Istio - installation and setup

***Note**: Please go to Below Link for Official Doc's*
https://istio.io/latest/docs/setup/getting-started/

Istio_architechure -
https://drive.google.com/file/d/1py8E0WKVKO8YkmxDn2rYVxqWy63FN1Or/view

➢ Download **Istio** & install **istioctl** using the following command:

- `curl -L https://istio.io/downloadIstio | sh -`
- `cd istio-1.15.3`
- `export PATH=$PWD/bin:$PATH`

➢ Install **Istio** using the following command:

1. For this installation, we use the demo configuration profile. It's selected to have a good set of defaults for testing, but there are other profiles for production or performance testing.

Link:
https://aws.amazon.com/blogs/containers/secure-end-to-end-traffic-on-amazon-eks-using-tls-certificate-in-acm-alb-and-istio/

```
istioctl install \
--set profile=demo \
 --set values.gateways.istio-ingressgateway.type=NodePort
```

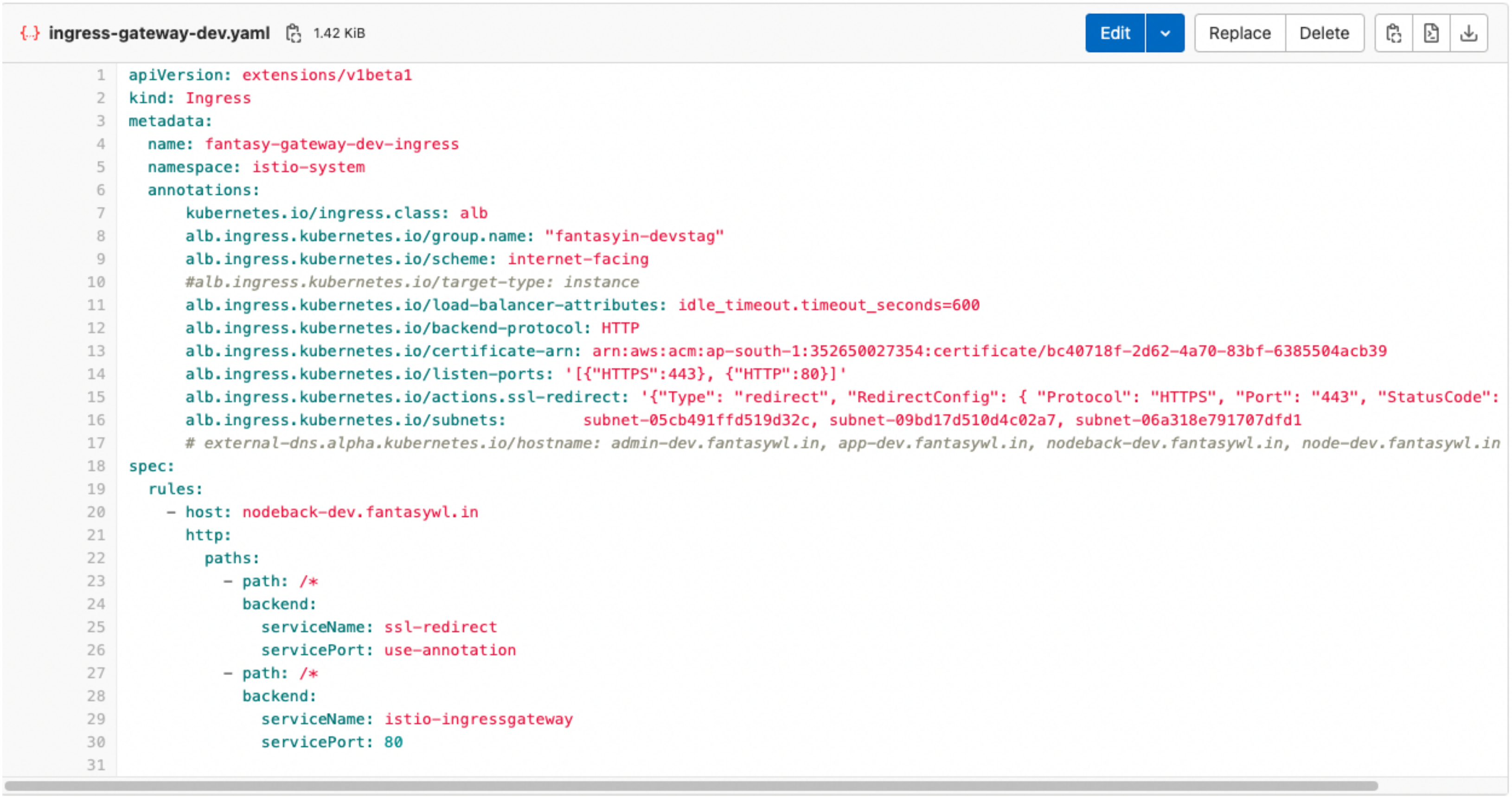
2. Add a namespace label to instruct Istio to automatically inject Envoy sidecar proxies when you deploy your application later:

