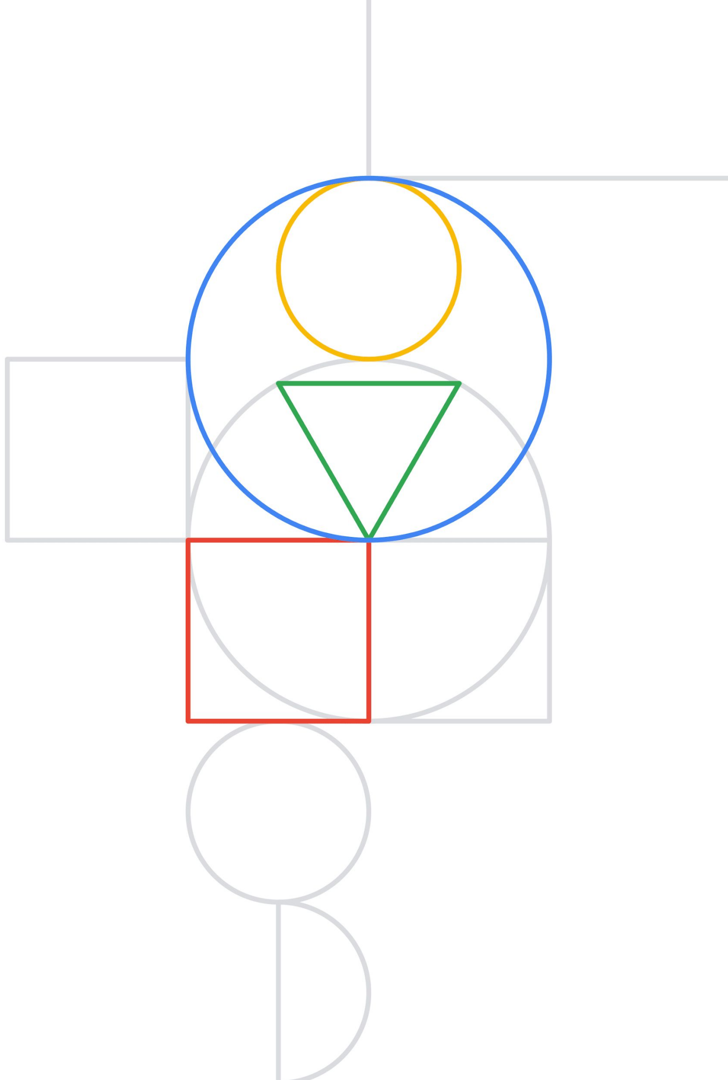
Secure your apps and your clusters with Anthos Service Mesh

