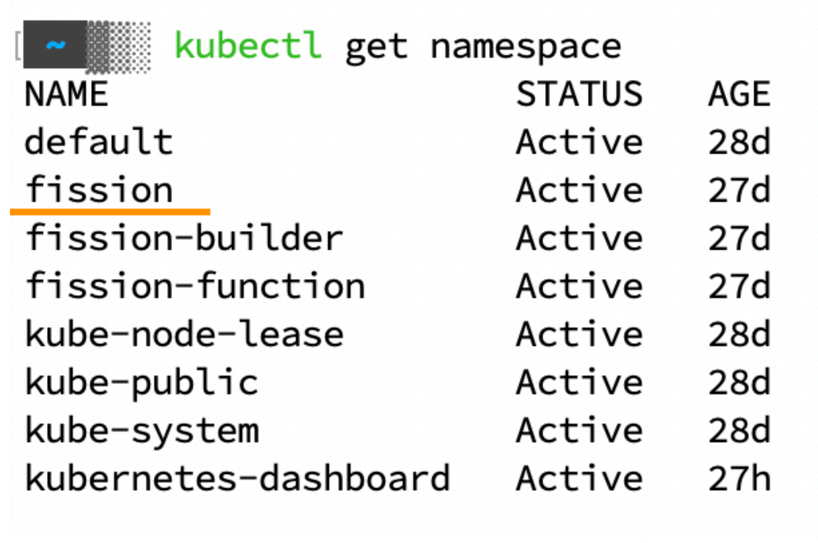
# Install fission

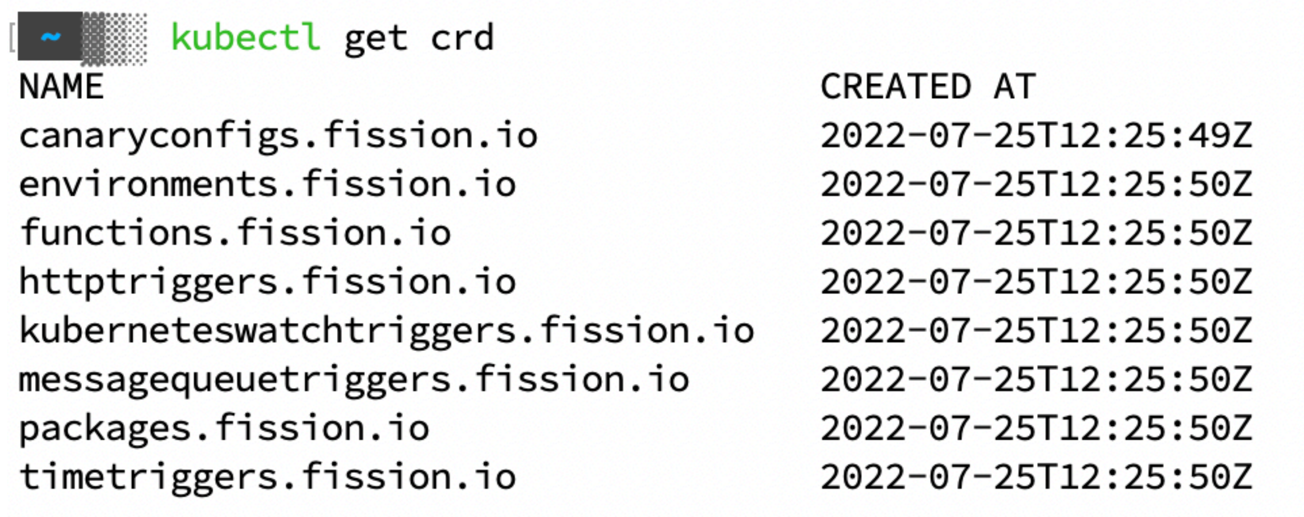
## prerequisite: install k8s, kubectl

- 安装k8s、kubectl
- 启动k8s cluster (minikube)，通过kubectl操控cluster
  - 注意：只有cluster运行起来，才能使用kubectl操控；
- kubectl create namespace fission
  - 

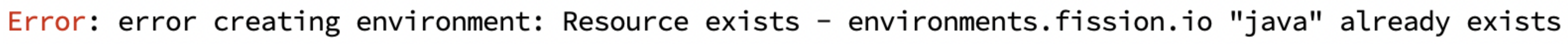
    ```
    ~  kubectl get namespace
    NAME                   STATUS   AGE
    default                Active   28d
    fission                Active   27d
    fission-builder        Active   27d
    fission-function       Active   27d
    kube-node-lease        Active   28d
    kube-public            Active   28d
    kube-system            Active   28d
    kubernetes-dashboard   Active   27h
    ```
  - fission相关**资源**、**服务**运行的指定空间；

