard.yaml
[root@k8smaster devin]# kubectl  -n kube-system create -f .
```

3.查看 pod

```
[root@k8smaster devin]# kubectl get svc,pod --all-namespaces | grep dashboard
```

注意：前方高能，报错多发区。

如果采用官方 YAML，就会出现以下报错



这是因为 K8S 在 1.6 版本以后启用了 RBAC 访问控制策略，可以使用 kubectl 或 Kubernetes API 进行配置。使用 RBAC 可以直接授权给用户，让用户拥有授权管理的权限，这样就不再需要直接触碰 Master Node。

下面教你如何解决以上报错问题，如果你下载的是官方版本，修改 kubernetes-dashboard.yaml 文件中的 ServiceAccount 名称

```
[root@k8smaster devin]# vi kubernetes-dashboard.yaml
146 serviceAccountName: kubernetes-dashboard-admin
[root@k8smaster devin]# kubectl apply -f kubernetes-dashboard.yaml -f kubernetes-dashboard-admin.rbac.yaml
```

至此 Dashboard 就可以采用 token 方式来访问，如果没有修改 dashboard YAML 文件里面的 ip 和端口的话可以直接访问 K8S Master IP 地址+30090 即可。

Minikube 安装与配置

本教程适合任何以下操作系统使用：

- Ubuntu - 16.04
- Arch - Manjaro 内核 4.14
- Centos - 7

涵盖了大部分主流的 Linux 操作系统.

具体细节详见：

https://rh01.gitbooks.io/learning-k8s/content/installation/minikube.html

https://github.com/kubernetes/minikube

1. 安装虚拟机

Minikube 的启动参数，大家可以看一下：

Flags:

--apiserver-ips ipSlice A set of apiserver IP Addresses which are used in the generated certificate for localkube/kubernetes. This can be used if you want to make the apiserver available from outside the machine (default [])

--apiserver-name string The apiserver name which is used in the generated certificate for localkube/kubernetes. This can be used if you want to make the apiserver available from outside the machine (default "minikubeCA")

--apiserver-names stringArray A set of apiserver names which are used in the generated certificate for localkube/kubernetes. This can be used if you want to make the apiserver available from outside the machine

--cache-images If true, cache docker images for the current bootstrapper and load them into the machine.

--container-runtime string The container runtime to be used

--cpus int Number of CPUs allocated to the minikube VM (default 2)

--disable-driver-mounts Disables the filesystem mounts provided by the hypervisors (vboxfs, xhyve-9p)

--disk-size string Disk size allocated to the minikube VM (format: <number>[<unit>], where unit = b, k, m or g) (default "20g")

--dns-domain string The cluster dns domain name used in the kubernetes cluster (default "cluster.local")

--docker-env stringArray Environment variables to pass to the Docker daemon. (format: key=value)

--docker-opt stringArray Specify arbitrary flags to pass to the Docker daemon. (format: key=value)

--extra-config ExtraOption A set of key=value pairs that describe configuration that may be passed to different components.

The key should be '.' separated, and the first part before the dot is the component to apply the configuration to.

Valid components are: kubelet, apiserver, controller-manager, etcd, proxy, scheduler.

--feature-gates string A set of key=value pairs that describe feature gates for alpha/experimental features.

-h, --help help for start

--host-only-cidr string The CIDR to be used for the minikube VM (only supported with Virtualbox driver) (default "192.168.99.1/24")

--hyperkit-vpnkit-sock string Location of the VPNKit socket used for networking. If empty, disables Hyperkit VPNKitSock, if 'auto' uses Docker for Mac VPNKit connection, otherwise uses the specified VSock.

--hyperkit-vsock-ports strings List of guest VSock ports that should be exposed as sockets on the host (Only supported on with hyperkit now).

--hyperv-virtual-switch string The hyperv virtual switch name. Defaults to first found. (only supported with HyperV driver)

--insecure-registry strings Insecure Docker registries to pass to the Docker daemon. The default service CIDR range will automatically be added.

--iso-url string Location of the minikube iso (default "https://storage.googleapis.com/minikube/iso/minikube-v0.28.0.iso")

--keep-context This will keep the existing kubectl context and will create a minikube context.

--kubernetes-version string The kubernetes version that the minikube VM will use (ex: v1.2.3)

OR a URI which contains a localkube binary (ex: https://storage.googleapis.com/minikube/k8sReleases/v1.3.0/localkube-linux-amd64) (default "v1.10.0")

--kvm-network string The KVM network name. (only supported with KVM driver) (default "default")

--memory int Amount of RAM allocated to the minikube VM in MB (default 2048)

--mount This will start the mount daemon and automatically mount files into minikube

--mount-string string The argument to pass the minikube mount command on start (default "/home/rh01:/minikube-host")

--network-plugin string The name of the network plugin

--nfs-share strings Local folders to share with Guest via NFS mounts (Only supported on with hyperkit now)

--nfs-shares-root string Where to root the NFS Shares (defaults to /nfsshares, only supported with hyperkit now) (default "/nfsshares")

--registry-mirror strings Registry mirrors to pass to the Docker daemon

--uuid string Provide VM UUID to restore MAC address (only supported with Hyperkit driver).

--vm-driver string VM driver is one of: [virtualbox vmwarefusion kvm xhyve hyperv hyperkit kvm2 none] (default "virtualbox")

--xhyve-disk-driver string The disk driver to use [ahci-hd|virtio-blk] (only supported with xhyve driver) (default "ahci-hd")

Global Flags:

--alsologtostderr log to standard error as well as files

-b, --bootstrapper string The name of the cluster bootstrapper that will set up the kubernetes cluster. (default "kubeadm")

--log_backtrace_at traceLocation when logging hits line file:N, emit a stack trace (default :0)

--log_dir string If non-empty, write log files in this directory

--loglevel int Log level (0 = DEBUG, 5 = FATAL) (default 1)

--logtostderr log to standard error instead of files

-p, --profile string The name of the minikube VM being used.

This can be modified to allow for multiple minikube instances to be run independently (default "minikube")

--stderrthreshold severity logs at or above this threshold go to stderr (default 2)

-v, --v Level log level for V logs

--vmodule moduleSpec comma-separated list of pattern=N settings for file-filtered logging