- `kubectl label namespace default istio-injection=enabled`
- `kubectl get ns -L istio-injection`

3. Now Deploy your application and to expose that file you have to create 3 Files which are.
    a. Ingress.yaml

b. Gateway.yaml
c. VirtualService.yaml

4. Creating Ingress file:
   Note: In the rule, you have to and backend server as istio-ingress gateway as shown below

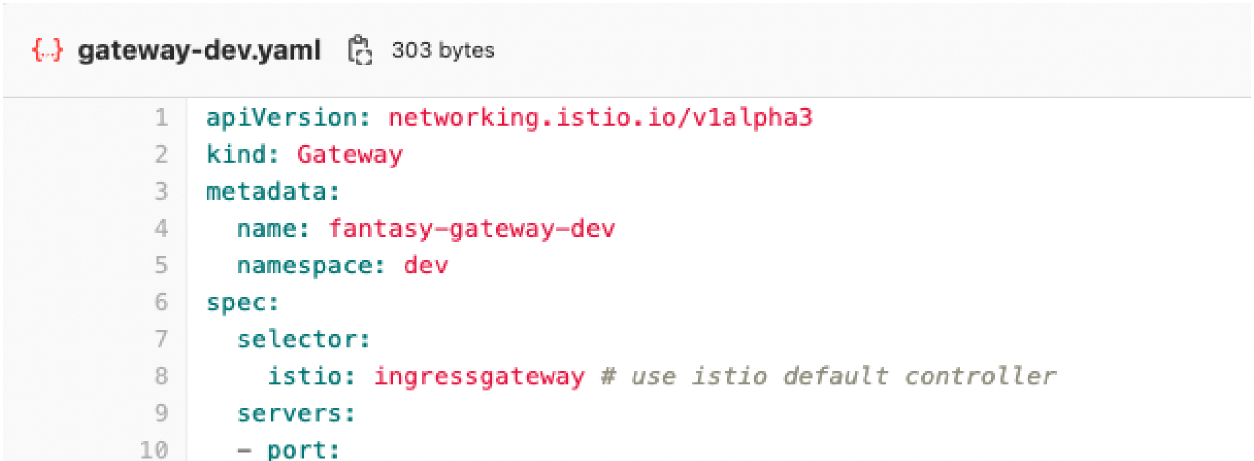


ingress-gateway-dev.yaml 1.42 KiB

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: fantasy-gateway-dev-ingress
  namespace: istio-system
  annotations:
      kubernetes.io/ingress.class: alb
      alb.ingress.kubernetes.io/group.name: "fantasyin-devstag"
      alb.ingress.kubernetes.io/scheme: internet-facing
      #alb.ingress.kubernetes.io/target-type: instance
      alb.ingress.kubernetes.io/load-balancer-attributes: idle_timeout.timeout_seconds=600
      alb.ingress.kubernetes.io/backend-protocol: HTTP
      alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:ap-south-1:352650027354:certificate/bc40718f-2d62-4a70-83bf-6385504acb39
      alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'
      alb.ingress.kubernetes.io/actions.ssl-redirect: '{"Type": "redirect", "RedirectConfig": { "Protocol": "HTTPS", "Port": "443", "StatusCode":
      alb.ingress.kubernetes.io/subnets:        subnet-05cb491ffd519d32c, subnet-09bd17d510d4c02a7, subnet-06a318e791707dfd1
      # external-dns.alpha.kubernetes.io/hostname: admin-dev.fantasywl.in, app-dev.fantasywl.in, nodeback-dev.fantasywl.in, node-dev.fantasywl.in
spec:
  rules:
    - host: nodeback-dev.fantasywl.in
      http:
        paths:
          - path: /*
            backend:
              serviceName: ssl-redirect
              servicePort: use-annotation
          - path: /*
            backend:
              serviceName: istio-ingressgateway
              servicePort: 80

```

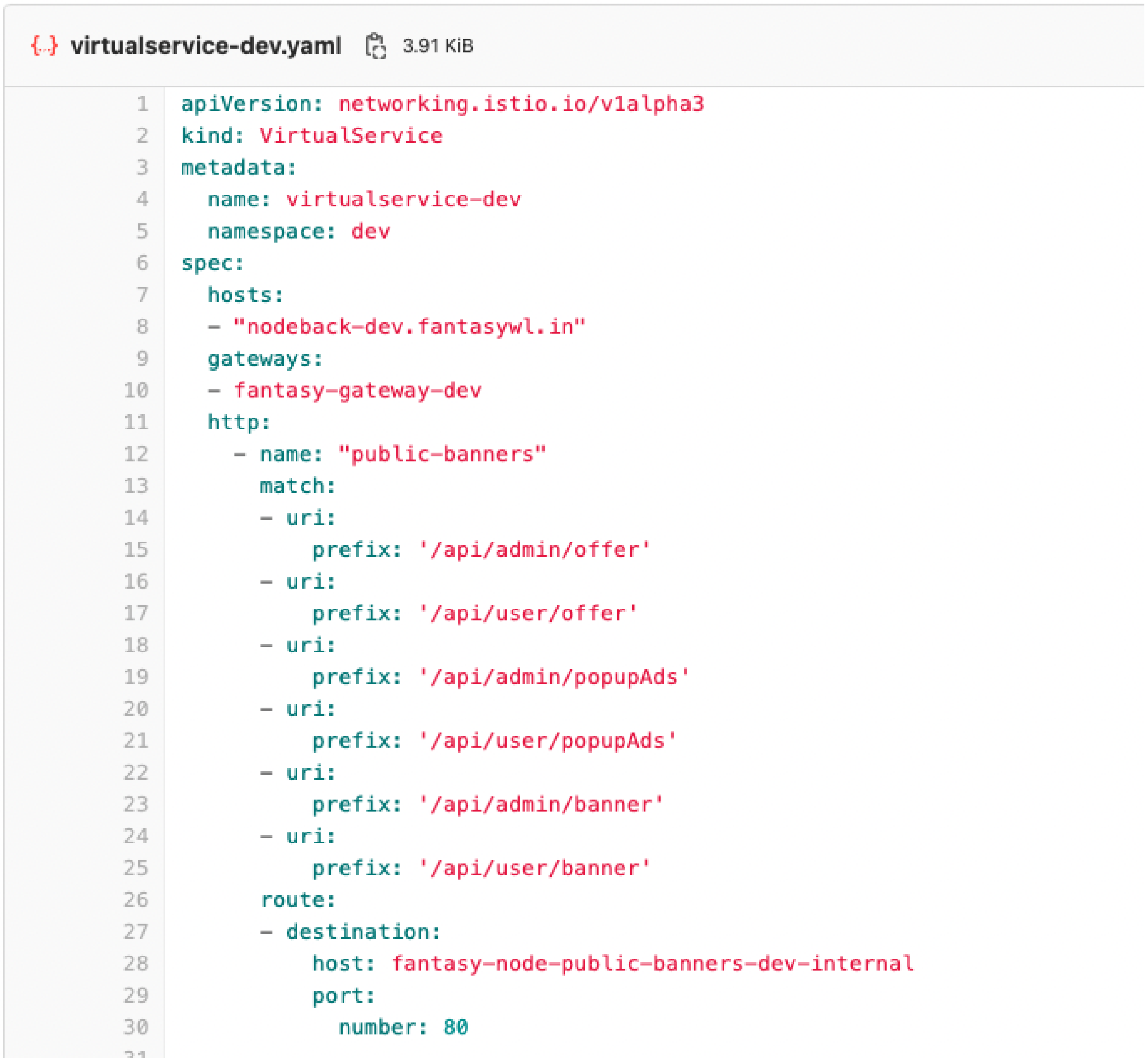
5. Creating Gateway file as below.



gateway-dev.yaml 303 bytes

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: fantasy-gateway-dev
  namespace: dev
spec:
  selector:
    istio: ingressgateway # use istio default controller
  servers:
  - port:
```