# Install fission

## install fission without helm

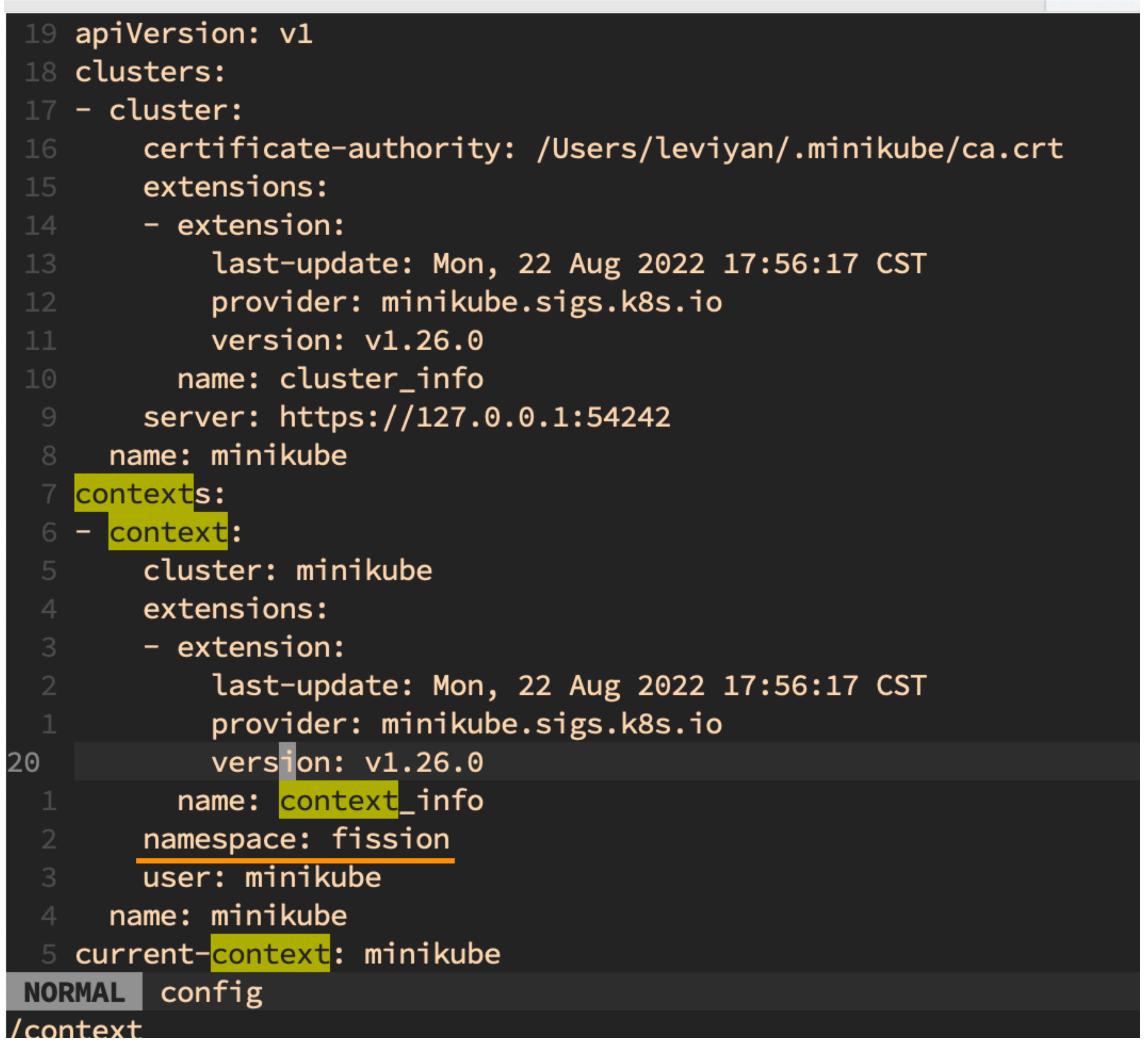
- kubectl create -k "github.com/fission/fission/crds/v1?ref=v1.16.0"
  - 创建fission crd
  - 

```
~ kubectl get crd
NAME                                 CREATED AT
canaryconfigs.fission.io             2022-07-25T12:25:49Z
environments.fission.io              2022-07-25T12:25:50Z
functions.fission.io                 2022-07-25T12:25:50Z
httptriggers.fission.io              2022-07-25T12:25:50Z
kuberneteswatchtriggers.fission.io   2022-07-25T12:25:50Z
messagequeuetriggers.fission.io      2022-07-25T12:25:50Z
packages.fission.io                  2022-07-25T12:25:50Z
timetriggers.fission.io              2022-07-25T12:25:50Z
```

  - fission创建的资源就存储在这些位置
    - fission environment create --name java --image fission/jvm-env --builder fission/jvm-builder --keeparchive --versi
    - 

```
Error: error creating environment: Resource exists - environments.fission.io "java" already exists
```

- export FISSION_NAMESPACE="fission"
- kubectl create namespace $FISSION_NAMESPACE
- kubectl config set-context --current --namespace=$FISSION_NAMESPACE
  - 把kubeconfig file中的namespace修改为fission，作用是**把cluster当前的namespace设置为fission**

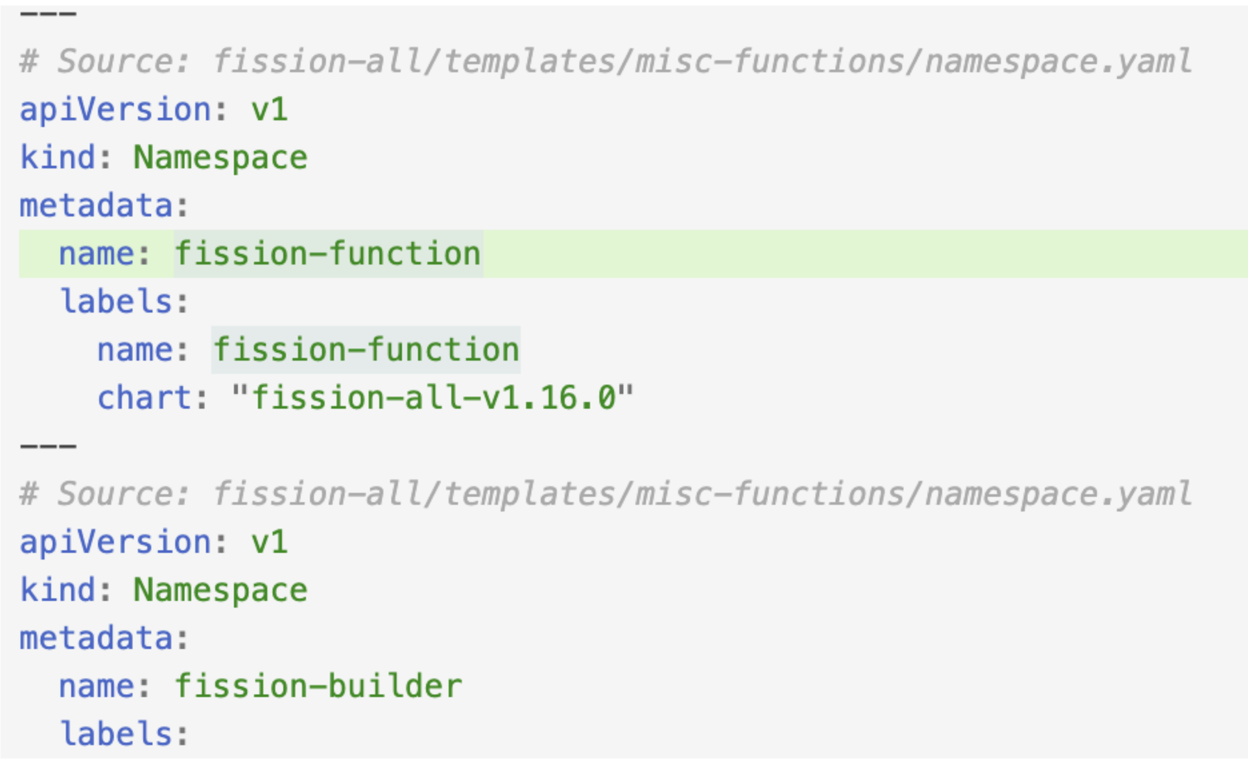


```
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /Users/leviyan/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Mon, 22 Aug 2022 17:56:17 CST
        provider: minikube.sigs.k8s.io
        version: v1.26.0
      name: cluster_info
    server: https://127.0.0.1:54242
  name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Mon, 22 Aug 2022 17:56:17 CST
        provider: minikube.sigs.k8s.io
        version: v1.26.0
      name: context_info
    namespace: fission
    user: minikube
  name: minikube
current-context: minikube
NORMAL  config
/context
```

