

AWS EKS Deployment Toolkit

Toolkit version 1.0.0. Publication date: 15 October 2026. This archive supplies deployment scripts, configuration templates, and installation/operations guidance. Obtain registry credentials and licensing separately; provision customer AWS infrastructure before deployment.

Platform compatibility

Toolkit version	Target AICC platform release	Approved Helm chart version	Compatibility status
1.0.0	8.3.121	Pending Engineering confirmation	AWS deployment validation pending

Confirm exact image tags and the approved chart version with IntelliBuddies before deployment. Platform images and the OCI chart have independent registry versions; registry availability does not establish compatibility. Record the tested combination and image digests. Toolkit versions follow MAJOR.MINOR.PATCH independently of platform/chart versions: compatible fixes increment PATCH, compatible functionality increments MINOR, and breaking changes increment MAJOR.

Contents

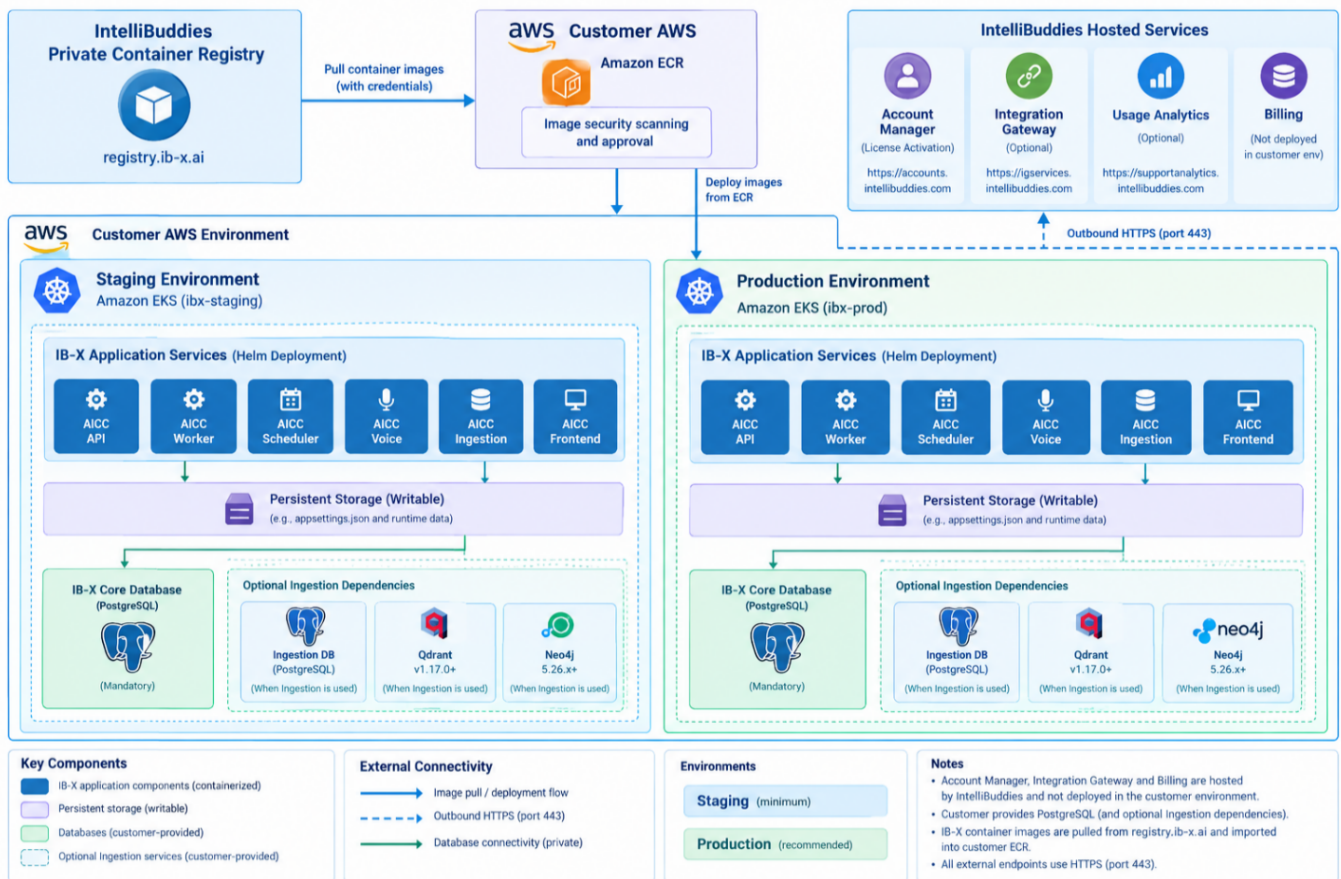
- [Deployment overview](#)
- [Deployment prerequisites](#)
- [Installation](#)
- [Deployment verification](#)
- [Enterprise operations](#)
- [Troubleshooting](#)

Deployment overview

Deploy the IB-X AI Command Center (AICC) and its associated services in your AWS environment using Amazon Elastic Kubernetes Service (EKS). IntelliBuddies supplies prebuilt container images and a Helm deployment package. Your Infrastructure and DevOps teams provision and operate the AWS infrastructure and deploy approved images from Amazon Elastic Container Registry (ECR).

Deployment architecture

This diagram shows the wider deployment landscape: separate Staging and Production environments, customer infrastructure, IntelliBuddies-hosted services, and the associated connections.

