



AKKA AUTOMATED OPERATIONS

BYOC

Bring Your Own Cloud: shared setup, AWS, Azure, GCP, and data flows.

DOC.AKKA.IO · AKKA

Under Bring Your Own Cloud (BYOC), Akka provisions and operates a region inside your own cloud account. You prepare the account with the `akka-bootstrap` utility and grant Akka a deployment identity; Akka then provisions the region, runs smoke tests, and hands it over.

This page lists the cloud-agnostic setup. For your cloud's resource inventory, private connectivity, and `values.hcl` specifics, follow the per-cloud page: [AWS](#), [Azure](#), or [GCP](#).

Installation flow

1. **Understand:** review the [technical overview](#) and receive the `akka-bootstrap` utility from your Akka Success Team, so you can inspect exactly which permissions Akka's deployment identity will require.
2. **Request a region:** submit a region request through the form in the Akka customer portal.
3. **Cloud account:** create a cloud account for Akka. Not strictly required, but a dedicated account simplifies cost management.
4. **Apply permissions:** run `akka-bootstrap` to configure the account, then attach the generated deployment-identity details to your region request.
5. **Provision:** Akka remotely provisions the region and attaches it to a platform organization created for you.
6. **DNS:** create record sets in your DNS zone that map to the region's load-balancer IP.
7. **Smoke tests:** Akka attaches the region to the Federation Plane and runs smoke tests.
8. **Region handover:** register a user at akka.io with a corporate email; Akka makes them your organization's primary owner. A production-labeled region is not fully handed over until you complete the [region-readiness steps](#).

The akka-bootstrap utility

`akka-bootstrap` prepares your cloud account for AAO by creating the Deployment Identity role the Federation Plane uses to create and manage infrastructure. It creates and manages the Deployment Identity operations role only. You create the Editor, Viewer, and SRE identities.

It is a Terragrunt project with a per-cloud module layout:

```
.
├─ main.tftpl      # generates provider config + module calls
├─ modules
│   ├── aws/       # IAM roles, trust policy
│   ├── azure/     # app registration, service principal, roles
│   └─ google/     # service account, IAM bindings
├─ README.md
└─ terragrunt.hcl  # orchestration: reads values.hcl
```

You create a `values.hcl` with your backend configuration and the region details for your cloud (the exact block differs per cloud; see the per-cloud page), then run the Terragrunt lifecycle:

COMMANDLINE |

```
terragrunt init    # initialize and validate the generated Terraform
terragrunt plan    # review the planned changes
terragrunt apply   # provision the identity, roles, and credentials
```

After a successful apply, share the generated deployment-identity details with Akka: the role ARN to assume on AWS, the service-principal credentials on Azure, or the service-account identity to impersonate on GCP.

The utility is the source of truth for the exact permissions Akka’s deployment identity requires. Review its generated output rather than a hand-maintained list, so the permissions you grant always match what Akka runs.

Identity model

AAO uses four roles across all clouds:

Role	Responsibilities	Who creates it
Deployment	Creation and management of infrastructure	akka-bootstrap
Editor	Management of infrastructure without delete permissions	You
Viewer	Read-only permissions for infrastructure	You
SRE	Identical to the Deployment role, for emergency response or upkeep	You

Within Kubernetes, Akka uses least-privilege access scoped to the Akka-managed namespaces and limits its access to customer-owned Kubernetes resources.

Choose your cloud

Continue with your cloud for its specifics:

- [BYOC on AWS](#): VPC and subnets, the IAM identity model, private connectivity, and the AWS `values.hcl`.
- [BYOC on Azure](#): virtual network, the Microsoft Entra ID identity model, resource providers, private connectivity, and the Azure `values.hcl`.

- [BYOC on GCP](#): VPC, the service-account identity model, private connectivity, and the GCP `values.hcl`.

- AWS Transit Gateway

Identity model

To begin, create a non-human privileged IAM role with the minimum permissions to set up and manage the Akka environment, plus a trust policy that lets the Federation Plane's identity assume it. This role handles initial infrastructure bootstrapping, release deployments, and ongoing maintenance.