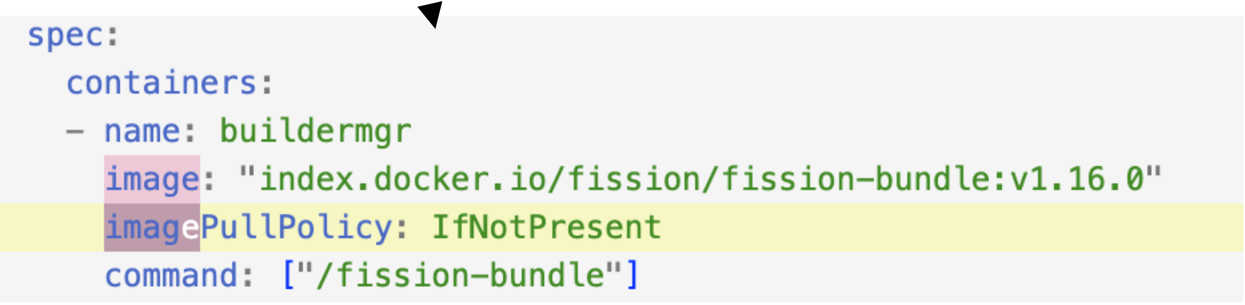
# Install fission

## install fission without helm

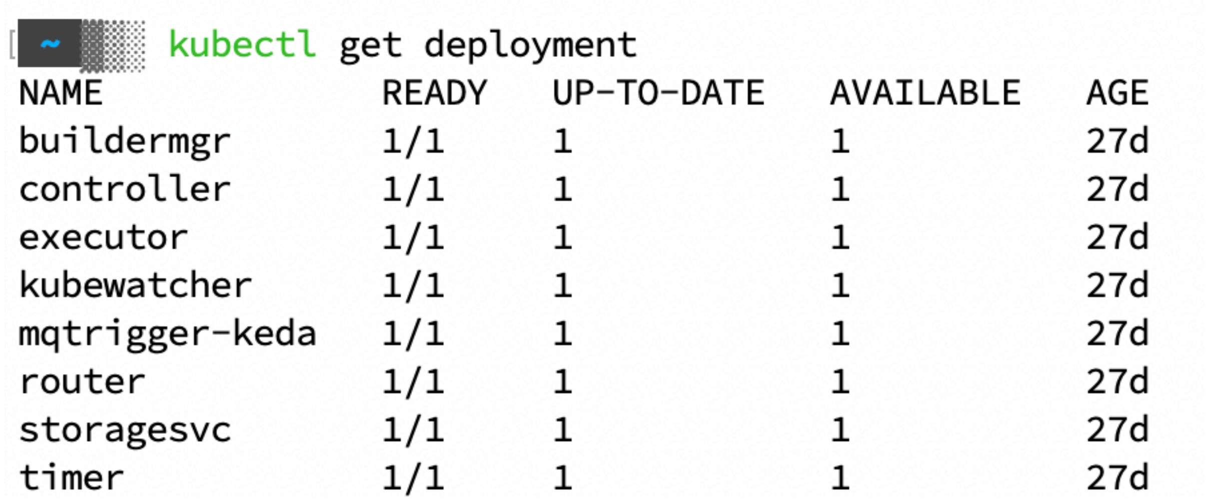
- kubectl apply -f https://github.com/fission/fission/releases/download/v1.16.0/fission-all-v1.16.0-minikube.yaml
  - Apply a configuration to a resource by file name or stdin
  - 把**fission的组件都部署到minikube集群**上，**提供fission服务，接收、执行fission cli发来的命令**；
    - yaml文件指明了组件存储的位置，如 image: "index.docker.io/fission/fission-bundle:v1.16.0"



