KubeCon | CloudNativeCon

North America 2023

KubeCon | CloudNativeCon
North America 2023

Untangling Your Service Mesh With Feature Gates

Niranjan Shankar

Software Engineer @ Microsoft
Azure Kubernetes Service

Introduction - About Me

- Azure Kubernetes Service (AKS)
  - Istio AKS Add-On
- Azure Container Upstream
  - Open Service Mesh (OSM)
  - OSM Azure-Arc Extension

Istio

Open Service Mesh

https://www.linkedin.com/in/niranjan-shankar-766164149/

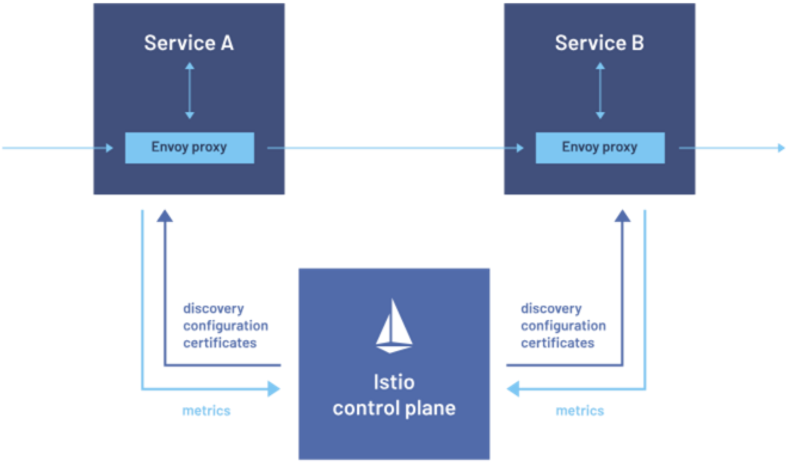
https://github.com/nshankar13/tutorials

Overview

- Problem: tangled service mesh
- Solution: how to untangle your service mesh with “feature-gates”
- Criteria for which features, configurations, and APIs to allow
- Takeaways and relevant developments.

Service Mesh Features and Configurations

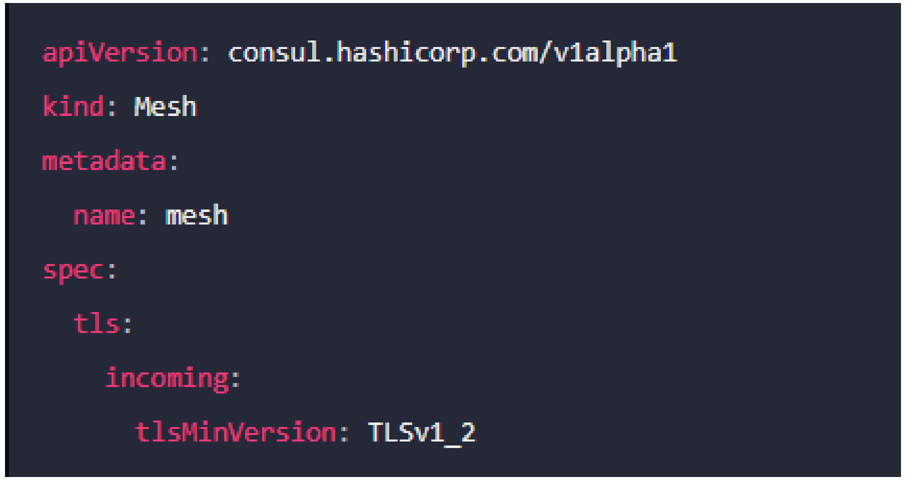
- Broad feature set and configurability
  - Traffic management, security, observability, etc.
- Power and extensibility of Envoy
  - Exceptions: Linkerd, sidecarless, eBPF-based service mesh



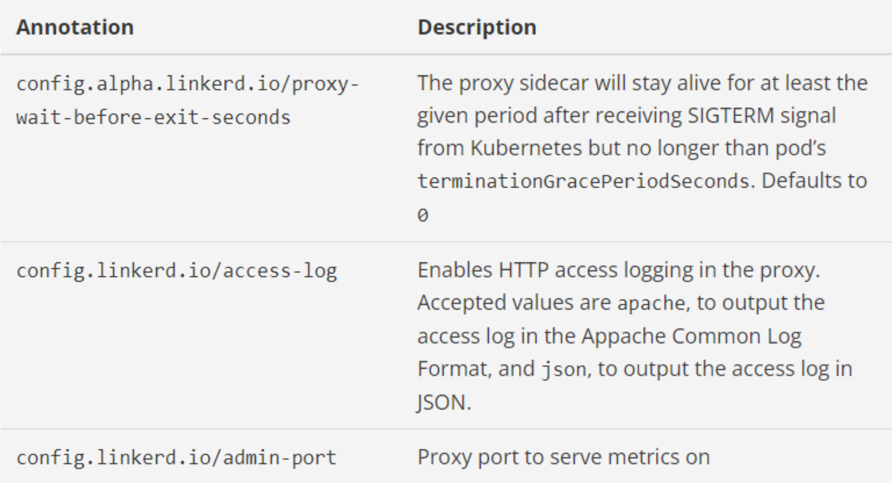
Service Mesh - Envoy sidecar proxy model

Service Mesh Features and Configurations

- Extensive configuration options, and multiple ways of configuring.
  - Installation values (CLI or Helm)
  - Feature flags and environment variables
  - [ConfigMaps]()
  - Custom resources
  - [Resource annotations]()



```
apiVersion: consul.hashicorp.com/v1alpha1
kind: Mesh
metadata:
  name: mesh
spec:
  tls:
    incoming:
      tlsMinVersion: TLSv1_2
```



Annotation	Description
`config.alpha.linkerd.io/proxy-wait-before-exit-seconds`	The proxy sidecar will stay alive for at least the given period after receiving SIGTERM signal from Kubernetes but no longer than pod's `terminationGracePeriodSeconds`. Defaults to `0`
`config.linkerd.io/access-log`	Enables HTTP access logging in the proxy. Accepted values are `apache`, to output the access log in the Appache Common Log Format, and `json`, to output the access log in JSON.
`config.linkerd.io/admin-port`	Proxy port to serve metrics on

Service Mesh Features and Configurations

- Upsides and advantages
  - Flexibility and versatility
    - Granularity: mesh-wide vs namespace vs workload-scoped
  - Some degree of complexity inevitable
    - Load balancing, networking resiliency, traffic control
    - Multicluster, external control plane deployment models

Service Mesh Features and Configurations

- Drawbacks
  - Operational complexity
    - APIs require steep learning curve
  - Confusion and noise
    - Organizations and end-users need a *subset*, not all, features
    - Multiple entry points of configuration and sources of truth
      - Ex: Istio proxy env variables - MeshConfig, resource annotations, and custom resource

Service Mesh Features and Configurations

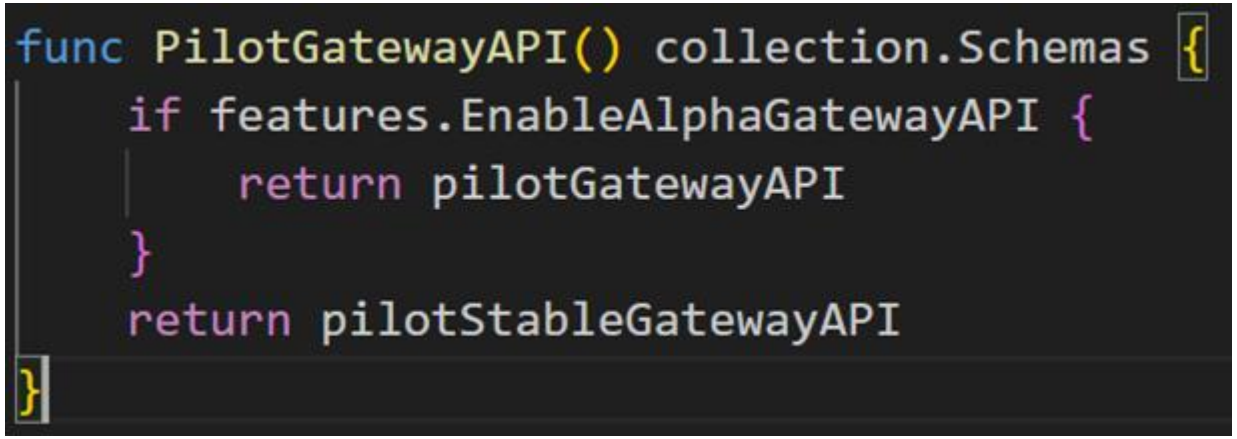
- Drawbacks - a “tangled” mesh
  - Misconfigurations and policy mismatches
    - Broken traffic and security vulnerabilities
  - Experimental, alpha, or deprecated features in production environments
  - Circumvention of administrative controls and desired constraints
  - High resource utilization
  - Bottlenecks between platform engineers and developers

Solution: “Feature Gates”

- A.K.A “feature flags” or “feature toggles”
  - Build-time flags or variables
  - Enable or disable execution of specific features and code paths
    - Encapsulate within if-statements and conditionals
- Feature gates in Kubernetes
  - Key=value pairs to turn features on or off.
  - Proposed changes (Tim Hockin)

Solution: “Feature Gates”

- For service mesh
  - Istio - Installation values (CLI / Helm) and Pilot env vars
    - Ex: PILOT_ENABLE_ALPHA_GATEWAY_API
    - Also MeshConfig, ProxyConfig, and proxy env variables
  - OSM - MeshConfig and installation values



```
func PilotGatewayAPI() collection.Schemas {
    if features.EnableAlphaGatewayAPI {
        return pilotGatewayAPI
    }
    return pilotStableGatewayAPI
}
```