```
--vm-driver string              VM driver is one of: [virtualbox vmwarefusion kvm xhyve hyperv hyperkit kvm2 none] (default "vi
```

可以看出 minikube 支持虚拟机的选项由 virtualbox，kvm，kvm2,none，...

这里主要讲 **kvm2，virtualbox**

https://github.com/kubernetes/minikube/blob/master/docs/drivers.md

1.1 安装 kvm2

http://blog.programmableproduction.com/2018/03/08/Archlinux-Setup-Minikube-using-KVM/

[Arch]

首先安装虚拟机管理器的驱动和 qemu 的相关库

```
sudo pacman -Sy libvirt qemu-headless ebtables dnsmasq
sudo systemctl enable libvirtd.service
sudo systemctl enable virtlogd.service
sudo systemctl startlibvirtd.service
sudo systemctl startvirtlogd.service
```

然后安装 Docker-machine

```
sudo pacman -Sy docker-machine
```

安装 KVM2

```
yaourt -Sy docker-machine-driver-kvm2
```

重启！！！！否则驱动不生效！

Note: 可能会出现这种情况：

```
E0306 08:05:44.945523      16047 start.go:159] Error starting host: Error starting stopped host: Error creating VM: virError(Code=55, Domain=19, Message='Requested operation is not valid: network 'default' is not active').


Retrying.
E0306 08:05:44.945839      16047 start.go:165] Error starting host:  Error starting stopped host: Error creating VM: virError(Code=55, Domain=19, Message='Requested operation is not valid: network 'default' is not active')================================================================================
```

An error has occurred. Would you like to opt in to sending anonymized crash information to minikube to help prevent future errors?
To opt out of these messages, run the command:
minikube config set WantReportErrorPrompt false==========

Please enter your response [Y/n]:

检查：

[mycomputer]$ sudo virsh net-list --all
[sudo] password for myuser:
Name State Autostart Persistent----------

default inactive no yes
minikube-net active yes yes

然后使 default 自动启动：

sudo virsh net-start default
sudo virsh net-autostart default

再次检查

[mycomputer]$ sudo virsh net-list --all
[sudo] password for myuser:
Name State Autostart Persistent----------

default active yes yes
minikube-net active yes yes

[ubuntu]

# Install libvirt and qemu-kvm on your system, e.g.# Debian/Ubuntu (for older Debian/Ubuntu versions, you may have to use libvirt-bin instead of libvirt-clients and libvirt-daemon-system)
$ sudo apt install libvirt-clients libvirt-daemon-system qemu-kvm
# Add yourself to the libvirt group so you don't need to sudo# NOTE: For older

Debian/Ubuntu versions change the group to `libvirtd`

$ sudo usermod -a -G libvirtd $(whoami)

Update your current session for the group change to take effect# NOTE: For older Debian/Ubuntu versions change the group to `libvirtd`

$ newgrp libvirtd

重启！

[centos]

$ sudo yum install libvirt-daemon-kvm qemu-kvm

Add yourself to the libvirt group so you don't need to sudo# NOTE: For older Debian/Ubuntu versions change the group to `libvirtd`

$ sudo usermod -a -G libvirt $(whoami)

Update your current session for the group change to take effect# NOTE: For older Debian/Ubuntu versions change the group to `libvirtd`

$ newgrp libvirt

重启！！！

但是可能也会出现 Arch 那种情况！！

做法一样！！！

1.2 virtualbox

[arch]

