Ingress 使用帮助

1. 简介

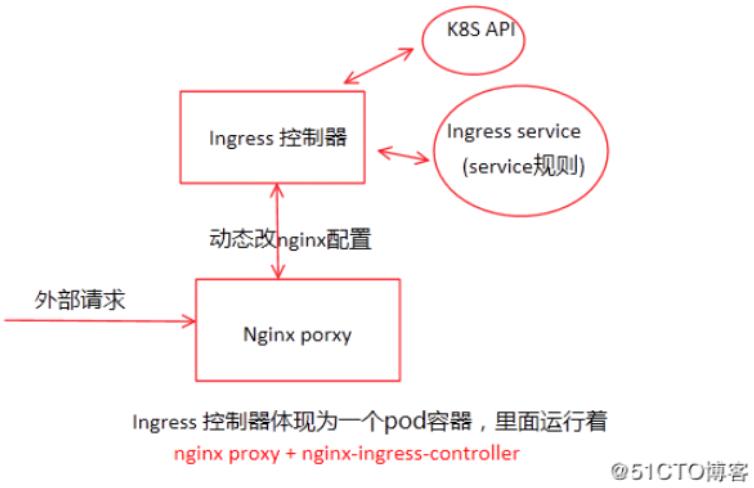
对外暴露集群服务，还有其他的暴露方式如(LoadBlancer Service, NodePort Service)
Ingress还可以提供负载平衡，SSL准入控制和基于名称的虚拟主机。

Ingress组成：

Ingress Controller
Ingress 服务

工作方式：

ingress Contronler 通过与 Kubernetes API 交互，动态的去感知集群中 Ingress 规则变化，然后读取它，按照自定义的规则，规则就是写明了哪个域名对应哪个service，生成一段 Nginx 配置，再写到 Nginx-ingress-control的 Pod 里，这个 Ingress Contronler 的pod里面运行着一个nginx服务，控制器会把生成的nginx配置写入/etc/nginx.conf文件中，然后 reload 一下 使用配置生效。以此来达到域名分配置及动态更新的问题。



2. 创建服务账号ServiceAccount

serviceAccount 作为ingress controller 去访问apiserver时的鉴权账号

```
+-------------+   service          +------------+
|             |   authentication   |            |
+  apiserver  +<-------------------+  ingress   |
|             |                    | controller |
+-------------+                    +------------+
```

2.1 rbac 基于角色的访问控制

需要创建 一个服务账号，一个普通角色，一个集群角色。规定好角色的对api资源的访问权限，然后将服务账号绑定至2个角色，这样服务账号就拥有了和角色相同的权限。

通过yaml文件创建如下对象：

- ServiceAccount
- Role
- ClusterRole
- RoleBinding
- ClusterRoleBinding

RBAC API中，通过如下的步骤进行授权：

1）定义角色：定义角色时会指定此角色对于资源的访问控制的规则；

定义角色三个要素：

- 主题：想要访问Kubernetes API的用户和进程
- 资源：集群中可用的Kubernetes API对象，像Pod、Deployments、Services、Nodes和PersistentVolumes等
- 动词：可以对上述资源执行的一组操作。可以使用不同的动词（像get、watch、create、delete等），但最终所有动词都是创建、读取更新或删除（CRUD）操作。

普通角色只能被授予访问单一命令空间中的资源。

集群角色(ClusterRole)能够被授予资源权限有：集群范围资源（Node、NameSpace）、非资源端点（/healthz）、集群所有命名空间资源（跨名称空间）

