solo.io

Gloo Mesh Management with GitOps

Agenda

- What is GitOps
- ArgoCD Overview
- ArgoCD Demo

solo.io

The What and Why of GitOps?

What is GitOps?

GitOps has evolved from automation practices such as DevOps and Infrastructure as Code (IaC).

GitOps is:

1. **Declarative** - The system under management must have its desired state expressed declaratively.
2. **Versioned & immutable** - The desired state is stored in a source that is immutable, versioned and retrains history.
3. **Pulled automatically** - Controllers automatically pull the declarations on desired state from the source.
4. **Continuously reconcile** - Controllers continuously observe the actual system state and apply the desired state.

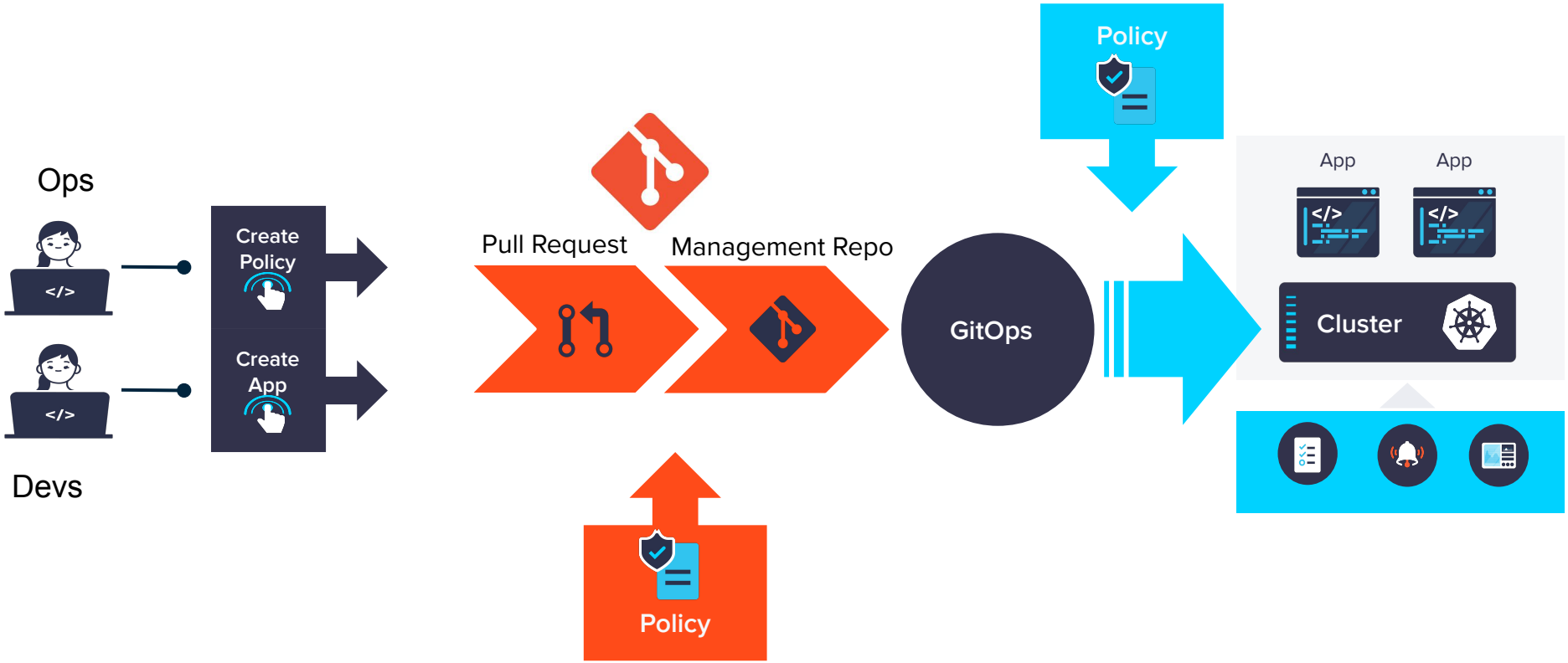
What and Why cont….

Why GitOps?

Organisations that implement GitOps experience many benefits.

