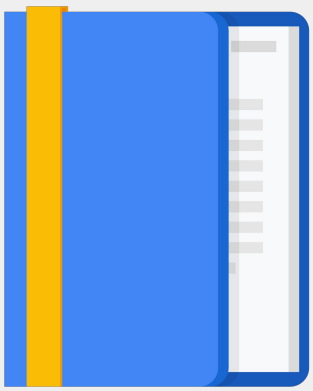
Architecting Hybrid Infrastructure with Anthos

Managing Policies using Anthos Config Management

Agenda



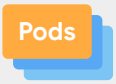
- **Config Across Clusters**
- Anthos Configuration Management
- Adopting ACM

Google Cloud

Cluster per Tenant

Tenant (Blue Team)

K8s API

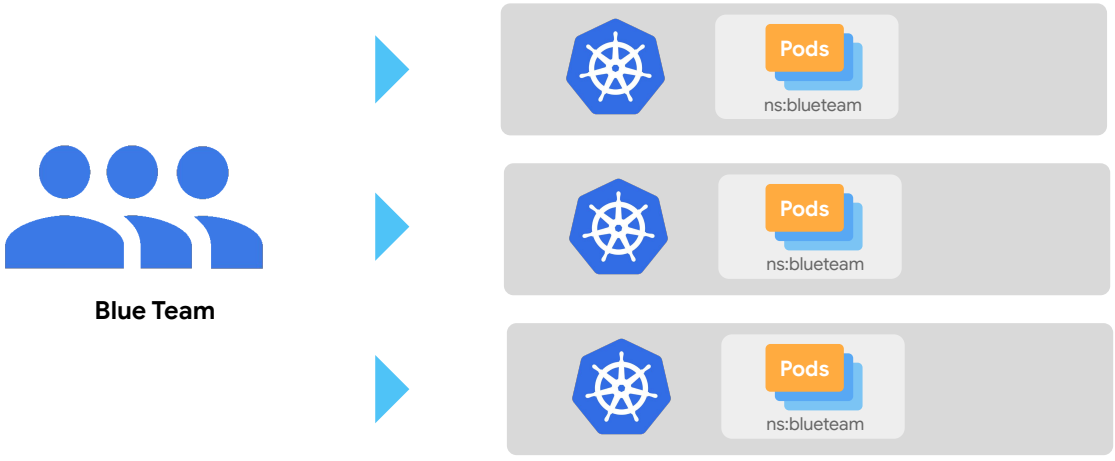


Default Namespace

Cluster

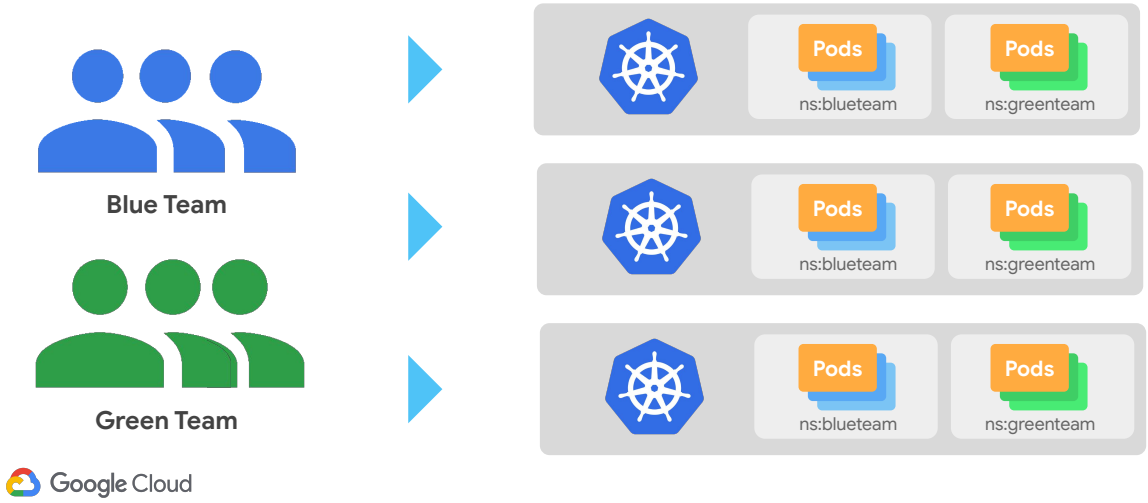
Google Cloud

One Tenant, Many Clusters



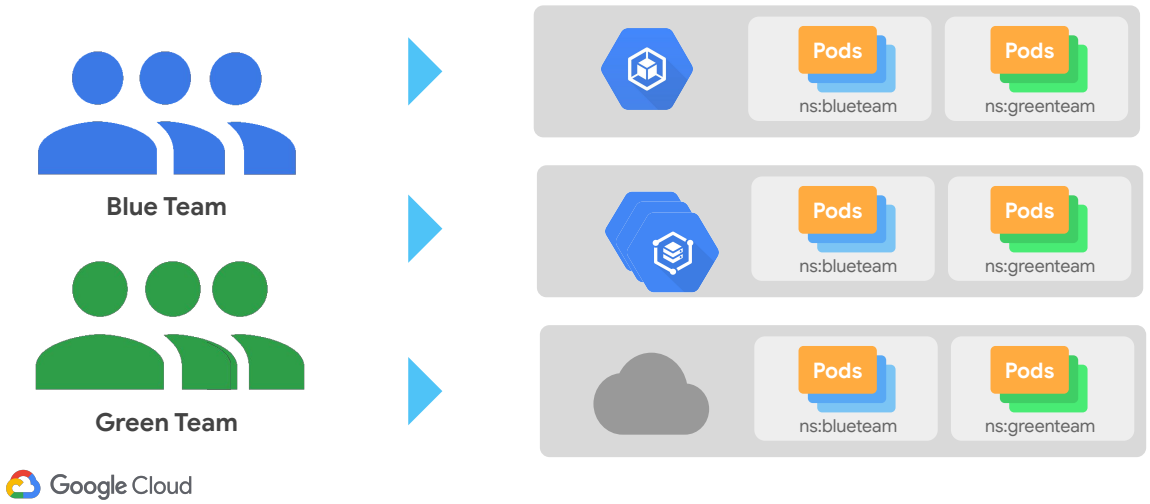
Google Cloud

Multi-cluster, Multi-tenant



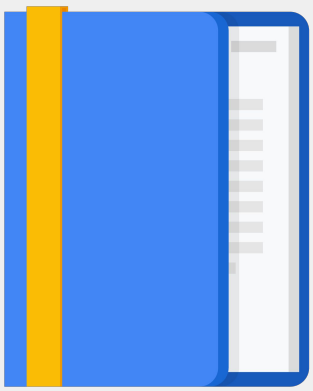
Google Cloud

Multi-cluster, Multi-tenant, Multi-Environment



Google Cloud

Agenda



- Config Across Clusters
- **Anthos Configuration Management**
- Adopting ACM

Google Cloud

Anthos Config Management

A single pane of glass for managing configurations and policies both on-prem and in the cloud

Multi-cluster management

Central Git repository manages access control policies, resource quotas, and namespaces

Hybrid support

Manages both on-prem and in the cloud; change central file and apply across the fleet

Declarative and continuous

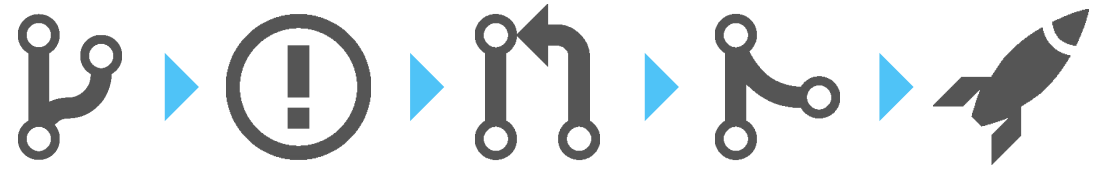
Declare new desired state, continuously checks for changes that go against state

