VoxaFoundry Deployment Guide

(AWS Reference Architecture & Implementation Manual)

Contact:

📧 voxafoundry@globant.com
🌐 www.globant.com

Document Version	1.0
Release Date	March 2026
Author(s)	Ritesh Menon, Jaydeep Sheth, Nishant Modi
Confidentiality	Globant Confidential – For Authorized Use Only

Document Summary

This guide explains how to deploy and operate **VoxaFoundry**, Globant's enterprise Voice AI platform, in a **customer-owned AWS account**. It uses an **automated CloudFormation-based deployment** to provision the required AWS infrastructure and install the platform on **Amazon EKS**, including secure DNS and TLS setup.

It includes:

- A high-level architecture overview and security model (networking, identity, encryption, and secrets handling).
- Step-by-step CloudFormation deployment and upgrade instructions (versioned releases).
- Operational guidance for health checks, backup and recovery, and routine maintenance.
- Cost drivers and support/SLA guidance.

Disclaimer

This document contains confidential and proprietary information intended solely for the recipient organization. Unauthorized distribution or disclosure is strictly prohibited.

Globant

Table Of Contents

1. Introduction

VoxaFoundry Platform enables enterprises to deploy secure, cost-effective, and enterprise-grade conversational AI directly on their own infrastructure, delivering a clear strategic advantage over traditional, vendor-dependent solutions.

Built for AWS, VoxaFoundry is an AI-powered voice platform that allows organizations to design, deploy, and operate intelligent, real-time voice experiences at scale.

The platform combines:

- A **production-grade application layer** (UI, APIs, background workers)
- A **real-time media plane** for low-latency voice processing
- **Secure authentication and access controls**
- **Managed data services** for scalability and reliability
- Wide integration with all CRMs, webhooks, and MCP support.

Together, these components provide a secure, scalable, and low-latency foundation for enterprise voice interactions.

1.1. Use Cases for the Solution

VoxaFoundry enables organizations to build and run voice-based automation and conversational experiences. Key use cases include:

- **Inbound Voice Agents**: Customer support and helpdesk voice experiences.
- **Outbound Voice Automation**: Follow-ups, notifications, and campaign-driven calling workflows.
- **Contact Center Automation:** AI voice agents for customer support, helpdesk triage, and call routing.
- **Voice Self-Service:** Automated appointment booking, balance/status checks, and FAQ handling without a live agent.
- **Enterprise Voice Experiences**: Secure, authenticated access for admins and users via managed identity.
- **Scalable Real-Time Voice**: High-concurrency voice sessions using a dedicated media and TURN layer.

Features:

- Real-time voice session handling using a media plane and TURN for connectivity.
- Web-based UI for administration and interaction.
- Backend services (API + workers) for orchestration and processing.
- Secure authentication and user onboarding.
- Managed database and secrets handling for production operations.

1.2. Solution Overview

Customers deploy VoxaFoundry using a single AWS CloudFormation stack that automatically provisions all required infrastructure components. The deployment creates a multi–Availability Zone Virtual Private Cloud (VPC) and provisions Amazon EKS for running the application services.

The CloudFormation deployment provisions the following resources:

- **Networking**: A new Amazon VPC with public and private subnets across two Availability Zones, including internet and outbound connectivity.
- **Kubernetes Platform**: An Amazon EKS cluster that hosts the VoxaFoundry application workloads.
- **Load Balancing and TLS**: Public load balancers for UI and API access with TLS termination.
- **Authentication**: Amazon Cognito for user pool management and application authentication.
- **Database**: A PostgreSQL database (Amazon RDS) provisioned as part of the stack (always created new).
- **Secrets Management**: AWS Secrets Manager for storing deployment-time credentials and application configuration.
- **Automation Runner**: An AWS CodeBuild-based runner that installs/updates the application on EKS using Helm.
- **DNS Automation**: Route 53 record automation for application and service subdomains (based on the provided base domain).
- **Storage**: An Amazon S3 bucket for platform storage needs.

Application services are installed on EKS via Helm and include:

- **Frontend UI service**
- **API service**
- **Worker service**
- **Real-time media plane components** (LiveKit + TURN, SIP server)

Note: External access is secured using TLS termination at the load balancer. Internal service-to-service traffic (e.g., UI-to-API) can remain HTTP within the cluster.

1.3. Solution Deployment

The standard deployment is configured for high availability across multiple Availability Zones within a single AWS Region. VoxaFoundry is deployed using a single CloudFormation stack in a customer-owned AWS account.

Deployment model supported in this guide:

1. **Multi-AZ (Production – Default)**: High-availability configuration across two AZs.

The deployment workflow is fully automated:

- CloudFormation provisions AWS infrastructure and triggers an automated install runner.
- The runner installs (or upgrades) the VoxaFoundry Helm chart on the EKS cluster.
- Application endpoints are published using the customer-provided domain and subdomains.

1.3.1. Deployment Time

Typical end-to-end deployment completes in **25–45 minutes**, depending on region and resource quotas:

Phase	Approx. Time	Description
Network provisioning	~10–15 min	Creates VPC, subnets, gateways
EKS cluster provisioning	~15–25 min	Provisions EKS control plane and baseline node capacity
Application install & DNS	~5–10 min	Runs automated Helm install and configures service endpoints

1.3.2. Supported AWS Regions

VoxaFoundry can be deployed in AWS Regions that support the required services, including Amazon EKS, Amazon RDS, Amazon Cognito, AWS Secrets Manager, AWS CodeBuild, Route 53, and ACM.

---

2. Prerequisites and Requirements

The **VoxaFoundry** deployment requires a customer-owned AWS account with permissions to deploy AWS CloudFormation stacks and manage the associated AWS services used by the solution. The deployment is designed to be automated end-to-end using CloudFormation and an installation runner, with minimal manual setup.

To ensure a smooth deployment, complete the prerequisites in this section before starting the deployment steps in Section 7.

2.1. Technical Prerequisites

To deploy VoxaFoundry, customers need:

- An active AWS account with permissions to create and manage the required resources, including: VPC, Subnets, Route Tables, NAT Gateway, and related networking components:
    1. Amazon EKS and supporting IAM roles
    2. Amazon RDS for PostgreSQL
    3. AWS CodeBuild (used for automated installation)
    4. Amazon Cognito (user authentication)
    5. AWS Secrets Manager (secure storage of credentials)
    6. Amazon Route 53 (DNS record management)
    7. AWS Certificate Manager (ACM) (TLS certificate for HTTPS)
- An IAM user or role with **AdministratorAccess** (or an equivalent custom policy) for initial deployment.
- A registered domain name that will be used for application access (example: `yourcompany.com`).
- A public Route 53 hosted zone for the selected base domain (example: `yourcompany.com`). This is required so the deployment can automatically create the required DNS records.
- Network access to AWS Console and CloudFormation, plus permission to create service-linked roles when required.

Recommended (for verification and troubleshooting):

- AWS CLI v2 installed
- kubectl installed (Kubernetes CLI)
- Helm installed (package manager for Kubernetes)

2.2. Prerequisite Skills

Deployment engineers (from Globant or the customer team) should have practical experience with AWS and Kubernetes-based application operations. A working knowledge of the following areas is required:

- Core AWS services such as VPC, IAM, EKS, RDS, Route 53, CloudFormation, and CloudWatch.
- Basic Kubernetes concepts including namespaces, services, deployments, pods, and ingress/load balancers.
- Reviewing CloudFormation stack events and troubleshooting failed stack operations.
- Understanding security best practices such as IAM least-privilege policies, encryption, and Secrets Manager usage.

To ensure a consistent and reliable deployment experience, it is recommended that engineers hold one or more of the following AWS Certifications:

- AWS Certified Solutions Architect – Associate
- AWS Certified AI Practitioner
- AWS Certified SysOps Administrator – Associate
- AWS Certified Developer – Associate
- AWS Certified Security – Specialty (Optional) – for advanced operations and compliance

2.3. Required Keys and Credentials

VoxaFoundry requires specific keys and credentials to enable voice and AI capabilities. Some values are entered during CloudFormation deployment, while others are configured post-deployment depending on the enabled features.