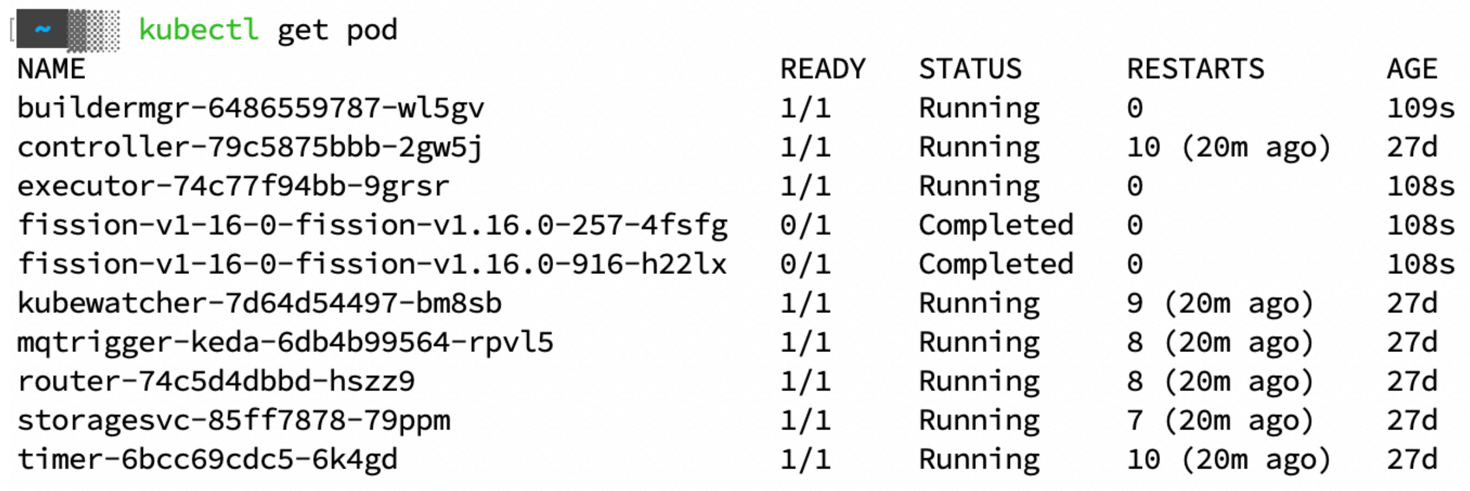
```
---
# Source: fission-all/templates/misc-functions/namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: fission-function
  labels:
    name: fission-function
    chart: "fission-all-v1.16.0"
---
# Source: fission-all/templates/misc-functions/namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: fission-builder
  labels:
```



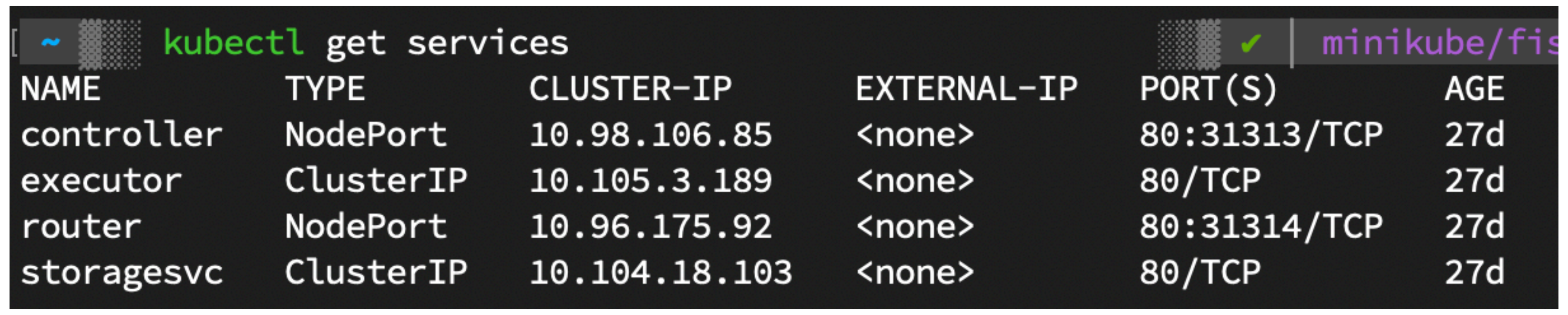
```
spec:
  containers:
  - name: buildermgr
    image: "index.docker.io/fission/fission-bundle:v1.16.0"
    imagePullPolicy: IfNotPresent
    command: ["/fission-bundle"]
```



```
~ kubectl get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
buildermgr       1/1     1            1           27d
controller       1/1     1            1           27d
executor         1/1     1            1           27d
kubewatcher      1/1     1            1           27d
mqtrigger-keda   1/1     1            1           27d
router           1/1     1            1           27d
storagesvc       1/1     1            1           27d
timer            1/1     1            1           27d
```



```
~ kubectl get pod
NAME                                       READY   STATUS      RESTARTS       AGE
buildermgr-6486559787-wl5gv                1/1     Running     0              109s
controller-79c5875bbb-2gw5j                1/1     Running     10 (20m ago)   27d
executor-74c77f94bb-9grsr                  1/1     Running     0              108s
fission-v1-16-0-fission-v1.16.0-257-4fsfg  0/1     Completed   0              108s
fission-v1-16-0-fission-v1.16.0-916-h22lx  0/1     Completed   0              108s
kubewatcher-7d64d54497-bm8sb               1/1     Running     9 (20m ago)    27d
mqtrigger-keda-6db4b99564-rpvl5            1/1     Running     8 (20m ago)    27d
router-74c5d4dbbd-hszz9                    1/1     Running     8 (20m ago)    27d
storagesvc-85ff7878-79ppm                  1/1     Running     7 (20m ago)    27d
timer-6bcc69cdc5-6k4gd                     1/1     Running     10 (20m ago)   27d
```

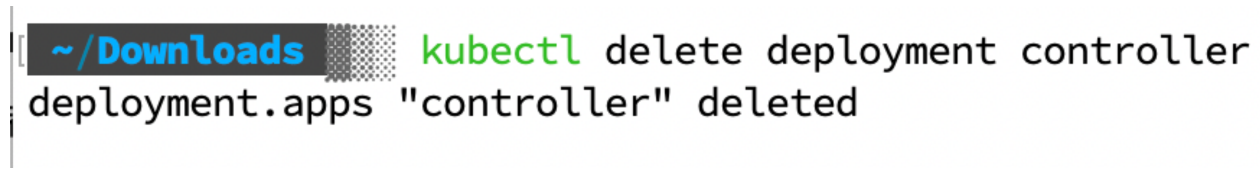


```
~ kubectl get services                                          ✓ | minikube/fis
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
controller   NodePort    10.98.106.85    <none>        80:31313/TCP   27d
executor     ClusterIP   10.105.3.189    <none>        80/TCP         27d
router       NodePort    10.96.175.92    <none>        80:31314/TCP   27d
storagesvc   ClusterIP   10.104.18.103   <none>        80/TCP         27d
```

