kubernetes-dashboard(1.8.3)部署与踩坑

Kubernetes Dashboard 是一个管理Kubernetes集群的全功能Web界面，旨在以UI的方式完全替代命令行工具（kubectl 等）。

目录

部署

Dashboard需要用到 `k8s.gcr.io/kubernetes-dashboard` 的镜像，由于网络原因，可以采用预先拉取并打Tag或者修改yaml文件中的镜像地址，本文使用后者：

```
kubectl apply -f http://mirror.faasx.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml
```

> 上面使用的yaml只是将 https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml 中的 *k8s.gcr.io* 替换为了 *reg.qiniu.com/k8s*。

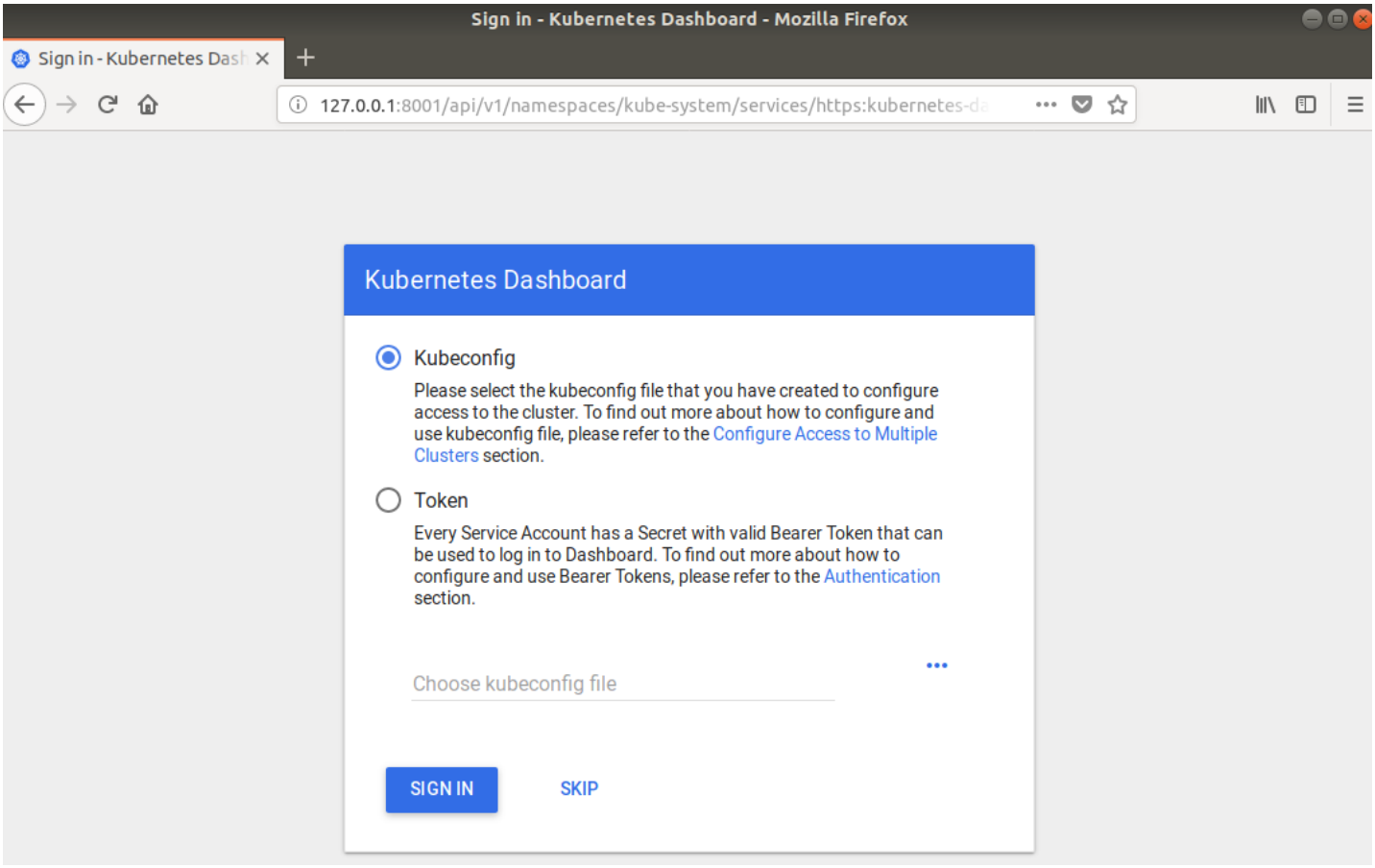
然后可以使用 `kubectl get pods` 命令来查看部署状态：

```
kubectl get pods --all-namespaces

# 输出
NAMESPACE     NAME                                    READY     STATUS    RESTARTS   AGE
kube-system   kubernetes-dashboard-7d5dcdb6d9-mf6l2   1/1       Running   0          9m
```