At a minimum, ensure the following are available before deployment:

- **Admin Email Address**
  Used to bootstrap the initial administrator account for the platform.
- **Database Master Password**
  Entered during the CloudFormation deployment. This value is securely stored in **AWS Secrets Manager** and injected to Kubernetes cluster that will be used by the platform at runtime.
- Depending on the features enabled for your deployment, you may also need:
- **AI / Model Provider Credentials**
  Example: Amazon Bedrock access configuration and any required model permissions.
  If using additional third-party providers (optional), obtain the required API keys.
- **Telephony / SIP Provider Credentials** (if integrating with external calling)
  Example: SIP trunk credentials and any required allowlists or callback URLs.

Note: This guide assumes credentials entered during deployment are securely managed using AWS Secrets Manager and are not stored in plaintext.

2.4. Instance Sizing (Production Defaults)

To deploy VoxaFoundry, customers need:

- **Database (RDS PostgreSQL):** `t3.micro`
- **EKS Standard Node Group:** `t3.medium`
- **EKS Compute Node Group:** `c6i.xlarge`

---

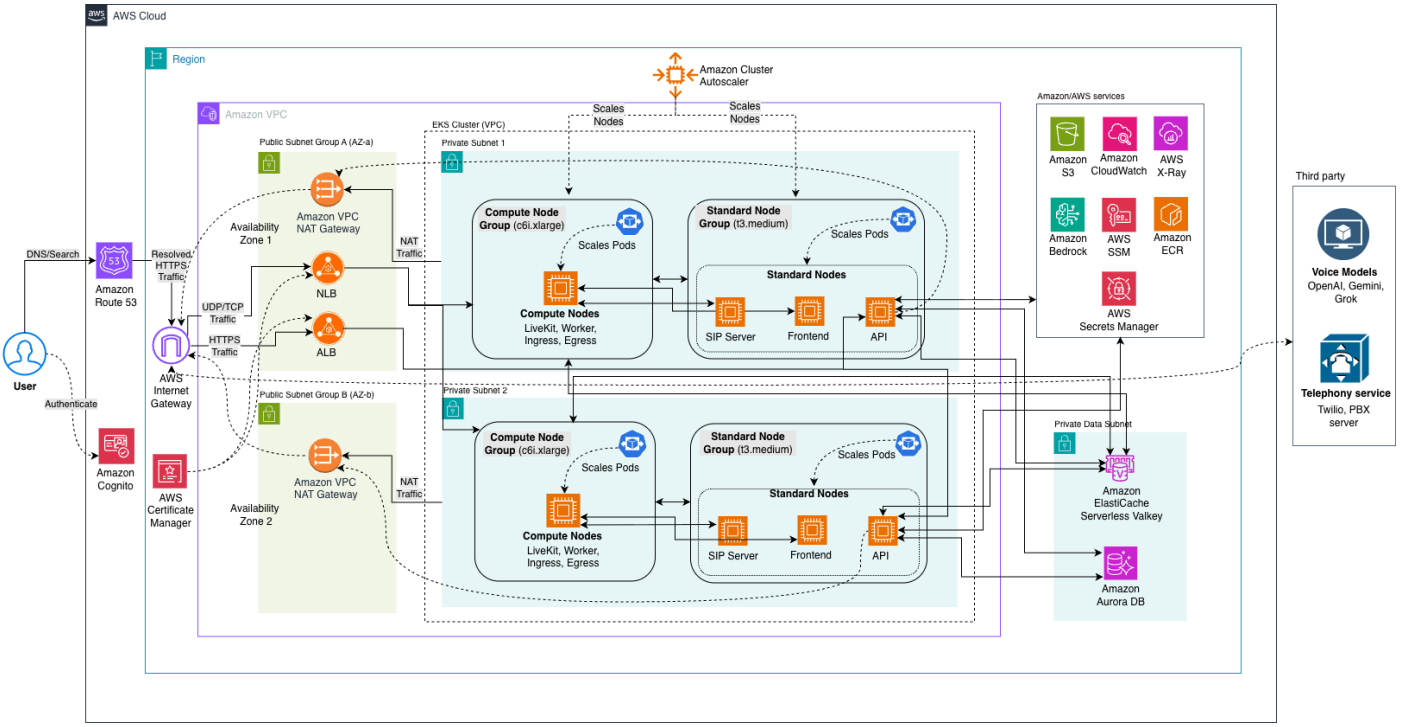
3. Solution Architecture and Security Design

This section describes the high-level architecture of **VoxaFoundry** and how the solution is deployed on AWS in a customer-owned account. The deployment is designed for high availability across two Availability Zones and follows a decoupled scaling model to isolate real-time media workloads from standard application workloads.

3.1. Architecture Overview

VoxaFoundry is deployed into a dedicated Amazon VPC within a single AWS Region. The solution provisions an Amazon EKS cluster and installs the platform services using a Helm-based deployment runner.

At a high level, the solution consists of:



3.1.1 Entry and Access

- Amazon Route 53 for DNS resolution
- AWS Certificate Manager (ACM) for TLS certificates
- Application Load Balancer (ALB) for HTTPS access to the web application and API
- Network Load Balancer (NLB) for UDP/TCP traffic (real-time voice/media use cases)

3.1.2 Kubernetes Platform (EKS)

- EKS cluster deployed across two Availability Zones
- Separate node groups for workload isolation:
    - **Compute Node Group** for media + high-throughput workloads
    - **Standard Node Group** for application services

3.1.3 Data Layer

- Amazon RDS / Aurora PostgreSQL for persistent data
- Amazon ElastiCache Serverless (Valkey) for caching and fast-access state

3.1.4 Security and Operations

- Amazon Cognito for authentication and user management
- AWS Secrets Manager for secure storage of sensitive values
- Amazon CloudWatch (and optional tracing) for monitoring and operational visibility
- Amazon ECR for container images

3.1.5 External Integrations

- Telephony provider (example: Twilio or customer PBX/SIP)
- Voice model providers (example: OpenAI, Gemini, Grok) or AWS-native model services (Bedrock / Nova Sonic 2)

3.2. Network Layout (VPC and Subnets)

VoxaFoundry is deployed into a new VPC that includes:

3.2.1 Public Subnets (Across Two AZs)

- Internet-facing load balancers (ALB / NLB)
- NAT Gateways (one per Availability Zone)

3.2.2 Private Subnets (Across Two AZs)

- EKS worker nodes (both compute and standard node groups)
- Internal Kubernetes services and pods

3.2.3 Private Data Subnet

- Database (RDS/Aurora PostgreSQL)
- Cache (ElastiCache Serverless Valkey)

This structure provides:

- Controlled ingress through load balancers only
- Private runtime for EKS workloads
- Private connectivity for database/cache traffic
- Outbound internet access via NAT Gateway, where required (e.g., calling external APIs)

3.3. Workload Isolation and Node Groups

VoxaFoundry uses a decoupled scaling approach:

3.3.1 Compute Node Group

- Intended for high-performance, real-time, or bursty workloads
- Hosts components such as:
    - Live media plane services (e.g., LiveKit)
    - Voice workers (background processing)
    - Ingress/egress services related to real-time traffic
- Scales independently to support concurrency and throughput demands

3.3.2 Standard Node Group

- Intended for steady-state application services
- Hosts components such as:
    - SIP server (if enabled)
    - Frontend UI
    - API services
- Scales independently to support API/UI traffic and operational workloads

This isolation helps ensure that real-time media workloads do not impact the stability of core application services.

3.4. Request and Traffic Flow (High Level)

1) User Access (UI / API)

- User accesses the application using a browser
- DNS resolves via Route 53
- HTTPS traffic terminates at the ALB (TLS at the load balancer)
- ALB forwards traffic to the Kubernetes services running in the private subnets

2) Real-Time Voice / Media

- Voice/media traffic uses UDP/TCP via the NLB (as required by real-time workloads)
- NLB routes traffic to the media-plane components running on the compute node group

3) Authentication

- User authentication is handled by Amazon Cognito
- After authentication, the user accesses platform APIs and UI through the load balancer endpoints
- Each transaction is authenticated with a JWT Token, except for the login public page.

