

AI Search Installation Guide

Introduction

This page covers deploying AI Search on a Docker host using the production Docker Compose stack. It is intended for infrastructure engineers and system administrators responsible for deploying containerized services. For Aras-hosted Innovator SaaS instances, see Installation of AI Search for SaaS instead.

What you are deploying

AI Search runs as three long-running services (`ais-api`, `ais-indexer-worker`, `ais-parser-worker`) plus two one-shot init containers (`ais-db-migrations`, `parser-staging-init`) that run once at startup and then exit. These connect to external services you must provision before installation:

```
AIS containers → Aras Innovator    (Server + OAuth + Vault)
                → Temporal        (workflow orchestration, port 7233)
                → SQL Server      (inside the existing Innovator database, port 1433)
                → RabbitMQ / Azure Service Bus (CDC events via Edge API, AMQP 1.0)
                → Embeddings provider (OpenAI-compatible API, port 443)
```

None of these services are bundled with this stack; all must be provisioned before installation. The prerequisites section below documents what must be in place.

Prepare the deployment files

This guide is the source of the deployment files. There is no separate archive or repository. You create a working directory on the Docker host and populate it by copying the file contents shown in this document.

Target layout:



```
ai-search/  
├─ docker-compose.yaml # Copy from the "docker-compose.yaml" section below  
├─ .env # Copy from the ".env" section below, then fill in your values (See  
"Configure the environment file" below)  
└─ certs/ # Directory for the OAuth certificate (See "Place the certificate"  
below)
```

Steps:

1. Create the working directory:

```
mkdir -p ai-search/certs  
cd ai-search
```

2. Create a file named `docker-compose.yaml` in this directory and paste the contents from the `docker-compose.yaml` section below.
3. Create a file named `.env` in this directory and paste the contents from the `.env` section below, then fill in your values as described in *Configure the environment file* section on this page.
4. Leave `certs` / empty. You will place the OAuth certificate there as per the *Place the certificate* section.

Run all subsequent commands from the `ai-search/` directory.

docker-compose.yaml



```

# Production AI Search stack – standalone, image-based.
# All services use published Docker Hub images; no source builds.
#
# Prerequisites:
# - Create .env.example to .env next to this file (See the .env section below) and fill in all
  required values.
# - Place the OAuth certificate file in the certificate directory (default: ./certs/).
#   The file must be named <INNOVATOR_OAUTH_CLIENT_ID>.pfx (e.g. IndexerWorker.pfx).
#   Override the directory with CERTS_HOST_DIR in .env if certs live elsewhere.
# - Innovator Server and its SQL Server are reachable from the host via the URLs in .env.
# - External RabbitMQ (from the Edge API/CDC stack) is reachable and AMQP_CONNECTION_STRING is
  set in .env.
#   The broker must have the rabbitmq_amqp1_0 plugin enabled (AI Search uses AMQP 1.0).
# - A Temporal frontend endpoint must be reachable at TEMPORAL_HOST (see .env section).
#   This file contains only AIS services – Temporal itself is an external dependency.
# - Vault Server must be reachable from the indexer worker containers (see .env section).
#   Ensure the vault hostname is resolvable from Docker containers via extra_hosts, DNS, or
#   host.docker.internal as appropriate for your deployment.
#
# IMAGE_TAG in .env controls which version of all AI Search images is deployed.
#
# Minimum recommended hardware:
#   CPU: 4 vCPUs (ais-parser-worker uses all available cores for ML inference)
#   RAM: 8 GB (parser-worker loads ~4 GB of ML models; remainder for API, indexer, and
  Temporal)
#   Disk: 20 GB for the parser-staging volume (transient workspace during file indexing;
  actual size depends on the number and size of files indexed concurrently)

x-logging: &default-logging
  driver: json-file
  options:
    max-size: "50m"
    max-file: "5"

name: aras-ai-search

services:

  # -----
  # One-shot init containers
  # -----
  ais-db-migrations:
    image: ${IMAGE_REGISTRY:?IMAGE_REGISTRY is required}/db-migrations:${IMAGE_TAG:?IMAGE_TAG is
  required}
    restart: "no"
    environment:
      ASPNETCORE_ENVIRONMENT: Production
      AisDb__SchemaInstallConnectionString: "${AIS_DB_SCHEMA_CONNECTION_STRING:?
  AIS_DB_SCHEMA_CONNECTION_STRING is required}"
      AisDb__MigrationCommandTimeoutSeconds: "${AIS_DB_MIGRATION_COMMAND_TIMEOUT_SECONDS:-}"
    extra_hosts:
      - "${DOCKER_HOST_GATEWAY:-host.docker.internal}:host-gateway"
    networks:
      - ais-network

  parser-staging-init:

