TrustHouse.AI on AWS Marketplace: Deployment Guide

- **Version:** TrustHouse 2.0.0 (container image tag `trusthouse-v2.0.2-mp2`)
- **Product:** TrustHouse.AI: Trust and Context Layer for Production AI, AWS Marketplace product `prod-6mzewdmbilgws`
- **Seller:** Arhasi AI with Integrity (Arhasi Inc.). Support: contactus@trusthouse.ai
- **Date:** October 2026

This guide explains, step by step, how TrustHouse is deployed into your own AWS account, what you do, what Arhasi does, what it costs to run, and how to operate it. It is written for your AWS administrator and your security team.

Contents

1. Overview and architecture
2. Who does what, in order
3. Prerequisites
4. Subscription and licence
5. The deployment role (ArhasiDeployer)
6. Connecting your AWS data: the TrustHouseAccess role
7. Sign-in
8. AgentOps working mode and scanning
9. External connections and downloads
10. Sensitive data, encryption and key rotation
11. Backup and restore
12. Health checks and monitoring
13. AWS service quotas
14. AWS cost estimate
15. Upgrades to new versions
16. Removing access, or removing TrustHouse
17. Support and known issues

Appendix A: Container images. Appendix B: What Arhasi runs.

1. Overview and architecture

TrustHouse is a trust and context layer for AI agents and models. It runs as containers inside your own AWS account and uses Amazon Bedrock in that account for AI models, so prompts, responses and business data are processed in your AWS account, except the lineage graph, which is kept in Neo4j Aura. Section 9 lists every connection outside your account.

How it is deployed. Arhasi deploys TrustHouse for you, into your AWS account and the Region you choose, on Amazon ECS with AWS Fargate (no EC2 instances). Arhasi uses its deployment tool and a limited IAM role that you create from Arhasi's CloudFormation template (section 5). Every TrustHouse container comes from this version's images in the AWS Marketplace registry (Appendix A). You do not run containers or scripts yourself.

1.1 What runs in your account

Part	AWS service	Default size (full install)
Application	Amazon ECS on AWS Fargate, one cluster `arhasi-<environment>-cluster`	48 services, one task each: about 21 vCPU and 55 GB in total. Autoscaling may add tasks under load
Network	Amazon VPC (10.0.0.0/16, two Availability Zones), one NAT gateway, Application Load Balancer	Services and databases in private subnets; only the load balancer is public
Web access	Route 53 hosted zone, AWS Certificate Manager certificate	HTTPS on port 443; HTTP redirects to HTTPS
Main database	Amazon DocumentDB 5.0	2 x db.t3.medium (writer and reader, two zones)
Cache and queues	Amazon ElastiCache for Valkey 7.2	2 x cache.t4g.micro (primary and replica)
Graph database	Neo4j Aura (section 9)	Agreed per customer

Part	AWS service	Default size (full install)
Files	Amazon S3, nine buckets `arhasi-<environment>-<account>-<name>`	Versioned, private, encrypted
Secrets	AWS Secrets Manager, names starting `arhasi/<environment>/`	
AI models	Amazon Bedrock in your account	Claude Sonnet 4.6, Opus 4.6, Opus 4.5, Haiku 4.5, Titan Text Embeddings V2
Licence	AWS License Manager in your account	Checked every 5 minutes
Logs and alarms	Amazon CloudWatch Logs and alarms, Amazon SNS topic `arhasi-<environment>-alarms`	Logs kept 30 days by default
AgentOps	A second VPC (10.60.0.0/16), AWS CodeBuild, a private Amazon ECR repository, a second ECS cluster	Up to 10 agents (section 8)
Small helpers	AWS Lambda	AWS CDK's own helper functions and one knowledge-base upload function

`<environment>` is the short name Arhasi agrees with you for your install, for example `acme`.

Arhasi installs the capabilities your subscription covers. A full install runs all 48 services. Services talk to each other inside the VPC through ECS Service Connect; users reach the web apps and APIs through the load balancer, each on its own address under your TrustHouse domain.