4) Data Access

- Application services communicate privately with:
  - PostgreSQL database for persistence
  - Valkey cache for fast state/caching use cases

5) External Integrations

- Telephony provider connectivity enables PSTN/SIP call flows (when enabled)
- Model integrations provide AI/voice intelligence (via AWS-native or third-party providers)

Important Note

- External traffic is secured using TLS termination at the load balancer.
- Internal service-to-service calls inside the cluster (for example, frontend-to-api) may use HTTP while remaining private within the VPC.
- VoxaFoundry integrates with third-party AI providers (OpenAI, Gemini, etc.) and telephony providers (Twilio/SIP/PBX). No on-premise integrations are required, but can be supported via SIP connectivity.

3.5. Scaling and High Availability

VoxaFoundry is deployed across two Availability Zones to improve resilience and availability.

Scaling occurs at two levels:

1) Node Scaling

- Cluster/node autoscaling increases or decreases compute and standard node capacity based on demand.

2) Pod Scaling

- Kubernetes workload scaling increases or decreases the number of pods for key services (API, workers, media components) based on resource utilization and configured policies.

This enables the platform to scale for:

- Increased concurrent calls/sessions
- Higher API request volume
- Spikes in background processing needs

3.6. Security Design (High Level)

VoxaFoundry follows common AWS security best practices:

3.6.1 Network Security

- Public ingress restricted to load balancers
- EKS workloads run in private subnets
- Database and cache run in private data subnets

Identity and Access

- Authentication via Amazon Cognito
- AWS IAM is used for resource provisioning and service permissions
- Deployment and installation actions executed through a controlled runner role

3.6.2 Secrets Management

- Sensitive values (for example, database credentials) are stored in AWS Secrets Manager
- Secrets are retrieved during deployment and made available to workloads via environment variables (without exposing plaintext in templates or code)

3.6.3 Encryption

- TLS for external access (certificate managed by ACM)
- Encryption at rest for managed services is enabled using AWS-native encryption capabilities (where applicable)

3.6.4 Logging and Monitoring

- Deployment and operational logs are available via CloudWatch
- Optional tracing can be enabled for deeper operational visibility

---

4. Security

This section describes the security controls and best practices applied in VoxaFoundry. The solution follows an AWS-aligned security model focused on least privilege access, secure credential handling, encrypted communications, and reduced exposure of public resources.

4.1. IAM and Access Management

VoxaFoundry uses AWS IAM to control access for deployment operations and runtime services.

4.1.1 Deployment Access

- The CloudFormation stack provisions the AWS resources required for the solution.
- The deployment process uses an automated runner (AWS CodeBuild) to install the application on EKS using Helm.
- Deployment permissions should be granted only to designated administrators or a controlled CI/CD role.

4.1.2 Runtime Access

- Platform services running on EKS should use IAM roles only when required (for example, reading secrets, writing logs, or accessing AWS APIs).
- Access should be scoped to the minimum set of actions and resources needed for operation.

4.2. Secrets Management

VoxaFoundry uses AWS Secrets Manager to store sensitive configuration and credentials.

4.2.1 How Secrets Are Handled

- Sensitive values entered during CloudFormation deployment (for example, database credentials) are stored in Secrets Manager.
- The installation runner retrieves required secrets during deployment.
- Application workloads receive required credentials via environment variables (without exposing plaintext in templates or code).

4.2.2 Database Secret Structure (Example)

The database secret is expected to follow a JSON structure similar to:

- username
- password
- host
- Dbname

4.3. Public Resources & Data Security Controls

VoxaFoundry is designed to minimize exposure of public resources.

4.3.1 Publicly Accessible Components

- Internet-facing access is limited to AWS-managed load balancers (ALB/NLB) used for application access and real-time media connectivity.
- DNS resolution is provided via Route 53 using customer-provided domains/subdomains.

4.3.2 Private Components

- EKS worker nodes run in private subnets.
- Database and cache services are placed in private data subnets and are not publicly accessible.
- Service-to-service communication occurs inside the VPC.

4.4. Data Encryption

VoxaFoundry implements encryption for data in transit and at rest.

4.4.1 Encryption In Transit

- External traffic uses TLS termination at AWS load balancers.
- Certificates are managed using AWS Certificate Manager (ACM).
- Internal service-to-service communication inside the VPC/cluster may remain HTTP, while external ingress remains encrypted.

4.4.2 Encryption At Rest

- Database storage encryption is enabled using AWS-managed encryption capabilities.
- Secrets stored in AWS Secrets Manager are encrypted by default.
- S3 should be configured with encryption at rest and appropriate bucket policies.

4.5. Sensitive Data Storage

VoxaFoundry may store and process sensitive business information depending on customer usage (for example, call metadata, transcripts, or operational logs).

4.5.1 Data Storage Locations

- PostgreSQL database (persistent platform data)
- Cache layer (temporary/fast-access data)
- Secrets Manager (credentials and sensitive configuration)
- CloudWatch logs (operational logs)
- S3

4.6. Instance Metadata Service (IMDS) Enforcement

VoxaFoundry enforces IMDSv2 (Session-based Metadata tokens) across all computer layers. IMDSv1 is explicitly disabled in the deployment by mandating the use of session-oriented tokens for all metadata requests.

Configuration:

- HttpTokens: required:
  Disables IMDSv1 and mandates session-based tokens.
- HttpPutResponseHopLimit: 2:
  A required setting for EKS that allows pods to retrieve the token while preventing it from traversing external network hops.

---

5. Cost

This section provides a cost overview for **VoxaFoundry**. Actual costs will vary based on usage, region, selected instance sizes, and traffic patterns.

VoxaFoundry is offered as a flat-fee subscription model. The platform is licensed at a fixed cost of $10 per month per deployment, independent of the number of users, requests, or usage volume.

This licensing model is separate from AWS infrastructure costs and any third-party service usage (e.g., AI model providers or telephony providers), which are billed independently based on consumption.

5.1. Cost Overview

VoxaFoundry uses AWS managed services and Kubernetes workloads running on Amazon EKS. The primary cost drivers are:

- Amazon EKS (control plane and Kubernetes platform operations)
- Amazon EC2 (EKS worker nodes for standard and compute node groups)
- Load Balancers (ALB/NLB) and data transfer
- Amazon RDS / Aurora PostgreSQL (database compute and storage)
- Amazon ElastiCache Serverless (Valkey) (cache usage)
- Amazon Route 53 (hosted zone and DNS queries)
- AWS Certificate Manager (ACM) (certificate is typically no-cost; related DNS validation and load balancers still incur costs)
- AWS Secrets Manager (secrets stored and API calls)
- Amazon CloudWatch (logs, metrics, and alarms)
- Amazon S3
- External AI/voice model usage (AWS Bedrock, Nova Sonic 2, or third-party providers)
- Telephony provider charges (Twilio / SIP trunk / PBX)

Customers can estimate expected AWS infrastructure costs, including EKS, EC2, RDS, load balancers, Route 53, and CloudWatch, using the AWS Pricing Calculator.

Retention and Lifecycle Cost Considerations

- Longer RDS backup retention increases backup storage usage and cost.
- Redis snapshot retention affects snapshot storage costs.
- CloudWatch log retention affects logging cost depending on ingestion volume and retention duration.
- S3 lifecycle policies reduce cost by transitioning older data to Glacier, but retrieval from Glacier may add cost if frequently accessed.

5.2. Major AWS Cost Components

5.2.1 Amazon EKS

- EKS control plane cost is billed per cluster per hour.
- This deployment assumes a single production EKS cluster.