Mathieu Benoit

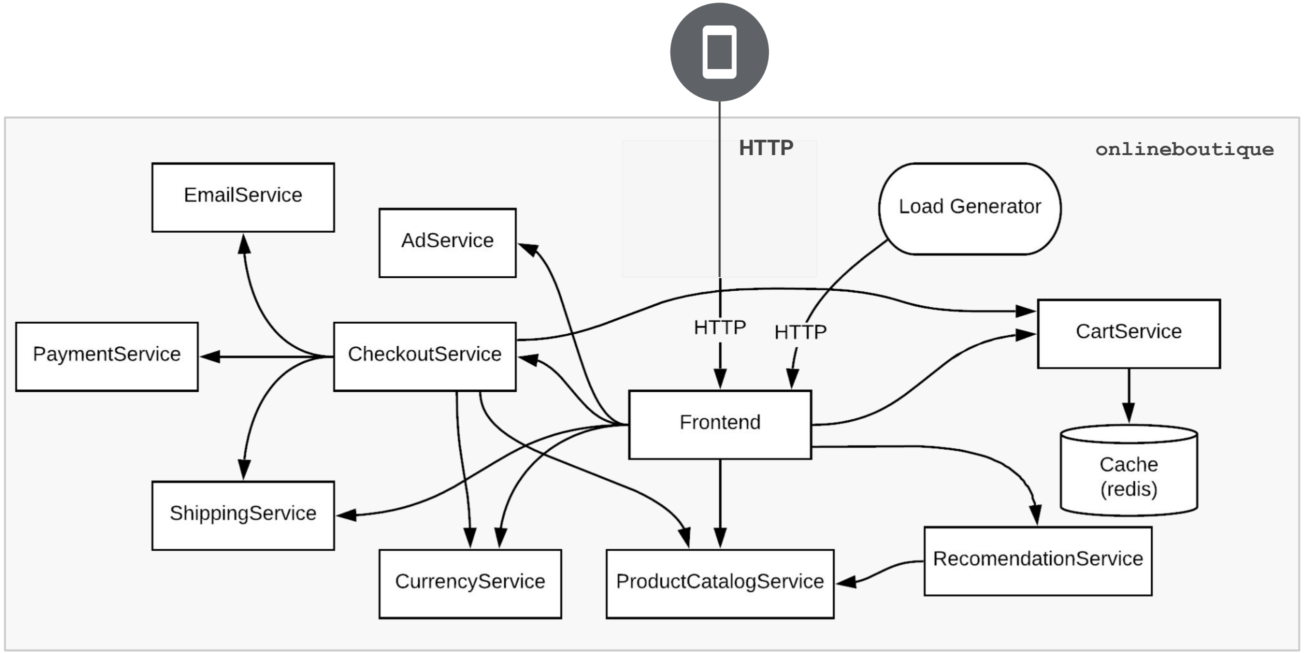
2022-01-14



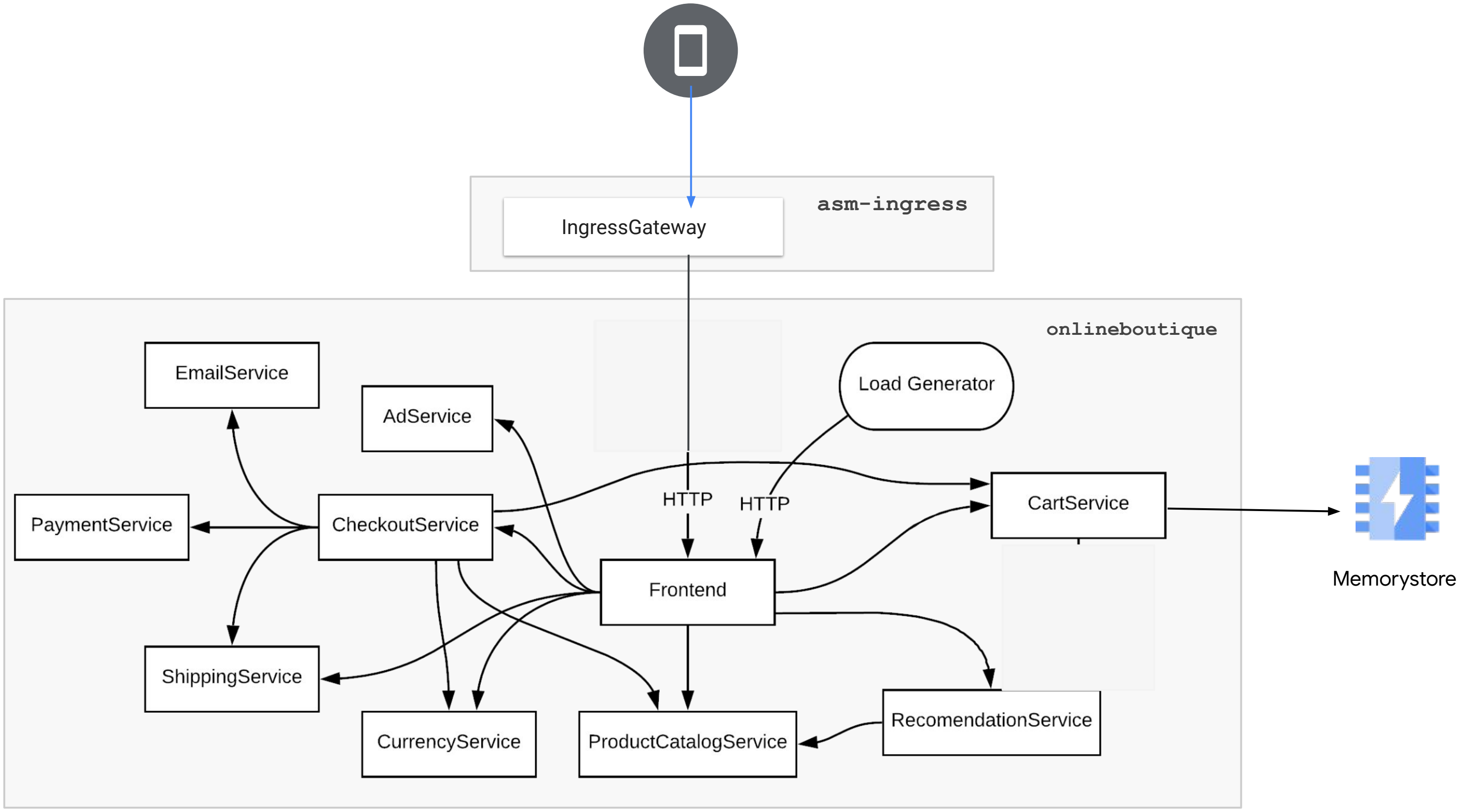
Features covered

1. Install a secure Managed ASM
2. Enable ASM
3. Enable mTLS STRICT
4. Configure a `Sidecar`
5. Define `AuthorizationPolicy`
6. Protect your Ingress `Gateway` with HTTPS GCLB and Cloud Armor
7. Wrap up: ASM in the GCP console