- Increased Developer & Operational Productivity
- Enhanced Developer Experience
- Improved Stability
- Higher Reliability
- Consistency and Standardization
- Stronger Security Guarantees

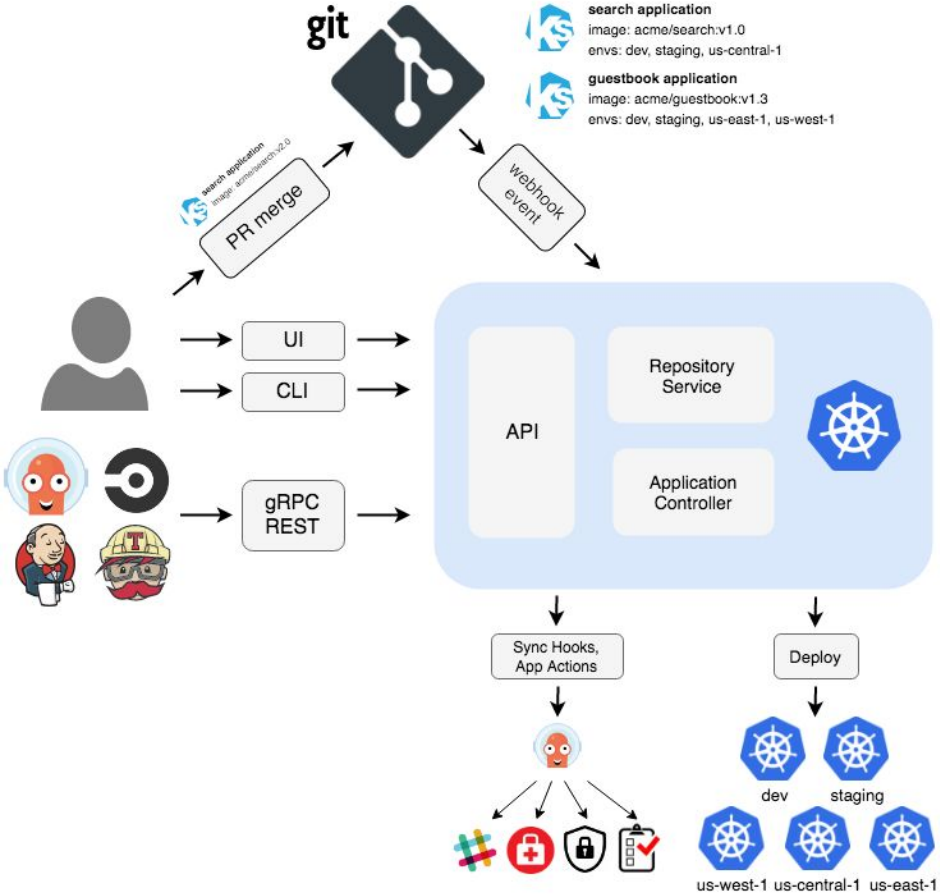
GitOps Benefits



solo.io

Overview of ArgoCD

ArgoCD Architecture



solo.io

ArgoCD Resources - Projects

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: my-project
  namespace: argocd
spec:
  description: Example Project
  sourceRepos:
  - '*'
  destinations:
  - namespace: guestbook
    server: https://kubernetes.default.svc
  roles:
  - name: read-only
    description: Read-only privileges to my-project
    policies:
    - p, proj:my-project:read-only, applications, get, my-project/*,
allow
    groups:
    - my-oidc-group
```

- Project - A grouping of Applications that share a logical responsibility.
  - Projects can control:
    - which Argo users can create/edit/delete Applications
    - which Repositories an Application can utilize
    - which Clusters (and namespaces within a cluster) Applications can target
    - which Kubernetes resources Applications can apply

ArgoCD Resources - Repositories

```
apiVersion: v1
kind: Secret
metadata:
  name: private-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/argoproj/private-repo
  password: my-password
  username: my-username
```

- Repository - A Secret that holds access credentials for a git (or Helm) repository where the manifests for an Application can be accessed.

ArgoCD Resources - Applications

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    server: https://kubernetes.default.svc
    namespace: guestbook
```

- Application - Defines the set of manifests that will be deployed and managed together as an atomic object.
  - The Application is not considered “in-sync” unless all instances of these manifests in the cluster (the *live state*) match the state in their repository (the *target state*).
  - An Application can deploy more Applications, allowing for a tree-like “app of apps” structure.