AAO uses four roles:

Role	Responsibilities	Who creates it
Deployment	Creation and management of infrastructure on AWS	akka-bootstrap
Editor	Management of infrastructure without delete permissions	You
Viewer	Read-only permissions for infrastructure	You
SRE	Identical to the Deployment role, for emergency response or upkeep	You

The exact IAM actions for each role are created and managed by `akka-bootstrap` and are the source of truth. Review the permission reference the utility generates rather than a copy here, so the permissions you grant always match what Akka runs. Within Kubernetes, Akka uses least-privilege access scoped to the Akka-managed namespaces and limits its access to customer-owned Kubernetes resources.

Installation

Prepare your AWS account with the `akka-bootstrap` utility, then Akka provisions the region. See [Bring Your Own Cloud setup](#) for the akka-bootstrap utility and the shared install flow.

Create a `values.hcl` file with your backend configuration and AWS region details:

```

inputs = {
  backend = {
    type = "s3"
    config = {
      bucket = "your-terraform-state-bucket"
      prefix = "akka-bootstrap"
    }
  }
}
akka_regions = {
  aws = [
    {
      environment      = "prod"          # dev | stage | prod
      akka_region_name = "aws-<tenant>--<tenant-env>--<region>"
      cloud_region_name = "<region>"
      profile           = "<profile>"
    }
  ]
  azure = []
  gcp   = []
}
}

```

Then run the Terragrunt lifecycle:

```

terragrunt init    # initialize and validate the generated Terraform
terragrunt plan    # review the planned changes
terragrunt apply   # provision the IAM role, policies, and credentials

```

After a successful apply, attach the generated role ARN to your region request so the Federation Plane can assume it and manage resources in your account.

Amazon EKS

Managed Kubernetes hosts the Application Plane, with workloads spread across availability zones. Akka deploys operators for route management, elasticity, metadata sync, rolling deployment, certificate management, key rotation, and multi-region failover.

Private connectivity

By default, traffic between the region and your internal networks traverses the internet. To keep it private, AAO on AWS uses an AWS Transit Gateway to route region-to-internal traffic across accounts and VPCs. The per-flow detail is on the [data flows](#) page.

The Transit Gateway pattern:

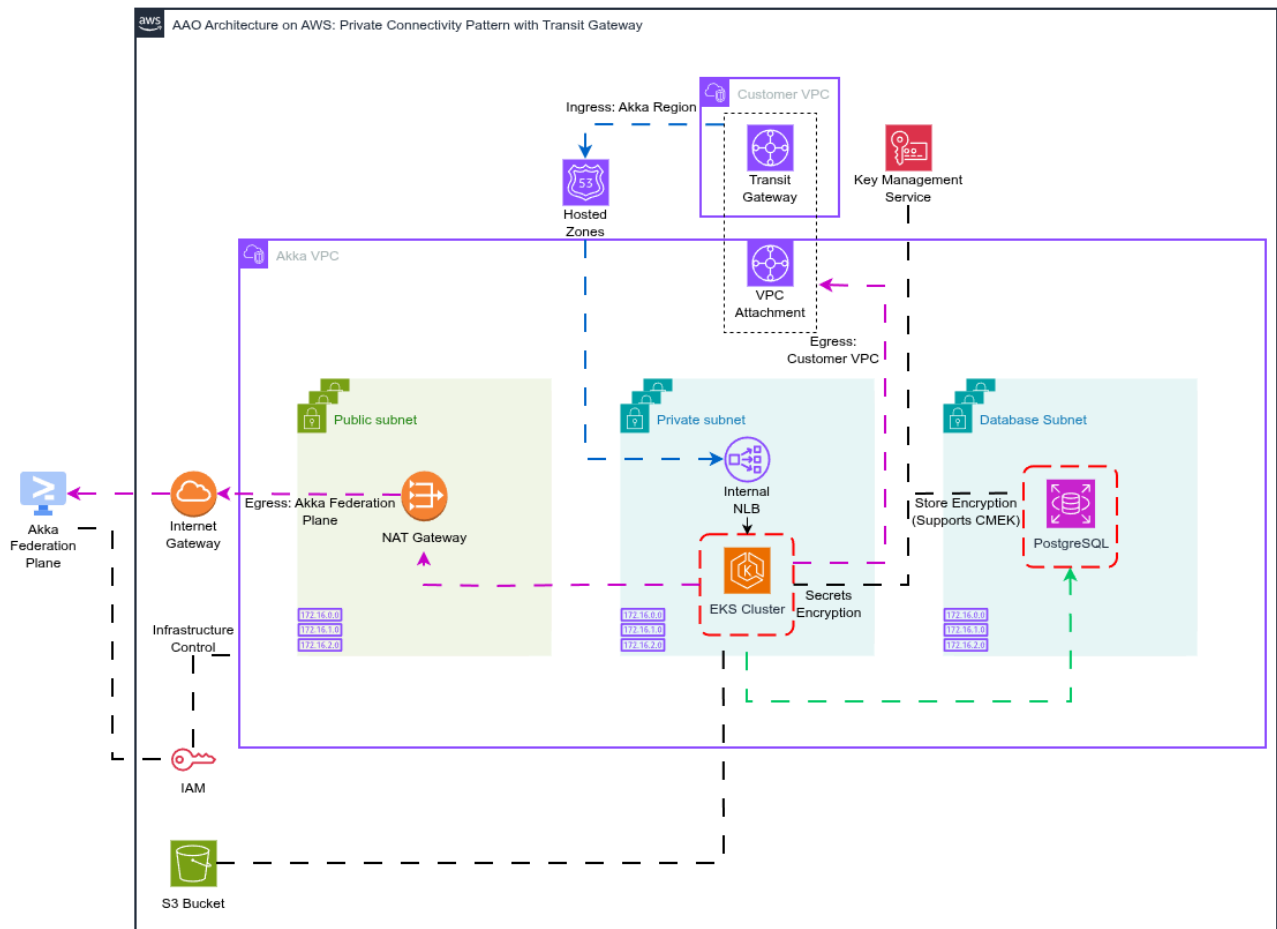


Figure 2. AAO on AWS: private connectivity with a Transit Gateway

A hub-and-spoke topology (private only, with platform images pulled from a container registry bucket):

