Running Istio at Scale for a Secure and Compliant Cloud

Lucas Copi
Rafael Polanco

IstioCon

#IstioCon

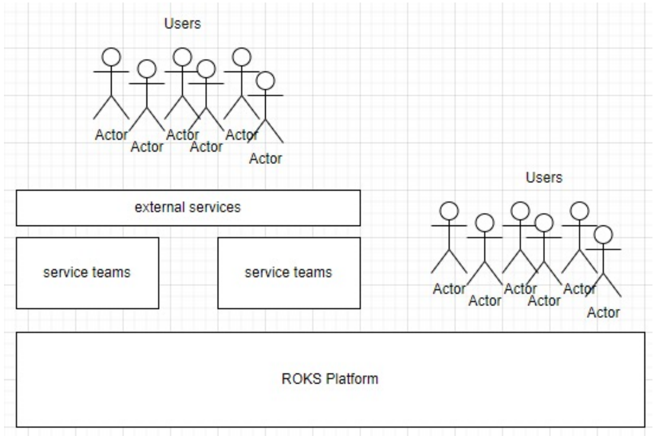
Introduction

Lucas Copi
Technical and Development Lead for Ingress on IBM Cloud

Rafael Polanco
Software Engineer on IBM Cloud Kubernetes Service

So… IBM Cloud, what's that?

- Offer a lot of different services[1], but the core is a Kubernetes and Openshift offering



- General Scale Numbers
  - Average 250rps per serving cluster – 10 Geos worldwide
  - High peak loads and very high burst rates
  - Mix of small and large volume payloads
- Translates to ~150-200 cluster create events per day