2. Who does what, in order

#	Who	Step	Where, and time
1	You	Subscribe to TrustHouse.AI in the AWS account where TrustHouse will run (public listing or your private offer) and choose your tier	AWS Marketplace, a few minutes
2	You	Email contactus@trusthouse.ai with your AWS account ID, Region, domain choice, sign-in choice (Google, Microsoft or both), and any optional features	Email
3	Arhasi	Sends you the deployment role template, and your External ID by a separate channel	
4	You	Create the deployment role stack and tell Arhasi it is created (section 5)	CloudFormation, about 5 minutes
5	You, guided by Arhasi	One-time AWS License Manager setup, and activate the licence if it shows as not activated (section 4)	License Manager console, about 2 minutes
6	Arhasi	Read-only checks through the role: access and Bedrock model access. Arhasi also gives you the quotas to check (section 13)	
7	You, guided by Arhasi	Close any gap Arhasi found: quota increase requests (section 13), Bedrock first-use form for Anthropic models	Consoles; quota reviews can take hours to a day
8	Arhasi	Prepares: subdomain and certificate, sign-in addresses (section 7), Neo4j Aura database, Bitbucket OAuth consumer, alarm email subscriptions	
9	Arhasi	Deploys network, storage and databases, sets the secrets, then deploys the 48 services from the Marketplace images	Your account, about 1 to 2 hours
10	Arhasi	Turns on AgentOps working mode (a second, short deploy that needs the network from step 9)	15 to 25 minutes
11	Arhasi	Verifies: every service healthy and running this version's Marketplace image, sign-in works, licence active. Hands over your TrustHouse address	
12	You	Confirm the alarm email subscription from your inbox; sign in; invite your users	
13	You	When TrustHouse should read your AWS data (Bedrock agents, S3, Athena, Secrets Manager): create the TrustHouseAccess role (section 6)	TrustHouse and CloudFormation, about 10 minutes
14	You	Keep the deployment role for upgrades and support, or delete it (section 5)	

No AWS access keys are created or exchanged at any step.

3. Prerequisites

Item	What is needed	Who arranges it
AWS account and Region	The account TrustHouse runs in, and one Region from the listing. The model identifiers TrustHouse uses today are Amazon Bedrock's US cross-Region inference profiles; for a Region outside the United States, Arhasi confirms model availability before deploying	You choose
Subscription	An active TrustHouse.AI contract in that account (or activated there, section 4)	You
AWS administrator	Someone who can create CloudFormation stacks that create named IAM roles (`CAPABILITY_NAMED_IAM`), and use the License Manager, Service Quotas, and Amazon Bedrock consoles	You
Service quotas	As in section 13	Arhasi says which; you check and request
Bedrock models	The five models in section 1.1 usable in your Region, including the Anthropic first-use form	You submit, Arhasi checks
Domain	A subdomain for TrustHouse with a Route 53 hosted zone in your account and an ACM certificate for it and its wildcard. Either a subdomain Arhasi provides under Arhasi's domain, or a subdomain of your own domain	Arhasi
Users	Google or Microsoft accounts for everyone who signs in	You
Graph database	A Neo4j Aura database, agreed per customer	Arhasi
AgentOps repositories	Your own Bitbucket or GitHub repositories	You

4. Subscription and licence

Pricing. TrustHouse uses AWS Marketplace contract pricing with three tiers: Essential, Production and Autonomous (purchase a quantity of 1). The listing describes what each tier covers.

How the licence is checked. TrustHouse checks your subscription itself through AWS License Manager in your own account. There is no licence key to enter and no call to any outside licensing server. TrustHouse checks at start-up and then every 5 minutes: it reads your TrustHouse licences, checks out the highest tier a live licence holds, and hands a counted unit straight back. Its IAM role may call `license-manager:ListReceivedLicenses`, `CheckoutLicense` and `CheckInLicense` in your Region only (AWS offers no narrower resource for these actions).

One-time setup (needed before first sign-in). A new account has not used License Manager yet, and Arhasi's deployment role cannot do this step. Your administrator:

1. Opens the AWS License Manager console in the TrustHouse Region.
2. Chooses **Start using License Manager**, ticks **I grant AWS License Manager the required permissions**, and chooses **Continue**. This creates the service-linked role `AWSServiceRoleForAWSLicenseManagerRole`. Without it, TrustHouse cannot see your licence and refuses every sign-in.

Activate the licence if needed. In License Manager, open **Granted licenses** and find TrustHouse.AI. If it shows as not activated (common when the subscription was bought in an AWS Organizations management account and shared to the member account TrustHouse runs in), select it and choose **Accept and activate**.

What users see.

Situation	Result
Active subscription	Sign-in works. A tier change in Marketplace is picked up at the next check, within 5 minutes
No subscription, contract ended, or licence not activated in this account	Sign-in is refused with "No active AWS Marketplace licence for this product"
License Manager cannot be reached (network, throttling, permissions)	Sign-in keeps working for up to 72 hours after the last successful check. Before the first successful check there is no grace period, so sign-in waits for the next check

5. The deployment role (ArhasiDeployer)

You create this role once, in the TrustHouse account and Region. It is the only way Arhasi reaches your account.

What the stack creates. Stack `arhasi-deployer-access` creates one IAM role, `ArhasiDeployer`, and three managed policies. It creates nothing that costs money. Only Arhasi's deployment account can assume the role, and only when it presents your External ID. Sessions last at most 4 hours, because a full deploy takes about 1 to 2 hours. No person at Arhasi gets console access and no long-lived access key exists.