Before

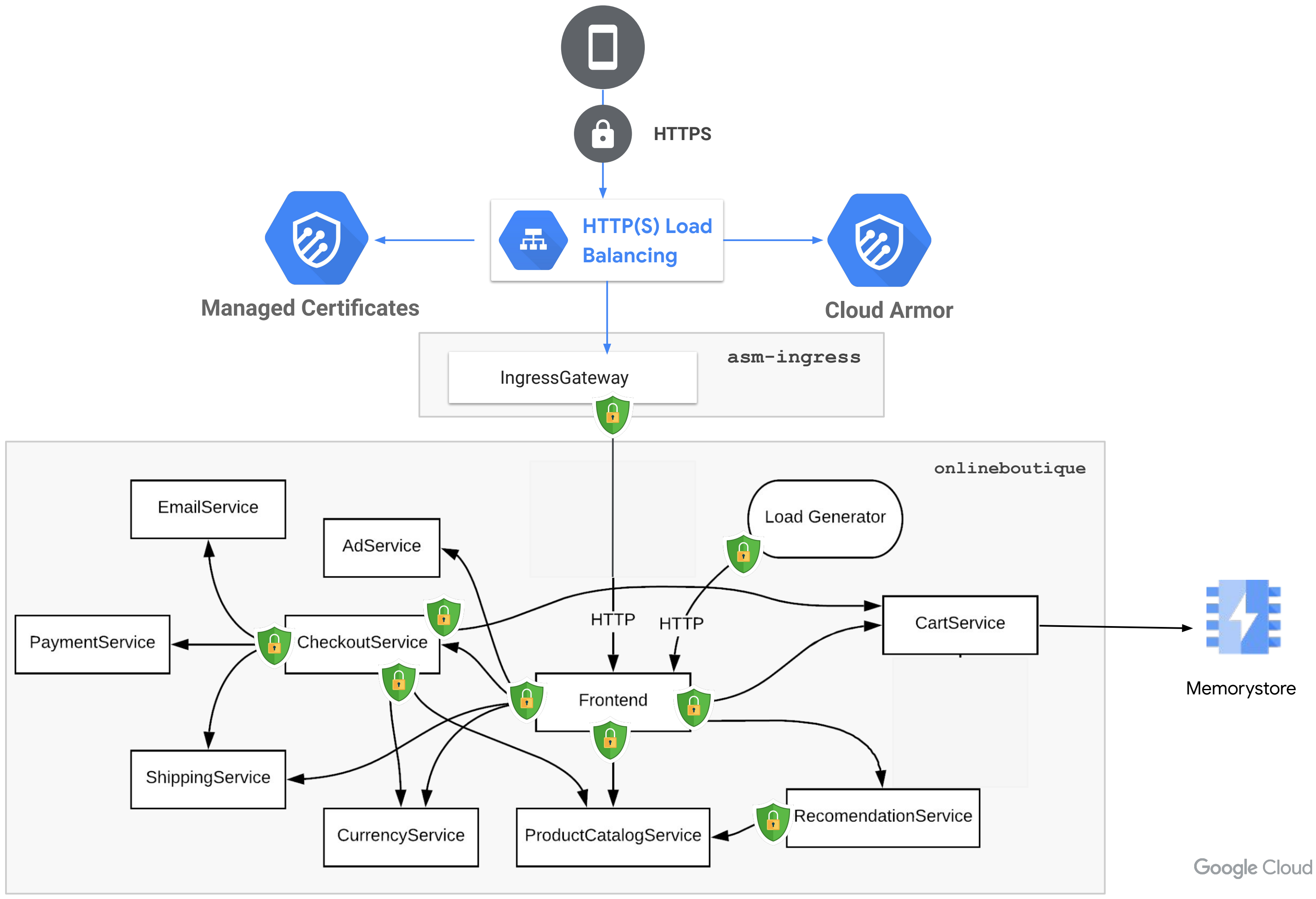

HTTP
onlineboutique
EmailService
AdService
Load Generator
PaymentService
CheckoutService
HTTP
HTTP
CartService
Frontend
Cache (redis)
ShippingService
CurrencyService
ProductCatalogService
RecomendationService

Half-way



asm-ingress
IngressGateway
onlineboutique
EmailService
AdService
Load Generator
PaymentService
CheckoutService
HTTP
HTTP
CartService
Memorystore
Frontend
ShippingService
CurrencyService
ProductCatalogService
RecomendationService

After


HTTPS
HTTP(S) Load Balancing
Managed Certificates
Cloud Armor
asm-ingress
IngressGateway
onlineboutique
EmailService
AdService
Load Generator
PaymentService
CheckoutService
HTTP
HTTP
CartService
Memorystore
Frontend
ShippingService
CurrencyService
ProductCatalogService
RecomendationService

Install Managed ASM on the cluster

TERMINAL

```
$ curl https://storage.googleapis.com/csm-artifacts/asm/asmcli_1.12 > ~/asmcli
$ chmod +x ~/asmcli

$ ASM_CHANNEL=rapid
$ ASM_LABEL=asm-managed
$ ASM_VERSION=$ASM_LABEL-$ASM_CHANNEL

$ gcloud container hub mesh enable

$ ~/asmcli install \
  --project_id $PROJECT_ID \
  --cluster_name $GKE_NAME \
  --cluster_location $ZONE \
  --enable-all \
  --managed \
  --channel $ASM_CHANNEL \
  --use_managed_cni
```

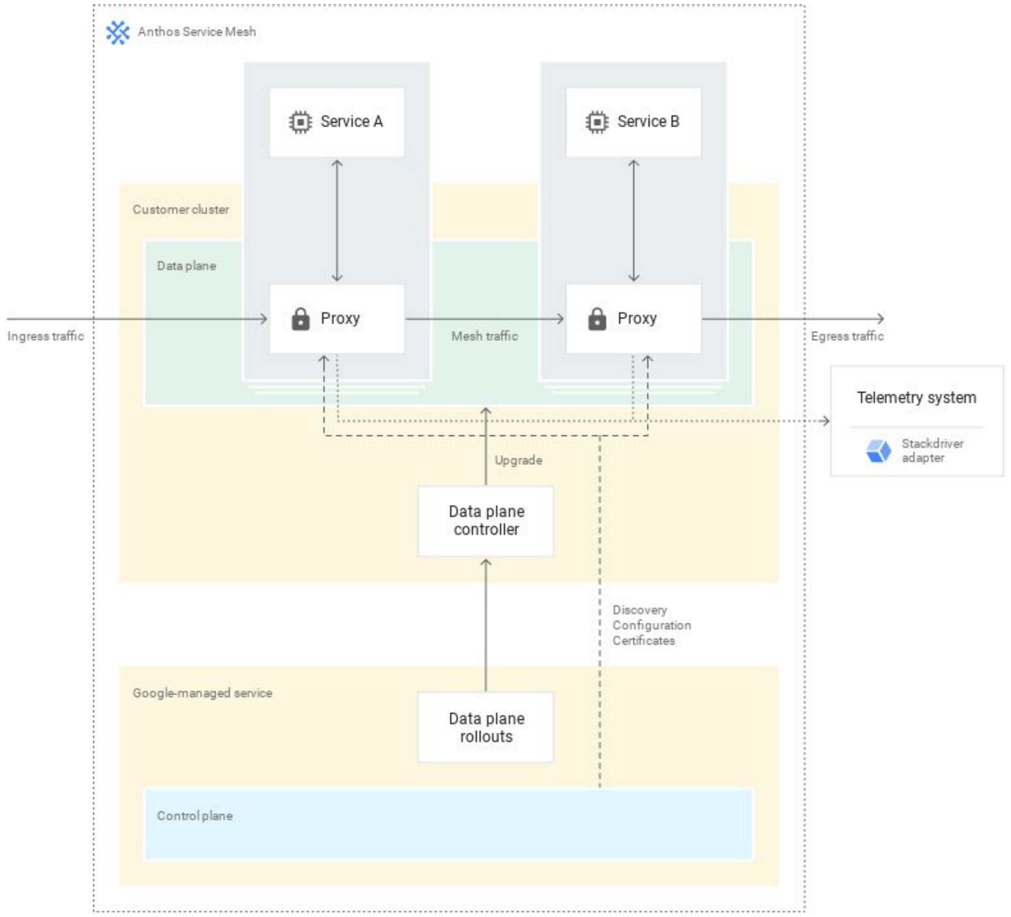
```
# meshconfig.yaml

apiVersion: v1
data:
  mesh: |-
    defaultConfig:
      image:
        imageType: distroless
kind: ConfigMap
metadata:
  name: istio-${ASM_VERSION}
  namespace: istio-system
```