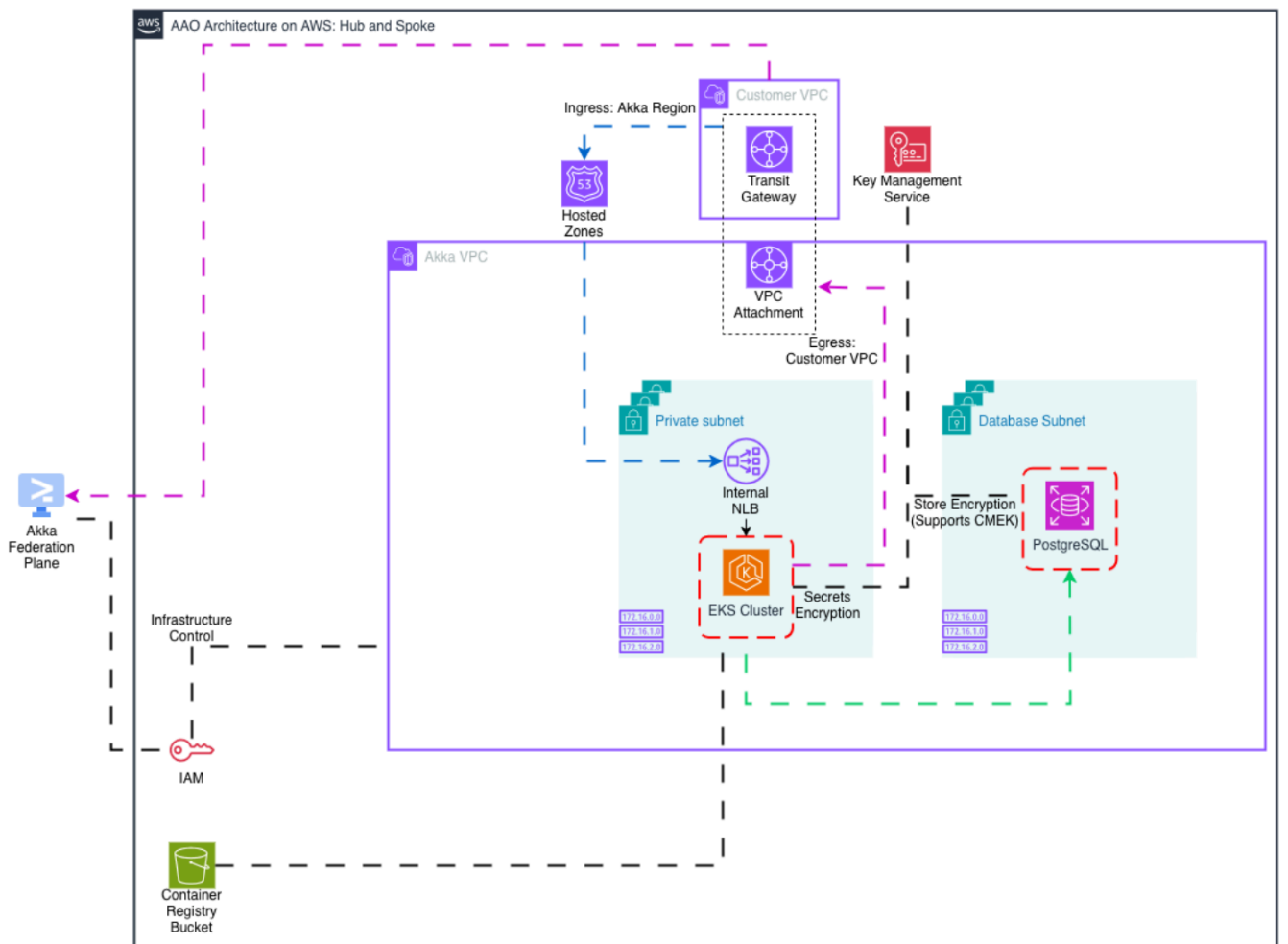


Figure 3. AAO on AWS: hub-and-spoke topology

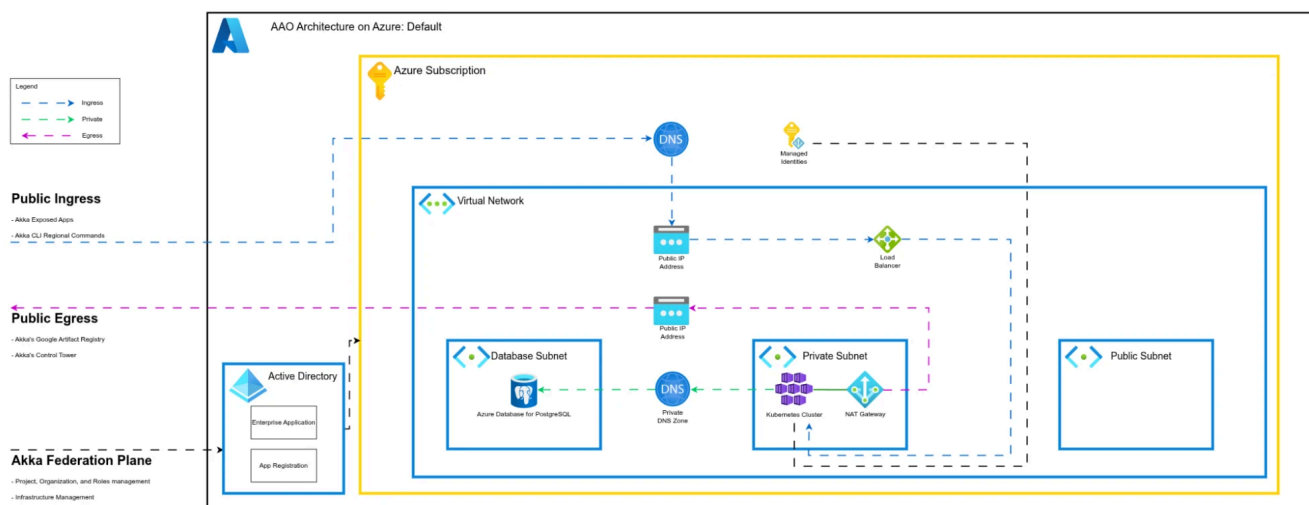
Bringing your own EKS cluster (BYOK8s)

