Introduction to controller pattern

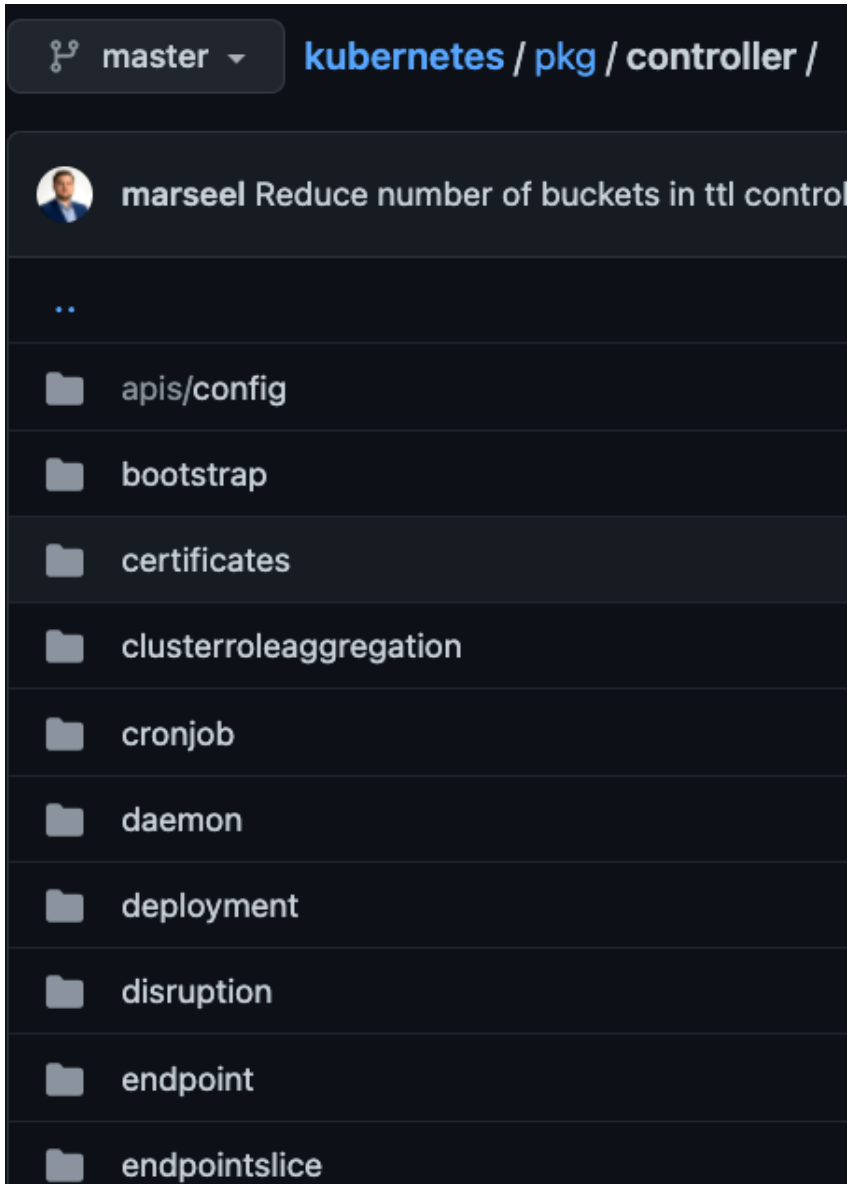
莊家雋

Outline

- Informer in Kubernetes
- Event handler

Controller Pattern

- Controller 使用api-server 的 watch API 收取 etcd 裡資源的期望配置，和收集到的實際配置做對比並修正差異
  - K8S有許多內建的Controller，例: deployment
- Controller是個無窮迴圈，不斷的取得資源的狀態
  - 要如何有效率的完成這件工作
- 當資源的某個事件發生時，透過Controller來實現自定的邏輯



