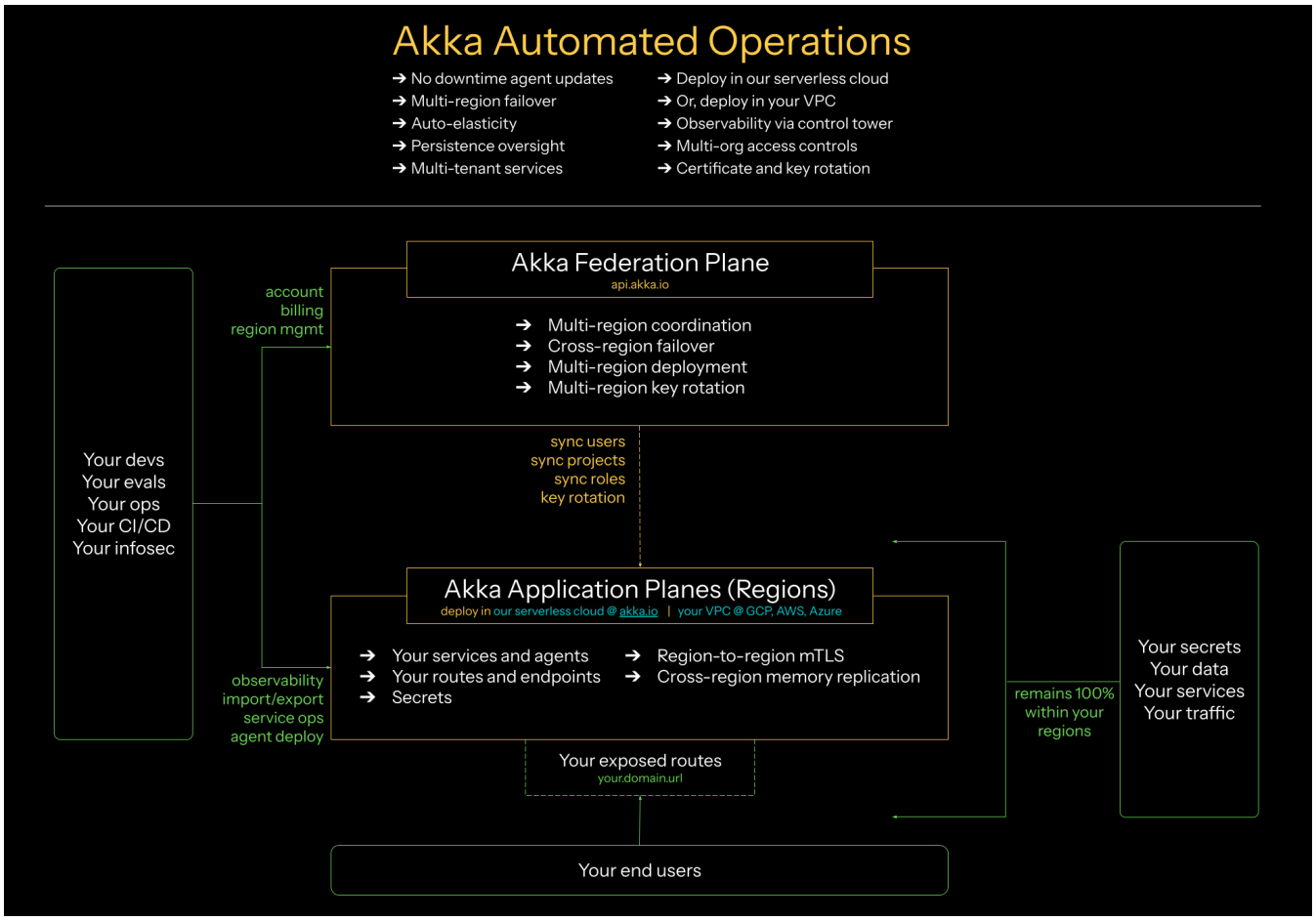
AKKA AUTOMATED OPERATIONS

Technical Overview

Architecture, installation, and operations across Kubernetes on AWS, Azure, and GCP.

DOC.AKKA.IO · AKKA

Akka Automated Operations (AAO) is a remotely installed, actively maintained Akka region that runs inside your own cloud account, on Kubernetes across AWS, Azure, or GCP. You keep custodial ownership of the infrastructure; Akka installs, monitors, patches, and scales it, targeting up to six-nines (99.9999%) availability with low latency.



