docker守护程序创建容器

github.com/docker/docker-ce/blob/master/components/engine/daemon/container.go

图形用户界面, 文本, 应用程序

描述已自动生成

因此，创建ID和名称的逻辑是在generateIDAndName函数中:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

如下，该值是一个随机值

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

cmd/kube-apiserver/apiserver.go:41

func main() {

// 保证每次运行时生成的随机数不一样

rand.Seed(time.Now().UnixNano())

pflag.CommandLine.SetNormalizeFunc(cliflag.WordSepNormalizeFunc)

command := app.NewAPIServerCommand()

logs.InitLogs()

defer logs.FlushLogs()

if err := command.Execute(); err != nil {

os.Exit(1)

}

}

cmd/kube-apiserver/app/server.go:142

func NewAPIServerCommand() \*cobra.Command {

s := options.NewServerRunOptions()

cmd := &cobra.Command{

Use: "kube-apiserver",

Long: `The Kubernetes API server validates and configures data

for the api objects which include pods, services, replicationcontrollers, and

others. The API Server services REST operations and provides the frontend to the

cluster's shared state through which all other components interact.`,

// stop printing usage when the command errors

SilenceUsage: true,

PersistentPreRunE: func(\*cobra.Command, []string) error {

// silence client-go warnings.

// kube-apiserver loopback clients should not log self-issued warnings.

rest.SetDefaultWarningHandler(rest.NoWarnings{})

return nil

},

RunE: func(cmd \*cobra.Command, args []string) error {

verflag.PrintAndExitIfRequested()

fs := cmd.Flags()

cliflag.PrintFlags(fs)

err := checkNonZeroInsecurePort(fs)

if err != nil {

return err

}

// set default options

completedOptions, err := Complete(s)

if err != nil {

return err

}

// validate options

if errs := completedOptions.Validate(); len(errs) != 0 {

return utilerrors.NewAggregate(errs)

}

return Run(completedOptions, genericapiserver.SetupSignalHandler())

},

Args: func(cmd \*cobra.Command, args []string) error {

for \_, arg := range args {

if len(arg) > 0 {

return fmt.Errorf("%q does not take any arguments, got %q", cmd.CommandPath(), args)

}

}

return nil

},

}

. . .

return cmd

}

cmd/kube-apiserver/app/server.go:174

func Run(completeOptions completedServerRunOptions, stopCh <-chan struct{}) error {

// To help debugging, immediately log version

klog.Infof("Version: %+v", version.Get())

server, err := CreateServerChain(completeOptions, stopCh)

if err != nil {

return err

}

prepared, err := server.PrepareRun()

if err != nil {

return err

}

return prepared.Run(stopCh)

}

cmd/kube-apiserver/app/server.go:210

func CreateServerChain(completedOptions completedServerRunOptions, stopCh <-chan struct{}) (\*aggregatorapiserver.APIAggregator, error) {

. . .

kubeAPIServer, err := CreateKubeAPIServer(kubeAPIServerConfig, apiExtensionsServer.GenericAPIServer)

if err != nil {

return nil, err

}

. . .

return aggregatorServer, nil

}

cmd/kube-apiserver/app/server.go:231

func CreateKubeAPIServer(kubeAPIServerConfig \*controlplane.Config, delegateAPIServer genericapiserver.DelegationTarget) (\*controlplane.Instance, error) {

kubeAPIServer, err := kubeAPIServerConfig.Complete().New(delegateAPIServer)

if err != nil {

return nil, err

}

return kubeAPIServer, nil

}

pkg/controlplane/instance.go:414

func (c completedConfig) New(delegationTarget genericapiserver.DelegationTarget) (\*Instance, error) {

. . .

// install legacy rest storage

if c.ExtraConfig.APIResourceConfigSource.VersionEnabled(apiv1.SchemeGroupVersion) {

legacyRESTStorageProvider := corerest.LegacyRESTStorageProvider{

StorageFactory: c.ExtraConfig.StorageFactory,

ProxyTransport: c.ExtraConfig.ProxyTransport,

KubeletClientConfig: c.ExtraConfig.KubeletClientConfig,

EventTTL: c.ExtraConfig.EventTTL,

ServiceIPRange: c.ExtraConfig.ServiceIPRange,

SecondaryServiceIPRange: c.ExtraConfig.SecondaryServiceIPRange,

ServiceNodePortRange: c.ExtraConfig.ServiceNodePortRange,

LoopbackClientConfig: c.GenericConfig.LoopbackClientConfig,

ServiceAccountIssuer: c.ExtraConfig.ServiceAccountIssuer,

ExtendExpiration: c.ExtraConfig.ExtendExpiration,

ServiceAccountMaxExpiration: c.ExtraConfig.ServiceAccountMaxExpiration,

APIAudiences: c.GenericConfig.Authentication.APIAudiences,

}

if err := m.InstallLegacyAPI(&c, c.GenericConfig.RESTOptionsGetter, legacyRESTStorageProvider); err != nil {

return nil, err

}

}

. . .

return m, nil

}

pkg/controlplane/instance.go:537

func (m \*Instance) InstallLegacyAPI(c \*completedConfig, restOptionsGetter generic.RESTOptionsGetter, legacyRESTStorageProvider corerest.LegacyRESTStorageProvider) error {

legacyRESTStorage, apiGroupInfo, err := legacyRESTStorageProvider.NewLegacyRESTStorage(restOptionsGetter)

if err != nil {

return fmt.Errorf("error building core storage: %v", err)

}

controllerName := "bootstrap-controller"

coreClient := corev1client.NewForConfigOrDie(c.GenericConfig.LoopbackClientConfig)

bootstrapController := c.NewBootstrapController(legacyRESTStorage, coreClient, coreClient, coreClient, coreClient.RESTClient())

m.GenericAPIServer.AddPostStartHookOrDie(controllerName, bootstrapController.PostStartHook)

m.GenericAPIServer.AddPreShutdownHookOrDie(controllerName, bootstrapController.PreShutdownHook)

if err := m.GenericAPIServer.InstallLegacyAPIGroup(genericapiserver.DefaultLegacyAPIPrefix, &apiGroupInfo); err != nil {

return fmt.Errorf("error in registering group versions: %v", err)

}

return nil

}

pkg/registry/core/rest/storage\_core.go:262

func (c LegacyRESTStorageProvider) NewLegacyRESTStorage(restOptionsGetter generic.RESTOptionsGetter) (LegacyRESTStorage, genericapiserver.APIGroupInfo, error) {

. . .

serviceRest, serviceRestProxy := servicestore.NewREST(serviceRESTStorage,

endpointsStorage,

podStorage.Pod,

serviceClusterIPAllocator,

secondaryServiceClusterIPAllocator,

serviceNodePortAllocator,

c.ProxyTransport)

. . .

return restStorage, apiGroupInfo, nil

}