```
sudo pacman -Ss virtualbox
sudo pacman -S linux414-virtualbox-guest-modules
sudo pacman -S linux414-virtualbox-host-modules
sudo pacman -S virtualbox
sudo modprobe vboxdrv
```

重启！！！

[ubuntu]

sudo apt-get install virtualbox

~~[centos]~~

~~sudo yum install -y virtualbox~~

2. 下载 kubectl

   https://kubernetes.io/docs/tasks/tools/install-kubectl/

3. 下载 minikube

   https://rh01.gitbooks.io/learning-k8s/content/installation/minikube.html

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.25.0/minikube-linux-amd64 && chmod +x minikube && sudo mv minikube /usr/local/bin/
# 注意这里的版本，我测试过，如果低版本的化，会有很多限制！
# 比如会要求 iptables 的版本支持..., iptables 1.5.0, 坑！！！
# 最好选择 v.28.0 以上
minikube version
```

4. 启动 minikube && 启动 kube-dashboard

```
minikube start --docker-env http_proxy=http://192.168.31.152:8123 --docker-env https_proxy=http://192.168.31.152:8123 --docker-env no_proxy=localhost,127.0.0.1,::1,192.168.31.0/24,192.168.99.0/24
```

注意：

http_proxy=http://192.168.31.152:8123 和 https_proxy=http://192.168.31.152:8123 的选择是通过

```
minikube ssh
```

登录到 minikube master，查看网关：

```
netstat -r
```

并且需要本地可以翻墙！！！

另一种更好的选择：

```
minikube start --vm-driver=kvm2 --registry-mirror=https://registry.docker-cn.com
```

启动 dashboard：

```
minikube dashboard # 启动，不走代理，这里看到的是 masterip + nodeport
minikube dashboard --url # 查看 url
```

```
kubectl edit svc kubernetes-dashboard --namespace kube-system #  实时修改配置文件
```

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
{"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"labels":{"addonmanager.kubernetes.io/mode":"Reconcile","app":"kubernetes-dashboard","kubernetes.io/minikube-addons":"dashboard","kubernetes.io/minikube-addons-endpoint":"dashboard"},"name":"kubernetes-dashboard","namespace":"kube-system"},"spec":{"ports":[{"nodePort":30000,"port":80,"targetPort":9090}],"selector":{"app":"kubernetes-dashboard"},"type":"NodePort"}}
  creationTimestamp: 2018-10-13T10:47:06Z
  labels:
    addonmanager.kubernetes.io/mode: Reconcile
    app: kubernetes-dashboard
    kubernetes.io/minikube-addons: dashboard
    kubernetes.io/minikube-addons-endpoint: dashboard
  name: kubernetes-dashboard
  namespace: kube-system
  resourceVersion: "40839"
  selfLink: /api/v1/namespaces/kube-system/services/kubernetes-dashboard
  uid: 5400a197-ced5-11e8-a6dc-1052a141641f
spec:
  clusterIP: 10.97.202.195
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 30000
    port: 80
    protocol: TCP
```

```
    targetPort: 9090
  selector:
    app: kubernetes-dashboard
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

这时候，我有一个需求：服务器一般来说是没有显示屏的，如果我不设置代理的话，默认的是打开本地服务器的虚拟 ip 地址，就是说，管理员就不能通过其他终端访问，但是这肯定不行，因此解决这个问题最好的方法就是使用代理，恰恰 minikube 提供了！！

```
kubectl proxy --address='0.0.0.0'
--accept-hosts='^localhost$,^127\.0\.0\.1$,^\[::1\]$,^172\.16\.3\.19$'
```

比如此时我的 master 的 ip 是 172.156.3.19,这样我就可以通过访问下面的地址就可以访问了.

http://172.16.3.19:8001/api/v1/namespaces/kube-system/services/kubernetes-dashboard/proxy/#!/overview?namespace=default