CHAI™ Platform — Infrastructure & Deployment Reference Guide

CHAI™

Infrastructure & Deployment Reference Guide

AWS Architecture · High Availability · Disaster Recovery · External Integrations · DORA Compliance

Prepared for: General Distribution
Prepared by: CloudHedge Technologies Inc.
Classification: Confidential
Version: 1.0 · March 2026
DORA Alignment: EU Regulation 2022/2554 — Digital Operational Resilience Act

"Migrate Anything. Modernize Everything. Break Nothing."

Document History

Version	Date	Author	Changes
1.0	March 2026	CloudHedge Technologies Inc.	Initial release — AWS architecture, HA, DR, integrations, DORA compliance, operational runbook

Table of Contents

How to Use This Document

Document Scope: AWS-Specific Reference

This document covers the CHAI platform deployed on **Amazon Web Services (AWS)**. All architecture diagrams, service references, deployment procedures, and cost estimates in this guide are AWS-specific.

CHAI supports multi-cloud deployment. The platform can be deployed on AWS, Microsoft Azure, and other cloud environments. This document is the AWS-specific deep-dive reference. A cloud-agnostic overview document covering deployment concepts generically across all clouds is planned as the parent reference.

If you are deploying CHAI on **Azure or another cloud**, the concepts and architecture patterns in this document apply — however the specific services, terminology, and configuration steps will differ. Refer to the relevant cloud-specific reference when available, or contact CloudHedge for guidance.

This guide is the authoritative reference for deploying, operating, and maintaining the CHAI platform on AWS. It is designed to be self-sufficient — no additional infrastructure documentation is required.

For end-user workflows (running assessments, triggering transformations, managing projects), refer to the CHAI User Guide.

If you are a...	Start with...	Key sections
Infrastructure / DevOps Engineer deploying CHAI for the first time	Appendix C (Deployment Checklist)	Sections 2, 3, 4, 5, Appendix A.2
Infrastructure Engineer operating an existing deployment	Section 11 (Operational Runbook)	Sections 5, 6, 7, 11
Security Architect reviewing CHAI for compliance approval	Section 10 (Security & DORA)	Sections 4.2, 10, Appendix D
Application Team Lead integrating CHAI with your SCM and CI/CD	Section 8 (External Integrations)	Sections 8, 9
Cloud Architect evaluating deployment topology options	Section 2.1 (Deployment Architecture)	Sections 2, 5, 6, Appendix B
Compliance Officer assessing DORA alignment	Section 10.1 (DORA Matrix)	Section 10

1. Executive Summary

CHAI™ (Context Rich Harness) is CloudHedge's flagship AI-powered modernization platform, purpose-built to accelerate enterprise application migration to the cloud at scale. CHAI supports deployment across major cloud providers — with AWS representing the current production-mature path. Deployed as a self-contained appliance on cloud infrastructure, CHAI orchestrates 17 containerized microservices — coordinating AI-driven discovery, deep application assessment, code transformation, and cloud infrastructure provisioning — all within a single hardened deployment unit. **This document is the AWS-specific infrastructure and deployment reference** for the CHAI platform, covering AWS architecture, high availability, disaster recovery, external integrations, and alignment with the EU Digital Operational Resilience Act (DORA).

The CHAI appliance runs on RHEL 8.10 or Amazon Linux 2023, uses Podman for rootless container orchestration, and persists all stateful data — including MongoDB 7.0.12 document storage — to Amazon EFS for decoupled, multi-AZ durability. The platform integrates natively with AWS Bedrock (Claude, Titan, Llama), AWS Transform, Bitbucket, GitHub, Jenkins, and all major container registries. The current production release is ch-rel-2.1.7.18 (March 25, 2026), distributed via AWS Marketplace and the CloudHedge ECR registry at <ecr-registry-url>.

This document is DORA-aligned per EU Regulation 2022/2554. It addresses ICT risk management, operational resilience testing, cross-region disaster recovery (RTO < 30 minutes, RPO < 1 hour), incident classification and reporting, third-party risk through SBOM management, and IAM least-privilege access control. All operational teams — infrastructure engineers, security architects, application owners, and compliance officers — will find actionable reference material in the sections that follow.

Platform Key Facts

Release: CHAI v2.1.7.x (latest: ch-rel-2.1.7.18, Mar 25 2026) | **OS:** RHEL 8.10 / Amazon Linux 2023 | **Container Runtime:** Podman (rootless) | **Database:** MongoDB 7.0.12 | **Services:** 17 microservices | **Primary Region:** Customer-Designated | **DR Region:** Customer-Designated | **DORA Alignment:** EU Regulation 2022/2554

Companion Document: For end-user workflows — running DART assessments, configuring transformation pipelines, managing projects and tenants, and using AI-powered recommendations — see the CHAI User Guide.

Operational Responsibility Matrix

Area	CloudHedge	Customer
CHAI microservice container images (builds, security patches, updates)	Owns — publishes updated images to ECR per release cycle	Pulls updated images and restarts appliance via sudo chctl upgrade
Host OS patching (RHEL 8.10 / Amazon Linux 2023)	Provides compatible AMI updates	Owns — applies OS patches per organizational patching policy
ALB TLS certificate renewal	N/A	Owns — managed via AWS Certificate Manager (auto-renewal)

Area	CloudHedge	Customer
DR failover activation	Provides procedure (Section 6.3) and on-call support	Owns — executes failover procedure or engages CloudHedge support
CloudWatch alarm configuration	Provides recommended alarm definitions (Section 11.3)	Owns — configures SNS topics and integrates with internal alerting
MongoDB backup and restore	Provides EFS-based durability architecture and procedure (Section 7)	Owns — manages EFS backup policies and validates restore procedures
AWS IAM policy management	Provides minimum IAM policy reference (Section 10.4)	Owns — creates and assigns IAM roles per organizational policy
Security vulnerability response	Publishes security advisories and patched images	Owns — applies updates within organizational SLA; reviews SBOM
Bedrock model access and cost	N/A	Owns — enables models in AWS console; manages Bedrock usage costs
License management	Issues and validates license keys	Owns — ensures license is current; contacts CloudHedge for renewal

For support engagement, incident escalation, and SLA terms, refer to the applicable CloudHedge Enterprise Support Agreement.

---

2. Infrastructure Overview

2.1 Deployment Architecture

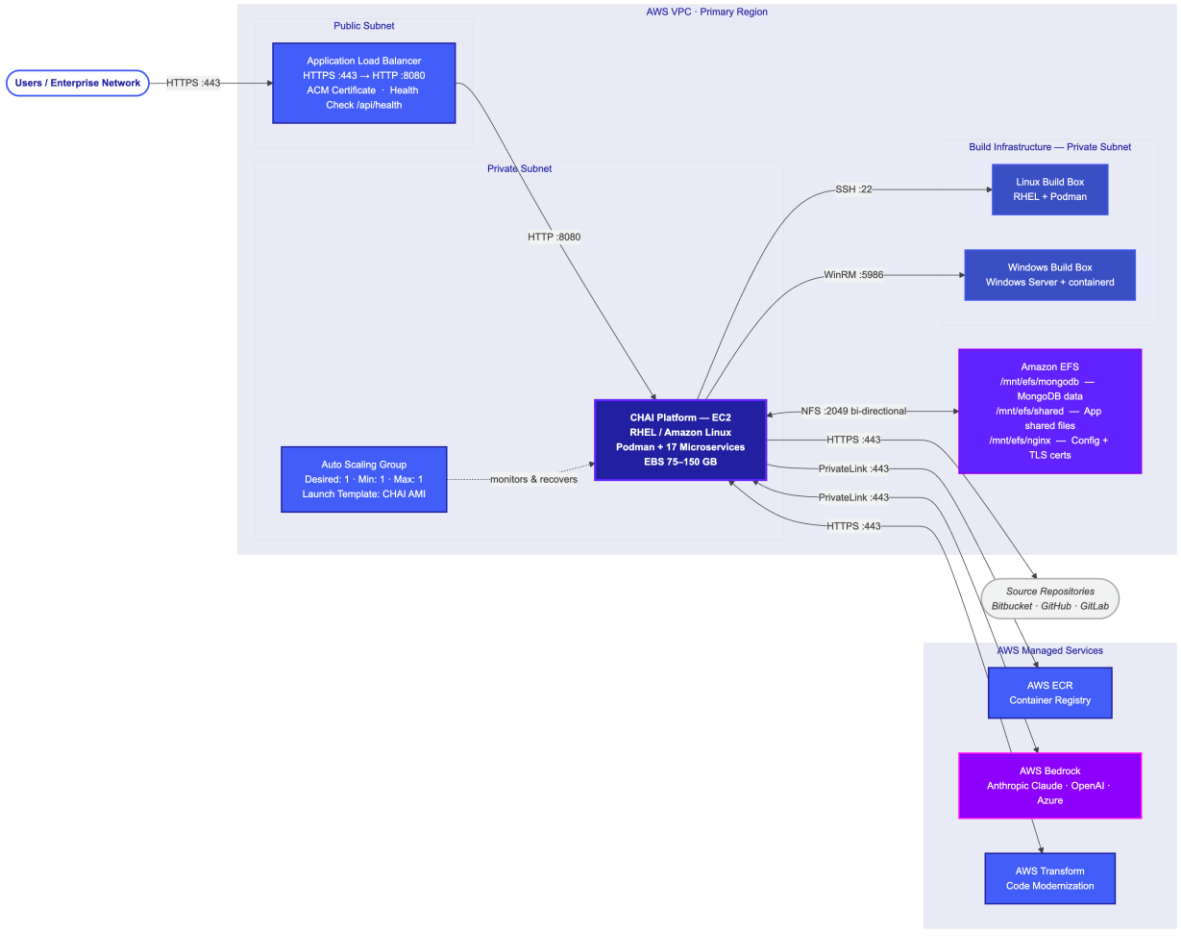
CHAI™ supports three primary deployment topologies, each suited to different organizational requirements for scalability, cost, and operational complexity. All three topologies use the same AMI, the same microservice configuration, and the same EFS-backed persistent storage — ensuring consistency across environments.

Deployment Option	Description	Pros	Cons
Single EC2 (Standard)	One EC2 instance running all 17 microservices behind an ALB. ASG maintains desired capacity of 1 for auto-recovery.	Lowest cost; simplest to operate; fastest to provision; suitable for most enterprise workloads	No active-active redundancy; brief downtime during AMI-based instance replacement (~5–10 min)

Deployment Option	Description	Pros	Cons
Multi-AZ Active-Standby	Primary EC2 in AZ-1; pre-warmed standby EC2 in AZ-2. ALB health checks trigger failover automatically within 60–90 seconds.	Fast automated failover; supports planned maintenance windows with zero downtime	Higher cost (two running instances); requires EFS-backed shared storage to synchronize state
Full HA with Blue/Green	Two ASGs (blue and green) behind a single ALB. Deployments shift traffic between environments via weighted target groups.	Zero-downtime deployments; instant rollback capability; supports canary traffic splitting	Highest complexity; requires CI/CD pipeline integration; approximately 2× EC2 cost during deployment windows

The recommended standard deployment uses the **Single EC2 with ALB and ASG** model, with EFS providing persistent multi-AZ storage durability and daily AMI snapshots enabling rapid instance replacement with no data loss.

Choosing a Deployment Topology: Most organizations start with the **Single EC2** model for its simplicity and lowest cost, then upgrade to **Multi-AZ Active-Standby** when production SLAs require automated failover. The **Blue/Green** model is recommended for organizations with mature CI/CD pipelines that require zero-downtime deployments. All three topologies use the same AMI, the same microservice configuration, and the same EFS-backed storage — upgrading between topologies requires no application changes.