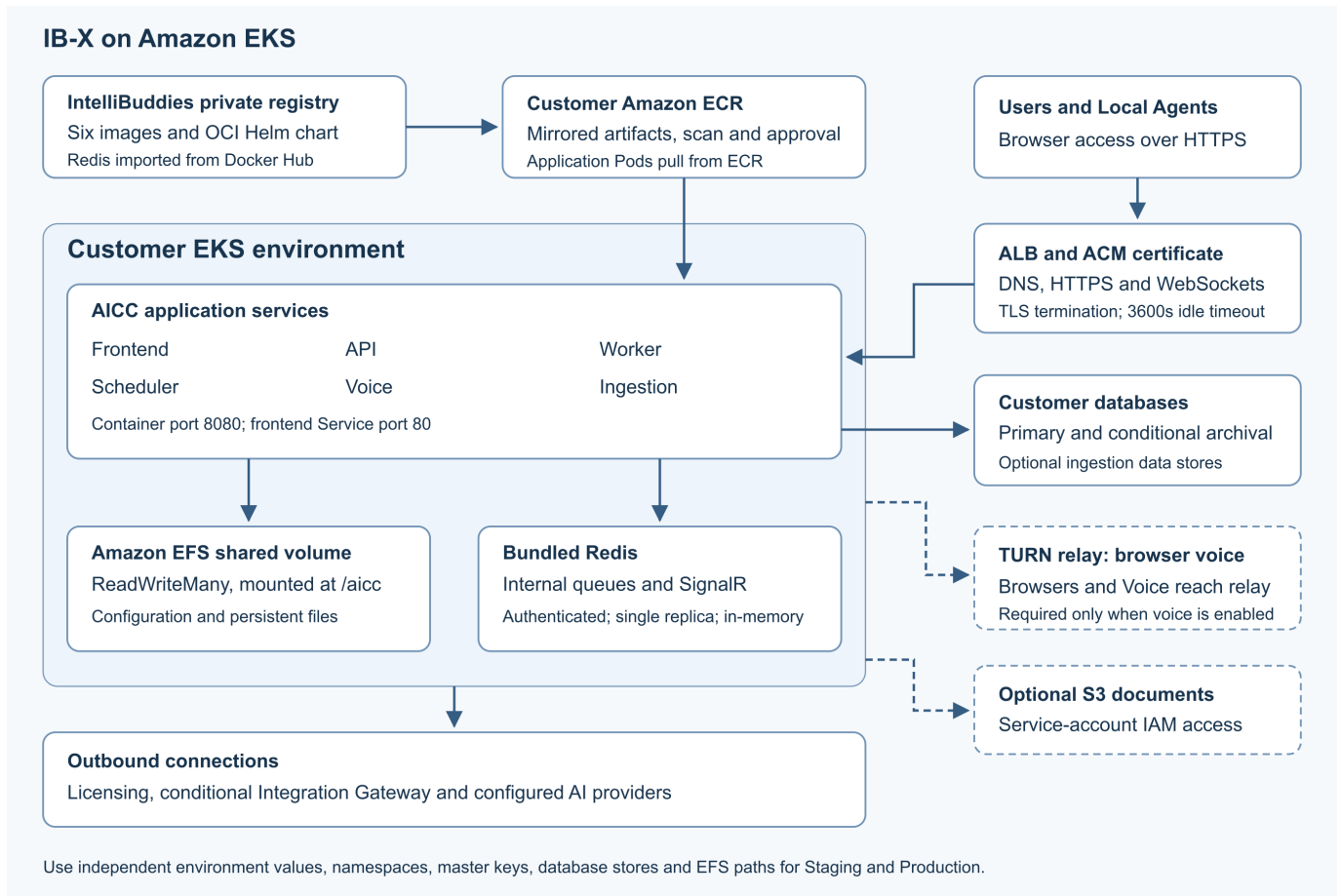


IB-X deployment landscape with Staging, Production, customer infrastructure, and IntelliBuddies-hosted services

Maintain separate Staging and Production environments. You may use independent EKS clusters or separate namespaces on a shared cluster according to your infrastructure standards. Isolate application configuration, secrets, databases, and persistent storage for each environment.

Runtime components and connectivity

This diagram expands the runtime dependencies within one environment: ALB and ACM, the six application services, bundled Redis, shared EFS storage, customer databases, and optional TURN and S3. Apply the same structure independently to Staging and Production.



IB-X EKS runtime with ECR, ALB, Redis, EFS, and optional TURN and S3

Application components

The deployment package includes the following services. The supplied configuration uses one replica per service; additional scaling and high availability require separate validation.

Component	Kubernetes resource	Default replicas	Container port
AICC API	Deployment	1	8080
AICC Worker	Deployment	1	8080
AICC Scheduler	Deployment	1	8080
AICC Voice	Deployment	1	8080
AICC Ingestion	Deployment	1	8080
AICC Frontend	Deployment	1	8080
Redis	Deployment and internal ClusterIP Service	1	6379

Supporting resources provide networking, configuration, secrets, and persistent storage.

Redis

Redis is mandatory for work queues and the SignalR backplane. Keep `redis.enabled: true`. The supplied release supports bundled Redis only; external services such as Amazon ElastiCache are not supported by this deployment configuration.

Redis uses password authentication, one replica, and in-memory storage. It is not a high-availability or durable messaging service. Services reconnect after a restart, but temporary messages and notifications can be interrupted or lost. Persisted workflow state remains in the application database; investigate affected operations before retrying them.

Mirror Redis into customer ECR alongside the six application images and OCI chart. The deployment toolkit (available in the documentation portal) contains scripts, configuration templates, and supporting guidance; licensed images and the chart are retrieved separately from the IntelliBuddies registry using release credentials.

Scope

This guide covers the customer-hosted AI Command Center platform. Requirements for customer-specific solutions and their integrations are documented separately.

Local Agents that perform desktop or UI automation require a separate Windows environment with Process Designer and Smart Station. These Windows components are outside the EKS deployment and are required only when Local Agent capabilities are used.

Responsibilities

Activity	IntelliBuddies	Customer Infrastructure and DevOps
Container images and release references	Provide and maintain	Import into ECR, scan, and approve
Helm deployment package	Provide package and configuration guidance	Deploy with agreed support
EKS compute and networking	Define application prerequisites	Provision, secure, and operate
Databases and optional ingestion dependencies	Provide connectivity requirements	Provision, back up, and maintain
Persistent storage	Define mount and access requirements	Provision and maintain
DNS, ingress, TLS, and outbound connectivity	Identify application requirements	Configure and manage
License activation	Provide licensing and activation guidance	Enable connectivity and coordinate activation
Deployment verification	Support application validation	Validate infrastructure and coordinate acceptance

Next steps

1. Review the [infrastructure prerequisites](#).
2. Obtain the deployment toolkit (available in the documentation portal) and follow the [installation steps](#).
3. Complete the [deployment verification checklist](#).
4. Establish [enterprise operating procedures](#) before opening Production.
5. Use [Troubleshooting](#) when a prerequisite or acceptance check fails.

Deployment prerequisites

Prepare infrastructure separately for Staging and Production. The [overview](#) defines deployment scope and responsibilities. The installation steps use the supplied scripts and configuration templates.

Application capacity and compute

The application baseline follows the Medium Deployment category in the AI Command Center system requirements (available in the documentation portal).

Resource	Staging minimum	Production recommended
CPU	8 cores, 2.5 GHz or faster	12 cores, 2.5 GHz or faster
RAM	16 GB	24 GB
Application disk	100 GB	250 GB
Network	1 Gbps	1 Gbps

These are aggregate application capacity figures, not per-node specifications or Helm allocations. Account for Kubernetes overhead and supporting workloads. Database and ingestion infrastructure are separate. Measure workload concurrency, voice sessions, and storage growth when selecting Production capacity.

Use Linux `amd64` (`x86_64`) worker nodes; ARM/Graviton application nodes are not supported by the supplied images. Two or more Ready nodes are recommended. Set CPU/memory requests and limits for each applicable service; application allocations are not supplied by default. Multiple nodes alone do not provide application high availability, and bundled Redis remains a single replica.

Infrastructure checklist

Requirement	Preparation
EKS	Operational cluster, suitable nodes, cluster access and workload-management permissions
ECR	Repositories for six application images, Redis, and the OCI Helm chart; import/push rights and node pull access
ALB	AWS Load Balancer Controller with required AWS permissions and appropriate subnet/security-group configuration
DNS and TLS	Environment hostname and an Issued ACM certificate covering it in the ALB region
EFS	Encrypted file system, mount targets in worker-node Availability Zones, TCP 2049 access, EFS CSI driver and controller IAM permissions
StorageClass	<code>efs.csi.aws.com</code> , access-point provisioning, UID/GID <code>10001</code> , reclaim policy <code>Retain</code>
Secrets	Independent master key per environment, secure backup outside the cluster, and TURN credential if used
Databases	Reachable primary database and conditional archival/ingestion stores
TURN	Reachable relay and sufficient bandwidth when browser voice is enabled
S3	Optional document bucket, IRSA role, and bucket/object permissions

The supplied configuration uses ALB target type `ip`, backend frontend Service port 80, and application container port 8080. TLS ends at the ALB. Set idle timeout to at least 3600 seconds, preserve forwarded client information, and support WebSockets and uploads up to the application's 1 GB limit. Validate large uploads through the complete ingress path. End-to-end TLS requires separate design and compatibility validation.