# Install fission

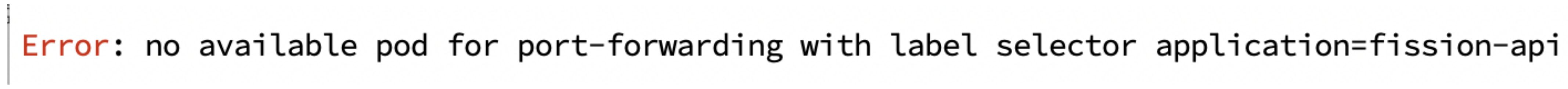
## install fission without helm

- 把fission的组件都部署到minikube集群上，提供fission服务，接收、执行fission cli发来的命令；

  - 

    ```
    ~/Downloads  kubectl delete deployment controller
    deployment.apps "controller" deleted
    ```

  - fission environment create --name java --image fission/jvm-env --builder fission/jvm-builder --keeparchive --version 3

  - 

    ```
    Error: no available pod for port-forwarding with label selector application=fission-api
    ```

# Install fission

## install fission without helm

- fission安装好之后的效果
  - fission的各个组件部署到k8s集群的pod中，fission本身就是在k8s 集群中。
  - 这个时候fission与k8s通信的话可以**直接从pod与api server交流**。而用户**通过fission cli与fission交流**（通过clientset）。

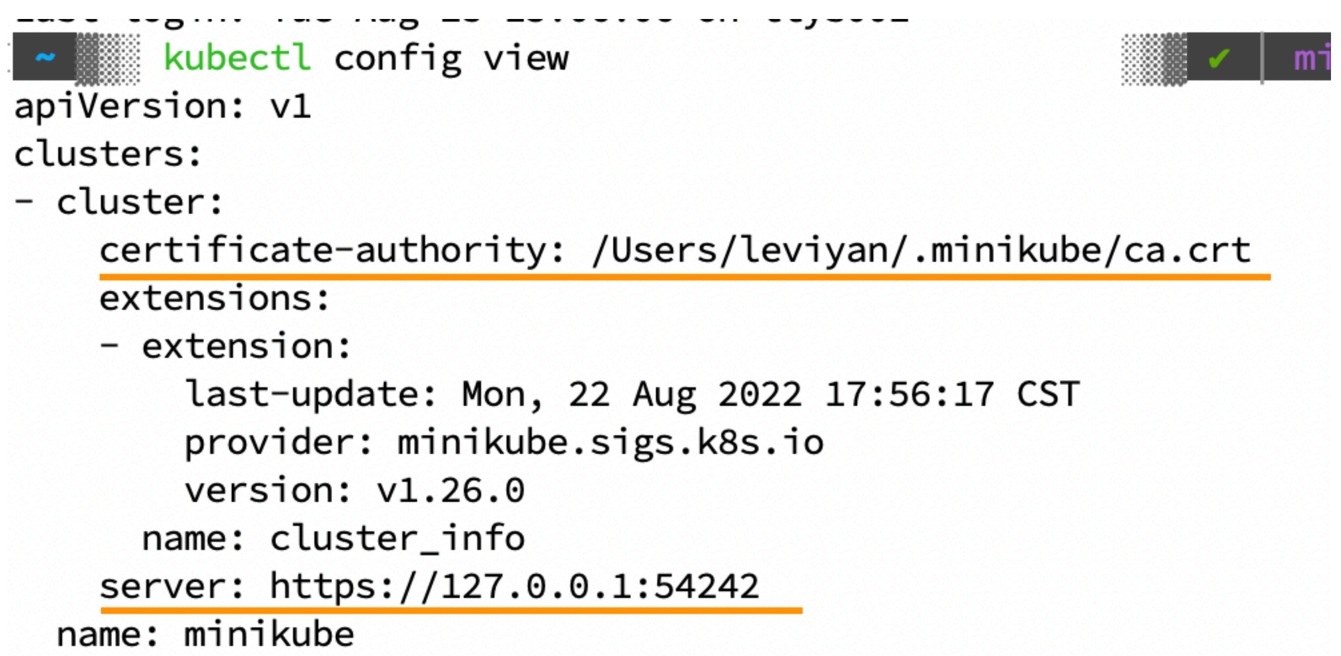
# Install Fission CLI

- curl -Lo fission https://github.com/fission/fission/releases/download/v1.16.0/fission-v1.16.0-darwin-amd64 \    && chmod +x fission && sudo mv fission /usr/local/bin/
    - 然后可以通过CLI与fission服务通信；

# Access Clusters Using the Kubernetes API

## kubeconfig file

- kubectl (kube command-line)
  - To access a cluster, you need to know **the location of the cluster and have credentials to access it.**
    - kubeconfg file:

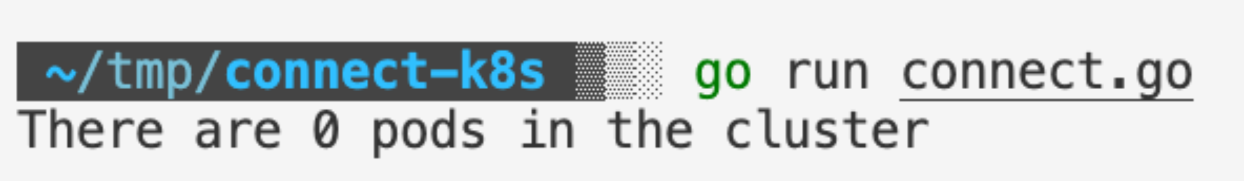


```
 ~   kubectl config view                                    ✔  mi
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /Users/leviyan/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Mon, 22 Aug 2022 17:56:17 CST
        provider: minikube.sigs.k8s.io
        version: v1.26.0
      name: cluster_info
    server: https://127.0.0.1:54242
  name: minikube
```