2.2 AWS Services Used

AWS Service	Purpose	Configuration Notes
EC2	Hosts the CHAI appliance and all 17 microservices	m7i.xlarge (recommended production) or t3.large (dev/staging); RHEL 8.10 or Amazon Linux 2023 AMI
Application Load Balancer (ALB)	TLS termination, HTTP/HTTPS routing, health checks	HTTPS :443 listener → Target Group :8080; ACM certificate; stickiness enabled
Auto Scaling Group (ASG)	Automatic instance recovery and lifecycle management	Desired: 1, Min: 1, Max: 1 (standard); uses Launch Template with IAM instance profile
Amazon EFS	Multi-AZ NFS persistent storage for MongoDB data and shared application files	General Purpose performance mode; Bursting throughput; encryption at rest (AES-256); mount targets in each AZ
AWS ECR	Private container image registry for all CHAI microservice images	Account: 720029083713; region: <your-region>; image scanning enabled; lifecycle policies configured
AWS Bedrock	Foundation model inference for AI-powered assessment, code transformation, and recommendations	Accessed via VPC Endpoint (PrivateLink) or NAT Gateway; models: Claude Sonnet/Opus/Haiku, Titan, Llama 3
AWS Transform	Automated code modernization service integrated with CHAI transform pipelines	HTTPS :443; requires IAM permissions for transform:StartJob, transform:GetJob
AWS Certificate Manager (ACM)	TLS/SSL certificate provisioning for ALB HTTPS listeners	Public certificate for customer domain; auto-renewal managed by ACM
Amazon Route 53	DNS routing, health checks, and failover between primary and DR regions	Failover routing policy; health check on ALB endpoint; TTL: 60 seconds
AWS IAM	Identity and access management for EC2 instance role and human operators	Instance profile with least-privilege policy; no embedded credentials in containers
Amazon CloudWatch	Metrics, log aggregation, alarms, and operational dashboards	Log groups per microservice; custom metrics via PutMetricData; 8 key alarms configured
AWS CloudTrail	API audit logging for all AWS API calls across the account	Multi-region trail; S3 bucket retention 90 days; CloudWatch Logs integration
AWS Systems Manager (SSM)	Secure parameter storage for secrets and configuration; optional session manager access	Parameter Store for ECR credentials, Bedrock endpoint config, license keys
Amazon S3	CloudTrail log archive; optional report export destination; AMI snapshot staging	Server-side encryption (SSE-S3); versioning enabled on audit buckets

AWS Service	Purpose	Configuration Notes
AWS VPC	Network isolation, subnet segmentation, security group enforcement	/16 CIDR; public subnet for ALB; private subnet for EC2, EFS, build boxes
VPC Endpoints (PrivateLink)	Private connectivity to ECR, Bedrock, S3, SSM — no internet traversal	Interface endpoints for ECR API, ECR DKR, Bedrock Runtime; Gateway endpoint for S3
AWS Marketplace	CHAI appliance AMI distribution and subscription management	SKU: 5xtx6bcy9hcg3lj48z84f2li9; billing through AWS account

2.3 Compute & Storage Specifications

Resource	Minimum (Dev/Staging)	Recommended (Production)	Notes
EC2 Instance Type	t3.large (2 vCPU, 8 GB RAM)	m7i.xlarge (4 vCPU, 16 GB RAM)	16 GB RAM required for concurrent AI inference workloads
EC2 vCPU	2 vCPU	4 vCPU	Podman parallel container startup benefits from 4+ cores
EC2 RAM	8 GB	16 GB	MongoDB 7 + Node.js services + Podman overhead = ~12 GB active
EBS Root Volume	75 GB gp3	150 GB gp3	Container images, OS, Podman layer cache; 3000 IOPS baseline
Amazon EFS	20 GB (burst)	100+ GB (General Purpose)	Auto-scales; MongoDB data + shared files + Nginx config
Linux Build Box EC2	t3.medium (2 vCPU, 4 GB)	t3.large (2 vCPU, 8 GB)	Runs Podman for Linux container build; SSH :22 access from CHAI EC2
Windows Build Box EC2	t3.medium (2 vCPU, 4 GB)	t3.large (2 vCPU, 8 GB)	Windows Server 2022 + containerd; WinRM :5986 access from CHAI EC2
Network Bandwidth	Up to 5 Gbps	Up to 12.5 Gbps (m7i.xlarge)	Required for ECR image pulls during startup (~8–12 GB total images)
ALB	1 ALB (shared)	1 ALB (dedicated)	1 listener, 1 target group, 1 ACM certificate
EFS Mount Targets	1 AZ	3 AZ (all in region)	Mount targets in each AZ ensure availability during AZ failure

3. Microservices Architecture

CHAI™ is composed of 17 containerized microservices orchestrated by Podman and managed via the chctl appliance lifecycle tool. All services communicate internally and only the webapp service is exposed externally through the ALB. Detailed service architecture and internal topology documentation is available to CloudHedge authorised teams on request.

3.1 Container Registry & Image Naming

All CHAI microservice images are distributed via the CloudHedge AWS ECR registry. Images follow a consistent naming and tagging convention to enable precise version pinning, rollback, and audit.

Component	Value
ECR Registry Account	720029083713
ECR Region	<your-region>
ECR Base URL	<ecr-registry-url>
Image Namespace	cloudhedge/<service-name>
Release Tag Format	ch-rel-<version>
Current Release Tag	ch-rel-2.1.7.18
Release Date	March 25, 2026

Example image references:

Service	Full Image URI
webapp	<ecr-registry-url>/cloudhedge/webapp:ch-rel-2.1.7.18
core-engine	<ecr-registry-url>/cloudhedge/core-engine:ch-rel-2.1.7.18
auth-gateway-service	<ecr-registry-url>/cloudhedge/auth-gateway-service:ch-rel-2.1.7.18
cruize-service	<ecr-registry-url>/cloudhedge/cruize-service:ch-rel-2.1.7.18
db-service (MongoDB)	<ecr-registry-url>/cloudhedge/db-service:ch-rel-2.1.7.18

ECR Authentication:

Copy

```
aws ecr get-login-password --region <your-region> \
  | podman login --username AWS \
    --password-stdin <ecr-registry-url>
```

All images are scanned for vulnerabilities at push time via ECR enhanced scanning (Inspector). Lifecycle policies retain the last 10 release tags per repository and expire untagged images after 14 days.

4. Networking & Connectivity

4.1 VPC & Subnet Design

Internal Communication Security: All 17 CHAI microservices communicate over the cloudhedge Podman network (10.90.0.0/16), which is isolated to the EC2 instance. Inter-service traffic does not traverse the VPC network or the internet. External traffic enters only via the ALB (HTTPS :443) and exits only via VPC Endpoints (PrivateLink) or NAT Gateway for explicitly configured integrations.

Component	Configuration	CIDR / Details
VPC	chai-prod-vpc	10.0.0.0/16 — 65,536 addresses
Public Subnet AZ-1	chai-public-1a — hosts ALB	10.0.1.0/24 — <region>-a
Public Subnet AZ-2	chai-public-1b — hosts ALB second node	10.0.2.0/24 — <region>-b
Public Subnet AZ-3	chai-public-1c — ALB third node	10.0.3.0/24 — <region>-c
Private Subnet AZ-1	chai-private-1a — EC2, EFS mount target	10.0.11.0/24 — <region>-a
Private Subnet AZ-2	chai-private-1b — EFS mount target, standby EC2	10.0.12.0/24 — <region>-b
Private Subnet AZ-3	chai-private-1c — EFS mount target	10.0.13.0/24 — <region>-c
Internet Gateway	chai-igw — attached to VPC	Provides internet access for ALB; NAT Gateway for EC2 egress
NAT Gateway	chai-nat-1a — in Public Subnet AZ-1	Provides outbound internet for private subnet EC2 (ECR, Bedrock if no VPC endpoint)
Route Table (Public)	Routes 0.0.0.0/0 → Internet Gateway	Associated with all public subnets
Route Table (Private)	Routes 0.0.0.0/0 → NAT Gateway	Associated with all private subnets
Podman Internal Network	cloudhedge network (pod-internal)	10.90.0.0/16 (IPv4), fd00:db8:2::/64 (IPv6) — container-to-container communication

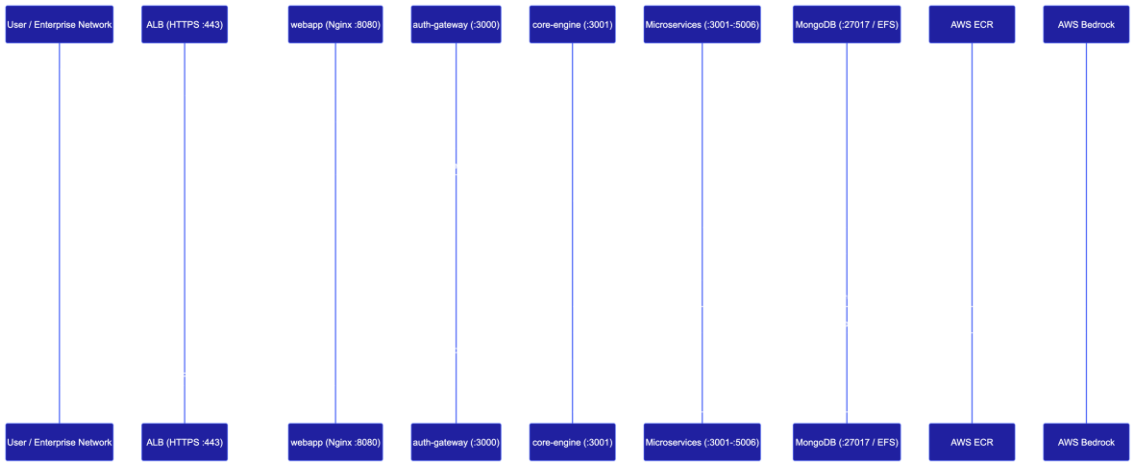
4.2 Security Groups & Port Matrix

⚠ Distribution Review Required — Internal Network Topology

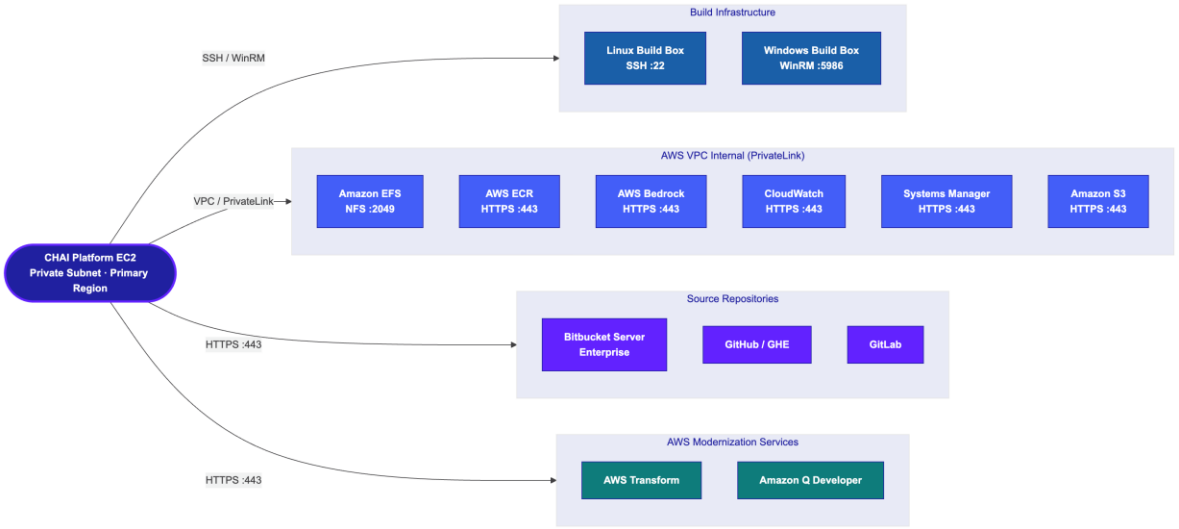
This section contains the full security group rule set and port matrix. Review carefully before external distribution.

Security Group	Rule Direction	Source / Destination	Port / Protocol	Purpose
chai-alb-sg	Inbound	0.0.0.0/0 (or Customer CIDR)	TCP :443 (HTTPS)	End-user access to CHAI web UI
chai-alb-sg	Inbound	0.0.0.0/0 (or Customer CIDR)	TCP :80 (HTTP)	HTTP → HTTPS redirect
chai-alb-sg	Outbound	chai-ec2-sg	TCP :8080	ALB to EC2 webapp forwarding
chai-ec2-sg	Inbound	chai-alb-sg	TCP :8080	Receive proxied traffic from ALB
chai-ec2-sg	Inbound	Customer Bastion / Admin CIDR	TCP :22 (SSH)	Administrative SSH access for ops team
chai-ec2-sg	Inbound	chai-ec2-sg (self)	All traffic	Inter-service communication on Podman network (internal)
chai-ec2-sg	Outbound	chai-efs-sg	TCP :2049 (NFS)	EC2 to EFS NFS mount
chai-ec2-sg	Outbound	Linux Build Box SG	TCP :22 (SSH)	CHAI to Linux Build Box for container builds
chai-ec2-sg	Outbound	Windows Build Box SG	TCP :5986 (WinRM HTTPS)	CHAI to Windows Build Box for Windows container builds
chai-ec2-sg	Outbound	0.0.0.0/0	TCP :443 (HTTPS)	ECR pulls, Bedrock inference, Transform, Bitbucket, GitHub
chai-efs-sg	Inbound	chai-ec2-sg	TCP :2049 (NFS)	EC2 to EFS NFS access
chai-efs-sg	Outbound	None	None	EFS does not initiate connections
chai-build-linux-sg	Inbound	chai-ec2-sg	TCP :22 (SSH)	CHAI EC2 SSH into Linux Build Box
chai-build-windows-sg	Inbound	chai-ec2-sg	TCP :5986 (WinRM)	CHAI EC2 WinRM into Windows Build Box