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: nginx-ingress-clusterrole
rules:        #规则
  - apiGroups:      # 所有核心api
      - ""
    resources:       # 资源
      - configmaps
      - endpoints
      - nodes
      - pods
      - secrets
    verbs:            #操作
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - get
  - apiGroups:
      - ""
    resources:
      - services
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - "extensions"
    resources:
      - ingresses
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
        - events
    verbs:
        - create
        - patch
  - apiGroups:
      - "extensions"
    resources:
      - ingresses/status
    verbs:
      - update
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: Role
metadata:
  name: nginx-ingress-role
  namespace: kube-system
rules:
  - apiGroups:
      - ""
    resources:
      - configmaps
      - pods
      - secrets
      - namespaces
    verbs:
      - get
  - apiGroups:
      - ""
    resources:
      - configmaps
    resourceNames:
      - "ingress-controller-leader-nginx"
    verbs:
      - get
      - update
  - apiGroups:
      - ""
    resources:
      - configmaps
    verbs:
```

```
      - create
  - apiGroups:
      - ""
    resources:
      - endpoints
    verbs:
      - get
      - create
      - update
```

2）定义主体：用户、组和服务账户

创建ingress的服务账号

```
apiVersion: v1
kind: ServiceAccount #角色
metadata:
  name: nginx-ingress-serviceaccount
  namespace: kube-system
```

更多使用示例

名称为 demo 用户：

```
subjects:
- kind:User
  name:"demo"
  apiGroup:rbac.authorization.k8s.io
```

名称为 demo 组：

```
subjects:
- kind:Group
  name:"demo-group"
  apiGroup:rbac.authorization.k8s.io
```

kube-system命名空间中，名称为default的服务帐户

```
subjects:
- kind:ServiceAccount
  name:default
  namespace:kube-system
```

so命名空间中，所有的服务帐户：

```
subjects:
- kind:Group
  name:system:serviceaccounts:so
  apiGroup:rbac.authorization.k8s.io
```

所有的服务帐户：

```
subjects:
- kind:Group
  name:system:serviceaccounts
  apiGroup:rbac.authorization.k8s.io
```

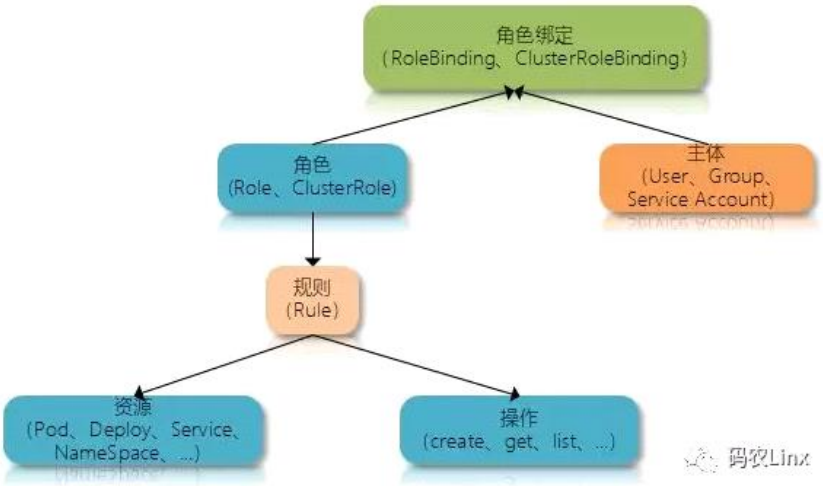
所有用户:

```
subjects:
- kind:Group
  name:system:authenticated    #授权用户
  apiGroup:rbac.authorization.k8s.io - kind:Group
  name:system:unauthenticated  #未授权用户
  apiGroup:rbac.authorization.k8s.io
```

3）绑定角色：将主体与角色进行绑定，对主体进行访问授权

角色绑定用于将角色与一个主体进行绑定，从而实现将对主体授权的目的，主体分为用户、组和服务帐户。

角色绑定分为：普通角色绑定和集群角色绑定



RBAC API中的对象关系图

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: RoleBinding
metadata:
  name: nginx-ingress-role-nisa-binding
  namespace: kube-system
roleRef:    # 上面定义的角色
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: nginx-ingress-role
subjects:   # 上面定义的服务账户
  - kind: ServiceAccount
    name: nginx-ingress-serviceaccount
    namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: nginx-ingress-clusterrole-nisa-binding
roleRef:    # 上面定义的角色
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: nginx-ingress-clusterrole
subjects:   # 上面定义的服务账户
  - kind: ServiceAccount
    name: nginx-ingress-serviceaccount
    namespace: kube-system
```