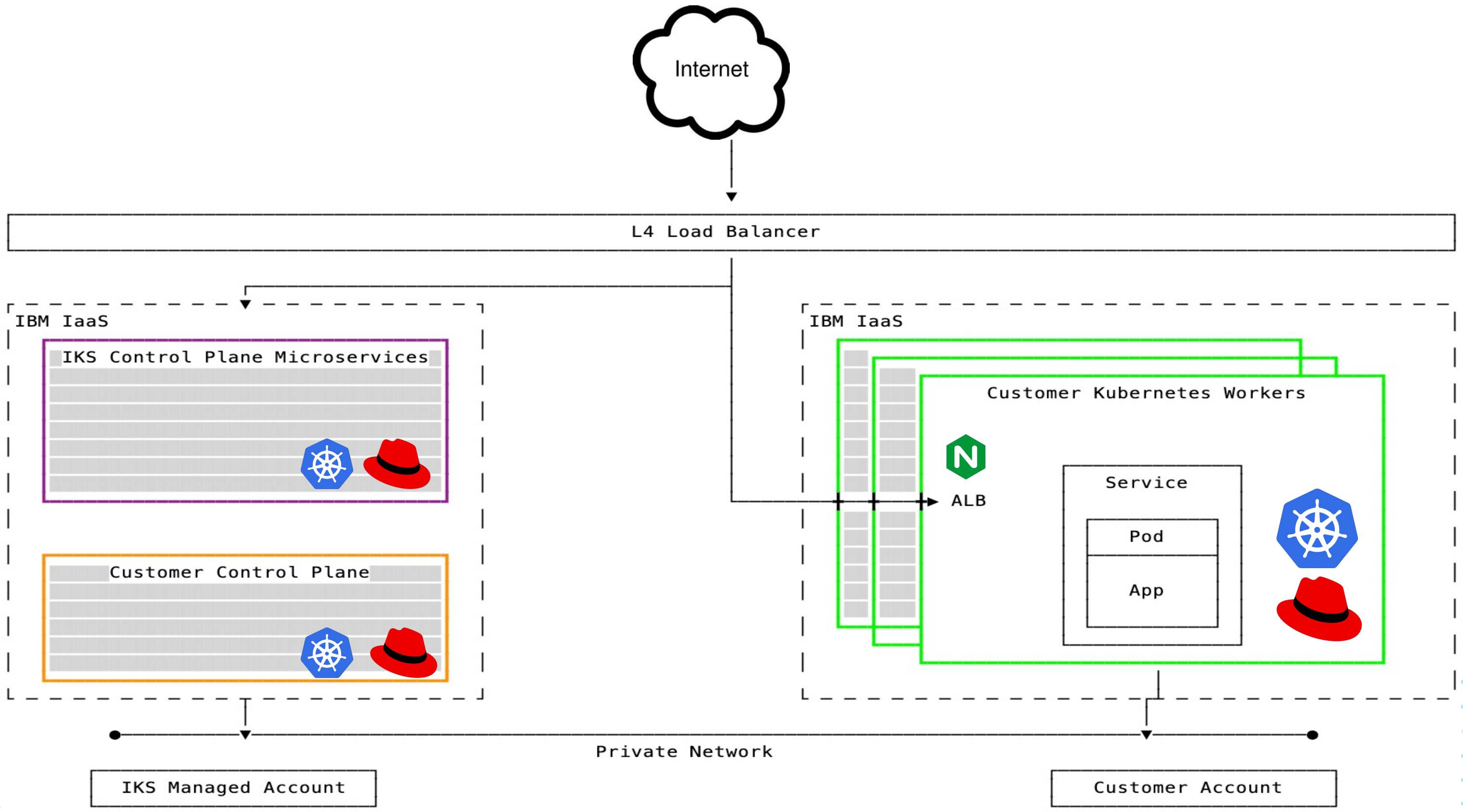
[1] https://www.ibm.com/cloud/products

And Istio too

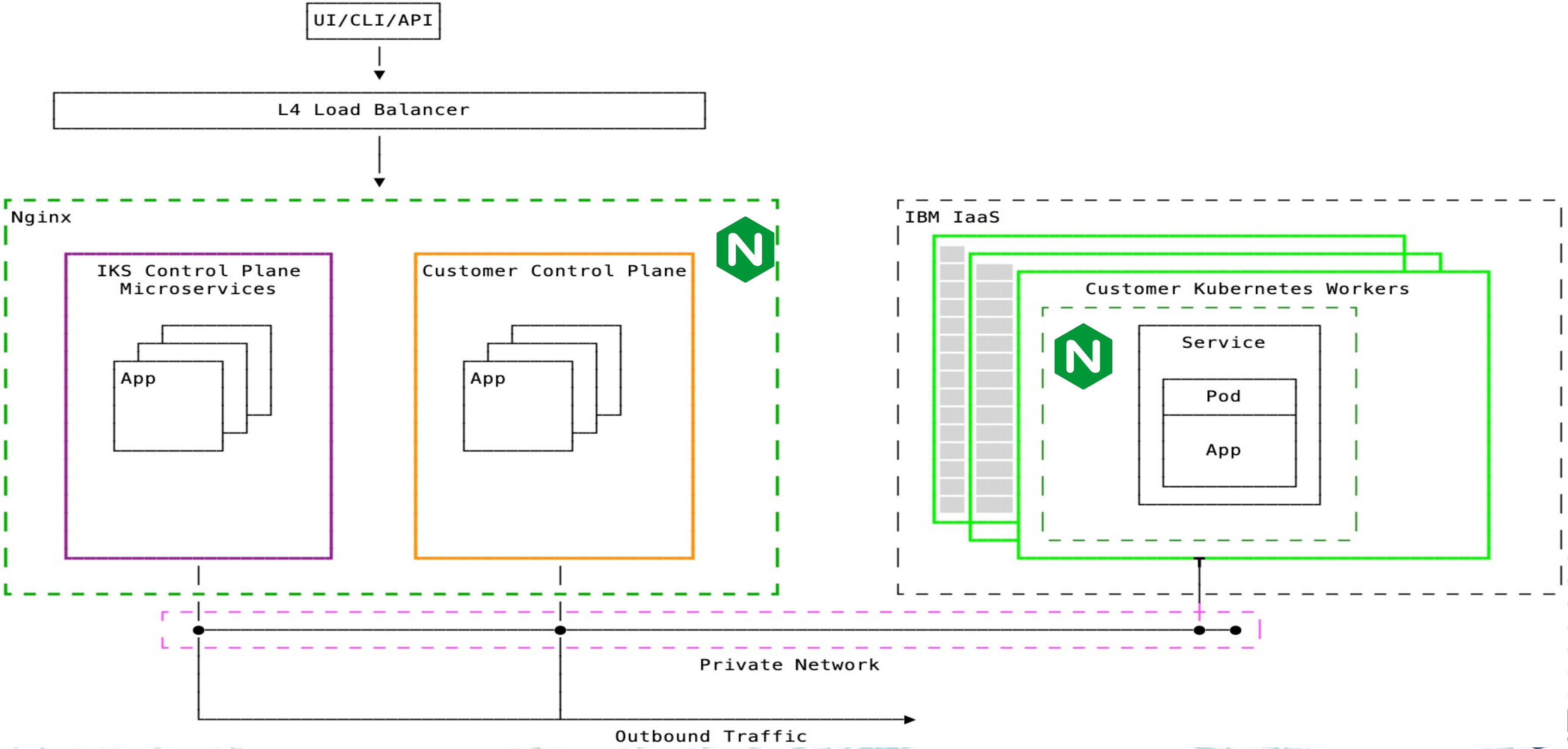
- Why do we need Istio? Great question
- Started focusing on financial services enablement – 18mo ago[1]
  - 500+ security and compliance controls
  - Based on NIST 800-53
  - Meets regulatory standards from over 75 institutions and 24 different countries
- Achieve highest level of security compliance from CISO
- Enhance and exceed our Service Level Objectives (SLOs)

[1] https://www.ibm.com/downloads/cas/JYB6MQRB

IBM Cloud Kubernetes Service (IKS) System Architecture



Managed Ingress Architecture (Prior to Istio)