4.3 Data Flow Diagram



4.4 Network Connectivity Map



5. High Availability Deployment Architecture

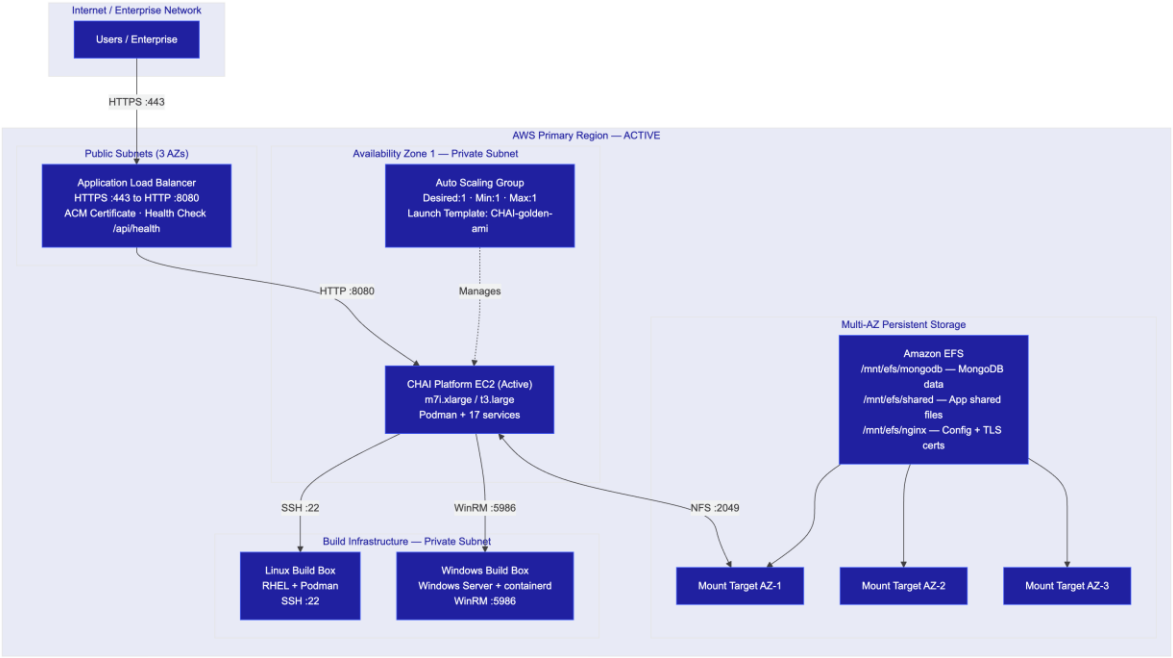
5.1 HA Architecture Overview

CHAI™ achieves high availability through a combination of ALB health checks, ASG auto-recovery, and EFS-decoupled persistent storage. Because all stateful data resides on Amazon EFS rather than the EC2 instance's local EBS volume, a replacement instance launched by the ASG immediately mounts the same data — eliminating recovery time associated with data reconstruction. The following configuration steps establish the core HA posture:

1. **Create the VPC** with public subnets in three Availability Zones for ALB placement, and private subnets in three AZs for EC2 and EFS mount targets.
2. **Deploy Amazon EFS** with mount targets in each private subnet AZ. Enable encryption at rest. Configure inbound NFS (:2049) from the EC2 security group.

3. **Launch the CHAI EC2 instance** from the marketplace AMI in the private subnet. Mount EFS at /mnt/efs via /etc/fstab. Migrate all stateful data to EFS paths.
4. **Create a golden AMI** from the configured and validated EC2 instance. This AMI includes the updated chctl-compose.yaml with EFS volume paths, the Podman configuration, and the CHAI software stack.
5. **Create a Launch Template** referencing the golden AMI, the EC2 security group, the IAM instance profile, and the chctl-compose.yaml user data script that starts the appliance on boot.
6. **Create an Auto Scaling Group** using the Launch Template. Set Desired: 1, Min: 1, Max: 1. Associate with the private subnet. Configure EC2 health check type with a 300-second grace period.
7. **Create an Application Load Balancer** in the public subnets. Add an HTTPS :443 listener with an ACM certificate. Create a Target Group pointing to :8080 with a health check on GET /api/health. Register the EC2 instance.
8. **Configure DNS** (Route 53 or customer DNS) to point the CHAI hostname to the ALB DNS name.
9. **Validate HA** by terminating the EC2 instance and confirming the ASG launches a replacement that mounts EFS and starts CHAI services within the defined recovery window.

5.2 HA Architecture Diagram



5.3 EFS Persistent Storage

Amazon EFS is the foundation of CHAI's data durability strategy. By decoupling all stateful data from the EC2 instance lifecycle, EFS enables rapid instance replacement without data loss, supports the DR replication model, and ensures that both primary and standby instances always operate on the same data set.

EFS Path	Container Mount Path	Contents	Criticality	Notes
/mnt/efs/mongodb/data/db	/opt/appliance/mongodb/data/db	All MongoDB 7.0.12 database files — application metadata, assessment results, user accounts, audit logs, transformation outputs	Critical	Must be backed up via EFS backup policy (daily, 35-day retention)
/mnt/efs/shared	/opt/appliance/shared	Shared application files: uploaded source artifacts, generated reports, intermediate transformation outputs, job artifacts	Critical	Large files; monitor EFS throughput; consider EFS provisioned throughput for large-scale transformations
/mnt/efs/nginx	/opt/appliance/nginx	Nginx configuration (nginx.conf), SSL/TLS certificates, reverse proxy rules, custom error pages	High	TLS certificates must be renewed before expiry; CHAI will fail to serve HTTPS if certs expire
/mnt/efs/nginx/ssl	/opt/appliance/nginx/ssl	TLS private key and certificate chain (PEM format) for the CHAI web UI HTTPS endpoint	High	Stored on EFS so replacement instances inherit certificate configuration without re-provisioning
/mnt/efs/shared/chctl-compose.yaml	/opt/appliance/chctl-compose.yaml	Appliance orchestration manifest — defines all 17 microservice containers,	Critical	Any volume path updates must reference /mnt/efs/ paths; changes take

EFS Path	Container Mount Path	Contents	Criticality	Notes
		volume mounts, network configuration, environment variables		effect on chctl restart appliance

5.4 EFS /etc/fstab Configuration

The following /etc/fstab entry mounts the Amazon EFS file system automatically on instance boot, ensuring CHAI services have access to persistent data before the appliance starts.

Copy

```
# /etc/fstab — Amazon EFS mount for CHAI persistent storage
# Replace <EFS_FILE_SYSTEM_ID> with your actual EFS ID (e.g., fs-0a1b2c3d4e5f6789a)
<EFS_FILE_SYSTEM_ID>.efs.<your-region>.amazonaws.com:/ /mnt/efs efs \
  _netdev,tls,iam,noresvport,nofail,\
  retrans=2,rsize=1048576,wsize=1048576,\
  hard,timeo=600 0 0
```

Manual mount command (for initial setup or troubleshooting):

Copy

```
sudo mount -t efs -o tls,iam <EFS_FILE_SYSTEM_ID>:/ /mnt/efs
```

Create required subdirectories after first mount:

Copy

```
sudo mkdir -p /mnt/efs/mongodb/data/db
sudo mkdir -p /mnt/efs/shared
sudo mkdir -p /mnt/efs/nginx/ssl
sudo chown -R 1000:1000 /mnt/efs/mongodb /mnt/efs/shared /mnt/efs/nginx
```

IAM Permissions Required for EFS IAM-based Mount: The EC2 instance IAM role must include elasticfilesystem:ClientMount, elasticfilesystem:ClientWrite, and elasticfilesystem:ClientRootAccess permissions on the EFS resource ARN. Without these permissions, the iam mount option will cause the mount to fail silently. Ensure the EFS file system policy also grants access to the EC2 instance role ARN. See Section 10.4 for the full IAM policy reference.

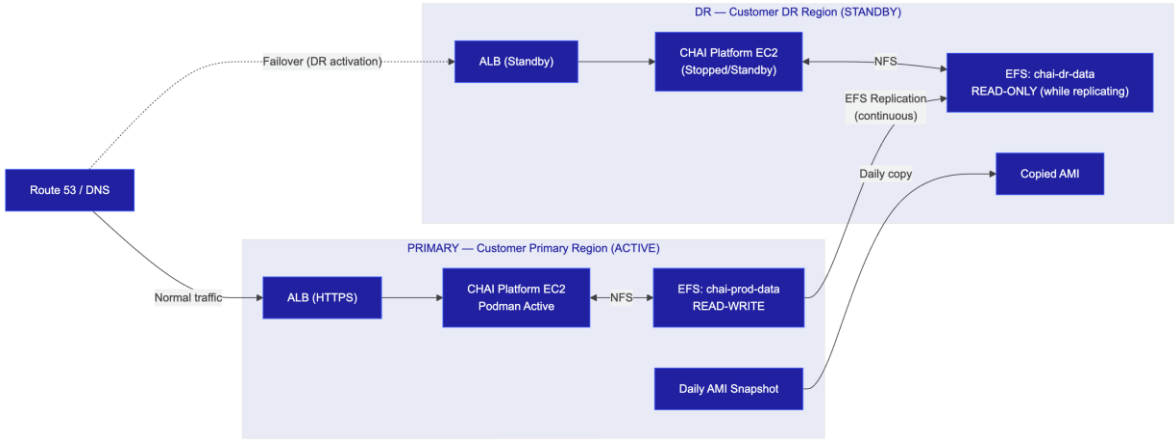
---

6. Disaster Recovery

6.1 Cross-Region DR Architecture

CHAI™ implements a cross-region warm-standby disaster recovery architecture with a primary deployment in the customer-designated primary region and a standby deployment in a customer-designated DR region.

EFS continuous replication keeps the DR data store synchronized in near-real-time. Daily AMI snapshots are copied to the DR region to ensure a consistent, bootable instance is always available. Under normal operations, the DR EC2 instance remains stopped to minimize cost. On DR activation, the instance starts, mounts the now-writable DR EFS (replication must be deleted before the DR EFS becomes writable), and services resume within the RTO window.



6.2 RTO/RPO Table

These figures require deliberate AWS environment configuration — CHAI does not arrive pre-configured for disaster recovery.

To achieve the RTO/RPO targets below, the following must be explicitly set up in your AWS environment prior to any recovery event:

- EFS cross-region replication enabled between primary and DR regions
- Route 53 health checks and failover routing policy configured for the CHAI ALB
- Daily AMI snapshot copy schedule to the designated DR region
- A stopped standby EC2 instance pre-launched and validated in the DR region

Until all of these components are in place and tested, the figures in this table do not apply to your deployment.

Figures are tentative and based on internal testing. The RTO/RPO values below were derived from testing conducted on a reference HA deployment in a controlled AWS environment. They are provided as planning guidance only — actual recovery times will vary depending on EC2 instance type, EFS data volume at the time of failure, network conditions between AWS regions, and the specific regions used. These targets should be validated through a DR activation drill in your own environment before being incorporated into SLAs or compliance commitments.

Objective	Target	Measurement Method	Notes
RTO (Recovery Time Objective)	< 30 minutes	Time from DR activation decision to CHAI web UI confirmed available on DR URL	Includes: EFS replication deletion (~5 min), EC2 start (~3 min), EFS mount and chctl start (~10 min), DNS propagation (~2 min), service health check validation (~5 min)

Objective	Target	Measurement Method	Notes
RPO (Recovery Point Objective)	< 1 hour	Maximum data loss measured from last EFS replication sync point to failure event	EFS replication is continuous (typically < 15 min lag); RPO < 1 hr is conservative; actual RPO is typically < 15 minutes under normal replication
AMI Recovery Point	< 24 hours	Age of most recent AMI copy in DR region	Daily AMI backup copied to customer-designated DR region; used only if EFS DR data is also unavailable
DR Test Frequency	Quarterly	DR activation drill results documented in DORA resilience testing log	Test activations are conducted on a non-production instance to avoid disrupting production; results reviewed by security and compliance teams
DNS Failover Propagation	< 5 minutes	Route 53 health check interval: 30 sec; TTL: 60 sec; failover effective within 2 health check cycles	Customer DNS may have additional propagation delay if not using Route 53 natively; coordinate DNS cutover timing during DR activation

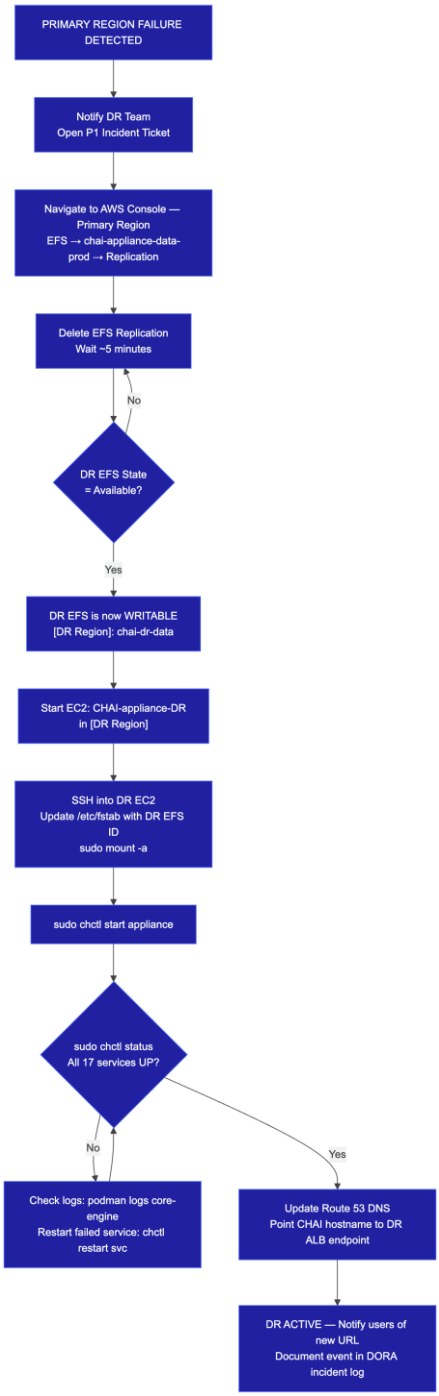
6.3 DR Activation Procedure

The following 9-step procedure is executed by the CloudHedge on-call operations team or designated customer infrastructure team upon confirmation that the primary region deployment is unavailable and cannot be recovered within the established RTO window.

