Kubernetes的引导加载代码摘要

main函数, hyperkube结构, Server抽象结构, APIServer, CMServer, SchedulerServer, ProxyServer, KubeletServer实现

1. 在Docker容器里运行Kubernetes v1.0

运行kubernetes大致包括:

- 第一个容器启动etcd服务, etcd是集群配置管理器
- 第二个容器启动kubelet, kubelet是管理POD的核心程序
- 第三个容器启动一个POD, 该POD是kubernetes的master
- 第四个容器启动controller, 是master的核心组件 - 平台控制器
- 第五个容器启动scheduler, 是master的核心组件 - 平台调度器
- 第六个容器启动api-server, 是master的REST api服务接口
- 第七个容器启动proxy，是提供通过外部网络访问平台内应用的网络服务，简单类比，就是为www服务器配置一个Internet地址供用户访问。

在Docker里运行Kubernetes可以降低分布式系统设备的投入. 如果有足够的虚拟机, 可以实验真实的分布式系统.
详细的指南, 可参考kubernetes.io上的文档 - **Running Kubernetes locally via Docker**. 该文档也可以在githu.com上的kubernetes仓库(.../docs/getting-started-guides/docker.md)上找到.

从第二个容器到第六个容器, 是由文档中所述的Step Two: Run the master完成:

```
docker run --net=host --privileged -d -v /sys:/sys:ro -v
/var/run/docker.sock:/var/run/docker.sock
gcr.io/google_containers/hyperkube:v1.0.1 /hyperkube kubelet
--api-servers=http://localhost:8080 --v=2 --address=0.0.0.0 --enable-server
--hostname-override=127.0.0.1 --config=/etc/kubernetes/manifests
```

这是一个docker run命令:

- --net参数为容器使用host模式 (相对于bridge模式)
- --privileged参数为容器解除Linux的AppArmor (Debian系列) 或SELinux (Redhat系列)的访问限制
- -d参数为容器运行在后台模式, 即无控制台,
- -v参数为容器挂在一个卷设备, 这里将docker服务的unix socket挂载到容器里, 目的是使容器运行的kubernetes可以通过socket连接调用Docker的remote api
- gcr.io/google_containers/hyperkube:v0.21.2是容器使用的镜像, 不在docker hub上，而在google cloud platform上 (gcr.io)
- /hyperkube kubelet … 一直到结束，是容器创建成功后，自动执行的程序hyperkube，在容器的/目录下，即是启动kubernetes的master

有关docker run命令可参考Docker的文档 - Docker run reference, 编程参考create api和start api.

2. hyperkube程序

Kyperkube程序由main.go, hyperkube.go, server.go...等作为引导加载代码，在仓库的cmd/hyperkube目录下.

2.1. main函数（main.go)

程序入口：

- 创建hyperkube结构:
  ```
  hk := HyperKube{...}
  ```
- 调用hyperkube的AddServer方法:
  ```
  hk.AddServer(NewKubeAPIServer())  // 创建kuber api server的Server结构，并加入
                                    // Hyperkube结构的private成员server slice
  hk.AddServer(NewKubeControllerManager())  // 同上，创建Controller Manager
  hk.AddServer(NewScheduler())      // 同上，创建Scheduler
  hk.AddServer(NewKubelet())        // 同上，创建Kubelet
  hk.AddServer(NewKubeProxy())      // 同上，创建KuberProxy
  ```
- 调用hyperkube的RunToExit方法
  ```
  hk.RunToExit(os.Args)             // 调用Hyperkube的RunToExit方法，传递命令行参数
  ```

2.2. Hyperkube结构 (hyperkube.go)

Hyperkube结构首先包含两个public成员: Name和Long, 4个private成员: server slice, pflag.FlatSet命令行参数解析导入类型, 打印输出设备out, 和逻辑变量helpFlagVal.

```
type HyperKube struct {
    Name string // The executable name, used for help and soft-link invocation
    Long string // A long description of the binary.  It will be world wrapped before
output.
    servers     []Server
    baseFlags   *pflag.FlagSet
    out         io.Writer
    helpFlagVal bool
}
```