```



```

image: busybox:1.36.1
# setgid 2770 + GID 5000 owner: new files inherit group ownership; workers get GID 5000 via
group_add.
command: ["sh", "-c", "mkdir -p /var/lib/aras/parser-staging && chown root:5000
/var/lib/aras/parser-staging && chmod 2770 /var/lib/aras/parser-staging"]
volumes:
- parser_staging:/var/lib/aras/parser-staging
restart: "no"
networks:
- ais-network

# -----
# AI Search API (SPA host + JSON API)
# -----
ais-api:
image: ${IMAGE_REGISTRY:?IMAGE_REGISTRY is required}/api:${IMAGE_TAG:?IMAGE_TAG is required}
depends_on:
ais-db-migrations:
condition: service_completed_successfully
environment:
ASPNETCORE_ENVIRONMENT: Production
ASPNETCORE_URLS: http://+:8080
Innovator__InstanceId: "${INNOVATOR_INSTANCE_ID:?INNOVATOR_INSTANCE_ID is required}"
Innovator__InnovatorServerUrl: "${INNOVATOR_SERVER_URL:?INNOVATOR_SERVER_URL is required}"
AisDb__DataConnectionString: "${AIS_DB_DATA_CONNECTION_STRING:?
AIS_DB_DATA_CONNECTION_STRING is required}"
AisDb__SearchConnectionString: "${AIS_DB_SEARCH_CONNECTION_STRING:?
AIS_DB_SEARCH_CONNECTION_STRING is required}"
Temporal__TargetHost: "${TEMPORAL_HOST:?TEMPORAL_HOST is required}"
Temporal__Namespace: "${TEMPORAL_NAMESPACE:-default}"
Temporal__Credentials__ApiKey: "${TEMPORAL_API_KEY:-}"
Amqp__ConnectionString: "${AMQP_CONNECTION_STRING:?AMQP_CONNECTION_STRING is required}"
Cdc__ConfiguratorUrl: "${CDC_CONFIGURATOR_URL:?CDC_CONFIGURATOR_URL is required}"
Cdc__Origin: "${CDC_ORIGIN:-aisearch}"
Cdc__InstanceName: "${CDC_INSTANCE_NAME:-default}"
Cdc__ConfiguratorUserName: "${CDC_CONFIGURATOR_USER:-admin}"
Cdc__ConfiguratorAttemptTimeout: "${CDC_CONFIGURATOR_ATTEMPT_TIMEOUT:-00:02:00}"
Cdc__ConfiguratorTotalTimeout: "${CDC_CONFIGURATOR_TOTAL_TIMEOUT:-00:05:00}"
Embeddings__Endpoint: "${EMBEDDINGS_ENDPOINT:?EMBEDDINGS_ENDPOINT is required}"
Embeddings__ApiKey: "${EMBEDDINGS_API_KEY:?EMBEDDINGS_API_KEY is required}"
Embeddings__ModelId: "${EMBEDDINGS_MODEL_ID:-text-embedding-3-small}"
VectorSearchSettings__VectorDimension: "${VECTOR_DIMENSION:-1536}"
ports:
# Bound to localhost – place a TLS-terminating reverse proxy (nginx, Traefik, etc.) in
front for external access.
# Health endpoints (no auth required):
# GET /health/live – liveness (instant, no external checks); suitable for K8s liveness
probe
# GET /health/ready – readiness (checks SQL, Temporal, CDC, Embeddings); suitable for
load-balancer / Traefik
# GET /health/deep – deep checks excluded from /health/ready (e.g. temporal-worker
task-queue polling);
# for diagnostics, not load-balancer/k8s probes – a transient
failure here should not pull the pod out of rotation
- "127.0.0.1:5080:8080"
restart: unless-stopped
extra_hosts:

```



```

- "${DOCKER_HOST_GATEWAY:-host.docker.internal}:host-gateway"
logging: *default-logging
networks:
- ais-network

# -----
# AI Search Temporal Worker (.NET – indexing workflows + incremental consumer)
# -----
ais-indexer-worker:
  image: ${IMAGE_REGISTRY:?IMAGE_REGISTRY is required}/indexer-worker:${IMAGE_TAG:?IMAGE_TAG
is required}
  depends_on:
    ais-db-migrations:
      condition: service_completed_successfully
    parser-staging-init:
      condition: service_completed_successfully
  environment:
    ASPNETCORE_ENVIRONMENT: Production
    Innovator__InstanceId: "${INNOVATOR_INSTANCE_ID:?INNOVATOR_INSTANCE_ID is required}"
    Innovator__InnovatorServerUrl: "${INNOVATOR_SERVER_URL:?INNOVATOR_SERVER_URL is required}"
    Innovator__OAuthServerUrl: "${INNOVATOR_OAUTH_URL:?INNOVATOR_OAUTH_URL is required}"
    Innovator__ClientId: "${INNOVATOR_OAUTH_CLIENT_ID:-IndexerWorker}"
    Innovator__CertificatePath: "/Container/Certificates/${INNOVATOR_OAUTH_CLIENT_ID:-
IndexerWorker}.pfx"
    Innovator__CertificatePassword: "${ARAS_OAUTH_CERTIFICATE_PASSWORD}"
    Innovator__ServiceUserName: "${INNOVATOR_SERVICE_USER:-vadmin}"
    AisDb__DataConnectionString: "${AIS_DB_DATA_CONNECTION_STRING:?
AIS_DB_DATA_CONNECTION_STRING is required}"
    Temporal__TargetHost: "${TEMPORAL_HOST:?TEMPORAL_HOST is required}"
    Temporal__Namespace: "${TEMPORAL_NAMESPACE:-default}"
    Temporal__Credentials__ApiKey: "${TEMPORAL_API_KEY:-}"
    Amqp__ConnectionString: "${AMQP_CONNECTION_STRING:?AMQP_CONNECTION_STRING is required}"
    IncrementalConsumer__StartupDelaySeconds: "5"
    IncrementalConsumer__HeartbeatSeconds: "60"
    IncrementalConsumer__MaxProcessingAttempts: "5"
    IncrementalConsumer__RetryBackoffSeconds: "2"
    ParserStaging__Root: "/var/lib/aras/parser-staging"
    ParserStaging__MaxFileBytes: "536870912"
    ParserStaging__TransferBufferBytes: "16777216"
    Embeddings__Endpoint: "${EMBEDDINGS_ENDPOINT:?EMBEDDINGS_ENDPOINT is required}"
    Embeddings__ApiKey: "${EMBEDDINGS_API_KEY:?EMBEDDINGS_API_KEY is required}"
    Embeddings__ModelId: "${EMBEDDINGS_MODEL_ID:-text-embedding-3-small}"
    VectorSearchSettings__VectorDimension: "${VECTOR_DIMENSION:-1536}"
    FileParsing__MaxTokens: "${FILE_PARSING_MAX_TOKENS:-1500}"
    FileParsing__MergePeers: "${FILE_PARSING_MERGE_PEERS:-true}"
    FileParsing__DoOcr: "${FILE_PARSING_DO_OCR:-false}"
    FileParsing__TableMode: "${FILE_PARSING_TABLE_MODE:-accurate}"
    FileParsing__DoTableStructure: "${FILE_PARSING_DO_TABLE_STRUCTURE:-true}"
    Temporal__ActivityPolicies__CheckFileProcessedStartToCloseSeconds: "10"
    Temporal__ActivityPolicies__DownloadFileStartToCloseSeconds: "300"
    Temporal__ActivityPolicies__DownloadFileHeartbeatSeconds: "120"
    Temporal__ActivityPolicies__ParseAndStageChunksStartToCloseSeconds: "900"
    Temporal__ActivityPolicies__ParseAndStageChunksHeartbeatSeconds: "120"
    Temporal__ActivityPolicies__PersistStagedChunksStartToCloseSeconds: "120"
    Temporal__ActivityPolicies__EmbedFileChunksStartToCloseSeconds: "300"
    Temporal__ActivityPolicies__EmbedFileChunksHeartbeatSeconds: "120"
    Temporal__ActivityPolicies__CleanupStagedFileStartToCloseSeconds: "30"