Tools and access

Run the scripts from Linux, macOS, WSL, Git Bash, or an appropriately configured CloudShell session. Verify tools rather than assuming they are preinstalled:

- AWS CLI with credentials for the intended account.
- `kubectl` configured for the cluster and compatible with its Kubernetes version.
- Helm 3.8 or newer for OCI chart support.
- Bash and common utilities, including `curl`, `tar`, `gzip`, `openssl`, and a SHA-256 tool.
- `unzip` when extracting the package or bootstrapping ORAS on Git Bash.

The mirror script uses an installed ORAS binary or downloads a pinned, checksum-verified binary. Permit its GitHub release download or preinstall an approved ORAS version. A private cluster also requires an operator environment with network access to its API endpoint.

Separate ECR import/push permissions, application deployment permissions, and runtime permissions. Deployment uses AWS read access for ECR/ACM checks, Helm registry authentication, and Kubernetes permissions for the release resources and Secrets. Applying the StorageClass requires cluster-level permission. The optional preflight database check also needs diagnostic-Pod create/attach/delete permissions.

Databases and setup

The Customer Infrastructure team provisions, secures, backs up, and operates databases. This guide uses PostgreSQL; the package recommends PostgreSQL 15 as a starting point. Obtain release compatibility guidance for the selected database version. IB-X also supports SQL Server for its primary database with the corresponding release-specific configuration.

Store	Technology	Requirement
Primary	PostgreSQL baseline	Required; platform data and execution history
Execution history archival	PostgreSQL	Required when archival is enabled
Ingestion	PostgreSQL	Required when Ingestion is enabled
Vector	Qdrant v1.17.0 or above	Required when Ingestion is enabled
Graph	Neo4j Server 5.26.x or above	Required when Ingestion is enabled

Use separate stores and credentials per environment. The primary database connection is entered through the first-run wizard, not Helm values. The database account must initialize and modify the application schema and, where required, create the database. Archive policies may move historical records to a separate database or permanently delete them; configure retention deliberately.

Redis is bundled, mandatory, password-authenticated, and internal to the cluster. Keep it enabled and mirror its image to ECR. External Redis/ElastiCache is not supported by this deployment configuration. Its in-memory messaging state requires separate consideration from the persistent application database.

Persistent configuration

All application services share writable EFS storage using `ReadWriteMany` at `/aicc`. An EBS single-writer volume is not a substitute for this shared mount. The supplied EFS access point uses UID/GID 10001 and directory permissions 775. Configure the EFS CSI controller through EKS Pod Identity or IRSA with the appropriate EFS permissions.

Create a mount target in each worker-node Availability Zone, not each individual subnet. Regional EFS supports one mount target per Availability Zone. See [AWS EFS mount-target guidance](#).

The default 5Gi PVC request does not cap EFS capacity. Isolate environments with separate file systems or access-point paths. A new volume need not be prepopulated: the wizard creates `/aicc/data/appsettings.json`. Back up configuration, documents, and other required persistent paths along with the matching master key. `Retain` protects backing storage from automatic reclamation; it is not a backup or a complete reinstall procedure.

Network connectivity

Use the following flows when planning security groups, firewalls, proxies, and network policies. Ports are defaults; match actual dependency endpoints and TURN configuration.

Source	Destination	Protocol/port	Purpose
Browsers and applicable Windows Local Agents	ALB	TCP 443	UI, APIs and WebSockets
ALB	Frontend Pods	TCP 8080	HTTP after TLS termination
Application workloads	Primary/archival PostgreSQL	TCP 5432	Database access
Application workloads	Bundled Redis	TCP 6379	Queues and SignalR
Ingestion	PostgreSQL, Qdrant, Neo4j	TCP 5432, 6333/6334, 7687	Conditional ingestion
Worker nodes	EFS mount targets	TCP 2049	Shared storage
Worker nodes	ECR and required AWS image-pull dependencies	TCP 443	Image pulls
Browsers and Voice	TURN	UDP/TCP 3478, TLS 5349 if configured, configured relay range	Conditional voice media
Voice and Worker	Configured AI providers	HTTPS 443 or provider-defined endpoints	Speech and language services
Mirror host	IntelliBuddies registry, ECR, Docker Hub, ORAS download host	HTTPS 443	Artifact import and tooling

Permit internal Kubernetes service traffic and cluster DNS. Confirm the Service ports and container target ports for the supplied chart; application container port 8080 and frontend Service port 80 are distinct.

IntelliBuddies endpoint	Purpose	Requirement
<code>https://accounts.intellibuddies.com</code>	License activation	Required
<code>https://registry.ib-x.ai</code>	Artifact import and updates	Required from the import host
<code>https://igservices.intellibuddies.com</code>	Integration Gateway	When corresponding services are used
<code>https://supportanalytics.intellibuddies.com</code>	Integration Gateway analytics	When corresponding services are used

Application Pods pull images from customer ECR. Direct image-pull access to the IntelliBuddies registry is unnecessary. Additional integrations need their own approved connectivity.

Browser voice

TURN is required when browser-based voice is enabled; non-voice deployments do not need it. The supplied deployment relays browser voice media through TURN. Provide browser and Voice-workload connectivity, transport/relay-port access, dedicated credentials, and bandwidth for concurrent calls. HTTPS is required for browser microphone access. The deployment guide uses Helm environment settings and a Kubernetes credential Secret.

Licensing

Activate a license for each environment before use. Enable required licensing connectivity and retain the licensing information supplied by IntelliBuddies. See [License Activation](#) (available in the documentation portal) for general guidance; complete activation through the EKS deployment's setup flow.

Installation

Prepare the [prerequisites](#), install and validate Staging, then promote the approved release to Production. The scripts deploy into an existing cluster; they do not provision EKS, EFS, the database, or the load balancer controller.

1. Download and connect

Obtain and verify the deployment toolkit (available in the documentation portal). It contains the required scripts and templates plus supporting guidance. Extract it, change into AICC-EKS-Package, and make its scripts executable:

```
chmod +x scripts/*.sh
```

Define the environment in this terminal session. Replace the example region and cluster name:

```
export AWS_REGION="eu-west-1"
CLUSTER="your-eks-cluster"
NAMESPACE="aicc-staging"
RELEASE="aicc"
VALUES="values-staging.yaml"
aws sts get-caller-identity
aws eks update-kubeconfig --region "$AWS_REGION" --name "$CLUSTER"
kubectl config current-context
kubectl get nodes
```

Confirm the account, cluster, and Ready amd64 nodes before changing anything. Production must use independent values, databases, storage paths, Secrets, and licensing. Supply namespace and release flags consistently; the scripts otherwise default to `aicc` for both.