结构提供了如下方法:

- AddServer, 加入一个Server结构到slice成员
- FindServer, 在slice中查找指定name的成员
- Servers, 返回server slice
- Flags, 返回pflag.FlatSet成员
- Out, 返回io.Writer类型的out成员
- SetOut, 设置out成员
- Print, out成员的Print方法间接调用
- Println, 同上
- Printf, 同上

- Run, 运行server的函数，根据前述docker run命令，参数kubelet为服务名，其后为kubelet服务的配置参数, 该Run函数实际调用的是Server结构申明的同名抽象函数，该抽象函数由各具体的Server实例实现
- RunToExit, Run函数的包装，返回控制台, 在(0, 1)取值
- Usage, 命令行输入的帮助函数

2.3. Server结构 (server.go)

结构定义如下：

```
type Server struct {
    SimpleUsage string        // server实例的在线说明.
    Long        string        // server实例提供的详细说明
    Run         serverRunFunc // Run the server.  This is not expected to return.
    flags *pflag.FlagSet      // server启动的命令行解析，见pflag.FlagSet导入类型
    name  string              // server命
    hk    *HyperKube          // server引用的Hyperkube结构
}
```

其中, serverRunFunc是抽象函数, 原型为:

```
type serverRunFunc func(s *Server, args []string) error
```

此外, Server结构提供两个get方法Name和Flags, 以及一个帮助函数Usage

2.4. kube-apiserver.go, kube-controller-manager.go, kube-proxy.go, kube-scheduler.go, kubelet.go

分别创建5个Server结构实例: APIServer，CMServer (Controller Manager), SchedulerServer, ProxyServer, KubeletServer.

3. Kubelet服务程序

Kubelet Server实现了服务启动代码，包括：

- 模块内部常数defaultRootDir = “/var/lib/kubelet”, 是容器内工作用户的家目录
- 运行时类型KubeletServer结构
- 构造函数NewKubeletServer()
- 方法AddFlags, 从命令行解析器获取运行时用户配置或默认配置
- 方法KubeletConfig, 根据运行时配置生成调用Docker remote api的配置
- 方法Run, 启动Kubelet Server, 即在命令行参数--address=0.0.0.0所指定地址和默认端口KubeletPort=10250上启动http服务(ListenAndServe); 配置Kubelet Client, 即命令行参数--api_servers=http://localhost:8080 所指定的api server
- 方法InitializeTLS, 为https初始化TLS证书
- private方法authPathClientConfig, 由证书文件反序列化用户证书信息结构
- private方法kubeconfigClientConfig, 在clientcmd模块
- private方法createClientConfig, 用于处理命令行参数--kubeconfig

- 方法CreateAPIServerClientConfig, 用于创建APIServer的配置
- private方法addChaosToClientConfig
- SimpleKubelet函数, 返回一个通用的KubeletConfig配置结构
- RunKubelet函数, 由KubeletServer运行时结构调用, 也用于第三方封装
- 内部函数startKubelet, 由RunKubelet调用, 实际调用http服务
- 内部函数makePodSourceConfig, 返回POD的配置
- 配置结构KubeletConfig
- KubeletConfig结构的内部方法createAndInitKubelet, 返回KubeletBootstrap和PodConfig
- KubeletBootstrap是初始化kubelet的接口， 详见pkg/kubelet/kubelet.go

3.1. 运行时类型KubeletServer结构成员[1]