1. **Declare DR Event.** Notify the DR response team via the incident communication channel. Confirm that primary region failure is not a transient AWS issue resolvable within 30 minutes. Reference the incident severity matrix in Section 10.2 to confirm P1 threshold is met.
2. **Navigate to EFS in your primary region.** In the AWS Console, navigate to EFS → chai-appliance-data-prod → Replication tab. Confirm the replication status shows the DR region EFS ID and current lag.
3. **Delete EFS Replication.** Click "Delete replication" on the primary EFS. This action disables the replication relationship and transitions the DR EFS (chai-dr-data in <your-dr-region>) from READ-ONLY to READ-WRITE. Wait approximately 5 minutes for the state transition to complete.
4. **Confirm DR EFS is Writable.** In the AWS Console, navigate to EFS in <your-dr-region>. Confirm the chai-dr-data file system shows state = Available and replication role = None (no longer a replica).
5. **Start DR EC2 Instance.** Navigate to EC2 in <your-dr-region>. Locate the instance tagged CHAI-appliance-DR. Start the instance. Wait for instance state = Running and health checks = 2/2 passed.
6. **SSH into DR EC2 and Mount EFS.** Connect to the DR EC2 instance. Update /etc/fstab to reference the DR EFS file system ID (chai-dr-data ID). Execute sudo mount -a to mount the DR EFS at /mnt/efs. Confirm with df -h | grep efs.
7. **Start CHAI Services.** Execute sudo chctl start appliance. Monitor startup with sudo chctl status. All 17 microservices should reach running state within 5–8 minutes. If any service fails to start, inspect logs with podman logs <service-name>.

8. **Validate Application Availability.** Open the CHAI web UI at the DR ALB DNS name. Confirm login is functional. Run a test discovery or assessment job to validate end-to-end service health.
9. **Update DNS.** Update the Route 53 failover record (or customer DNS) to point the CHAI hostname to the DR ALB endpoint. Notify customer operations team and end users of the temporary DR URL if DNS propagation will take > 5 minutes. Document the event in the DORA incident log.

6.4 DR Activation Flowchart



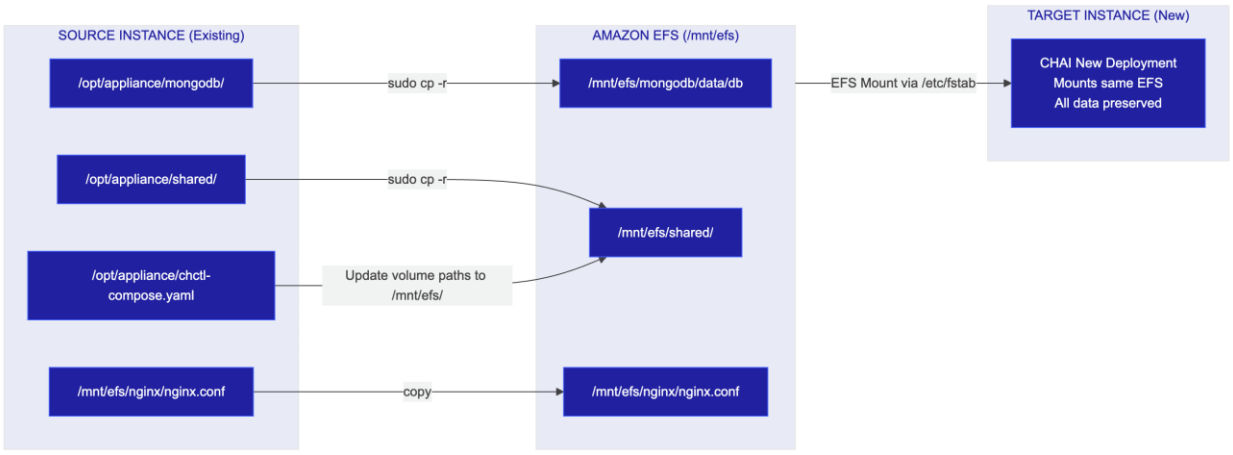
7. Data Migration & Transfer

7.1 Critical Data Inventory

Data Asset	Source Path (Existing Instance)	EFS Destination Path	Contents	Criticality
MongoDB Database	/opt/appliance/mongodb/data/db	/mnt/efs/mongodb/data/db	All CHAI application data: users, tenants, assessments, discovery results, transformation jobs, reports, activity logs	Critical — loss of this data requires full re-discovery and re-assessment
Shared Application Files	/opt/appliance/shared	/mnt/efs/shared	Uploaded source artifacts, generated PDF reports, intermediate job files, transformation output archives	High — recoverable through re-execution but costly in time
Appliance Compose Manifest	/opt/appliance/chctl-compose.yaml	/mnt/efs/shared/chctl-compose.yaml (then symlinked to /opt/appliance/)	Full microservice orchestration definition including environment variables, volume mounts, network config, image references	Critical — without this file the appliance cannot start
Nginx Configuration	/mnt/efs/nginx/nginx.conf or /opt/appliance/nginx/nginx.conf	/mnt/efs/nginx/nginx.conf	Nginx reverse proxy configuration, upstream definitions, SSL settings, custom headers	High — CHAI web UI inaccessible without valid Nginx config
TLS Certificates	/opt/appliance/nginx/ssl/	/mnt/efs/nginx/ssl/	SSL private key (server.key) and certificate chain	High — expired or missing certs

Data Asset	Source Path (Existing Instance)	EFS Destination Path	Contents	Criticality
			(server.crt) for CHAI web UI HTTPS	cause browser security warnings and HTTPS failures
License File	/opt/appliance/license.key	/mnt/efs/shared/license.key	CHAI platform license key — base64 encoded entitlement token issued by CloudHedge	Critical — CHAI will enter degraded mode after grace period if license is missing

7.2 Migration Architecture



7.3 Step-by-Step Migration Procedure

The following procedure migrates an existing CHAI deployment to an EFS-backed architecture, or transfers data from one instance to a new deployment. Execute all steps as root or with sudo.

1. **Stop the CHAI appliance on the source instance.** This ensures MongoDB performs a clean shutdown and all write-ahead logs are flushed before copying data.

Copy

```
sudo chctl stop appliance
sudo chctl status  # Confirm all services are stopped
```

2. **Create and mount the EFS file system.** In the AWS Console, create an EFS file system in the same VPC. Create mount targets in each private subnet. On the source (or new target) EC2 instance, install the EFS mount helper and mount:

Copy

```
sudo yum install -y amazon-efs-utils   # RHEL/AL2023
sudo mkdir -p /mnt/efs
sudo mount -t efs -o tls,iam <EFS_ID>:/ /mnt/efs
```

3. **Create the required EFS directory structure.**

Copy

```
sudo mkdir -p /mnt/efs/mongodb/data/db
sudo mkdir -p /mnt/efs/shared
sudo mkdir -p /mnt/efs/nginx/ssl
sudo chown -R 1000:1000 /mnt/efs/mongodb /mnt/efs/shared /mnt/efs/nginx
```

4. **Copy MongoDB data to EFS.** This is the most time-consuming step for large databases. Use rsync for resume capability:

Copy

```
sudo rsync -avz --progress \
  /opt/appliance/mongodb/data/db/ \
  /mnt/efs/mongodb/data/db/
```

5. **Copy shared application files to EFS.**

Copy

```
sudo rsync -avz --progress \
  /opt/appliance/shared/ \
  /mnt/efs/shared/
```

6. **Copy Nginx configuration and TLS certificates.**

Copy

```
sudo cp /opt/appliance/nginx/nginx.conf /mnt/efs/nginx/nginx.conf
sudo cp /opt/appliance/nginx/ssl/server.key /mnt/efs/nginx/ssl/server.key
sudo cp /opt/appliance/nginx/ssl/server.crt /mnt/efs/nginx/ssl/server.crt
```

7. **Update chctl-compose.yaml volume paths.** Edit the appliance manifest to replace all local /opt/appliance/ volume references with /mnt/efs/ paths. The critical paths are the MongoDB data directory and the shared files directory. Copy the updated manifest to EFS:

Copy

```
sudo cp /opt/appliance/chctl-compose.yaml /mnt/efs/shared/chctl-compose.yaml
```

8. **Add EFS to /etc/fstab for persistent mounting.** Add the fstab entry from Section 5.4 to ensure EFS mounts automatically on every reboot before the CHAI appliance service starts.
9. **Start the CHAI appliance and validate.** Start services and confirm all 17 microservices are running correctly, MongoDB is reading data from the EFS-backed path, and the web UI is accessible:

Copy

```
sudo chctl start appliance
sudo chctl status
# Open web UI and confirm login and data integrity
```

7.4 Amazon Linux 2023 Deployment Notes

CHAI supports both RHEL 8.10 and Amazon Linux 2023 as host operating systems. Organizations may choose either based on their OS standardization policy, licensing model, and AWS optimization preferences. The following notes document the key differences between the two platforms that affect CHAI deployment and operations.

Amazon Linux 2023 vs. RHEL 8.10 — Key Differences:

Package Manager: AL2023 uses dnf exclusively (not yum). All yum install commands must be replaced with dnf install.

EFS Mount Helper: On AL2023, install with sudo dnf install -y amazon-efs-utils. The package is available in the default AL2023 repository — no EPEL or third-party repo required.

Podman Version: AL2023 ships with Podman 4.x (vs. Podman 3.x on RHEL 8.10). The chctl-compose.yaml format is compatible with both versions, but Podman 4.x changes the default network backend from CNI to netavark. Confirm the cloudhedge network is created with podman network create cloudhedge --subnet 10.90.0.0/16 before starting the appliance.

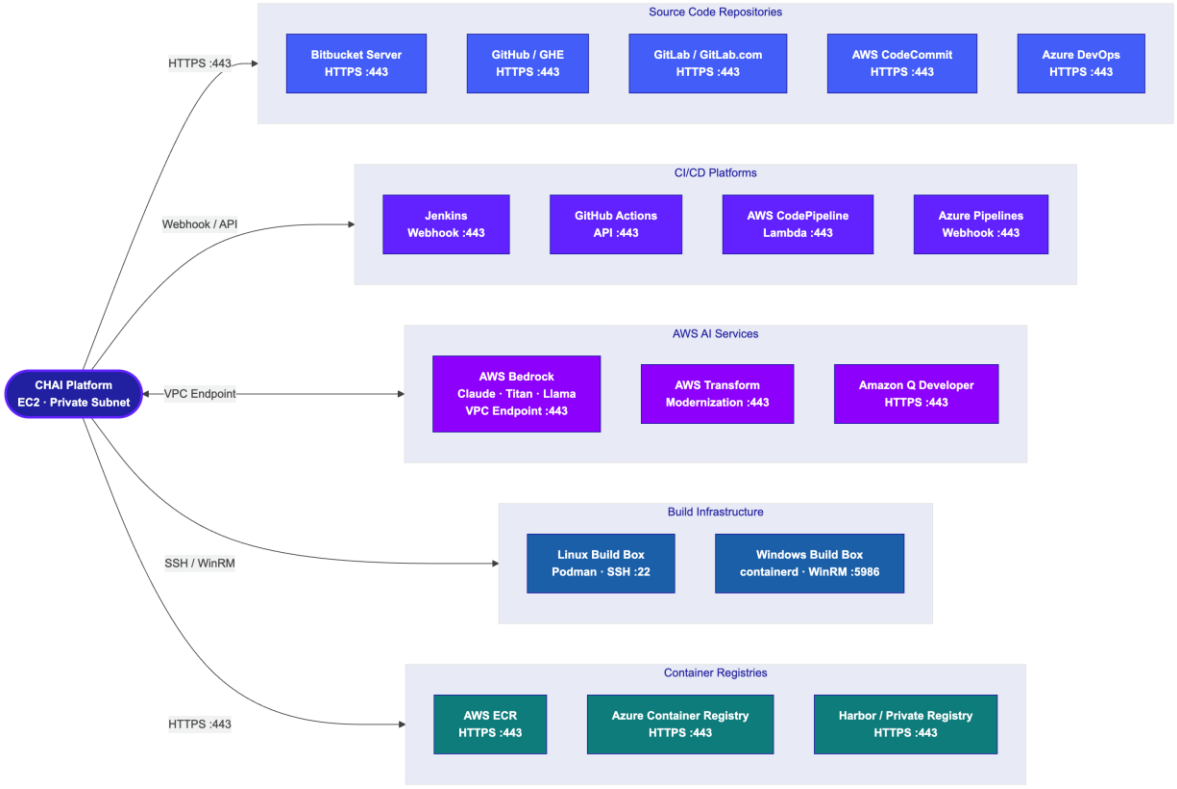
SELinux: AL2023 enables SELinux in enforcing mode by default. EFS mounts require the sefcontext for /mnt/efs to be set correctly. Run sudo restorecon -r /mnt/efs after mounting if Podman containers report permission denied on EFS-mounted volumes.