2. Configure EFS storage

Edit `config/storageclass-efs.yaml` and replace `<FILE_SYSTEM_ID>` with your encrypted EFS file system ID. Configure the EFS CSI driver and its IAM permissions first. Create a mount target in each worker-node Availability Zone and allow TCP 2049 from the nodes.

```
kubectl apply -f config/storageclass-efs.yaml
kubectl get storageclass efs-aicc -o yaml
```

Check provisioner `efs.csi.aws.com`, reclaim policy `Retain`, and access-point provisioning with `UID/GID 10001`. The `StorageClass` is cluster-scoped: verify an existing class rather than overwriting it. Use distinct class names where environments use different EFS file systems.

The application mounts shared storage at `/aicc` and requires `ReadWriteMany`. The 5Gi PVC request does not impose an EFS capacity limit. A new volume starts empty; the setup wizard creates the configuration. Do not copy another environment's `appsettings.json` into it.

3. Mirror release artifacts into ECR

Toolkit 1.0.0 targets AICC platform 8.3.121; AWS deployment validation and the approved Helm chart version remain pending. Obtain the registry project, username/token, exact image release tag, chart version, and license from IntelliBuddies through the agreed delivery process before deployment. Confirm the compatibility information (available in the documentation portal). The ZIP contains scripts and templates, not licensed images or a Helm archive.

Set these to the exact versions confirmed for AICC 8.3.121. Do not infer the chart version from the platform version:

```
IMAGE_TAG="REPLACE_WITH_APPROVED_IMAGE_TAG"
CHART_VERSION="REPLACE_WITH_APPROVED_CHART_VERSION"
./scripts/aicc-mirror-to-ecr.sh \
  --source registry.ib-x.ai/your-project \
  --tag "$IMAGE_TAG" --chart-version "$CHART_VERSION" --region "$AWS_REGION" --dry-run
```

Review the plan, then repeat without `--dry-run`. The script prompts for registry credentials without echoing the token. For automation, supply `HARBOR_USERNAME` and `HARBOR_TOKEN` through an approved secrets mechanism.

Artifact	Default ECR repository
Six application services	aicc/ib-api, aicc/ib-worker, aicc/ib-scheduler, aicc/ib-voice, aicc/ib-ingestion, aicc/ib-frontend
Redis	aicc/redis
OCI Helm chart	aicc/aicc

The script creates repositories unless `--no-create-repos` is specified. Its default prefix is `aicc`. Retain its final image-registry, Redis-image, and chart-reference output. All six application images use one release tag. A four-part image tag such as `8.3.121.1` maps to chart version `8.3.121-1`; these are format examples, not a compatibility statement. Supply `--chart-version` if IntelliBuddies specifies a different version.

Scan and approve the mirrored images before installing. Do not skip Redis or the chart for a new installation. The mirror host needs the source registry, ECR, Docker Hub for Redis, and GitHub release downloads if it needs to obtain the pinned ORAS binary.

4. Create the namespace and master key

```
kubectl create namespace "$NAMESPACE"
```

If it already exists, verify it belongs to this environment. This walkthrough uses a customer-created Secret referenced by `global.masterKey.existingSecret`. Generate a key only for a new environment; an existing installation must retain its matching key.

```
umask 077
KEY_DIR=$(mktemp -d)
openssl rand -hex 32 | tr -d '\n' > "$KEY_DIR/master-key"
kubectl -n "$NAMESPACE" create secret generic aicc-master-key \
  --from-file=master-key="$KEY_DIR/master-key"
```

Store this key securely outside the cluster using your approved secrets-management process. After confirming that backup, remove the temporary file:

```
rm -- "$KEY_DIR/master-key"
rmdir -- "$KEY_DIR"
unset KEY_DIR
```

Do not print, lose, replace, or routinely rotate the master key. Losing it prevents decryption of saved configuration. Use separate keys per environment and preserve them through upgrades and recovery.

The chart can alternatively generate `<release>-aicc-master-key` when no existing Secret is configured. Do not mix that alternative with this walkthrough's `aicc-master-key` reference. Back up the actual key in use. Namespace deletion removes Secrets even where Helm retention applies.

5. Prepare values

```
cp config/values-eks.yaml "$VALUES"
```

Replace every active placeholder. Remove the optional `global.objectStore` and `global.serviceAccount` blocks if S3 is not used. Keep passwords out of this file.

Setting	Value
<code>global.imageRegistry</code>	Registry and prefix printed by the mirror script
<code>global.masterKey.existingSecret</code>	<code>aicc-master-key</code> from step 4
Six <code>aicc-*.image.tag</code> entries	Same approved release tag
<code>redis.enabled</code> , <code>redis.image</code>	<code>true</code> , and the mirrored ECR Redis image
<code>sharedVolume.storageClass</code> , <code>accessMode</code>	EFS StorageClass and ReadWriteMany

Setting	Value
dnsResolver	Leave empty for cluster DNS discovery
ingress.host	This environment's HTTPS hostname
Certificate annotation	Issued ACM certificate for that hostname, in the ALB region
Ingress scheme	Explicitly choose internal or internet-facing
ALB settings	Class alb, target type ip, HTTPS redirect, /index.html health check, idle timeout 3600 seconds
Service resources	Measured CPU/memory requests and limits for each applicable service

The script revision supplied here reads tags from the template's one-line service entries. Preserve that format when adding resources or replicas:

Add the measured resource requests and limits inside the same service entry, preserving the one-line image-tag format required by this script revision.

```
# Keep the service image tag on one line.
aicc-api: { image: { tag: "YOUR_RELEASE_TAG" }, replicaCount: 1 }
```

Keep ingress routed through the frontend service, which routes the API, chat, and voice paths. Do not add a separate /api ingress rule. If knowledge bases are not used, ingestion can be disabled by adding `replicaCount: 0` to the one-line `aicc-ingestion` entry; otherwise retain it.

Configure all applicable services. The chart does not supply application resource requests and limits by default. Do not convert these service entries into multiline YAML with this script revision.

Optional S3 documents

Configure `global.objectStore` and the service-account role annotation from the template. The supplied configuration uses IRSA and needs the appropriate bucket/object permissions to list, read, write, and delete documents. Provision the role and trust relationship first. S3 does not replace EFS for configuration, and its content requires separate backup.

Browser voice and TURN

Provide a relay reachable from browsers and the Voice workload. Create its credential Secret without placing the password in values:

```
umask 077
TURN_DIR=$(mktemp -d)
read -r -s -p "TURN credential: " TURN_CREDENTIAL
printf '\n'
printf '%s' "$TURN_CREDENTIAL" > "$TURN_DIR/credential"
unset TURN_CREDENTIAL
kubectl -n "$NAMESPACE" create secret generic aicc-turn \
  --from-file=credential="$TURN_DIR/credential"
rm -- "$TURN_DIR/credential"
rmdir -- "$TURN_DIR"
unset TURN_DIR
```

Uncomment the complete `global.extraEnv` block in the template and replace the TURN hostname and username. Configure all three ICE URL entries: STUN, TURN over UDP, and TURN over TCP. Credentials reference `aicc-turn`. Match the transports and ports to the relay; where TCP is unavailable, use the supported UDP URL for the third entry as described by the package.