如果要在本地访问dashboard，我们需要创建一个安全通道，可运行如下命令：

```
kubectl proxy
```

现在就可以通过 http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/ 来访问Dashborad UI了。



创建用户

如上图，跳转到了登录页面，那我们就先创建个用户：

1.创建服务账号

首先创建一个叫 `admin-user` 的服务账号，并放在 `kube-system` 名称空间下：

```
# admin-user.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kube-system
```

执行 `kubectl create` 命令：

```
kubectl create -f admin-user.yaml
```

2.绑定角色

默认情况下， `kubeadm` 创建集群时已经创建了 `admin` 角色，我们直接绑定即可：

```
# admin-user-role-binding.yaml
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kube-system
```

执行 `kubectl create` 命令：

```
kubectl create -f  admin-user-role-binding.yaml
```

3.获取Token

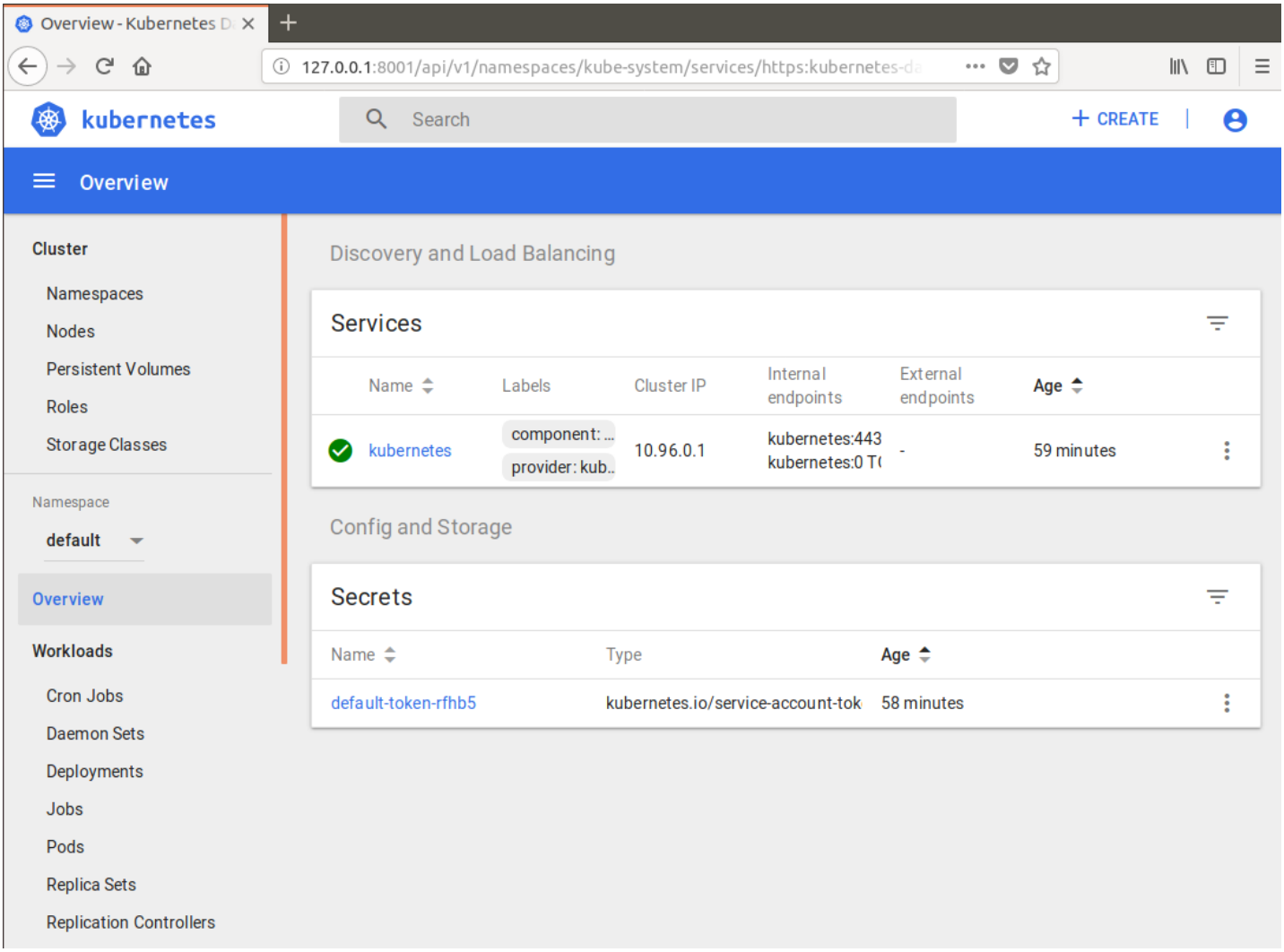
现在我们需要找到新创建的用户的Token，以便用来登录dashboard：

```
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')
```