ArgoCD Resources - Clusters

```
apiVersion: v1
kind: Secret
metadata:
  name: mycluster-secret
  labels:
    argocd.argoproj.io/secret-type: cluster
type: Opaque
stringData:
  name: mycluster.com
  server: https://mycluster.com
  config: |
    {
      "bearerToken": "<authentication token>",
      "tlsClientConfig": {
        "insecure": false,
        "caData": "<base64 encoded certificate>"
      }
    }
```

- Cluster - A Secret that references a Kubernetes cluster that can be targeted by an Application
  - Argo requires a ServiceAccount with specific privileges on the cluster in order to manage Applications. It can create this automatically on cluster addition or you can give it credentials for a pre-created SA.

ArgoCD Deployment Considerations

solo.io

One Management ArgoCD to Many Clusters

Benefits and tradeoffs include:

- A centralized ArgoCD instance cuts down on repeated deployments of Argo and allows for visibility of deployed Applications across all clusters.
- More strict RBAC and governance configurations
- Strict usage of Argo project configurations and whitelisting of resources to be allowed to sync to what namespaces on what clusters
- HA concerns with ArgoCD availability
  - Using HPA and multiple replicas across many nodes on management cluster

ArgoCD Instance Installed on Each Cluster

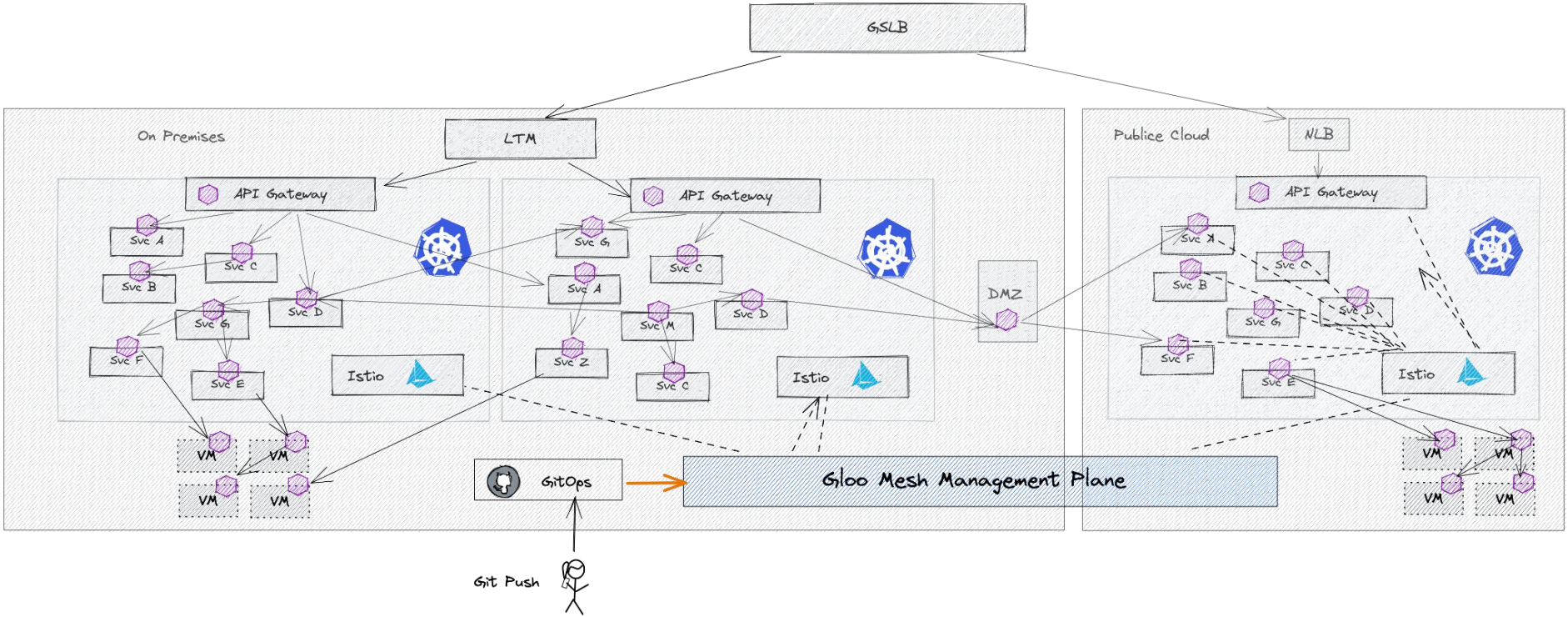
Benefits and tradeoffs include:

- ArgoCD used as an agent on each cluster to be responsible for the resources on its local cluster
- Each ArgoCD instance shares a common configuration
  - One helm chart configuration deployed over multiple clusters
- Less complicated configuration of RBAC and project restrictions
  - Overall less complicated configuration of ArgoCD itself
  - Less management of each individual ArgoCD instance
- Possibly many ArgoCD instances to be aware of

Demo
ArgoCD & Gloo Mesh

solo.io

Drive everything through GitOps!