If you provision the cluster yourself instead of having Akka provision into your account, see [Akka Automated Operations on AWS with your own Kubernetes](#) for the complete, self-contained guide (the shared requirements plus the AWS specifics).

See also

- [Technical overview](#)
- [BYOK8s requirements](#)
- [Data flows](#)
- [BYOC disaster recovery](#)

The default architecture. Akka provisions a region into your virtual network with private and database subnets, using Microsoft Entra ID for the deployment identity.



AAO on Azure uses these Azure services:

- Microsoft Entra ID (applications, app registrations, service principals)
- Microsoft Azure Resource Manager (subscription, resource group, roles and role assignments)
- Networking: virtual network, subnet, public IP address, NAT gateway, load balancer, private and public DNS zones
- Compute: AKS cluster, node pools (virtual machine scale sets), network security groups
- Data: Azure Database for PostgreSQL flexible server
- Storage: storage accounts and blob containers
- Identity: managed identities and federated credentials

An enterprise application is registered in Microsoft Entra ID to create its identity, and a service principal is created for it. A non-human privileged role with the minimum permissions is configured and used by the Federation Plane to securely bootstrap and manage the environment.

AAO uses four roles: Deployment (created by `akka-bootstrap`), and Editor, Viewer, and SRE (created by you). The exact permissions are created and managed by `akka-bootstrap` and are the source of truth; review the generated permission reference rather than a copy here. The Editor role cannot delete resources and cannot read or write federated identity credentials.

Resource providers to register

These register automatically through `akka-bootstrap`. Verify they are enabled on the subscription:

- `Microsoft.Authorization`
- `Microsoft.ContainerService`
- `Microsoft.DBforPostgreSQL`
- `Microsoft.ManagedIdentity`
- `Microsoft.Resources`
- `Microsoft.Network`
- `Microsoft.Storage`

The subscription must also have EncryptionAtHost enabled:

```
az feature register --name EncryptionAtHost --namespace Microsoft.Compute  
az provider register -n Microsoft.Compute
```

COMMAND LINE |

Installation

Prepare your subscription with the `akka-bootstrap` utility, then Akka provisions the region. Create a `values.hcl` with your backend configuration and Azure region details:

```

inputs = {
  backend = {
    type = "azurerm"
    config = {
      bucket = "your-terraform-state-bucket"
      prefix = "akka-bootstrap"
    }
  }
}

akka_regions = {
  azure = [
    {
      environment      = "prod"      # dev | stage | prod
      akka_region_name = "az-<tenant>--<region>"
      tenant_id        = "azure-tenant-id"
      subscription_id   = "azure-subscription-id"
    }
  ]
  aws = []
  gcp = []
}

```

Then run `terragrunt init`, `terragrunt plan`, and `terragrunt apply`. After a successful apply, share the generated service-principal credentials with Akka so the Federation Plane can authenticate and manage resources in your subscription.

Private connectivity

By default, traffic between the region and your internal networks traverses the internet. To keep it private, AAO on Azure uses Virtual Network Peering with a user-defined NAT gateway. The per-flow detail is on the [data flows](#) page.