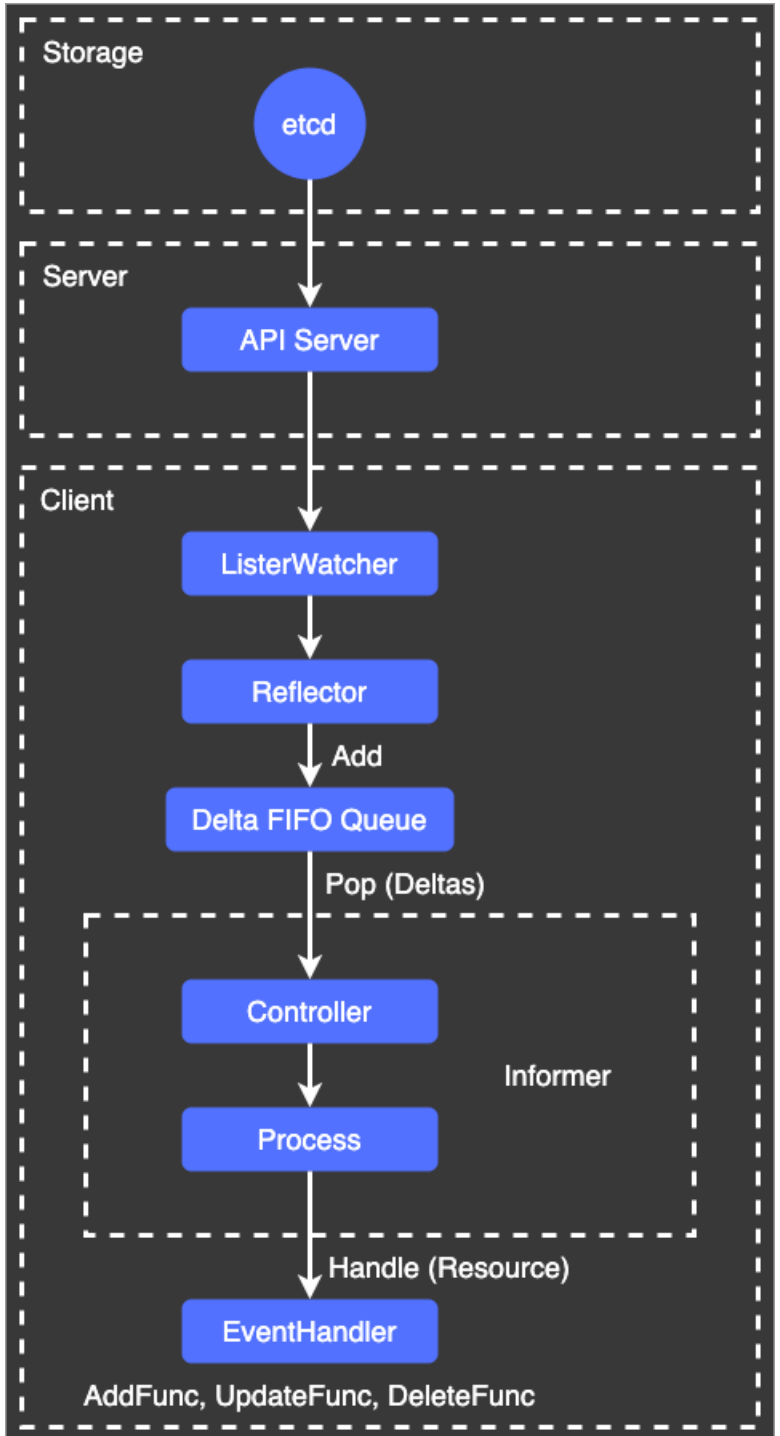
https://tw.wbsnail.com/p/dive-into-kubernetes-informer

Informer

- Controller是透過informer實現高效、低延遲
- Informer
  - objectType: 一種Type要有一個informer
  - ListerWatcher: 才能由API-Server取得資源狀態
  - ResourceEventHandler: 事件發生時如何處理
  - Indexer: 依Key查詢資源，預設為NamespaceIndex
  - resyncPeriod: 多久和APIServer 同步

```go
func NewIndexerInformer(
    lw ListerWatcher,
    objType runtime.Object,
    resyncPeriod time.Duration,
    h ResourceEventHandler,
    indexers Indexers,
) (Indexer, Controller) {
    // This will hold the client state, as we know it.
    clientState := NewIndexer(DeletionHandlingMetaNamespaceKeyFunc, indexers)

    return clientState, newInformer(lw, objType, resyncPeriod, h, clientState)
}
```