输出类似：

```
Name:         admin-user-token-qrj82
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name=admin-user
              kubernetes.io/service-account.uid=6cd60673-4d13-11e8-a548-00155d000529

Type:  kubernetes.io/service-account-token

Data
====
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLXFyajgyIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiI2Y2Q2MDY3My00ZDEzLTExZTgtYTU0OC0wMDE1NWQwMDA1MjkiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.C5mjsa2uqJwjscWQ9x4mEsWALUTJu3OSfLYecqpS1niYXxp328mgx0t-QY8A7GQvAr5fWoIhhC_NOHkSkn2ubn0U22VGh2msU6zAbz9sZZ7BMXG4DLMq3AaXTXY8LzS3PQyEOCaLieyEDe-tuTZz4pbqoZQJ6V6zaKJtE9u6-zMBC2_iFujBwhBViaAP9KBbE5WfREEc0SQR9siN8W8gLSc8ZL4snndv527Pe9SxojpDGw6qP_8R-i51bP2nZGlpPadEPXj-lQqz4g5pgGziQqnsInSMpctJmHbfAh7s9lIMoBFW7GVE8AQNSoLHuuevbLArJ7sHriQtDB76_j4fmA
ca.crt:     1025 bytes
namespace:  11 bytes
```

然后把Token复制到登录界面的Token输入框中，登入后显示如下：



集成Heapster

Heapster是容器集群监控和性能分析工具，天然的支持Kubernetes和CoreOS。

Heapster支持多种储存方式，本示例中使用 `influxdb` ，直接执行下列命令即可：