校验权限:

```
kubectl get service -n kube-system --as system:serviceaccount:kube-system:nginx-ingress-serviceaccount
```

```
[root@bigdata ~]# kubectl get service -n kube-system --as system:serviceaccount:kube-system:nginx-ingress-serviceaccount
NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)             AGE
default-http-backend   ClusterIP   10.110.58.84     <none>        80/TCP              18h
heapster               ClusterIP   10.99.21.25      <none>        80/TCP              25d
jenkins                ClusterIP   10.110.110.81    <none>        8080/TCP,5000/TCP   16h
kube-dns               ClusterIP   10.96.0.10       <none>        53/UDP,53/TCP       25d
kubernetes-dashboard   NodePort    10.102.255.174   <none>        443:30000/TCP       25d
monitoring-grafana     ClusterIP   10.102.150.81    <none>        80/TCP              25d
monitoring-influxdb    ClusterIP   10.108.130.247   <none>        8086/TCP            25d
```

```
[root@bigdata ~]# kubectl describe clusterrole nginx-ingress-clusterrole
Name:         nginx-ingress-clusterrole
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources                  Non-Resource URLs  Resource Names  Verbs
  ---------                  -----------------  --------------  -----
  configmaps                 []                 []              [list watch]
  endpoints                  []                 []              [list watch]
  events                     []                 []              [create patch]
  nodes                      []                 []              [list watch get]
  pods                       []                 []              [list watch]
  secrets                    []                 []              [list watch]
  services                   []                 []              [get list watch]
  ingresses.extensions       []                 []              [get list watch]
  ingresses.extensions/status  []               []              [update]
[root@bigdata ~]# kubectl get events -n kube-system --as system:serviceaccount:kube-system:nginx-ingress-serviceaccount
Error from server (Forbidden): events is forbidden: User "system:serviceaccount:kube-system:nginx-ingress-serviceaccount" cannot list events in the namespace "kube-system"
[root@bigdata ~]#
```

3. 创建ingress 默认后端

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: default-http-backend
  labels:
    k8s-app: default-http-backend
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        k8s-app: default-http-backend
    spec:
      terminationGracePeriodSeconds: 60
      containers:
      - name: default-http-backend
        image: gcr.io/google_containers/defaultbackend:1.0
        imagePullPolicy: IfNotPresent
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 30   #30s检测一次/healthz
          timeoutSeconds: 5
        ports:
        - containerPort: 8080
        resources:
          limits:
            cpu: 10m
            memory: 20Mi
          requests:
            cpu: 10m
            memory: 20Mi
      nodeSelector:
        kubernetes.io/hostname: bigdata
---
apiVersion: v1
kind: Service
metadata:
  name: default-http-backend
  namespace: kube-system
  labels:
    k8s-app: default-http-backend
spec:
  ports:
  - port: 80
    targetPort: 8080
  selector:
    k8s-app: default-http-backend
```