Parameters

Parameter	Value
ArhasiDeployerAccountId	Leave the default (Arhasi's deployment account) unless Arhasi tells you otherwise
ExternalId	The value Arhasi sent you by a separate channel. It is hidden in the console (NoEcho)
RoleName	Leave `ArhasiDeployer` unless the name is taken; then tell Arhasi
AllowedRegions	Your TrustHouse Region, for example `us-east-1`. The role is refused everywhere else

Create it in the console: CloudFormation, **Create stack**, **With new resources**, upload the template, stack name `arhasi-deployer-access`, fill the parameters, tick the IAM acknowledgement, create. Or with the AWS CLI:

```
aws cloudformation create-stack --stack-name arhasi-deployer-access \
  --template-body file://client-onboarding.yaml --capabilities CAPABILITY_NAMED_IAM \
  --parameters ParameterKey=ExternalId,ParameterValue=<value from Arhasi> \
               ParameterKey=AllowedRegions,ParameterValue=<your Region>
```

Then tell Arhasi the stack is created and send the **RoleArn** output. Never send the External ID back.

Purpose of each policy

Policy	Purpose
`ArhasiDeployer-cdk`	Guardrails (below); AWS CDK bootstrap and deploy; creating the IAM roles that TrustHouse services run with
`ArhasiDeployer-infrastructure`	ECS and Fargate, VPC and load balancer, DocumentDB and ElastiCache, backup schedules and event rules, CDK helper Lambda functions. All limited to AllowedRegions
`ArhasiDeployer-support`	Secrets Manager, Route 53, ACM certificates, CloudWatch logs and alarms, the alarm SNS topic, a read-only Bedrock model check, use of AWS-managed encryption keys, read-only cost reports

Always denied (Deny statements, so nothing can grant them back): changing the role itself; creating IAM users, access keys, console passwords or identity providers; AWS Organizations and account settings; domain registration; launching EC2 instances. Regional permissions apply only in AllowedRegions.

Be aware. The role builds infrastructure, so it is powerful inside those limits. Arhasi's first deploy runs AWS CDK bootstrap, which creates the `CDKToolkit` stack: a staging S3 bucket, an ECR repository for deployment assets, and roles, including a CloudFormation execution role with AdministratorAccess (this is how AWS CDK works). CloudFormation, not the ArhasiDeployer role, then creates TrustHouse's resources.

Revoking. Delete the `arhasi-deployer-access` stack. New access stops at once; sessions already issued expire within 4 hours. Nothing deployed is deleted, and TrustHouse keeps running, but Arhasi can no longer upgrade or repair it until you create the stack again. To also remove the CDK roles, delete `CDKToolkit` afterwards, never before.

Changing the External ID. Ask Arhasi for a new value, update your stack's ExternalId parameter, and Arhasi updates its side. Expect a short window in which Arhasi's access fails.

6. Connecting your AWS data: the TrustHouseAccess role

TrustHouse never asks for AWS access keys, and refuses them if entered. To let TrustHouse read your Bedrock agents, S3 buckets, Athena tables or Secrets Manager secrets, you create one IAM role from a template TrustHouse prepares. You decide what the role may reach, and you can narrow or delete it at any time.

6.1 Create the role

1. In TrustHouse open **Integrations**, then **AWS role** (needs the permission to add integrations). The same panel appears in each AWS connection form. It shows the TrustHouse AWS account, your organisation's External ID and

your role name `TrustHouseAccess-<8 characters>`.

2. Choose **Download CloudFormation template**. The file `trusthouse-access-role.yaml` is already filled with those values.
3. In the AWS account that holds your data (the TrustHouse account, or another of yours), create a CloudFormation stack from it named after your role. Leave the four pre-filled parameters as they are (ExternalId, TrustHouseAccountId, TrustHouseStackPrefix, RoleNameSuffix), set the ones you need from the table below, tick the IAM acknowledgement and create. The panel also shows the equivalent `aws cloudformation deploy` command.
4. Copy the stack output **RoleArn** into the AWS connection form (Bedrock, Bedrock LLM, S3, Athena or AWS Secrets Manager) with the Region. Athena also needs the S3 output location and, optionally, the workgroup (`primary` if empty).

Connection	Parameters	Default
Bedrock agents (Agent Catalog, Agent Lineage, Agent Quality, agent chat)	`EnableBedrockAgents=true`	true
Configure Agent Lineage (one-time monitoring setup)	`EnableMonitoringSetup=true`, `CreateInvocationLoggingRole=true`	both true
Bedrock models (model chat)	`EnableBedrockModels=true`	true
S3 (also metadata exports for file crawls, and the SSIS, Azure Data Factory and PySpark folders the lineage stitcher reads)	`S3BucketNames=<bucket-1>,<bucket-2>`; add `S3ReadWrite=true` if agents should upload, rename or delete files	empty, false
Athena (Data Catalog, Data Lineage, Data Quality, Data Profile)	`AthenaWorkGroup`, `AthenaResultsBucket` (bucket of the S3 output location), `AthenaDataBuckets` (buckets of the queried tables), optionally `AthenaDatabases`	empty; all databases
AWS Secrets Manager	`SecretArns=<arn-1>,<arn-2>` (wildcards allowed)	empty
Your own KMS keys on those buckets or secrets	`KmsKeyArns=<key-arn>,...`	empty
Bedrock agents whose action groups read API schemas from S3	`AgentSchemaBuckets=<bucket>,...`	empty

A bucket name may end in `*` (for example `acme-data-*`). An empty list turns that part off. Create one stack per organisation per AWS account: each organisation has its own External ID.