6. Creating VirtualService file as below.



virtualservice-dev.yaml 3.91 KiB

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: virtualservice-dev
  namespace: dev
spec:
  hosts:
  - "nodeback-dev.fantasywl.in"
  gateways:
  - fantasy-gateway-dev
  http:
    - name: "public-banners"
      match:
      - uri:
          prefix: '/api/admin/offer'
      - uri:
          prefix: '/api/user/offer'
      - uri:
          prefix: '/api/admin/popupAds'
      - uri:
          prefix: '/api/user/popupAds'
      - uri:
          prefix: '/api/admin/banner'
      - uri:
          prefix: '/api/user/banner'
      route:
      - destination:
          host: fantasy-node-public-banners-dev-internal
          port:
            number: 80
```

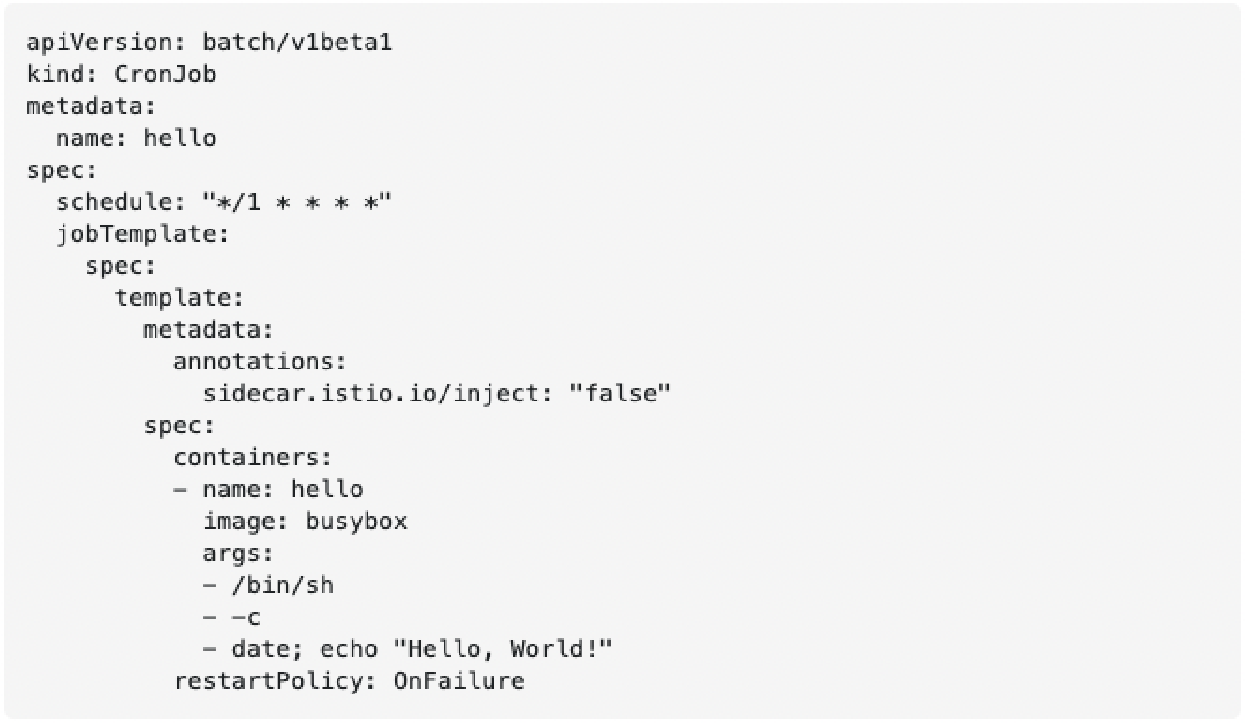
7. After applying all this file’s now u can access your application via Domain.

➢ Disable ***Istio side-car*** to inject in ***Cronjobs***

➔ Use the below **Annotation** to disable **side-car** in a single Deployment or Cronjobs

annotations: sidecar.istio.io/inject: "false"

Link:
https://stackoverflow.com/questions/65807748/disable-istio-sidecar-injection-to-the-job-pod



