



AKKA AUTOMATED OPERATIONS

BYOC on AWS

Shared BYOC setup, AWS specifics, 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`.

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

The default multi-tenant architecture. Akka provisions a region into your VPC with public, private, and database subnets, and reaches it from the Federation Plane for control-plane operations.

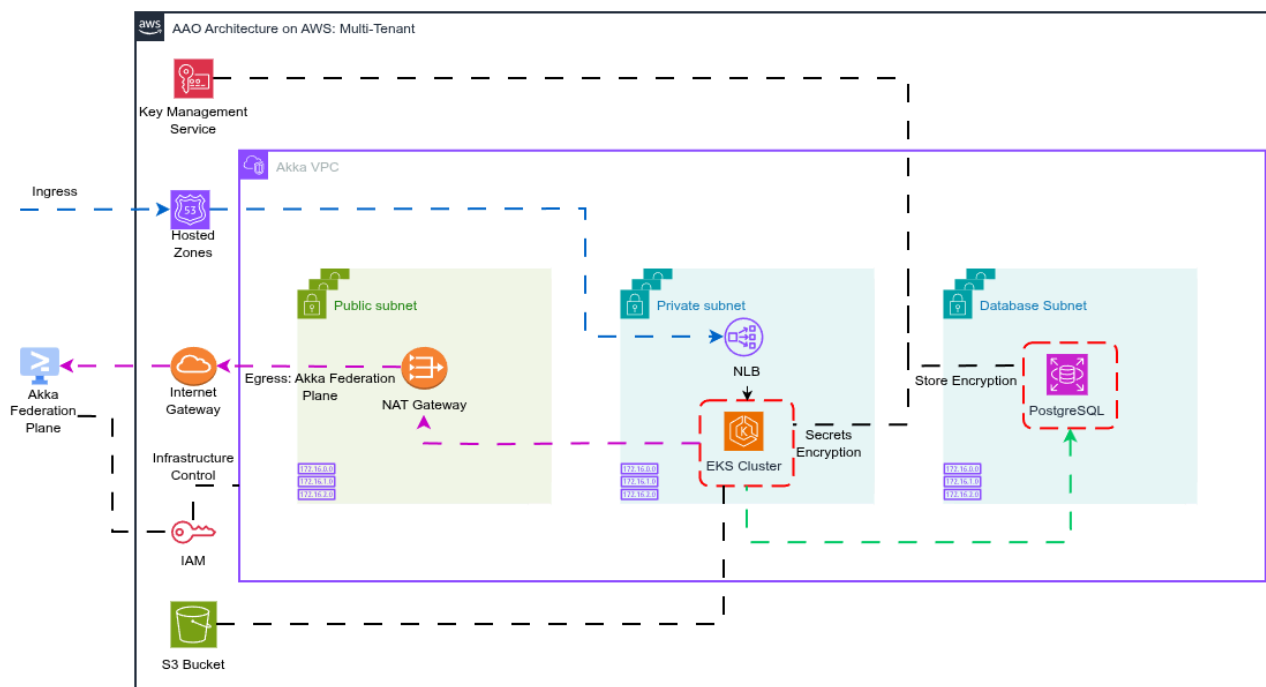


Figure 1. AAO on AWS: multi-tenant architecture (default)

AWS resources

AAO on AWS uses these AWS services:

- AWS Identity and Access Management (IAM)
- AWS RDS for PostgreSQL
- AWS Elastic Kubernetes Service (EKS)
- AWS Virtual Private Cloud (VPC)
- AWS Elastic Load Balancing
- AWS Route 53
- AWS Simple Storage Service (S3)
- AWS Key Management Service (KMS)

- 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:

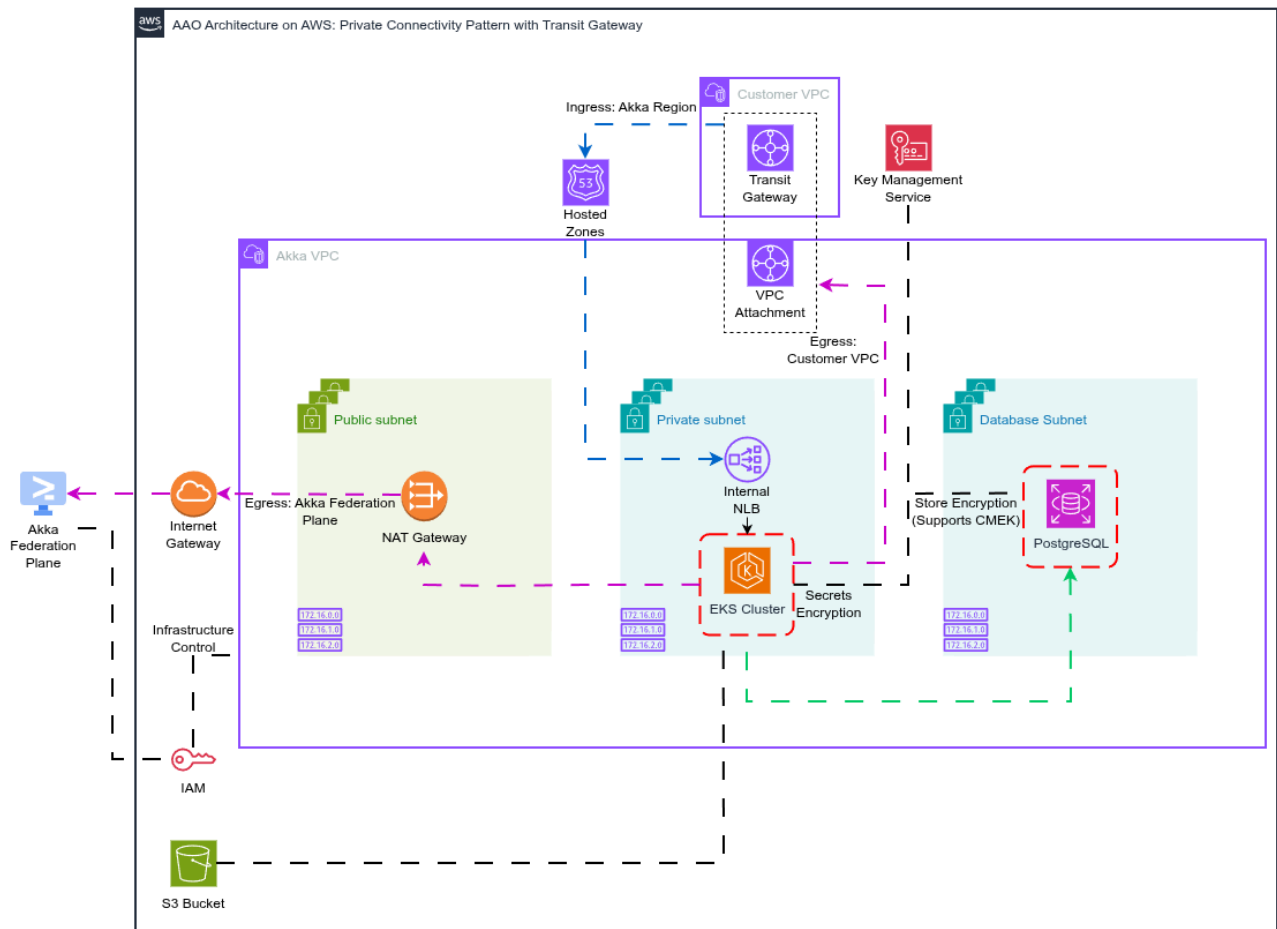


Figure 2. AAO on AWS: private connectivity with a Transit Gateway

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

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.