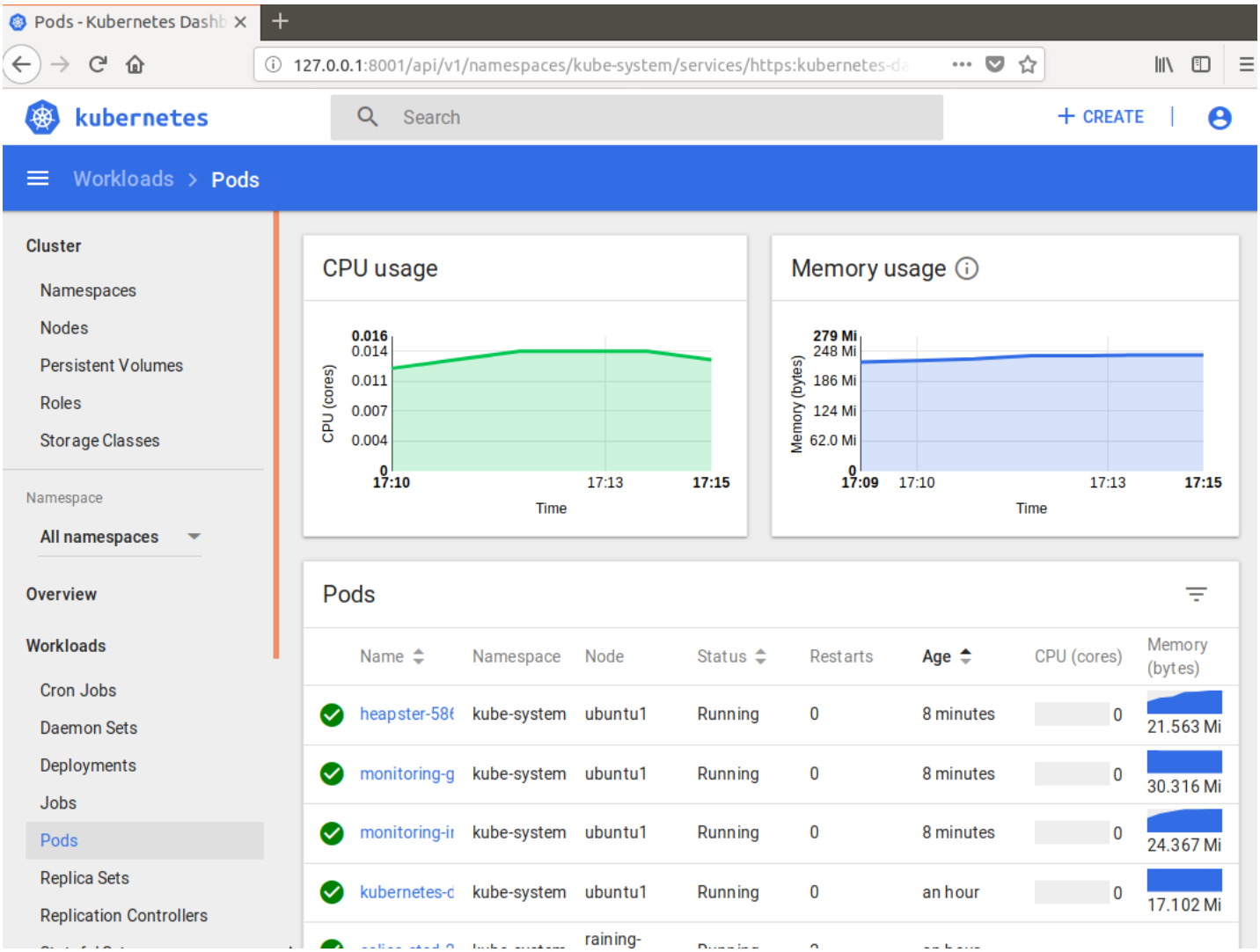
```
kubectl create -f http://mirror.faasx.com/kubernetes/heapster/deploy/kube-config/influxdb/influxdb.yaml
kubectl create -f http://mirror.faasx.com/kubernetes/heapster/deploy/kube-config/influxdb/grafana.yaml
kubectl create -f http://mirror.faasx.com/kubernetes/heapster/deploy/kube-config/influxdb/heapster.yaml
kubectl create -f http://mirror.faasx.com/kubernetes/heapster/deploy/kube-config/rbac/heapster-rbac.yaml
```

> 上面命令中用到的yaml是从 https://github.com/kubernetes/heapster/tree/master/deploy/kube-config/influxdb复制的，并将 `k8s.gcr.io` 修改为国内镜像。

然后，查看一下Pod的状态：

```
raining@raining-ubuntu:~/k8s/heapster$ kubectl get pods --namespace=kube-system
NAME                                    READY     STATUS    RESTARTS   AGE
...
heapster-5869b599bd-kxltn               1/1       Running   0          5m
monitoring-grafana-679f6b46cb-xxsr4     1/1       Running   0          5m
monitoring-influxdb-6f875dc468-7s4xz    1/1       Running   0          6m
...
```