Solution: “Feature Gates”

- But on/off feature toggles aren’t enough
  - Usually just tied to feature lifecycle
  - Not always reliable or practical

Solution: “Feature Gates”

- Via external mechanisms, also need to:
  - Restrict configurations and remediate configuration drift
  - Abstract and “hide” APIs and error-prone configurations
  - Disallow and validate custom resources
  - Block specific annotations or labels
  - Enforce policies and constraints

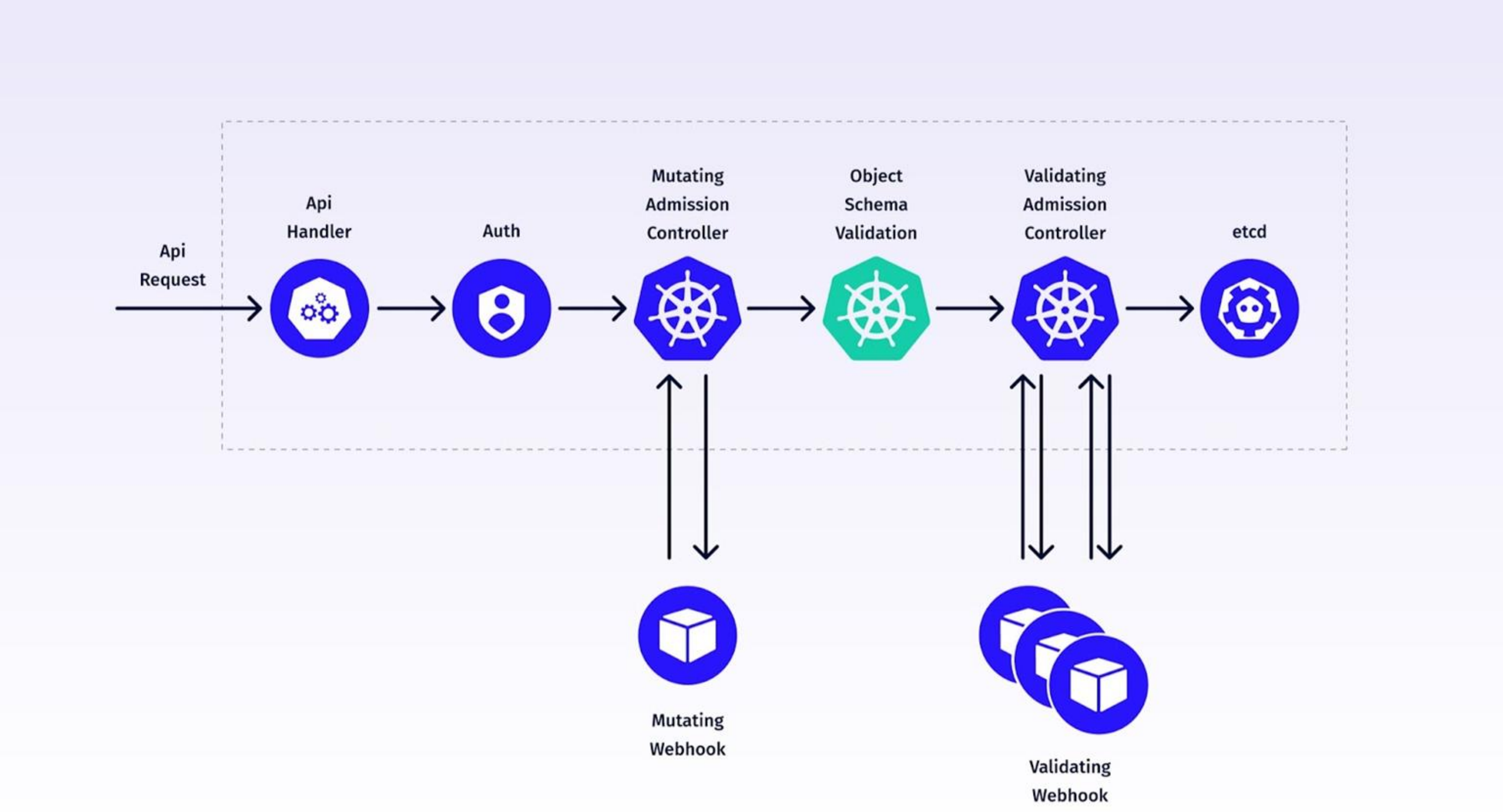
Solution: “Feature Gates”

- Typically done by mesh administrator
  - A.K.A platform admin, cluster admin
  - Establish guardrails and restrictions for the mesh
- Why?
  - Simplify service mesh operations
    - Narrower feature set and smaller configuration surface area
  - Guardrails
    - Prevent misconfigurations and configuration drift
    - Enforce desired behavior and best practices
  - Mitigate resource consumption

Solution: “Feature Gates”

- How?
  - Admission controllers
    - Shift-left approach (CI linters)
  - API abstractions
  - GitOps tools
  - Role-Based Access Control

Admission Controllers



https://www.armosec.io/blog/kubernetes-admission-controller/

Admission Controllers

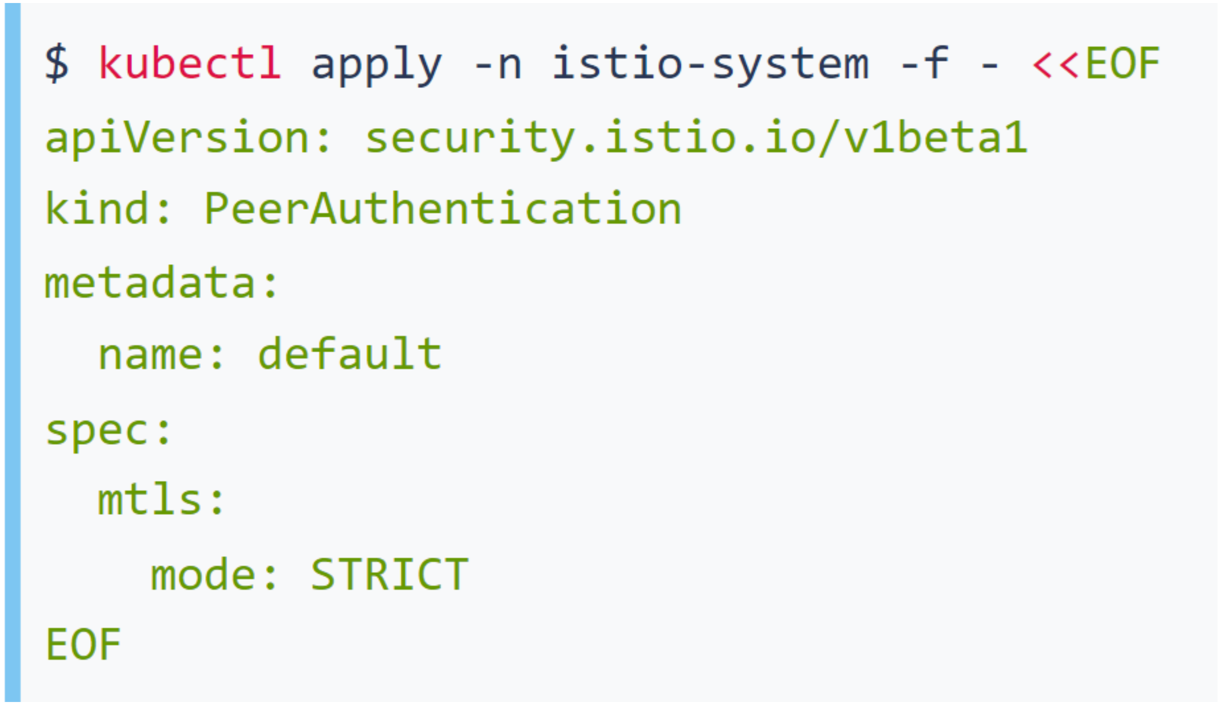
- Gatekeeper
    - Admission controller that uses Open Policy Agent
    - Write policies in Rego
    - Istio and Linkerd examples
- Kyverno
    - No unique programming language
    - Istio, Linkerd, and Consul policies
- K8s 1.28 and C.E.L

Admission Controllers

- Gatekeeper
  - ConstraintTemplate:
    - Defines policy violation
  - Constraint
    - Which resources ConstraintTemplate rules apply to
    - Passes in parameters to ConstraintTemplate

Admission Controllers

- Custom resources
  - Istio PeerAuthentication and mTLS STRICT
    - Linkerd: Authorization Policy

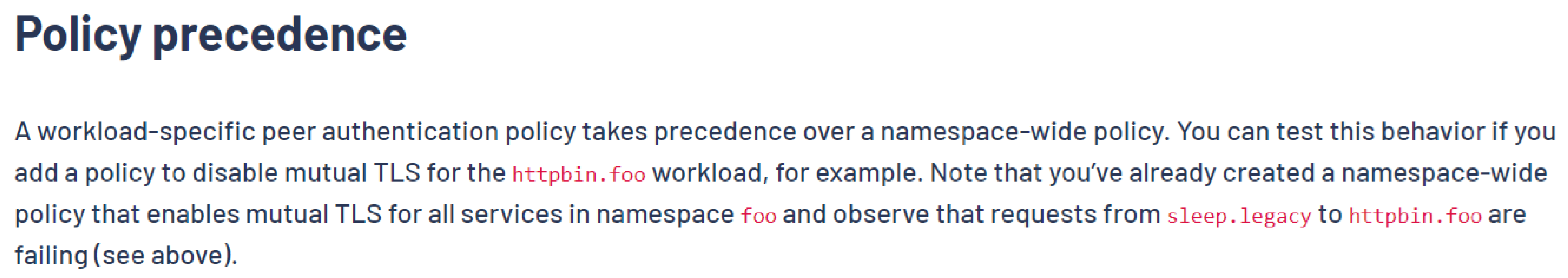


```
$ kubectl apply -n istio-system -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
spec:
  mtls:
    mode: STRICT
EOF
```

https://istio.io/latest/docs/tasks/security/authentication/mtls-migration/

Admission Controllers

- Custom resources
  - Istio PeerAuthentication and mTLS STRICT
    - Prevent overwriting of global STRICT mTLS



