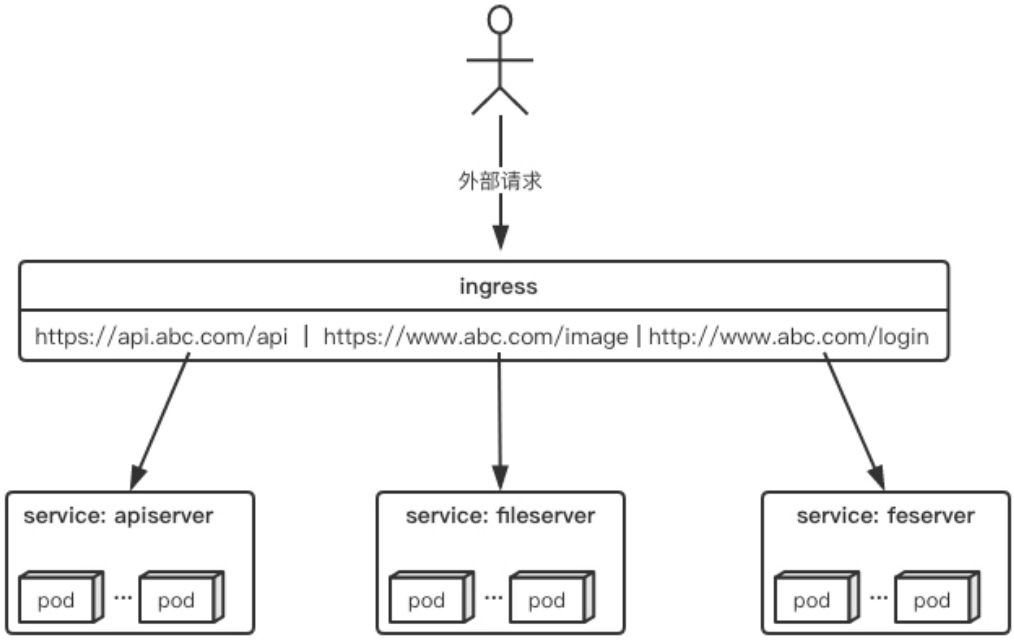
Ingress

一、为什么引入Ingress?

我们说k8s 的服务(service)时说暴露了service的三种方式ClusterIP、NodePort与LoadBalance，这几种方式都是在service的维度提供的，service的作用体现在两个方面，对集群内部，它不断跟踪pod的变化，更新endpoint中对应pod的对象，提供了ip不断变化的pod的服务发现机制，对集群外部，他类似负载均衡器，可以在集群内外部对pod进行访问。但是，单独用service暴露服务的方式，在实际生产环境中不太合适：

- ClusterIP的方式只能在集群内部访问。
- NodePort方式的话，测试环境使用还行，当有几十上百的服务在集群中运行时，NodePort的端口管理是灾难。
- LoadBalance方式受限于云平台，且通常在云平台部署ELB还需要额外的费用。

  所幸k8s还提供了一种集群维度暴露服务的方式，也就是ingress。ingress可以简单理解为service的service，他通过独立的ingress对象来制定请求转发的规则，把请求路由到一个或多个service中。这样就把服务与请求规则解耦了，可以从业务维度统一考虑业务的暴露，而不用为每个service单独考虑。
  举个例子，现在集群有api、文件存储、前端3个service，可以通过一个ingress对象来实现图中的请求转发：



ingress规则是很灵活的，可以根据不同域名、不同path转发请求到不同的service，并且支持https/http。

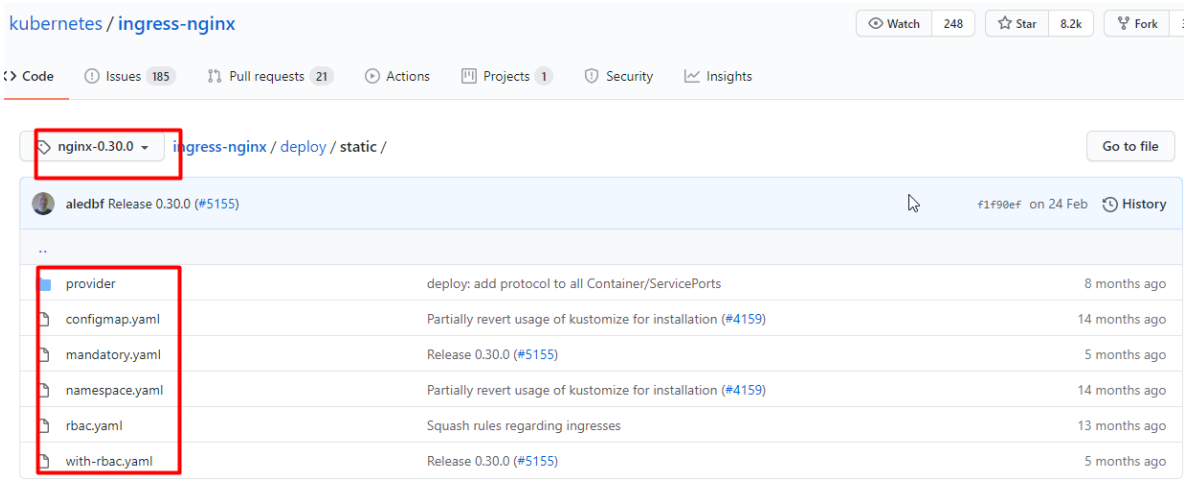
二、Ingress部署

1、部署文件

Ingress-nginx gitHub地址：最新版本地址：https://github.com/kubernetes/ingress-nginx/

指定版本的下载地址： https://github.com/kubernetes/ingress-nginx/tree/nginx-0.30.0/deploy

Ingress-nginx 官方网站地址：https://kubernetes.github.io/ingress-nginx/



部署文件介绍

① namespace.yaml

创建独立的命名空间 ingress-nginx

② configmap.yaml

③ rbac.yaml

负责Ingress的RBAC授权的控制，其创建了Ingress用到的ServiceAccount、ClusterRole、Role、RoleBinding、ClusterRoleBinding