This walkthrough uses Helm environment overrides. Verify the effective settings in Voice logs; do not assume administration settings override deployment values. Use a dedicated TURN account and sufficient relay bandwidth. Its credentials are supplied to browsers for media relay.

6. Run preflight

```
./scripts/aicc-preflight.sh --values "$VALUES" --namespace "$NAMESPACE" \
  --chart-version YOUR_CHART_VERSION
```

Resolve every [FAIL] and review each [WARN]. Checks cover active placeholders, tags, cluster access, architecture, storage class, ingress class, master key, ECR artifacts, and ACM status. Skipped AWS checks are not passed prerequisites. Preflight does not establish application readiness, capacity, or complete compatibility.

Optionally check database TCP reachability:

```
./scripts/aicc-preflight.sh --values "$VALUES" --namespace "$NAMESPACE" \
  --chart-version YOUR_CHART_VERSION --db your-db-host:5432
```

With `--db`, preflight creates and removes a diagnostic Pod using the mirrored Redis image. It needs permission to create, attach to, and delete the Pod. This verifies TCP reachability, not credentials or schema permissions.

7. Install and configure DNS

```
./scripts/aicc-eks.sh install --values "$VALUES" \
  --namespace "$NAMESPACE" --release "$RELEASE" \
  --chart-version YOUR_CHART_VERSION --dry-run
```

Review the output, then repeat without `--dry-run`. The script runs preflight, requests confirmation, authenticates Helm to ECR, and waits for workloads; the default timeout is 15 minutes. Dry-run output may include Secrets: protect it. A rehearsal still performs read checks and registry authentication.

```
kubectl -n "$NAMESPACE" get ingress "$RELEASE"
```

Point the hostname to the ALB address using a suitable DNS alias or CNAME. Check the ingress scheme, certificate, and security groups, then test HTTPS from an authorized network.

8. Complete first-run setup

Open the HTTPS hostname and complete the wizard:

1. Choose PostgreSQL, the baseline for this guide. The product also supports SQL Server with release-specific configuration guidance.
2. Enter a cluster-reachable database endpoint, port, name, and credentials. Never use `localhost`. Use lowercase names; the wizard normalizes names to lowercase with underscores.
3. Use an account permitted to initialize and modify the schema. It can create a missing database when granted the necessary rights; otherwise supply an empty database with the required ownership.
4. Create the initial administrator and activate the environment's license.

The wizard writes encrypted configuration to `/aicc/data/appsettings.json`; services consume it without a routine manual restart. Primary database credentials are not Helm values. Before setup, `/api/setup/status` may return HTTP 400; use the wizard to establish setup status rather than treating that response alone as failure.

9. Verify and promote

```
./scripts/aicc-eks.sh status --namespace "$NAMESPACE" --release "$RELEASE"
```

Complete the [verification checklist](#), take configuration/database backups, and record Staging acceptance. Deploy the same approved artifacts to Production with independent configuration, databases, storage, Secrets, and licensing. Complete Production setup and acceptance before opening it for use.

Use [Enterprise Operations](#) for subsequent changes and [Troubleshooting](#) for failed checks.

Deployment verification

Complete these checks in Staging before promoting a release. Record the account, cluster, namespace, release, chart/image versions, results, and acceptance owner. Use the same environment variables as the [installation guide](#):

```
NAMESPACE="aicc-staging"
RELEASE="aicc"
VALUES="values-staging.yaml"
kubectl config current-context
./scripts/aicc-eks.sh status --namespace "$NAMESPACE" --release "$RELEASE"
kubectl -n "$NAMESPACE" get pods -l "app.kubernetes.io/instance=$RELEASE"
kubectl -n "$NAMESPACE" get pvc
kubectl -n "$NAMESPACE" get ingress "$RELEASE"
kubectl -n "$NAMESPACE" get events --sort-by=.lastTimestamp
```

Infrastructure and release

Check	Acceptance
Artifacts	Six application images, Redis, and matching OCI chart exist in customer ECR; image scanning is approved
Deployments	Ready replicas match desired replicas; Pods are Running and Ready, without persistent restart loops
Persistent volume	Shared PVC is Bound, uses the intended EFS class, and has ReadWriteMany access
Ingress	ALB address exists; certificate, DNS, access scheme, security groups, and HTTPS hostname are correct
Dependencies	Required databases, Redis, optional ingestion stores, and external services are reachable
Isolation	Environment values, keys, databases, namespaces, and storage paths are independent

The status command is diagnostic: its successful exit does not certify Pod readiness, HTTPS reachability, or application setup. Inspect READY/WANTED counts, readiness columns, events, and the response actually reported.

Application acceptance

Test	Acceptance
First-run setup	Database/schema initialization and first administrator setup complete; license is activated
Sign-in	Administrator can reach the dashboard through the HTTPS hostname
Representative execution	An agreed agent completes with expected results and execution history
Background work	Scheduling and worker processing function without stalled executions
Real-time communication	SignalR WebSocket connection and updates operate through ingress
Upload	An approved 500 MB test file uploads successfully through the complete ingress path
Optional ingestion	Ingestion and its PostgreSQL, Qdrant, and Neo4j connections work
Optional voice	Browser microphone works over HTTPS and a two-way call succeeds through the configured TURN relay
Optional integrations/archival	Configured external integrations and archival policies operate as intended

Use test data and an agreed upload destination. The 500 MB test is an acceptance check, not proof that every file type or the application's full 1 GB upload limit has been validated. Test the required workload limits separately.

API `/health/live` and frontend `/index.html` support monitoring. Liveness alone does not validate database configuration or business execution. Before setup completes, `/api/setup/status` may return HTTP 400; establish initialization state using the setup wizard.

Persistence and recovery

In Staging, save an identifiable, non-sensitive application setting, restart the API, and verify that it survives:

```
kubectl -n "$NAMESPACE" rollout restart deployment/"$RELEASE-api"  
kubectl -n "$NAMESPACE" rollout status deployment/"$RELEASE-api" --timeout=15m
```

Confirm that the same configuration survives Pod recreation and the tested upgrade. Validate shared access across the intended node placement. Do not restart Production workloads merely to complete this example without an agreed maintenance window.

Create the configuration backup and separately verify database, required EFS paths, S3 where used, and matching master-key backups outside the cluster. Test restoration and record achieved recovery objectives. Complete the [Enterprise Operations acceptance criteria](#).

Promotion

Promote the approved artifacts with Production-specific configuration, complete setup and licensing, and repeat applicable acceptance checks. Retain evidence before making Production available. Use [Troubleshooting](#) for failed checks; do not bypass a failed prerequisite merely to finish installation.

Enterprise operations

Use this guide to prepare and operate an enterprise AI Command Center deployment on AWS EKS. Establish operating procedures with Customer Infrastructure, DevOps, Security, application owners, and IntelliBuddies before making Production available. Review the [prerequisites](#) and [deployment verification checklist](#) alongside these requirements.

