Search Medium

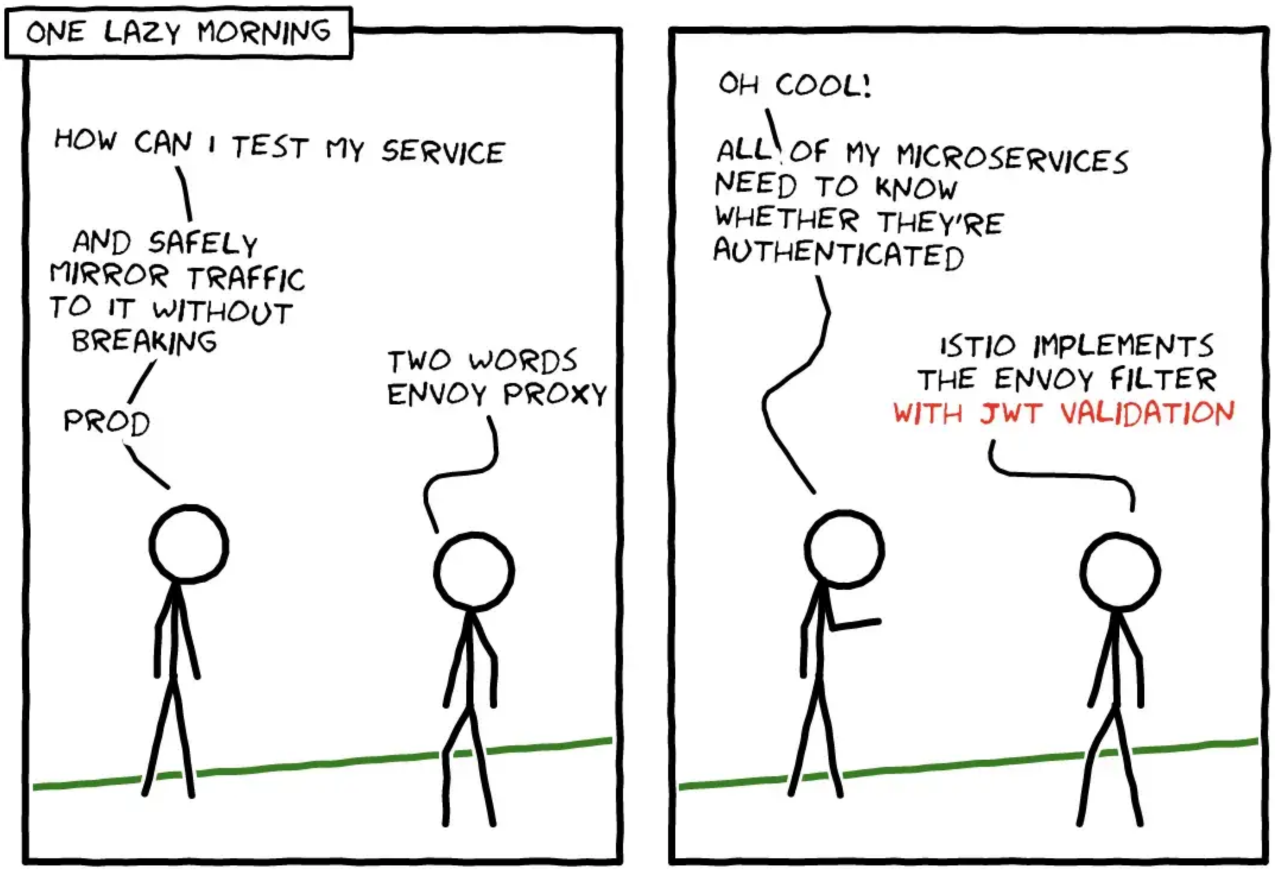
Published in ITNEXT

Shray Kumar Follow
Nov 2, 2020 · 5 min read · Listen

Save

Ten Tips for Running Istio in Production

If you've read all of the generic copy pasta BookInfo blog posts, here are ten tips about Istio that might assist you in your pursuit of a doctorate in Istio and Envoy Proxy.