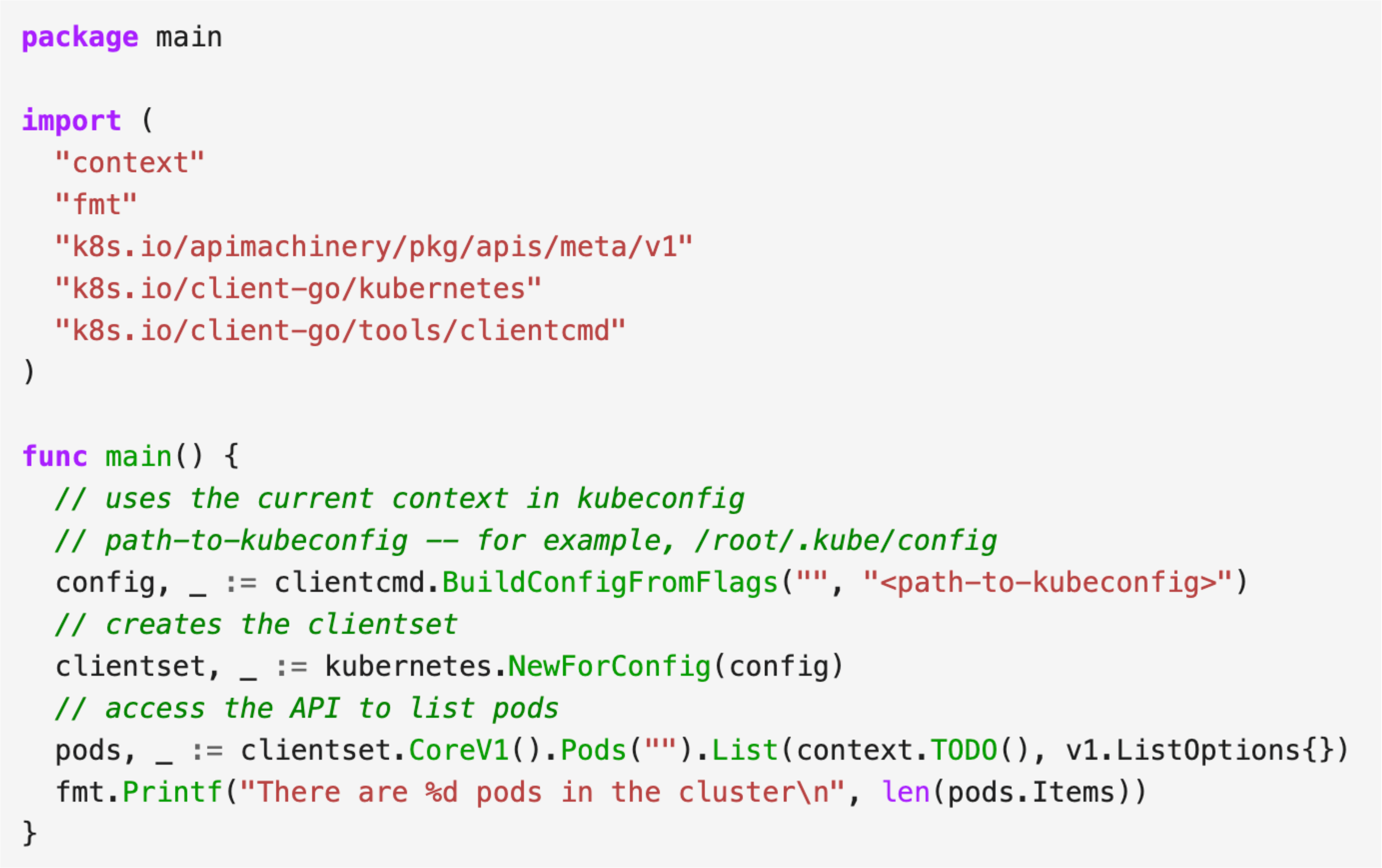
- Programmatic access to the API (go client)
  - cluster-outside: The **Go client** can use **the same kubeconfig file as the kubectl CLI does** to locate and authenticate to the API server. (fission cli)
  - cluster-inside: pod内的应用使用go client，go client会找到api server并验证；(fission component)

# Access Clusters Using the Kubernetes API

## kubeconfig file

- go client
  - example
    - 启动minikube，下载依赖，运行代码
    - 

```
~/tmp/connect-k8s  go run connect.go
There are 0 pods in the cluster
```