The guidance below defines operational expectations. Product-specific scaling, recovery, and upgrade procedures must match the deployed IB-X release and the configuration agreed with IntelliBuddies.

Operational ownership

Area	Customer responsibility	IntelliBuddies responsibility
Infrastructure	Operate compute, networking, storage, databases, and supporting services	Provide application infrastructure requirements
Security	Manage access, certificates, secrets, infrastructure patching, and security monitoring	Provide application security and release guidance
Availability	Define service objectives, capacity, maintenance windows, and incident response	Confirm supported application deployment and scaling behavior
Recovery	Implement backups, protect recovery credentials, and conduct restoration exercises	Provide application recovery requirements and guidance
Releases	Approve, test, deploy, and record releases	Supply release artifacts, compatibility information, and upgrade guidance
Support	Collect sanitized diagnostics and coordinate incidents	Investigate application issues through the agreed support process

Record named owners, escalation contacts, and support coverage for each environment. Agree on availability targets, acceptable maintenance downtime, and incident severity definitions.

Supported configuration and release inventory

Maintain an inventory of the IB-X release, Helm chart version, container image references, EKS version, node operating system and architecture, Helm client, ingress controller, storage drivers, PostgreSQL, and applicable ingestion or supporting services. Confirm compatibility with IntelliBuddies before introducing a version change.

Record the release name, namespace, environment configuration, storage mappings, and external endpoints. Store configuration in controlled version management without plaintext credentials. Prefer immutable image references and retain the approved artifacts needed for recovery.

Bundled Redis is mandatory and single-replica, with authenticated, in-memory messaging. Monitor it and account for loss of temporary messages during outages. External Redis is not supported by this deployment configuration. Optional S3 document storage requires its own access, availability, and recovery controls.

Security and access

Apply least privilege to AWS identities, Kubernetes access, deployment automation, and application administration. Separate routine deployment access from administrative access, and review permissions periodically. Give application workloads only the AWS permissions they require.

Restrict access to the cluster API, application endpoints, databases, and supporting services according to customer policy. Configure security groups and supported network controls for necessary traffic. Separate Staging and Production access; namespaces alone do not define the complete security boundary.

Use HTTPS for external application access and monitor certificate expiry. Define encryption requirements for databases, persistent volumes, backups, and internal connections according to customer policy and supported application configuration.

Protect database credentials, registry credentials, application secrets, and the application master key. Restrict access to configuration files containing sensitive values. Use an approved secrets-management process and validate credential rotation with the deployed application before applying it in Production. Preserve the master key required to recover existing application data. Do not rotate it without a supported key-migration procedure. Back up the matching key securely outside the cluster; configuration recovery requires it.

Scan approved images, define vulnerability remediation ownership, and maintain node and cluster components. Collect appropriate infrastructure, Kubernetes, and application audit records with controlled access and retention. Exclude credentials and sensitive customer data from routine diagnostic exports.

For infrastructure controls, follow the [Amazon EKS security guidance](#).

Availability and capacity

The default single replica per application service does not establish application high availability. Confirm supported replica counts and coordination behavior for each service before scaling. In particular, confirm scheduler coordination, worker execution behavior, shared configuration access, and voice-session affinity and draining.

Design node placement and database availability to meet the agreed failure scenarios. Verify that persistent storage supports the intended placement across nodes and Availability Zones. Multiple replicas do not remove a shared storage or database dependency.

Set resource requests and limits using representative Staging measurements. Measure concurrent executions, scheduling delay, ingestion throughput, voice concurrency, database connections, and storage growth. Maintain capacity headroom and define thresholds for review. Introduce autoscaling only for services whose scaling behavior has been confirmed.

Plan node maintenance and Pod disruption around active executions and voice conversations. Validate drain behavior and service recovery before relying on disruption policies or rolling updates for continuity. Test failure scenarios and define whether affected work resumes, fails, or requires operator action; do not assume automatic replay is safe.

Use the [Amazon EKS reliability guidance](#) when designing infrastructure availability.

Monitoring and incident response

Centralize operational logs and monitoring using customer-approved tools. Define retention, access controls, alert thresholds, and an owner for each alert. Infrastructure monitoring should be accompanied by application checks that establish usability and execution progress.

Area	Monitor and investigate
Kubernetes	Unavailable workloads, restart loops, failed scheduling, image pulls, node capacity, and resource exhaustion
Application	Endpoint availability, error rates, latency, failed or stalled executions, and scheduling delays
Databases and supporting services	Connectivity, capacity, storage growth, service health, and backup failures
Storage	Mount failures, permissions, free capacity, and persistence of configuration and runtime data
Networking	DNS, ingress errors, TLS expiry, and required external connectivity
Optional capabilities	Ingestion failures, voice connectivity and session errors, and archival processing

Where an IB-X release does not expose a required metric directly, agree on a supported log-based or functional check. Monitor license status and required licensing connectivity under the applicable licensing configuration.

For incidents, record the environment, release, time range and timezone, affected functions, workload impact, and recent changes. Collect relevant Pod status, Kubernetes events, sanitized logs, and dependency health. Assign an incident owner and preserve evidence before making changes. Escalate application issues through the agreed IntelliBuddies support channel; provide credentials only through an approved secure mechanism when required.

Backup and disaster recovery

Define a recovery point objective (RPO), the acceptable amount of data loss, and a recovery time objective (RTO), the acceptable restoration time, for each environment. Set backup frequency, retention, protection, and recovery location to meet those objectives. Document which failures the recovery plan addresses, including database loss, volume loss, and loss of the deployment environment.

Asset	Recovery requirements
Primary and archival databases	Consistent backups, retention, database-native recovery procedures, and restored connectivity
Ingestion data stores	Coordinated recovery of PostgreSQL, Qdrant, and Neo4j when used; confirm any supported re-ingestion procedure
Persistent application data	Back up writable configuration, documents, and applicable runtime files
Secrets and master key	Protect recoverable copies or references and retain access to required encryption keys
Deployment configuration	Preserve release-specific Helm values, infrastructure configuration, namespaces, and storage mappings
Release artifacts	Retain approved image references and access to compatible images and Helm packages
Supporting services	Back up persistent data or document supported reconstruction according to their role

Confirm application consistency requirements before selecting backup timing or restoring databases and files from different recovery points. A volume snapshot alone does not establish application-consistent recovery. Monitor backup failures and protect backups from unauthorized modification or deletion.

Recovery exercise

1. Select a recovery point and compatible release using the agreed recovery procedure.
2. Provision an isolated recovery environment and restrict outbound integrations to avoid duplicate executions or unintended external actions.
3. Restore databases, persistent data, required secrets and keys, and deployment configuration according to the agreed consistency and startup requirements.
4. Deploy the matching application release and verify database access, configuration, licensing, documents, and applicable integrations.
5. Check execution state and determine how interrupted work should be handled before enabling workers, scheduling, or external traffic.
6. Measure achieved RPO and RTO, record discrepancies, and obtain approval before switching Production traffic or execution ownership.

Perform restoration exercises periodically and after material architecture changes. Successful backup jobs are insufficient evidence of recoverability.