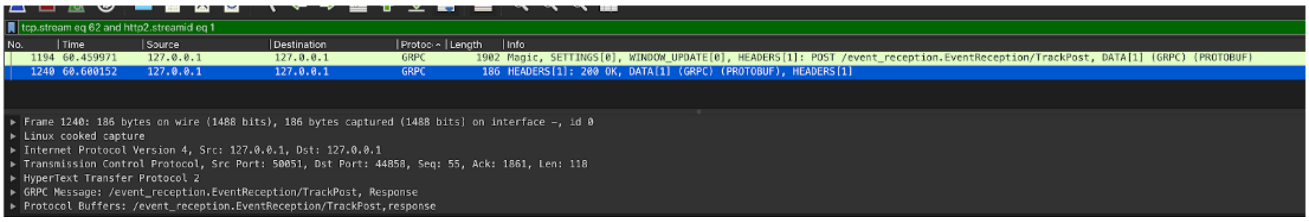

ONE LAZY MORNING
HOW CAN I TEST MY SERVICE
AND SAFELY MIRROR TRAFFIC TO IT WITHOUT BREAKING
PROD
TWO WORDS ENVOY PROXY
OH COOL!
ALL OF MY MICROSERVICES NEED TO KNOW WHETHER THEY'RE AUTHENTICATED
ISTIO IMPLEMENTS THE ENVOY FILTER WITH JWT VALIDATION

How we ended up picking Istio

1. Use Kubectl Sniff and Wireshark — If you haven't spent your summers net stumbling and wardriving, you may not have heard of Wireshark. Sniff provides an abstraction to allow you to listen to packets entering and exiting your Istio proxy via Wireshark. Sniff is excellent for low level analysis allows you to isolate root cause to application level versus Envoy level issues. Start by enabling privileged mode `global.proxy.privileged=true` on your IstioOperator CRD and add the following annotations to your deployment.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reception
spec:
  template:
    metadata:
      annotations:
        sidecar.istio.io/capNetBindService: "true"
```

In Wireshark, you can enable the capture of GRPC requests and filter based on them and "follow" a request's lifecycle to determine its ingress and egress. Port forward your service locally and use BloomRPC or Postman to isolate connectivity issues.



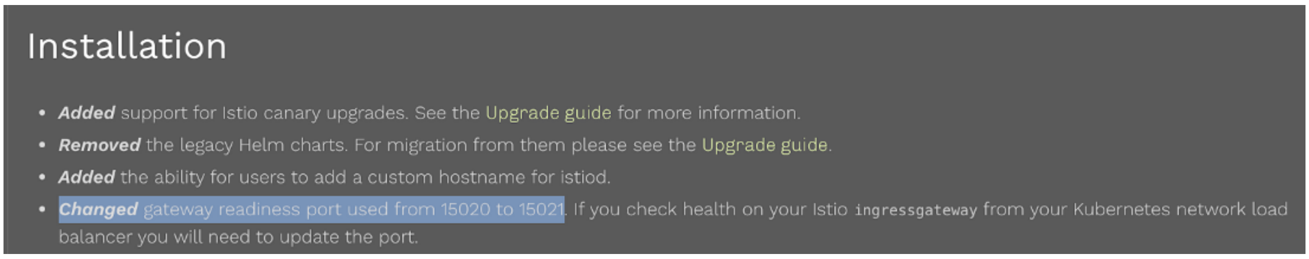
```
tcp.stream eq 62 and http2.streamid eq 1
No.   Time       Source     Destination  Protoc  Length Info
 1194 60.459971  127.0.0.1  127.0.0.1    GRPC      1902 Magic, SETTINGS[0], WINDOW_UPDATE[0], HEADERS[1]: POST /event_reception.EventReception/TrackPost, DATA[1] (GRPC) (PROTOBUF)
 1240 60.600152  127.0.0.1  127.0.0.1    GRPC       186 HEADERS[1]: 200 OK, DATA[1] (GRPC) (PROTOBUF), HEADERS[1]

▸ Frame 1240: 186 bytes on wire (1488 bits), 186 bytes captured (1488 bits) on interface -, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 50051, Dst Port: 44858, Seq: 55, Ack: 1861, Len: 118
▸ HyperText Transfer Protocol 2
▸ GRPC Message: /event_reception.EventReception/TrackPost, Response
▸ Protocol Buffers: /event_reception.EventReception/TrackPost,response
```

Filtered GRPC Requests in Wireshark

2. Use Envoy first. It's far easier to prototype changes such as request mirroring or WASM by running Envoy in a docker-compose file locally to understand how to configure Envoy filter chains. Building confidence in your understanding of Envoy will pay dividends in the process of converting it to Istio configuration.

3. Read the release notes religiously. This is worth reiterating as in the past we've missed key changes by glazing over release notes. During our previous Istio upgrade, we missed a seemingly benign change where the Envoy status port was modified (15020 -> 15021), causing us to believe our load balancer was unhealthy.


Installation

- *Added* support for Istio canary upgrades. See the Upgrade guide for more information.
- *Removed* the legacy Helm charts. For migration from them please see the Upgrade guide.
- *Added* the ability for users to add a custom hostname for istiod.
- *Changed* gateway readiness port used from 15020 to 15021. If you check health on your Istio `ingressgateway` from your Kubernetes network load balancer you will need to update the port.

Release Notes and Upgrade Guide on Istio's Site

4. Generate manifests using Istioctl over the Istio Operator — we recently did an upgrade from 1.6 -> 1.7 and noticed the operator was unable to complete its reconciliation loop due to a change in the struct (label -> labels in kind: EnvoyFilter). We expected an operator to be able to handle this upgrade in a backwards compatible manner. The generated manifests are easier to manipulate via the Istio Overlay API or Kustomize for fields that aren't exposed via the Operator CRD (e.g: Service annotations).