```
[root@bigdata ~]# kubectl get service -n kube-system
NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)             AGE
default-http-backend   ClusterIP   10.110.58.84     <none>        80/TCP              18h
heapster               ClusterIP   10.99.21.25      <none>        80/TCP              25d
jenkins                ClusterIP   10.110.110.81    <none>        8080/TCP,5000/TCP   16h
kube-dns               ClusterIP   10.96.0.10       <none>        53/UDP,53/TCP       25d
kubernetes-dashboard   NodePort    10.102.255.174   <none>        443:30000/TCP       25d
monitoring-grafana     ClusterIP   10.102.150.81    <none>        80/TCP              25d
monitoring-influxdb    ClusterIP   10.108.130.247   <none>        8086/TCP            25d
[root@bigdata ~]# curl 10.110.58.84
default backend - 404[root@bigdata ~]#
```

4. 创建ingress控制器

ingress 控制器可以使用daemonset和deployment两种方式创建，为了避免资源浪费不建议用daemonset。我们用deployment启动，hostport方式暴露宿主机端口：80， 443， 8080

--ingress-class must be changed to a value unique for the cluster within the definition of the replication controller

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-ingress-controller
  labels:
    k8s-app: nginx-ingress-controller
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        k8s-app: nginx-ingress-controller
    spec:
      terminationGracePeriodSeconds: 60
```

```
      hostNetwork: true
      serviceAccountName: nginx-ingress-serviceaccount
      containers:
      - image: gcr.io/google_containers/nginx-ingress-controller:0.9.0-beta.1
        imagePullPolicy: IfNotPresent
        name: nginx-ingress-controller
        readinessProbe:
          httpGet:
            path: /healthz
            port: 10254
            scheme: HTTP
        livenessProbe:
          httpGet:
            path: /healthz
            port: 10254
            scheme: HTTP
          initialDelaySeconds: 10
          timeoutSeconds: 1
        ports:
        - containerPort: 80
          hostPort: 80
          name: http
          protocol: TCP
        - containerPort: 8080
          hostPort: 8080
          name: http
          protocol: TCP
        - containerPort: 443
          hostPort: 443
          name: http
          protocol: TCP
        env:
          - name: POD_NAME
            valueFrom:
              fieldRef:
                fieldPath: metadata.name
          - name: POD_NAMESPACE
            valueFrom:
              fieldRef:
                fieldPath: metadata.namespace
        args:
        - /nginx-ingress-controller
        - --default-backend-service=$(POD_NAMESPACE)/default-http-backend
      nodeSelector:
        kubernetes.io/hostname: bigdata
```

5. 创建后端服务

我们起一个jenkins应用做示范

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: jenkins
  namespace: kube-system
spec:
  replicas: 1
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 2
      maxUnavailable: 0
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
      - name: jenkins
        image: jenkins:2.7.2
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
          name: web
          protocol: TCP
        - containerPort: 50000
          name: agent
          protocol: TCP
        volumeMounts:
        - name: jenkinshome
          mountPath: /jenkins_home
        env:
        - name: JAVA_OPTS
          value: "-Duser.timezone=Asia/Shanghai"
      volumes:
      - name: jenkinshome
        hostPath:
          path: /var/jenkins_home
---
kind: Service
apiVersion: v1
metadata:
  name: jenkins
  namespace: kube-system
spec:
```

```
  selector:
    app: jenkins
  ports:
  - name: web
    port: 8080
    targetPort: 8080
  - name: agent
    port: 5000
    targetPort: 5000
```

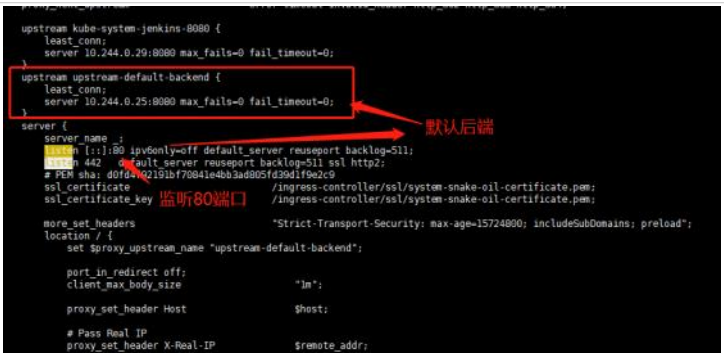
6. 创建ingress 路由规则

6.1 在不创建ingress规则的情况下

直接访问node ip

进入容器查看nignx配置

kubectl exec -it nginx-ingress-controller-67d99bfd6-7kpbf sh -n kube-system
cat /etc/nginx/nginx.conf



6.2 创建ingress， 下面的ingress创建了2条路由规则

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: jenkins-ingress
  namespace: kube-system
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/affinity: cookie
    nginx.ingress.kubernetes.io/session-cookie-hash: sha1
    nginx.ingress.kubernetes.io/session-cookie-name: route
spec:
  rules:
  - http:     # 直接通过ingress controller 的ip访问
      paths:
      - backend:
          serviceName: jenkins
          servicePort: 8080
  - host: bar.foo.com    # 通过虚拟域名访问
    http:
      paths:
      - backend:
          serviceName: jenkins
          servicePort: 8080
```

```
[root@bigdata ingress]# kubectl create -f jenkins-ingress.yaml
ingress.extensions "jenkins-ingress" created
[root@bigdata ingress]# kubectl describe ingress -n kube-system
Name:             jenkins-ingress
Namespace:        kube-system
Address:
Default backend:  default-http-backend:80 (10.244.0.25:8080)
Rules:
  Host         Path  Backends
  ----         ----  --------
  *
                     jenkins:8080 (10.244.0.29:8080)
  bar.foo.com
                     jenkins:8080 (10.244.0.29:8080)
Annotations:
  kubernetes.io/ingress.class:                      nginx
  nginx.ingress.kubernetes.io/affinity:             cookie
  nginx.ingress.kubernetes.io/session-cookie-hash:  sha1
  nginx.ingress.kubernetes.io/session-cookie-name:  route
Events:
  Type    Reason  Age   From                  Message
  ----    ------  ----  ----                  -------
  Normal  CREATE  11s   ingress-controller  Ingress kube-system/jenkins-ingress
```