💡 *MCP moves `istiod` into Google's infrastructure and ensure it is always up to date.*

💡 *Istio CNI and `distroless` improve performance at scale with Istio.*

Google Cloud

Managed ASM Architecture



Install *In-cluster* ASM on the cluster

TERMINAL

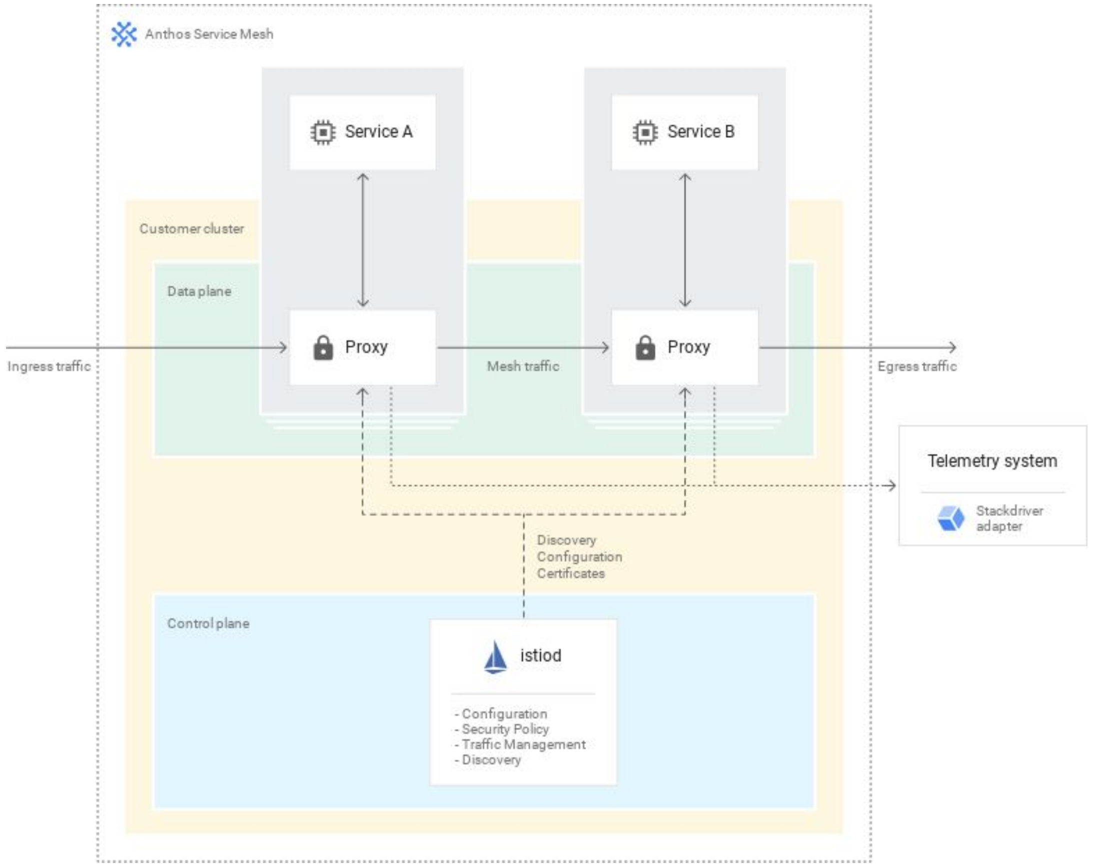
```
$ curl https://storage.googleapis.com/csm-artifacts/asm/asmcli_1.12 >
~/asmcli
$ chmod +x ~/asmcli

$ ~/asmcli install \
      --project_id $PROJECT_ID \
      --cluster_name $GKE_NAME \
      --cluster_location $ZONE \
      --enable-all \
      --option cni-gcp \
      --custom_overlay ditroless-proxy.yaml
```

```
# ditroless-proxy.yaml
---
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  meshConfig:
    defaultConfig:
      image:
        imageType: distroless
```