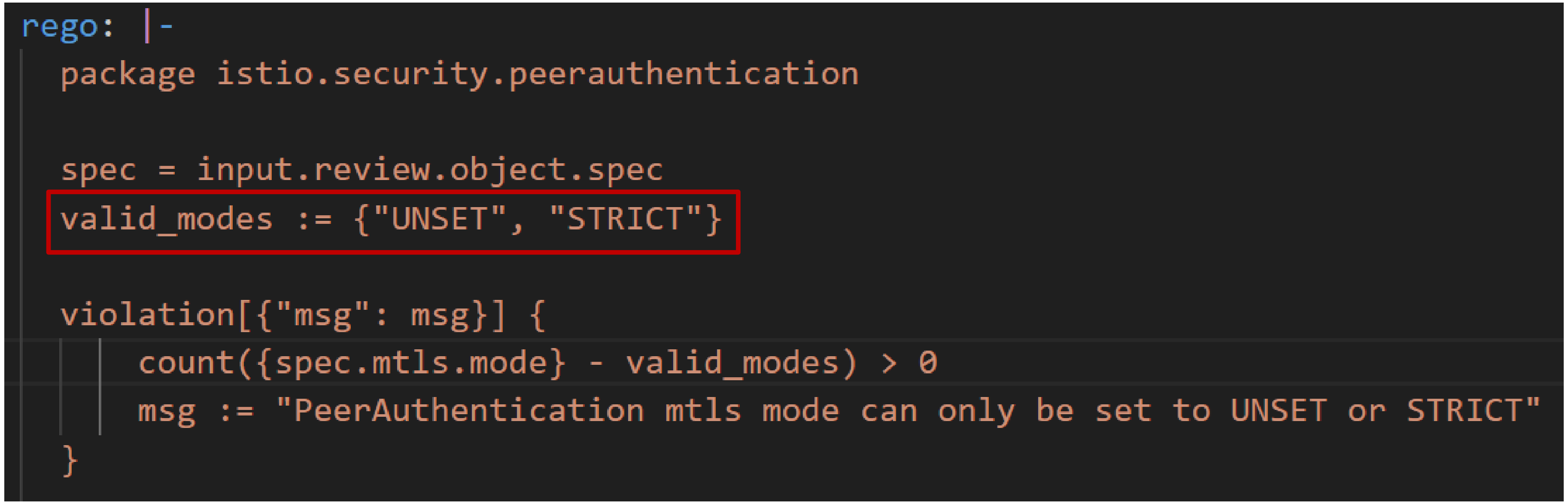
Policy precedence

A workload-specific peer authentication policy takes precedence over a namespace-wide policy. You can test this behavior if you add a policy to disable mutual TLS for the `httpbin.foo` workload, for example. Note that you've already created a namespace-wide policy that enables mutual TLS for all services in namespace `foo` and observe that requests from `sleep.legacy` to `httpbin.foo` are failing (see above).

https://istio.io/latest/docs/tasks/security/authentication/authn-policy/#policy-precedence

Admission Controllers

- Custom resources
  - Istio PeerAuthentication and mTLS STRICT
    - Prevent overwriting of global STRICT mTLS

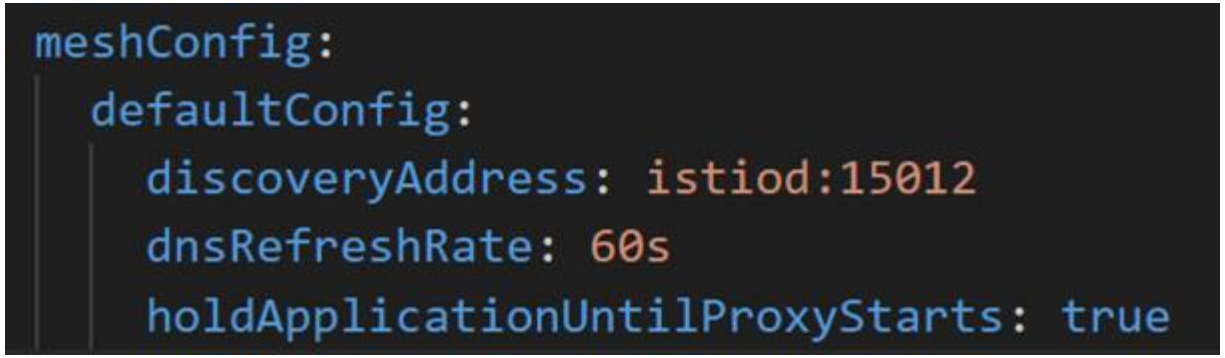


```
rego: |-
  package istio.security.peerauthentication

  spec = input.review.object.spec
  valid_modes := {"UNSET", "STRICT"}

  violation[{"msg": msg}] {
    count({spec.mtls.mode} - valid_modes) > 0
    msg := "PeerAuthentication mtls mode can only be set to UNSET or STRICT"
  }
```

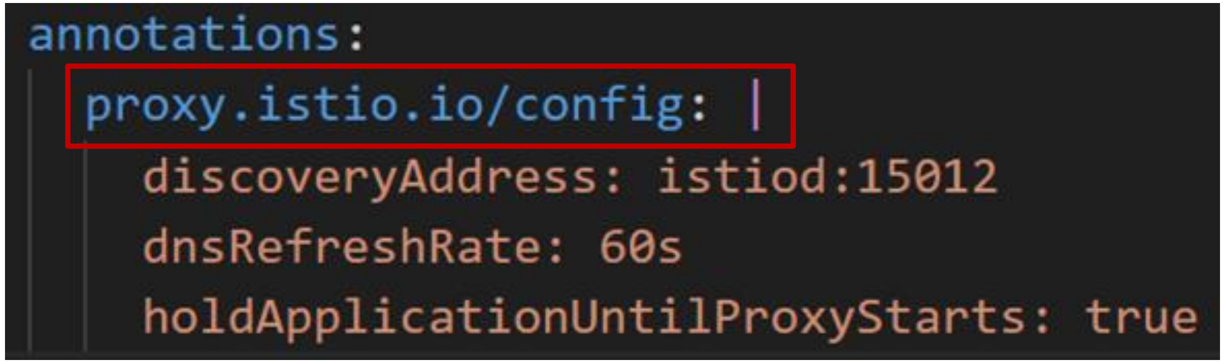
https://github.com/mathieu-benoit/istio-gatekeeper-demos/tree/main/policies/constrainttemplates/strict-mtls

Admission Controllers

- Mesh and Proxy Configuration



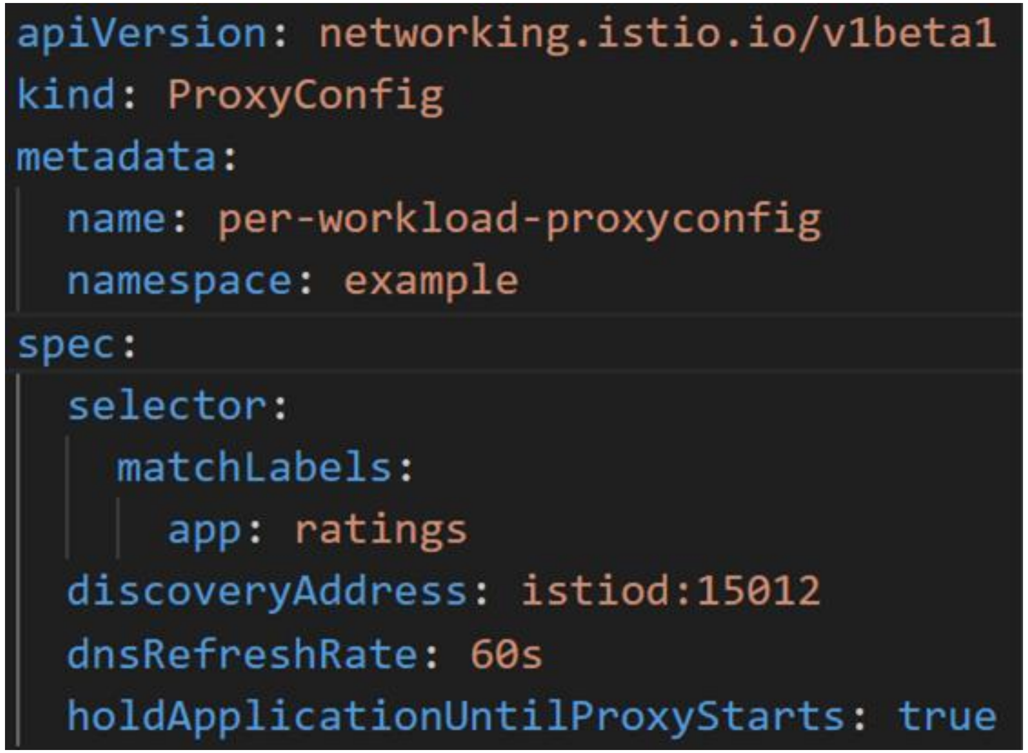
```
meshConfig:
  defaultConfig:
    discoveryAddress: istiod:15012
    dnsRefreshRate: 60s
    holdApplicationUntilProxyStarts: true
```



```
annotations:
  proxy.istio.io/config: |
    discoveryAddress: istiod:15012
    dnsRefreshRate: 60s
    holdApplicationUntilProxyStarts: true
```