```



```

    Cdc__ConfiguratorUrl: "${CDC_CONFIGURATOR_URL:?CDC_CONFIGURATOR_URL is required}"
    Cdc__InstanceName: "${CDC_INSTANCE_NAME:-default}"
volumes:
  - "${CERTS_HOST_DIR:-./certs}/Container/Certificates:ro"
  - parser_staging:/var/lib/aras/parser-staging:rw
# ais-staging (GID 5000): write access to the parser-staging shared volume.
group_add:
  - "5000"
restart: unless-stopped
stop_grace_period: 60s
extra_hosts:
  - "${DOCKER_HOST_GATEWAY:-host.docker.internal}:host-gateway"
logging: *default-logging
networks:
  - ais-network

# -----
# File Parser Worker (Python – docling-based file parsing)
# -----
ais-parser-worker:
  image: ${IMAGE_REGISTRY:?IMAGE_REGISTRY is required}/parser-worker:${IMAGE_TAG:?IMAGE_TAG is required}
  depends_on:
    ais-db-migrations:
      condition: service_completed_successfully
    parser-staging-init:
      condition: service_completed_successfully
  environment:
    PARSER_WORKER__TEMPORAL__TARGET_HOST: "${TEMPORAL_HOST:?TEMPORAL_HOST is required}"
    PARSER_WORKER__TEMPORAL__NAMESPACE: "${TEMPORAL_NAMESPACE:-default}"
    PARSER_WORKER__TEMPORAL__INNOVATOR_INSTANCE_ID: "${INNOVATOR_INSTANCE_ID:?INNOVATOR_INSTANCE_ID is required}"
    PARSER_WORKER__TEMPORAL__API_KEY: "${TEMPORAL_API_KEY:-}"
    PARSER_WORKER__STAGING_ROOT: "/var/lib/aras/parser-staging"
    PARSER_WORKER__MAX_CONCURRENT_ACTIVITIES: "${PARSER_WORKER__MAX_CONCURRENT_ACTIVITIES:-1}"
  volumes:
    - parser_staging:/var/lib/aras/parser-staging:rw
# ais-staging (GID 5000): write access to the parser-staging shared volume.
group_add:
  - "5000"
restart: unless-stopped
stop_grace_period: 60s
extra_hosts:
  - "${DOCKER_HOST_GATEWAY:-host.docker.internal}:host-gateway"
logging: *default-logging
networks:
  - ais-network

volumes:
  parser_staging:

networks:
  ais-network:
    name: ais-network
    driver: bridge

```



.env



```

# AI Search – Production stack environment variables
# Used by: docker-compose.yaml
# Copy this file to .env in the same directory and fill in all required values.
# Run: docker compose up -d
#
# Required values are marked with (required).
# Optional values with defaults show the default in the comment.

# — Image version —————

# Tag of the AI Search Docker images to deploy. (required)
# Use the CI build tag from the Actions run that was validated (e.g. 0.1.0-build.1047).
IMAGE_TAG=

# Registry + repository prefix for the AI Search images. (required)
# Resolved as: <IMAGE_REGISTRY>/<component>:<IMAGE_TAG>
#
# AI Search images are published publicly to Docker Hub under the arascorporation organisation.
# No login is required to pull them.
#
# Example:
#   IMAGE_REGISTRY=arascorporation
IMAGE_REGISTRY=

# — Temporal —————

# Temporal frontend gRPC address. (required)
# Examples:
#   temporal:7233                                ← local self-hosted Temporal on the same Docker host
#   <namespace>.tmprl.cloud:7233                 ← Temporal Cloud
#   temporal.company.com:7233                   ← self-hosted managed by another team
TEMPORAL_HOST=

# Temporal namespace. Default: default
# For Temporal Cloud this is typically your account-scoped namespace (e.g. myorg.acme).
# TEMPORAL_NAMESPACE=default

# Temporal API key for authentication. Optional.
# Required for Temporal Cloud; leave empty for a self-hosted Temporal cluster
# that does not require authentication.
# TEMPORAL_API_KEY=

# — Innovator —————

# Unique identifier for this Innovator instance.
# Used as the Temporal task-queue suffix (ai-search-index-<INNOVATOR_INSTANCE_ID>).
# Must match ^[A-Za-z0-9_-]{1,64}$. (required)
INNOVATOR_INSTANCE_ID=

# Base URL of the Innovator Server – must be reachable from Docker.
# For Innovator running on the same Docker host, use host.docker.internal (Mac/Win)
# or set DOCKER_HOST_GATEWAY below (Linux). (required)
# Example (external server): https://your-innovator.example.com/
# Example (same Docker host): http://host.docker.internal:18081/
INNOVATOR_SERVER_URL=

```



```

# Base URL of the Innovator OAuth server. (required)
# Example: https://your-innovator.example.com/OAuthServer/
# Example (same Docker host): http://host.docker.internal:18082/
INNOVATOR_OAUTH_URL=

# On Linux Docker, host.docker.internal is not set automatically.
# Uncomment and set to enable containers to reach services on the Docker host.
# Default: host.docker.internal
# DOCKER_HOST_GATEWAY=host.docker.internal

# OAuth client ID registered in Innovator for the indexer worker.
# Default: IndexerWorker
INNOVATOR_OAUTH_CLIENT_ID=IndexerWorker

# Host path to the directory containing OAuth client certificate(s).
# The directory must contain a file named <INNOVATOR_OAUTH_CLIENT_ID>.pfx.
# Example: CERTS_HOST_DIR=/etc/aras/certs → expects /etc/aras/certs/IndexerWorker.pfx
# Default: ./certs (a certs/ sub-directory next to this file)
# CERTS_HOST_DIR=./certs

# Password for the OAuth PFX certificate file.
# Leave empty if the PFX is not password-protected.
ARAS_OAUTH_CERTIFICATE_PASSWORD=

# Service account used by the indexer worker to perform Innovator operations.
# Default: vadmin
INNOVATOR_SERVICE_USER=vadmin

# — Database —————

# SQL Server connection string for data read/write operations.
# Used by ais-api and ais-indexer-worker for general operations. (required)
#
# Copy one of the examples below and substitute your database name, user, and password:
# External SQL Server:
# Server=tcp:sql.company.com,1433;Database=innovator;User
# Id=aisearch;Password=yourpassword;Encrypt=True;TrustServerCertificate=True;
# SQL Server on the Docker host (Mac / Windows Docker Desktop / Linux):
# Server=tcp:host.docker.internal,1433;Database=innovator;User
# Id=aisearch;Password=yourpassword;Encrypt=True;TrustServerCertificate=True;
# (Linux only: also set DOCKER_HOST_GATEWAY=host.docker.internal above)
AIS_DB_DATA_CONNECTION_STRING=

# SQL Server connection string for vector/full-text search queries.
# Can point to a read-only replica for scaling. (required)
AIS_DB_SEARCH_CONNECTION_STRING=

# SQL Server connection string for schema migrations.
# Needs CREATE TABLE / ALTER TABLE permissions. (required)
# In simple deployments all three strings can be set to the same value.

AIS_DB_SCHEMA_CONNECTION_STRING=

# Command timeout in seconds for schema migration steps. Optional.
# The DiskANN vector index migration may time out on instances with large existing datasets.
# Increase if ais-db-migrations exits with a timeout error during upgrade.
# Default: 3600