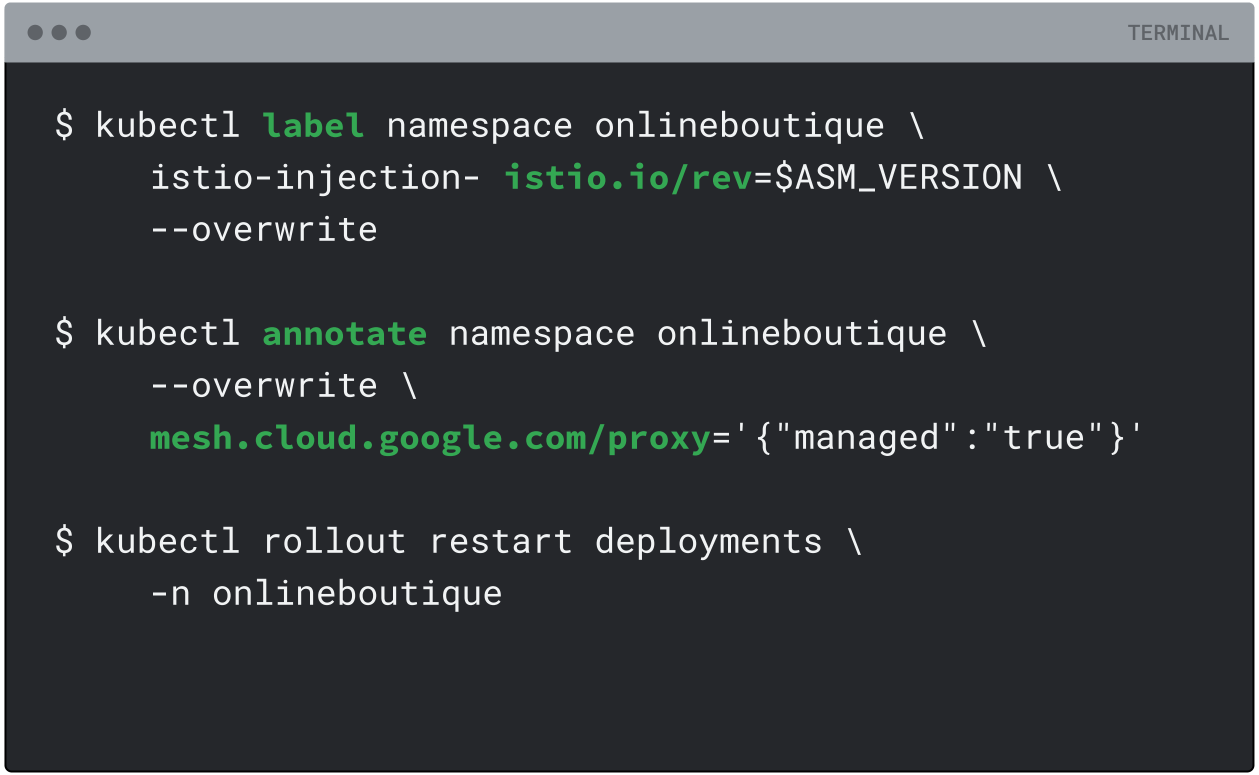
💡 *This is also a well-known setup to improve performance at scale with Istio.*

Google Cloud

In-cluster ASM Architecture



Enable Managed ASM within the onlineboutique namespace



```
$ kubectl label namespace onlineboutique \
    istio-injection- istio.io/rev=$ASM_VERSION \
    --overwrite

$ kubectl annotate namespace onlineboutique \
    --overwrite \
    mesh.cloud.google.com/proxy='{"managed":"true"}'

$ kubectl rollout restart deployments \
    -n onlineboutique
```

💡 *Managed Control Plane (MCP) upgrades proxies to ensure compatibility with current control plane.*

💡 *Managed Data Plane (MDP) upgrades the Envoy sidecars so that they are always up to date.*

Enable *In-cluster* ASM within the onlineboutique namespace

TERMINAL

```
$ ASM_VERSION=$(kubectl get deploy \
    -n istio-system \
    -l app=istiod \
    -o jsonpath={.items[*].metadata.labels.'istio\.io\/rev'}'{"\n"}')

$ kubectl label namespace onlineboutique \
      istio-injection- istio.io/rev=$ASM_VERSION \
      --overwrite

$ kubectl rollout restart deployments \
      -n onlineboutique
```

Configure a `Sidecar` in the onlineboutique namespace

TERMINAL

```
$ kubectl apply -f default-sidecar.yaml \
    loadgenerator-sidecar.yaml \
    -n onlineboutique
```