Admission Controllers

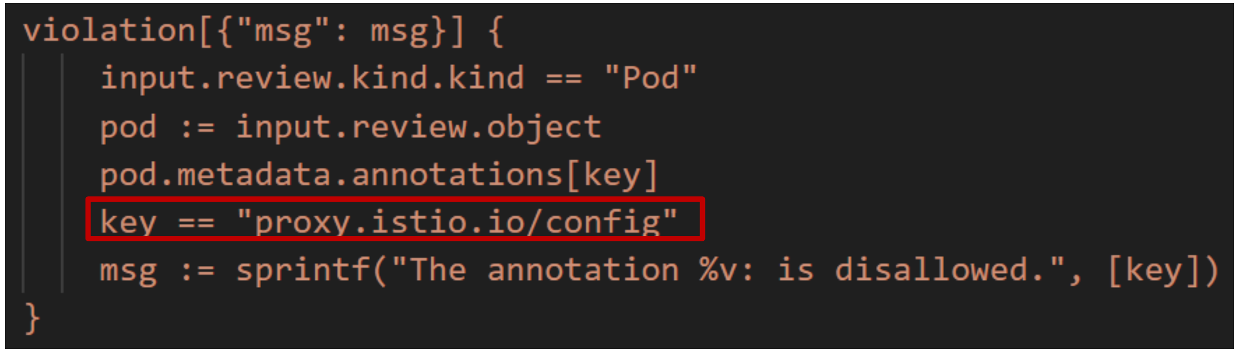
- Mesh and Proxy Configuration
  - ProxyConfig CR and proxy.config.io/config override MeshConfig



```yaml
apiVersion: networking.istio.io/v1beta1
kind: ProxyConfig
metadata:
  name: per-workload-proxyconfig
  namespace: example
spec:
  selector:
    matchLabels:
      app: ratings
  discoveryAddress: istiod:15012
  dnsRefreshRate: 60s
  holdApplicationUntilProxyStarts: true
```

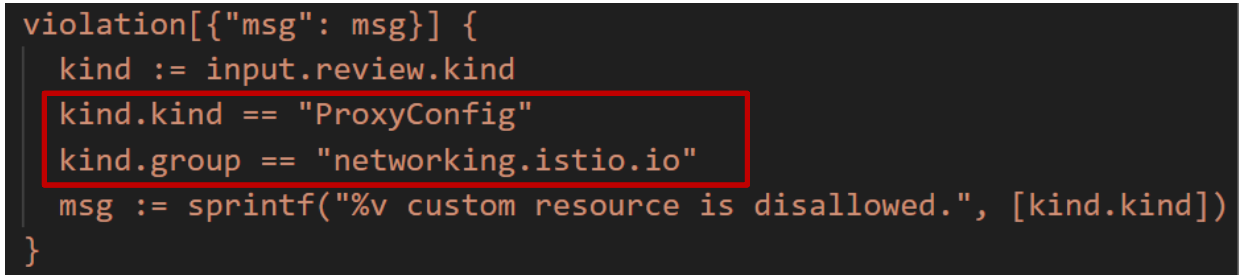
Admission Controllers

- Mesh and Proxy Configuration
  - Gatekeeper constraint template policies:



```
violation[{"msg": msg}] {
    input.review.kind.kind == "Pod"
    pod := input.review.object
    pod.metadata.annotations[key]
    key == "proxy.istio.io/config"
    msg := sprintf("The annotation %v: is disallowed.", [key])
}
```

Disallow Pods and Deployments with proxy.istio.io/config annotation



```
violation[{"msg": msg}] {
  kind := input.review.kind
  kind.kind == "ProxyConfig"
  kind.group == "networking.istio.io"
  msg := sprintf("%v custom resource is disallowed.", [kind.kind])
}
```

Disallow the ProxyConfig Custom Resource

Admission Controllers

- MeshConfig and ProxyConfig
  - Enforce desired mesh-wide behavior
    - Prevent bypassing
  - Narrowed down to one source of truth
  - Caveat: just one of many operational patterns
    - Some environments and setups may necessitate granularity

Admission Controllers

- Other features and configurations to “gate”
  - Enforce that sidecar injection takes place
    - Namespace label: istio-injection=enabled
    - Pod label: sidecar.istio.io/inject
  - Disallow overly permissive traffic configurations
    - Ex: VirtualServices and Gateways that attempt to use wildcards
  - Block experimental, alpha, or deprecated custom resources

Continuous Integration Linting

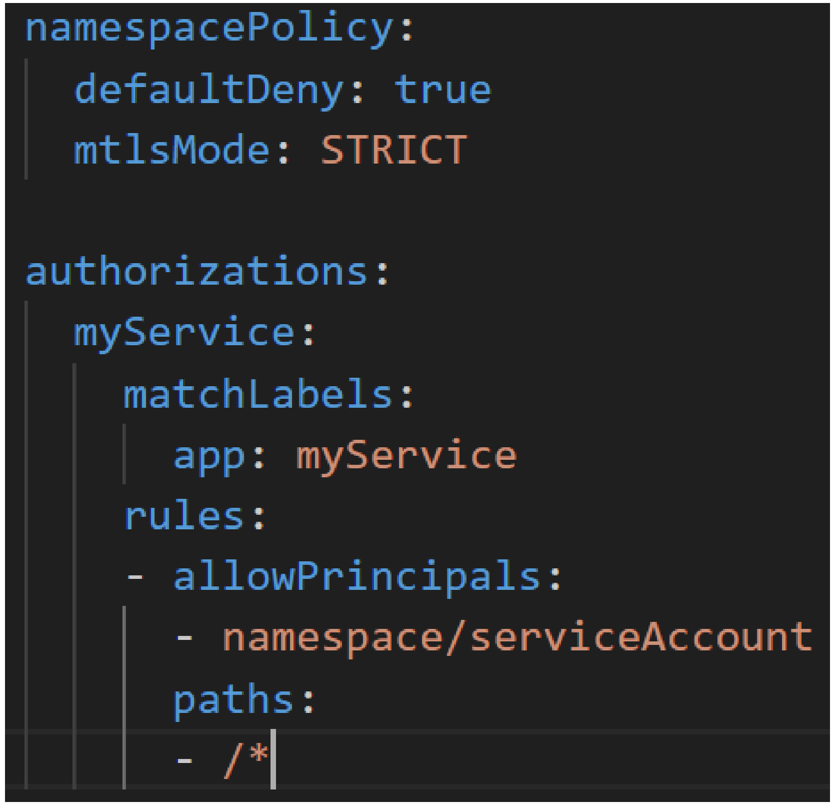
- Continuous Integration
  - Shift-left approach
  - Use linters to validate CRs and manifests (can also be written in Rego)
    - Using gator CLI
      - Example
    - conftest

API Abstractions

- Feature gating?
  - “Hiding” low-level service mesh APIs from service owners
  - Selectively exposing particular features and configuration options
- Highly popular approach:
  - Istio:
    - [Salesforce]
    - [Airbnb]
    - [Splunk]
    - [Intuit]
    - [GoPay]

API Abstractions

- Example: Istio AuthorizationPolicy and Helm

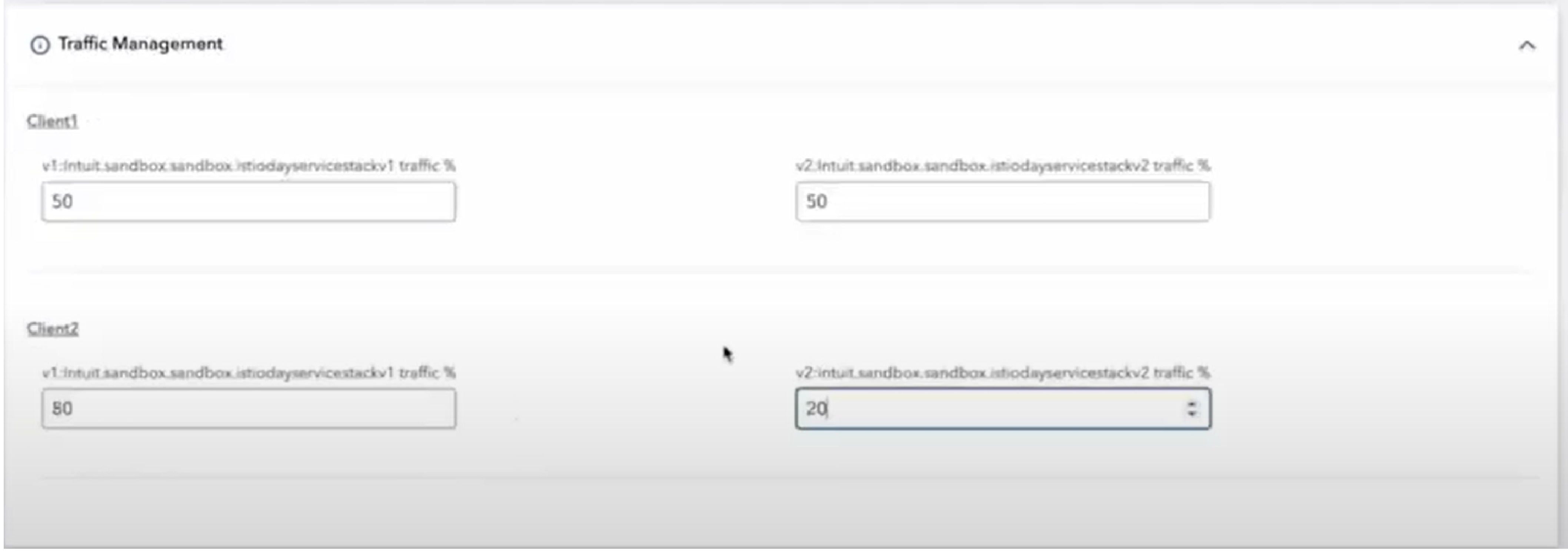


```yaml
namespacePolicy:
  defaultDeny: true
  mtlsMode: STRICT

authorizations:
  myService:
    matchLabels:
      app: myService
    rules:
    - allowPrincipals:
      - namespace/serviceAccount
      paths:
      - /*
```