6.2 See what the role can reach

In TrustHouse 2.0.0 you check this in AWS:

1. **CloudFormation console:** open the stack `TrustHouseAccess-<suffix>`, tab **Parameters**. It lists which connections are on and every bucket, workgroup, database, secret and key you named.
2. **IAM console: Roles**, `TrustHouseAccess-<suffix>`, tab **Permissions**. Expand each policy to see the exact actions and resources. Tab **Trust relationships** shows that only TrustHouse's service roles of your install, with your External ID, may use it.

TrustHouse 2.0.0 has no in-app view of the role's permissions; use the two consoles above.

6.3 Add buckets or services later

Do not create a second stack; update the one you have. Console: select the stack, **Update**, **Use existing template**, change the parameters, update. CLI: run the same `aws cloudformation deploy` command with the new `--parameter-overrides`. Parameters you do not name keep their values, but **a list you do name replaces the old list**, so repeat every bucket or secret you still need. Use the same RoleArn in the new connection.

Use the stack parameters rather than editing the role by hand in IAM: a later stack update can undo manual edits.

6.4 What TrustHouse can and cannot do with the role

Who can use it	Only TrustHouse service roles of your install (IAM path `/arhasi-<environment>/`), and only with your organisation's External ID
How long	At most one hour per session, renewed automatically. Download links it creates stop working when that hour ends
Never	TrustHouse's own buckets and secrets (explicit Deny); creating or changing IAM roles

Wide by design (narrow it if you wish)	Listing agents, guardrails, bucket names and secret names (never secret values); reading any Bedrock agent, its action-group Lambda (including code) and foundation model in the account; all Glue databases unless you set AthenaDatabases

6.5 Configure Agent Lineage: read before running

It runs through your role in the Region of your Bedrock connection and creates or reuses: buckets `arhasi-bedrock-logs-<account>`, `arhasi-cloudtrail-<account>` and `arhasi-athena-results-<account>`; the CloudTrail trail `arhasi-audit-trail`; the Glue database `cloudtrail_db`; the log group `/aws/bedrock/modelinvocations`; and Bedrock model invocation logging delivered with the role `service-role/model-invocation-role`, which your stack creates.

- It **replaces** any Bedrock model invocation logging settings already in that Region. If your account already has invocation logging, or CloudTrail trails or Glue tables you rely on, contact support before running it.
- If the role `service-role/model-invocation-role` already exists in the account, set `CreateInvocationLoggingRole=false`, or the stack fails.
- If your agents run in the TrustHouse account, the invocation logs also hold TrustHouse's own model calls. Agent Lineage leaves them out and only its log reader may open them; to keep them fully apart, use another account or Region for your agents.

6.6 Messages you may see

Message starts with	What to do
"AWS access keys are not accepted on TrustHouse from AWS Marketplace"	Recreate the connection with a Role ARN
"TrustHouse could not assume ..." or "AWS refused TrustHouse access ..."	The stack is missing or was made for another install or organisation: download the template again and update or recreate the stack
"AccessDenied" from a connection, or "Error connecting to the database" on a Data Catalog or Data Quality connection	The role does not allow that bucket, workgroup or secret: add it to the stack parameters and update the stack
"Bedrock model invocation logging is not set up in this account"	Run Configure Agent Lineage first
"The role service-role/model-invocation-role is not in this account"	Update the stack with `EnableMonitoringSetup=true` and `CreateInvocationLoggingRole=true`

To stop TrustHouse reaching your account, delete the stack.

7. Sign-in

Users sign in with their Google or Microsoft account. Sign-in uses Arhasi's Google and Microsoft app registrations, the same ones every TrustHouse install uses: Arhasi adds your install's web addresses to them. On request, Arhasi can point your install at your organisation's own Microsoft Entra ID or Google app registration instead; the client ID and tenant are per-install settings. Each sign-in exchanges data with Google or Microsoft (section 9).

8. AgentOps working mode and scanning

AgentOps builds and runs **your own agent code** inside your account. You connect a Bitbucket or GitHub repository, TrustHouse builds it into a container image with AWS CodeBuild, stores it in a private ECR repository, scans it, and runs it on Fargate in a separate VPC.

Connecting repositories. Bitbucket: one click from the Integrations screen, through a Bitbucket OAuth consumer Arhasi registers for your install; its client secret is stored in Secrets Manager in your account. You can instead enter a Bitbucket API token with your email. GitHub: a personal access token (read access to contents is enough), entered on the Integrations screen. Tokens are stored encrypted in your account (section 10). The worker only clones from bitbucket.org and github.com.