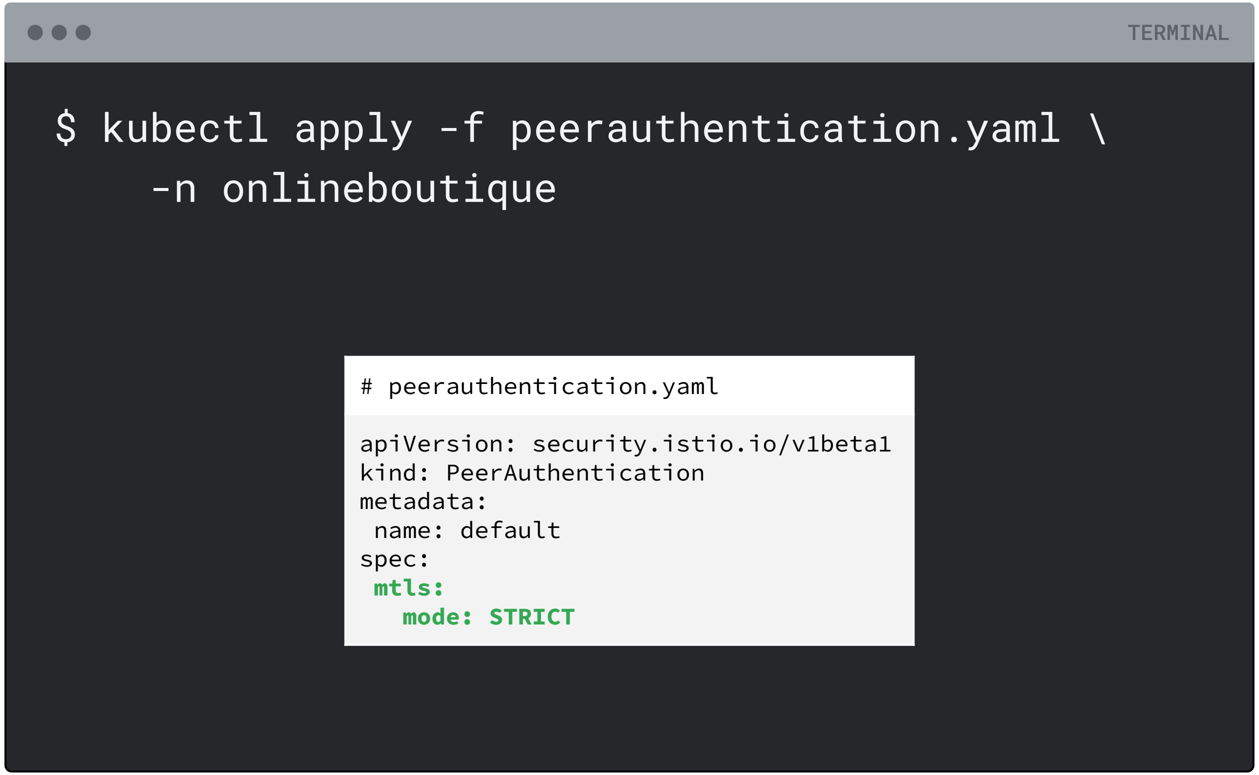
```
# default-sidecar.yaml

apiVersion:
networking.istio.io/v1beta1
kind: Sidecar
metadata:
  name: default
spec:
  egress:
  - hosts:
    - "./*"
    - "istio-system/*"
```

```
# loadgenerator-sidecar.yaml

apiVersion: networking.istio.io/v1beta1
kind: Sidecar
metadata:
  name: loadgenerator
spec:
  workloadSelector:
    labels:
      app: loadgenerator
  egress:
  - hosts:
    - "istio-system/*"
    - "./frontend.onlineboutique.svc.cluster.local"
```

Google Cloud

💡 *This is also a well-known setup to avoid performance issues at scale with Istio.*

Enable mTLS strict within the onlineboutique namespace



TERMINAL

```
$ kubectl apply -f peerauthentication.yaml \
    -n onlineboutique
```

```yaml
# peerauthentication.yaml

apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
 name: default
spec:
 mtls:
   mode: STRICT
```

Setup `AuthorizationPolicy`

TERMINAL

```
$ kubectl apply -f authz-denyall.yaml authz-cartservice.yaml \
      -n onlineboutique
```

```
# authz-denyall.yaml

apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: deny-all
spec:
  {}
```

```
# authz-cartservice.yaml

apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: cartservice
spec:
  selector:
    matchLabels:
      app: cartservice
  rules:
  - from:
    - source:
        principals: ["cluster.local/ns/onlineboutique/sa/frontend",
"cluster.local/ns/onlineboutique/sa/checkoutservice"]
    to:
      - operation:
          paths: ["/hipstershop.CartService/AddItem",
"/hipstershop.CartService/GetCart", "/hipstershop.CartService/EmptyCart"]
          methods: ["POST"]
```

Setup Cloud Armor and a public static IP address

```
TERMINAL

$ gcloud compute security-policies create asm-ingressgateway \
    --description "Block XSS attacks"
$ gcloud compute security-policies rules create 1000 \
    --security-policy asm-ingressgateway \
    --expression "evaluatePreconfiguredExpr('xss-stable')" \
    --action "deny-403" \
    --description "XSS attack filtering"
$ gcloud compute security-policies rules create 12345 \
    --security-policy asm-ingressgateway \
    --expression "evaluatePreconfiguredExpr('cve-canary')" \
    --action "deny-403" \
    --description "CVE-2021-44228 and CVE-2021-45046"

$ gcloud compute addresses create asm-ingressgateway --global
```

Setup Ingress `Gateway` with HTTPS/GCLB

TERMINAL

```
$ kubectl apply -f ingress.yaml service.yaml backendconfig.yaml
managedcertificate.yaml \
     -n asm-ingress
```

