Introduction to clinet-go API

莊家雋

Outline

- Create Resource
- BuildConfig
- Resource client
- Run inside cluster
  - Build Image
  - 設定 RBAC

client-go vs. kubectl流程

- 建立resource object
- 建立config
- 建立client
  - 建立clientset
  - 取得resource所要用的client
- 利用client執行CRUD等操作

- 建立yaml
- 讀kubeconfig
- kubectl
- 利用kubectl執行apply、delete等操作

Create Resource

- 所有內建Resource的structure都定義在k8s.io/api

```
oc example configmap
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap
data:
  data-1: value-1
  data-2: value-2
```

```
import (
    "context"
    "flag"
    "fmt"

    corev1 "k8s.io/api/core/v1"
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
    "k8s.io/client-go/kubernetes"
```

```
cm := &corev1.ConfigMap{
    ObjectMeta: metav1.ObjectMeta{
        Name:      "",
        Namespace: "default",
    },
    Data: map[string]string{"foo": "bar"},
}
```

https://iximiuz.com/en/posts/kubernetes-api-go-types-and-common-machinery/

ConfigMap v1 core

Group	Version	Kind
core	v1	ConfigMap

Appears In:

- ConfigMapList [core/v1]

Field	Description
apiVersion *string*	APIVersion defines the versioned schema of this repres should convert recognized schemas to the latest interr unrecognized values. More info: https://git.k8s.io/comr architecture/api-conventions.md#resources
binaryData *object*	BinaryData contains the binary data. Each key must co '-', '_' or '.'. BinaryData can contain byte sequences tha keys stored in BinaryData must not overlap with the on enforced during validation process. Using this field will kubelet.
data *object*	Data contains the configuration data. Each key must c '-', '_' or '.'. Values with non-UTF-8 byte sequences mu keys stored in Data must not overlap with the keys in th enforced during validation process.
immutable *boolean*	Immutable, if set to true, ensures that data stored in th (only object metadata can be modified). If not set to tr time. Defaulted to nil.
kind *string*	Kind is a string value representing the REST resource t infer this from the endpoint the client submits requests CamelCase. More info: https://git.k8s.io/community/co architecture/api-conventions.md#types-kinds

templateinstance_controller_test.go

```
import ...

var _ = Describe( text: "TemplateInstance Controller", func() {...})

func TestTemplateInstanceCondition(t *testing.T) {...}

func newConfigMap(name string) *v1.ConfigMap {

    cm := &v1.ConfigMap{
        TypeMeta: metav1.TypeMeta{
            Kind:       "ConfigMap",
            APIVersion: "v1",
        },
        ObjectMeta: metav1.ObjectMeta{
            Name: name,
        },

        Data: map[string]string{"k1": "v1"},
    }

    return cm
}

func newServiceAccount() *v1.ServiceAccount {...}

func newRole(resource string) *rbacv1.Role {...}

func newRoleBinding() *rbacv1.RoleBinding {...}
```