5.2.2 EKS Worker Nodes (EC2)

● Worker node costs are typically the largest cost driver.
Costs depend on:

- Instance types used for Standard Node Group and Compute Node Group. Standard nodes `t3.medium`, compute nodes `c6i.xlarge`, DB `t3.micro`.
- Minimum/maximum node scaling configuration
- Average CPU/memory utilization
- Number of concurrent sessions/calls and background workloads

5.2.3 Load Balancing and Data Transfer

Application access uses load balancers (ALB for HTTPS and NLB for real-time traffic where applicable).
Costs depend on:

- Number of load balancers
- Request volume / new connections
- Data processed
- Cross-AZ traffic and internet egress

5.2.4 Database (RDS / Aurora PostgreSQL)

Database costs include:

- Instance compute (or capacity units, if serverless)
- Storage and IOPS (depending on storage type)
- Backups and snapshots retention

This deployment model provisions a new database as part of the stack.

5.2.5 Cache (ElastiCache Serverless Valkey)

- Cache usage is billed based on consumption and throughput.
- Costs vary based on workload patterns and request rate.

5.2.6 Logging and Monitoring (CloudWatch)

CloudWatch costs depend on:

- Log ingestion volume
- Log retention period
- Custom metrics and alarms
- Optional tracing usage (if enabled)

5.2.7 Secrets Manager

Secrets Manager charges depend on:

- Number of secrets stored
- API calls made to retrieve secrets (including during deployment and runtime)

5.2.8 Storage (S3)

S3 costs depend on:

- Total storage volume
- Requests (PUT/GET)
- Retrieval patterns and lifecycle policies

5.3. External Service Costs (Non-AWS)

Depending on the enabled integrations, additional costs may include:

5.3.1 AI/Model provider costs

- Amazon Bedrock model usage (tokens/requests)
- AWS Nova Sonic 2 usage (where applicable)
- Third-party model APIs (OpenAI / Gemini / etc.), if used

5.3.2 Telephony costs

- PSTN minutes and call routing charges
- SIP trunk costs
- Phone number rental and messaging add-ons (provider-specific)

5.4. Cost Management Recommendations

The solution is well optimized post-deployment, with a few additional enhancements possible based on evolving requirements.

- Use autoscaling policies for EKS node groups and key workloads.
- Configure CloudWatch log retention (avoid infinite retention unless required).
- Use AWS Budgets and Cost Explorer to track spend and set alerts.
- Apply tagging standards to all resources for cost allocation (environment, application, owner).
- Use lifecycle policies for S3 and define retention policies for logs and recordings.

5.5 License Management

VoxaFoundry is licensed as a **flat-fee subscription of $10 per month per deployment**. This license is separate from AWS infrastructure costs and any optional third-party provider charges.

5.5.1 Obtaining the License

Customers obtain VoxaFoundry through the approved Globant commercial process or the applicable AWS Marketplace listing, depending on the engagement model.

5.5.2 Activating the License

No separate license key or quota activation is required. The license becomes effective once the customer's subscription or commercial agreement is active and the approved VoxaFoundry deployment package is used in the customer AWS account.

5.5.3 Monitoring License Status

Customers should monitor their active subscription or commercial agreement through their procurement/billing process and confirm that the deployed environment remains within the agreed licensed scope.

5.5.4 Renewal

Customers should renew the VoxaFoundry subscription before the end of the active billing term through Globant or AWS Marketplace, depending on how the license was obtained.

5.5.5 Support for Licensing Questions

For licensing, billing, or renewal questions, customers should contact Globant through the agreed commercial or support channel.

---

6. Troubleshooting (Deployment)

This section covers issues that may occur during **initial deployment** or **CloudFormation stack updates** (install/upgrade). It is focused on CloudFormation provisioning, DNS/TLS setup, and the automated installation runner.

6.1. Deployment Troubleshooting Table

Symptom	Where to Check	What to Look For	What to Share with Support
The CloudFormation stack does not complete	CloudFormation → Stacks → Events	Failing resource name, failure reason, rollback state	Stack name, Region, failing resource name, error message from Events
Deployment runner (CodeBuild) fails	CodeBuild → Build history → Logs	Failed command, Helm install/upgrade errors, kubeconfig/cluster access issues	Build ID, timestamp, error snippet around failure
EKS cluster not ready	EKS → Clusters / Node groups	Cluster not Active, nodes not Ready, node group still creating	Cluster name, cluster status, node group status
Endpoints do not resolve (DNS)	Route 53 → Hosted Zone	Missing records for app/api/livekit/turn, wrong target values	BaseDomain + subdomains used, screenshot/list of missing records
HTTPS not working / browser certificate warning	ACM → Certificates; EC2 → ALB → Listeners	Certificate not Issued, missing validation CNAMEs, ALB listener not on 443	Certificate status, domain name, ALB name, listener configuration
Stack update completes but app not upgraded	CloudFormation → Events; CodeBuild logs; EKS workloads	AppVersion updated but runner didn't execute, Helm upgrade didn't apply	Old/new AppVersion, stack update timestamp, CodeBuild build ID

6.2. Minimum Deployment Validation (After Stack Completion)

Check	Where to Verify	Expected Result
Stack status	CloudFormation → Stacks	`CREATE_COMPLETE` (deploy) or `UPDATE_COMPLETE` (upgrade)

Runner execution	CodeBuild → Build history	Latest build is `Succeeded`
TLS certificate	ACM → Certificates	Wildcard certificate is `Issued`
DNS records	Route 53 → Hosted Zone	Records exist for app/api/livekit/turn
Cluster status	EKS → Clusters	Cluster is `Active` and node groups ready

6.3. Information Required for a Deployment Support Request

Item	Details
AWS Region	Region where the stack was launched
Stack name	CloudFormation stack name
Stack status	Current status (including rollback states if any)
CloudFormation failure details	Failing resource name + error message from Events
CodeBuild details	Build ID + log snippet around failure
Domain configuration	BaseDomain + all subdomains (app/api/livekit/turn)
Certificate status	ACM status (`Issued` / `Pending validation`)
Timestamp	When the deployment/update was executed

---

7. Deployment

This section describes how to deploy VoxaFoundry using a single AWS CloudFormation stack in a customer-owned AWS account. The CloudFormation stack provisions the required AWS infrastructure and automatically installs the VoxaFoundry platform on Amazon EKS.

7.1. Deployment Overview

VoxaFoundry is deployed using the following approach:

- A CloudFormation template is launched in the customer's AWS account.
- CloudFormation provisions the foundational infrastructure (VPC, EKS, database, IAM roles, and deployment runner).
- The deployment runner installs the application on EKS using Helm.
- After deployment completes, CloudFormation outputs the platform URLs and key endpoints.

This guide assumes:

- Region: **us-east-1**
- EKS Cluster Name: **voice-ai-cluster**
- Environment: **Production** (single environment only)

7.2. Pre-Deployment Checklist

Before launching the CloudFormation stack, confirm:

- You have access to an AWS account with permissions to create VPC, EKS, IAM roles, RDS, CodeBuild, Route 53 records, and ACM certificates.
- You have a public hosted zone in Route 53 for your base domain (example: example.com).
- By default, AWS allows up to 5 VPCs per account. Please ensure the account has available VPC capacity. If required, raise a support ticket to request a quota increase.
- You have decided the subdomains to be used for the solution endpoints:
    - Application UI subdomain (example: app)
    - API subdomain (example: api)
    - LiveKit subdomain (example: livekit)
    - TURN subdomain (example: turn)
- You have the admin email address that should be used to bootstrap initial platform access.
- You have the required external keys available (if applicable) to configure after deployment.

7.3. CloudFormation Deployment Steps

7.3.1 Launch the Stack

1. Sign in to the AWS Console.
2. Navigate to **CloudFormation → Stacks → Create stack → With new resources (standard)**.
3. Under the **Prepare template**, select **Template is ready**.
4. Under **Template source**, upload the VoxaFoundry CloudFormation template (provided as part of the release / Marketplace package).

5. Select **Next**.

7.3.2 Provide Stack Details

Stack name
Use a clear production naming convention, for example:
– `ai-voice-agent-prod`

Parameters
Fill in the required parameters as described below.

7.4. CloudFormation Parameters

Use the following table while completing the CloudFormation stack launch form. Parameters not listed here should be left at the default unless your organization requires a change.