等待状态变成 `Running` ，刷新一下浏览器，最新的效果如下：



关于Heapster更详细的用法可参考官方文档：https://github.com/kubernetes/heapster。

访问

Kubernetes提供了以下四种访问服务的方式：

kubectl proxy

在上面的示例中，我们使用的便是 `kubectl proxy` ，它在您的机器与Kubernetes API之间创建一个代理，默认情况下，只能从本地访问（启动它的机器）。

我们可以使用 `kubectl cluster-info` 命令来检查配置是否正确，集群是否可以访问等：

```
raining@raining-ubuntu:~$ kubectl cluster-info
Kubernetes master is running at https://192.168.0.8:6443
Heapster is running at https://192.168.0.8:6443/api/v1/namespaces/kube-system/services/heapster/proxy
KubeDNS is running at https://192.168.0.8:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
monitoring-grafana is running at https://192.168.0.8:6443/api/v1/namespaces/kube-system/services/monitoring-grafana/proxy
monitoring-influxdb is running at https://192.168.0.8:6443/api/v1/namespaces/kube-system/services/monitoring-influxdb/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

启动代理只需执行如下命令：

```
$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

我们也可以使用 `--address` 和 `--accept-hosts` 参数来允许外部访问：

```
kubectl proxy --address='0.0.0.0'  --accept-hosts='^*$'
```

然后我们在外网访问 `http://<master-ip>:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/` ，可以成功访问到登录界面，但是却无法登录，这是因为Dashboard只允许 `localhost` 和 `127.0.0.1` 使用HTTP连接进行访问，而其它地址只允许使用HTTPS。因此，如果需要在非本机访问Dashboard的话，只能选择其他访问方式。

NodePort

NodePort是将节点直接暴露在外网的一种方式，只建议在开发环境，单节点的安装方式中使用。

启用NodePort很简单，只需执行 `kubectl edit` 命令进行编辑：

```
kubectl -n kube-system edit service kubernetes-dashboard
```

输出如下：

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"labels":{"k8s-app":"kubernetes-
dashboard"},"name":"kubernetes-dashboard","namespace":"kube-system"},"spec":{"ports":[{"port":443,"targetPort":8443}],"selector":
{"k8s-app":"kubernetes-dashboard"}}}
  creationTimestamp: 2018-05-01T07:23:41Z
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
  resourceVersion: "1750"
  selfLink: /api/v1/namespaces/kube-system/services/kubernetes-dashboard
  uid: 9329577a-4d10-11e8-a548-00155d000529
spec:
  clusterIP: 10.103.5.139
  ports:
  - port: 443
    protocol: TCP
    targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
  sessionAffinity: None
  type: ClusterIP
```

