RBAC：基于角色的访问控制，Role-Based Access Control。他是一种基于企业内个人角色来管理一些资源的访问方法。

Jenkins使用基于角色的用户权限管理。

RBAC：4中顶级资源，Role、ClusterRole、RoleBinding、ClusterRoleBinding。

Role：角色，包含一组权限的规则。没有拒绝规则，只是附加允许。Namespace隔离，只作用于命名空间内。

ClusterRole：和Role的区别，Role是只作用于命名空间内，作用于整个集群。

RoleBinding：作用于命令空间内，将ClusterRole或者Role绑定到User、Group、ServiceAccount。

ClusterRoleBinding：作用于整个集群。

ServiceAccount、User、Group。

Basic-auth-file

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

*# "namespace" omitted since ClusterRoles are not namespaced*

name: secret-reader

labels:

self-cluster-role: test

rules:

- apiGroups: [""]

*#*

*# at the HTTP level, the name of the resource for accessing Secret*

*# objects is "secrets"*

resources: ["secrets"]

verbs: ["get", "watch", "list"]

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: monitoring

aggregationRule:

clusterRoleSelectors:

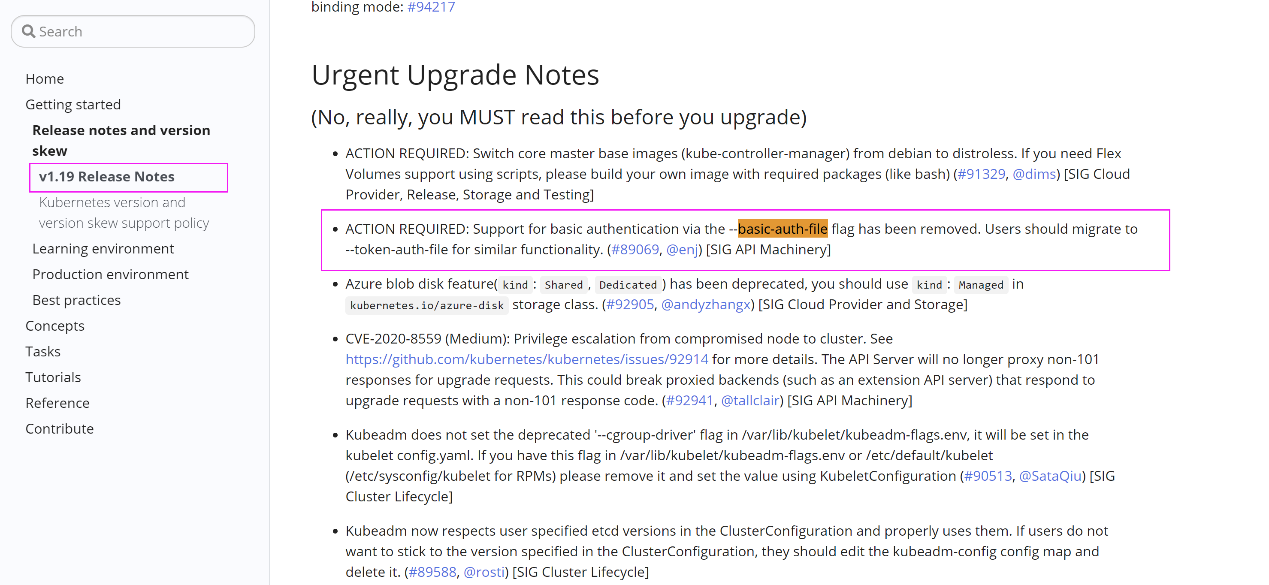
- matchLabels:

self-cluster-role: test

rules: [] *# The control plane automatically fills in the rules*

<https://kubernetes.io/docs/reference/access-authn-authz/rbac/>

注意：k8s 1.19+已经弃用basic\_auth\_file的认证方式，如果集群大于1.19+ 请使用课程讲解的serviceaccount的认证方式，更灵活更好用。



基于用户名密码实现不同用户有不同的权限

基于ServiceAccount实现不同的SA有不同的权限

- --authentication-mode=basic,token

- --token-ttl=86400

注意：k8s 1.18+后可以直接使用kubectl alpha debug使用临时容器（也需要开启feature），比如启动一个session直接进行调试：

kubectl alpha debug alpine-85949985b6-qqztl -i --image=busybox

或者给某个pod加上一个临时容器，名字叫debugger：

kubectl alpha debug --image=myproj/debug-tools -c debugger mypod

k8s 1.20+ 直接使用kubectl debug mypod -it --image=busybox即可，省去了kubectl alpha的alpha

临时容器：1.16+

就是在原有的Pod上，添加一个临时的Container，这个Container可以包含我们排查问题所有的工具，netstat、ps、top，jstat、jmap。

**--feature-gates=EphemeralContainers=true \**

【课程配置的修改是针对二进制安装方式的，kubeadm的配置文件需要参考kubeadm安装文档最后说明的配置文件的位置】

{

"apiVersion": "v1",

"kind": "EphemeralContainers",

"metadata": {

"name": "demo-nginx-7987dc97bf-4sqpl"

},

"ephemeralContainers": [{

"command": [

"sh"

],

"image": "busybox",

"imagePullPolicy": "IfNotPresent",

"name": "debugger",

"stdin": true,

"tty": true,

"terminationMessagePolicy": "File"

}]

}

[root@k8s-master01 ~]# kubectl replace --raw /api/v1/namespaces/default/pods/demo-nginx-7987dc97bf-4sqpl/ephemeralcontainers -f ec.json

{"kind":"EphemeralContainers","apiVersion":"v1","metadata":{"name":"demo-nginx-7987dc97bf-4sqpl","namespace":"default","selfLink":"/api/v1/namespaces/default/pods/demo-nginx-7987dc97bf-4sqpl/ephemeralcontainers","uid":"32d8ddda-1dfd-487a-aa83-40f1c1bf89ee","resourceVersion":"4345959","creationTimestamp":"2020-03-25T15:34:26Z"},"ephemeralContainers":[{"name":"debugger","image":"busybox","command":["sh"],"resources":{},"terminationMessagePolicy":"File","imagePullPolicy":"IfNotPresent","stdin":true,"tty":true}]}

文档地址：<https://kubernetes.io/docs/concepts/workloads/pods/ephemeral-containers/>

DaemonSet需要单独配置：shareProcessNamespace

Deployment不需要单独配置。

StatefulSet