Overview

When you build with the Akka SDK your services are already production-grade, self-clustering, and elastic, but you own the operations. AAO is a managed platform, controlled within your own environment, that automates the Day-2 concerns of deploying and running those services.

AAO offers three deployment options:

- Akka's serverless environment hosted at akka.io.
- **Bring Your Own Cloud (BYOC)** — Akka-provisioned regions in your own VPC on AWS, GCP, or Azure.
- **Bring Your Own Kubernetes (BYOK8s)** — installation on Kubernetes clusters you provide, including your own data centers.

With both self-managed models, regions operate entirely within your own environment: application data never leaves it or interacts with the Akka Federation Plane. Akka installs, monitors, and updates these regions, so your teams get Akka's operational expertise while retaining complete control of the environment.

Your cloud, any provider

AWS, Azure, and GCP; services can even be deployed across multiple providers simultaneously.

Data stays put

Data remains in your chosen environment, aiding GDPR / HIPAA and regional regulatory alignment.

Your contracts

Use existing cloud pricing and reduce cross-cloud egress and transfer fees.

Federation Plane and Application Plane

AAO splits into two planes: a global coordination layer run by Akka, and one or more runtime regions that live inside your accounts.

Federation Plane

The central coordination point for organizations, accounts, billing, user access, key rotation, token issuance, and app deployment across regions, which runs at akka.io. It is distributed and highly resilient, but in the extreme exception that it does become unavailable, running applications continue without interruption.

Application Plane

The runtime that hosts your workloads, where each federated region independently pulls and deploys application images, scales instances, and connects to peer regions. Application data stays isolated within your own environment and never reaches the Federation Plane.

What an Application Plane region embeds

- A managed **Kubernetes cluster** for orchestrating Akka instances as OCI containers.
- An **encrypted application data store** (append-only, event-sourced, non-queryable journal).
- A set of **Akka operators** for elasticity, observability, routing, and service management.
- **Proxies** for traffic steering between regions.
- Physical **compute and storage** for application execution.
- An **image repository** caching packed applications available in that region.

Key terminology

Term	Description
Akka Application	An app built with the Akka SDK: APIs, workflows, streaming consumers, timers, and views. Packed into OCI images and deployed as self-clustering service instances that act as their own in-memory, durable database.
Application Plane	Runtime environment hosting Akka apps within one or more regions; provides compute, storage, I/O, autoscaling, observability, and infrastructure management to meet SLAs.
Federation Plane	Global coordination point federating multiple regions into a single deployment substrate. Runs and is managed at akka.io.
Akka CLI	Interface for developers, operators, and InfoSec teams: build, test, pack, deploy, observe, and manage secrets and accounts.
Data Persistence	Durable, encrypted-at-rest event store using event sourcing; optimized for Akka's internal processing and not directly queryable.
VPC	A private, isolated network environment in a major cloud provider (general term, not exclusively AWS's implementation).
Cloud Account	The billing and administrative entity: a *subscription* (Azure), an *account* (AWS), or a *project* (GCP).

What AAO automates

Beyond operations, AAO provides behavioral extensions that would otherwise require custom application code and libraries.

Capability	Description
Runtime Patching	Live-updates JVM and infrastructure runtimes without triggering app versioning or a customer redeployment; no repackaging required.
Rolling Updates	Zero-disruption rolling deployment, even across data-model changes; old and new versions run side by side during the transition.
Elastic	Cold starts and automatic instance adjustment to traffic; persistence expands and can shard to >1M writes/sec at <20ms write latency.

Capability	Description
HA / DR	Cross-region zero-trust with rotating mTLS, multi-AZ database and Kubernetes clustering, continuous point-in-time backups, full region recovery, and CLI-based failover/failback.
App Data Replication	Replicate app data across regions (pinned single-region, write-local, or active-active) with developer-defined replication filters.
Multi-Region Deploy	Deploy a service once across one or more regions, with optional global routes for cross-region traffic.
Multi-Tenancy	Multiple projects and teams share compute and data infrastructure for substantial cost savings. Dedicated single-tenant regions are available for isolation requirements.
Observability	A Control Tower aggregates traces, spans, metrics, logs, and agentic evaluations across regions (agentic evaluations for testing use only).