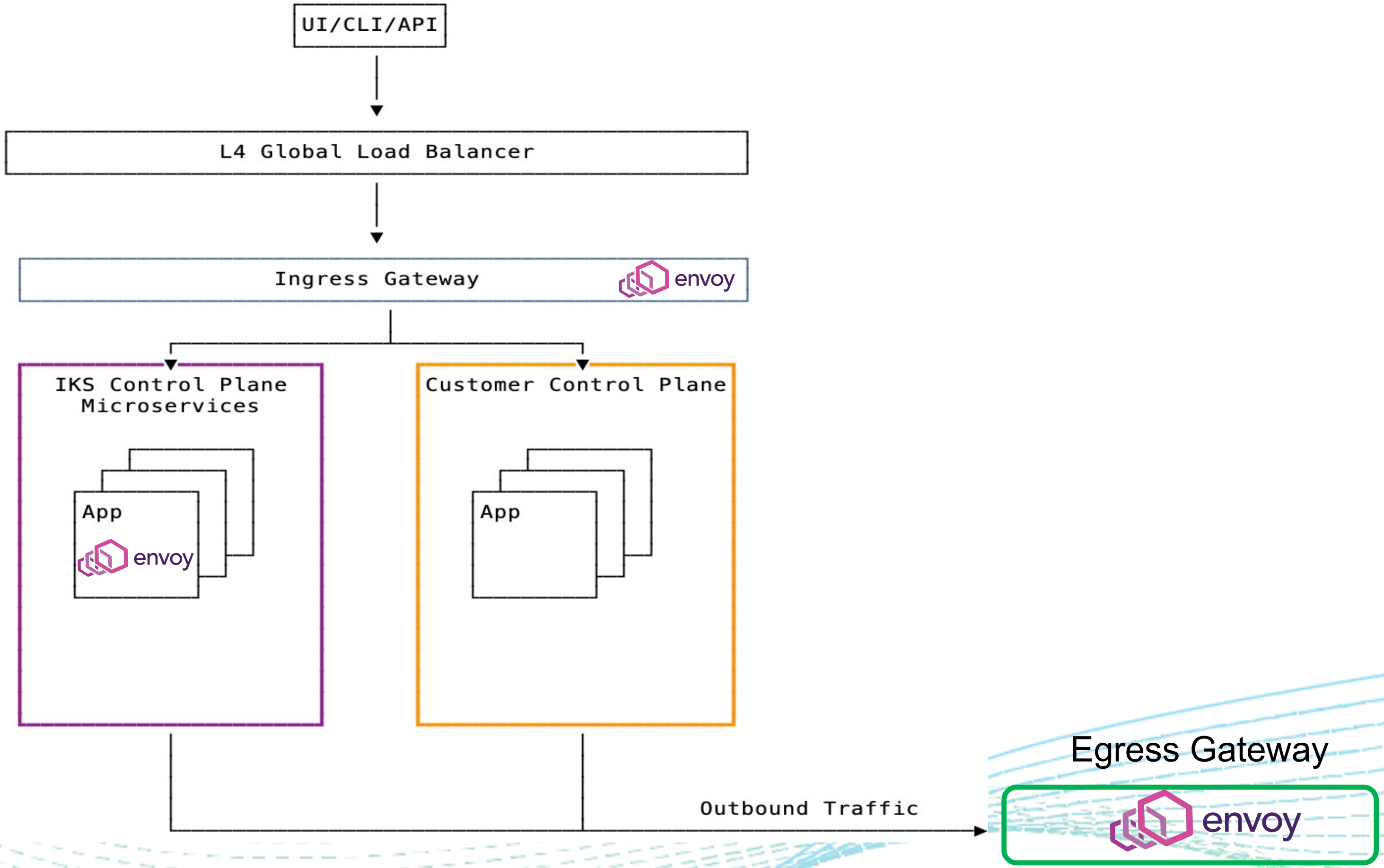
How Istio Helps Us

- Fine Grained Traffic Controls and Policy Enforcement
  - Helps us enforce security and regulatory controls across our service and development teams
  - E.g. Ingress/Egress network policies, strict mTLS
- Security and Authentication
  - Automatic TLS and strict mTLS connections out of the box
  - Secure Control of Egress Traffic – all outbound traffic must be known and documented for compliance requirements
- Observability and Resiliency
  - Out of the box network retries, failover and circuit breaking
  - Detailed telemetry instrumentation helps us better understand how our services are being used which will enable us to better meet our SLOs and make improvements

Initial Rollout

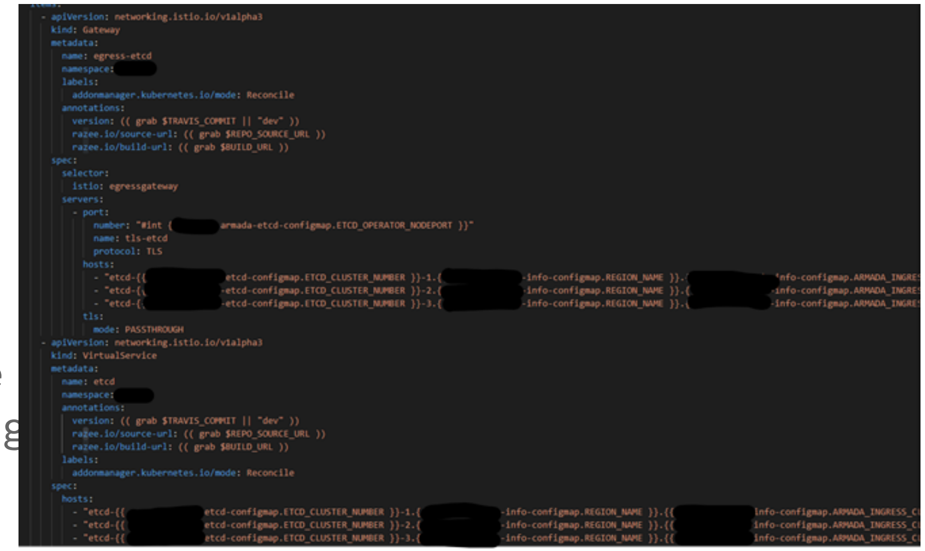
- Started experimenting beginning with Istio 1.4
- Tolerated both mTLS and plaintext traffic
- Went live to production with Istio 1.8.x
- Manual Canary deployments

Managed Ingress Architecture (after Istio adoption)