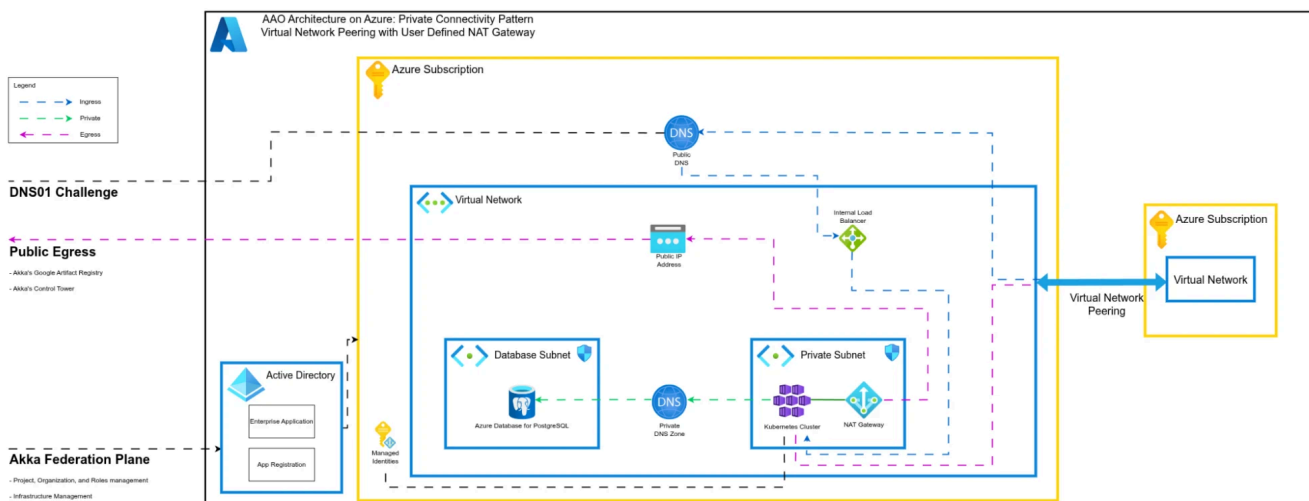


Figure 2. AAO on Azure: private connectivity with Virtual Network Peering and a user-defined NAT gateway

Bringing your own AKS cluster (BYOK8s)

If you provision the cluster yourself, see [Akka Automated Operations on Azure with your own Kubernetes](#) for the complete, self-contained guide.

See also

- [Technical overview](#)
- [BYOK8s requirements](#)
- [Data flows](#)

This page covers running Akka Automated Operations (AAO) on GCP. For the architecture and concepts, start with the [technical overview](#).

The architecture Akka provisions into your GCP project, with a VPC network, GKE, Cloud SQL, and the supporting services below.