Simple migration

Uses YAML or JSON, so no rewriting of existing Kubernetes configs

Google Cloud

Configuration as code



Branch Validate Review Merge Deploy

Google Cloud

Git as a Source of truth

- Handles all K8s/Istio resources
- Integrates with customers' on-prem source control
- Configuration is expressed in native formats (YAML)
- Validators plug easily into CD pipelines

```
/git-policy-repo$ tree
├── CODEOWNERS
├── foo-corp
│   ├── audit
│   │   └── namespace.yaml
│   ├── namespace-reader-clusterrolebinding.yaml
│   ├── namespace-reader-clusterrole.yaml
│   ├── online
│   │   └── shipping-app-backend
│   │       ├── pod-creator-rolebinding.yaml
│   │       ├── quota.yaml
│   │       ├── shipping-dev
│   │       │   ├── job-creator-rolebinding.yaml
│   │       │   ├── job-creator-role.yaml
│   │       │   ├── namespace.yaml
│   │       │   └── quota.yaml
│   │       └── shipping-prod
│   │           └── namespace.yaml
```

Google Cloud

Git as a Source of truth

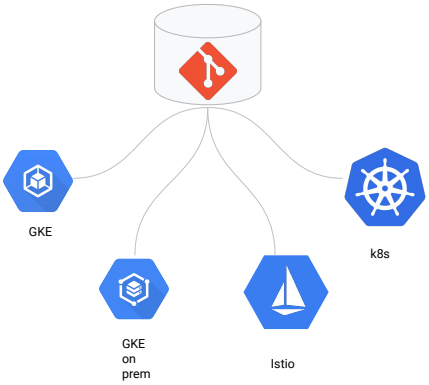
- Hierarchical policy enforcement
  - Blank NetworkPolicy into every namespace from the root
  - Set an SRE rolebinding that gets inherited across all namespaces
  - Apply pod security policy across namespaces

Configuration as Code

Immutable: A Git commit is an exact declaration of the desired state of policies.

Auditable: Changes are reviewed and approved by administrators.

Revertable: Misconfiguration is one of the most common reasons for service outages. It should be fast and easy to revert to a known good state.



Google Cloud

Multi-cluster, Multi-tenant, Multi-Environment

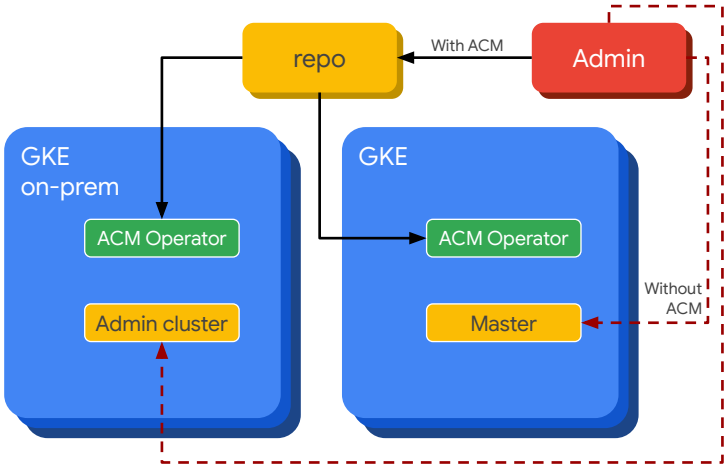
Admins declares desired state of master configuration outside of the cluster

↓

Desired configuration is pulled and applied across all clusters via ACM operator

↓

ACM continuously watches for any changes to managed configurations reconcile it according to the desired configuration