```
status:
  loadBalancer: {}
```

然后我们将上面的 `type: ClusterIP` 修改为 `type: NodePort` ，保存后使用 `kubectl get service` 命令来查看自动生产的端口：

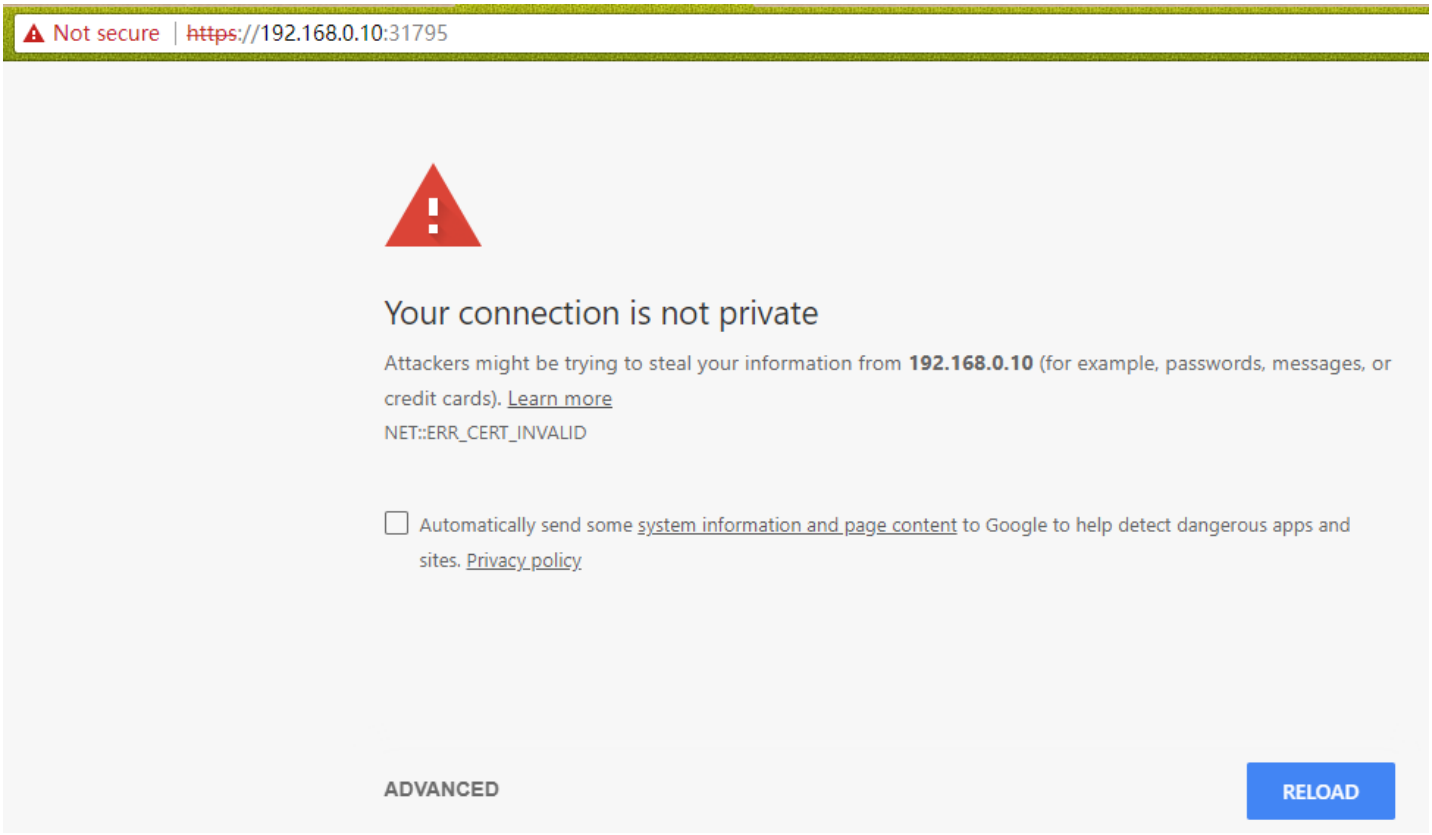
```
kubectl -n kube-system get service kubernetes-dashboard
```

输出如下：

```
NAME                   TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
kubernetes-dashboard   NodePort   10.103.5.139   <none>        443:31795/TCP    4h
```

如上所示，Dashboard已经在 `31795` 端口上公开，现在可以在外部使用 `https://<cluster-ip>:31795` 进行访问。需要注意的是，在多节点的集群中，必须找到运行Dashboard节点的IP来访问，而不是Master节点的IP，在本文的示例，我部署了两台服务器，MatserIP为 `192.168.0.8` ，ClusterIP为 `192.168.0.10` 。

但是最后访问的结果可能如下：



遗憾的是，由于证书问题，我们无法访问，需要在部署Dashboard时指定有效的证书，才可以访问。由于在正式环境中，并不推荐使用NodePort的方式来访问Dashboard，故不再多说，关于如何为Dashboard配置证书可参考：Certificate management。

API Server

如果Kubernetes API服务器是公开的，并可以从外部访问，那我们可以直接使用API Server的方式来访问，也是比较推荐的方式。

Dashboard的访问地址为：
`https://<master-ip>:<apiserver-port>/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/` ，但是返回的结果可能如下：

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "services \"https:kubernetes-dashboard:\" is forbidden: User \"system:anonymous\" cannot get services/proxy in the namespace \"kube-system\"",
  "reason": "Forbidden",
  "details": {
    "name": "https:kubernetes-dashboard:",
```

```
    "kind": "services"
  },
  "code": 403
}
```