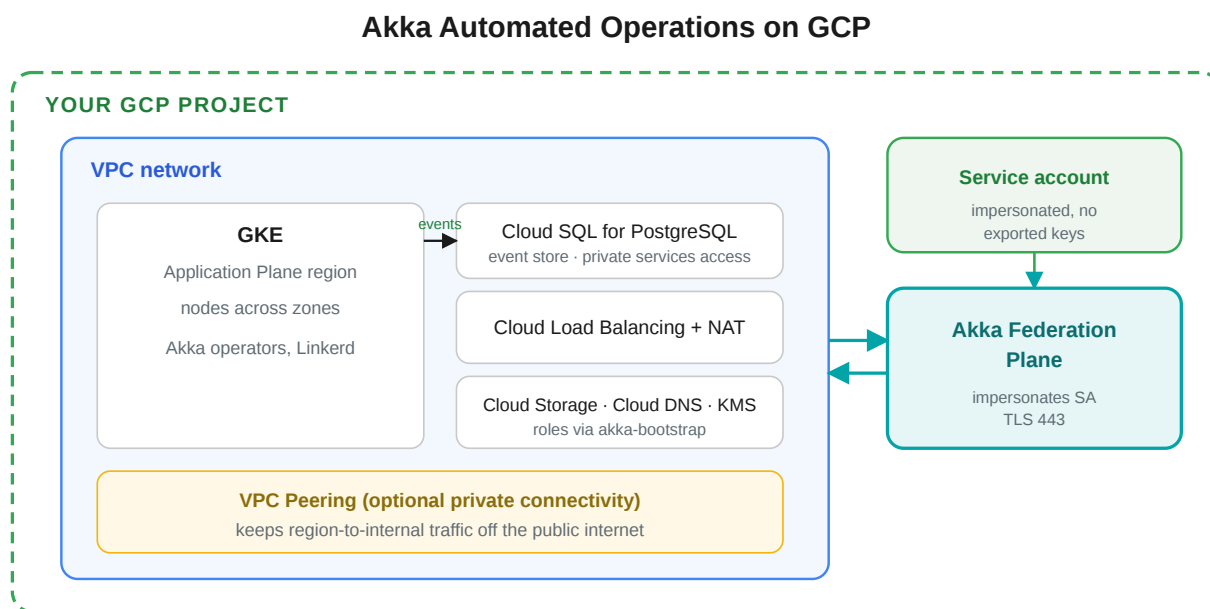


Figure 1. AAO on GCP: architecture

GCP resources

AAO on GCP uses these Google Cloud services (from the `akka-bootstrap` `modules/google` module):

- Cloud IAM and Cloud Resource Manager (service account, custom roles, and bindings)
- Compute Engine networking (VPC, subnets, Cloud NAT, external IP, Cloud Load Balancing)
- Google Kubernetes Engine (GKE)
- Cloud SQL for PostgreSQL
- Service Networking (private services access for the database)
- Cloud DNS (public and private zones)
- Cloud KMS
- Cloud Storage

Identity model

The cloud account is a GCP project. A dedicated Google service account is created for the region and assigned a non-human privileged role with the minimum permissions to set up and manage it. The service account is then configured so the Federation Plane's identity can impersonate it to bootstrap and manage the region.

Rather than long-lived exported keys, the Federation Plane impersonates the dedicated service account, which keeps bootstrap and management credential-free on your side.

AAO uses four roles: Deployment (created by `akka-bootstrap`), and Editor, Viewer, and SRE (created by you). The exact roles and bindings are created and managed by `akka-bootstrap` and are the source of truth; review the generated permission reference rather than a copy here.

Google GKE

Managed Kubernetes hosts the Application Plane, with instances spread topologically across zones. The event store is Cloud SQL for PostgreSQL.

Installation

Prepare your project with the `akka-bootstrap` utility. In `values.hcl`, use a `gcs` backend and populate the `gcp` block of `akka_regions`, then run `terraform init`, `terraform plan`, and `terraform apply`. After a successful apply, share the generated service-account details with Akka so the Federation Plane can impersonate it. Bootstrap is driven by the `modules/google` Terraform module.

Private connectivity

By default, traffic between the region and your internal networks traverses the internet. To keep it private, AAO on GCP uses VPC Peering between the region VPC and your internal networks. The per-flow detail is on the [data flows](#) page.

Bringing your own GKE cluster (BYOK8s)

If you provision the cluster yourself, see [Akka Automated Operations on GCP with your own Kubernetes](#) for the complete, self-contained guide.

See also

- [Technical overview](#)

- BYOK8s requirements
- Data flows

This page documents the individual network flows between your environment and the Akka platform. Each flow lists the data carried, its source and destination, the connectivity used, and how it is authenticated. "Your ..." denotes a resource in your own environment.

Connectivity terms used below:

- **Peering:** a private network path between your environment and the Akka region.
- **Firewall, Federation Plane allowlisted:** a customer-initiated outbound flow to the Federation Plane over the internet, from an allowlisted egress path.
- **Akka NAT gateway:** a region-initiated egress flow. Akka can work with you to route these through your firewall instead.
- **Pre-authorized IP ranges:** Federation-Plane-to-region flows over the internet. A private-connectivity option is available.