What Akka provisions in a region

Networking

Set up using best practices for the specific cloud provider: VPC, subnets, load balancers, NAT.

DNS Zones

Automatically provisioned in your cloud account to resolve names for services deployed in the region.

Persistence

Encrypted-at-rest data store for app data, Akka events, and read-only views. Append-only, lock-free, and sharded; benchmarked past 1M TPS.

Kubernetes

A managed cluster from your cloud provider orchestrates Akka instances deployed as OCI containers.

Registry

An OCI registry hosting immutable, signed, versioned service assets. You may optionally use your own registry.

Akka Operators

Route management, elasticity automation, metadata sync, rolling deploys, certificate management, key rotation, and multi-region failover/recovery.

Observability split. Akka runs a *platform observability* stack for internal backend monitoring. Separately, *your* application logs, metrics, traces, and evaluations are forwarded to your observability vendor; some also go to the Federation Plane for consolidated cross-region reporting. See Observability and monitoring.

Shared responsibility

System management is shared between you and Akka across provisioning and ongoing operations. The table below is a summary; for the authoritative, row-by-row breakdown with the full Responsible / Accountable / Consulted / Informed split, see the production readiness section and its BYOC RACI.

Area	Customer	Akka
Account	Set up a dedicated cloud account with non-human privileged roles (deploy), read-only ops (monitoring), and ops (infra). Define human user roles per your policy.	Akka remote access verification.
Bootstrapping	Review default networking/compute/persistence config; recommend changes; integrate cloud security services for audit logging.	Infrastructure provisioning, configuration, and validation.
Maintenance	Provide a recurring maintenance window (typically 1–4 hrs/week). In practice most updates are non-events.	Security patches and upgrades to the Federation and Application planes.
Encryption	Provide secrets via a Key Management Store (KMS); create region-group root and per-region intermediate certificates.	Rotate secrets and keys across the Federation and Application planes.
IAM	Provide enterprise identity, authN/authZ systems, and policies.	Integrate with known IAM systems; ACL- and token-based enterprise IAM in your services.
DNS	Configure DNS record sets so the region is reachable by hostname.	Region-specific routes, service names, certificates, and optional global routes.

Area	Customer	Akka
Connectivity	Review required ingress/egress ports and optional private connectivity patterns.	Documentation and implementation details for per-cloud private connectivity.
Monitoring	Export app metrics, logs, and traces to your observability tools.	Monitoring and availability of the Federation Plane.
Backup / DR	Own the app-side DR runbook; schedule exports outside the prod region; join the periodic joint DR exercise.	Regular infrastructure and persistence-store backups.
Cost Controls	Review subscription charges (compute, egress, data); tune instance sizing and ahead-of-use allocation.	Auto-provision/de-provision compute; autoscale persistence to demand.
SDLC Tooling	Provide IDE, source repo, CI/CD, and (optionally) your own deployment registry.	IDE AI assistants, local dev, test services, deployment tooling, optional image registry.

Installation (BYOC)

Eight steps take you from first contact to a region ready for deployments. This is the **BYOC** path, where Akka provisions and operates a region inside your own cloud account. Bringing your own cluster instead? See Bring Your Own Kubernetes (BYOK8s).

1. **Understand.** This overview is your introduction to AAO. Next, receive and review the `akka-bootstrap` utility from your Akka Success Team. Reviewing it lets you inspect exactly which permissions Akka's deployment identity will require in your account.
2. **Request a region.** Submit a region request through the form in the Akka customer portal. It captures all the details Akka needs to provision the environment.
3. **Cloud account.** Create a new cloud account for Akka. Not strictly required, but a dedicated account simplifies cost management.
4. **Apply permissions.** Run the `akka-bootstrap` utility to configure your account, then attach the generated deployment-identity details to the region request case you opened.
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 mapping to the region's load-balancer IP so users and the Federation Plane can reach it.
7. **Smoke tests.** Akka attaches the region to the Federation Plane and runs smoke tests, so the region can then accept deployments.

8. **Region handover.** Register a user at akka.io with a corporate email; Akka makes them your organization's primary owner, so your team can begin using the region through the console and CLI. A production-labeled region isn't fully handed over until you complete the region-readiness steps with your Akka Success Team.

The akka-bootstrap utility

A utility that 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; the Editor, Viewer, and SRE identities must be created by you. Its outputs are consumed by the Federation Plane's provisioning module to enable automated resource management.