这是因为最新版的k8s默认启用了RBAC，并为未认证用户赋予了一个默认的身份： `anonymous` 。

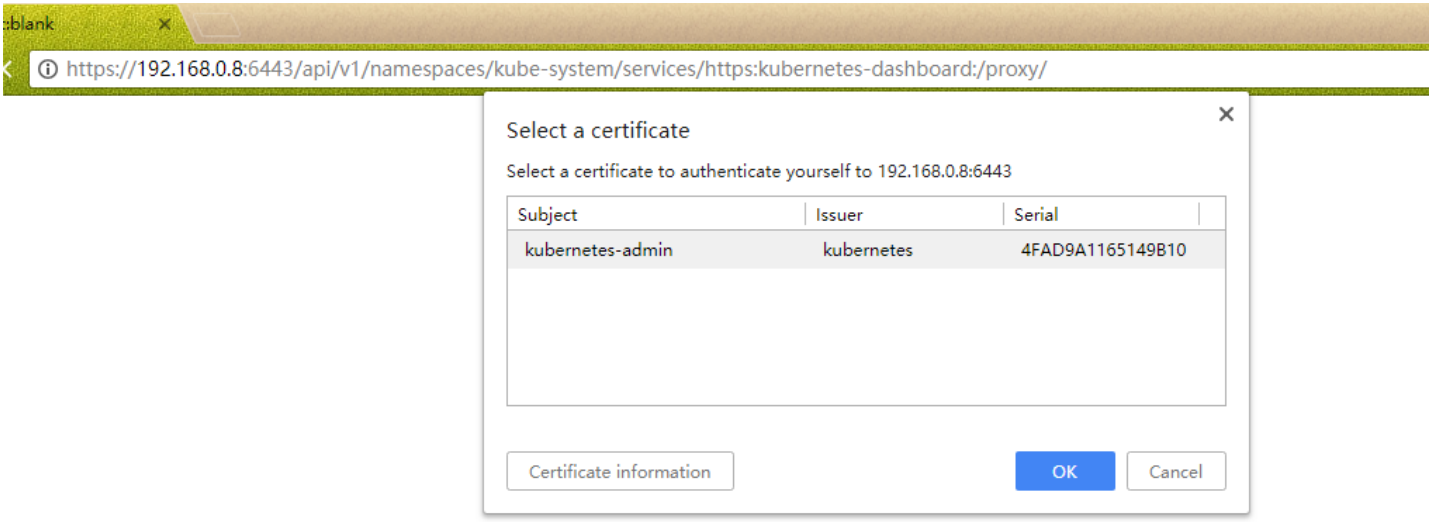
对于API Server来说，它是使用证书进行认证的，我们需要先创建一个证书：

1.首先找到 `kubectl` 命令的配置文件，默认情况下为 `/etc/kubernetes/admin.conf` ，在 上一篇 中，我们已经复制到了 `$HOME/.kube/config` 中。