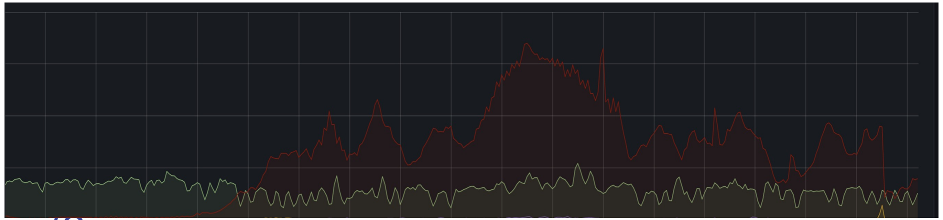
Current Istio Pipeline

- Use opensource tool Razee https://razee.io/
- Template out deployment files
- Deployment pipeline merges templated files with configuration from the environment to deploy the full set of istio resources from a sing set of yamls
- Combination of Jenkins for operational procedures and razee for auto-deployments
- Can roll out globally in less than half a day vs a week

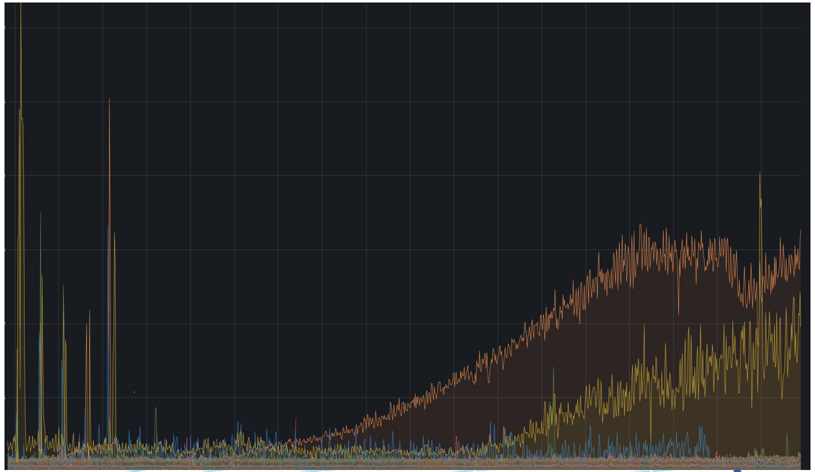


```
  - apiVersion: networking.istio.io/v1alpha3
    kind: Gateway
    metadata:
      name: egress-etcd
      namespace:[redacted]
      labels:
        addonmanager.kubernetes.io/mode: Reconcile
      annotations:
        version: (( grab $TRAVIS_COMMIT || "dev" ))
        razee.io/source-url: (( grab $REPO_SOURCE_URL ))
        razee.io/build-url: (( grab $BUILD_URL ))
    spec:
      selector:
        istio: egressgateway
      servers:
        - port:
            number: "#int {[redacted]armada-etcd-configmap.ETCD_OPERATOR_NODEPORT }}"
            name: tls-etcd
            protocol: TLS
          hosts:
            - "etcd-{{[redacted]etcd-configmap.ETCD_CLUSTER_NUMBER }}-1.{[redacted]-info-configmap.REGION_NAME }}.[redacted]info-configmap.ARMADA_INGRE
            - "etcd-{[redacted]-etcd-configmap.ETCD_CLUSTER_NUMBER }}-2.{[redacted]-info-configmap.REGION_NAME }}.{[redacted]-info-configmap.ARMADA_INGRE
            - "etcd-{[redacted]-etcd-configmap.ETCD_CLUSTER_NUMBER }}-3.{[redacted]-info-configmap.REGION_NAME }}.[redacted]-info-configmap.ARMADA_INGRE
          tls:
            mode: PASSTHROUGH
  - apiVersion: networking.istio.io/v1alpha3
    kind: VirtualService
    metadata:
      name: etcd
      namespace:[redacted]
      annotations:
        version: (( grab $TRAVIS_COMMIT || "dev" ))
        razee.io/source-url: (( grab $REPO_SOURCE_URL ))
        razee.io/build-url: (( grab $BUILD_URL ))
      labels:
        addonmanager.kubernetes.io/mode: Reconcile
    spec:
      hosts:
        - "etcd-{{[redacted]etcd-configmap.ETCD_CLUSTER_NUMBER }}-1.{[redacted]-info-configmap.REGION_NAME }}.{{[redacted]info-configmap.ARMADA_INGRESS_C
        - "etcd-{{[redacted]etcd-configmap.ETCD_CLUSTER_NUMBER }}-2.{[redacted]-info-configmap.REGION_NAME }}.{{[redacted]info-configmap.ARMADA_INGRESS_C
        - "etcd-{{[redacted]etcd-configmap.ETCD_CLUSTER_NUMBER }}-3.{[redacted]-info-configmap.REGION_NAME }}.{{[redacted]info-configmap.ARMADA_INGRESS_C
```