Directory structure

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

Each module ships the standard Terraform files (`main.tf`, `variables.tf`, `outputs.tf`, `versions.tf`) plus provider-specific IAM/permission definitions.

Usage

Create a `values.hcl` in the root with your backend and region details:

```
HCL |
inputs = {
  backend = {
    type = "gcs"                          # or "s3", "azurerm", etc.
    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"
      }
    ]
    gcp = []
    aws = []
  }
}
```

Then run the standard Terragrunt lifecycle:

```
COMMAND LINE |
terragrunt init     # initialize + validate generated main.tf
terragrunt plan     # review planned changes
terragrunt apply    # provision identity, roles, and credentials
```

> **Share identity details securely.** After a successful apply, attach the generated deployment-identity details to your region request case so the Federation Plane can manage resources in your account: the role ARN to assume on AWS, the service-principal credentials on Azure, or the service-account identity to impersonate on GCP.

Setup by platform

The Application Plane always runs on Kubernetes. Below are the shared Kubernetes model, followed by the identity and resource specifics for each cloud provider. These specifics cover the **BYOC** path, where Akka provisions into your cloud. If you're bringing your own Kubernetes cluster, see Bring Your Own Kubernetes (BYOK8s).

KUBERNETES

Each region embeds a **managed Kubernetes cluster** from the cloud provider (AKS / GKE / EKS) that physically orchestrates Akka instances as OCI containers. On top of it, Akka deploys

operators for route management, elasticity, metadata sync, rolling deployment, certificate management, key rotation, and multi-region failover.

Kubernetes access. The `akka-bootstrap` utility creates least-permission roles for the cloud provider. Within Kubernetes, Akka actively ensures least privilege is used, for both impersonation and actual usage. Akka takes no access to customer-owned Kubernetes resources.

Key Encryption Keys (KEKs) are stored separately as Kubernetes secrets and rotated periodically; rotating a KEK re-encrypts associated Data Encryption Keys (DEKs) without re-encrypting the underlying data.

AWS

The cloud account to be used is an **AWS account**. 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, enabling it to securely bootstrap and manage the environment.

This privileged role handles initial infrastructure bootstrapping, release deployments, and ongoing maintenance. Roles for human users (Editor, Viewer, SRE) can be created by you, configured with limited, time-based access tailored to your security and compliance requirements.

Amazon EKS

Managed Kubernetes hosts the Application Plane; workloads spread across availability zones.

Transit Gateway

Optional AWS Transit Gateway keeps region-to-internal traffic off the public internet.

Bootstrap is driven by the `modules/aws` Terraform module in the `akka-bootstrap` utility.

AZURE

The cloud account to be used is an **Azure subscription**. An enterprise application is registered in **Microsoft Entra ID** (Azure AD) to create its identity, and a **service principal** is created for it. A non-human privileged role with minimum permissions is configured and injected into the Federation Plane to securely bootstrap and manage the environment.

Resource providers to register. These register automatically through `akka-bootstrap's usage; 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:

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

Azure resources used:

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

VNet Peering + Azure Firewall

Virtual Network Peering with user-defined NAT gateways keeps region-to-internal traffic private.

GCP

The cloud account to be used 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.

Google GKE

Managed Kubernetes hosts the Application Plane; instances spread topologically across zones.

VPC Peering

Optional VPC Peering limits internet exposure for region-to-internal communication.

Impersonation model. Rather than long-lived exported keys, the Federation Plane's identity *impersonates* the dedicated service account, keeping bootstrap and management credential-free on the customer side.

Bootstrap is driven by the `modules/google` Terraform module in the `akka-bootstrap` utility.

Bring Your Own Kubernetes (BYOK8s)

AAO can also be installed on a Kubernetes cluster *you provision*, in your cloud or your own data center. You build the infrastructure to Akka's specification and install Teleport to grant access; Akka then configures the cluster as an application-plane region, runs smoke tests, and hands the region over. Want Akka to provision against your existing cloud infrastructure instead? See Installation (BYOC).

> **Coordinate changes.** Once the region is running, do not change any provisioned infrastructure without first informing Akka; it could break the Akka region.