https://github.com/salesforce/helm-starter-istio

API Abstractions

- Example: Istio VirtualService and declarative UI



Traffic Management

Client1

v1:intuit.sandbox.sandbox.istiodayservicestackv1 traffic %
50

v2:intuit.sandbox.sandbox.istiodayservicestackv2 traffic %
50

Client2

v1:intuit.sandbox.sandbox.istiodayservicestackv1 traffic %
80

v2:intuit.sandbox.sandbox.istiodayservicestackv2 traffic %
20

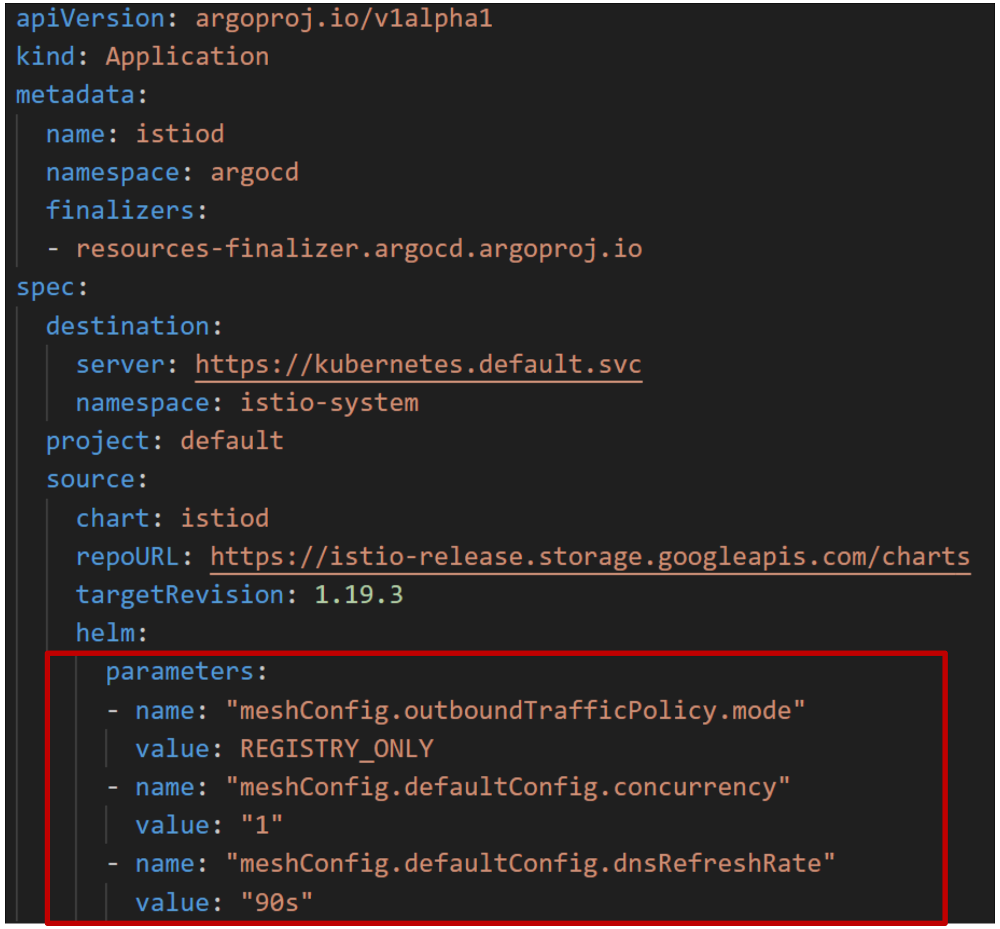
Intuit: https://www.youtube.com/watch?v=LOhfCrxnghI&list=PL7wB27eZmdffVBMSaXzXz02mZb3gZWlsi&index=14

GitOps Tools

- Declarative mesh management
  - Configuration-as-Code
- Continuous reconciliation
  - Prevent configuration drift - remediate changes to:
    - Mesh installation values and Kubernetes resources
    - Mesh configuration (ConfigMaps or custom resources)

GitOps Tools

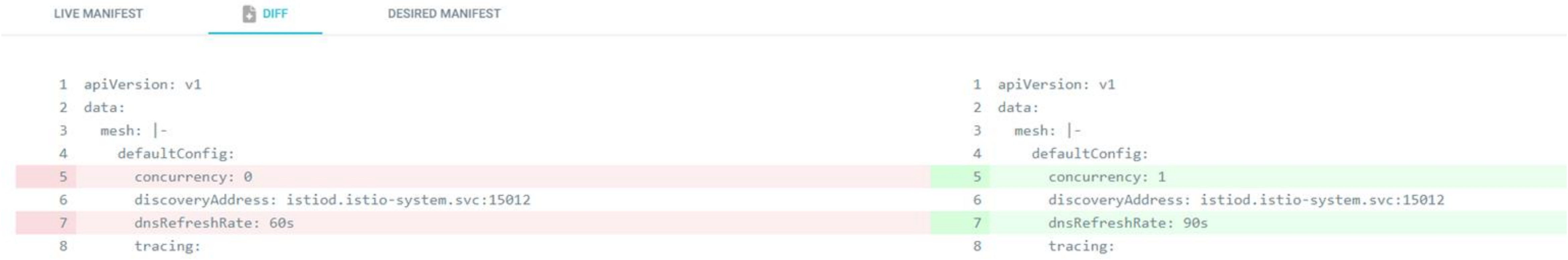
- Example: Istio and ArgoCD
  - [Linkerd demo]()



```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: istiod
  namespace: argocd
  finalizers:
  - resources-finalizer.argocd.argoproj.io
spec:
  destination:
    server: https://kubernetes.default.svc
    namespace: istio-system
  project: default
  source:
    chart: istiod
    repoURL: https://istio-release.storage.googleapis.com/charts
    targetRevision: 1.19.3
    helm:
      parameters:
      - name: "meshConfig.outboundTrafficPolicy.mode"
        value: REGISTRY_ONLY
      - name: "meshConfig.defaultConfig.concurrency"
        value: "1"
      - name: "meshConfig.defaultConfig.dnsRefreshRate"
        value: "90s"
```

GitOps Tools

- Example: Istio and ArgoCD
  - Configuration drift detection for Istio MeshConfig
    - Automatic Self-Healing: spec.syncPolicy.automated.selfHeal = true

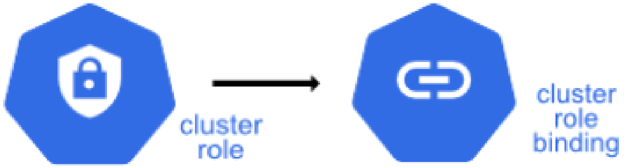


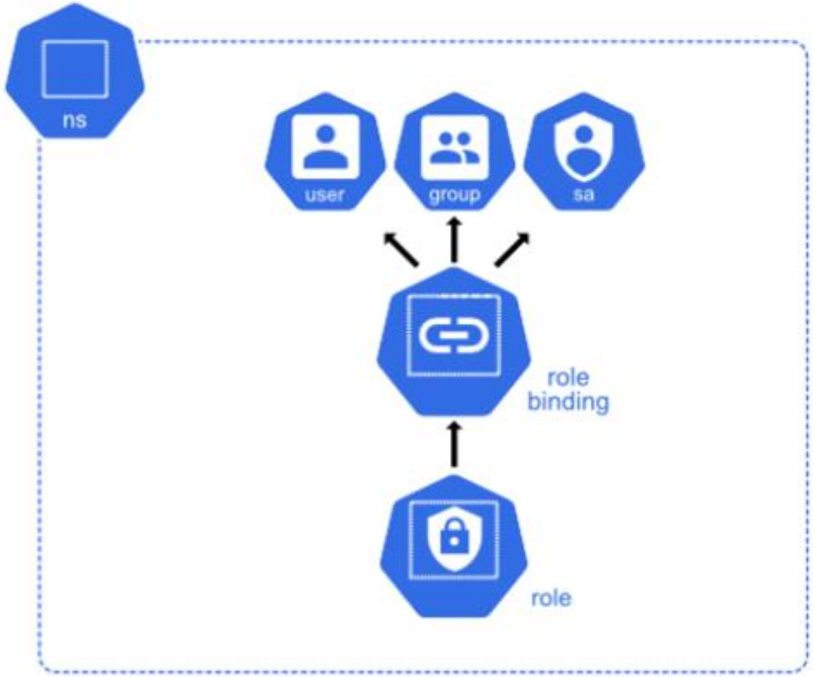
```
apiVersion: v1
data:
  mesh: |-
    defaultConfig:
      concurrency: 0
      discoveryAddress: istiod.istio-system.svc:15012
      dnsRefreshRate: 60s
      tracing:
```

```
apiVersion: v1
data:
  mesh: |-
    defaultConfig:
      concurrency: 1
      discoveryAddress: istiod.istio-system.svc:15012
      dnsRefreshRate: 90s
      tracing:
```

Kubernetes RBAC

- Restrict access to administrative namespaces
  - Ex: kube-system, control plane, and ingress and egress namespaces
- Limit who can manage specific resources
  - Based on criteria (complexity, risk-level, etc)