“I either win or I learn” – Nelson Mandela …Lessons Learned

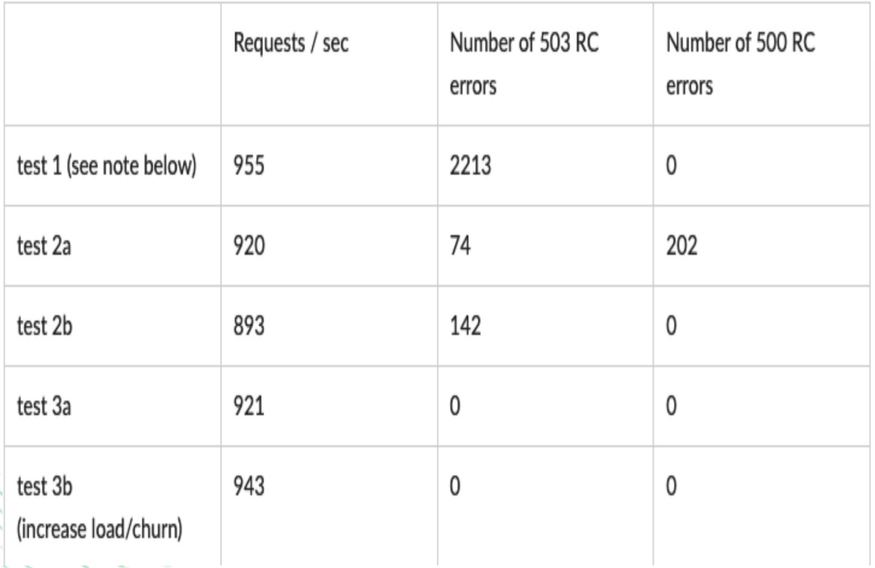


- Scale is hard
- Mestastable failures https://sigops.org/s/conferences/hotos/2021/papers/hotos21-s11-bronson.pdf
  - Distributed system outages that occur when there are no hardware failures, configuration errors, or software bugs
  - Increase load causes a trigger event but the failures persist even after the trigger is removed
- High burst adds latency combined with increased errors, errors cascade, system goes down



Summer School

- Thundering herd issue with restarts



	Requests / sec	Number of 503 RC errors	Number of 500 RC errors
test 1 (see note below)	955	2213	0
test 2a	920	74	202
test 2b	893	142	0
test 3a	921	0	0
test 3b (increase load/churn)	943	0	0

#IstioCon

HA Microservice Guidelines:

Pod Priority and Preemption
- Most important pods get scheduling priority

Configure Liveness, Readiness and Startup Probes
- initialDelaySeconds, periodSeconds, timeoutSeconds and failureThreshold are configured

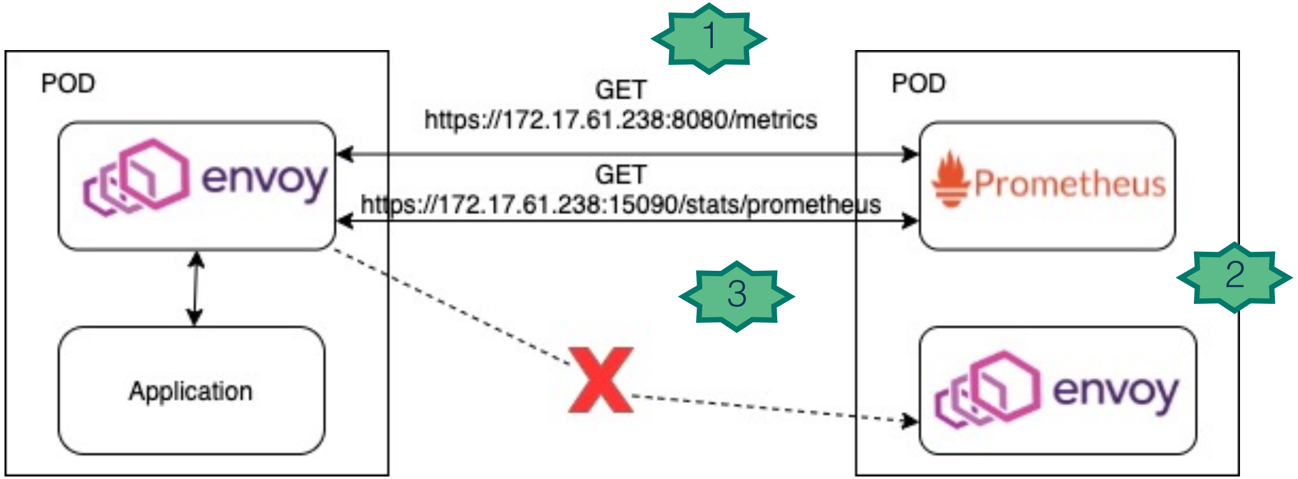
Managing Resources for Containers
- Realistic compute constraints built from existing workload. kubectl top pods -A --containers=true

Deployment Rollout Strategy
- use a RollingUpdate rollout strategy with maxUnavailable set to 1 and maxSurge set to 0.

Unexpected Outcomes

- Egress traffic blows away ingress traffic 10x
  - Dedicated egress gateways for high volume traffic
  - Spread sni proxies across zones w/ dedicated nodes
- Hyper latency sensitive operations to DB bypass istio (additional milliseconds of latency due to strict mtls and request hijacking) was too much for the system