Installation flow

1. **Understand.** This overview is your introduction to AAO. Contact your Akka Success Team for a package specific to a bring-your-own-Kubernetes (BYOK8s) deployment.
2. **Capacity planning.** Work with your Akka Success Team to size the environment for your requirements.
3. **Infrastructure requirements.** Work with your Akka Success Team to agree on infrastructure requirements, such as DNS subdomain zones, certificate provider, and related decisions.
4. **Infrastructure provisioning.** Provision the network, cluster, and database; set up the domain allowlist for egress traffic.
5. **Customer smoke tests.** Run the smoke-testing checklist below before handing the cluster to Akka.
6. **Teleport.** Install the Akka-provided Teleport Helm chart to grant Akka its access. Deploy it promptly, as the join token Akka gives you has a 3-hour TTL.
7. **Installation.** Akka configures the cluster as an application-plane region and sets up internal platform observability. You seed database credentials into the namespaces provided by Akka.
8. **Akka smoke tests.** Akka attaches the region to the Federation Plane and runs smoke tests, so the region can then accept deployments.
9. **Region handover.** Akka concludes smoke tests and assigns the region to your Akka organization on the console. A production-labeled region isn't fully handed over until you complete the region-readiness steps with your Akka Success Team.

Infrastructure requirements

Requirement	Specification
Kubernetes	Version `1.34` at minimum. Cilium or Calico is mandatory for network-policy enforcement; configure it as an overlay network only when the cloud-native CNI is unfeasible due to IPAM constraints.
Cluster CIDRs	`/17` for the pod network and `/17` for the service network.
Network	Minimum `/21` CIDR block from your IPAM allocation, subdivided into three database subnets (`/27` each) and three private subnets (`/24` each).
Nodes	16 vCPU / 64 GB RAM instances (e.g. AWS `m5.4xlarge` family, GCP `n2d-standard-16`, Azure `Standard_D16s_v6`). Minimum one node; Kubernetes autoscales node count with demand.
Service Mesh	Akka deploys **Linkerd**, currently the only supported service mesh.
Load Balancer	Public Layer 4 by default; an internal load balancer is supported, and a subnet or IP can be specified on supported clouds.
DNS & Certificates	Two subdomain DNS zones, one for platform APIs and one for deployed services (e.g. `akka.acme.com`, `apps.akka.acme.com`), plus cert-manager `ClusterIssuer` objects for automatic certificate generation on both.
PostgreSQL	Version `17+`, provisioned and configured by you, highly available across zones (RDS on AWS, CloudSQL on GCP, Azure Database for PostgreSQL Flexible Server).
CSI Driver	The cloud-specific secrets-store CSI (Container Storage Interface) driver provider (AWS / GCP / Azure).
Registry	Platform images can be pulled through your Artifactory, using Akka's container registry as a mirror.

Database credentials

You seed connection credentials (username, password, host, database name) as Kubernetes Secrets into both the `kalix-system` and `kalix-management-system` namespaces. You create these namespaces if needed, but must not manage them: AAO imports them into its management scope and actively reconciles them. Akka recommends the external-secrets operator to mirror credentials from your cloud secret store.

Access

Akka's access to a customer-provisioned cluster is restricted to least-privilege permissions, scoped to the Akka-managed namespaces.

Capacity planning

Database size is driven by total application data operations per second at peak load. Work with your Akka Success Team to determine sizing. Multiple databases can also be set up if you have stricter data-segregation needs.

DB Size	Data ops/sec	CPU	Memory (GB)	Storage (GB)	Max Connections
XSmall	1,000	1	4	100	200
Small	3,000	2	16	200	500
Medium	6,000	4	32	400	1,000
Large	12,000	8	64	800	1,500
XLarge	24,000	16	128	1,600	2,000

Customer smoke-testing checklist

- Check connectivity between the database instance and the Kubernetes cluster.
- Ensure the `ClusterIssuer` is ready and in a good state; verify all tokens it uses.
- In multi-region configurations, ensure traffic flows in both directions between regions.

Networking and ports

The default network configuration includes a VPC with three subnets per region plus load balancers. By default, connectivity between the region and your environment traverses the internet; private connectivity options are available per cloud.

Region port usage

Each flow below is inbound (ingress) or outbound (egress) relative to the region. All use TLS on port 443, except the infrastructure identity platform, which also uses ports 3023, 3024, and 3026.