Your end users

Flow	From	To	Connectivity	Auth
End user accessing an Akka service from outside your environment	End user	Akka Region	Public (through your firewall)	Your keys, secrets, and tokens
End user accessing an Akka service from inside your environment	End user	Akka Region	Private (peering)	Your keys, secrets, and tokens

Your deployed Akka services

Flow	From	To	Connectivity	Auth
Access to your private internal systems	Your service	Your APIs	Private (peering)	Your keys, secrets, and tokens
Pushing observability metrics for your services to your observability	Akka Region	Your observability	Private (peering)	Your keys, secrets, and tokens

Your operators, admins, and developers

Flow	From	To	Connectivity	Auth
Akka CLI or console authentication initiation	CLI or console	Federation Plane	Public (firewall, Federation Plane allowlisted)	Akka authentication
OpenID authentication	Console	Your IAM	Public	OpenID
Add a region to a project	CLI	Federation Plane	Public (firewall, Federation Plane allowlisted)	Akka authenticated, valid access token
Initiate multi-region failover	CLI	Akka Region	Private (peering)	Akka authenticated, valid access token
List services and projects	CLI	Akka Region	Private (peering)	Akka authenticated, valid access token
Create a project	CLI	Federation Plane	Public (firewall, Federation Plane allowlisted)	Akka authenticated, valid access token
Add a user, role, or permission	CLI	Federation Plane	Public (firewall, Federation Plane allowlisted)	Akka authenticated, valid access token
Inspect service components in a region	CLI	Akka Region	Private (peering)	Akka authenticated, valid access token
Deploy a service: initiate deployment	CI/CD or developer	Akka Region	Private (peering)	Akka authenticated, valid access token
Deploy a service: region pulls the image	Akka Region	Your image registry	Private (peering)	Akka authenticated, valid access token

Akka SRE

Flow	From	To	Connectivity	Auth
Troubleshoot a region (cloud console or kubectl)	Your VDI, when needed	Akka Region	Private (your remote desktop)	Your cloud credentials, granted through Teleport

Akka Region

Flow	From	To	Connectivity	Auth
Region observability metadata to the Federation Plane	Akka Region	Federation Plane	Public (Akka NAT gateway)	Kubernetes RBAC and Federation Plane observability token
Region requests the public key set	Akka Region	Federation Plane	Public (Akka NAT gateway)	Internal Akka JWT authentication
Certificate expiration monitoring (can be disabled)	Akka Region	Akka SRE	Public (Akka NAT gateway)	Kubernetes RBAC
Replication to another Akka Region	Akka Region	Akka Region	Private (peering)	Mutual TLS and ACLs

Akka Federation Plane

All flows below use the bootstrap credential you create and share with Akka.

Flow	From	To	Connectivity
Bootstrap a new Akka Region	Federation Plane	Akka Region	Public (pre-authorized IP ranges)
Install a new version of Akka into a region	Federation Plane	Akka Region	Public (pre-authorized IP ranges)

Flow	From	To	Connectivity
Maintain infrastructure (drift detection)	Federation Plane	Akka Region	Public (pre-authorized IP ranges)
Upgrade infrastructure (Kubernetes, database)	Federation Plane	Akka Region	Public (pre-authorized IP ranges)
Multi-region root CA management for cross-region mTLS	Federation Plane	Akka Region	Public (pre-authorized IP ranges)
Synchronize project, user, role, organization, and region metadata	Federation Plane	Akka Region	Public (pre-authorized IP ranges)

Private connectivity by cloud

Under BYOC, Akka can set up a private-connectivity option per cloud to keep region-to-internal traffic off the public internet: Transit Gateway on AWS, VPC Peering on GCP, and VNet Peering with Azure Firewall on Azure. See the [BYOC on AWS](#), [BYOC on Azure](#), and [BYOC on GCP](#) pages.