What Arhasi creates for it (stack `arhasi-<environment>-agentops`, deployed with your install):

Piece	Details
Agents' network	A separate VPC (10.60.0.0/16, two public subnets, no NAT), VPC flow logs kept 7 days, and a private peering link to TrustHouse's VPC

Piece	Details
Firewalls	Agents may call out only on ports 80 and 443 to the internet, never into TrustHouse's network. A private agent accepts port 8080 from TrustHouse only. An agent you publish with a public endpoint accepts port 8080 from the internet over HTTP
Build	One CodeBuild project (Linux small, 2 builds at a time, 30 minutes), on AWS's own build image. Build files in bucket `arhasi-<environment>-<account>-agentops-builds`, deleted after 1 day
Images	ECR repository `agentops/<environment>/user-images`: scan on push, tags cannot be overwritten, images kept 14 days
Runtime	ECS cluster `arhasi-<environment>-agentops`: at most 1 vCPU and 2 GB per agent, 10 agents in all, 3 per organisation by default, each stopped 8 hours after its last deploy
Permissions	Agents get no AWS permissions. Agent secret bindings are stored in Secrets Manager (`agentops/<environment>/agents/<agent>`) and reach the agent as ECS secrets, never as plain text
Alarm	Agents' log volume above 500 MB per hour

New agents are **private by default**; publishing a public endpoint is your choice per agent.

Vulnerability scanning options

Option	What happens	Notes
Amazon ECR basic scanning (default)	ECR scans each agent image on push; TrustHouse shows the findings (CVE, package, fixed version, severity) on the build	No extra setup; source code never leaves your account
Amazon ECR enhanced scanning (Amazon Inspector), your choice	ECR fetches findings from Amazon Inspector; TrustHouse shows them the same way. For this the scanner's role holds the read-only `inspector2:ListFindings` and `inspector2:ListCoverage` on all resources (AWS offers no narrower scope)	An account-level registry setting that also affects your other ECR repositories, and Amazon Inspector bills per image. Your registry's scanning rules must cover `agentops/<environment>/user-images`, or agent images are not scanned. Arhasi agrees the change with you first
Veracode (optional, off by default)	Source code scanning by Veracode, only with your own Veracode account and licence (paid by you to Veracode); Arhasi stores your Veracode API credentials in Secrets Manager in your account and turns it on	When on, your agent source code is sent to Veracode (api.veracode.com)

9. External connections and downloads

No TrustHouse container image comes from outside AWS Marketplace. Every TrustHouse service runs from this version's images in the AWS Marketplace registry `709825985650.dkr.ecr.us-east-1.amazonaws.com` (Appendix A). The registry is in us-east-1; installs in other Regions pull across Regions through the NAT gateway, and AWS charges that transfer. Besides the images, the deployment uploads AWS CDK's own helper Lambda functions and one small TrustHouse Lambda function (Python, reacts to knowledge-base uploads) through the CDK staging bucket in your account.

Connection	When	What goes there	Required?	Extra payment?
AWS services in your account (Bedrock, License Manager, S3, Secrets Manager and others in section 1.1)	Always	Your data stays in AWS. Bedrock's US cross-Region inference profiles may process a prompt in another US Region (for example us-east-1, us-east-2, us-west-2)	Yes	AWS usage on your AWS bill
Google or Microsoft sign-in	Every sign-in	The sign-in exchange	Yes, one of them	No
Neo4j Aura (TLS)	Always	The lineage graph, including agent prompts and responses once Agent Lineage runs	Yes	Agreed per customer
Hugging Face (huggingface.co)	AI Risk Engine, first use of its entity-extraction model	Downloads open-source model files only; no customer data is sent	Yes	No

Connection	When	What goes there	Required?	Extra payment?
Users' browsers	While using TrustHouse	Downloads only: web fonts from Google Fonts (fonts.googleapis.com, fonts.gstatic.com), sign-in button icons (img.icons8.com) and the PDF viewer's script (unpkg.com). Those hosts see the user's IP address and browser	Yes	No
Arhasi's DNS (only if your TrustHouse address is under Arhasi's domain)	Always	DNS delegation of your TrustHouse subdomain, which Arhasi controls	Optional	No
Bitbucket (bitbucket.org, api.bitbucket.org) or GitHub (github.com, api.github.com)	Connecting a repository, each AgentOps build	Your token; your source code is downloaded into your account	Optional (AgentOps)	Your own account
Public base images and packages	Each AgentOps build	Downloads only: base images from Amazon ECR Public (Python, Node.js, Java), any registry your own Dockerfile names, and packages from PyPI, npm, Maven Central or Gradle, and Debian mirrors. Python agents also get fastapi and uvicorn from PyPI	Optional (AgentOps)	No
Your running agents	While running	Whatever your agent code calls, on ports 80 and 443	Optional	No
Veracode	Only if turned on	Your agent source code	Optional, off by default	Your own Veracode account, paid by you (section 8)
Your own data sources and MCP servers (for example Amazon Redshift, PostgreSQL, SQL Server, Snowflake, Databricks)	When you connect them	Queries and metadata, with your own logins	Optional	Your own accounts

To stop an optional connection, delete it in TrustHouse (and revoke the token or key at the provider). The licence check never leaves your account. AgentOps sends your source code outside your account only to Veracode, and only if you turn it on. TrustHouse sends no telemetry to Arhasi.