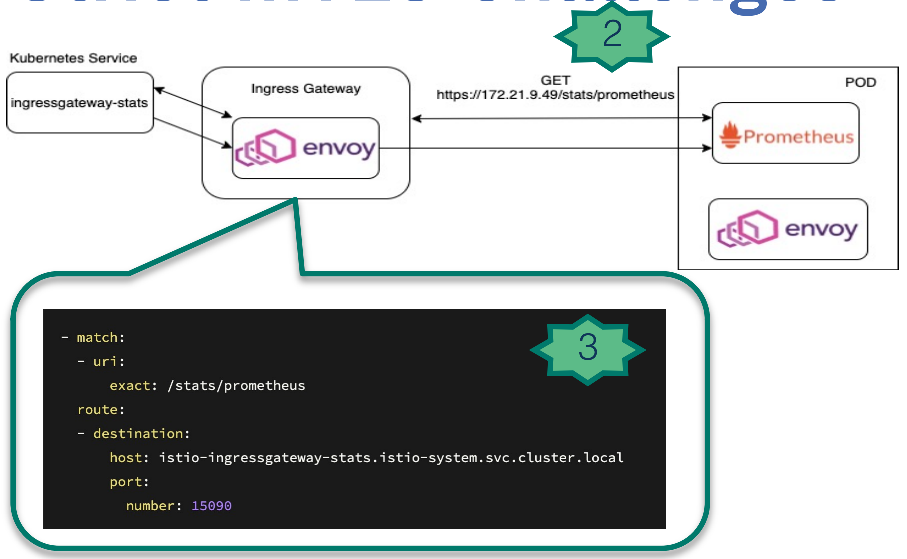
Strict mTLS Challenges – Prometheus



1
POD
GET https://172.17.61.238:8080/metrics
GET https://172.17.61.238:15090/stats/prometheus
POD
envoy
Prometheus
2
3
Application
envoy

1. Prometheus will scrape both Istio and application metric endpoints via new jobs
2. Prometheus will leverage Istio certificates generated by Envoy container
3. The sidecar (Envoy) should NOT intercept traffic for Prometheus since it needs direct endpoint access

Strict mTLS Challenges– Prometheus (Cont.)



```yaml
- match:
  - uri:
      exact: /stats/prometheus
  route:
  - destination:
      host: istio-ingressgateway-stats.istio-system.svc.cluster.local
      port:
        number: 15090
```

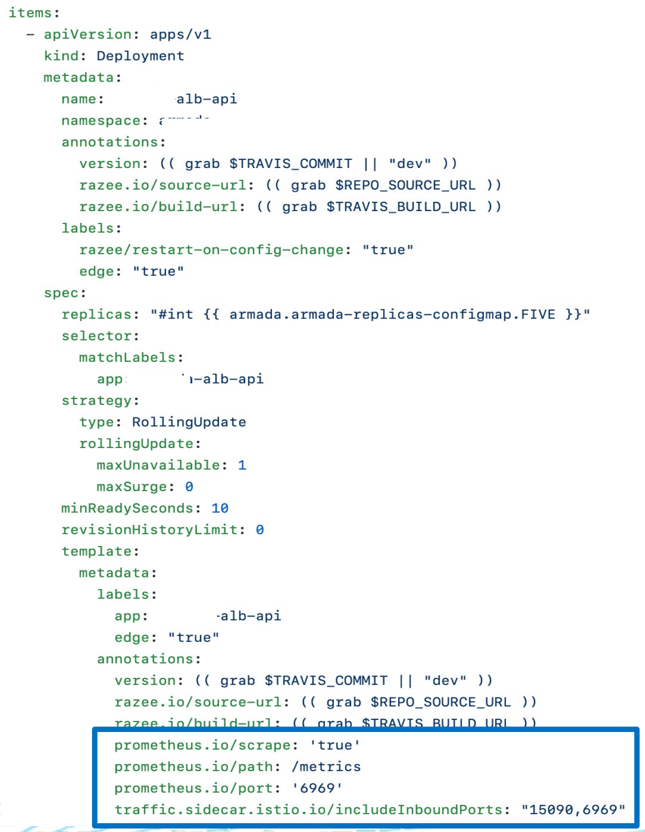
1

```yaml
# Scrape Istio gateway stats
- job_name: 'istio-ingressgateway-service'
  metrics_path: /stats/prometheus
  scheme: https
  tls_config:
    ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    insecure_skip_verify: true
  kubernetes_sd_configs:
  - role: service
  relabel_configs:
  - source_labels: [__meta_kubernetes_service_port_name]
    action: keep
    regex: 'https'
  - source_labels: [__meta_kubernetes_service_name]
    action: keep
    regex: 'istio-ingressgateway-dal.*'
  - source_labels: [__meta_kubernetes_namespace]
    action: replace
    target_label: kubernetes_namespace
  - source_labels: [__meta_kubernetes_service_name]
    action: replace
    target_label: service_name
  - source_labels: [__meta_kubernetes_service_type]
    action: replace
    target_label: service_type
  - action: labelmap
    regex: __meta_kubernetes_service_label_(.+)
```

1. Prometheus job to scrape ingress gateway services
2. Prometheus will scrape ingress gateway metric endpoint via the Istio Gateway
3. The Istio gateway will route the request to the ingress gateway pods