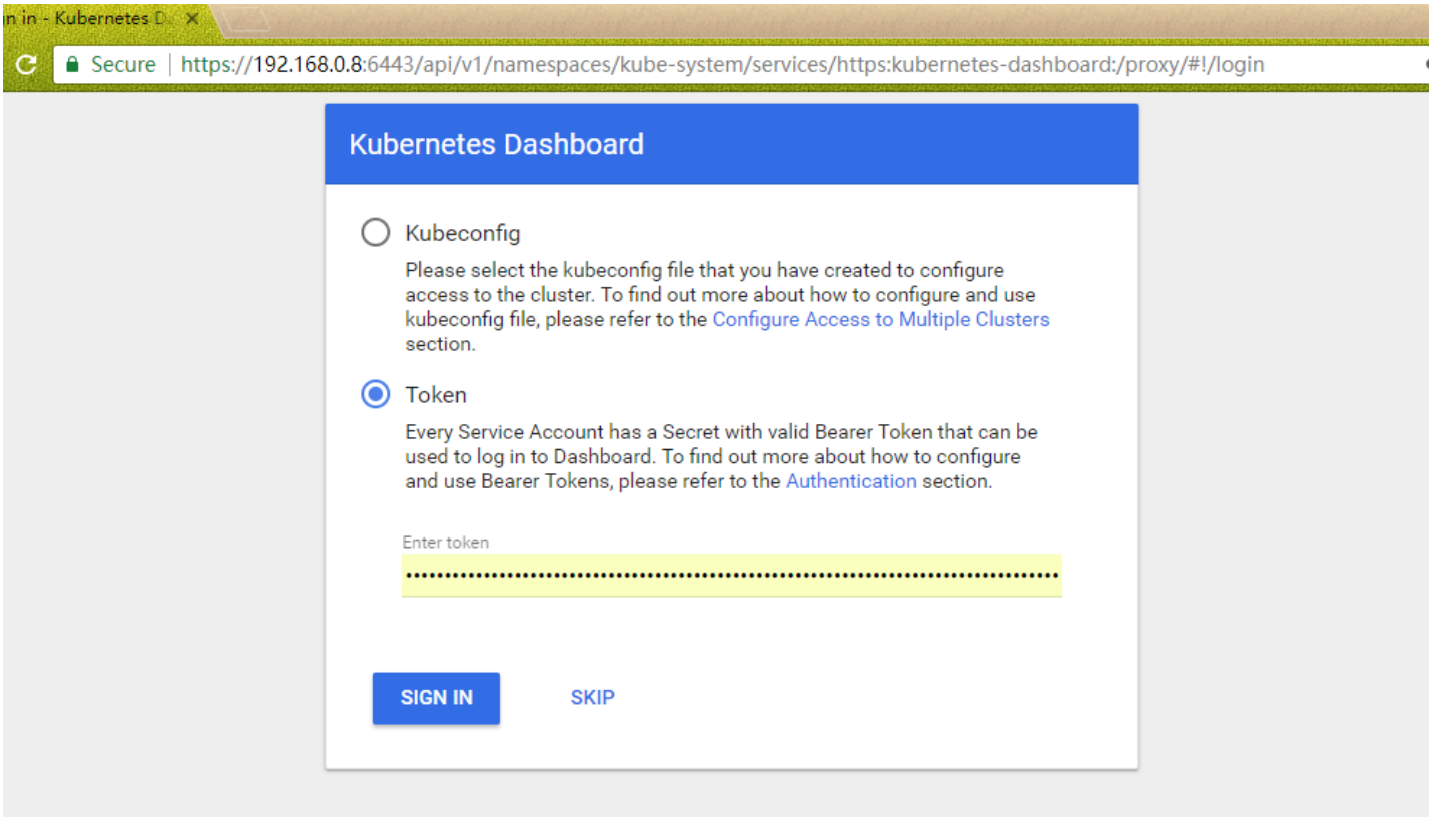
2.然后我们使用 `client-certificate-data` 和 `client-key-data` 生成一个*p12*文件，可使用下列命令：

```
# 生成client-certificate-data
grep 'client-certificate-data' ~/.kube/config | head -n 1 | awk '{print $2}' | base64 -d >> kubecfg.crt

# 生成client-key-data
grep 'client-key-data' ~/.kube/config | head -n 1 | awk '{print $2}' | base64 -d >> kubecfg.key

# 生成p12
openssl pkcs12 -export -clcerts -inkey kubecfg.key -in kubecfg.crt -out kubecfg.p12 -name "kubernetes-client"
```

3.最后导入上面生成的p12文件，重新打开浏览器，显示如下：



点击确定，便可以看到熟悉的登录界面了：



我们可以使用一开始创建的 `admin-user` 用户的token进行登录，一切OK。

> 对于生产系统，我们应该为每个用户应该生成自己的证书，因为不同的用户会有不同的命名空间访问权限。

Ingress

Ingress将开源的反向代理负载均衡器（如 Nginx、Apache、Haproxy等）与k8s进行集成，并可以动态的更新Nginx配置等，是比较灵活，更为推荐的暴露服务的方式，但也相对比较复杂，以后再来介绍。