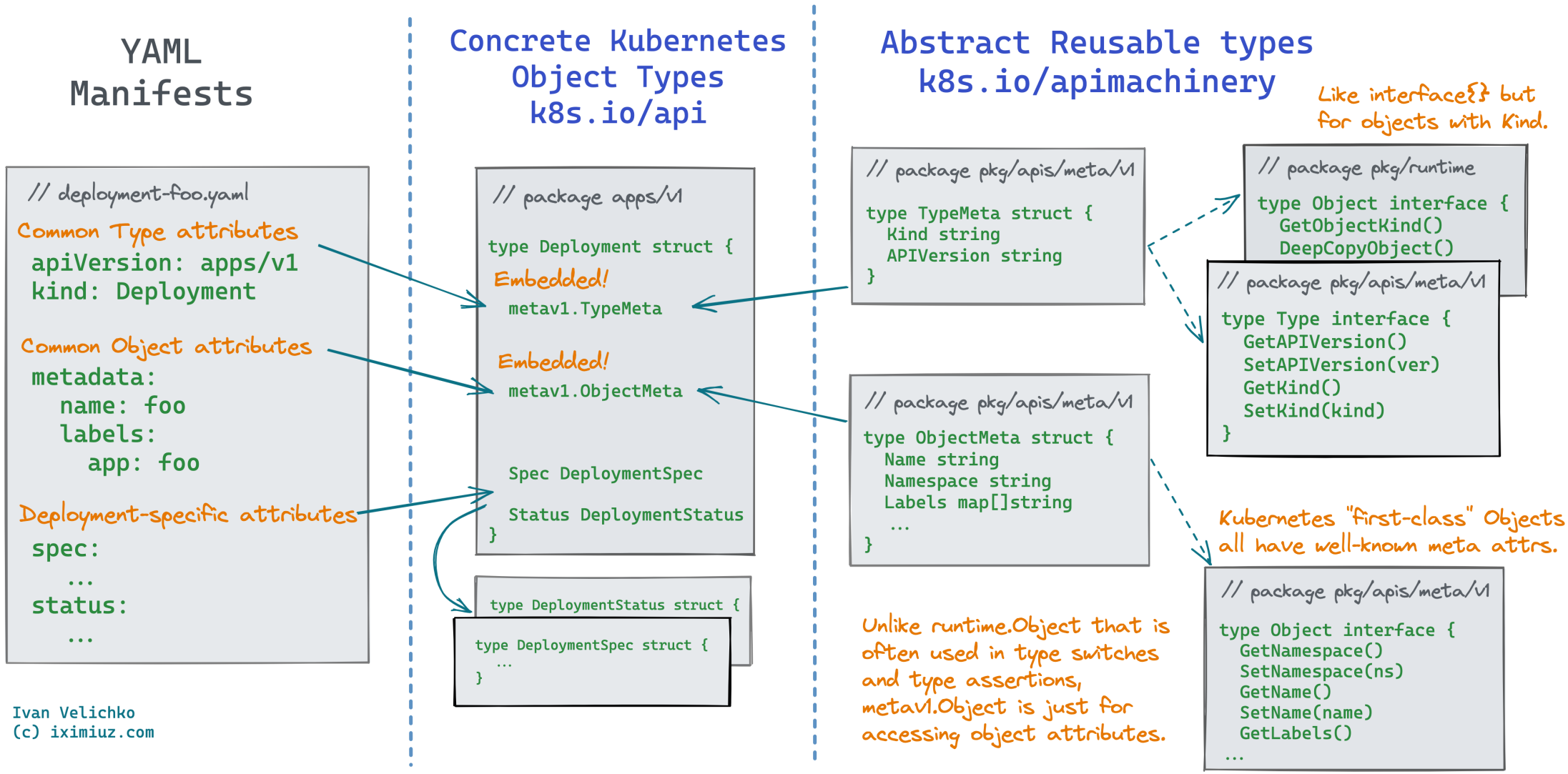
types.go

```
type ConfigMap struct {
    metav1.TypeMeta `json:",inline"`
    // Standard object's metadata.
    // More info: https://git.k8s.io/community/contributors/devel/sig-architec
    // +optional
    metav1.ObjectMeta `json:"metadata,omitempty" protobuf:"bytes,1,opt,name=me

    // Immutable, if set to true, ensures that data stored in the ConfigMap ca
    // be updated (only object metadata can be modified).
    // If not set to true, the field can be modified at any time.
    // Defaulted to nil.
    // +optional
    Immutable *bool `json:"immutable,omitempty" protobuf:"varint,4,opt,name=im

    // Data contains the configuration data.
    // Each key must consist of alphanumeric characters, '-', '_' or '.'.
    // Values with non-UTF-8 byte sequences must use the BinaryData field.
    // The keys stored in Data must not overlap with the keys in
    // the BinaryData field, this is enforced during validation process.
    // +optional
    Data map[string]string `json:"data,omitempty" protobuf:"bytes,2,rep,name=da

    // BinaryData contains the binary data.
    // Each key must consist of alphanumeric characters, '-', '_' or '.'.
    // BinaryData can contain byte sequences that are not in the UTF-8 range.
    // The keys stored in BinaryData must not overlap with the ones in
    // the Data field, this is enforced during validation process.
    // Using this field will require 1.10+ apiserver and
    // kubelet.
    // +optional
    BinaryData map[string][]byte `json:"binaryData,omitempty" protobuf:"bytes,
}
```