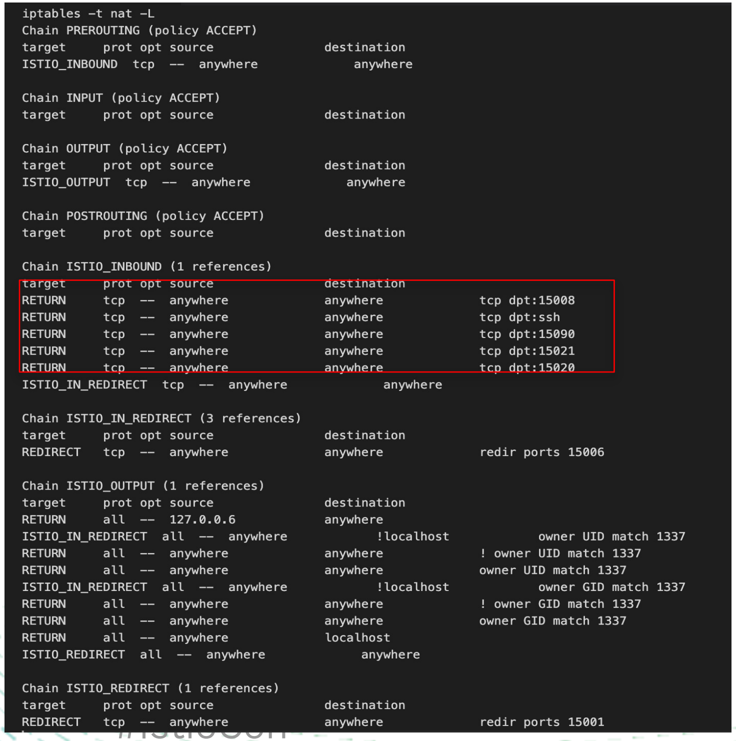
Strict mTLS Challenges – Istio Sidecar

- Incoming telemetry traffic to the application needs to be redirected to Envoy in order for Prometheus to successfully scrape the metric endpoints via mTLS
- Failing to do so will result in the application throwing errors such as: “*read: connection reset by peer*” and “*http: server gave HTTP response to HTTPS client*”
- The Istio sidecar resource annotation *traffic.sidecar.istio.io/includeInboundPorts* is used and required by all of our control plane microservices to enable inbound ports to redirect traffic to Envoy and allow Prometheus to scrape the metrics endpoint successfully


```
items:
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: [redacted]alb-api
    namespace: [redacted]
    annotations:
      version: (( grab $TRAVIS_COMMIT || "dev" ))
      razee.io/source-url: (( grab $REPO_SOURCE_URL ))
      razee.io/build-url: (( grab $TRAVIS_BUILD_URL ))
    labels:
      razee/restart-on-config-change: "true"
      edge: "true"
  spec:
    replicas: "#int {{ armada.armada-replicas-configmap.FIVE }}"
    selector:
      matchLabels:
        app: [redacted]-alb-api
    strategy:
      type: RollingUpdate
      rollingUpdate:
        maxUnavailable: 1
        maxSurge: 0
    minReadySeconds: 10
    revisionHistoryLimit: 0
    template:
      metadata:
        labels:
          app: [redacted]-alb-api
          edge: "true"
        annotations:
          version: (( grab $TRAVIS_COMMIT || "dev" ))
          razee.io/source-url: (( grab $REPO_SOURCE_URL ))
          razee.io/build-url: (( grab $TRAVIS_BUILD_URL ))
          prometheus.io/scrape: 'true'
          prometheus.io/path: /metrics
          prometheus.io/port: '6969'
          traffic.sidecar.istio.io/includeInboundPorts: "15090,6969"
```

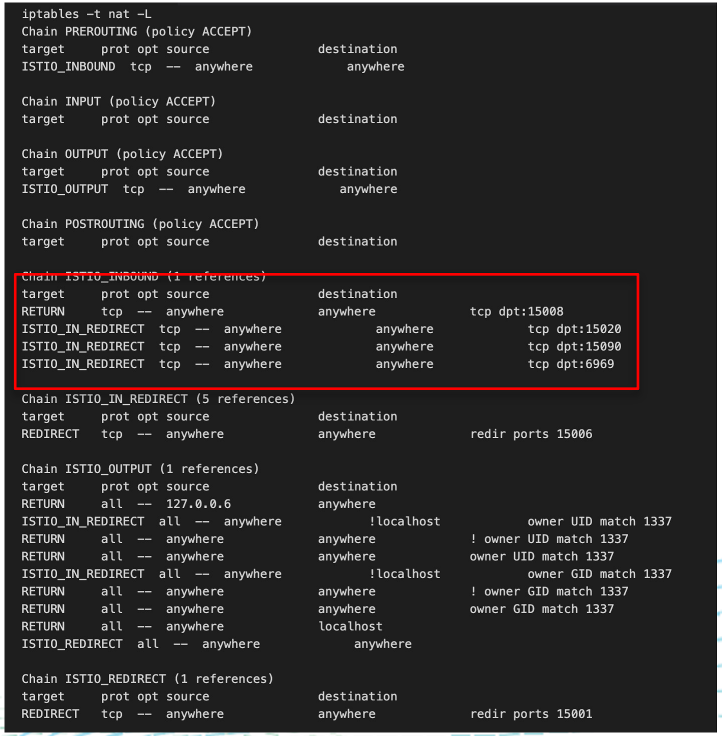
Getting it (mTLS) to Work – Istio Sidecar (Cont.)