node ip访问

虚拟域名访问

6.3 官方示例： https://kubernetes.io/docs/concepts/services-networking/ingress/

```
foo.bar.com -> 178.91.123.132 -> / foo    s1:80
                                  / bar    s2:80
```

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: foo.bar.com
```

```
    http:
      paths:
      - path: /foo
        backend:
          serviceName: s1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: s2
          servicePort: 80
```

6.4 TLS 路由创建

具体使用这里暂不做介绍

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    # Enable client certificate authentication
    nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"
    # Create the secret containing the trusted ca certificates with `kubectl create secret generic auth-tls-chain --from-
file=ca.crt --namespace=default`
    # NB: The file _must_ be named "ca.crt" and nothing else. This filename is expected to be found in the secret.
    nginx.ingress.kubernetes.io/auth-tls-secret: "default/auth-tls-chain"
    # Specify the verification depth in the client certificates chain
    nginx.ingress.kubernetes.io/auth-tls-verify-depth: "1"
    # Specify an error page to be redirected to verification errors
    nginx.ingress.kubernetes.io/auth-tls-error-page: "http://www.mysite.com/error-cert.html"
    # Specify if certificates are passed to upstream server
    nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream: "false"
  name: nginx-test
  namespace: default
spec:
  rules:
  - host: ingress.test.com
    http:
      paths:
      - backend:
          serviceName: http-svc:80
          servicePort: 80
        path: /
  tls:
  - hosts:
    - ingress.test.com
    secretName: tls-secret
```

7. 在kubernetes集群中大量使用

7.1 隔离

通过污点的功能将ingress controller 节点和其他节点隔离

7.2 一个应用创建一个ingress controller

同一台机器上要起多个ingress controller，为避免端口冲突，需要做好端口规划，并将容器的80端口通过桥接的方式暴露出来

7.3 一个应用创建一个ingress rule

service 和 绑定到指定的ingress上通过注释 kubernetes.io/ingress.class: test2

而 ingress controller 在args中生命 --ingress-class=test2

8. 参考链接

ingress 部署: http://blog.51cto.com/newfly/2060587

rbac: https://mp.weixin.qq.com/s?__biz=MzI3MzQ3NDMzNw==&mid=2247483765&idx=1&sn=aa0fe555392d7c767757a9d5b80b69ad&chksm=eb23f73bdc547e2db6ef5af5cd218b0bee8f9e58ca8ba4d948b6a3ed3261822d83c60c331739&scene=7#rd

jenkins 安装: https://yq.aliyun.com/articles/622521

ingress-nginx project： https://github.com/kubernetes/ingress-nginx

annotations: https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/nginx-configuration/annotations.md