Feature Gating - Criteria

- Operational complexity and ease-of-use
- Feature status
- Risk-level and mesh security
- Impact on resource consumption and performance
- Organizational requirements and use-cases

Feature Gating - Criteria

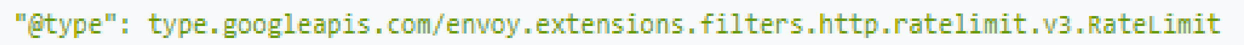
- Operational complexity and ease-of-use
    - How easy to understand and configure?
    - Adequate documentation and support?
    - Multiple ways of enabling and configuring?
        - Try to restrict to one

Feature Gating - Criteria

- Operational complexity and ease-of-use
  - Example: Istio and EnvoyFilter API
    - Highly complex
    - Currently alpha
    - But widely used:
      - Rate limiting
      - Lua scripts
      - WASM extensions
      - PROXY protocol
      - Gzip compression

Feature Gating - Criteria

- Operational complexity and ease-of-use
  - Example: Istio and EnvoyFilter API
    - Solution:
      - Allow in limited capacities (via admission control):
        - Rate limiting:

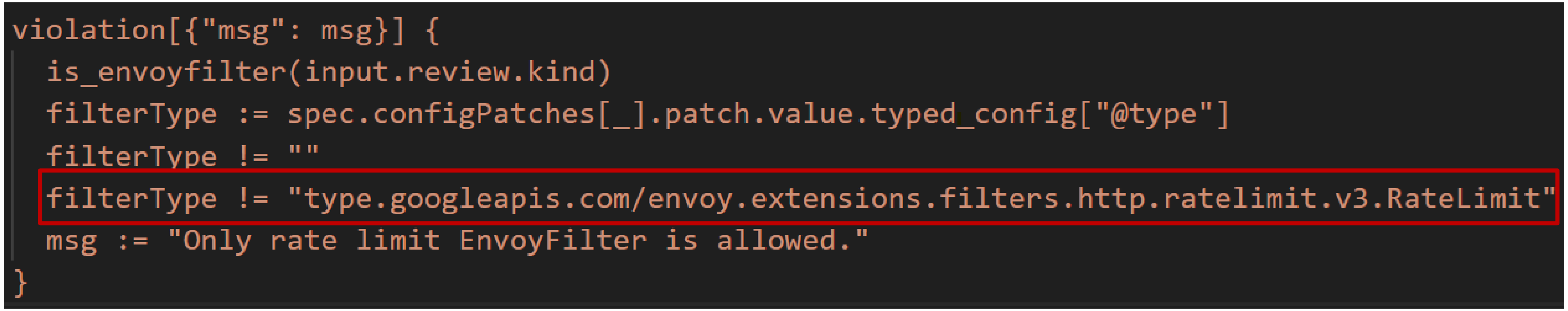


```
"@type": type.googleapis.com/envoy.extensions.filters.http.ratelimit.v3.RateLimit
```

      - Create an abstraction layer
      - Restrict to experts and cluster admins (RBAC or admission control)

Feature Gating - Criteria

- Operational complexity and ease-of-use
  - Example: Istio and EnvoyFilter API

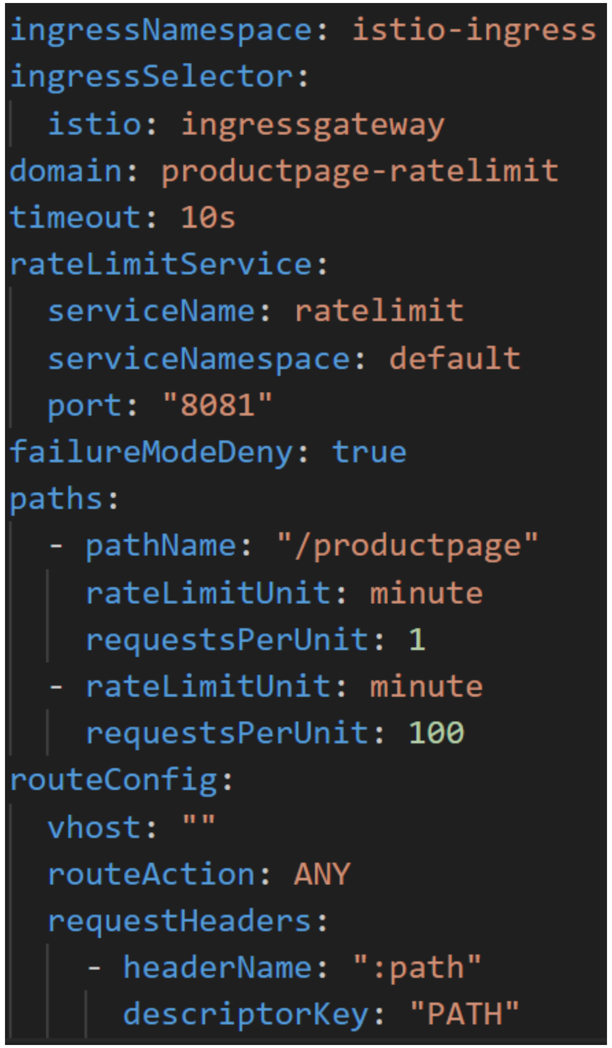


```
violation[{"msg": msg}] {
  is_envoyfilter(input.review.kind)
  filterType := spec.configPatches[_].patch.value.typed_config["@type"]
  filterType != ""
  filterType != "type.googleapis.com/envoy.extensions.filters.http.ratelimit.v3.RateLimit"
  msg := "Only rate limit EnvoyFilter is allowed."
}
```

Reject all non rate-limit EnvoyFilters

Feature Gating - Criteria

- Operational complexity and ease-of-use
  - Example: Istio and EnvoyFilter API
    - API abstraction layer
      - GoPay example: RateLimit CRD via K8s operator



```yaml
ingressNamespace: istio-ingress
ingressSelector:
  istio: ingressgateway
domain: productpage-ratelimit
timeout: 10s
rateLimitService:
  serviceName: ratelimit
  serviceNamespace: default
  port: "8081"
failureModeDeny: true
paths:
  - pathName: "/productpage"
    rateLimitUnit: minute
    requestsPerUnit: 1
  - rateLimitUnit: minute
    requestsPerUnit: 100
routeConfig:
  vhost: ""
  routeAction: ANY
  requestHeaders:
    - headerName: ":path"
      descriptorKey: "PATH"
```

EnvoyFilter Helm Values Abstraction

Feature Gating - Criteria

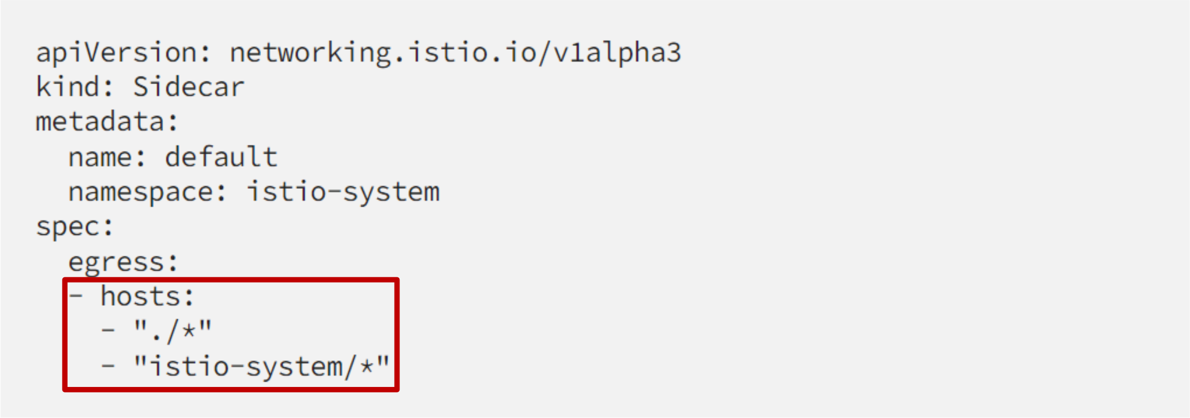
- Feature status
  - Experimental, Alpha, Beta, or Stable
    - Istio Telemetry API (currently alpha)
    - Disallow through admission control, use MeshConfig instead
  - Deprecated:
    - Some Istio MeshConfig fields and annotations

Feature Gating - Criteria

- Risk-level and mesh security
  - mTLS and zero-trust framework
    - PeerAuthentication, RequestAuthentication, AuthorizationPolicy, DestinationRule
    - ServiceEntry, Gateway, Sidecar
      - Securing inbound and outbound traffic
  - Possibility of misconfiguration or policy mismatch
    - Be selective in exposing fields
    - Validate via admission control
    - Delegate to admins and/or mesh experts

Feature Gating - Criteria

- Impact on resource consumption and performance
  - Istio Sidecar custom resource in root namespace (istio-system)
    - Enforce to limit Envoy config size
    - Prevent bypassing on a namespace or workload level
  - MeshConfig - concurrency field