建立 informer

- 每種resource有一個對應的informer處理
- 利用factory pattern 建立informer
  - 建立工廠
  - 利用工廠產生informer

```go
func NewConfigMapController(client *kubernetes.Clientset) *ConfigMapController {
    factory := informers.NewSharedInformerFactoryWithOptions(client, 5*time.Second, informers.WithNamespace(namespace))
    informer := factory.Batch().V1().Jobs()

    c := &ConfigMapController{
        informerFactory: factory,
        informer:        informer,
        clientSet:       client,
    }
    informer.Informer().AddEventHandler(
        // Your custom resource event handlers.
        cache.ResourceEventHandlerFuncs{
            // Called on creation
            AddFunc: c.onAdd,
            // Called on resource update and every resyncPeriod on existing resources.
            UpdateFunc: c.onUpdate,
            // Called on resource deletion.
            DeleteFunc: c.onDelete,
        },
    )
    return c
}
```

https://tw.wbsnail.com/p/dive-into-kubernetes-informer-shared-informer-factory

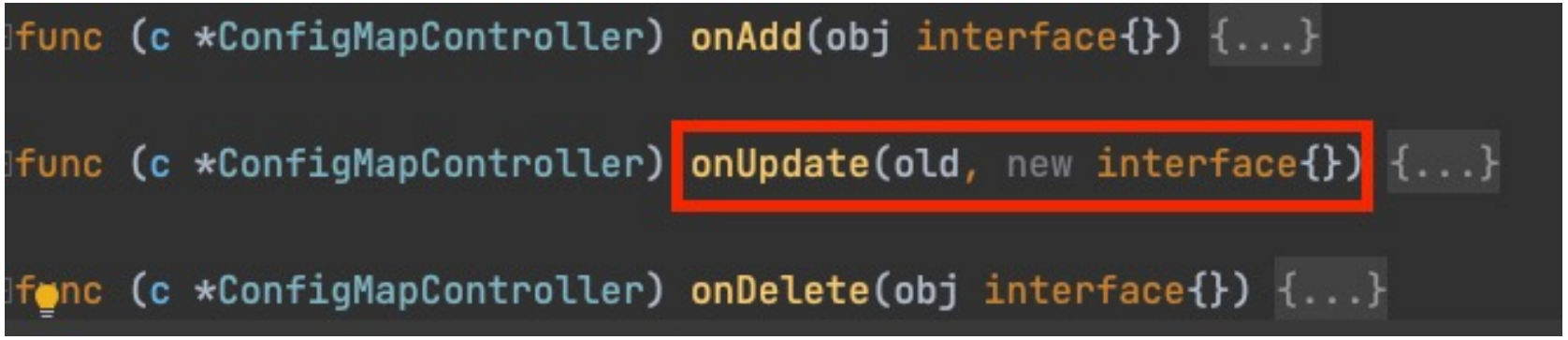
註冊Event Handler

- 每個Informer有三個event handler要實作
  - AddFunc
  - UpdateFunc
  - DeleteFunc

```go
func NewConfigMapController(client *kubernetes.Clientset) *ConfigMapController {
    factory := informers.NewSharedInformerFactoryWithOptions(
        client, 5*time.Second, informers.WithNamespace(namespace))
    informer := factory.Batch().V1().Jobs()

    c := &ConfigMapController{
        informerFactory: factory,
        informer:        informer,
        clientSet:       client,
    }
    informer.Informer().AddEventHandler(
        // Your custom resource event handlers.
        cache.ResourceEventHandlerFuncs{
            // Called on creation
            AddFunc: c.onAdd,
            // Called on resource update and every resyncPeriod on existing resources.
            UpdateFunc: c.onUpdate,
            // Called on resource deletion.
            DeleteFunc: c.onDelete,
        },
    )
    return c
}
```

- AddFunc:
  - Resource建立時
  - Informer啟動時，做為初始化之用
- UpdateFunc: 週期性呼叫，判斷新、舊狀態有無差異



Exercise

- 利用informer機制寫一個informer，watch deployment的event
  - onAdd():
    - nginx deployment後，自動建立service
    - 和前二次的Exercise一樣
  - onUpdate():
    - 一直取得deployment的資訊
    - 和前一次的Exercise一樣
  - onDelete():
    - 刪除nginx deployment後，會自動刪除service
    - 和前一次的Exercise一樣
- 寫Dockerfile建立Image, 部署至K8S上執行
  - 和前一次的Exercise一樣