- Config, 从命令行参数--config=”/etc/kubernetes/manifests”设置
- SyncFrequency, 由构造函数提供默认值10s
- FileCheckFrequency, 同上, 20s
- HttpCheckFrequency, 同上, 20s
- ManifestURL
- ManifestURLHeader
- EnableServer, 由构造函数提供默认值true
- Address, 同上, 0.0.0.0
- Port, 同上, 10250, 见KubeletPort
- ReadOnlyPort, 同上, 10255, 见KubeletReadOnlyPort
- HostnameOverride, 从命令行参数--hostname_override=”127.0.0.1”设置
- PodInfraContainerImage, 由构造函数提供默认值”gcr.io/google_containers/pause:0.8.0”, 见docker.go模块
- DockerEndpoint
- RootDirectory, 由构造函数提供默认值”/var/lib/kubelet”, 见模块常数defaultWorkDir
- AllowPriveleged, 从命令行参数--privileged设置
- HostNetworkSources, 由构造函数提供默认值”file”, 见types.go模块
- RegistryPullQPS
- RegistryBurst, 由构造函数提供默认值10
- RunOnce, 布尔类型默认值false
- EnableDebuggingHandlers, 由构造函数提供默认值true
- MinimumGCAge, 由构造函数提供默认值1minute
- MaxPerPodContainerCount, 由构造函数提供默认值2
- MaxContainerCount, 由构造函数提供默认值200
- AuthPath, 由构造函数提供默认值“/var/lib/kubelet/kubenetes_auth”
- KubeConfig, 由构造函数提供默认值”/var/lib/kubelet/kubeconfig”
- CadvisorPort, 由构造函数提供默认值4194
- HealthzPort, 由构造函数提供默认值10248
- HealthzBindAddress, 由构造函数提供默认值127.0.0.1

[1] TBD

- OOMScoreAdj, 由构造函数提供默认值-999, 见memory_policy.go模块
- APIServerList, 从命令行参数--api_servers="http://localhost:8080"设置
- RegisterNode, 由构造函数提供默认值true
- StandaloneMode, 布尔类型默认值false
- ClusterDomain
- MasterServiceNamespace, 由构造函数提供默认值"default", 见types.go模块
- ClusterDNS
- StreamConnectionIdelTimeout
- ImageGCHighThresholdPercent, 由构造函数提供默认值90
- ImageGCLowThresholdPercent, 由构造函数提供默认值80
- LowDiskSpaceThresholdMB, 由构造函数提供默认值256
- NetworkPluginName, 由构造函数提供默认值""
- NetworkPluginDir, 由构造函数提供默认/usr/libexec/kubenetes/kuberlet-plugins/net/exec
- CloudProvider
- CloudConfigFile
- TLSCertFile
- TLSPrivateKeyFile
- CertDirectory, 由构造函数提供默认值"/var/run/kubernetes"
- NodeStatusUpdateFrequency, 由构造函数提供默认值10s
- ResourceContainer, 由构造函数提供默认值"/kubelet"
- CgroupRoot, 由构造函数提供默认值""
- ContainerRuntime, 由构造函数提供默认值"docker"
- RktPath, 由构造函数提供默认值""
- DockerDaemonContainer, 由构造函数提供默认值"docker-daemon"
- SystemContainer, 由构造函数提供默认值""
- ConfigCBR0, 由构造函数提供默认值false
- PodCIDR
- MaxPods
- DockerExecHandlerName, 由构造函数提供默认值"native"
- ResolverConfig
- ReallyCrashForTesting
- ChaosChannel
- Containerized

4. API服务程序

API Server启动REST服务, 包括:

- 运行时类型APIServer结构
- 构造函数NewAPIServer
- 方法AddFlags, 从命令行解析器获取运行时用户配置或默认配置
- private方法verifyClusterIPFlag
- 模块内部函数newEtcd
- 方法Run, 启动REST服务

- private方法getRuntimeConfigValue

4.1. 运行时类型KubeletServer结构成员[2]