10. Sensitive data, encryption and key rotation

10.1 Where sensitive data is stored

Data	Where (all in your account unless stated)	Protection
Users, organisations, roles, policies, catalogs, settings	DocumentDB cluster `arhasi-<environment>-docdb`, private subnets	Encrypted at rest (AWS-managed key `aws/rds`); TLS in transit
Credentials you enter in connection forms (database passwords, Snowflake keys, Databricks tokens, repository tokens)	DocumentDB, encrypted by TrustHouse with its application encryption key before storage	Fernet (AES-128-CBC with HMAC-SHA256), plus DocumentDB encryption
Secrets you create in TrustHouse's secret store	Secrets Manager, names starting `arhasi-<environment>/<organisation id>/`	AWS-managed key `aws/secretsmanager`
AgentOps agent secrets	Secrets Manager `agentops/<environment>/agents/<agent>`, readable only by the agents' execution role at start	Same

Data	Where (all in your account unless stated)	Protection
Platform secrets: token signing keys, application encryption key, internal service keys, Bitbucket OAuth client secret, Veracode API ID and key if Veracode is on	Secrets Manager `arhasi/<environment>/...`	Passed to containers as Secrets Manager references, never as plain text in task definitions or templates
Database credentials and connection strings	Secrets Manager `arhasi/<environment>/mongodb/...`, `.../redis/...`, `.../neo4j/credentials`	DocumentDB password rotated every 30 days
Cache, queues, sessions	ElastiCache for Valkey	Encrypted at rest and in transit (TLS with AUTH token)
Files: uploads, knowledge-base documents, lineage exports, agent outputs, manifests	Nine S3 buckets `arhasi-<environment>-<account>-<name>`	Server-side encryption (SSE-S3), Block Public Access, TLS required, versioning; kept if stacks are deleted
Lineage graph	Neo4j Aura	TLS; stored by Neo4j
Application logs	CloudWatch Logs, 30 days by default	Encrypted by CloudWatch Logs
Bedrock invocation logs (after Configure Agent Lineage)	Bucket `arhasi-bedrock-logs-<account>`, log group `/aws/bedrock/modelinvocations`	Readable only by the lineage reader's session through your role

Decrypting. Users never decrypt anything. TrustHouse services decrypt a stored credential only when they use that connection. Anyone in your account with `secretsmanager:GetSecretValue` on these secrets can read them, so limit that permission. Web traffic is HTTPS (TLS) at the load balancer with your ACM certificate; service-to-service traffic stays inside the VPC's private subnets.

10.2 Key rotation

Material	Rotation
DocumentDB master password	Automatic, every 30 days (Secrets Manager rotation)
ACM certificate	Renewed automatically by ACM while its DNS validation record stays in place
Valkey AUTH token	Not automatic; rotate as a planned change with Arhasi
Token signing key, job token key, internal service keys	Not automatic, by design. Rotated on request in a planned window: changing the token signing key signs every user out; changing the job token key fails jobs in progress
Application encryption key	Never change or delete it on its own: stored credentials become unreadable and your organisations' External IDs change. Rotate only with Arhasi support, then re-enter saved connection credentials and update each TrustHouseAccess stack
Deployment role External ID	On request (section 5)
Your own tokens, keys and database passwords	Rotate them at the source, then update the connection in TrustHouse
AWS access	No access keys exist. TrustHouseAccess sessions last at most one hour, deployment role sessions at most four

11. Backup and restore

Data	Backup	Restore
DocumentDB (main database)	Automatic daily snapshots in the 02:00 to 03:00 UTC window, kept 7 days (up to 35 on request). Deletion protection is on, and deleting the cluster leaves a final snapshot	Below
Valkey (cache, queues, sessions)	Automatic daily snapshots, same window, kept 7 days	ElastiCache console, **Backups**, restore to a new cluster. Rarely needed: this data rebuilds itself

Data	Backup	Restore
S3 buckets	Versioning keeps earlier versions of every object; buckets are kept if stacks are deleted	S3 console, **Show versions**, restore the version you need
Neo4j Aura	Neo4j's own backups for your Aura plan	Aura console
Secrets Manager	Deleted secrets stay recoverable during their recovery window	Secrets Manager console

Take a manual DocumentDB snapshot (before an upgrade or a big change): DocumentDB console, **Clusters**, select `arhasi-<environment>-docdb`, **Actions**, **Take snapshot**.

Restore DocumentDB: DocumentDB console, **Snapshots**, select the snapshot, **Actions**, **Restore**. This creates a **new** cluster with a new endpoint. Arhasi then points the connection secret `arhasi/<environment>/mongodb/uri` at it and restarts the services. We recommend testing a restore once, before you need one.