Flow	Type	Port(s)
Akka Applications → Akka Control Tower	Egress	443
Akka Federation Plane → Akka Region API	Ingress	443
Data Import / Export	Ingress	443
Container Registry	Ingress	443
Backoffice Functions	Ingress	443
Kubernetes Cluster → Akka Platform Container Registry (platform image pulls)	Egress	443
Akka Region → Akka Federation Plane & Console	Egress	443
Platform Observability Agent → Akka	Egress	443
Infrastructure Identity Platform	Egress	443, 3023, 3024, 3026

The full hostname and IP allowlist for egress traffic (Teleport, console, Federation Plane, Control Tower, observability, and container registry endpoints) is provided in the cloud-specific specification document.

Private connectivity by cloud

By default, traffic between the region and your internal networks traverses the internet. Each cloud offers a private-connectivity option to keep it off the public internet:

AWS

Transit Gateway — route region-to-internal traffic privately across accounts and VPCs.

GCP

VPC Peering — peer the region VPC with internal networks to avoid public exposure.

AZURE

VNet Peering + Firewall — VNet Peering with Azure Firewall and user-defined NAT gateways.

Command and control

The Federation Plane and Application Plane expose the **Management API** and **Execution API** respectively, secured with TLS and bearer tokens in the HTTP Authorization header. By default these APIs are internet-facing with certificates from a globally trusted CA; private connectivity options keep endpoints off the public internet entirely. Inter-region traffic uses **mutual TLS** with per-client/per-service certificates, coordinated by a multi-region ingress service embedded in each region. See Multi-region infrastructure for the multi-region certificate and ingress details.

Security and encryption

Application-plane data is completely isolated from the Federation Plane and stays within your own environment. It is stored in an encrypted, durable event store that cannot be externally queried.

Encryption layers

Data at rest uses the store's default encryption; keys can be provided by Akka or your KMS. Akka adds a further encryption layer for:

- Secrets synced to regional execution clusters from the Federation Plane via the management API.
- Authentication secrets such as TOTP secrets and OpenID refresh tokens.
- Secrets you provide through environment variables or your KMS.

DEK / KEK rotation

Data Encryption Keys encrypt data and are stored alongside it, themselves encrypted by a Key Encryption Key. KEKs live separately as Kubernetes secrets and rotate periodically; rotating a KEK re-encrypts DEKs without re-encrypting the underlying data.

Inter-region mTLS

Each client and service gets unique certificates (issued by Akka's or your KMS). Communication is established only if both sides mutually authenticate; otherwise it is blocked.

> Akka adheres to a range of compliance standards. Full certification detail lives in the Akka Trust Center. Infrastructure change control uses a declarative model with drift detection, self-healing, RBAC, and audit logging.

Default sizing and scaling

A new region and its services start with these defaults; each scales automatically with demand and can be tuned with your Akka Success Team.

Persistence — 1M+ TPS ceiling

Default is a small store: thousands of TPS, lowest initial cost. Scales and shards to well past 1M writes/sec at <20ms latency.

Compute — Multi-AZ

Always-available instance pools spread topologically across zones. Default mixes on-demand and spot instances to balance cost and performance.

Service instance — 3× instances

Each deployed service defaults to 3 instances across AZs with 2,560 MB RAM each. Autoscales on CPU usage; sizing tunable with your Akka Success Team.

Backups and disaster recovery

Single-region Kubernetes resources and databases are backed up and recoverable. Infrastructure and configuration upgrades can improve these commitments.

Persistence RPO — 5 min

Maximum data-loss window for the persistence store.

Persistence RTO — 24 hr

Target recovery time for the persistence store.

Kubernetes RTO / RPO — 1 hr

Recovery targets for Kubernetes cluster state and configuration.

Higher availability and resiliency are available through multi-region support, which replicates your application across regions for low-latency failover. See Regions. Akka also provides data-export capabilities to move customer data to external storage for additional processing or long-term retention.

Support and maintenance

Akka continuously applies security patches and software upgrades to the underlying platform, including infrastructure upgrades needed to keep it secure and performant. Compute nodes are elastic and expand/contract with deployed services; database storage autoscales, while additional database CPU/memory is applied by the Akka team when workloads exceed utilization thresholds. On request, Akka can pre-warm environments ahead of anticipated spikes.

If issues arise, Akka provides a customer support portal where cases can be created per the Customer Support Policy and addressed based on severity and impact.