**新增API的过程**

1. **新增API的过程**

以新增一个Snapshot API为例：

1. **定义API结构体**

编辑pkg/api/types.go和pkg/api/v1/types.go,添加API的结构体定义

|  |
| --- |
| type SnapshotList struct {  TypeMeta `json:",inline"`  ListMeta `json:"metadata,omitempty"`  Items []Snapshot `json:"items" description:"list of imagecommits"`  }  type Snapshot struct {  TypeMeta `json:",inline"`  ObjectMeta `json:"metadata,omitempty"`  Spec SnapshotSpec `json:"spec,omitempty"`  Status SnapshotStatus `json:"status,omitempty"`  }  type SnapshotSpec struct {  Finalizers []FinalizerName `json:"finalizers,omitempty"`  Volumes []string `json:"volumes"`  RestartPolicy RestartPolicy `json:"restartPolicy,omitempty"`  PodName string `json:"podName,omitempty"`  NodeName string `json:"nodeName,omitempty"`  }  type SnapshotStatus struct {  VolumeSnapshotStatuses []VolumeSnapshotStatus `json:"volumeSnapshotStatuses,omitempty"`  Message string `json:"message,omitempty"`  }  const (  FinalizerNBSSnapshot FinalizerName = "NBSSnapshot"  )  type SnapshotPhase string  const (  SnapshotPending SnapshotPhase = "Pending"  SnapshotRunning SnapshotPhase = "Running"  SnapshotPushing SnapshotPhase = "Pushing"  SnapshotSucceeded SnapshotPhase = "Succeeded"  SnapshotFailed SnapshotPhase = "Failed"  // NamespaceTerminating means the namespace is undergoing graceful termination  SnapshotTerminating SnapshotPhase = "Terminating"  SnapshotDeleteFailed SnapshotPhase = "DeleteFailed"  )  type VolumeSnapshotStatus struct {  VolumeName string `json:"volumeName"`  VolumeID string `json:"volumeID"`  SnapshotID string `json:"snapshotID"`  Phase SnapshotPhase `json:"phase"`  RestartCount int `json:"restartCount,omitempty"`  } |

1. **注册API**

编辑pkg/api/register.go和pkg/api/v1/register.go, 注册API

|  |
| --- |
| func init() {  Scheme.AddKnownTypes("",  //add-by-lmq 2015-04-28 disk-snapshot start  &Snapshot{},  &SnapshotList{},  //add-by-lmq 2015-04-28 disk-snapshot end  )  }  //add-by-lmq 2015-04-28 disk-snapshot start  func (\*Snapshot) IsAnAPIObject() {}  func (\*SnapshotList) IsAnAPIObject() {}  //add-by-lmq 2015-04-28 disk-snapshot end |

1. **生成api之间的编码解码函数**

运行/hack/update-generated-conversion.go和/hack/update-generated-deep-copies.go, 生成API多个版本间的转换函数即可

1. **实现API Server写Etcd时的各个函数**

在/pkg/registry/下添加一个snapshot文件夹，实现snapshot相关的各个函数即可

* pkg/registry/snapshot/etcd/etcd.go里是API server在CRUD一个资源时会调用的函数
* pkg/registry/snapshot/etcd/rest.go里一般放一些validation和filter的函数

1. **注册路由**

编辑pkg/master/master.go，注册路由

1. **添加client**

仿照其他资源，在pkg/client下添加该资源的client

1. **添加list watch的cache相关函数**

如果有组件需要list&watch这个资源，需要在pkg/client/cache/lister.go里添加cache相关的一些结构体