Systemd Unit: The chctl systemd service unit path is the same on both platforms (/etc/systemd/system/appliance.service). No changes required.

AWS SSM Agent: Pre-installed on AL2023 AMIs. On RHEL 8.10, SSM Agent must be installed manually. AL2023 supports SSM Session Manager out-of-box, enabling SSH-free access for ops teams with appropriate IAM permissions.

8. External System Integrations

8.1 Integration Overview Diagram



This section covers infrastructure-level connectivity and credentials setup. For step-by-step instructions on connecting repositories and running discovery from the CHAI web UI, see the CHAI User Guide.

8.2 Source Code Repository Connectivity Table

Repository System	Protocol	Port	Authentication	CHAI Usage	Configuration Notes
Bitbucket Server (Data Center)	HTTPS	443	HTTP Basic Auth or App Password (read-only token)	Source discovery, clone for transformation, repo metadata extraction	Common enterprise SCM. Configure with your Bitbucket Server base URL and service account credentials stored in the CHAI vault. See Section 8.3 for a step-by-step example.
Bitbucket Cloud	HTTPS	443	App Password with Repositories:Read scope	Source discovery and clone	Workspace URL format differs from server; use https://bitbucket.org as base URL
GitHub.com	HTTPS	443	Personal Access Token (PAT) — repo:read scope	Source discovery, clone, PR creation for transformed code	Fine-grained PATs preferred; classic PATs acceptable with repo scope
GitHub Enterprise Server	HTTPS	443	PAT or GitHub App installation token	Same as GitHub.com; self-hosted base URL required	Ensure TLS certificate is trusted by CHAI EC2 or add to trust store
GitLab.com	HTTPS	443	Project/Group Access Token — read_repository scope	Source discovery and clone	OAuth2 app integration also supported for SSO
GitLab Self-Managed	HTTPS	443	Personal or Project Access Token	Same as GitLab.com; custom hostname required	Internal CA certificate must be added to EC2 trust store
AWS CodeCommit	HTTPS	443	IAM HTTPS Git credentials (not IAM role)	Source discovery and clone	Git credentials generated in IAM console per IAM user; not compatible with IAM roles directly
Azure DevOps / TFS	HTTPS	443	PAT with Code:Read scope	Source discovery, clone, push transformed code	Organization URL format: https://dev.azure.com/<org>
SVN (Apache Subversion)	HTTP/HTTPS / SVN	443 / 3690	Username + password	Legacy source discovery for SVN repositories	CHAI extracts SVN working copy and converts to Git-compatible structure for transformation

Repository System	Protocol	Port	Authentication	CHAI Usage	Configuration Notes
Generic Git (SSH)	SSH	22	SSH key pair (ED25519 recommended)	Any Git host accessible via SSH	CHAI generates an SSH key pair; public key must be added to the repository host's authorized keys

8.3 Bitbucket Server Configuration (Reference Example)

The following walkthrough uses Bitbucket Server as a reference example. CHAI supports all repository systems listed in Section 8.2 — including GitHub, GitLab, Azure DevOps, CodeCommit, and generic Git over SSH. The configuration steps (create service account → generate token → store in CHAI vault → test connection) follow the same pattern for all platforms. For platform-specific setup instructions, see the CHAI User Guide.

Bitbucket Server Integration — Step-by-Step Configuration:

Step 1 — Create Service Account: In Bitbucket Server, create a dedicated service account (e.g., chai-svc-readonly). Grant this account read access to all projects and repositories that CHAI will discover or transform.

Step 2 — Generate HTTP Access Token: Log in as the service account. Navigate to Account Settings → HTTP Access Tokens → Create Token. Set token name: CHAI-Discovery-Token. Set permissions: Repositories: Read, Projects: Read. Set expiry: 1 year. Copy the token value immediately — it will not be shown again.

Step 3 — Store Credentials in CHAI Vault: In the CHAI web UI, navigate to Settings → Source Control → Add Repository. Enter the Bitbucket Server base URL (e.g., https://bitbucket.yourcompany.internal), username (chai-svc-readonly), and the HTTP access token as the password. Click Test Connection.

Step 4 — Verify TLS Certificate: If Bitbucket Server uses an internal CA-signed certificate, the CA certificate must be added to the CHAI EC2 system trust store:
sudo cp your-internal-ca.crt /etc/pki/ca-trust/source/anchors/ && sudo update-ca-trust

Step 5 — Configure Webhook (Optional for CI/CD): In Bitbucket Server, navigate to Repository Settings → Webhooks → Create Webhook. Set URL to the CHAI webhook endpoint: https://chai.yourcompany.internal/api/webhooks/bitbucket. Select events: Push, Pull Request Created. Save. CHAI will receive push events and trigger automated re-assessment on code changes.

Step 6 — Test Discovery: In CHAI, initiate a new Discovery job against the Bitbucket repository. Confirm that repository metadata, branch list, and file tree are successfully retrieved.

8.4 CI/CD Integration Table

CI/CD Platform	Integration Method	Trigger	CHAI Capability	Configuration
Jenkins	Webhook POST to CHAI API	Post-build step or Pipeline stage	Trigger automated re-assessment after build; push transformation artifacts back to repo	Add POST https://chai-host/api/webhooks/jenkins to Jenkins post-build actions; configure CHAI Jenkins credentials in vault
GitLab CI	CHAI pipeline stage in .gitlab-ci.yml	On merge to main/release branch	Run CHAI assessment as a pipeline gate; block merge if assessment score below threshold	Use CHAI REST API in gitlab-ci.yml; store CHAI API key in GitLab CI/CD variables
AWS CodePipeline	Lambda action invoking CHAI API	Source change in CodeCommit/S3	Automated discovery and assessment triggered by code change	Lambda function with CHAI API key calls /api/jobs/assess; CodePipeline waits for CHAI job completion
Azure Pipelines	REST API call in pipeline task	On push to feature/main branch	Assessment and transformation as pipeline stage	Use Azure Pipelines InvokeRestAPI task or custom script task to call CHAI API
GitHub Actions	CHAI action in workflow YAML	On push, pull_request, workflow_dispatch	Automated assessment and report generation in PR workflow	Workflow step calls CHAI REST API using repository secret CHAI_API_KEY

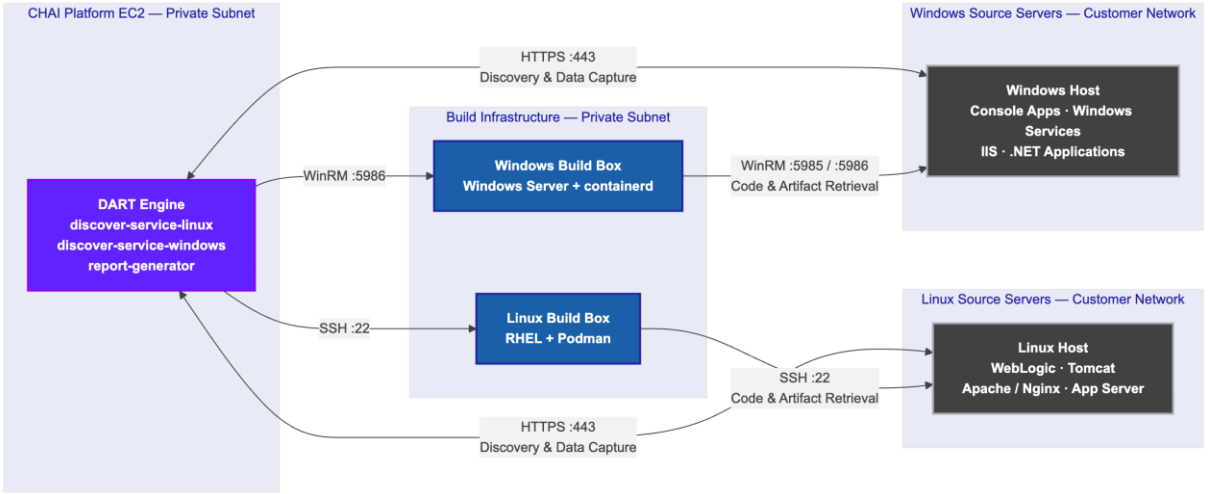
8.5 Build Box Connectivity Table

Build Box	OS	Protocol	Port	Authentication	Purpose	Configuration
Linux Build Box	RHEL 8.10	SSH	22	SSH key pair (stored in CHAI vault)	Builds Linux container images using Podman; executes podman build for transformed Linux workloads; pushes images to ECR	CHAI EC2 must have the build box's IP/hostname in Security Group outbound. Add CHAI EC2's public SSH key to build box's ~/.ssh/authorized_keys
Windows Build Box	Windows Server 2022	WinRM (HTTPS)	5986	Username + password (stored in CHAI vault)	Builds Windows container images using containerd; executes docker build for	Enable WinRM HTTPS on Windows Build Box: winrm quickconfig -transport:https. Add CHAI EC2's outbound :5986

Build Box	OS	Protocol	Port	Authentication	Purpose	Configuration
					transformed Windows workloads; pushes to ECR	to Security Group. Store credentials in CHAI vault

8.5.1 Build Box ↔ Source Server Network Connectivity

For networking and infrastructure teams: This diagram shows the full network path from the CHAI platform through the build boxes to the customer's source servers. Use this to configure security groups, firewall rules, and network ACLs before deployment. Incorrect connectivity at this layer is the most common cause of CHAI deployment issues in enterprise environments.



Security Group Requirements

The following security group rules must be in place **before** CHAI deployment. Coordinate with your network team to open these paths across VPCs, peering connections, or VPNs as applicable to your environment.

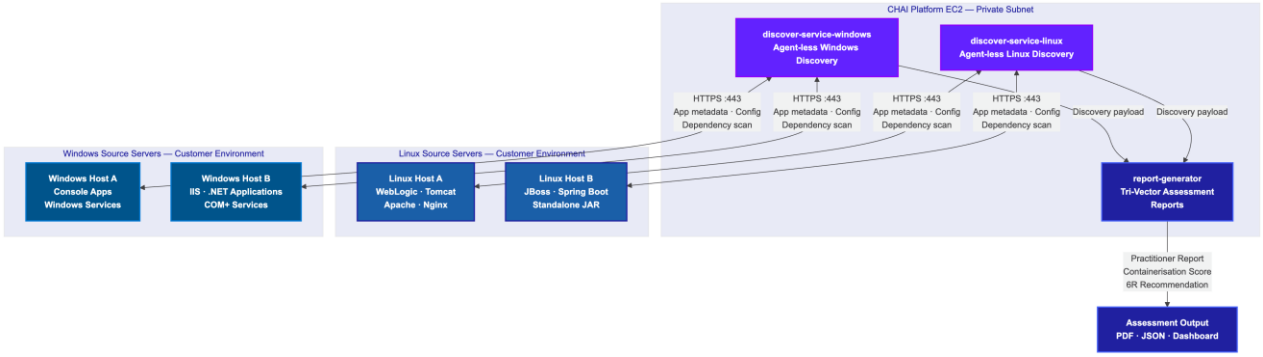
#	Source	Destination	Protocol	Port(s)	Direction	Purpose
1	CHAI EC2 SG	Linux Build Box SG	SSH / TCP	22	Outbound from CHAI	Build orchestration — CHAI issues container build commands to Linux Build Box
2	CHAI EC2 SG	Windows Build Box SG	WinRM / TCP	5986	Outbound from CHAI	Build orchestration — CHAI issues container build commands to Windows Build Box
3	CHAI EC2 SG	Source Server CIDR	HTTPS / TCP	443	Outbound from CHAI	DART discovery — captures application metadata, configs, and dependency data from Linux source hosts

#	Source	Destination	Protocol	Port(s)	Direction	Purpose
4	CHAI EC2 SG	Source Server CIDR	HTTPS / TCP	443	Outbound from CHAI	DART discovery — captures application metadata, configs, and dependency data from Windows source hosts
5	Linux Build Box SG	Source Server CIDR	SSH / TCP	22	Outbound from Linux Build Box	Transformation — Build Box pulls source code and artifacts from Linux servers for container image generation
6	Windows Build Box SG	Source Server CIDR	WinRM / TCP	5985, 5986	Outbound from Windows Build Box	Transformation — Build Box pulls source code and artifacts from Windows servers for container image generation

Note: Rules 5 and 6 (Build Box → Source Server) are the most commonly missed. Ensure firewall rules are opened in both the AWS Security Group **and** any on-premises or customer-side network firewall sitting in front of the source servers.

8.6 DART Engine — Source Discovery

For application and network teams: This diagram shows how the DART Engine discovers application metadata from source servers. The discover-service-linux and discover-service-windows microservices reach out to source hosts over HTTPS (:443), collect application signatures, dependencies, and configuration, and feed the report-generator to produce the Tri-Vector assessment report. No agent is installed on source hosts.