```



```
# AIS_DB_MIGRATION_COMMAND_TIMEOUT_SECONDS=3600

# — Message bus (AMQP) —————

# AMQP 1.0 connection string to the RabbitMQ broker provided by the Edge API/CDC stack.
# (required)
# This must point at the same broker that the Edge API exporter publishes CDC events to.
# Note: AI Search uses the AMQP 1.0 protocol (AmqpNetLite). The RabbitMQ broker must have
#       the rabbitmq_amqp1_0 plugin enabled.
#
# Exchange: the Edge API exporter creates and publishes to exchange "ams.exchange"
# automatically.
#           AI Search does not need to create or configure this exchange.
# Queues:   AI Search auto-provisions its own durable queues and bindings on the broker when
#           CDC configuration is applied (via CDC_CONFIGURATOR_URL below). No manual queue
#           setup is needed on the RabbitMQ side.
#
# Examples:
#   amqps://user:password@rabbitmq.example.com:5671/vhost      (TLS – recommended for production)
#   amqp://user:password@rabbitmq.example.com:5672/vhost      (plain – development only,
#   credentials sent in cleartext)
AMQP_CONNECTION_STRING=

# — CDC (Change Data Capture) —————

# URL of the EdgeAPI CDC configurator service. (required)
# The configurator provisions AI Search queues and bindings on the RabbitMQ broker above,
# and enables SQL Server CDC tracking for the configured ItemTypes.
# This service is part of the Edge API/CDC stack – it must be reachable from Docker.
# Example: http://edgeapi-configurator-service:8080/
CDC_CONFIGURATOR_URL=

# CDC origin identifier. Default: aisearch
CDC_ORIGIN=aisearch

# CDC instance name identifying this Innovator deployment. Default: default
CDC_INSTANCE_NAME=default

# Innovator user the indexer worker impersonates when calling the configurator during an apply.
# The configurator authorises by matching the token's username claim against its own
# Configurator__Innovator__Auth__Username__Value, so this must equal that value (default:
# admin).
# Deliberately separate from INNOVATOR_SERVICE_USER, which is the vault identity and needs the
# FileAdministrators identity instead.
CDC_CONFIGURATOR_USER=admin

# Resilience timeouts for calls the indexer worker makes to the CDC configurator during an
# apply.
# Format: hh:mm:ss. Optional – increase on slow networks or a heavily loaded configurator.
# Defaults: 00:02:00 (per attempt) / 00:05:00 (total, must be greater than the per-attempt
# value).
# CDC_CONFIGURATOR_ATTEMPT_TIMEOUT=00:02:00
# CDC_CONFIGURATOR_TOTAL_TIMEOUT=00:05:00

# — Embeddings provider —————

# Full base URL of the embeddings API endpoint. (required)
```



```
# Example: https://your-openai.openai.azure.com/openai/v1
EMBEDDINGS_ENDPOINT=

# API key for the embeddings provider. (required)
EMBEDDINGS_API_KEY=

# Model identifier used for generating embeddings.
# Default: text-embedding-3-small
EMBEDDINGS_MODEL_ID=text-embedding-3-small

# Vector dimension – must match the model above.
# Currently only 1536 is supported. Default: 1536
VECTOR_DIMENSION=1536

# — File parsing —————

# Maximum tokens per parsed chunk. Default: 1500
FILE_PARSING_MAX_TOKENS=1500

# Merge peer chunks during parsing. Default: true
FILE_PARSING_MERGE_PEERS=true

# Enable OCR during parsing (slow; enable for scanned PDFs). Default: false
FILE_PARSING_DO_OCR=false

# Table detection mode: fast or accurate. Default: accurate
FILE_PARSING_TABLE_MODE=accurate

# Enable table structure recognition. Default: true
FILE_PARSING_DO_TABLE_STRUCTURE=true

# — Parser worker —————

# Number of parallel file-parsing activities the parser worker runs simultaneously.
# Increase on hosts with more CPU cores and memory to speed up initial indexing. Default: 1
# PARSER_WORKER__MAX_CONCURRENT_ACTIVITIES=1
```

Prerequisites

Before starting, verify that every item below is in place on the target host.

Docker

Component	Minimum version
Docker Engine	24.0
Docker Compose plugin	2.20

Check:



```
docker version          # Engine version
docker compose version  # Compose plugin version
```

Operating system

- **Linux (recommended for production):** any distribution that supports Docker Engine 24 or later (Ubuntu 22.04 LTS, RHEL 9, Debian 12, or later)
- **Windows 10 / 11:** Docker Desktop 4.25 or later with WSL 2 backend (test or pilot only)

Important

Shell commands in this guide assume a bash-like shell. On Windows, run them from WSL 2, Git Bash, or a Linux jump host; native PowerShell equivalents differ (e.g. `cp` → `Copy-Item`).

Aras Innovator

AI Search requires a running Innovator instance that is reachable from the Docker host:

Requirement	Details
Innovator version	14.0.15 or later
HTTP access	Innovator Server URL reachable from the Docker host
OAuth server	OAuth Server URL reachable from the Docker host
Vault access	Vault Server HTTP endpoint reachable from the AIS containers (required for file content indexing)
Service account	A dedicated service account (default: <code>vadmin</code>) with access to all ItemTypes that will be indexed

Important

Vault on the same Docker host: on Linux, `host.docker.internal` is not resolved automatically. Set `DOCKER_HOST_GATEWAY=host.docker.internal` in `.env`; the compose file injects it as `extra_hosts` so containers can reach the host. On Mac and Windows Docker Desktop this is automatic.