12. Health checks and monitoring

Check that TrustHouse is healthy (your AWS console, in the TrustHouse Region):

1. Open your TrustHouse address and sign in. The home page loads and your organisation is shown.
2. **Amazon ECS** console, **Clusters**, `arhasi-<environment>-cluster`, tab **Services**: every service shows running tasks equal to desired tasks, and its last deployment is completed. Services are named `arhasi-<environment>-<service>`.
3. **EC2** console, **Target Groups**: every target is **healthy**. The load balancer checks each web service on `/health` and ECS replaces tasks that fail.
4. **CloudWatch** console, **Alarms**: every alarm whose description starts with `[<environment>]` is **OK**.
5. **CloudWatch Logs**: each service has a log group whose name starts with `arhasi-<environment>-<service>`. Filter on `ERROR` to see recent errors.
6. **License Manager**, **Granted licenses**: the TrustHouse licence is active.

Alarms go to the SNS topic `arhasi-<environment>-alarms`. Arhasi subscribes the email addresses you give; each person must confirm from their inbox, or nothing is delivered.

Alarm	Fires when (default layout)
CPU, memory	A service above 85% CPU or memory for two 5-minute periods
No running tasks	A service has no running task (this needs Container Insights, which is on by default)
Unhealthy hosts	Any load balancer target is unhealthy
5xx errors	More than 10 server errors per 5 minutes from a service
Databases	DocumentDB or Valkey above 80% CPU, or more than 500 connections
AgentOps	Agents' logs above 500 MB per hour

Neo4j Aura has no AWS alarm; use the Aura console.

13. AWS service quotas

Arhasi's deployment role cannot read quotas, so your administrator checks these before the deploy, with Arhasi's help on what to raise. To check or raise one: **Service Quotas** console, **AWS services**, choose the service, choose the quota, **Request increase at account level**. Approval can take hours; start early. A higher quota costs nothing by itself; you pay only for resources you run (section 14).

Quota	Needed for a full install	Notes
Amazon ECS: Fargate On-Demand vCPU resource count	About 21 vCPU steady, up to about 42 during an upgrade (old and new tasks overlap), plus up to 10 for AgentOps agents	New accounts can start lower
Amazon ECS: Fargate Spot vCPU resource count	Extra tasks that autoscaling adds under load	Autoscaling can take a service to 4 tasks
AWS CodeBuild: concurrently running builds for Linux/Small environment	At least 2	Below 2 the AgentOps stack can fail; new accounts can start at 1
Amazon VPC: VPCs per Region	2 more	TrustHouse and agents

Quota	Needed for a full install	Notes
Amazon VPC: NAT gateways per Availability Zone; Amazon EC2: Elastic IP addresses	1 more of each	The NAT gateway's public address
Elastic Load Balancing: Application Load Balancers; rules per load balancer	1; about 19 host rules	Default limits are usually enough
Amazon Bedrock: tokens and requests per minute for the five models	Enough for your usage; Arhasi flags any model below 100,000 tokens per minute	Model use is billed per token

14. AWS cost estimate

These are **AWS charges on your own AWS bill** for the infrastructure TrustHouse runs on. They are separate from the TrustHouse subscription price. Figures come from Arhasi's deployment tool, at us-east-1 list prices for a full install (all capabilities, 48 services) on the default layout. **Idle, a full install costs about USD 34 a day.** With fewer capabilities it costs less; Arhasi gives you an estimate for your install before deploying. Actual cost varies with usage, Region, number of users and agents, and the infrastructure layout you choose.

Line	Per month	Per day
48 containers on Fargate, one task each, on-demand (about 21 vCPU, 55 GB)	USD 792	USD 26.01
DocumentDB, 2 x db.t3.medium, plus storage and backup (50 GB assumed)	USD 120	USD 3.94
ElastiCache for Valkey, 2 x cache.t4g.micro	USD 23	USD 0.77
NAT gateway (100 GB of traffic assumed)	USD 37	USD 1.23
Application Load Balancer and capacity units (3 assumed)	USD 34	USD 1.12
S3 (9 buckets, 50 GB each assumed)	USD 10	USD 0.34
CloudWatch Logs (5 GB per container assumed)	USD 122	USD 4.00
Secrets Manager	USD 7	USD 0.23
Tool estimate, list prices	**USD 1,146**	**USD 37.64**

Monthly lines are rounded, so they add up to about a dollar less than the total. The CloudWatch Logs line assumes heavy logging; with the light logging of an idle install the total is about USD 34 a day. Not included above:

Extra	Typical cost
Amazon Bedrock model use	Per token; usually the largest variable cost, depends on how much you use TrustHouse
Container Insights (on by default; it powers the "no running tasks" alarm)	A few dollars a day, depending on the number of services and tasks
Public IPv4 addresses	About USD 0.40 to 0.60 a day
AgentOps agents	About USD 1.18 a day per running agent (1 vCPU, 2 GB), stopped 8 hours after its last deploy; CodeBuild a few cents per build; ECR basic scanning free; Amazon Inspector per image if chosen
Autoscaling under load	Up to 4 tasks per service, so busy periods can raise compute cost
Data transfer	Out to the internet, and cross-Region image pulls outside us-east-1
Agent Lineage (after Configure Agent Lineage)	CloudTrail trail (a second copy of management events is billed), Bedrock invocation logs in CloudWatch Logs and S3, Athena queries per TB scanned
Neo4j Aura	Agreed per customer

A high-availability layout (three Availability Zones, two tasks per service, larger databases) costs more. Before deploying, Arhasi runs the same estimate for your exact choices and shares it with you.