④ with-rbac.yaml

使用带rbac的方式创建ingress-controller，是整个ingress的核心部署文件。

⑤ mandatory.yaml

> 该文件整合了前面四项文件的内容，是用于实际部署ingress服务的yaml文件。即只需要使用该文件就可以完成ingress-controller的全部部署工作。

2、部署Ingress

```
# 下载所需的Ingress
# 部署ingress-controller，注意这个配置文件下载的地址需要解决科学上网的问题，建议从github下载即可
wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/mandatory.yaml
# 对外部提供服务所需
wget  https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/baremetal/service-nodeport.yaml

#修改mandatory.yaml，替换镜像地址部署文件中默认镜像地址为：(此镜像地址在国内无法下载，解决科学上网问题)
quay.io/kubernetes-ingress-controller/nginx-ingress-controller:0.23.0
```

```
# 修改为（阿里云镜像地址），版本可以根据github tag来进行选择
registry.cn-hangzhou.aliyuncs.com/google_containers/nginx-ingress-
controller:0.23.0

# 修改文件中部署方式的部分，将默认的Deployment修改为Daemonset，去掉replicas字段：
...省略部分...
apiVersion: apps/v1
kind: DaemonSet  # 这里默认为Deployment，修改为Daemonset
metadata:
  name: nginx-ingress-controller
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
spec:
  replicas: 1  # 删除这行
...省略部分...

# 修改service-nodeport.yaml，增加NodePort，固定端口
apiVersion: v1
kind: Service
metadata:
  name: ingress-nginx
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
spec:
  type: NodePort
  ports:
    - name: http
      port: 80
      targetPort: 80
      nodePort: 30080  #http
      protocol: TCP
    - name: https
      port: 443
      targetPort: 443
      nodePort: 30443  #https
      protocol: TCP
  selector:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx


# 部署nginx-ingress-controller
kubectl apply -f mandatory.yaml
kubectl apply -f service-nodeport.yaml

# 查看Ingress组件
kubectl get pods -n ingress-nginx
kubectl get pod -n ingress-nginx --show-labels
kubectl get svc -n ingress-nginx

# 查看daemonset的状态
kubectl get daemonset -n ingress-nginx

# 验证ingress-nginx服务
```

```
curl http://192.168.66.10:30080/healthz -I
HTTP/1.1 200 OK
Server: nginx/1.17.8
Date: Mon, 03 Aug 2020 18:56:44 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive


# 创建后端服务
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
spec:
  selector:
    app: nginx
  ports:
  - port: 80
    targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: hub.kaikeba.com/java12/myapp:v1
        ports:
        - containerPort: 80

# 创建ingress资源，将nginx服务添加到Ingress-nginx中
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: ingress.kaikeba.com
    http:
      paths:
```

```
        - path: /
          backend:
            serviceName: nginx
            servicePort: 80

# 访问方式
ingress.kaikeba.com:30080

# 创建tomcat后端服务，文件内容如下
apiVersion: v1
kind: Service
metadata:
  name: tomcat
  namespace: default
spec:
  selector:
    app: tomcat
  ports:
  - port: 8080
    targetPort: 8080
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: hub.kaikeba.com/java12/tomcat:v1
        ports:
        - containerPort: 8080

# 修改ingress-nginx.yaml，将tomcat服务增加到Ingress-nginx中。这里有多种策略配置。
# 不同域名转发到不同的服务
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  rules:
  - host: ingress1.kaikeba.com
    http:
      paths:
        - path: /
          backend:
```

```
            serviceName: nginx
            servicePort: 80
  - host: ingress2.kaikeba.com
    http:
      paths:
        - path: /
          backend:
            serviceName: tomcat
            servicePort: 8080

# 同一域名，使用不同的URL转发到不同的服务上
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  rules:
  - host: ingress.kaikeba.com
    http:
      paths:
        - path: /nginx
          backend:
            serviceName: nginx
            servicePort: 80
        - path: /tomcat
          backend:
            serviceName: tomcat
            servicePort: 8080
```

3、https

这里用给tomcat服务添加证书为例。

- 生成私钥

```
openssl genrsa -out tls.key 2048
```

- 自签发证书

```
openssl req -new -x509 -key tls.key -out tls.crt -subj
/C=CN/ST=Shanghai/L=Shanghai/O=DevOps/CN=tomcat.ikiwi.me
```

- 创建K8S使用的证书配置文件Secret

```
kubectl create secret tls tomcat-ingress-secret --cert=tls.crt --key=tls.key
```

- 查看Secret描述

```
kubectl describe secret tomcat-ingress-secret
```

- 创建带tls认证的tomcat后端服务

ingress-tomcat-tls.yaml

```yaml
# 创建好后，访问服务，结果如下图所示，由于是自签发证书不受信任。HTTPS已可以成功访问。
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-tomcat-tls
  namespace: default
  annotations:
    kubernetes.io/ingress.class: "nginx"
  labels:
    app: tomcat
spec:
  tls:
  - hosts:
    - tomcat.ikiwi.me
    secretName: tomcat-ingress-secret
  rules:
  - host: tomcat.ikiwi.me
    http:
      paths:
        - backend:
            serviceName: tomcat
            servicePort: 8080
```