Upgrades and rollback

Review release notes and compatibility information before changing IB-X, its chart, cluster components, or dependencies. Identify database migrations, configuration changes, expected downtime, and recovery requirements with IntelliBuddies.

Validate the upgrade in Staging with representative data and workloads. Confirm application setup, execution, scheduling, optional capabilities, persistence, and master-key continuity. Take the required backups, record the previous configuration and artifact references, and agree on a maintenance and rollback decision window.

During Production maintenance, control new work and active sessions according to the supported application procedure. Deploy the approved release, perform the acceptance checks, and monitor application and dependency health before closing the change.

A Helm rollback does not automatically reverse database migrations or restore persistent data. Confirm whether the previous application release can use the modified schema and configuration. If it cannot, use the agreed database and data recovery procedure. Test rollback or recovery in Staging before making it part of the Production change plan.

Operational commands

Run from the extracted package folder. Define these variables in each new terminal session and confirm the cluster context; substitute Production values when appropriate:

```
NAMESPACE="aicc-staging"
RELEASE="aicc"
VALUES="values-staging.yaml"
kubectl config current-context
```

Status and logs

```
./scripts/aicc-eks.sh status --namespace "$NAMESPACE" --release "$RELEASE"
./scripts/aicc-eks.sh logs api --namespace "$NAMESPACE" --release "$RELEASE"
./scripts/aicc-eks.sh logs voice -f --namespace "$NAMESPACE" --release "$RELEASE"
```

Log targets are api, worker, scheduler, voice, ingestion, frontend, and redis. Stop following logs with Ctrl+C. Services write structured logs to stdout; collect them with customer tooling such as Fluent Bit and CloudWatch Logs. Monitor API /health/live, frontend /index.html, dependencies, Pod readiness, and persistent storage. A liveness response or the status script's exit code alone does not establish functional readiness.

Configuration backup

```
./scripts/aicc-backup.sh --namespace "$NAMESPACE" --release "$RELEASE" \
  --out "/safe/backup/location/aicc-staging-$(date +%Y%m%d-%H%M%S)"
```

Choose a restricted, writable destination that does not already exist. The script requires a running API Pod and saves:

Output	Coverage
shared-volume.tar.gz	/aicc/data and /aicc/documents when present; --no-documents excludes documents
values-in-cluster.yaml	Helm values applied to the release
history.txt	Helm revisions
deployments.txt	Deployed image references

This is not a full EFS backup. It excludes other persistent paths, databases, S3 objects, the master key, and TURN credentials. Back these up separately using the agreed recovery process. Encrypt and restrict the backup destination; preserve the matching master key outside the cluster. Test recovery with IntelliBuddies before restoring Production data. A live file archive is not a coordinated database/filesystem snapshot.

Upgrade and configuration changes

1. Review release guidance and validate the change in Staging.
2. Mirror and approve the new images and OCI chart using `aicc-mirror-to-ecr.sh`.
3. Back up databases, required EFS paths, S3 where used, and Secrets outside the cluster.
4. Change all six image tags in the environment values file; preserve the key, storage, and other settings. Retain the one-line service format required by this script revision.
5. Review a rehearsal, then perform the upgrade during the agreed maintenance window.

```
./scripts/aicc-eks.sh upgrade --values "$VALUES" \
  --namespace "$NAMESPACE" --release "$RELEASE" \
  --chart-version YOUR_CHART_VERSION --dry-run
```

```
./scripts/aicc-eks.sh upgrade --values "$VALUES" \
  --namespace "$NAMESPACE" --release "$RELEASE" \
  --chart-version YOUR_CHART_VERSION --backup
```

--backup invokes the configuration backup script only; it does not back up the database or the entire EFS file system. Protect Helm rehearsal output because it can include Secrets. Apply the [verification checks](#) after upgrades and configuration changes. Changes using the same release tag use this same workflow.

Rollback

```
helm history "$RELEASE" -n "$NAMESPACE"
./scripts/aicc-eks.sh rollback --namespace "$NAMESPACE" --release "$RELEASE"
```

To select a specific revision, add its revision number immediately after `rollback`. Helm restores previous release resources, not database schema or persistent data. Do not assume the previous application can use a migrated database. Coordinate restoration and interrupted-work handling with IntelliBuddies before retrying Production work.

Restart and credential changes

```
./scripts/aicc-eks.sh restart --namespace "$NAMESPACE" --release "$RELEASE"
```

The script initiates restart of all deployments carrying the release label, then waits for their rollouts. It does not restart one deployment at a time and can include bundled Redis. Active calls can be interrupted and temporary messaging lost. Use a maintenance window and verify affected work afterward.

Update TURN credentials using the approved Secret-management process, then restart applicable workloads so environment references are refreshed. Do not delete a credential Secret before its replacement is ready. This operation is distinct from the application master key, which must remain unchanged unless an explicit migration procedure is supported.

Uninstall and retained data

Back up and verify recoverability before uninstalling:

```
./scripts/aicc-eks.sh uninstall --namespace "$NAMESPACE" --release "$RELEASE"
```

The command removes release resources, including workloads and ingress, and requests confirmation. With an exclusively owned ALB, its removal is managed by the load balancer controller; do not assume a shared load balancer is removed. The database, ECR artifacts, S3 content, and EFS file system are separately managed.

A Retain policy leaves backing persistent data for deliberate recovery or cleanup. Inspect the actual PVC/PV and retained Secrets after uninstall. The externally created master-key Secret remains unless explicitly deleted or its namespace is removed. Redis secret retention depends on the chart's resource policy. Retention does not automatically reconnect a new installation to the previous volume; confirm the reattachment and recovery procedure with IntelliBuddies.

Full decommissioning additionally addresses namespaces/Secrets, retained volumes/access points, databases, S3, ECR, DNS, certificates, and IAM roles. Delete only environment-owned resources after checking shared dependencies and retention obligations.

Production acceptance

Record evidence and approval for the following criteria before opening Production:

- Supported release and infrastructure versions are documented, with approved images in ECR.
- Security controls, secrets ownership, access reviews, and certificate renewal procedures are established.
- Representative workloads meet agreed performance targets with sufficient capacity headroom.
- Node or Pod failure and maintenance scenarios demonstrate the agreed application recovery behavior.
- Configuration and runtime data survive restarts and the tested upgrade procedure.
- Backups are monitored and a restoration exercise meets the agreed RPO and RTO.
- Upgrade and rollback or recovery procedures are tested and assigned to named owners.
- Monitoring, actionable alerts, escalation contacts, and support diagnostics are available.
- The deployment verification checklist is completed, including licensing and optional capabilities in use.

Retain the acceptance record, configuration inventory, recovery results, and change history with the customer's operational documentation.

Troubleshooting

Run from the extracted package folder using the affected environment's configuration. Confirm the account and Kubernetes context first:

```
NAMESPACE="aicc-staging"
RELEASE="aicc"
VALUES="values-staging.yaml"
kubectl config current-context
./scripts/aicc-eks.sh status --namespace "$NAMESPACE" --release "$RELEASE"
./scripts/aicc-preflight.sh --values "$VALUES" --namespace "$NAMESPACE" \
  --chart-version YOUR_CHART_VERSION
kubectl -n "$NAMESPACE" get events --sort-by=.lastTimestamp
```

Resolve [FAIL] entries and review warnings, including checks skipped for missing AWS credentials. Preflight is not a complete readiness check. Its optional `--db` test creates a temporary Pod and tests only TCP reachability.

