Introducing environs

Environs are a Google Cloud concept for logically organizing clusters and other resources, letting you use and manage multi-cluster capabilities and apply consistent policies across your systems. Environs form a crucial part of how enterprise multi-cluster functionality works in Anthos. They are also used in some Google Kubernetes Engine (GKE) features.

This guide introduces you to environs: what we mean by an environ, where environs are used in our components, and how to set up your systems to take advantage of environ-level features. We also provide some examples (/anthos/multicluster-management/environs/examples) to illustrate how environs can help simplify your cluster and system management, and best practices (/anthos/multicluster-management/environs/best-practices) to follow when building and operating multi-cluster systems with environs.

This guide is designed for technical readers, including system architects, platform operators, and service operators, who want to leverage multiple clusters and related infrastructure. These concepts are useful wherever your organization happens to be running multiple clusters, whether in Google Cloud, across multiple cloud providers, or on-premises.

You should be familiar with basic Kubernetes concepts such as clusters; if you're not, see Kubernetes basics (https://kubernetes.io/docs/tutorials/kubernetes-basics/), the GKE documentation (/kubernetes-engine/docs/concepts/kubernetes-engine-overview), and Preparing an application for Anthos Service Mesh (/service-mesh/docs/prepare-app-for-asm).

If you want to learn more about Anthos and the components that use environs, see our Anthos technical overview (/anthos/docs/concepts/overview) and Explore Anthos (/anthos/docs/tutorials/explore-anthos) tutorial. However, you don't need to be familiar with Anthos to follow this guide.

Introduction

Typically, as organizations embrace cloud-native technologies like containers, container orchestration, and service meshes, they reach a point where running a single cluster is no longer sufficient. There are a variety of reasons why organizations choose to deploy multiple clusters to achieve their technical and business objectives; for example, separating production from non-production environments, or separating services across tiers, locales, or teams. You can

read more about the benefits and tradeoffs involved in multi-cluster approaches in multi-cluster use cases (/anthos/multicluster-management/use-cases).

As the number of clusters grows, providing management and governance over these clusters and the resources inside them becomes increasingly difficult. Often at this point, organizations resort to building custom tooling and operational policies to obtain the level of control that they require.

Google Cloud provides the *environ* concept to help administrators manage multiple clusters. An environ provides a way to logically group and normalize clusters, making administration of infrastructure easier. Environs can be used in the context of both Anthos and GKE; you can see a list of the Anthos and GKE components that can leverage environs in the environ-enabled components (#environ-enabled-components) section later in this document.

Adopting environs helps your organization uplevel management from individual clusters to entire groups of clusters. Furthermore, the normalization that environs require can help your teams adopt similar best practices to those used at Google. For comparison, just as the Organization resource is the root node of the Google Cloud resource hierarchy (/resource-manager/docs/cloud-platform-resource-hierarchy) and is used for policy and control over resources grouped under it, the environ forms the root for managing multiple clusters.

Terminology

The following are some important terms we use when talking about environs.

Environ-aware resources

Environ-aware resources are Google Cloud project resources that can be logically grouped and managed as environs. Only Kubernetes clusters can currently be environ members, although we envisage virtual machine (VM) instances and possibly other resources being able to join environs in future platform iterations. Google Cloud provides a Connect (/anthos/multicluster-management/connect/overview) service to register resources as environ members.

Environ host project

The implementation of environs, like many other Google Cloud resources, is rooted in a Google Cloud project, which we refer to as the environ host project. A given Cloud project can only have a *single environ* (or no environs) associated with it. This restriction reinforces using Cloud projects to provide stronger isolation between resources that are not governed or consumed together.

Environ-enabled components

The following Anthos and GKE components all leverage environ concepts such as namespace and identity sameness to provide a simplified way to work with your clusters and services. For any current requirements or limitations for using environs with each component, see the component requirements (/anthos/multicluster-management/environs/best-practices#component_requirements).