Parameter	Default Value	Description
ClusterName	marketplace-cluster	Name of the Amazon EKS cluster created for VoxaFoundry.
AppVersion	1.1.15	Application version to deploy. Changing this value during a stack update triggers an upgrade.
DatabasePassword	(No default)	PostgreSQL master password. This value is hidden in the CloudFormation UI and should be treated as sensitive.
RetainDataOnDelete	true	Controls data cleanup behavior during stack deletion. Use `true` for production to reduce the risk of accidental data removal.
BaseDomain	globant.com	Base domain used for solution endpoints. A Route 53 Public Hosted Zone must already exist for this exact domain name.
AppSubdomain	app	Subdomain for the Frontend UI endpoint (example: `app.example.com`).
ApiSubdomain	api	Subdomain for the API endpoint (example: `api.example.com`).
LiveKitSubdomain	livekit	Subdomain for LiveKit endpoint (example: `livekit.example.com`).
S3BucketName	my-voice-agent-bucket-123	Optional bucket name for recordings/data storage. Leave blank if not required.
AdminEmail	admin@globant.com	Email used to bootstrap initial admin access. A temporary password invite is sent to this email.
RDSBackupRetentionPeriod	30	Days to retain automated RDS backups.

RDSDeletionProtection	true	Enables deletion protection on the RDS instance.
S3GlacierTransitionDays	30	Days before objects transition to Glacier storage.
S3ExpirationDays	60	Total days before objects are permanently deleted.
LogRetentionDays	7	CloudWatch Log Group retention for deployment and operational logs.
RedisBackupRetentionPeriod	30	Days to retain Redis backups/snapshots. A daily snapshot window is configured (3 AM – 4 AM).

Notes

- The deployment provisions production-grade infrastructure in a customer-owned AWS account.
- Database deletion protection is enabled by default. If the stack is deleted, deletion protection must be disabled before the database can be removed.
- S3 lifecycle policies apply only when S3 bucket provisioning is enabled.
- Default Instance Sizes (Provisioned by Template)
    - DB: t3.micro
    - Standard node: t3.medium
    - Compute node: c6i.xlarge

7.5. Create the Stack

1. After completing the parameters, select **Next**.
2. Review stack options (leave defaults unless your organization requires tags, permissions boundaries, or stack policies).
3. Select **Next**.
4. Review the final page.
5. Acknowledge IAM resource creation (if prompted).
6. Select **Create stack**.

7.6. Monitor Deployment Progress

During deployment:

- In CloudFormation, monitor stack status and events until it reaches:
  – `CREATE_COMPLETE`
- If the stack shows progress but takes time, this is expected due to provisioning of:
  – EKS cluster and node groups
  – Database resources
  – Load balancers
  – Certificate issuance and validation
- The installation runner executes during stack creation. You can view deployment logs in:
  – CloudWatch logs (log group output is available after deployment)

7.7. Post-Deployment Outputs

After stack creation completes, CloudFormation provides key outputs such as:

- Frontend URL (UI access)
- API URL
- Database endpoint
- EKS cluster name
- Deployment runner log location

7.8. Initial Verification: Login

- The Frontend URL loads successfully in a browser.
- Authentication is enabled for the configured admin email. A temporary password is sent to the registered email address, after which the admin must reset the password to continue.
- The API endpoint is reachable from the UI.
- All EKS workloads are running (pods are healthy).
- DNS records resolve correctly for the configured subdomains

7.9. Import Twilio Number

- Go to Telephone → Click Import Telephone → Add Twilio credentials → Confirm successful message;

7.10. Create your first AI Agent

- Go to Agents → Create Agent → Add AI voice instructions → Select model and voice → Click Create
- Test AI Agent by entering an outbound number → Optionally use Playground to test the web-based version.

7.11. Stack Deletion Behavior

VoxaFoundry standardizes deletion behavior for core resources to prevent orphaned infrastructure.

- Core resources such as database, cache, and S3 are configured with DeletionPolicy: Delete.
- When the CloudFormation stack is deleted, these resources are removed unless protected by service-level safeguards.

Important Notes

- If RDSDeletionProtection is enabled, the database cannot be deleted until deletion protection is disabled.
- Customers should confirm retention/export requirements before deleting the stack.

---

8. Health Check

The VoxaFoundry deployment includes built-in health checks and monitoring through AWS services (CloudWatch, Load Balancers, EKS) and application-level endpoints. This section explains what to watch, where to look, and how to interpret the results.

8.1. Application Health Monitoring

Component	Where to Monitor	Key Metrics / Checks	Expected Normal Values / Notes
Application Load Balancer (ALB) – UI/API	AWS Console → EC2 → Load Balancers / Target Groups	• Target group health (Healthy/Unhealthy) • HTTPCode_Target_4XX_Count • HTTPCode_Target_5XX_Count • RequestCount • TargetResponseTime	• Healthy targets should remain 100% healthy • Unhealthy should be 0 • 5XX errors should remain near 0; repeated spikes indicate API/service issues • Latency should remain stable; sustained increases indicate backend/database/model latency
Network Load Balancer (NLB) – LiveKit/TURN (real-time media)	AWS Console → EC2 → Load Balancers / Target Groups	• Target group health (where applicable) • ActiveFlowCount / NewFlowCount (if used) • Listener availability	• Targets should remain healthy • Flow counts should correlate with call volume • Sudden drops during active usage indicate connectivity issues
EKS Workloads (API / Worker / Frontend)	AWS Console → EKS → Clusters → Workloads (or Kubernetes view)	• Pod status (Running/Ready) • Restarts • CrashLoopBackOff / ImagePullBackOff • Deployment desired vs available replicas	• All pods Running and Ready • Restarts should be minimal/occasional • Persistent CrashLoopBackOff indicates config/secrets/db connectivity problems
API Health Endpoint	Browser / HTTP client via API endpoint	• Health endpoint response (example: `/healthz`) • API reachable from UI	• Endpoint returns HTTP 200 with simple OK response • UI can load pages that require API calls

Worker Health	EKS workload view + logs	• Worker pods Running/Ready • No repeated restarts • Job/queue processing continues (based on platform usage)	• Worker pods stable • Errors should be occasional; persistent error loops indicate integration or dependency failures
LiveKit Server	EKS workload view + logs (and LiveKit endpoint reachability)	• Pod status Running/Ready • LiveKit endpoint reachable • (If enabled) metrics endpoint / basic server logs	• LiveKit service stable under load • Errors should not be continuous • Connectivity issues usually show as call setup failures
Amazon RDS / PostgreSQL	AWS Console → RDS → Databases → Monitoring	• CPUUtilization • FreeStorageSpace • DatabaseConnections • Read/Write latency • FreeableMemory	• CPU typically moderate under normal load • Connections stable (no rapid growth) • Latency stable; sustained spikes indicate DB saturation or heavy queries
ElastiCache Serverless (Valkey)	AWS Console → CloudWatch → Metrics → AWS/ElastiCache	• CPUUtilization • MemoryUsage / UsedMemory • Network throughput	• CPU and memory should remain below sustained high utilization • Sudden jumps usually correlate with traffic spikes or session bursts
Amazon Cognito	AWS Console → CloudWatch → Metrics → AWS/Cognito	• SignInSuccesses • UserAuthenticationFailures	• Sign-in failures should remain low relative to successful sign-ins • Sudden rises often indicate redirect URL/domain mismatch or user lockouts
Application Logs (API / Worker / Frontend)	AWS Console → CloudWatch → Log Groups	• Error rate in logs • Repeated exceptions • Startup/config errors	• Mostly INFO-level logs during normal operations • Repeated ERROR logs indicate missing secrets, integration failures, or DB/cache connectivity issues

Deployment Runner Logs (CodeBuild)	CloudWatch → Log Groups	• Log availability • Latest run logs present	• Logs retained for configured duration (default 7 days)

8.2. Tracing and Distributed Monitoring

If distributed tracing is enabled for VoxaFoundry:

- Use the configured tracing backend (or CloudWatch/X-Ray if integrated) to review service latency.
- Normal traces show stable, low-latency internal calls.
- Requests that repeatedly exceed expected thresholds indicate API latency, database latency, or external model/provider timeouts.