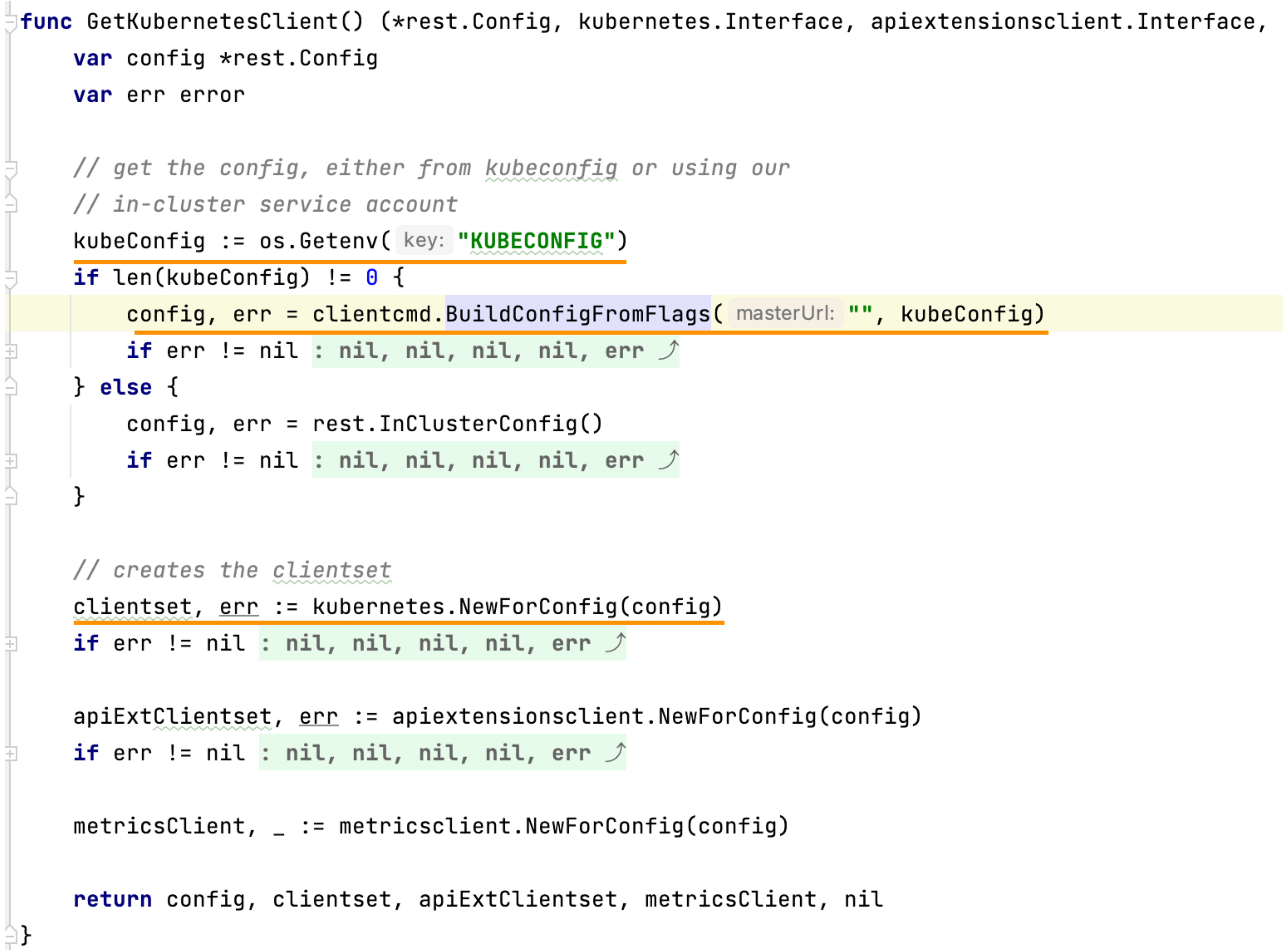


```
package main

import (
  "context"
  "fmt"
  "k8s.io/apimachinery/pkg/apis/meta/v1"
  "k8s.io/client-go/kubernetes"
  "k8s.io/client-go/tools/clientcmd"
)

func main() {
  // uses the current context in kubeconfig
  // path-to-kubeconfig -- for example, /root/.kube/config
  config, _ := clientcmd.BuildConfigFromFlags("", "<path-to-kubeconfig>")
  // creates the clientset
  clientset, _ := kubernetes.NewForConfig(config)
  // access the API to list pods
  pods, _ := clientset.CoreV1().Pods("").List(context.TODO(), v1.ListOptions{})
  fmt.Printf("There are %d pods in the cluster\n", len(pods.Items))
}
```

# Access Clusters Using the Kubernetes API

## kubeconfig file

- fission



```
func GetKubernetesClient() (*rest.Config, kubernetes.Interface, apiextensionsclient.Interface,
    var config *rest.Config
    var err error

    // get the config, either from kubeconfig or using our
    // in-cluster service account
    kubeConfig := os.Getenv( key: "KUBECONFIG")
    if len(kubeConfig) != 0 {
        config, err = clientcmd.BuildConfigFromFlags( masterUrl: "", kubeConfig)
        if err != nil : nil, nil, nil, nil, err ⤴
    } else {
        config, err = rest.InClusterConfig()
        if err != nil : nil, nil, nil, nil, err ⤴
    }

    // creates the clientset
    clientset, err := kubernetes.NewForConfig(config)
    if err != nil : nil, nil, nil, nil, err ⤴

    apiExtClientset, err := apiextensionsclient.NewForConfig(config)
    if err != nil : nil, nil, nil, nil, err ⤴

    metricsClient, _ := metricsclient.NewForConfig(config)

    return config, clientset, apiExtClientset, metricsClient, nil
}
```