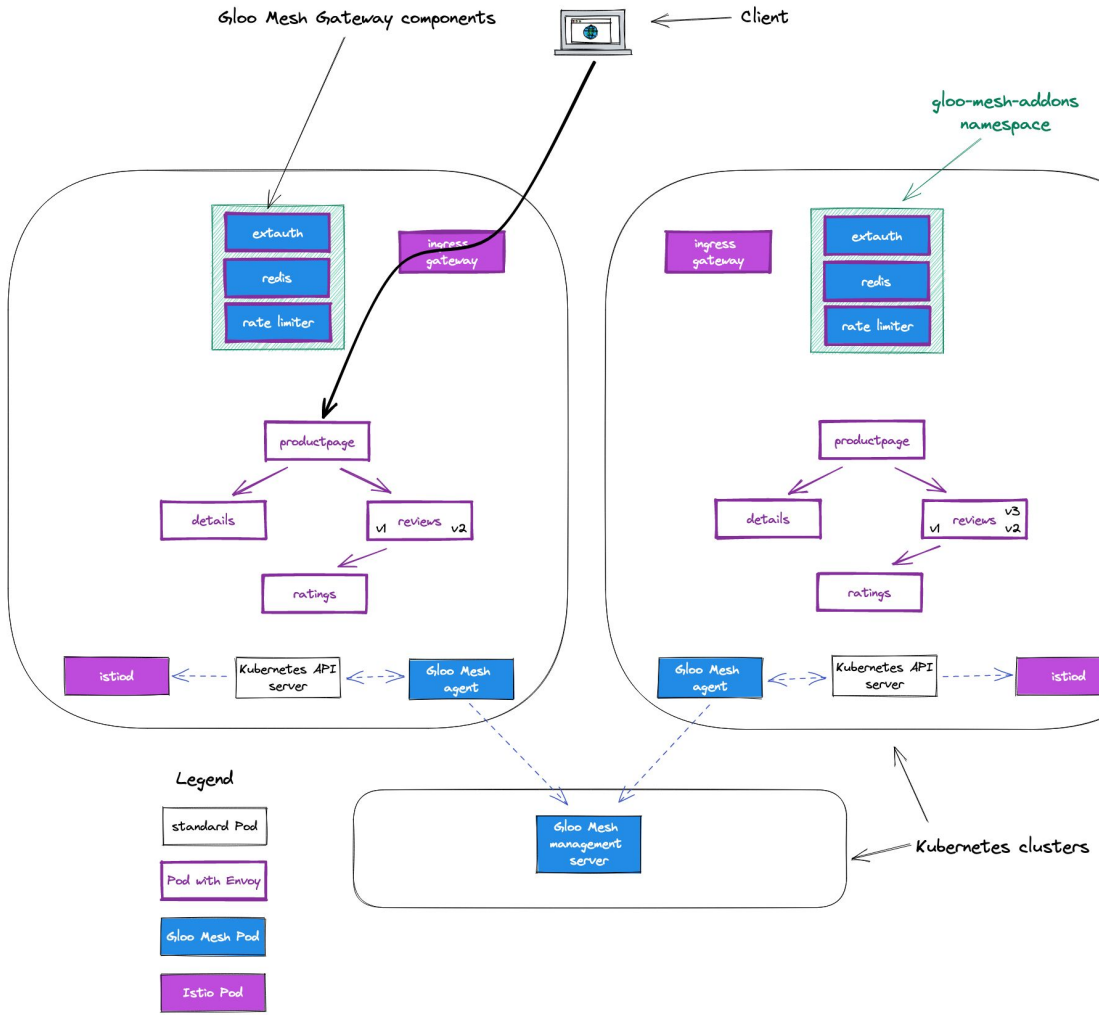
solo.io

Demo

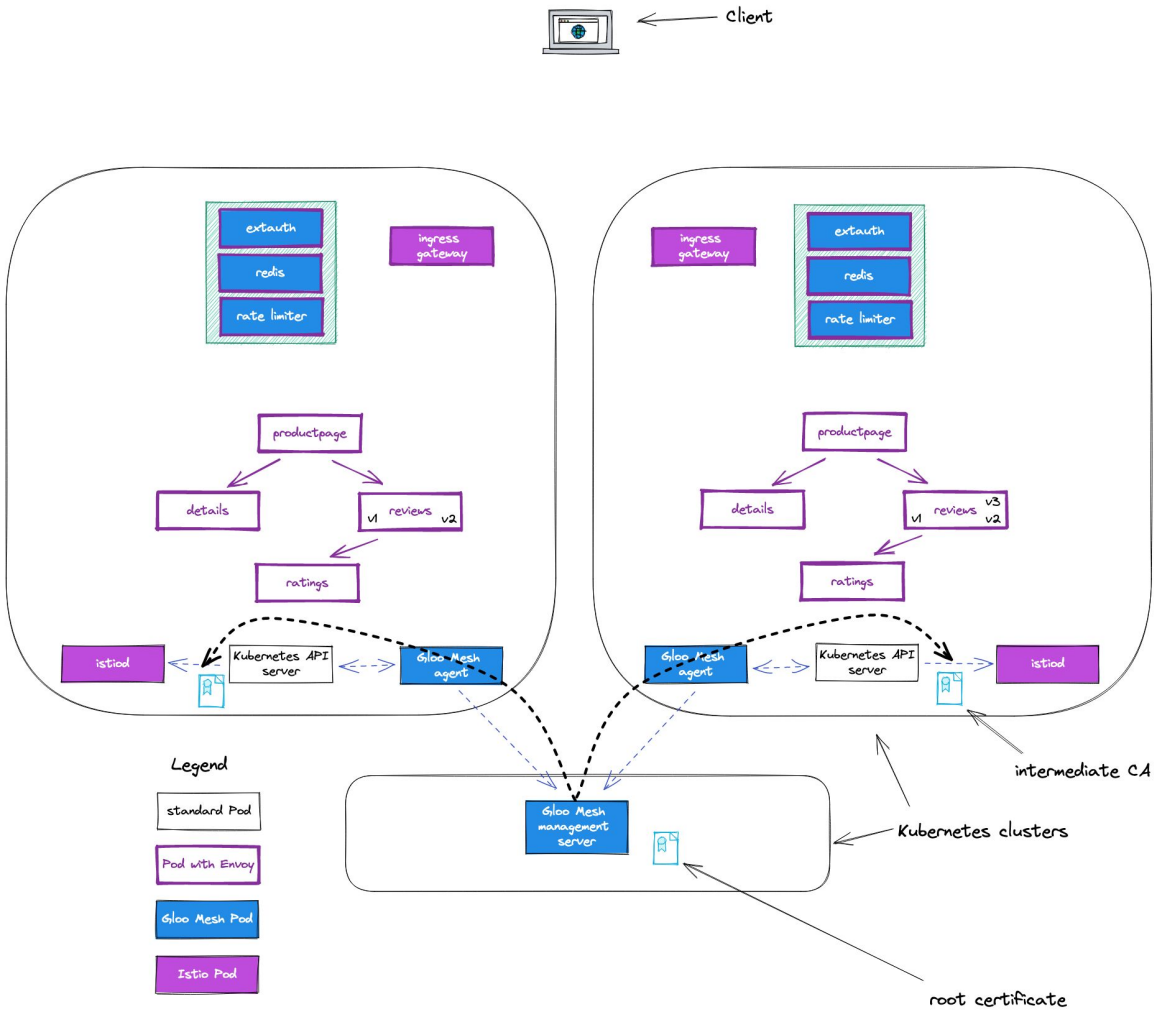
1. GitOps Gloo Mesh Configuration

    a. Enable Zero trust
        i. Enable Istio TLS
        ii. Enable Zero Trust with shared root trust