8.3. Automated Health Checks

VoxaFoundry relies on automated health checks at multiple layers:

8.3.1 Load Balancer Health Checks

- ALB health checks validate the UI/API targets before sending production traffic.
- Unhealthy targets are removed from rotation automatically.

8.3.2 Kubernetes Readiness/Liveness

- Pods are only marked Ready after passing readiness checks.
- Liveness failures trigger pod restarts to recover unhealthy containers.

8.4. Alerting and Anomalies

Customers can configure CloudWatch alarms (and/or operational alerts) to notify the operations team when:

- ALB 5XX errors exceed a defined threshold over a short window
- ALB target health drops below the expected healthy percentage
- EKS pods show frequent restarts or enter CrashLoopBackOff
- RDS CPU or storage crosses defined limits
- Cache memory/CPU exceeds expected thresholds
- Cognito authentication failures rise sharply compared to normal login volume

Alerts can send notifications via SNS, email distribution lists, or the customer’s incident management tooling.

8.5. Summary of Normal KPI Ranges

Metric	Normal Range / Target
ALB Healthy Targets	100% healthy
ALB 4XX Error Rate	≤ 5% (depends on expected auth/invalid request patterns)
ALB 5XX Error Rate	≈ 0% (sustained > 0 indicates a service issue)
API Pod Availability	Desired replicas = Available replicas
API CPU Utilization	Typically 40–70% under normal traffic (autoscaling dependent)
Frontend CPU Utilization	Typically 30–60% under normal traffic
Pod Restarts	Low and non-continuous (no repeated restart loops)
Database Connections	Stable (no continuous upward trend)
DB Latency	Stable; sustained spikes indicate saturation or heavy queries
Cache CPU Utilization	Typically ≤ 60% under normal load
Cognito Auth Failures	Typically ≤ 5% of total attempts

9. Backup and Recovery

This section describes the backup and recovery approach for VoxaFoundry. VoxaFoundry relies on AWS managed services for backups and recovery of persistent data.

9.1. Backup Scope

VoxaFoundry stores persistent data primarily in:

- Amazon RDS (PostgreSQL)
- Amazon S3

Other components, such as Kubernetes pods and node instances, are considered **stateless** and are recreated automatically by EKS/Helm during normal operations.

9.2. Database Backups (Amazon RDS – PostgreSQL)

9.2.1 Backup Retention

- Automated backup retention is configurable at deployment time.
- Default retention: 30 days (RDSBackupRetentionPeriod).

9.2.2 Deletion Protection

- The database is protected from accidental deletion by default.
- Default: enabled (RDSDeletionProtection).

9.2.3 Recovery Options

Snapshot restore and point-in-time recovery (PITR) are available within the backup retention window.

9.3. Point-in-Time Recovery (PITR)

Amazon RDS supports point-in-time recovery for PostgreSQL within the configured retention period.

Use cases

- Recover from accidental data deletion
- Recover from application errors that impacted data
- Restore the database to a known good point in time

Notes

- PITR availability depends on backup retention and successful capture of transaction logs.
- Recovery results in a new database instance restored to the target time.

9.4. S3 Backups

If the deployment enables S3 usage (for example, storage of operational artifacts or platform data), customers should configure S3 with appropriate durability and retention controls.

Lifecycle Policies

- Objects transition to Glacier after 30 days (S3GlacierTransitionDays).
- Objects are permanently deleted after 60 days total (S3ExpirationDays).

Customers can adjust these values during CloudFormation deployment based on retention and compliance requirements.

9.5. Recovery Procedure Overview

In the event of a failure, Here is the Step-by-Step Recovery Procedure

9.5.1 Database Recovery (RDS PostgreSQL):

1. Navigate to AWS Console → RDS → Databases.
2. Select the target database instance.
3. Choose **Restore to point in time** or **Restore from snapshot**.
4. Select the desired restore timestamp or snapshot.
5. Launch a new restored database instance.
6. Update application configuration (if endpoint changes).
7. Validate connectivity from EKS workloads to the restored database.

9.5.2 Application Recovery (EKS Workloads):

1. Verify EKS cluster and node group health.
2. Re-run CloudFormation stack update (or re-trigger deployment runner).
3. Ensure Helm deployment reinstalls all services.
4. Validate all pods are in Running/Ready state.

9.5.3 Secrets Recovery (Secrets Manager):

1. Navigate to AWS Secrets Manager.
2. Restore previous secret version (if available) or recreate secret.
3. Verify IAM roles allow access to the secret.
4. Restart affected workloads if required.

9.5.4 Validation Steps:

- Confirm application UI is accessible.

- Verify API endpoints respond successfully.
- Validate database connectivity and data integrity.
- Check logs in CloudWatch for errors.

9.5.5 Recovery Testing:

- Perform backup and restore testing at least quarterly.
- Validate RPO/RTO objectives.
- Document recovery steps and outcomes for audit/compliance.

9.6. Recommended Operational Practices

For production readiness:

- Define RPO/RTO targets with business stakeholders.
- Test database restore procedures periodically (at least quarterly).
- Confirm monitoring/alerts are configured for database storage, CPU, and connection thresholds.
- Maintain clear ownership for backup/restore execution (customer operations team or managed services team).

9.7. Redis (ElastiCache) Backups

VoxaFoundry provisions an ElastiCache Redis/Valkey component and enables automated backups.

9.7.1 Snapshot Window

- Daily snapshot window: 3 AM – 4 AM.

9.7.2 Backup Retention

- Snapshot retention is configurable during deployment.
- Default retention: 30 days (RedisBackupRetentionPeriod).

---

10. Maintenance

This section provides recommended maintenance practices for operating VoxaFoundry in a customer-owned AWS account. Maintenance activities include platform upgrades, routine operational checks, log and data retention management, and periodic validation of backups and security controls.

10.1. Routine Operational Checks

Perform the following checks on a regular cadence (daily/weekly based on usage):

10.1.1 CloudFormation Stack Status

- Confirm the production stack remains in a stable state (`CREATE_COMPLETE` / `UPDATE_COMPLETE`).

10.1.2 Deployment Runner Visibility

- Confirm CloudWatch logs are available for installation/upgrade runs and no unexpected failures are present.

10.1.3 Load Balancers (ALB/NLB)

- Confirm targets remain healthy and there are no sustained increases in errors or latency.

10.1.4 EKS Cluster Health

- Confirm cluster is Active, and nodes are Ready.
- Confirm key workloads are running and stable (API, Worker, Frontend, LiveKit/TURN).

10.1.5 Database Health (RDS PostgreSQL)

- Confirm CPU, memory, storage, and connection counts remain within expected ranges.

10.1.6 Cache Health (Valkey / ElastiCache Serverless)

- Confirm cache utilization remains within expected limits.

10.2. Platform Updates (Application Upgrades)

VoxaFoundry upgrades are performed through **CloudFormation stack updates**.

- Updates are applied by changing the AppVersion parameter and running a stack update.
- The installation runner performs a Helm upgrade on the EKS cluster to apply the new version.
- Upgrades should be scheduled during a maintenance window appropriate to the customer's operational policy.

10.3. Security Maintenance

Customers should maintain the following security-related controls:

10.3.1 IAM Access Review

- Periodically review IAM users/roles who can update or delete the CloudFormation stack.
- Ensure least privilege is maintained for runtime permissions.

10.3.2 Secrets Rotation (If Applicable)

10.3.2.1 Database Credential Rotation

1. Navigate to **AWS Secrets Manager** in the AWS Console.
2. Locate the database secret used by VoxaFoundry.
3. Update the secret value with the new database username/password or other required fields.
4. Apply the corresponding credential change to the Amazon RDS PostgreSQL database.
5. Confirm the secret JSON structure remains valid and includes all required keys.
6. Restart or redeploy affected application workloads on Amazon EKS so they consume the updated secret.
7. Validate that the application can successfully connect to the database and that no authentication errors appear in CloudWatch logs.
8. After validation, decommission the old credential according to customer security policy.