Important

Vault URL resolution: the vault hostname is obtained from Innovator's internal vault configuration at indexing time and is **not** set in AIS `.env`. Ensure the hostname returned by Innovator is resolvable from Docker containers via DNS, `extra_hosts`, or



host.docker.internal as appropriate for your deployment.

Temporal

AI Search uses **Temporal** for workflow orchestration. A Temporal cluster must be running and reachable before deploying AIS. Temporal is **not** bundled with this stack.

Requirement	Details
Temporal version	1.24 or later
Frontend address	<host>:7233 reachable from the Docker host
Namespace	A dedicated namespace is recommended (default: default)
Authentication	Temporal Cloud requires an API key —set it as TEMPORAL_API_KEY in .env. Self-hosted clusters without authentication can leave it empty.

Options:

- **Temporal Cloud:** use your cloud namespace frontend address (<namespace>.tmprl.cloud:7233)
- **Self-hosted:** see the [Temporal self-hosting guide](#)

SQL Server

AI Search installs the [ais] schema and its tables inside the **Innovator database**; no separate database is created. The same SQL Server instance that hosts Innovator is used:

Requirement	Details
SQL Server version	SQL Server 2025 or later, required for the native vector type used by AI Search. Azure SQL Database with vector support is also supported. If you are on SQL Server 2019 or 2022, contact Aras for the compatibility upgrade path.
Connectivity	TCP port 1433 reachable from the Docker host
Permissions	See "Grant SQL Server permissions" below



Edge API / CDC stack

AI Search receives real-time change events from Innovator via the **Aras InnovatorEdge CDC pipeline**, a separate Aras product that is **a prerequisite** for AI Search and must be installed and running **before** deploying AIS.

For installation instructions, see the [Aras InnovatorEdge API on-premise installation guide](#) .

The following components are part of that stack:

Component	Role
edgeapi-eventing-configurator	Registers ItemTypes for SQL Server CDC tracking and provisions message broker queues. Must expose an HTTP endpoint reachable from the AIS containers (CDC_CONFIGURATOR_URL).
edgeapi-eventing-exporter	Streams CDC change events from SQL Server to the message broker. Must connect to the same SQL Server instance as Innovator.
Message broker	Either RabbitMQ or Azure Service Bus. AI Search uses the AMQP 1.0 protocol. For RabbitMQ, the rabbitmq_amqp1_0 plugin must be enabled.

Verify the exporter is connected and polling before starting AIS:

```
docker logs <edgeapi-exporter-container> --tail 5
# Expected: CDC reader mode: Normal, synchronized tables: N, pending: 0
```

If using RabbitMQ, verify the `rabbitmq_amqp1_0` plugin is enabled:

```
docker exec <rabbitmq-container> rabbitmq-plugins list | grep amqp1_0
# Expected: [E*] rabbitmq_amqp1_0
```

Important

Be aware that eventing.* tables are created by AI Search activity, and CDC will report errors until it has run.

Network access summary



The following outbound connections must be open from the Docker host:

Destination	Port	Purpose
Innovator Server	80 / 443	Item data, OAuth token requests
OAuth Server	80 / 443	OAuth token requests
Vault Server	80 / 443	File content download for indexing
Temporal frontend	7233 (TCP)	Workflow orchestration
SQL Server	1433 (TCP)	AI Search database
RabbitMQ (AMQP)	5671 (TLS) or 5672 (plain)	CDC event stream
Azure Service Bus	5671 (AMQP TLS) or 443 (AMQP over WebSocket)	CDC event stream (alternative to RabbitMQ)
Edge API configurator	varies	Queue provisioning
Embeddings API (OpenAI-compatible)	443	Vector generation
Docker Hub	443	Pulling Docker images

Minimum hardware

Resource	Minimum	Notes
CPU	4 vCPUs	<code>ais-parser-worker</code> uses all available cores for ML inference
RAM	8 GB	Parser worker loads approximately 4 GB of ML models; remainder for API, indexer, and Temporal
Disk	20 GB	For the parser-staging volume (transient workspace during file indexing; actual size depends on the number and size of files indexed concurrently)

Pre-installation

Generate the certificate



AI Search authenticates to Innovator using an X.509 certificate (OAuth 2.0 impersonate grant). You need two files:

- **.pfx**: private key and certificate, mounted into **ais-indexer-worker**
- **.cer**: public key only, registered in Innovator OAuth

Run the following (requires OpenSSL 3.0 or later):

```
# Generate a 2048-bit RSA key and self-signed certificate valid for 5 years
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem \
  -days 1825 -noenc -subj "/CN=IndexerWorker"

# Extract public key – needed for Innovator registration (see "Configure Innovator OAuth" below)
openssl x509 -in cert.pem -out IndexerWorker.cer

# Bundle into .pfx without a password (leave ARAS_OAUTH_CERTIFICATE_PASSWORD empty in .env):
openssl pkcs12 -export -out IndexerWorker.pfx -inkey key.pem -in cert.pem -passout pass:

# Or with a password (set ARAS_OAUTH_CERTIFICATE_PASSWORD=<password> in .env):
openssl pkcs12 -export -out IndexerWorker.pfx -inkey key.pem -in cert.pem -passout pass:
<password>

# Linux / macOS
rm key.pem cert.pem

# Windows (PowerShell)
Remove-Item key.pem, cert.pem
```

If the **IndexerWorker** client **already exists** in Innovator, you don't need to generate a new certificate. The existing **IndexerWorker.pfx** may be located in the Innovator OAuth certificate directory. Copy it from there:

```
<InnovatorRoot>/OAuthServer/App_Data/Certificates/IndexerWorker.pfx
```

If you only have the **.pfx**, extract the public **.cer** from it:

```
openssl pkcs12 -in IndexerWorker.pfx -nokeys -clcerts -out IndexerWorker.cer -passin pass:
```

Configure Innovator OAuth

If the **IndexerWorker** OAuth client does not yet exist in your Innovator installation, register it now.