    b. Enable Traffic Policies
        i. Shift 75% of “reviews” traffic to cluster2

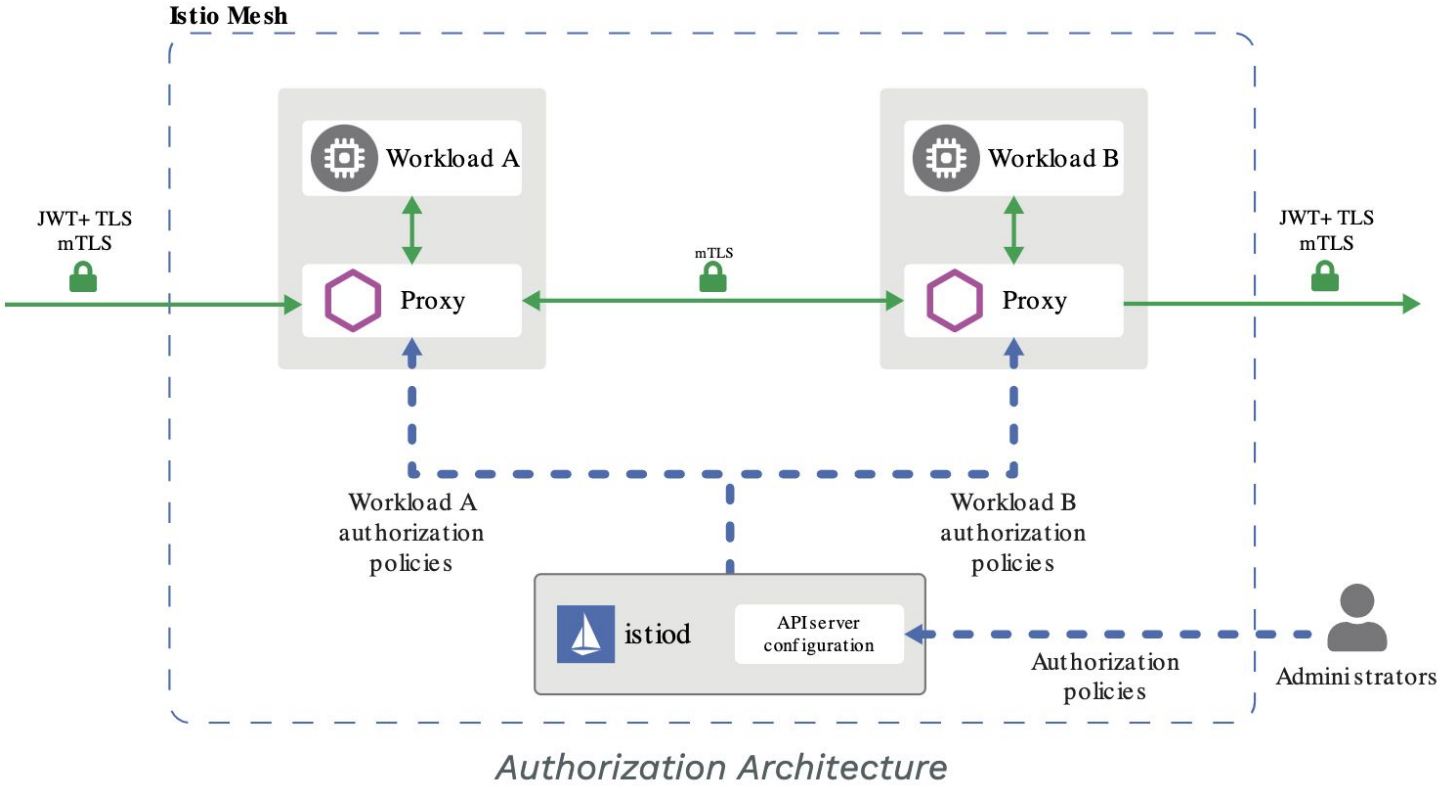

Gloo Mesh Gateway components
Client
gloo-mesh-addons namespace
extauth
redis
rate limiter
ingress gateway
productpage
details
v1
reviews
v2
ratings
istiod
Kubernetes API server
Gloo Mesh agent
ingress gateway
extauth
redis
rate limiter
productpage
details
v1
reviews
v3
v2
ratings
Gloo Mesh agent
Kubernetes API server
istiod
Legend
standard Pod
Pod with Envoy
Gloo Mesh Pod
Istio Pod
Gloo Mesh management server
Kubernetes clusters

solo.io


Client
extauth
redis
rate limiter
ingress gateway
productpage
details
reviews
v1
v2
ratings
istiod
Kubernetes API server
Gloo Mesh agent
ingress gateway
extauth
redis
rate limiter
productpage
details
reviews
v1
v2
v3
ratings
Gloo Mesh agent
Kubernetes API server
istiod
intermediate CA
Legend
standard Pod
Pod with Envoy
Gloo Mesh Pod
Istio Pod
Gloo Mesh management server
Kubernetes clusters
root certificate