10.3.2.2 Integration/API Credential Rotation

1. Generate or obtain the new credential from the external provider (for example, AI/model provider or telephony provider).
2. Navigate to **AWS Secrets Manager** and update the relevant secret value.
3. Confirm the secret format remains valid for the application.
4. Restart or redeploy the affected workloads so the updated credential is loaded.
5. Test the integration end-to-end to confirm successful authentication and expected behavior.
6. Review CloudWatch logs for any credential or connectivity errors.
7. Revoke or decommission the old credential after successful validation.

10.3.2.2 TLS Certificate Rotation

1. Check the certificate status in **AWS Certificate Manager (ACM)**.
2. Confirm automatic renewal remains valid for ACM-managed certificates.
3. If a certificate must be replaced manually, import or request the new certificate in ACM.
4. Update the load balancer listener to use the new certificate if required.
5. Validate HTTPS connectivity and confirm the certificate is in **Issued** status.
6. Confirm there are no TLS or browser trust errors after rotation.

10.3.2.2 Restart or Redeploy Affected Workloads

1. Sign in to the **AWS Console**.
2. Navigate to **Amazon EKS** → **Clusters** → select the VoxaFoundry cluster.
3. Open the **Workloads** view and identify the workloads that consume the rotated secret (for example: **API**, **Worker**, and **SIP server**).

4. For each affected workload, trigger a rollout restart by updating the deployment or restarting the pods so the workload reloads the secret from AWS Secrets Manager.
5. Wait until the workload returns to **Running** and **Ready** state.
6. Confirm that the new pods start successfully and that there are no secret or authentication errors in **Amazon CloudWatch Logs**.
7. Validate the dependent function end-to-end:
    - Database login succeeds for database credential rotation
    - External provider authentication succeeds for integration/API key rotation
    - Application UI/API remains healthy after restart

10.3.3 TLS Certificates

- Monitor certificate status in ACM.
- Confirm domain validation remains valid and certificates remain in “Issued” status.

10.3.4 Audit and Logging

- Ensure CloudTrail is enabled for auditing infrastructure and IAM actions.
- Maintain log retention in line with compliance requirements.

10.4. Log Retention and Storage Hygiene

VoxaFoundry generates operational logs and may generate data artifacts depending on enabled features.

10.4.1 CloudWatch Logs

VoxaFoundry configures CloudWatch log retention to control operational logging costs.

- Default log retention: 7 days (LogRetentionDays).
- Deployment logs (installation runner logs) follow the same retention policy.

10.4.2 S3 Lifecycle Management

If S3 is enabled, lifecycle policies are applied automatically:

- Glacier transition after 30 days (S3GlacierTransitionDays).
- Permanent deletion after 60 days total (S3ExpirationDays).

Customers should align these retention settings with internal compliance policies.

10.4.3 Database Data Growth

- Monitor database storage usage trends.
- Plan capacity increases before storage thresholds are reached.

10.5. Backup Validation

Backup readiness should be validated periodically.

- Confirm automated backups for RDS are enabled, and retention is set appropriately.
- Perform scheduled restore tests (snapshot restore or point-in-time recovery) based on customer policy.
- Validate that restored instances can be used for recovery scenarios (connectivity and schema integrity).

10.6. Capacity and Performance Management

VoxaFoundry performance depends on workload patterns (calls, concurrency, integrations).

- Monitor node and pod utilization trends.
- Tune autoscaling settings where required (node groups and Kubernetes workloads).
- Review database and cache utilization during peak periods.
- Validate load balancer behavior under expected traffic volumes.

10.7. Housekeeping and Release Management

- Maintain a record of deployed AppVersion and upgrade history.
- Apply platform updates in a controlled and documented manner.
- Maintain change records for configuration changes (domains, keys, networking).
- Define clear rollback and recovery procedures for production change events.

---

11. Uninstall / Delete

This section describes what happens when the VoxaFoundry CloudFormation stack is deleted and how data retention settings affect deletion behavior.

11.1. Stack Deletion Overview

VoxaFoundry is deployed and managed through a CloudFormation stack. Deleting the stack removes the resources created by CloudFormation, based on the data retention parameters selected at deployment time.

By default, core resources are configured to be removed when the stack is deleted. Some services may still prevent deletion if protection settings are enabled (example: RDS deletion protection).

11.2. Data Retention Controls

11.2.1 RetainDataOnDelete

- When set to `true`, the deployment avoids aggressive cleanup during stack deletion.
- When set to `false`, the deployment performs additional cleanup of Kubernetes persistent resources (PVCs) during deletion.

11.2.2 RDSDeletionProtection

- When set to true (default), the RDS database cannot be deleted until deletion protection is disabled.
- If the stack is deleted while deletion protection is enabled, the stack deletion may pause/fail until deletion protection is removed.

11.2.3 RDSBackupRetentionPeriod

- Automated backups remain available for the configured retention period, subject to the database lifecycle and account policies.

11.2.4 S3 Lifecycle (if S3 is enabled)

- Objects are permanently deleted after S3ExpirationDays.
- Objects transition to Glacier after S3GlacierTransitionDays.

11.2.5 RedisBackupRetentionPeriod

- Redis snapshots are retained for the configured number of days.
- Snapshot window is configured daily (3 AM – 4 AM).

11.3. Recommended Deletion Process

1. Confirm you have exported or retained anything required for compliance or auditing.
2. If required by your organization, take an RDS snapshot before deletion.
3. If the stack must fully delete the database, disable RDS deletion protection before deleting the stack.
4. Delete the CloudFormation stack from the AWS Console.

11.4. Expected Outcome After Stack Deletion

After stack deletion, expected results depend on retention settings:

- Application services in EKS are uninstalled as part of the deletion workflow.
- If `RetainDataOnDelete=false`, Kubernetes PVCs may also be deleted.
- If `RDSDeletionProtection=true`, the database will remain until deletion protection is disabled.
- The S3 bucket will be removed with the stack unless organizational policies prevent deletion.

---

12. Emergency Maintenance

This section guides handling emergencies after deployment. It focuses on fault conditions that may impact availability and the software recovery steps to restore VoxaFoundry services and critical data using AWS managed capabilities.

12.1. Fault Conditions Table

Fault / Symptom	Where to Check First	What to Confirm
Platform UI is not reachable	EC2 Console → Load Balancers (ALB); Target Groups	ALB is active, listener 443 exists, target group targets are Healthy
API returning 5XX / UI cannot load data	ALB Target Group (API); CloudWatch Logs; RDS Monitoring	API targets Healthy, API logs show errors, RDS not saturated (CPU/connections/latency)
DNS resolves incorrectly / endpoints are not resolving	Route 53 → Hosted Zone	Records exist for app/api/livekit/turn and point to the correct load balancer targets
TLS/HTTPS issues	ACM → Certificates; ALB listeners	Certificate is Issued, DNS validation records exist, ALB has HTTPS/443 configured
Authentication/login issues	Cognito → User Pool; CloudWatch Logs	Admin user exists, UI/API domains correct, auth/token errors visible in logs
Voice/media connectivity issues (LiveKit/TURN)	NLB (if used); EKS workloads	NLB is active, LiveKit/TURN pods are Running, TURN domain resolves and reachable
Pods restarting / not running	EKS → Workloads; Pod events/logs	CrashLoopBackOff/ImagePullBackOff/Pending causes identified (capacity/secrets/dependencies)
Database connectivity issues	RDS → Connectivity & Security; RDS Monitoring; app logs	DB available, reachable from VPC, app uses correct endpoint, no access denied

Secrets/configuration errors	Secrets Manager; app logs	Secret exists, JSON structure includes `username/password/host/dbname`, no access denied

12.2. Software Recovery Table

Component	Recovery Action	Notes
Amazon RDS (PostgreSQL)	Restore from latest automated backup/snapshot or use PITR within retention window	Verify restored DB is reachable from private subnets; update endpoint/config if endpoint changes
AWS Secrets Manager	Restore a previous version (if available) or recreate secret using expected JSON schema	Ensure runtime roles can read the secret; validate pods start after restoration
ElastiCache (Valkey/Redis)	Verify service status; confirm networking allows access from EKS private subnets	Use AWS-managed recovery/snapshot capabilities as configured
Amazon Cognito	Verify User Pool exists; recreate admin user if required	If recreated, update app configuration to reference new pool/client details
Amazon EKS (Application Services)	Confirm cluster/node groups healthy; re-run CloudFormation stack update to reinstall/repair Helm release	Re-apply the same AppVersion to trigger runner repair
Route 53 and ACM	Recreate missing DNS records; reissue certificate if needed and allow DNS validation	Confirm certificate returns to Issued and ALB uses correct certificate
Amazon S3 (Optional)	Restore objects using versioning (if enabled) or customer-defined backup/replication policy	Ensure bucket policy and IAM allow access patterns

13. Support and SLAs

Globant provides application-level support for the **VoxaFoundry** solution with a standard **24-hour response time**. Customers can reach the support team through the **Globant Enterprise AI Help Center** or the support team at voxafoundry@globant.com, provided as part of the customer engagement.