- **Workload identity pools (Anthos and GKE clusters)**
  Within an environ, a common workload identity pool can be used to allow services to be easily authenticated and authorized within a service mesh and to external services.
- **Anthos Service Mesh (Anthos)**
  Anthos Service Mesh is a suite of tools that helps you monitor and manage a reliable service mesh (/service-mesh/docs/overview) on Google Cloud or on-premises. You can form a service mesh across the resources (such as clusters and VMs) that are part of the same environ.
- **Anthos Config Management (Anthos) and Config Sync (GKE)**
  Anthos Config Management lets you deploy and monitor declarative policy and configuration changes for your system stored in a central Git repository, leveraging core Kubernetes concepts such as namespaces, labels, and annotations. With Anthos Config Management (and its sibling product Config Sync for non-Anthos GKE clusters), policy and configuration is defined across the environ, but applied and enforced locally in each of the member resources.
- **Ingress for Anthos (Anthos)**
  Ingress for Anthos uses the environ to define the set of clusters and service endpoints that traffic can be load balanced over, enabling low-latency and high-availability services.

Grouping infrastructure

The first important concept of environs is the concept of *grouping*—that is, choosing which pieces of *related* environ-aware resources (#environ-aware-resources) should be made part of an environ. The decision about what to group together requires answering the following questions:

- Are the resources related to one another?
    - Resources that have large amounts of cross-service communication benefit the most from being managed together in an environ.
    - Resources in the same deployment environment (for example, your production environment) should be managed together in an environ.
- Who administers the resources?
    - Having unified (or at least mutually trusted) control over the resources is crucial to ensuring the integrity of the environ.

To illustrate this point, consider an organization that has multiple lines of business (LOBs). In this case, services rarely communicate across LOB boundaries, services in different LOBs are managed differently (for example, upgrade cycles differ between LOBs), and they might even have a different set of administrators for each LOB. In this case, it might make sense to have environs per LOB. Each LOB also likely adopts multiple environs to separate their production and non-production services.

As other environ concepts are explored in the following sections, you might find other reasons to create multiple environs as you consider your specific organizational needs.

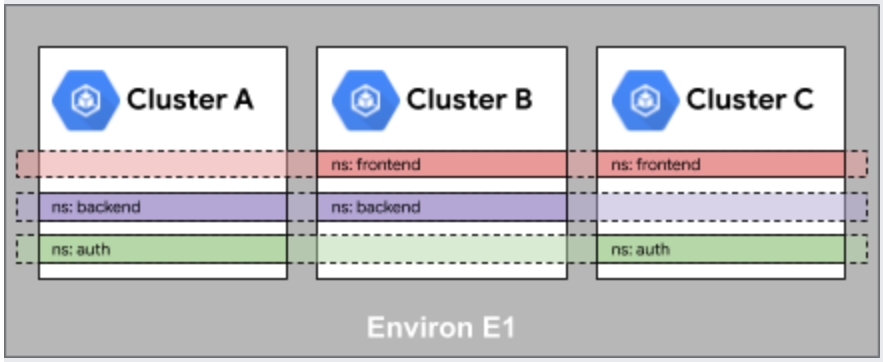
Sameness

An important concept in environs is the concept of sameness. This means that some Kubernetes objects such as clusters with the same name in different contexts are treated as the same thing. This normalization is done to make administering environ resources more tractable. It provides some strong guidance about how to set up namespaces, services, and identities. However, it also follows what we find most organizations already implementing themselves.

Namespace sameness

The fundamental example of sameness in an environ is namespace sameness. Namespaces with the same name in different clusters are considered the same by many components. Another way to think about this property is that a namespace is logically defined across an entire environ, even if the instantiation of the namespace exists only in a subset of the environ resources.

Consider the following `backend` namespace example. Although the namespace is instantiated only in Clusters A and B, it is implicitly reserved in Cluster C (it allows the `backend` service to also be scheduled into Cluster C if necessary). This means that namespaces are allocated for the entire environ and not per cluster. As such, namespace sameness requires consistent namespace ownership across the environ.