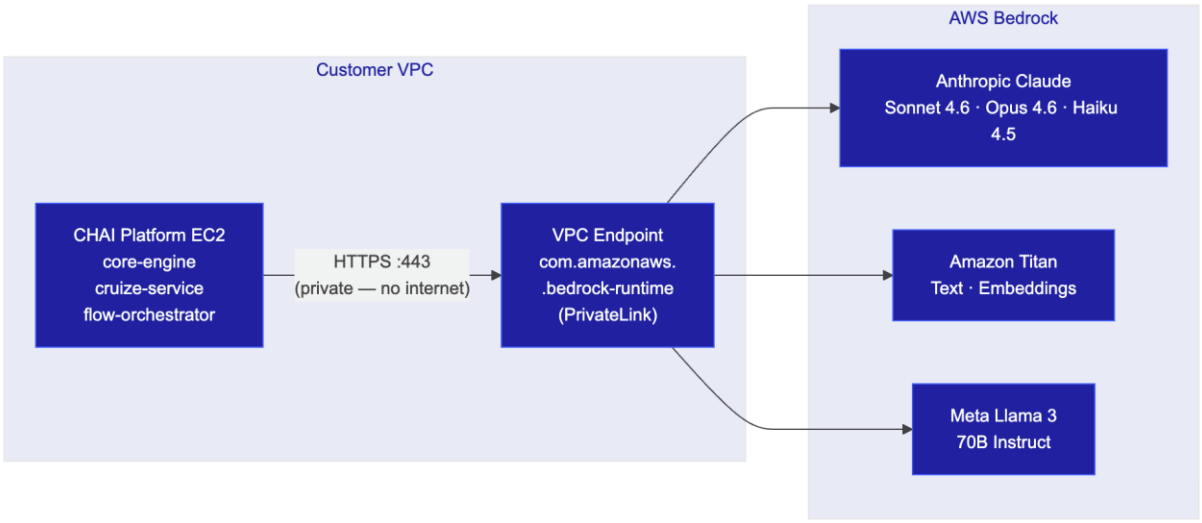
Discovery Service	Target OS	Protocol	Port	Data Collected
discover-service-linux	RHEL, Ubuntu, Amazon Linux	HTTPS / TCP	443	Process list, installed packages, listening ports, JVM args, app server config, file system layout
discover-service-windows	Windows Server 2016–2022	HTTPS / TCP	443	Running services, IIS config, .NET framework versions, registry keys, scheduled tasks, COM+ components

No agent required. DART discovery is agent-less. The CHAI EC2 Security Group must have outbound TCP :443 to the source server CIDR block. No inbound rule changes are needed on the source servers beyond allowing the CHAI EC2 IP to reach them on :443.

9. AWS AI Services Integration

9.1 AWS Bedrock Connectivity

CHAI™ leverages AWS Bedrock as the primary AI inference layer for all large language model operations — including application assessment, code transformation, architecture recommendation, and natural language report generation. Bedrock is accessed via a VPC PrivateLink endpoint, ensuring all model traffic remains within the AWS network and never traverses the public internet. The core-engine and cruize-service microservices are the primary Bedrock consumers; cruize-service handles complex multi-step AI orchestration workflows using chained model invocations.



9.2 Foundation Models Table

Model	Provider	Model ID (Bedrock)	CHAI Use Case	Notes
Claude Sonnet 4.6	Anthropic	anthropic.claude-sonnet-4-6	Primary model for application assessment, 6R rationalization, Tri-Vector scoring, code analysis, and infrastructure recommendation generation	Default model for most CHAI DART and CHAI Flow operations; best balance of quality and throughput
Claude Opus 4.6	Anthropic	anthropic.claude-opus-4-6	Complex multi-file code transformation, architecture pattern inference for large monoliths, deep dependency analysis for 100K+ LOC codebases	Reserved for highest-complexity assessment tasks; higher cost per token; invoked selectively by cruize-service
Claude Haiku 4.5	Anthropic	anthropic.claude-haiku-4-5-20251001	Batch metadata extraction, rapid file classification, service dependency	Fast and cost-efficient; used for pre-processing large application portfolios

Model	Provider	Model ID (Bedrock)	CHAI Use Case	Notes
			parsing, high-volume discovery augmentation	before deeper Sonnet/Opus analysis
Amazon Titan Text	Amazon	amazon.titan-text-express-v1	Fallback text generation for summary reports and natural language descriptions when Anthropic models are unavailable or rate-limited	Available as a secondary option; configured in CHAI model fallback chain
Amazon Titan Embeddings	Amazon	amazon.titan-embed-text-v2:0	Semantic search across application discovery results, similarity matching between applications for portfolio grouping, RAG retrieval for CHAI knowledge base	Used by cruize-service for vector similarity operations; embeddings stored in MongoDB vector index
Meta Llama 3 70B	Meta	meta.llama3-70b-instruct-v1:0	Open-source code analysis tasks, license-sensitive environments where Anthropic models cannot be used, secondary transformation validation	Available as an alternative inference path; configured per-tenant in CHAI vault service

Model Selection Flexibility: CHAI does not require a specific foundation model. Organizations can enable any combination of the models listed above based on their performance, cost, and data-residency requirements. Model selection is configurable per CHAI project — different teams can use different models within the same deployment. For model selection guidance, see the CHAI User Guide.

9.3 Bedrock VPC Endpoint Setup

Step-by-Step: Create Bedrock VPC Endpoint (PrivateLink):

Step 1: In AWS Console, navigate to VPC → Endpoints → Create Endpoint.

Step 2: Set Name tag: chai-bedrock-runtime-endpoint. Select Service category: AWS services. Search for: com.amazonaws.<your-region>.bedrock-runtime. Select the Interface type endpoint.

Step 3: Select the CHAI VPC (chai-prod-vpc). Select all private subnets (AZ-1, AZ-2, AZ-3) for endpoint placement.

Step 4: Security Group: Create a new SG chai-bedrock-vpce-sg allowing inbound HTTPS :443 from chai-ec2-sg.

Step 5: Policy: Select Full access (or use a custom resource policy to restrict to specific model ARNs).

Step 6: Create endpoint. Wait for state = Available (~2 minutes).

Step 7: Enable Private DNS: Ensure "Enable DNS name" is checked on the endpoint. This allows CHAI to use the standard Bedrock Runtime endpoint hostname (bedrock-runtime.<your-region>.amazonaws.com) without any application configuration changes — all traffic is automatically routed through the VPC endpoint.

Step 8: In the CHAI web UI, navigate to Settings → AI Configuration → Test Bedrock Connection to confirm private connectivity.

9.4 AWS Transform

AWS Transform provides additional automated code modernization capabilities that CHAI integrates with for .NET and Java legacy code conversion. CHAI's transform-service-linux and transform-service-windows microservices can invoke AWS Transform jobs as part of the CHAI Flow™ orchestration pipeline, enabling hybrid AI-powered transformation where CHAI handles assessment and workflow orchestration while AWS Transform executes specific language upgrade tasks.

Connectivity: HTTPS :443 to transform.<your-region>.amazonaws.com. No VPC endpoint required; outbound via NAT Gateway or dedicated VPC endpoint.

IAM Permissions Required for AWS Transform:

Permission	Resource	Purpose
transform:StartJob	arn:aws:transform:<your-region>:*:job/*	Initiate a new code transformation job
transform:GetJob	arn:aws:transform:<your-region>:*:job/*	Poll job status and retrieve results
transform:ListJobs	*	List active and completed transformation jobs
s3:GetObject	Source code S3 bucket ARN	Read source code from S3 staging bucket
s3:PutObject	Output S3 bucket ARN	Write transformed code to S3 output bucket
codecommit:GetRepository	Repository ARNs	Read source code from CodeCommit repos
codecommit:CreateBranch	Repository ARNs	Create output branches for transformed code

9.5 Model Usage Guidance

Best Practices: Selecting the Right Model for Each CHAI Task:

Claude Sonnet 4.6 — General Assessment & Standard Transformation: Use Sonnet for the majority of CHAI DART assessment operations, including Tri-Vector scoring, 6R rationalization, dependency map generation, architecture description, and standard containerization transformation for applications under ~50K lines of code. Sonnet provides the best throughput-to-quality ratio and is the default model for all

CHAI DART™ report generation.

Claude Opus 4.6 — Complex Reasoning & Large Codebase Transformation: Reserve Opus for large monolithic applications (100K+ LOC), complex multi-service dependency inference, architectural pattern recognition across polyglot codebases, and transformation tasks requiring multi-turn reasoning chains. Opus is invoked by cruize-service for CHAI Flow™ workflows that require multi-step reasoning. Expect 3–5× higher token cost vs. Sonnet.

Claude Haiku 4.5 — Batch Processing & Fast Inference: Use Haiku for high-volume, low-complexity tasks: file-type classification across large portfolios, metadata extraction from manifests and configuration files, rapid service inventory generation, and pre-screening applications before deep Sonnet/Opus analysis. Haiku delivers near-instant responses and is ideal for portfolios of 100+ applications where initial triage speed matters most.

Titan Embeddings — Semantic Search & Portfolio Clustering: Use Titan Embeddings for semantic similarity operations: grouping similar applications in portfolio analysis, retrieval-augmented generation (RAG) over CHAI's internal knowledge base, and searching assessment results by natural language query. Embeddings are stored in MongoDB with vector indexing.

Cost Optimization: Configure CHAI's AI model routing in Settings → AI Configuration to define per-task model preferences. Use Haiku for triage, Sonnet for standard operations, and Opus only when cruize-service escalates a task due to complexity scoring.

For guidance on how AI models are used within CHAI assessment and transformation workflows — including prompt strategies, output review, and confidence scoring — see the CHAI User Guide.

---

10. Security & DORA Compliance

The European Union's Digital Operational Resilience Act (DORA), established under Regulation (EU) 2022/2554, mandates that financial entities and their critical ICT third-party service providers demonstrate structured, measurable operational resilience. DORA is organized around five pillars: (1) ICT Risk Management — continuous identification, classification, and mitigation of technology risks; (2) ICT-Related Incident Reporting — structured classification, internal escalation, and regulatory notification of significant ICT incidents; (3) Digital Operational Resilience Testing — periodic testing of business continuity, disaster recovery, and threat-led penetration testing (TLPT) for systemically important entities; (4) ICT Third-Party Risk Management — due diligence, contractual obligations, and ongoing monitoring of technology vendors; and (5) Information and Intelligence Sharing — participation in industry threat intelligence sharing arrangements. CHAI's infrastructure architecture, operational procedures, and security controls have been designed to support the licensee's compliance obligations under each of these five pillars.

10.1 DORA Compliance Matrix

DORA Article	Requirement	CHAI™ Implementation	Evidence / Artifact
Art. 5 — ICT Risk Management Framework	Maintain and update ICT risk management framework	VPC isolation, Security Groups, IAM least privilege, TLS encryption in transit, EFS encryption at rest (AES-256)	VPC architecture diagram; IAM policy JSON; EFS encryption settings
Art. 9 — Protection & Prevention	Protect ICT systems from unauthorized access and ensure data integrity	EC2 in private subnet (no public IP); ALB as sole ingress; SSH restricted to admin CIDR; Podman rootless containers; no embedded credentials (all via IAM and SSM Parameter Store)	Security group rules; IAM instance profile; SSM Parameter Store config
Art. 10 — Detection	Detect anomalous activities promptly	CloudWatch alarms on CPU, memory, error rates, service health; CloudTrail API logging; CHAI activity-service audit log for all user and system actions	CloudWatch dashboard; CloudTrail S3 bucket; activity-service MongoDB audit collection
Art. 11 — Response & Recovery	Define and test response and recovery procedures	DR activation procedure (Section 6.3); ASG auto-recovery (Section 5.1); RTO < 30 min, RPO < 1 hr documented and tested quarterly	DR runbook; ASG configuration; quarterly DR drill log
Art. 17 — Incident Classification	Classify ICT incidents by severity and report per	P1–P4 severity matrix (Section 10.2); notification-service automates P1/P2 alerts; DORA-specific incident report	Incident severity matrix; notification-service webhook configuration

DORA Article	Requirement	CHAI™ Implementation	Evidence / Artifact
	regulatory requirements	template maintained in shared documentation	
Art. 24 — Resilience Testing	Conduct regular resilience testing including DR drills and (for significant entities) TLPT	Quarterly DR activation drills; ECR image scanning via AWS Inspector; ASG recovery tests; MongoDB backup restoration tests	Quarterly test reports; Inspector scan results; backup restoration logs
Art. 25 — TLPT (significant entities)	Conduct threat-led penetration testing every 3 years	AWS Inspector network reachability; third-party VAPT engagement recommended for organizations classified as DORA significant entities	Inspector findings; VAPT engagement scope documentation
Art. 28 — Third-Party ICT Risk	Assess and monitor ICT third-party service providers	SBOM published with all third-party components (Section 10.3); ECR image scanning; CloudHedge vendor SLA; read-only source repo tokens; no third-party write access to production	SBOM document; ECR scan reports; vendor contract SLA terms
Art. 45 — Information Sharing	Participate in cyber threat intelligence sharing	Security advisories published per release; SBOM shared with customer security team upon request; CloudHedge security advisory mailing list; release notes include security fix summaries	Release notes; security advisory emails; SBOM distribution records