Enterprise AI Help Center:
🔗 https://www.globant.com/globant-enterprise-ai/help-center

13.1. Support Ownership

VoxaFoundry is deployed in a **customer-owned AWS account**, so operational responsibilities are shared between the customer and Globant.

13.1.2 Customer Responsibilities

- Own and manage the AWS account, including billing, service quotas, and account governance.
- Own and operate AWS infrastructure used by the deployment, including:
    - VPC, subnets, route tables, security groups, NAT gateways
    - Amazon EKS cluster and node groups
    - Load balancers (ALB/NLB)
    - Amazon RDS (PostgreSQL), backups, retention, and restore operations
    - AWS Secrets Manager secret lifecycle and rotation policies
    - Amazon Route 53 hosted zone and domain ownership (BaseDomain/subdomains)
- Manage external provider accounts and related costs, including:
    - Telephony/PSTN/SIP provider (if integrated)
    - Any third-party AI/model providers (if used)

13.1.3 Globant Responsibilities

Provide deployment documentation and guidance for CloudFormation-based installation and upgrades.

- Provide platform configuration guidance and recommended best practices.
- Support troubleshooting for:
    - Installation runner / deployment failures
    - Application runtime behavior (UI/API/Worker)
    - Authentication flow and application-level access issues
    - Live media plane behavior and configuration (LiveKit/TURN)
    - AI/voice integrations within the application (where enabled)
- Provide supported application releases and version updates.

13.1.4 Shared Responsibilities

- Coordinate release planning, upgrade scheduling, and rollout validation.
- Jointly troubleshoot environment-specific constraints (enterprise network controls, firewall/proxy restrictions, DNS policies).
- Validate security and compliance requirements and confirm the deployment meets customer controls and governance.

13.2. AWS Support Requirements

A **Basic AWS Support plan** is sufficient to deploy and operate **VoxaFoundry**.

Globant recommends **Business or Enterprise Support** for production workloads to receive 24×7 AWS assistance for:

- EKS cluster and node group issues
- CloudFormation provisioning
- VPC, DNS, and load balancer issues
- IAM permission troubleshooting
- RDS availability and database operational events
- Service quota increases

AWS Support covers the underlying AWS platform; Globant Support covers the **VoxaFoundry** application layer.

13.3. Globant Support SLAs

Support Category	Description	SLA / Response Target	Availability
General Support Requests	Deployment help, configuration questions, and non-urgent issues.	Response within 24 hours.	Monday–Friday (Business Hours)
Production-Impacting Issues	Errors affecting core VoxaFoundry features (e.g., login failures, call flow failures, API timeouts, worker processing errors).	Response within 24 hours.	24×5 (Weekdays)
Critical Service Outage	VoxaFoundry is completely unavailable in the customer's AWS account.	Response within 24 hours.	24×5 (Weekdays)
AI / Voice Integration Issues	Problems related to configured AI/voice providers or service keys used by VoxaFoundry (where enabled).	Response within 24 hours.	Monday–Friday
Feature Requests	Enhancements or non-urgent product suggestions.	Reviewed within 5 business days.	Monday–Friday
Support Channels	Globant Enterprise AI Help Center and the support channel provided during the engagement.	Submit tickets anytime.	24×7 ticket submission

14. Appendices

14.1. CloudFormation Execution Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManagedResourceProvisioning",
            "Effect": "Allow",
            "Action": [
                "eks:*",
                "ec2:*",
                "bedrock:*",
                "s3vectors:*",
                "rds:*",
                "elasticache:*",
                "s3:*",
                "cognito-idp:*",
                "codebuild:*",
                "lambda:*",
                "secretsmanager:*",
                "acm:*",
                "logs:*",
                "route53:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IAMDelegation",
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "iam:PassRole",
                "iam:TagRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/*-eks-cluster-role",
                "arn:aws:iam::*:role/*-eks-node-role",
                "arn:aws:iam::*:role/*-codebuild-role",
                "arn:aws:iam::*:role/*-lambda-execution-role"
            ]
        }
    ]
```

```
}
```

NOTE:
The CloudFormation Execution Role is assigned Allow on infrastructure service wildcards to facilitate automated stack updates and deletions across the complex EKS/VPC surface area. However, the role is denied the ability to escalate privileges because all iam actions and PassRole operations are strictly scoped to the specific ARNs of the Voxafoundry service roles, preventing any modification of core account security or unauthorized identity creation.

14.2. EKS Cluster Control Plane Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs",
                "elasticloadbalancing:*"
            ],
            "Resource": "*"
        }
    ]
}
```

14.3. EKS Node Instance Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ContainerAndLogging",
            "Effect": "Allow",
```

```
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AppDataAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:s3:::*-resource-bucket",
                "arn:aws:s3:::*-resource-bucket/*",
                "arn:aws:secretsmanager:*:*:secret:*-db-credentials-*"
            ]
        }
    ]
}
```

14.4. CodeBuild Runner Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EksAndEcrAccess",
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
```

```
                "ecr:BatchGetImage"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DnsAndLbDiscovery",
            "Effect": "Allow",
            "Action": [
                "route53:ChangeResourceRecordSets",
                "route53:ListHostedZones",
                "elasticloadbalancing:DescribeLoadBalancers",
                "cloudformation:SignalResource"
            ],
            "Resource": "*"
        }
    ]
}
```

14.5. Lambda Trigger Execution Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CodeBuildTrigger",
            "Effect": "Allow",
            "Action": "codebuild:StartBuild",
            "Resource": "*"
        },
        {
            "Sid": "CognitoAdmin",
            "Effect": "Allow",
            "Action": [
                "cognito-idp:AdminCreateUser",
                "cognito-idp:AdminSetUserPassword"
            ],
            "Resource": "arn:aws:cognito-idp:*:*:userpool/*"
        },
        {
            "Sid": "Logging",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

14.6. S3 Bucket Resource Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "EnforceHttpsOnly",
           "Effect": "Deny",
           "Principal": "*",
           "Action": "s3:*",
           "Resource": [
               "arn:aws:s3:::*-resource-bucket",
               "arn:aws:s3:::*-resource-bucket/*"
           ],
           "Condition": {
               "Bool": {
                   "aws:SecureTransport": "false"
               }
           }
       },
       {
           "Sid": "AllowVoxafoundryAccess",
           "Effect": "Allow",
           "Principal": {
               "AWS": "arn:aws:iam::*:role/*-eks-node-role"
           },
           "Action": [
               "s3:PutObject",
               "s3:GetObject",
               "s3:ListBucket"
           ],
           "Resource": [
               "arn:aws:s3:::*-resource-bucket",
               "arn:aws:s3:::*-resource-bucket/*"
           ]
           }
       ]
}
```

14.7. Secrets Manager Resource Policy

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "RestrictToVoxafoundryNodes",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::*:role/*-eks-node-role"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        },
        {
            "Sid": "DenyExternalAccess",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*",
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalArn": [
                        "arn:aws:iam::*:role/*-eks-node-role",
                        "arn:aws:iam::*:role/*-lambda-execution-role",
                        "arn:aws:iam::*:role/*-codebuild-role"
                    ]
                }
            }
        }
    ]
}
```