(/anthos/multicluster-management/images/namespace-sameness.png)

Namespace sameness in an environ

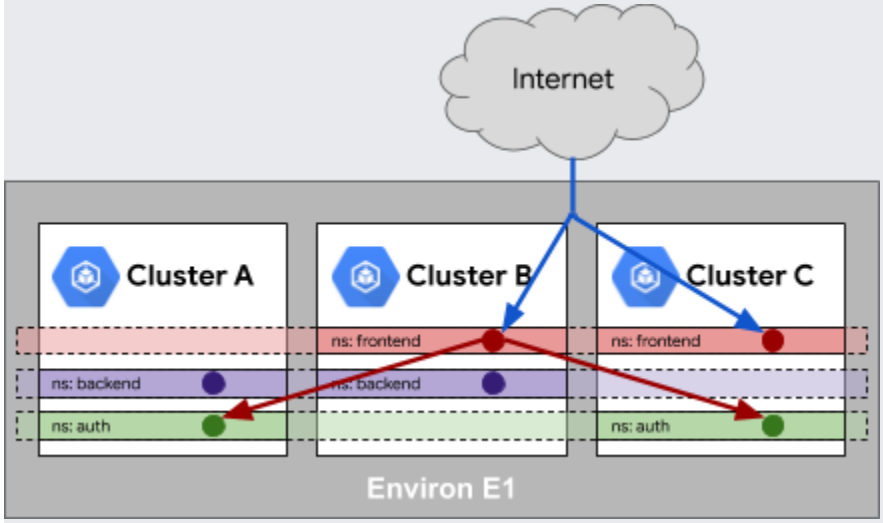
Service sameness

Anthos Service Mesh and Ingress for Anthos use the concept of sameness of services within a namespace. Like namespace sameness, this implies that services with the same namespace and service name are considered to be the same service.

The service endpoints can be merged across the mesh in the case of Anthos Service Mesh. With Ingress for Anthos, a MultiClusterService (MCS) resource (/kubernetes-engine/docs/concepts/ingress-for-anthos#multiclusterservice_resources) makes the endpoint merging more explicit; however, we recommend similar practices with respect to naming. Because of this, it's important to ensure that identically named service names within the same namespace are actually the same thing.

In the following example, internet traffic is load balanced across a same-named service in the `frontend` namespace present in both Clusters B and C. Similarly, using the service mesh

properties within the environ, the `frontend` service can reach a same-named service in the `auth` namespace present in Clusters A and C.



(/anthos/multicluster-management/images/service-sameness.png)

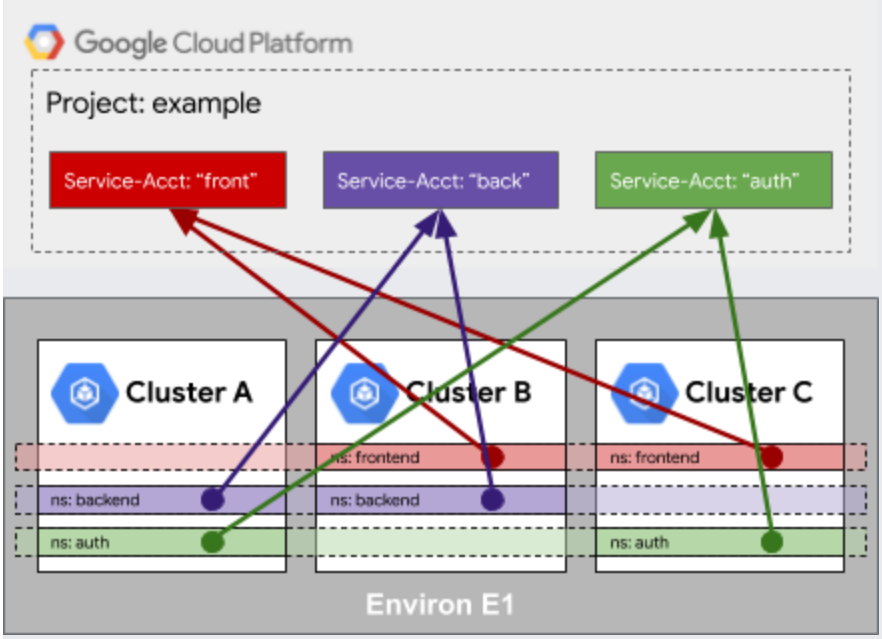
Service sameness in an environ

Identity sameness when accessing external resources

Services within an environ can leverage a common identity as they egress to access external resources such as Google Cloud services, object stores, and so on. This common identity makes it possible to give the services within an environ access to an external resource once rather than cluster-by-cluster.