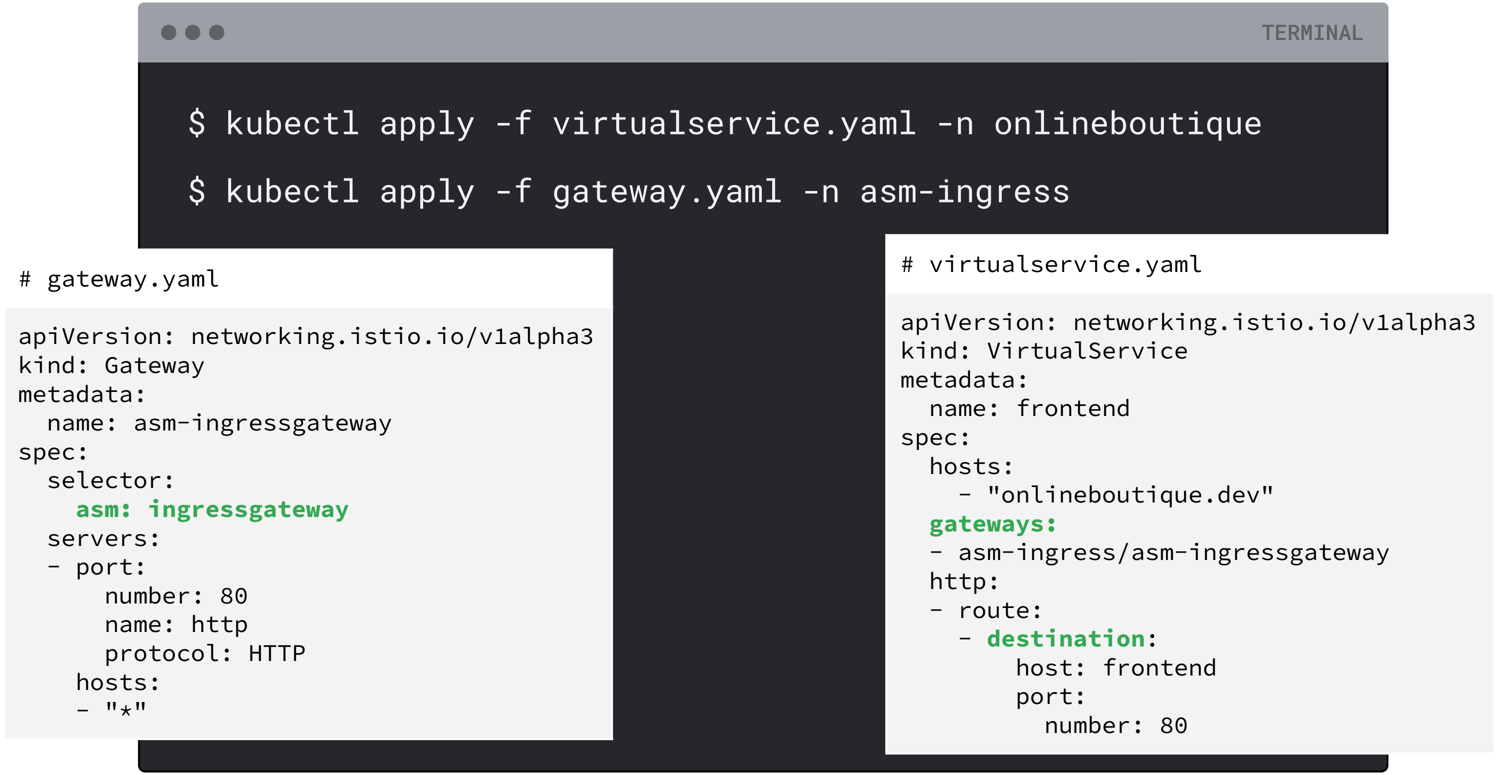
```
# ingress.yaml

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: asm-ingressgateway
  annotations:
    kubernetes.io/ingress.global-static-ip-name: asm-ingressgateway
    networking.gke.io/managed-certificates: onlineboutique
spec:
  rules:
  - host: "*"
    http:
      paths:
      - path: /*
        pathType: ImplementationSpecific
        backend:
          service:
            name: asm-ingressgateway
            port:
              number: 80
```

```
# backendconfig.yaml

apiVersion: cloud.google.com/v1
kind: BackendConfig
metadata:
  name: asm-ingressgateway
spec:
  healthCheck:
    requestPath: /healthz/ready
    port: 15021
    type: HTTP
  securityPolicy:
    name: asm-ingressgateway
```

```
# service.yaml

apiVersion: v1
kind: Service
metadata:
  name: asm-ingressgateway
  annotations:
    cloud.google.com/neg: '{"ingress": true}'
    cloud.google.com/backend-config: '{"default": "asm-ingressgateway"}'
  labels:
    app: asm-ingressgateway
    asm: ingressgateway
spec:
  ports:
  - name: status-port
    port: 15021
    protocol: TCP
    targetPort: 15021
  - name: http2
    port: 80
    targetPort: 8081
  - name: https
    port: 443
    targetPort: 8443
  selector:
    asm: ingressgateway
    app: asm-ingressgateway
  type: ClusterIP
```

```
# managedcertificate.yaml

apiVersion: networking.gke.io/v1
kind: ManagedCertificate
metadata:
  name: onlineboutique
spec:
  domains:
    - mydomain.com
```

Deploy shared `Gateway` and application's `VirtualService`


TERMINAL

```
$ kubectl apply -f virtualservice.yaml -n onlineboutique

$ kubectl apply -f gateway.yaml -n asm-ingress
```

```
# gateway.yaml

apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: asm-ingressgateway
spec:
  selector:
    asm: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
```

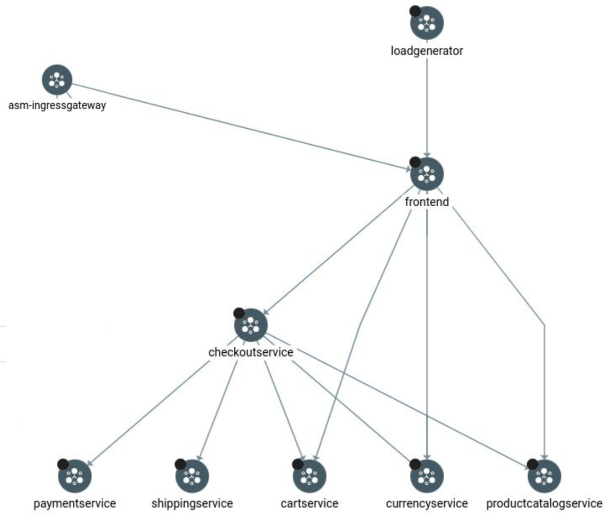
```
# virtualservice.yaml

apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: frontend
spec:
  hosts:
    - "onlineboutique.dev"
  gateways:
  - asm-ingress/asm-ingressgateway
  http:
  - route:
    - destination:
        host: frontend
        port:
          number: 80
```