1. Copy Certificate to Innovator



Copy `IndexerWorker.cer` to the Innovator OAuth certificate directory:

```
<InnovatorRoot>/OAuthServer/App_Data/Certificates/IndexerWorker.cer
```

2. Update oauth.config

Edit `<InnovatorRoot>/OAuthServer/oauth.config` and add the following block inside `<server><clientRegistries>` :

```
<clientRegistry id="IndexerWorker" enabled="true">
  <secrets>
    <secret type="JwtBearerAssertionServerSecret">
      <certificate filePath="App_Data/Certificates/IndexerWorker.cer" />
    </secret>
  </secrets>
  <allowedScopes>
    <scope name="Innovator" />
  </allowedScopes>
  <allowedGrantTypes>
    <grantType name="impersonate" />
  </allowedGrantTypes>
  <tokenLifetime accessTokenLifetime="3600" />
</clientRegistry>
```

Configuration details:

- `id="IndexerWorker"` must match `INNOVATOR_OAUTH_CLIENT_ID` in `.env` (default: `IndexerWorker`)
- `filePath` is the relative path to the `.cer` file inside the OAuthServer directory

Restart the OAuth Server

Restart the OAuth server process to apply the new client registration.

Grant SQL Server permissions

Important

Managed deployments: if the three `AIS_DB_*` connection strings have already been provisioned by a DBA or platform administrator, skip this section; pass the provided strings to `.env` and proceed *Obtain the AMQP connection string* below.

AI Search migrations run inside the Innovator database and create the `[ais]` schema and its tables. No new database is created. All three connection strings should point to the existing Innovator



database.

The login used by AIS_DB_SCHEMA_CONNECTION_STRING must have:

Permission	Purpose
db_owner on the Innovator database	Create schema, tables, and manage role membership

The logins for AIS_DB_DATA_CONNECTION_STRING and AIS_DB_SEARCH_CONNECTION_STRING receive their minimum required database roles (ais_data_role, ais_search_role) automatically during migration. In simple deployments all three connection strings may reference the same login.

```
-- Create a dedicated login for AI Search (substitute your own name and password)
CREATE LOGIN aisearch WITH PASSWORD = '<strong-password>';

-- Grant db_owner on the Innovator database
USE [innovator]; -- substitute your actual Innovator database name
CREATE USER aisearch FOR LOGIN aisearch;
ALTER ROLE db_owner ADD MEMBER aisearch;
```

Important

db_owner is required only for the initial migration run. The migration creates the [ais] schema, custom roles (ais_schema_install_role, ais_data_role, ais_search_role), and assigns logins to them automatically. After the first successful run you can demote the schema login. Refer to the code block that follows.

```
USE [innovator];
ALTER ROLE ais_schema_install_role ADD MEMBER aisearch;
ALTER ROLE db_owner DROP MEMBER aisearch;
```

If a database's owner is a Windows domain account instead of a SQL login, and it can't be resolved (VPN/domain unreachable, account renamed, etc.), CDC fails to start. To fix it, reassign ownership to the sa login (or another valid SQL login) so that enabling CDC doesn't require an AD lookup.



```
-- Confirm the owner
SELECT SUSER_SNAME(owner_sid) AS db_owner FROM sys.databases WHERE name = 'InnovatorSolutions';

-- Reassign ownership to the sa login (or another valid SQL login) so CDC enable doesn't need AD
lookups
ALTER AUTHORIZATION ON DATABASE::innovator TO sa;
```

Obtain the AMQP connection string

AI Search uses the AMQP 1.0 protocol to consume CDC events from the message broker. Two broker options are supported.

RabbitMQ (on-premises, self-hosted):

```
# TLS (recommended for production, port 5671):
amqps://<user>:<password>@<rabbitmq-host>:5671/

# Plain (development / internal network only, port 5672):
amqp://<user>:<password>@<rabbitmq-host>:5672/
```

Obtain the credentials from the team that manages the Edge API / CDC stack.

Important

RabbitMQ must have the rabbitmq_amqp1_0 plugin enabled (see the Edge API prerequisites check above). Use amqps:// (TLS) in production.

Azure Service Bus (cloud-hosted):

```
# Standard Azure Service Bus connection string (from Azure Portal):
Endpoint=sb://<namespace>.servicebus.windows.net/;SharedAccessKeyName=<policy>;SharedAccessKey=
<key>

# AMQP URI form (port 5671 – TLS):
amqps://<policy>:<key-url-encoded>@<namespace>.servicebus.windows.net
```

Obtain the connection string from **Azure Portal** → **your Service Bus namespace** → **Shared access policies** → **your policy** → **Primary Connection String**.

Important

Azure Service Bus natively supports AMQP 1.0; no plugin configuration is required. Use the Standard or Premium tier; the Basic tier does not support topics.



Set the value as `AMQP_CONNECTION_STRING` in `.env`.

Obtain embeddings API credentials

AI Search generates vector embeddings via any OpenAI-compatible embeddings API; the application calls the standard `/v1/embeddings` endpoint.

You can use Azure OpenAI, OpenAI, or any compatible provider.

Variable	Description
EMBEDDINGS_ENDPOINT	Base URL of the embeddings API (e.g. <code>https://openai.azure.com/openai/v1/</code> for Azure OpenAI, <code>https://api.openai.com/...</code> for OpenAI)
EMBEDDINGS_API_KEY	API key for the embeddings provider
EMBEDDINGS_MODEL_ID	Deployment or model name. Use a model that outputs 1536-dimensional vectors (e.g. <code>text-embedding-3-small</code>).
VECTOR_DIMENSION	Output dimension of the model. Must be <code>1536</code> ; fixed by the database schema.

Important

Azure OpenAI: find the endpoint and key in Azure Portal → your Azure OpenAI resource → Keys and Endpoint. Use the deployment name you gave the model as `EMBEDDINGS_MODEL_ID`.

Installation

All commands in this section are run from the `ai-search/` directory:

```
cd ai-search/
```

Configure the Temporal connection

Set `TEMPORAL_HOST` in `.env` to your Temporal frontend address:



```
# Temporal Cloud:
TEMPORAL_HOST=<namespace>.tmprl.cloud:7233

# Self-hosted cluster:
TEMPORAL_HOST=temporal.company.com:7233
```

If your cluster uses a non-default namespace, set `TEMPORAL_NAMESPACE` in `.env`.
Temporal Cloud requires an API key for authentication. Set it in `.env`:

```
TEMPORAL_API_KEY=<your-temporal-cloud-api-key>
```

The key is passed to every AI Search service as `Temporal:Credentials:ApiKey` (the .NET services) and `PARSER_WORKER__TEMPORAL__API_KEY` (the parser worker). Leave `TEMPORAL_API_KEY` empty for a self-hosted Temporal cluster that does not require authentication.