```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        metadata:
          annotations:
            sidecar.istio.io/inject: "false"
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo "Hello, World!"
          restartPolicy: OnFailure
```

➔ Disable **Istio side-car** in all Cronjobs with a single change

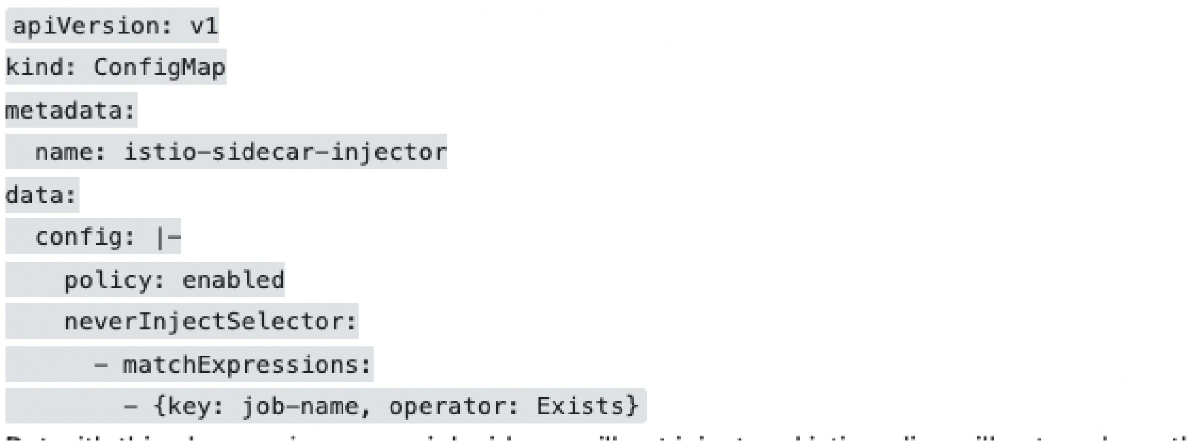
Link:
https://stackoverflow.com/questions/54921054/terminate-istio-sidecar-istio-proxy-for-a-kubernetes-job-cronjob

- Search for `istio-sidecar-injector` Configmap under `istio-system` namespace

And add this section as shown below.

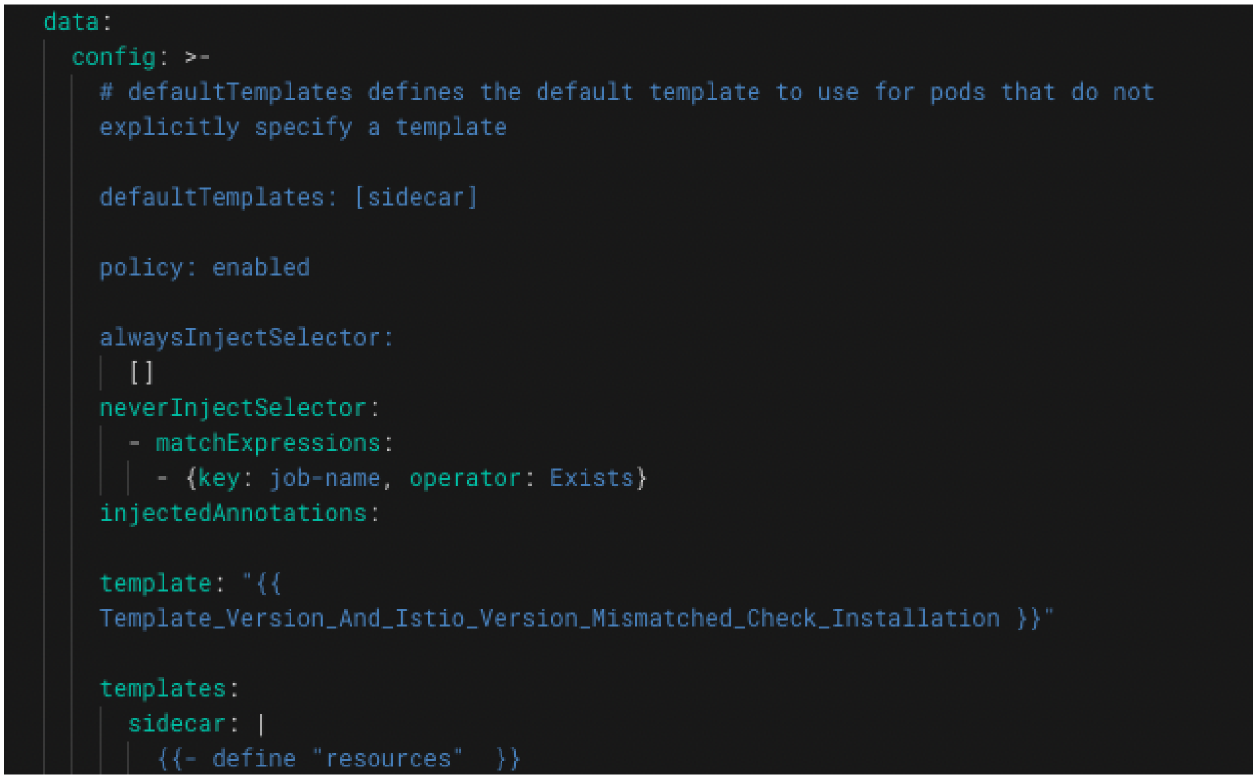
neverInjectSelector:
  - matchExpressions:
    - {key: job-name, operator: Exists}

- You can refer below image also for better understanding



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: istio-sidecar-injector
data:
  config: |-
    policy: enabled
    neverInjectSelector:
      - matchExpressions:
        - {key: job-name, operator: Exists}
```

- It will look like this



```
data:
  config: >-
    # defaultTemplates defines the default template to use for pods that do not
    explicitly specify a template

    defaultTemplates: [sidecar]

    policy: enabled

    alwaysInjectSelector:
      []
    neverInjectSelector:
      - matchExpressions:
        - {key: job-name, operator: Exists}
    injectedAnnotations:

    template: "{{
    Template_Version_And_Istio_Version_Mismatched_Check_Installation }}"

    templates:
      sidecar: |
        {{- define "resources"  }}
```