10.2 Incident Severity Matrix

Severity	Definition	Example Scenarios	Response SLA	DORA Reporting
P1 — Critical	Complete CHAI platform unavailability affecting all users; primary region EC2 failure with ASG unable to recover; EFS data loss or corruption	Primary EC2 terminated and ASG fails to launch replacement within 15 min; EFS file system unavailable; MongoDB data corruption detected; all 17 services down; security breach or unauthorized data access	Acknowledge: 15 min; Mitigate: 30 min (RTO target); Resolve: 4 hours	Report to customer DORA Incident Manager within 2 hours of detection per the applicable support agreement; CloudHedge escalation to CTO
P2 — High	Partial service degradation affecting key CHAI functions; 1–3 critical microservices down but platform partially operational;	core-engine or auth-gateway unavailable; Bedrock connectivity lost; ECR pull failures preventing service restart; ALB health check failures on > 1 target	Acknowledge: 30 min; Mitigate: 2 hours; Resolve: 8 hours	Notify customer operations team within 4 hours; CloudHedge support ticket P2

Severity	Definition	Example Scenarios	Response SLA	DORA Reporting
	performance degraded > 50%			
P3 — Medium	Non-critical service degradation; minor feature unavailability; intermittent errors affecting < 10% of requests	report-service slow or failing (reports not generating); notification-service down (email alerts delayed); ch-user-guide unavailable; non-critical integration (Bitbucket webhook) not functioning	Acknowledge: 2 hours; Mitigate: 8 hours; Resolve: 24 hours	Notify customer operations team within 24 hours; CloudHedge support ticket P3
P4 — Low	Informational; minor issues with no user impact; planned maintenance windows; non-urgent configuration changes	SSL certificate expiry warning (> 30 days out); log volume approaching CloudWatch retention limit; minor security advisory in non-critical dependency; documentation update needed	Acknowledge: 24 hours; Resolve: 5 business days	No regulatory notification required; document in change log

10.3 SBOM Management

CloudHedge maintains a published Software Bill of Materials (SBOM) for every CHAI release. The SBOM documents all first-party and third-party software components and their versions. Image scanning is performed automatically on every ECR image push via AWS Inspector Enhanced Scanning, and a full SBOM report is generated with each release. Customers may request the SBOM for any release from their CloudHedge account team.

SBOM Technology Stacks:

Layer	Components
Runtime — Node.js	Node.js 16.16.x / >=18.20.x, Express.js 4.18.x, jsonwebtoken 9.0.x, bcrypt 5.1.x, passport-saml 3.1.x, mongoose 7.x
Runtime — Frontend	React 18.2.x, webpack 5.x, axios 1.x
Runtime — Go	Go >=1.10.x (cruize-service, aws-lift-shift, discover-service-windows, transform-service-windows, discover-service-aix)
Runtime — Python	Python 3.6.x (discover-service-linux, transform-service-linux, discover-service-aix)
Infrastructure	Nginx 1.22.x, MongoDB 7.0.12, Podman 3.x/4.x
Base Images	RHEL UBI 8.10.x, RHEL UBI 9.5.x (selected services)
Auth	jsonwebtoken 9.0.x, bcrypt 5.1.x, passport-saml 3.1.x

10.4 IAM Policy Reference

The following IAM policy represents the minimum permissions required for the CHAI EC2 instance role. This policy follows the principle of least privilege and should be attached to an IAM role assigned to the CHAI EC2 instance profile. No inline policies, wildcard accounts, or administrative permissions should be granted.

Copy

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "EFSAccess",
   "Effect": "Allow",
   "Action": [
    "elasticfilesystem:ClientMount",
    "elasticfilesystem:ClientWrite",
    "elasticfilesystem:ClientRootAccess"
   ],
   "Resource": "arn:aws:elasticfilesystem:<your-region>:<ACCOUNT_ID>:file-system/<EFS_ID>"
  },
  {
   "Sid": "SSMParameterAccess",
   "Effect": "Allow",
   "Action": [
    "ssm:GetParameter",
    "ssm:GetParameters",
    "ssm:GetParametersByPath"
   ],
   "Resource": "arn:aws:ssm:<your-region>:<ACCOUNT_ID>:parameter/chai/*"
  },
  {
   "Sid": "ECRAccess",
   "Effect": "Allow",
   "Action": [
    "ecr:GetAuthorizationToken",
    "ecr:BatchGetImage",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchCheckLayerAvailability"
   ],
   "Resource": "*"
  },
  {
   "Sid": "BedrockAccess",
   "Effect": "Allow",
```

```
    "Action": [
      "bedrock:InvokeModel",
      "bedrock:InvokeModelWithResponseStream"
    ],
    "Resource": [
      "arn:aws:bedrock:<your-region>::foundation-model/anthropic.claude-sonnet-4-6",
      "arn:aws:bedrock:<your-region>::foundation-model/anthropic.claude-opus-4-6",
      "arn:aws:bedrock:<your-region>::foundation-model/anthropic.claude-haiku-4-5-20251001",
      "arn:aws:bedrock:<your-region>::foundation-model/amazon.titan-text-express-v1",
      "arn:aws:bedrock:<your-region>::foundation-model/amazon.titan-embed-text-v2:0",
      "arn:aws:bedrock:<your-region>::foundation-model/meta.llama3-70b-instruct-v1:0"
    ]
  },
  {
    "Sid": "CloudWatchMetricsAndLogs",
    "Effect": "Allow",
    "Action": [
      "cloudwatch:PutMetricData",
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents",
      "logs:DescribeLogGroups",
      "logs:DescribeLogStreams"
    ],
    "Resource": "*"
  }
 ]
}
```

10.5 Access Control Matrix

Role	CHAI Web UI	SSH to EC2	AWS Console (CHAI resources)	ECR Push	Bedrock Direct Access	Incident Response
CHAI End User	Full access (assigned tenants)	No	No	No	No	View P3/P4 tickets
CHAI Platform Administrator	Full access (all tenants)	No	Read-only (CloudWatch)	No	No	Create/manage P1–P4 tickets
Customer Infrastructure Engineer	No	Yes (via bastion, key-based)	Read/Write (EC2, EFS, ALB)	No	No	P1/P2 DR activation authority

Role	CHAI Web UI	SSH to EC2	AWS Console (CHAI resources)	ECR Push	Bedrock Direct Access	Incident Response
CloudHedge Support Engineer	Admin access (support sessions only)	Yes (authorized during incidents)	Read-only (CloudWatch, ECR)	No	No	P1/P2 incident response; escalation to CloudHedge CTO
CloudHedge Release Engineer	No	No	ECR Push (CI/CD pipeline role only)	Yes (specific ECR repos)	No	N/A
Customer Security / Compliance	No	No	Read-only (CloudTrail, Config, Inspector)	No	No	DORA incident reporting; SBOM review
Automated (CI/CD Pipeline)	No	No	Scoped: ECR push, ASG launch, ALB registration	Yes	No	Automated P4 alert creation on deployment failure

11. Operational Runbook

This section covers infrastructure operations (start, stop, upgrade, backup, monitor). For application-level operations — managing users, tenants, projects, and running assessments — see the CHAI User Guide.

11.1 Appliance Lifecycle Commands

All CHAI appliance lifecycle operations are performed using the chctl command-line tool, which abstracts Podman orchestration into human-readable, atomic operations. chctl must be run as root or with sudo.

Command	Description	When to Use	Expected Output
sudo chctl start appliance	Start all 17 CHAI microservices in dependency order	Post-boot startup; after a planned maintenance window; after DR activation	All services transition to running state within 5–8 minutes

First-Time Deployment? After starting the appliance for the first time, access the CHAI web UI at https://<alb-hostname> to complete initial setup: create the first admin user, activate the license key, and configure your first project. See the CHAI User Guide — Getting Started for the step-by-step setup wizard walkthrough.

| sudo chctl stop appliance | Gracefully stop all 17 services in reverse dependency order (MongoDB last) | Before taking an AMI snapshot; before data migration; planned maintenance | All services transition to stopped state; MongoDB performs clean shutdown |

| sudo chctl restart appliance | Stop then start all 17 services | After updating chctl-compose.yaml; after a

configuration change affecting multiple services | Full restart cycle; expect ~10 minutes total |
sudo chctl restart <service>	Restart a single named service (e.g., core-engine)	After a single service crash; after updating a single service's environment variable	Named service restarts without affecting other services
sudo chctl status	Display running state of all 17 services	Health verification; post-startup validation; troubleshooting	Table showing service name, container ID, state, port, uptime
sudo chctl status <service>	Display detailed status of a single named service	Deep troubleshooting of a specific service	Container metadata, environment variables, mount points, recent log tail
sudo chctl logs <service>	Tail live logs for a named service (wraps podman logs -f)	Real-time troubleshooting; monitoring startup sequences	Streaming log output; Ctrl+C to exit
sudo chctl upgrade	Pull latest container images from ECR and restart services with new images	During planned release upgrades (e.g., 2.1.6-12 → 2.1.7.18)	Pulls new images, stops old containers, starts new containers; data preserved on EFS
sudo chctl backup	Trigger an on-demand MongoDB backup to the configured backup path	Before major changes; supplement to automated EFS backup policy	Backup archive created at /mnt/efs/backups/<timestamp>/
sudo chctl health	Run end-to-end health check across all services and external integrations	Pre-incident validation; post-upgrade verification	Pass/Fail status per service and integration; total health score

11.2 Appliance Upgrade Procedure

To upgrade CHAI to a new release version:

1. **Review release notes** for the target version at the CloudHedge support portal
2. **Create a backup AMI** of the current running instance: sudo chctl snapshot
3. **Pull new images** from ECR: sudo chctl pull --tag ch-rel-<new-version>
4. **Stop the appliance:** sudo chctl stop appliance
5. **Start with new images:** sudo chctl start appliance
6. **Verify health:** sudo chctl health — all services should show PASS
7. **Verify version:** sudo chctl version — should show the new release tag

Rollback: If the upgrade fails health checks, run sudo chctl pull --tag ch-rel-<previous-version> followed by sudo chctl stop appliance && sudo chctl start appliance to restore the previous version. MongoDB data on EFS is version-independent and is not affected by container image upgrades.

For detailed release notes and upgrade advisories, contact CloudHedge support or visit the CHAI User Guide.

11.3 Log Management

CHAI microservice logs are written to stdout/stderr within Podman containers and captured by CloudWatch Logs Agent running on the EC2 instance. Log retention is configured per-service based on volume and compliance requirements.

Log File Paths (on EC2 instance):

Log Type	Path / Log Group	Retention	Description
Podman Container Logs	podman logs <service-name> (real-time)	In-memory (container lifetime)	Accessed via chctl logs <service> or directly with podman logs
CloudWatch Log Group — Core	/chai/core-engine	90 days	Core orchestration, job dispatch, Bedrock API calls
CloudWatch Log Group — Auth	/chai/auth-gateway-service	90 days	Authentication events, JWT validation failures, SAML assertions
CloudWatch Log Group — All Services	/chai/<service-name>	30 days (standard services)	Per-service application logs for all 17 microservices
CloudWatch Log Group — Activity	/chai/activity-service	365 days	DORA-relevant audit log — all user actions and system events; extended retention for compliance
CloudTrail Logs	S3: s3://chai-audit-logs-<account>/cloudtrail/	90 days	All AWS API calls; IAM actions; EC2, EFS, ECR operations
Nginx Access Logs	/mnt/efs/nginx/logs/access.log	30 days (rotated by logrotate)	All HTTP/HTTPS requests to the CHAI web UI
Nginx Error Logs	/mnt/efs/nginx/logs/error.log	30 days (rotated by logrotate)	Nginx-level errors; upstream connection failures

11.4 Monitoring & Alerting

CloudWatch alarms are configured to alert the operations team on the following conditions. All P1/P2-threshold alarms route to the customer incident management system via SNS topic → webhook → ticketing integration.

Metric	CloudWatch Namespace	Alarm Threshold	Alarm Action	DORA Relevance
EC2 CPU Utilization	AWS/EC2	> 85% for 5 consecutive minutes	SNS → P2 alert; consider EC2 resize	ICT Risk — performance degradation signal
EC2 Status Check Failed	AWS/EC2	StatusCheckFailed_Instance = 1	SNS → P1 alert; ASG terminates and replaces instance	ICT Risk — instance-level hardware failure

Metric	CloudWatch Namespace	Alarm Threshold	Alarm Action	DORA Relevance
EFS ClientConnections	AWS/EFS	= 0 for 3 consecutive minutes (during business hours)	SNS → P1 alert	ICT Risk — EFS connectivity lost; MongoDB data unavailable
EFS PercentIOLimit	AWS/EFS	> 80% for 10 consecutive minutes	SNS → P2 alert; consider EFS provisioned throughput	ICT Risk — storage I/O saturation affecting MongoDB performance
ALB Healthy Host Count	AWS/ApplicationELB	< 1 for 2 consecutive minutes	SNS → P1 alert; trigger DR assessment	ICT Incident — all CHAI instances unhealthy
ALB HTTP 5xx Error Rate	AWS/ApplicationELB	> 5% of requests over 5 minutes	SNS → P2 alert	Incident Detection — backend service errors
CloudWatch Logs — ERROR rate	/chai/core-engine (custom metric filter)	> 10 ERROR log events per minute	SNS → P3 alert	Incident Detection — application-level error spike
MongoDB EFS Bytes Written	Custom (from activity-service)	= 0 for 15 minutes during active jobs	SNS → P2 alert	ICT Risk — MongoDB not writing; potential data integrity issue

11.5 Health Check Procedure

Execute this procedure after any appliance restart, upgrade, DR activation, or infrastructure change to confirm full operational status.

1. **Verify all services are running.**