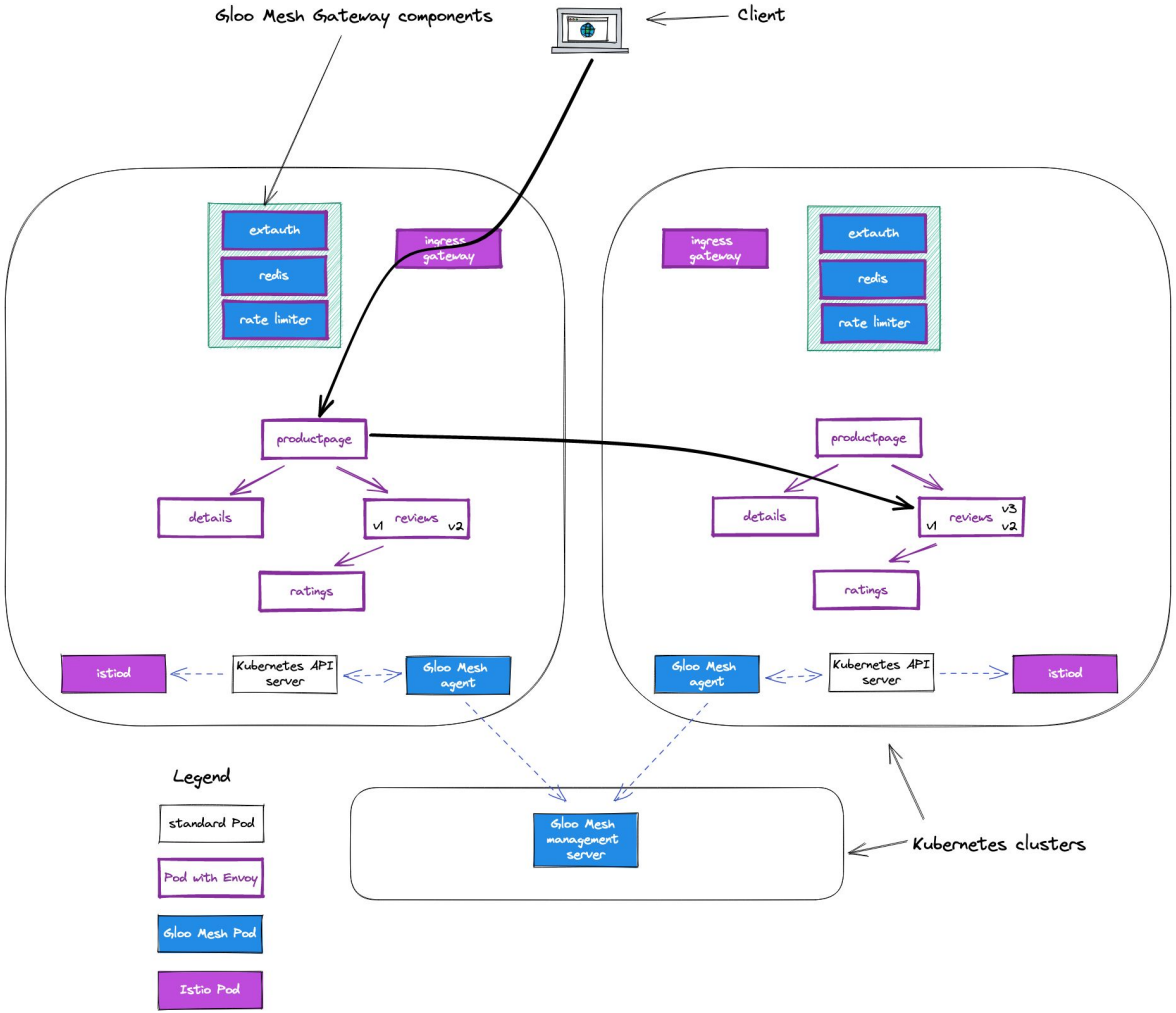
solo.io

Istio Authorization



Authorization Architecture

solo.io


Gloo Mesh Gateway components
client
extauth
redis
rate limiter
ingress gateway
productpage
details
v1
reviews
v2
ratings
istiod
Kubernetes API server
Gloo Mesh agent
ingress gateway
extauth
redis
rate limiter
productpage
details
v1
reviews
v3
v2
ratings
Gloo Mesh agent
Kubernetes API server
istiod
Legend
standard Pod
Pod with Envoy
Gloo Mesh Pod
Istio Pod
Gloo Mesh management server
Kubernetes clusters

solo.io

Thank You