15. Upgrades to new versions

Arhasi publishes each new version in this AWS Marketplace listing with release notes that label it Critical, Important or Optional. Arhasi deploys it for you through the deployment role; you do not pull images or run commands.

1. Arhasi tells you the version, what changes and when. You confirm the deployment role stack still exists.
2. Take a manual DocumentDB snapshot (section 11), or ask Arhasi to.
3. Arhasi deploys the new version's images. Services are replaced as a rolling update (new tasks normally start before old ones stop), so TrustHouse stays available, with possible brief interruptions. Your data is kept: DocumentDB, Valkey, S3, Secrets Manager and Neo4j are not replaced.
4. Arhasi checks health (section 12) and that every service runs the new version's Marketplace image.
5. If the release notes say the TrustHouseAccess template changed, download it again from **Integrations**, **AWS role** and update your stack with **Replace existing template**, keeping your parameters.

If an upgrade must be undone, Arhasi redeploys the previous version's images; the snapshot from step 2 is the fallback for data. Fixes for the known issues in section 17 are announced in the release notes of the version that contains them.

16. Removing access, or removing TrustHouse

- **Stop TrustHouse reading your AWS data:** delete the `TrustHouseAccess-<suffix>` stack.
- **Stop Arhasi reaching your account:** delete the `arhasi-deployer-access` stack (section 5). TrustHouse keeps running.
- **Remove TrustHouse:** ask Arhasi. Arhasi removes the stacks in the right order. Data is protected on purpose: S3 buckets are kept, DocumentDB leaves a final snapshot, and log groups may remain. Arhasi lists what is left so you can delete it once you no longer need it. Then cancel your subscription in AWS Marketplace and, if you wish, delete `CDKToolkit` and the Neo4j Aura database.

17. Support and known issues

Support: contactus@trusthouse.ai, web trusthouse.ai. Support covers deployment assistance, configuration guidance and issue resolution for TrustHouse in your account. Incident response by tier: business hours (Essential), priority (Production), 24x7 (Autonomous). When you write, include your environment name, Region, the time, and the message or screenshot. Never send passwords, tokens, keys or your External ID.

Known issues in 2.0.0

1. Configure Agent Lineage replaces existing Bedrock model invocation logging in that Region and can add management events to an existing CloudTrail trail, which adds CloudTrail cost. If your account already has these set up, contact support before running it.
2. If the IAM role `service-role/model-invocation-role` already exists, set `CreateInvocationLoggingRole=false` in the TrustHouseAccess template, or the stack fails.
3. When the TrustHouseAccess role may not read a bucket, the connection shows "Error connecting to the database". Add the bucket to `S3BucketNames` (or the Athena bucket fields) and update the stack.
4. Updating a bucket list in the TrustHouseAccess stack replaces the previous list, so repeat every bucket you still need.
5. In some AWS connection forms the AWS Region list can open outside the visible area. Contact support if you cannot select a Region.

Appendix A: Container images in this version

This version's 48 images are listed on its AWS Marketplace launch page. All are in the AWS Marketplace registry `709825985650.dkr.ecr.us-east-1.amazonaws.com`, tag `trusthouse-v2.0.2-mp2`.

Every image is Linux (amd64), runs as a non-root user, and was scanned by Arhasi for vulnerabilities and secrets before publishing. Each ECS task execution role may pull only its own repository (`ecr:BatchGetImage`, `ecr:GetDownloadUrlForLayer`, `ecr:BatchCheckLayerAvailability`), plus `ecr:GetAuthorizationToken`, which AWS offers only on all resources. Your subscription is what allows your account to pull; a pull refused with a 403 means the subscription is missing or not yet active.

Appendix B: What Arhasi runs

Arhasi's deployment tool runs from Arhasi's side, assumes your ArhasiDeployer role with your External ID, and deploys with AWS CDK and CloudFormation, using only this version's images. For each install it runs these steps:

Step	What it does
Pre-check	Read-only: proves access through your role, and checks the configuration, the images and the Bedrock models
Estimate	Read-only: the cost estimate in section 14
Provision	Creates the network, storage, databases and all services. About 1 to 2 hours
Set secrets	Writes each secret value straight into Secrets Manager in your account; values are never stored in files or typed on a command line
Turn on AgentOps	A second, short step that starts the AgentOps services once the private subnets exist
Health check	Read-only: checks every service is running and answering

No service image is built during your deploy: every container comes from the Marketplace registry.