Images and chart

Symptom	Investigation
ImagePullBackOff or ErrImagePull	Verify the repository/tag in ECR, node pull permissions, and network access to image-pull dependencies. Mirror Redis as well as the six application images.
Chart cannot be pulled	Verify the OCI repository, chart version, and ECR Helm authentication. Default repository is <code>aicc/aicc</code> ; a four-part image tag maps to a hyphenated chart version.
Unexpected tag or ECR check failure	Preserve one-line <code>aicc-*.image.tag</code> entries in this script revision and keep all six tags consistent.
Unsupported executable format or scheduling	Check node architecture. Supplied application images require <code>linux/amd64</code> .

The mirror script's `--dry-run` does not test registry credentials or the availability of source artifacts. An actual successful import and image approval are separate requirements.

Storage

If the PVC or Pods remain Pending, inspect the claim and a representative affected Pod:

```
kubectl -n "$NAMESPACE" get pvc
kubectl -n "$NAMESPACE" describe pvc YOUR_PVC_NAME
kubectl -n "$NAMESPACE" describe pod YOUR_POD_NAME
kubectl get storageclass efs-aicc -o yaml
kubectl -n kube-system get pods
```

Confirm EFS CSI availability and IAM permissions, the file system ID, access-point provisioning, RWX access, and UID/GID 10001. Check mount targets in worker-node Availability Zones and TCP 2049 access. Healthy CSI Pods alone do not prove controller authorization: `CreateAccessPoint`, `Failed to create Access point`, or `role-discovery` errors can indicate missing IAM configuration.

For permission failures, verify application write access to `/aicc` and access-point ownership. Do not remove the PVC or recreate the namespace as a routine troubleshooting step; configuration and Secrets are required for recovery.

Ingress, DNS and TLS

```
kubectl -n "$NAMESPACE" describe ingress "$RELEASE"
kubectl -n "$NAMESPACE" get service "$RELEASE-frontend"
```

- No ALB address: inspect controller events, IngressClass `alb`, IAM permissions, subnet discovery, and security groups.
- Certificate warning: verify Issued status, matching hostname, correct ARN, and ALB region.
- No response: check DNS against the ALB address and whether an internal ALB is reachable from the caller's network.
- HTTP 502/503: inspect frontend readiness, Service endpoints/target ports, ALB target health, and network access.

- Chat/API returns 404: keep ingress routed through the frontend service; do not add a separate `/api` ingress rule.
- Dropped WebSockets: verify the 3600-second idle timeout and any additional proxy timeouts.
- Upload failures: check the complete ingress path and application limit; NGINX annotations do not configure ALB.

For an authorized diagnostic session before ingress is ready:

```
kubect1 -n "$NAMESPACE" port-forward service/"$RELEASE-frontend" 8080:80
```

Open `http://localhost:8080` on that workstation. This is a diagnostic path, not the published production endpoint. Stop forwarding with `Ctrl+C`.

Database and setup

Use a cluster-reachable endpoint, not `localhost`, and an account with the required database/schema permissions. Check security groups, DNS, and the matching master-key Secret. The primary connection is configured in the wizard, not Helm values.

```
./scripts/aicc-preflight.sh --values "$VALUES" --namespace "$NAMESPACE" \
  --chart-version YOUR_CHART_VERSION --db your-db-host:5432
./scripts/aicc-eks.sh logs api --namespace "$NAMESPACE" --release "$RELEASE"
```

Database names are normalized to lowercase with underscores. Initial HTTP 400 from `/api/setup/status` is not sufficient evidence of failure. If work stays at Initiated after setup, inspect API, Worker, Scheduler, and Redis logs and readiness before scheduling a restart. The restart utility targets all release deployments and can interrupt messaging and calls.

If encrypted configuration cannot be read after a redeployment, compare the key and retained storage with the known-good backup. Do not generate a replacement master key for existing encrypted data.

Reset saved database setup

Use this procedure only when a replaced, restored, or emptied database requires the setup wizard to run again. Investigate network, credentials, and master-key errors first. Schedule a maintenance window, verify configuration and database backups, and confirm the account, cluster context, namespace, and release.

```
./scripts/aicc-eks.sh reset-setup --namespace "$NAMESPACE" --release "$RELEASE"
```

The command prompts for confirmation, copies `/aicc/data/appsettings.json` to `appsettings.json.bak-<date>` alongside the original, clears saved connection fields (`DefaultConnection`, `SecretKey`, `ConnectionString`), resets numeric `DatabaseType` to zero, and restarts all deployments bearing the release label. It requires a running API deployment, permission to execute commands in its Pod, and write access to the configuration file. Restarting can interrupt Redis messaging and voice sessions.

The database itself is not changed or deleted. Keep the existing master key and select the intended database when completing setup again. Protect the dated backup as sensitive configuration because it can contain credentials or encrypted connection settings; apply your access and retention policy. After setup, repeat the [deployment verification checks](#).

Browser voice

1. Verify HTTPS, browser microphone permission, and an appropriately configured conversational agent/AI provider.
2. Verify the complete Helm `global.extraEnv ICE` configuration and referenced `aicc-turn` credential Secret without printing its contents.
3. Start a test call, then check the effective ICE servers:

```
./scripts/aicc-eks.sh logs voice --namespace "$NAMESPACE" --release "$RELEASE" \
  | grep 'ICE servers in use'
```

4. Confirm the expected TURN hostname, transports, credentials, relay-port range, and connectivity from browsers and Voice Pods.
5. Verify outbound connectivity to the configured speech and language providers.

After changing Helm values, apply an upgrade. After changing Secret-backed environment credentials, restart applicable workloads during the agreed maintenance window. Administration settings should not be assumed to override deployment environment values.

Support information

Provide the environment, release/chart versions, time range and timezone, recent changes, sanitized status/preflight output, relevant events, and service logs. Review diagnostics before sharing them: Helm output, application logs, and incorrectly populated values can contain sensitive data. Do not send master keys, database passwords, TURN credentials, or registry tokens in ordinary tickets.

For a failed upgrade, coordinate database compatibility and recovery before rolling back application resources. See [Enterprise Operations](#).

Third-party software references

The toolkit contains deployment scripts and templates, not third-party executable binaries or container images. Separately acquired software remains subject to its own terms; this document does not change any applicable agreement.

Redis is retrieved separately by the mirroring utility. The floating redis:7-alpine tag does not identify a fixed license or version. Record the approved exact image version/digest and applicable license with the deployment release.

Redis licensing information

ORAS is obtained separately when needed by the mirroring utility. The project identifies its license as Apache-2.0; preserve the notices applicable to the acquired binary.

ORAS project and license

Other externally installed tools and services, including AWS CLI, Kubernetes, Helm, Bash, PostgreSQL, Qdrant, and Neo4j, retain their respective terms. Product names, logos, and trademarks belong to their respective owners.