YAML Manifests
Concrete Kubernetes Object Types k8s.io/api
Abstract Reusable types k8s.io/apimachinery
// deployment-foo.yaml
Common Type attributes
apiVersion: apps/v1
kind: Deployment
Common Object attributes
metadata:
name: foo
labels:
app: foo
Deployment-specific attributes
spec:
...
status:
...
// package apps/v1
type Deployment struct {
Embedded!
metav1.TypeMeta
Embedded!
metav1.ObjectMeta
Spec DeploymentSpec
Status DeploymentStatus
}
type DeploymentStatus struct {
type DeploymentSpec struct {
...
}
// package pkg/apis/meta/v1
type TypeMeta struct {
Kind string
APIVersion string
}
Like interface{} but for objects with Kind.
// package pkg/runtime
type Object interface {
GetObjectKind()
DeepCopyObject()
// package pkg/apis/meta/v1
type Type interface {
GetAPIVersion()
SetAPIVersion(ver)
GetKind()
SetKind(kind)
}
// package pkg/apis/meta/v1
type ObjectMeta struct {
Name string
Namespace string
Labels map[]string
...
}
Kubernetes "first-class" Objects all have well-known meta attrs.
// package pkg/apis/meta/v1
type Object interface {
GetNamespace()
SetNamespace(ns)
GetName()
SetName(name)
GetLabels()
...
Unlike runtime.Object that is often used in type switches and type assertions, metav1.Object is just for accessing object attributes.
Ivan Velichko
(c) iximiuz.com

Build config

- 在Cluster內部
  - App 以pod形式執行在K8S裡

```
} else {
    // creates the in-cluster config
    config, err := rest.InClusterConfig()
    if err != nil : err.Error() *
    // creates the clientset
    clientset, err = kubernetes.NewForConfig(config)
    if err != nil : err.Error() *
}
```

- 在Cluster外部
  - App 並不是執行在K8S裡

```
if *outsideCluster {
    // creates the out-cluster config
    home, err := os.UserHomeDir()
    if err != nil : err *
    config, err := clientcmd.
        BuildConfigFromFlags( masterUrl: "", path.Join(home, ".kube/config"))
    if err != nil : err.Error() *
    // creates the clientset
    clientset, err = kubernetes.NewForConfig(config)
    if err != nil : err.Error() *
```

建立resource的client

- Clientset是所有resource的client集合
- 每個resource都有其相對應的client
- 利用工廠模式，產生所需要的client

Resource Verb

- 每個Resource (eg. Namespace、Pod、Deployment、等)皆有
  - Get
  - Create
  - List
  - Delete
  - Update
  - Apply
  - Patch
  - Watch
  - DeleteCollection

RBAC

- 當client在Cluster外執行時，
  - 是依當時kubeconfig context所使用的身份決定權限
  - 預設為cluster-admin

- 當Client在Cluster內執行時， 需要
  - 設定Client是用什麼身份執行: Service Account
  - 此身份可以執行那些操作: Role & RoleBinding
  - 預設用default SA, 沒有認何的權限

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cm-incluster
  namespace: default
```

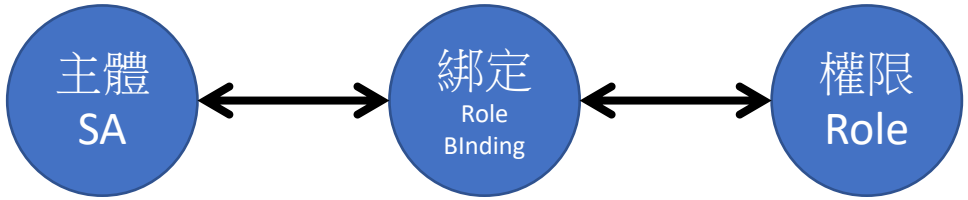
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: in-cluster-example
spec:
  replicas: 1
  selector:
    matchLabels:
      app: in-cluster-example
  template:
    metadata:
      labels:
        app: in-cluster-example
    spec:
      serviceAccountName: cm-incluster
      containers:
        - name: in-cluster-example
          image: ogre0403/incluster:latest
          imagePullPolicy: IfNotPresent
```

RBAC

- 當Client在Cluster內執行時，需要
  - 設定Client是用什麼身份執行: Service Account
  - 此身份可以執行那些操作: Role & RoleBinding

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: cm-role
  namespace: default
rules:
  - apiGroups:
      - ""
    resources:
      - configmaps
    verbs:
      - get
      - create
      - delete
```

- Role: 有權限做什麼事情
- RoleBinding: 將身份與權限做綁定




```yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata: ObjectMeta
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: cm-role
subjects:
  - kind: ServiceAccount
    name: cm-incluster
    namespace: default
```

Exercise

- 利用client-go library寫一個client program
  - 建立nginx deployment與service
    - 與第一次Exercise相同，但是改用golang完成
  - 建立後一直取得deployment訊息
  - 中止client program時，同時刪除nginx deployment與service
- 寫Dockerfile建立Image, 部署至K8S上執行