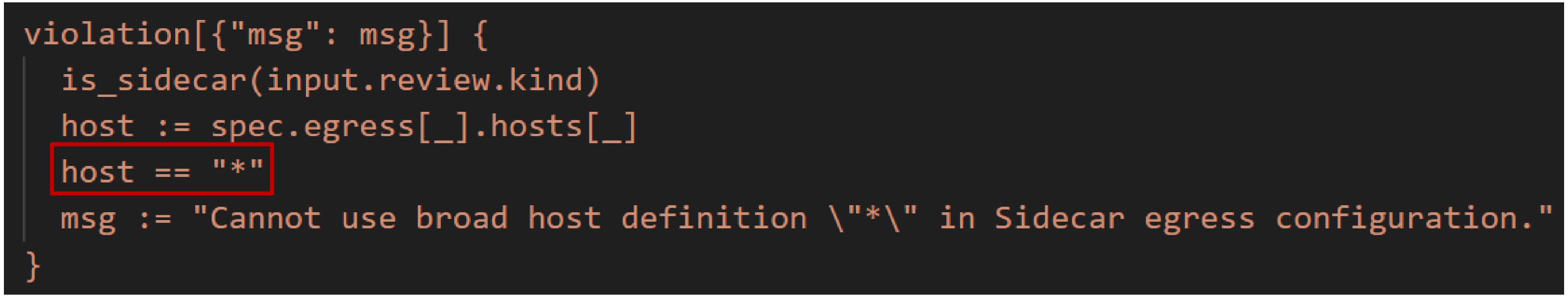


```
apiVersion: networking.istio.io/v1alpha3
kind: Sidecar
metadata:
  name: default
  namespace: istio-system
spec:
  egress:
  - hosts:
    - "./*"
    - "istio-system/*"
```

Sidecar CR to restrict all sidecar proxy configuration to services in the same namespace, and istio-system namespace

Feature Gating - Criteria

- Impact on resource consumption and performance
  - Istio Sidecar custom resource in root namespace (istio-system)
    - Enforce to limit Envoy config size
    - Prevent bypassing on a namespace or workload level



```
violation[{"msg": msg}] {
  is_sidecar(input.review.kind)
  host := spec.egress[_].hosts[_]
  host == "*"
  msg := "Cannot use broad host definition \"*\" in Sidecar egress configuration."
}
```

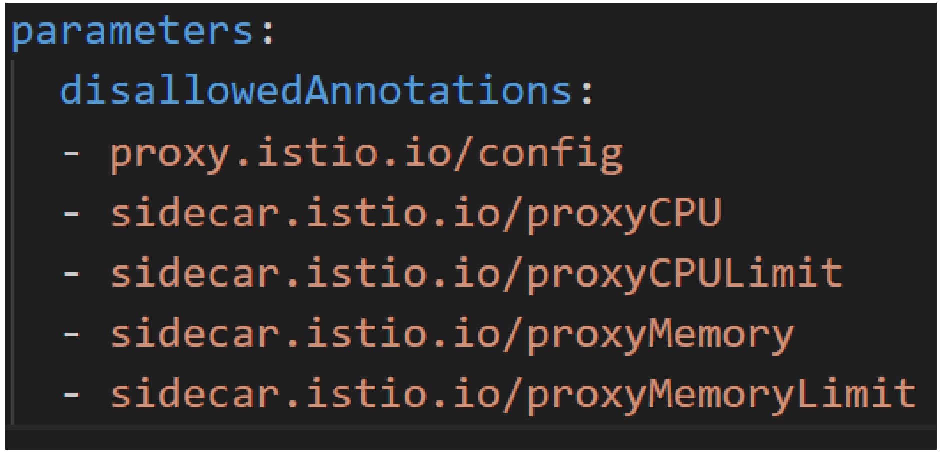
Block other sidecars that override mesh-wide egress config restrictions with a wildcard host

Feature Gating - Criteria

- Impact on resource consumption and performance
  - Envoy and WASM
    - OSM
      - wasmStats + enablePermissiveTrafficPolicy
    - Istio - [WasmPlugin] and [Wasm-Based Telemetry]
      - Significant increase in proxy CPU and memory utilization
      - Alpha and experimental (respectively)

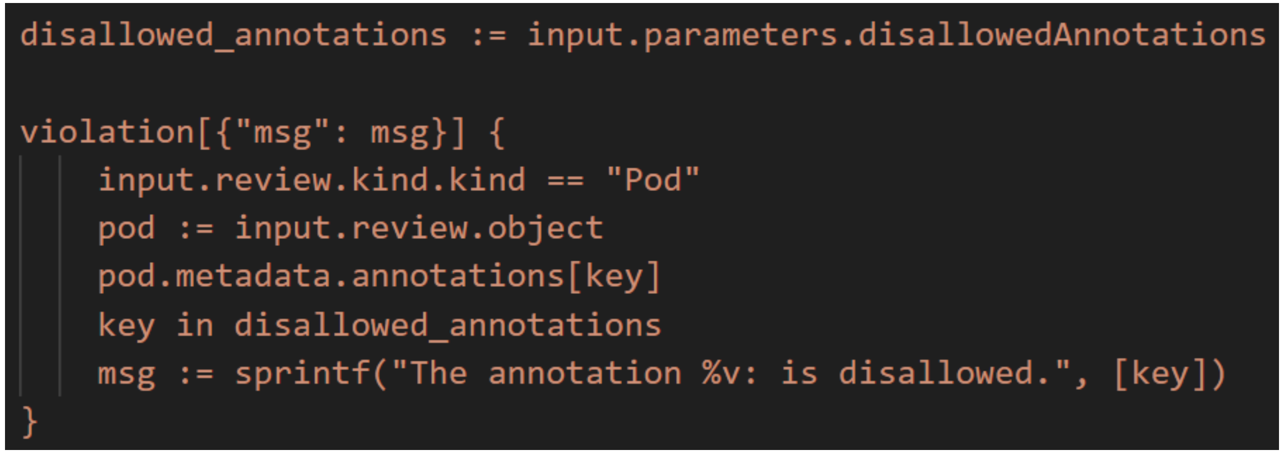
Feature Gating - Criteria

- Impact on resource consumption and performance
  - Prevent bypassing of mesh-wide resource quotas for proxy via annotations



```yaml
parameters:
  disallowedAnnotations:
  - proxy.istio.io/config
  - sidecar.istio.io/proxyCPU
  - sidecar.istio.io/proxyCPULimit
  - sidecar.istio.io/proxyMemory
  - sidecar.istio.io/proxyMemoryLimit
```

Gatekeeper constraint with disallowed annotations



```
disallowed_annotations := input.parameters.disallowedAnnotations

violation[{"msg": msg}] {
    input.review.kind.kind == "Pod"
    pod := input.review.object
    pod.metadata.annotations[key]
    key in disallowed_annotations
    msg := sprintf("The annotation %v: is disallowed.", [key])
}
```

Gatekeeper constraint template to block Pods with annotations

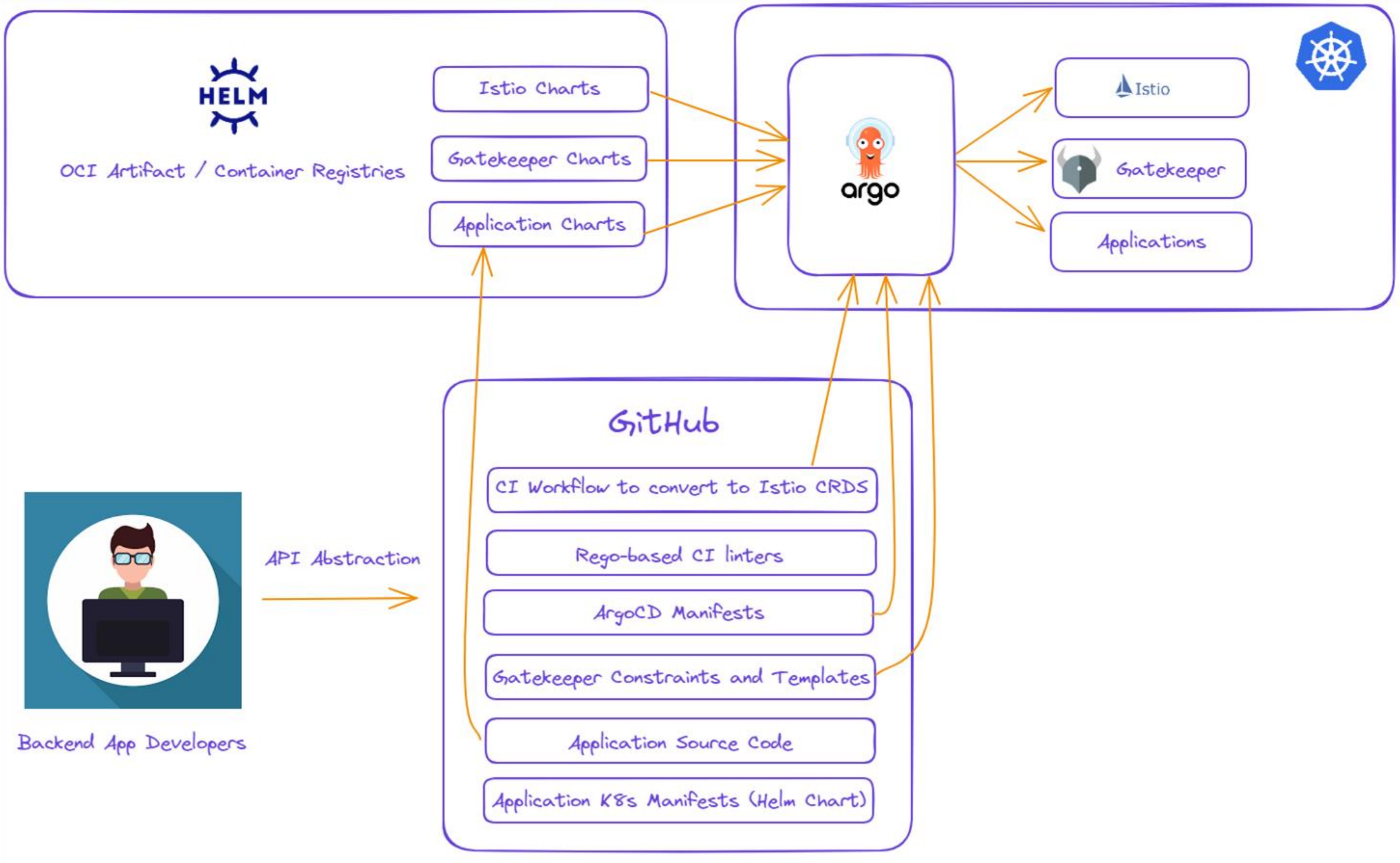
Feature Gating - Criteria

- Organizational requirements and use-cases
  - Focus on core use-cases
  - Iterative adoption is key
    - Start off with small subset of features and configurations
    - Expand allowlists and abstraction layers as you build confidence