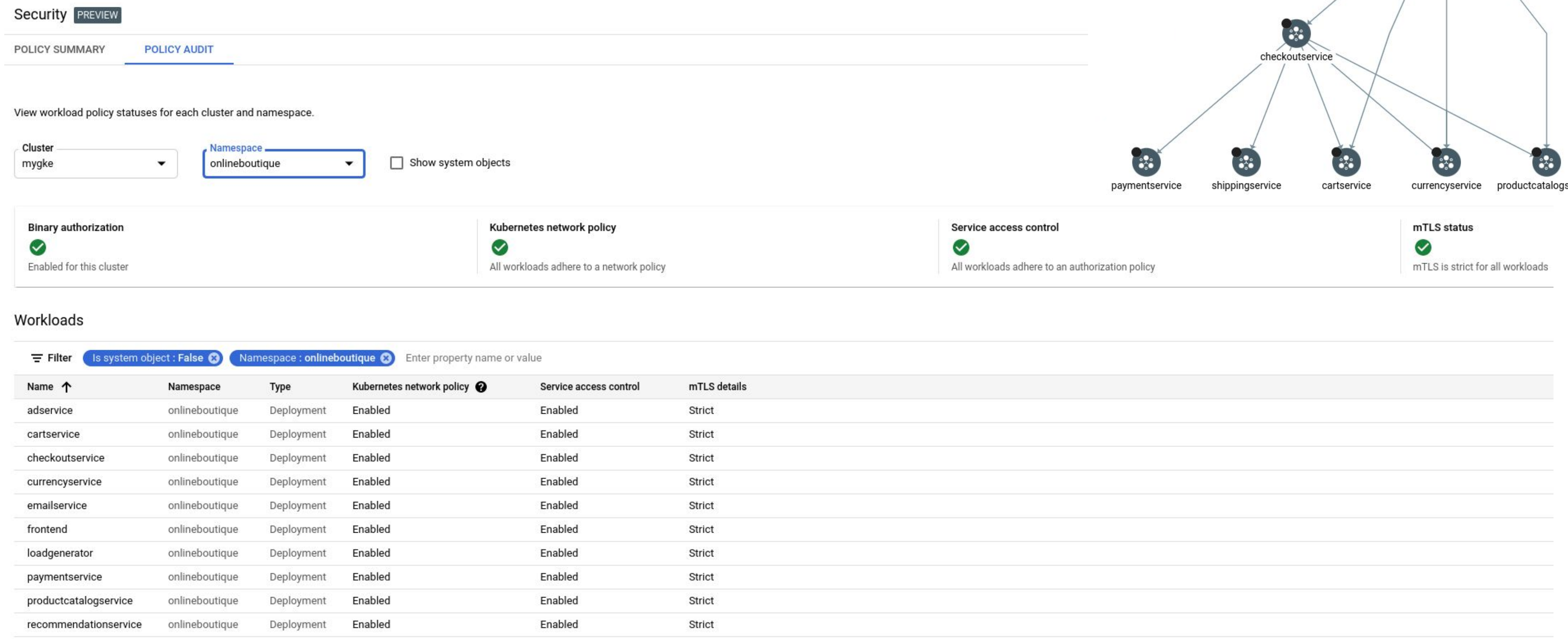
💡 *The shared `Gateway` is created in the asm-ingress namespace owned by the platform admin.*

💡 *The `VirtualService` is created in the application namespace owned by the application owner.*

Google Cloud

That's a wrap!




Security PREVIEW

POLICY SUMMARY | POLICY AUDIT

View workload policy statuses for each cluster and namespace.

Cluster: mygke
Namespace: onlineboutique
Show system objects

Binary authorization	Kubernetes network policy	Service access control	mTLS status
✓ Enabled for this cluster	✓ All workloads adhere to a network policy	✓ All workloads adhere to an authorization policy	✓ mTLS is strict for all workloads

Workloads

Filter: Is system object : False, Namespace : onlineboutique. Enter property name or value

Name ↑	Namespace	Type	Kubernetes network policy	Service access control	mTLS details
adservice	onlineboutique	Deployment	Enabled	Enabled	Strict
cartservice	onlineboutique	Deployment	Enabled	Enabled	Strict
checkoutservice	onlineboutique	Deployment	Enabled	Enabled	Strict
currencyservice	onlineboutique	Deployment	Enabled	Enabled	Strict
emailservice	onlineboutique	Deployment	Enabled	Enabled	Strict
frontend	onlineboutique	Deployment	Enabled	Enabled	Strict
loadgenerator	onlineboutique	Deployment	Enabled	Enabled	Strict
paymentservice	onlineboutique	Deployment	Enabled	Enabled	Strict
productcatalogservice	onlineboutique	Deployment	Enabled	Enabled	Strict
recommendationservice	onlineboutique	Deployment	Enabled	Enabled	Strict

Resources

See the entire story here: alwaysupalwayson.com/asm-security

Resources:

- Istio by Examples (thanks Megan!)
- Secured Ingress Gateway (thanks Ameer and Alex!)
- Istio Security best practices

Code used for this session:

- asm-ingress manifests
- onlineboutique manifests
- mygkecluster setup

Complementary to this, you should implement this below too:

- Secured Egress Gateway (thanks Ameer and James!)
- Network Policies
- Policy Controller / OPA Gatekeeper

Thank you!
Sail safe out there! ;)