Configure the environment file

Open `.env` in a text editor and fill in all required values:

Variable	Description
IMAGE_TAG	AI Search image version to deploy. Obtain the tag from the Aras release notification.
IMAGE_REGISTRY	Docker Hub organization prefix for AI Search images (e.g. arascorporation)
TEMPORAL_HOST	Temporal frontend address (e.g. <code>temporal:7233</code>)
INNOVATOR_INSTANCE_ID	Unique identifier for this Innovator instance. Allowed characters: A-Z, a-z, 0-9, -, _; maximum 64 characters. Example: innovator-prod
INNOVATOR_SERVER_URL	Innovator HTTP base URL
INNOVATOR_OAUTH_URL	OAuth Server base URL
AIS_DB_DATA_CONNECTION_STRING	SQL Server connection string for data operations
AIS_DB_SEARCH_CONNECTION_STRING	SQL Server connection string for search queries
AIS_DB_SCHEMA_CONNECTION_STRING	SQL Server connection string for schema migrations



AMQP_CONNECTION_STRING	AMQP 1.0 connection string: RabbitMQ (amqps://user:pass@host:5671/) or Azure Service Bus (Endpoint=sb://...)
CDC_CONFIGURATOR_URL	Edge API configurator service URL
EMBEDDINGS_ENDPOINT	Embeddings API base URL (OpenAI-compatible)
EMBEDDINGS_API_KEY	Embeddings API key
AIS_DB_MIGRATION_COMMAND_TIMEOUT_SECONDS	(Optional) Command timeout in seconds for schema migration steps. Set to 3600 or higher if ais-db-migrations exits with a timeout error during upgrade. Default: 3600 .

Optional variables with defaults (see .env template previously for the full list):

INNOVATOR_OAUTH_CLIENT_ID, INNOVATOR_SERVICE_USER, ARAS_OAUTH_CERTIFICATE_PASSWORD, CERTS_HOST_DIR, TEMPORAL_NAMESPACE, EMBEDDINGS_MODEL_ID, VECTOR_DIMENSION, CDC_CONFIGURATOR_USER, CDC_INSTANCE_NAME, CDC_ORIGIN, and others.

Important

Linux Docker hosts: set DOCKER_HOST_GATEWAY=host.docker.internal in .env. On Linux, host.docker.internal is not injected automatically; this variable tells the containers how to reach services running on the Docker host (Innovator, SQL Server, RabbitMQ).

Place the certificate

Copy the IndexerWorker.pfx file (see *Generate the certificate* above) into the certificate directory. The default location is ai-search/certs/IndexerWorker.pfx :

```
cp /path/to/IndexerWorker.pfx certs/IndexerWorker.pfx
```

If you use a different path, set CERTS_HOST_DIR in .env to that directory's absolute path.

Pull images and start the stack

AI Search images are published to Docker Hub; no Docker Hub login is required.



```
# Pull all AI Search images
docker compose pull

# Start the stack in the background
docker compose up -d
```

The `ais-db-migrations` init container runs automatically on first start and creates the AI Search database schema. It exits when complete; the main services start afterward.

Important

External access: `ais-api` binds to `127.0.0.1:5080` (loopback only). To expose it outside the host, place a reverse proxy (such as `nginx` or a cloud load balancer) in front of it.

Important

After a host reboot: all containers start automatically (restart: `unless-stopped`). During the first 15 to 30 seconds, transient connection errors may appear in logs while Temporal initializes. This is expected and resolves automatically.

Verification

Allow 30 to 60 seconds for all containers to reach a running state, then run the checks below.

Container status

```
docker compose ps
```

Expected: `ais-api`, `ais-indexer-worker`, and `ais-parser-worker` are **Up**; `ais-db-migrations` and `parser-staging-init` are **Exited (0)**.

Readiness health check

```
curl -s -o /dev/null -w "%{http_code}" http://localhost:5080/health/ready
```

Expected: **200**

Individual probe results (one line per subsystem):

```
curl -s http://localhost:5080/health/ready | jq . # or: python3 -m json.tool
```



You can also query each subsystem individually:

```
curl -s http://localhost:5080/health/ready/sql-data
curl -s http://localhost:5080/health/ready/sql-search
curl -s http://localhost:5080/health/ready/innovator
curl -s http://localhost:5080/health/ready/embeddings
curl -s http://localhost:5080/health/ready/temporal
curl -s http://localhost:5080/health/ready/cdc-configurator
```

Important

The subsystem names in these health endpoints correspond directly to the six service health indicator dots visible in the AI Search Configurator. See the **AI Search Configurator - Admin Guide** for reference.

Temporal workers

1. Open the Temporal UI (Temporal Cloud dashboard or your self-hosted Temporal UI).
2. Navigate to **Workers** for the namespace configured in `TEMPORAL_NAMESPACE` (default: `default`).
3. Confirm that workers are shown as polling the `ai-search-index-<INNOVATOR_INSTANCE_ID>task` queue.

Container logs

```
docker compose logs --tail 50 ais-api
docker compose logs --tail 50 ais-indexer-worker
```

A healthy startup log for `ais-api` includes:

```
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
```

First-time configuration

After the stack is running and healthy, install the AI Search component and configure which Innovator ItemTypes AI Search should index.

Install the AI Search component

The AI Search component (`AI Search` AML package) adds the AI Search button to the Innovator command bar and is required for the Search UI to be accessible.



1. Obtain the AISearch package from the Aras AI Search release archive.
2. Install it in Innovator using the **Aras Update** tool.
3. In Innovator, navigate to **Administration** → **Variables** → `ais_SearchUIUrl` and set its value to the URL through which users' browsers can reach the AI Search UI:
 1. With a reverse proxy (recommended for production): use the proxy's public URL, for example `https://ais.company.com`
 2. Without a reverse proxy: `ais-api` binds to `127.0.0.1:5080` (loopback only by default). Expose the port to the network and use the Docker host's accessible address, for example `http://<docker-host-ip>:5080`
4. Reload Innovator. The **AI Search** button appears in the command bar.