Takeaways

- Several tools for feature-gating and limiting configurability
  - Admission controllers and policy enforcement engines
    - And CI linters
  - API abstractions layers
  - GitOps
  - RBAC

Takeaways



(Can also use GitHub repository directly instead of Helm)

https://www.linkedin.com/in/niranjan-shankar-766164149/

https://github.com/nshankar13/tutorials

Takeaways

- Criteria
  - Complexity
  - Feature status
  - Security and risk
  - Resource consumption
  - Organizational requirements

Takeaways

- Still complexity and operational overhead
  - Platform engineers still need to understand features, APIs, configurations
    - And evaluate against criteria
  - Additional tooling
    - Abstractions and new CRDs
    - Gatekeeper, Argo/Flux, etc

Takeaways

- Benefits
  - Untangled, decluttered mesh environment
    - Easier to operate
    - Less error-prone
    - More secure and compliant
  - Better harmony between platform engineers and service owners
    - Declarative enforcement
      - Desired mesh-wide behavior
      - Allowed and disallowed features and configurations
    - Developers can prioritize business logic
  - Tailored to your organization’s specific needs and use-cases

Relevant Developments

- Istio community:
  - Istio Foundational Mode
  - Safer by Default EnvoyFilters
  - Feature maturity and graduation process enhancement
    - Could also improve feature flags in Istio
    - Ex: Telemetry API to Beta
  - Ambient Mesh
  - Preference for “one source of truth”
    - Moving configuration for MeshConfig and complex CRs (EnvoyFilter) to dedicated, first-class APIs.

Helpful Resources

Demos: https://github.com/nshankar13/tutorials/tree/main/istio-feature-gating-demo

Istio:

- IstioCon2023 Feature-Gating Talk: https://www.youtube.com/watch?v=MckZnnq-OGk
- Debunking the "Istio is Complex" meme: https://www.youtube.com/watch?v=v54hITjHdh0&list=PL7wB27eZmdffVBMSaXzXz02mZb3gZWlsi&index=17&pp=iAQB
- Feature status: https://istio.io/latest/docs/releases/feature-stages/
- MeshConfig: https://istio.io/latest/docs/reference/config/istio.mesh.v1alpha1/
- Pilot environment variables: https://github.com/istio/istio/blob/1.19.3/pilot/pkg/features/pilot.go
- Custom resources:
  - Traffic Management: https://istio.io/latest/docs/concepts/traffic-management/
  - PeerAuthentication: https://istio.io/latest/docs/tasks/security/authentication/mtls-migration/
  - Policy Precedence: https://istio.io/latest/docs/tasks/security/authentication/authn-policy/#policy-precedence
  - AuthorizationPolicy: https://istio.io/latest/docs/reference/config/security/authorization-policy/
  - EnvoyFilter: https://istio.io/latest/docs/reference/config/networking/envoy-filter/
  - ProxyConfig: https://istio.io/latest/docs/reference/config/networking/proxy-config/
  - Sidecar: https://istio.io/latest/docs/reference/config/networking/sidecar/
  - WASMplugin: https://istio.io/latest/docs/reference/config/proxy_extensions/wasm-plugin/
  - Telemetry API: https://istio.io/latest/docs/tasks/observability/telemetry/
- Memory consumption: https://medium.com/geekculture/watch-out-for-this-istio-proxy-sidecar-memory-pitfall-8dbd99ea7e9d
- Common misconfigurations:
  - Traffic: https://istio.io/latest/docs/ops/common-problems/network-issues/
  - Security: https://istio.io/latest/docs/ops/common-problems/security-issues
- Resource annotations: https://istio.io/latest/docs/reference/config/annotations/
- Deployment models: https://istio.io/latest/docs/ops/deployment/deployment-models/
- WASM-based telemetry: https://istio.io/latest/docs/reference/config/proxy_extensions/wasm_telemetry/

https://www.linkedin.com/in/niranjan-shankar-766164149/ https://github.com/nshankar13/tutorials

Helpful Resources

Istio:

- API abstractions:
    - Salesforce: https://www.youtube.com/watch?v=eOEO-EGwENQ
    - Airbnb: https://events.istio.io/istiocon-2022/sessions/building-simplified-service-mesh-api-for-developers/
    - Splunk: https://events.istio.io/istiocon-2022/sessions/istio-at-splunk/
    - Intuit: https://www.youtube.com/watch?v=LOhfCrxnghI&list=PL7wB27eZmdffVBMSaXzXz02mZb3gZWlsi&index=14
    - GoPay: https://events.istio.io/istiocon-2022/sessions/improving-rate-limit-experience/
- Relevant developments:
    - Foundational Mode: https://docs.google.com/document/d/1aaORW2Ak4Vfpr-N68Q04qS7iskDdF3v7lcZFQsFA_L0/edit
    - Safer by Default EnvoyFilters: https://docs.google.com/document/d/11oD8Yp1MsES7f73gm1wqxr4v_ZqEiB_bNpck8jBwUII/edit#heading=h.af5dxlqomql
    - Feature Maturity and Enhancements: https://docs.google.com/document/d/101cCNqoHrazFvn-P1O5UYzyvo6LCgEJhhv-__v6C1t8/edit#heading=h.v544s7vgn3bs
    - Telemetry API to Beta: https://docs.google.com/document/d/1oozEgNnlaZqHorCb4oUtlAu3ucxtny-PhqDYyddidPA/edit
    - Ambient Mesh: https://istio.io/latest/blog/2022/introducing-ambient-mesh/
        - Progress to Beta: https://docs.google.com/document/d/1BLIoPlSOyd4E6QmMAwmLY69_Kf381Fp0iwh1OajLLxs/edit

Linkerd:

- Features: https://linkerd.io/2.14/features/
- Authorization Policy: https://linkerd.io/2.14/features/server-policy/
- Annotations and proxy configuration: https://linkerd.io/2.14/reference/proxy-configuration/
- Rust-based proxy vs Envoy: https://linkerd.io/2020/12/03/why-linkerd-doesnt-use-envoy/index.html

Helpful Resources

GitOps:

- ArgoCD:
  - Istio:
    - https://tetrate.io/blog/implementing-gitops-and-canary-deployment-with-argo-project-and-istio/
    - https://www.solo.io/blog/gitops-with-argo-cd-and-gloo-mesh-part-1/
  - Linkerd: https://linkerd.io/2.14/tasks/gitops/
- FluxCD / Flagger:
  - Istio:
    - https://fluxcd.io/flagger/tutorials/istio-progressive-delivery/
    - https://trstringer.com/install-istio-flux/
    - https://kccnceu2022.sched.com/event/ytt9/simplifying-service-mesh-operations-with-flux-and-flagger-mitch-connors-google-stefan-prodan-weaveworks
    - https://ruzickap.github.io/k8s-flagger-istio-flux/
  - Linkerd:
    - https://linkerd.io/2023/05/15/real-world-gitops/
    - https://fluxcd.io/flagger/tutorials/linkerd-progressive-delivery/
- Consul and GitOps: https://itnext.io/helm-3-updating-consul-the-gitops-way-1ee45af8fe4d

Kubernetes:

- Feature flags: https://kubernetes.io/docs/reference/command-line-tools-reference/feature-gates/
- K8s 1.28: https://kubeops.net/blog/whats-inside-kubernetes-v-1-28
- Tim Hockin Proposal: https://docs.google.com/document/d/1roVAHyF7eWZAccmCKw7MXYUNgx4BCDSXF2IMS8h10oY/edit?resourcekey=0-x6Tw2qz1SpCIPhbec6Qa2A#

Helpful Resources

Gatekeeper:

- https://www.youtube.com/watch?v=90RHTBinAFU
- Rego: https://www.openpolicyagent.org/docs/latest/policy-language/
- Gator CLI: https://open-policy-agent.github.io/gatekeeper/website/docs/gator/
    - CI Linting: https://github.com/mathieu-benoit/istio-gatekeeper-demos/tree/main/.github/workflows
- Istio:
    - https://events.istio.io/istiocon-2022/sessions/gatekeeper-istio/
    - https://github.com/mathieu-benoit/istio-gatekeeper-demos
    - https://github.com/crcsmnky/gatekeeper-istio
    - https://www.openpolicyagent.org/docs/latest/envoy-tutorial-istio/
- Linkerd:
    - https://github.com/ihcsim/linkerd-opa
    - https://buoyant.io/media/enforcing-automatic-mtls-linkerd-opa-gatekeeper

Conftest rego CI linter: https://www.blokje5.dev/posts/compliance-in-cicd/

Kyverno:

- Istio: https://kyverno.io/policies/?policytypes=Istio
- Linkerd: https://kyverno.io/policies/?policytypes=Linkerd
- Consul: https://kyverno.io/policies/?policytypes=Consul

Thanks!

https://www.linkedin.com/in/niranjan-shankar-766164149/

https://github.com/nshankar13/tutorials/tree/main/istio-feature-gating-demo

Please scan the QR Code above
to leave feedback on this session