```
Copy
sudo chctl status
# Expected: All 17 services in "running" state; no "exited" or "error" status
```

2. **Confirm EFS is mounted.**

Copy

```
df -h | grep efs
# Expected: /mnt/efs mounted with expected available capacity
mount | grep efs
# Expected: EFS ID visible with tls,iam options
```

3. **Confirm MongoDB is accessible and data is present.**

Copy

```
podman exec -it db-service mongosh --eval "db.adminCommand('ping')"
# Expected: { ok: 1 }
podman exec -it db-service mongosh --eval "show dbs"
# Expected: chai_db visible with non-zero size
```

4. **Test ALB health endpoint.**

Copy

```
curl -k https://<ALB_DNS_NAME>/api/health
# Expected: HTTP 200; JSON body with {"status":"healthy","services":17}
```

5. **Confirm Bedrock connectivity.**
   In the CHAI web UI: Settings → AI Configuration → Test Bedrock Connection.
   Expected: "Connection successful" with response latency < 2 seconds.
6. **Verify CloudWatch log ingestion.**
   In AWS Console: CloudWatch → Log Groups → /chai/core-engine. Confirm log events are being written within the last 5 minutes.
7. **Run CHAI health command.**

Copy

```
sudo chctl health
# Expected: All services PASS; all integrations PASS; overall health score 100%
```

8. **Confirm web UI login.**
   Open the CHAI web URL in a browser. Log in with an admin account. Confirm the dashboard loads and displays existing application portfolio data from MongoDB.

Appendix A — Port Reference

A.1 Internal Ports

Ports used by CHAI microservices for inter-service communication. All traffic runs on the cloudhedge Podman network (10.90.0.0/16) and does not traverse the VPC or internet. These ports are never exposed to end users or external systems.

#	Service	Port	Protocol	Direction	Notes
1	webapp (Nginx HTTP)	8080	HTTP / TCP	ALB → webapp	SSL terminated at ALB; webapp receives plain HTTP from ALB
2	webapp (Nginx HTTPS)	8443	HTTPS / TCP	Internal	Direct HTTPS if bypassing ALB; used in dev/test only
3	auth-gateway-service	3000	HTTP / TCP	webapp → auth-gateway	Called by webapp for all authentication and authorization checks
4	core-engine	3001	HTTP / TCP	webapp, auth-gateway → core-engine	Central orchestration; receives all API calls from webapp
5	cloud-infra-service	3002	HTTP / TCP	core-engine → service	Called by core-engine for cloud resource provisioning operations
6	notification-service	3005	HTTP / TCP	core-engine → service	Called by core-engine; delivers email and webhook notifications
7	vault-service	3006	HTTP / TCP	core-engine, all services → vault	Encrypted credential storage; called by all services needing secrets
8	activity-service	3007	HTTP / TCP	All services → activity	Audit log service; all user and system events recorded here
9	report-service	3009	HTTP / TCP	core-engine → service	Report generation; produces DART assessment and compliance artifacts
10	license-service	3011	HTTP / TCP	core-engine → service	License validation; called on startup and periodically during operation
11	cruize-service	3012	HTTP / TCP	core-engine → service	AI orchestration; manages Bedrock model invocations and prompt chaining
12	aws-lift-shift	4005	HTTP / TCP	core-engine → service	Lift-and-shift migration executor for EC2/ECS/EKS rehosting
13	discover-service-linux	5001	HTTP / TCP	core-engine → service	Linux discovery coordinator; manages remote SSH discovery sessions

#	Service	Port	Protocol	Direction	Notes
14	transform-service-linux	5003	HTTP / TCP	core-engine → service	Linux transformation; AI-assisted Dockerfile generation for Linux apps
15	transform-service-windows	5004	HTTP / TCP	core-engine → service	Windows transformation; converts Windows apps via Windows Build Box
16	discover-service-windows	5005	HTTP / TCP	core-engine → service	Windows discovery coordinator; manages remote WinRM discovery sessions
17	discover-service-aix	5006	HTTP / TCP	core-engine → service	AIX/UNIX discovery coordinator; manages remote SSH discovery on IBM AIX
18	ch-user-guide	8001	HTTP / TCP	webapp → service (proxied)	Static in-product documentation; proxied via Nginx in webapp
19	db-service (MongoDB)	27017	MongoDB Wire / TCP	core-engine, vault, activity, report → DB	Primary datastore; all service state and assessment data
20	Amazon EFS	2049	NFS v4.1 / TCP	CHAI EC2 → EFS	NFS mount for persistent storage; encrypted via TLS mount helper

A.2 External Ports

Ports used by users or external systems to communicate with CHAI. These ports must be open on the relevant security groups and accessible from the specified sources.

#	Component	Port	Protocol	Direction	Source	Notes
1	ALB (HTTPS)	443	HTTPS / TCP	Inbound → ALB	End users / enterprise network	Primary access point; ACM certificate; routes to webapp :8080
2	ALB (HTTP redirect)	80	HTTP / TCP	Inbound → ALB	End users / enterprise network	HTTP to HTTPS redirect only; no application traffic
3	CHAI EC2 (Admin SSH)	22	SSH / TCP	Inbound → EC2	Ops team / bastion host	Administrative access; key-based authentication; restrict to admin CIDR
4	Linux Build Box	22	SSH / TCP	CHAI EC2 → Build Box	CHAI platform EC2	CHAI initiates SSH for Linux container builds; key-based auth via vault-service
5	Windows Build Box	5986	WinRM HTTPS / TCP	CHAI EC2 → Build Box	CHAI platform EC2	CHAI initiates WinRM for Windows container builds; credentials via vault-service

#	Component	Port	Protocol	Direction	Source	Notes
6	Source Hosts (Linux discovery)	22	SSH / TCP	CHAI EC2 → source hosts	CHAI platform EC2	Outbound SSH to Linux application servers during DART discovery phase
7	Source Hosts (Windows discovery)	5986	WinRM HTTPS / TCP	CHAI EC2 → source hosts	CHAI platform EC2	Outbound WinRM to Windows application servers during DART discovery phase
8	AWS ECR / Bedrock / SSM / S3	443	HTTPS / TCP	CHAI EC2 → AWS services	CHAI platform EC2	Via VPC PrivateLink endpoints where configured; NAT Gateway fallback
9	Source Code Repositories	443	HTTPS / TCP	CHAI EC2 → SCM	CHAI platform EC2	GitHub, GitLab, Bitbucket; read-only token-based access
10	MCP SSE Server	30080	HTTP SSE / TCP	CHAI Universe → MCP server	CHAI Universe client	Optional CHAI MCP™ integration; configurable port; see CHAI MCP Installation Guide

Appendix B — AWS Cost Estimate

Cost Estimate Notes: The following estimates are monthly costs for a single CHAI deployment in your primary region using on-demand pricing. Actual costs vary based on deployment topology (single EC2 vs Multi-AZ), instance family, EFS usage, Bedrock inference volume, and applicable AWS enterprise discount. Reserved Instance or Savings Plans pricing can reduce EC2 costs by 30–60%. For a customized TCO estimate based on your deployment requirements, contact CloudHedge sales.

The following estimates are based on primary region on-demand pricing as of March 2026. Actual costs will vary based on usage patterns, reserved instance commitments, and enterprise discount programs. All estimates assume continuous 24/7 operation.

AWS Service	Configuration	Estimated Monthly Cost (USD)	Notes
EC2 — CHAI Primary	m7i.xlarge, on-demand, 24/7	$220	~$0.302/hr; use Reserved Instance (1-yr) for ~40% savings (~$132/mo)
EC2 — Linux Build Box	t3.large, on-demand, 24/7	$60	Can be stopped when no builds running; savings-mode: ~$15/mo if stopped 75% of time
EC2 — Windows Build Box	t3.large, Windows, on-demand, 24/7	$115	Windows licensing premium; can be stopped when not in use
Application Load Balancer	1 ALB, ~100 GB processed/mo	$20	$0.008/hr + $0.008/LCU-hr; minimal LCU usage for typical CHAI traffic

AWS Service	Configuration	Estimated Monthly Cost (USD)	Notes
Amazon EFS	100 GB average stored, General Purpose	$35	$0.30/GB-month standard storage; 100 GB × $0.30 = $30 + throughput
EFS Replication (DR)	Continuous replication to DR region	$25	$0.25/GB replicated; assumes 100 GB × $0.25
EC2 — DR Standby (DR Region)	m7i.xlarge, stopped (EBS charges only)	$15	EBS 150 GB gp3 at $0.10/GB-month; EC2 stopped = no instance charge
EFS DR (DR Region)	100 GB replica	$30	Same rate as primary EFS
AWS ECR	30 GB stored (all release tags)	$3	$0.10/GB-month; data transfer within region free
AWS Bedrock	Claude Sonnet 4.6 — ~2M tokens/month	$60	$3.00/1M input tokens, $15/1M output tokens; estimate 2M mixed
Amazon CloudWatch	Logs 10 GB/month, 8 alarms, dashboards	$25	Log ingestion $0.50/GB; metrics and alarms additional
AWS CloudTrail	Multi-region trail, S3 storage	$10	First trail free; S3 storage for logs
NAT Gateway	1 NAT GW, ~50 GB processed/mo	$35	$0.045/hr ($33/mo) + $0.045/GB data processed
Data Transfer Out	~20 GB/month to users	$2	First 100 GB/month at $0.09/GB
AWS Systems Manager	Parameter Store standard tier	$0	Standard parameters free; advanced: $0.05/parameter/month
Total Estimated Monthly	—	~$655/month	Excludes Marketplace AMI subscription fee; excludes enterprise discounts
With Reserved Instances (1-yr)	EC2 m7i.xlarge + t3.large reserved	~$470/month	~28% savings via RI commitment on compute

Appendix C — Deployment Checklist

Appendix D — Legal & Trademarks

Intellectual Property

This document and all referenced CHAI platform software are the intellectual property of CloudHedge Technologies Inc. All rights reserved. No part of this document may be reproduced, distributed, or

transmitted in any form or by any means without the prior written permission of CloudHedge Technologies Inc., except as expressly authorized in the applicable software license agreement between CloudHedge Technologies Inc. and the licensee.

Trademarks

CHAI™, CHAI DART™, CHAI Flow™, and CHAI Universe™ are trademarks of CloudHedge Technologies Inc., registered or pending registration in the United States and other jurisdictions. R6Ai® is a registered trademark of CloudHedge Technologies Inc. The CloudHedge name and logo are trademarks of CloudHedge Technologies Inc.

All other trademarks, service marks, product names, and company names mentioned in this document are the property of their respective owners. References to third-party products, services, and trademarks do not constitute or imply endorsement, sponsorship, or affiliation by CloudHedge Technologies Inc.

Regulatory References

- EU Regulation 2022/2554 (Digital Operational Resilience Act / DORA) — European Parliament and Council, December 14, 2022
- AWS Shared Responsibility Model — Amazon Web Services Inc.
- NIST SP 800-53 Rev. 5 — Security and Privacy Controls for Information Systems and Organizations
- CIS AWS Foundations Benchmark v1.4

License & Warranty Disclaimer

THE CHAI PLATFORM SOFTWARE IS PROVIDED UNDER THE TERMS OF THE CLOUDHEDGE ENTERPRISE LICENSE AGREEMENT. THE PLATFORM IS PROVIDED "AS IS" WITH RESPECT TO THIRD-PARTY OPEN-SOURCE COMPONENTS. CLOUDHEDGE TECHNOLOGIES INC. MAKES NO WARRANTIES, EXPRESS OR IMPLIED, REGARDING THIRD-PARTY SOFTWARE COMPONENTS IDENTIFIED IN THE SBOM. CLOUDHEDGE'S OBLIGATIONS WITH RESPECT TO DEFECTS, UPTIME, AND SUPPORT ARE EXCLUSIVELY GOVERNED BY THE APPLICABLE ENTERPRISE LICENSE AGREEMENT AND SUPPORT ADDENDUM.

Legacy Product Note

For historical reference only: CHAI is the successor platform to the CloudHedge legacy product line. All new deployments use the CHAI platform. References to legacy product nomenclature in prior customer agreements should be interpreted as referencing the current CHAI platform for all purposes of the applicable agreement.

Contact

CloudHedge Technologies Inc.
33 Wood Avenue S, Suite 600
Iselin, NJ 08830, USA
Web: cloudhedge.io
Support: support@cloudhedge.io
Security: security@cloudhedge.io