```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: wasm-hmac
spec:
  workloadSelector:
    labels: # labels, not label
      app: pubsub-gateway
```

5. When you find an example manifest, confirm the example aligns with the version of the Envoy API/Protobuf in your proxy container. The typed_config examples associated with the latest Envoy are sparsely documented for some key features.

```
patch:
     operation: INSERT_BEFORE
     value:
       name: envoy.filters.http.wasm
       typed_config:
         "@type": type.googleapis.com/udpa.type.v1.TypedStruct
         type_url:
type.googleapis.com/envoy.config.filter.http.wasm.v3.Wasm
         value:
           config:
             name: "signature_auth"
             root_id: "signature_auth"
```

```
              configuration:
                "@type":
"type.googleapis.com/google.protobuf.Struct"
                value:
                  signing_key: ""
                  signature_header: "X-My-Hmac-Header-SHA256"
              vm_config:
                runtime: "envoy.wasm.runtime.v8"
                code:
                  local:
                    filename: "/etc/wasm/your-binary.wasm"
                allow_precompiled: true
```

6. If you're using Istio Ingress Gateway as ingress to your cluster on GKE, and want to enable HTTPS, set up an Ingress object with a single upstream of Istio Ingress Gateway's service. Add a Static IP Compute Address, Managed Certificate, and BackendConfig (to set up health checks) so you provide secure access via your Load Balancer. Here's a tutorial on this topic.

7. Request Mirroring is a powerful means of "testing in production". If you've enabled this in your installation, and can't explain why downstream services are rejecting requests, look at your Host Header. Envoy appends "-shadow" to your host headers. You can work around this by creating a separate listener if you're using vanilla Envoy. Rewriting this header through Istio isn't exactly feasible in Istio, so instead we opted to add a separate Envoy deployment to rewrites headers to our intended destination.

In a similar vein, if you're implementing request mirroring at the edge using `istio-ingressgateway` and want in cluster mirroring, add the local service name and the keyword `mesh` to your virtualservice.

```
kind: VirtualService
metadata:
  name: mirror-policy
  namespace: your-app
spec:
  gateways:
    - istio-system/your-gateway
    - mesh
  hosts:
    - 'your-service.your-app.svc.cluster.local'
    - 'your-api.dot.com'
```

8. JWT Validation provides an interesting abstraction for allowing your service to know whether it's authenticated or not. It seems risky to have to "opt in" to authenticated services that may be publicly exposed, as opposed to opting out based on the workload selector label scheme. Enforce required labels on your workloads using tooling such as Open Policy Agent to ensure all services have an authentication label on them. In addition, when you enable "outputPayloadToHeader" on your JWTRule to propagate context, expect the payload to be missing the base64 padding when being consumed by your service.

9. GRPC JSON Transcoding is a convenient abstraction that saves engineers time in having to write helper code to transcode JSON to GRPC. However, the process around deploying changes to your proto descriptor seems a little clunky. Our initial workaround consisted of base64 encoding our proto descriptor and mounting it as a secret. This quickly grew unwieldy as no engineer wants to base64 encode a binary whenever your Protobuf definition changes. A slightly more elegant approach is to build a container image with your descriptor and run it as an initContainer with a shared memory volume. Upon pod startup, your initContainer will copy the binary into the shared volume space and will be accessible to your Istio proxy container.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reception
spec:
  template:
    metadata:
      annotations: # only good for prototyping!
        sidecar.istio.io/userVolumeMount: '[{"name":"my-cert",
"mountPath":"/etc/envoy", "readonly":true}, {"name": "cache-
volume", "mountPath":"/tmp"}]'
        sidecar.istio.io/userVolume: '[{"name":"my-cert",
"secret":{"secretName":"event-reception-pb"}}, {"name": "cache-
volume", "emptyDir":{}}]'
```

10. Envoy offers a way of modifying virtual host specific configurations for filters. In Istio terms, this translates to a way of being able to change the functionality of an existing filter on a specific virtual host path. This feature works with specific filters that mention that they support a "Per Route Configuration", but doesn't work with filters such as WASM.

Bonus: if you have to add a label, it's probably the pod template spec label, not the deployment label. Use istioctl analyze to provide you with fixes for low hanging fruit.

Istio Kubernetes Gke Microservices

422 1

About Help Terms Privacy

Get the Medium app

Download on the App Store GET IT ON Google Play