To illustrate this point further, consider the following example. Clusters A, B, and C are enrolled in common identity within their environ. When services in the **backend** namespace access Google Cloud resources, their identities are mapped to a common Google Cloud service account called **back**. The Google Cloud service account **back** can be authorized on any number of managed services, from Cloud Storage to Cloud SQL. As new environ resources such as clusters are added in the **backend** namespace, they automatically inherit the workload identity sameness properties.

Because of identity sameness, it is important that all resources in an environ are trusted and well-governed. Revisiting the previous example, if Cluster C is owned by a separate, untrusted team, they too can create a **backend** namespace and access managed services as if they were the **backend** in Cluster A or B.



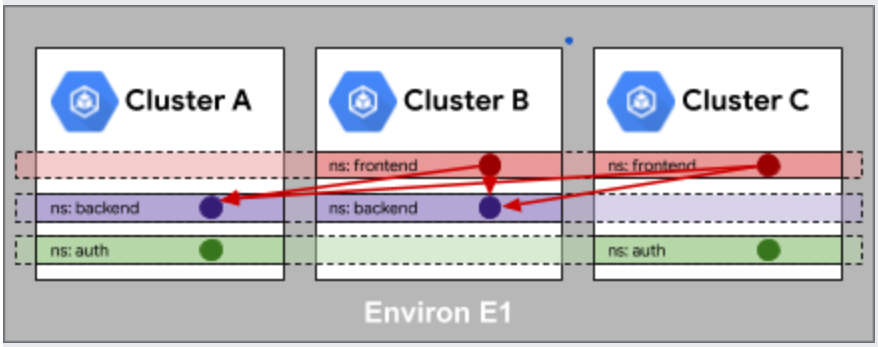
(/anthos/multicluster-management/images/identity-external.png)

Identity sameness accessing resources outside an environ

Identity sameness within an environ

Within the environ, identity sameness is used similarly to the external identity sameness we previously discussed. Just as environ services are authorized once for an external service, they can be authorized internally as well.

In the following example, we have given `frontend` access to `backend`. With environs, we don't need to specify that `frontend` in clusters B and C can access `backend` in Clusters A and B. Instead, we just specify that `frontend` in the environ can access `backend` in the environ. This property not only makes authorization simpler, it also makes the resource boundaries more flexible; now workloads can easily be moved from cluster to cluster without affecting how they are authorized. As with workload identity sameness, governance over the environ resources is crucial to ensuring the integrity of service-to-service communication.



(/anthos/multicluster-management/images/identity-internal.png)

Identity sameness inside an environ

Exclusivity

Environ-aware resources can only be members of a single environ at any given time, a restriction that is enforced by Google Cloud tools and components. This restriction ensures that there is only one source of truth governing a cluster. Without exclusivity, even the most simple components would become complex to use, requiring your organization to reason about and configure how multiple components from multiple environs would interact.

High trust

Service sameness, workload identity sameness, and mesh identity sameness are built on top of a principle of high trust between members of an environ. This trust makes it possible to uplevel management of these resources to the environ, rather than managing resource-by-resource (that is, cluster-by-cluster for Kubernetes resources), and ultimately makes the cluster boundary less important.

Put another way, within an environ, clusters provide protection from blast radius concerns, availability (of both the control plane and underlying infrastructure), noisy neighbors, and so on. However, they are not a strong isolation boundary for policy and governance because administrators of any member in an environ can potentially affect the operations of services in other members of the environ.

For this reason, we recommend that resources that are not trusted by the environ administrator be placed in their own environs to keep them isolated. Then, as necessary, individual services can be authorized across the environ boundary.

What's next?

Ready to think about applying these concepts to your own systems? See our environ requirements and best practices (/anthos/multicluster-management/environs/best-practices).

Last updated 2020-06-22 UTC.