Access the Configurator

The AI Search UI runs inside Innovator as part of the command-bar integration. Opening the AIS API URL directly in a browser shows "Authentication Required" because the SPA requires the Innovator session context.

To reach the AI Search Configurator:

1. Open Innovator as an administrator.
2. Select the **AI Search** button in the command bar; the AI Search UI opens within Innovator.
3. Administrator users see a gear icon in the top-right corner of the AI Search UI. Select it to open the Configurator.

Define the indexing scope

1. In the AI Search Configurator, add the ItemTypes you want to make searchable (for example, `Part`, `Document`, `ECO`).
2. For each ItemType, select which properties to include in the search index.
3. Select **Apply**.

AI Search will provision the required message broker queues and bindings via the CDC configurator, enable SQL Server CDC tracking on the selected tables, and start initial indexing workflows in Temporal to index existing items.

Verify initial indexing

Monitor progress in the Temporal UI:



- Navigate to **Workflows** and filter by Workflow Type `initial-item` (item indexing) or `initial-file-indexer` (file indexing).
- Running workflows indicate indexing is in progress; they complete automatically.

You can also monitor progress in the Indexing Status section of the AI Search Configurator. See the **AI Search Configurator - Admin Guide** for details.

Troubleshooting

Certificate not found or authentication errors

Symptom: `ais-indexer-worker` logs show `FileNotFoundException` for the PFX, OAuth `401 Unauthorized` responses, or a `CryptographicException` wrapping `UnauthorizedAccessException: Access to the path '...pfx' is denied / IOException: Permission denied`.

Checks:

1. Confirm the file exists at `<CERTS_HOST_DIR>/<INNOVATOR_OAUTH_CLIENT_ID>.PFX`
2. Verify the container can read it:

```
docker compose exec ais-indexer-worker sh -c "echo /Container/Certificates/*"
```

This only proves the file is listed, not that it is readable; the AIS images run as a fixed non-root user. Check which one:

```
docker inspect <indexer-worker-image> --format '{{.Config.User}}'
```

If the `.pfx` was generated or copied with a restrictive mode (e.g. `0600`, owner-only) and is owned by root, the non-root container user cannot open it — the path exists but the load still fails with `UnauthorizedAccessException / Permission denied`. Inspect the file's owner and mode as seen from a Linux container (works the same way from a Windows or Mac Docker host):

```
docker run --rm -v <CERTS_HOST_DIR>:/certs busybox stat /certs/<INNOVATOR_OAUTH_CLIENT_ID>.pfx
```

Fix by making the file world-readable (matching the default mode of the `.cer` file), then restart the worker so it reloads the certificate:



```
docker run --rm -v <CERTS_HOST_DIR>:/certs busybox chmod 644  
/certs/<INNOVATOR_OAUTH_CLIENT_ID>.pfx  
docker compose restart ais-indexer-worker
```

3. Confirm `ARAS_OAUTH_CERTIFICATE_PASSWORD` in `.env` matches the PFX password (can be empty).
4. Verify the OAuth client is registered in Innovator with the correct Client ID.

AMQP connection refused or TLS errors

Symptom: `ais-indexer-worker` or `ais-api` logs show AMQP connection errors on startup.

Checks:

1. Verify the `AMQP_CONNECTION_STRING` scheme and broker type:
 1. **RabbitMQ:** `amqs://` (port 5671) or `amqp://` (port 5672)
 2. **Azure Service Bus:** `Endpoint=sb://...` or `amqs://...servicebus.windows.net`
2. **RabbitMQ only:** confirm the broker is reachable on the AMQP port:

```
nc -zv <rabbitmq-host> 5671 # TLS (production)
```

Expected: Connection to `<RABBITMQ-HOST>` ... succeeded

3. **RabbitMQ only:** verify the `rabbitmq_amqp1_0` plugin is enabled:

```
docker exec <rabbitmq-container> rabbitmq-plugins list | grep amqp1_0
```

4. **Azure Service Bus only:** confirm the namespace and policy name are correct and the policy has Send and Listen permissions on the topic or queue.

Embeddings API key invalid

Symptom: `ais-indexer-worker` or `ais-api` logs show `401 Unauthorized` responses from the embeddings endpoint.

Health check:

```
curl -s http://localhost:5080/health/ready/embeddings
```



If it returns `Unhealthy` , the API key or endpoint URL is incorrect.

Fix: Correct `EMBEDDINGS_API_KEY` and `EMBEDDINGS_ENDPOINT` in `.env` , then restart:

```
docker compose restart ais-api ais-indexer-worker
```

Database migration failed

Symptom: `ais-db-migrations` container exits with non-zero code; `ais-api`, `ais-indexer-worker`, and `ais-parser-worker` fail to start.

Check the migration logs:

```
docker compose logs ais-db-migrations
```

Common causes:

Error	Fix
Login failed for user	Verify AIS_DB_SCHEMA_CONNECTION_STRING credentials
Cannot open server...	SQL Server not reachable; check host and port and firewall
User does not have permission to perform this action	Grant db_owner to the login on the Innovator database
Timeout expired or execution timeout expired	The DiskANN vector index migration may time out on large datasets. Set AIS_DB_MIGRATION_COMMAND_TIMEOUT_SECONDS to 3600 or higher in <code>.env</code> , then rerun: <code>docker compose up ais-db-migrations</code>

Temporal worker not polling

Symptom: Temporal UI shows no workers on the expected task queue; indexing workflows do not start.

Checks:

- 1. Confirm `ais-indexer-worker` container is **Up**:



```
docker compose ps ais-indexer-worker
```

2. Check worker logs for connection errors:

```
docker compose logs --tail 100 ais-indexer-worker | grep -i "temporal\|error"
```

3. Verify the health probe for Temporal:

```
curl -s http://localhost:5080/health/ready/temporal
```

4. Confirm TEMPORAL_HOST in .env is correct and reachable from Docker:

```
docker run --rm --network ais-network busybox nc -zv <temporal-host> 7233
```

Health endpoint returns Degraded or Unhealthy

The `/health/ready` response body contains the status of each subsystem. Query individual probes to isolate the failing component:

```
curl -s http://localhost:5080/health/ready | jq . # or: python3 -m json.tool
```

Cross-reference with the subsystem-specific troubleshooting sections above.

If issues persist after following the steps above, contact Aras Technical Support via the Subscriber Portal with your tenant ID, the symptom observed, and any error details from the relevant container logs.