iptable rules for pod w/o `traffic.sidecar.istio.io/includeInboundPorts` annotation



```
iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
ISTIO_INBOUND  tcp  --  anywhere             anywhere

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ISTIO_OUTPUT  tcp  --  anywhere             anywhere

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain ISTIO_INBOUND (1 references)
target     prot opt source               destination
RETURN     tcp  --  anywhere             anywhere             tcp dpt:15008
RETURN     tcp  --  anywhere             anywhere             tcp dpt:ssh
RETURN     tcp  --  anywhere             anywhere             tcp dpt:15090
RETURN     tcp  --  anywhere             anywhere             tcp dpt:15021
RETURN     tcp  --  anywhere             anywhere             tcp dpt:15020
ISTIO_IN_REDIRECT  tcp  --  anywhere             anywhere

Chain ISTIO_IN_REDIRECT (3 references)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere             redir ports 15006

Chain ISTIO_OUTPUT (1 references)
target     prot opt source               destination
RETURN     all  --  127.0.0.6            anywhere
ISTIO_IN_REDIRECT  all  --  anywhere            !localhost            owner UID match 1337
RETURN     all  --  anywhere             anywhere             ! owner UID match 1337
RETURN     all  --  anywhere             anywhere             owner UID match 1337
ISTIO_IN_REDIRECT  all  --  anywhere            !localhost            owner GID match 1337
RETURN     all  --  anywhere             anywhere             ! owner GID match 1337
RETURN     all  --  anywhere             anywhere             owner GID match 1337
RETURN     all  --  anywhere             localhost
ISTIO_REDIRECT  all  --  anywhere             anywhere

Chain ISTIO_REDIRECT (1 references)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere             redir ports 15001
```

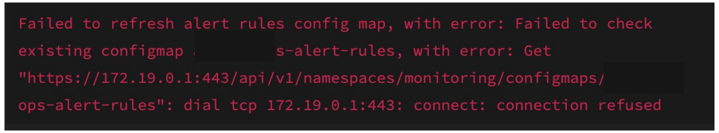
iptables rules for pod **with** `traffic.sidecar.istio.io/includeInboundPorts="15090, 6969"`



```
iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
ISTIO_INBOUND  tcp  --  anywhere             anywhere

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ISTIO_OUTPUT  tcp  --  anywhere             anywhere

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain ISTIO_INBOUND (1 references)
target     prot opt source               destination
RETURN     tcp  --  anywhere             anywhere             tcp dpt:15008
ISTIO_IN_REDIRECT  tcp  --  anywhere             anywhere             tcp dpt:15020
ISTIO_IN_REDIRECT  tcp  --  anywhere             anywhere             tcp dpt:15090
ISTIO_IN_REDIRECT  tcp  --  anywhere             anywhere             tcp dpt:6969

Chain ISTIO_IN_REDIRECT (5 references)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere             redir ports 15006

Chain ISTIO_OUTPUT (1 references)
target     prot opt source               destination
RETURN     all  --  127.0.0.6            anywhere
ISTIO_IN_REDIRECT  all  --  anywhere            !localhost            owner UID match 1337
RETURN     all  --  anywhere             anywhere             ! owner UID match 1337
RETURN     all  --  anywhere             anywhere             owner UID match 1337
ISTIO_IN_REDIRECT  all  --  anywhere            !localhost            owner GID match 1337
RETURN     all  --  anywhere             anywhere             ! owner GID match 1337
RETURN     all  --  anywhere             anywhere             owner GID match 1337
RETURN     all  --  anywhere             localhost
ISTIO_REDIRECT  all  --  anywhere             anywhere

Chain ISTIO_REDIRECT (1 references)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere             redir ports 15001
```

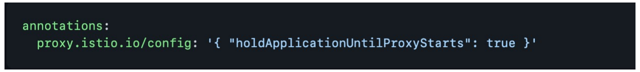
Challenges – Sidecar and Jobs Don't Play Well

- Our Kubernetes jobs were running into a race condition with the Envoy sidecar



```
Failed to refresh alert rules config map, with error: Failed to check
existing configmap [redacted]s-alert-rules, with error: Get
"https://172.19.0.1:443/api/v1/namespaces/monitoring/configmaps/[redacted]
ops-alert-rules": dial tcp 172.19.0.1:443: connect: connection refused
```

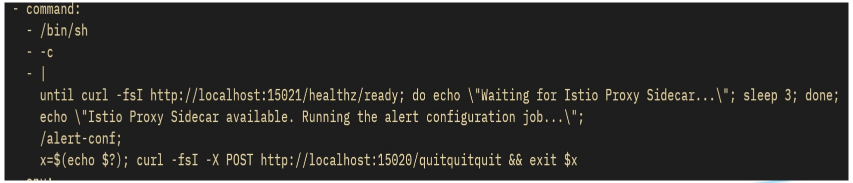
holdApplicationUntilProxyStarts is a hook which delays application startup until the proxy pod is ready to accept traffic



```
annotations:
  proxy.istio.io/config: '{ "holdApplicationUntilProxyStarts": true }'
```

- These jobs will keep running as long as the sidecar is running

Documented Hack: Override the job container's entrypoint with the following:



```
- command:
  - /bin/sh
  - -c
  - |
    until curl -fsI http://localhost:15021/healthz/ready; do echo \"Waiting for Istio Proxy Sidecar...\"; sleep 3; done;
    echo \"Istio Proxy Sidecar available. Running the alert configuration job...\";
    /alert-conf;
    x=$(echo $?); curl -fsI -X POST http://localhost:15020/quitquitquit && exit $x
```

#IstioCon

What’s Next

- Leverage distributed tracing
- Implement Rate Limiting
- A/B Testing for API Gateway
- Move away from the Istio operator to a Razee managed deployment
- Optimize for performance

Thank you!

Lucas Copi 👉 https://www.linkedin.com/in/lucas-copi/

Rafael Polanco 👉 www.linkedin.com/in/rafaelpolanco

IBM **Cloud** 👉 https://cloud.ibm.com/docs/containers?topic=containers-istio-about