- InsecureBindAddress net.IP
- InsecurePort int
- BindAddress net.IP
- AdvertiseAddress net.IP
- SecurePort int
- ExternalHost string
- APIRate float32
- APIBurst int
- TLSCertFile string
- TLSPrivateKeyFile string
- CertDirectory string
- APIPrefix string
- ExpAPIPrefix string
- StorageVersion string
- ExpStorageVersion string
- CloudProvider string
- CloudConfigFile string
- EventTTL time.Duration
- BasicAuthFile string
- ClientCAFile string
- TokenAuthFile string
- OIDCIssuerURL string
- OIDCClientID string
- OIDCCAFile string
- OIDCUsernameClaim string
- ServiceAccountKeyFile string
- ServiceAccountLookup bool
- KeystoneURL string
- AuthorizationMode string
- AuthorizationPolicyFile string
- AdmissionControl string
- AdmissionControlConfigFile string
- EtcdServerList []string
- EtcdConfigFile string
- EtcdPathPrefix string
- CorsAllowedOriginList []string
- AllowPrivileged bool

[2] TBD

- ServiceClusterIPRange net.IPNet // TODO: make this a list
- ServiceNodePortRange util.PortRange
- EnableLogsSupport bool
- MasterServiceNamespace string
- RuntimeConfig util.ConfigurationMap
- KubeletConfig client.KubeletConfig
- ClusterName string
- EnableProfiling bool
- MaxRequestsInFlight int
- MinRequestTimeout int
- LongRunningRequestRE string
- SSHUser string
- SSHKeyfile string
- MaxConnectionBytesPerSec int64

5. Controller Manager服务程序

CM Server启动控制器, 包括:

- 运行时类型CMServer结构
- 构造函数NewCMServer
- 方法AddFlags
- 方法Run

5.1. 运行时类型CMServer结构成员变量[3]

- Port int
- Address net.IP
- CloudProvider string
- CloudConfigFile string
- ConcurrentEndpointSyncs int
- ConcurrentRCSyncs int
- ServiceSyncPeriod time.Duration
- NodeSyncPeriod time.Duration
- ResourceQuotaSyncPeriod time.Duration
- NamespaceSyncPeriod time.Duration
- PVClaimBinderSyncPeriod time.Duration
- HorizontalPodAutoscalerSyncPeriod time.Duration
- RegisterRetryCount int
- NodeMonitorGracePeriod time.Duration
- NodeStartupGracePeriod time.Duration
- NodeMonitorPeriod time.Duration
- NodeStatusUpdateRetry int

[3] TBD

- PodEvictionTimeout time.Duration
- DeletingPodsQps float32
- DeletingPodsBurst int
- ServiceAccountKeyFile string
- RootCAFile string
- ClusterName string
- ClusterCIDR net.IPNet
- AllocateNodeCIDRs bool
- EnableProfiling bool
- EnableHorizontalPodAutoscaler bool
- Master string
- Kubeconfig string

6. Scheduler服务程序

Scheduler Server启动调度器, 包括:

- 运行时类型Scheduler结构
- 构造函数NewSchedulerServer
- 方法AddFlags
- 方法Run
- private方法createConfig

6.1. 运行时类型SchedulerServer结构成员变量[4]

- Port int
- Address net.IP
- AlgorithmProvider string
- PolicyConfigFile string
- EnableProfiling bool
- Master string
- Kubeconfig string
- BindPodsQPS float32
- BindPodsBurst int

7. Proxy服务程序

Proxy Server启动外部网络代理, 包括:

- 运行时类型ProxyServer结构
- 构造函数NewProxyServer
- 方法AddFlags
- 方法Run
- private方法birthCry

[4] TBD

7.1. 运行时类型ProxyServer结构成员变量

- [5]BindAddress net.IP
- HealthzPort int
- HealthzBindAddress net.IP
- OOMScoreAdj int
- ResourceContainer string
- Master string
- Kubeconfig string
- PortRange util.PortRange
- Recorder record.EventRecorder
- HostnameOverride string
- ForceUserspaceProxy bool
- SyncPeriod time.Duration
- nodeRef *api.ObjectReference // Reference to this node.
- MasqueradeAll bool
- CleanupAndExit bool

[5] TBD