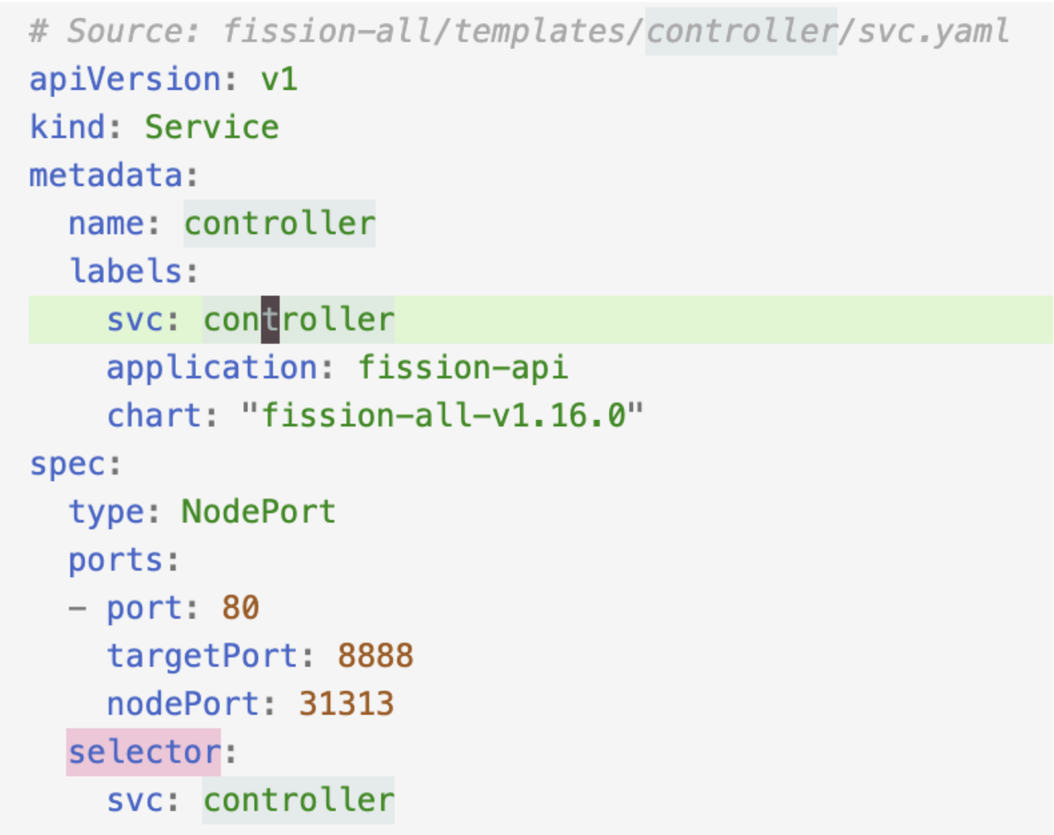
# fission运行流程

## fission结构

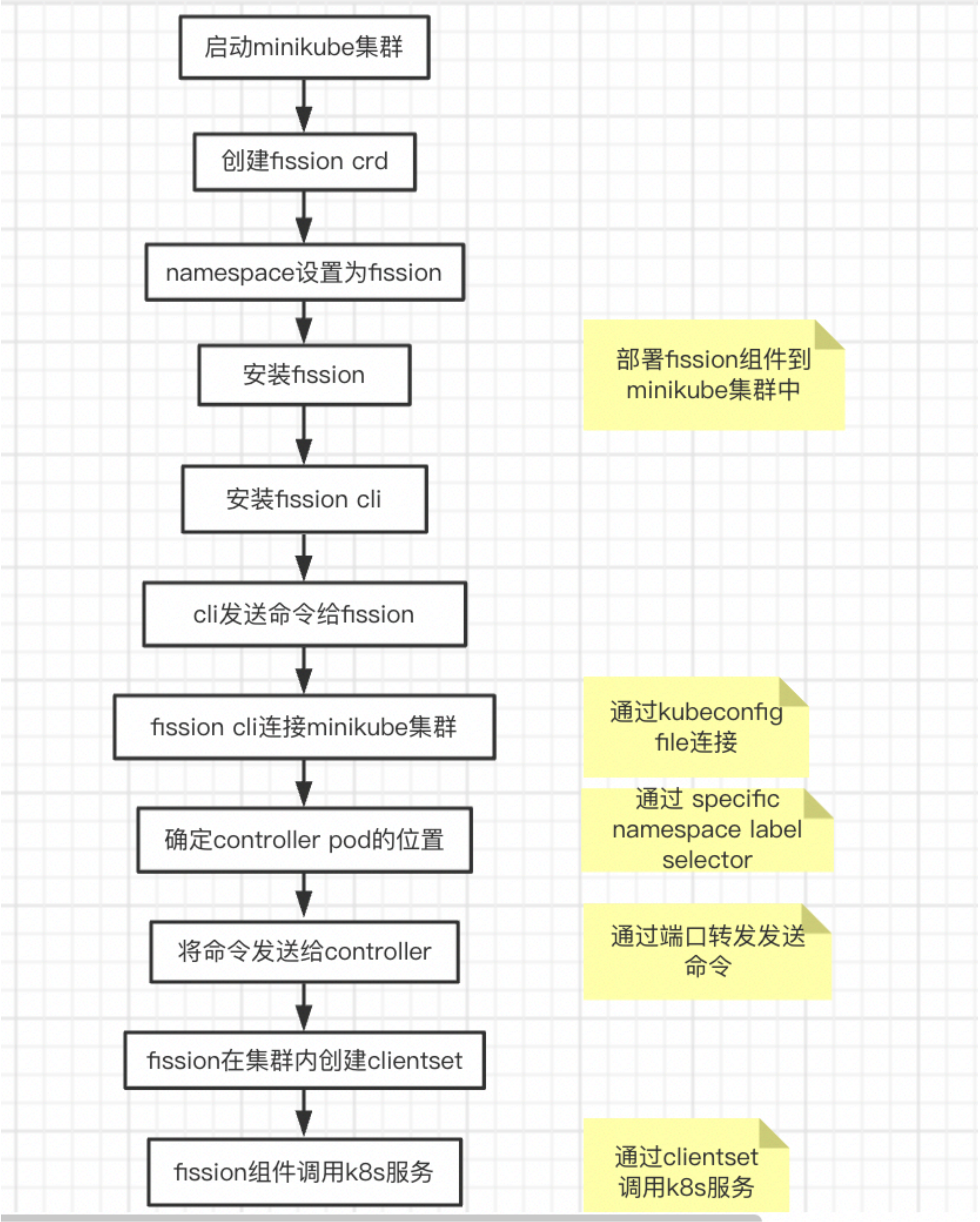
- fission项目可以分成两部分
    - 第一部分是**fission组件**，如controller、trigger、executor；
        - 这一部分部署在k8s的集群中，提供服务；
    - 第二部分是**fission cli**
        - 用户用cli发送命令，cli把命令发送给fission controller执行；

# fission运行流程

## 流程

- 备注
  - 在指定namespace内，通过label selector确定controller pod位置
  - 

```
# Source: fission-all/templates/controller/svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: controller
  labels:
    svc: controller
    application: fission-api
    chart: "fission-all-v1.16.0"
spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 8888
    nodePort: 31313
  selector:
    svc: controller
```



启动minikube集群

创建fission crd

namespace设置为fission

安装fission

部署fission组件到minikube集群中

安装fission cli

cli发送命令给fission

fission cli连接minikube集群

通过kubeconfig file连接

确定controller pod的位置

通过 specific namespace label selector

将命令发送给controller

通过端口转发发送命令

fission在集群内创建clientset

fission组件调用k8s服务

通过clientset调用k8s服务

# 参考

- Installing Fission
- Access Clusters Using the Kubernetes API
- Kubeconfig File Explained With Practical Examples
- Client Libraries