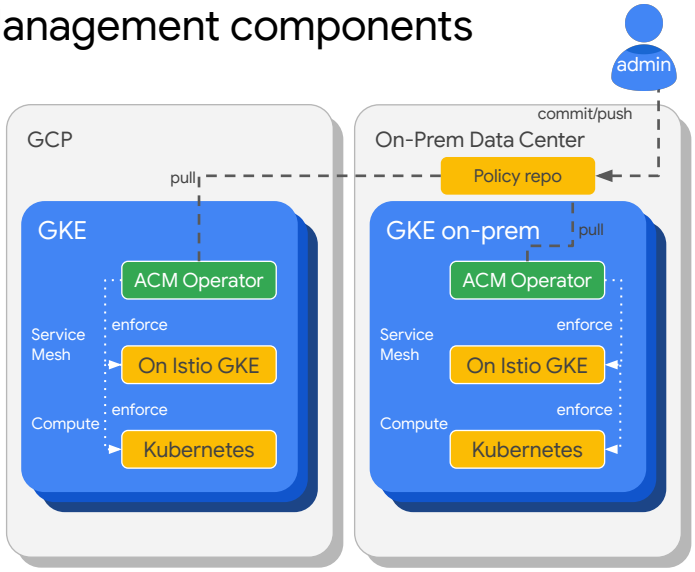


Google Cloud

Anthos Config Management components

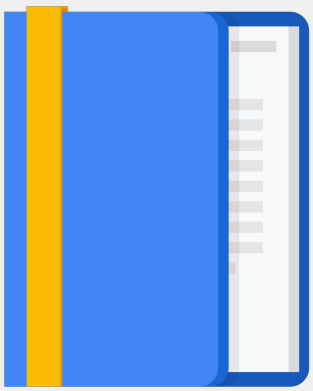
Git Repository is a single source of truth

Top down enforcement, lower level modification will be overridden



Google Cloud

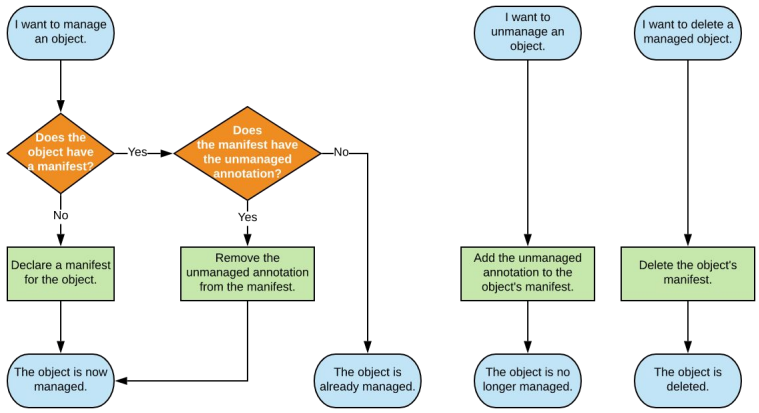
Agenda



- Config Across Clusters
- Anthos Configuration Management
- **Adopting ACM**

Google Cloud

Object Lifecycle



Google Cloud

Adopting Anthos Configuration Management

An object in a cluster is managed by Anthos Config Management if it has the annotation `configmanagement.gke.io/managed: enabled`

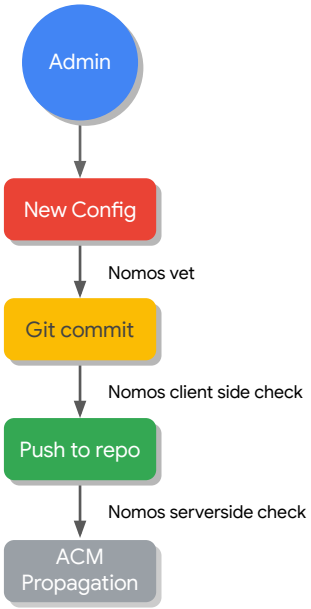
If an object does not have the label `configmanagement.gke.io/managed` at all, or if it set to anything other than enabled, the object is unmanaged

```yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: job-creators
subjects:
- kind: User
  name: sam@foo-corp.com
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: job-creator
  apiGroup: rbac.authorization.k8s.io
annotations:
  configmanagement.gke.io/managed: disabled
```

Using nomos CLI

Configuration validation

- Automatically checking for syntax errors when committing `nomos vet`
- Client or server hook support
- Will not apply any corrupt config already in repository



Google Cloud

Using nomos

Check the installation status of clusters using `nomos status`

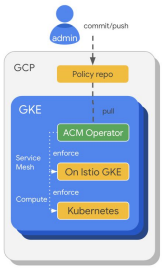
```
Context             Status           Last Synced Token
-------             ------           -----------------
managed-cluster-1   NOT INSTALLED
managed-cluster-2   NOT CONFIGURED
managed-cluster-3   SYNCED           f52a11e4

Config Management Errors:
managed-cluster-2   missing git-creds Secret
```

Lab

Managing Policies in Kubernetes Engine using Anthos Config Management

60 min



Objectives

- Install the Config Management Operator
- Set up your ACM config repo in Cloud Source Repositories (CSR)
- Connect your GKE cluster to the CSR config repo
- Examine the configs in your cluster and repo
- Review automated drift management
- Update a config in the repo