How to Install a Simple OpenCTI Instance (Beginner Guide)

1 Overview

This guide walks you through installing a basic, working OpenCTI instance using Docker, and then safely adding connectors (official or custom).

2 Step 1: Install Docker

Install Docker Desktop and verify the installation:

```
docker --version
docker compose version
```

3 Step 2: Create a Project Folder

Create a project folder (example: `opencti-docker`) and create the following files inside it:

- `docker-compose.yml`
- `.env`

Optional (recommended for connectors):

- `docker-compose.override.yml`

4 Step 3: Create the .env File

Create a file named `.env` in the project folder with the following example values (highly recommend for security purposes change all the default passwords). You will notice in the environment file we are integrating API keys for AlienVault OTX. In this example, we integrate an AlienVault OTX connector to retrieve the latest threat-reporting indicators. This could be used mainly for consistent threat intelligence integration. For the purposes of the experiment, we will be manually adding indicator domains to the OpenCTI platform from the malicious activity periods of the BCCC DNS dataset.

```
OPENCTI_ADMIN_EMAIL=admin@opencti.local
OPENCTI_ADMIN_PASSWORD=ChangeMe123!
OPENCTI_ADMIN_TOKEN=ChangeThisToken123!

# Base services
REDIS_PASSWORD=opencti
ELASTIC_PASSWORD=opencti

# MinIO object storage
MINIO_ROOT_USER=opencti
MINIO_ROOT_PASSWORD=opencti123

# Optional connector tokens (create in OpenCTI UI and paste here)
OPENCTI_ALIENVAULT_TOKEN=PASTE_TOKEN_HERE

# AlienVault OTX API key (required if you use the AlienVault connector)
ALIENVAULT_API_KEY=PASTE_OTX_API_KEY_HERE
```

5 Step 4: Create docker-compose.yml (Minimal Working Stack)

Create `docker-compose.yml` with the minimum services required to run OpenCTI. Docker volumes are used for persistent data.

5.1 docker-compose.yml

```
services:

  redis:
    image: redis:8.2.1
    restart: always
    volumes:
      - redisdata:/data
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      timeout: 5s
      retries: 3

  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.19.2
    volumes:
      - esdata:/usr/share/elasticsearch/data
    environment:
```

```
    - discovery.type=single-node
    - xpack.ml.enabled=false
    - xpack.security.enabled=false
    - thread_pool.search.queue_size=5000
    - logger.org.elasticsearch.discovery="ERROR"
    - "ES_JAVA_OPTS=-Xms${ELASTIC_MEMORY_SIZE} -Xmx${ELASTIC_MEMORY_SIZE}"
  restart: always
  ulimits:
    memlock:
      soft: -1
      hard: -1
    nofile:
      soft: 65536
      hard: 65536
  healthcheck:
    test: ["CMD-SHELL", "curl -s http://elasticsearch:9200/_cluster/health?
  wait_for_status=yellow >/dev/null || exit 1"]
    interval: 5s
    timeout: 3s
    retries: 50

minio:
  image: minio/minio:RELEASE.2025-06-13T11-33-47Z
  volumes:
    - s3data:/data
  ports:
    - "9000:9000"
  environment:
    MINIO_ROOT_USER: ${MINIO_ROOT_USER}
    MINIO_ROOT_PASSWORD: ${MINIO_ROOT_PASSWORD}
  command: server /data
  restart: always
  healthcheck:
    test: ["CMD", "mc", "ready", "local"]
    interval: 10s
    timeout: 5s
    retries: 3

rabbitmq:
  image: rabbitmq:4.1-management
  environment:
    - RABBITMQ_DEFAULT_USER=${RABBITMQ_DEFAULT_USER}
    - RABBITMQ_DEFAULT_PASS=${RABBITMQ_DEFAULT_PASS}
    - RABBITMQ_NODENAME=rabbit01@localhost
```

```
    volumes:
      - type: bind
        source: ./rabbitmq.conf
        target: /etc/rabbitmq/rabbitmq.conf
      - amqpdata:/var/lib/rabbitmq
    restart: always
    healthcheck:
      test: rabbitmq-diagnostics -q ping
      interval: 30s
      timeout: 30s
      retries: 3

  opencti:
    image: opencti/platform:6.7.17
    environment:
      - NODE_OPTIONS=--max-old-space-size=8096
      - APP__PORT=8080
      - APP__BASE_URL=${OPENCTI_BASE_URL}
      - APP__ADMIN__EMAIL=${OPENCTI_ADMIN_EMAIL}
      - APP__ADMIN__PASSWORD=${OPENCTI_ADMIN_PASSWORD}
      - APP__ADMIN__TOKEN=${OPENCTI_ADMIN_TOKEN}
      - APP__APP_LOGS__LOGS_LEVEL=error
      - REDIS__HOSTNAME=redis
      - REDIS__PORT=6379
      - ELASTICSEARCH__URL=http://elasticsearch:9200
      - ELASTICSEARCH__NUMBER_OF_REPLICAS=0
      - MINIO__ENDPOINT=minio
      - MINIO__PORT=9000
      - MINIO__USE_SSL=false
      - MINIO__ACCESS_KEY=${MINIO_ROOT_USER}
      - MINIO__SECRET_KEY=${MINIO_ROOT_PASSWORD}
      - RABBITMQ__HOSTNAME=rabbitmq
      - RABBITMQ__PORT=5672
      - RABBITMQ__PORT_MANAGEMENT=15672
      - RABBITMQ__MANAGEMENT_SSL=false
      - RABBITMQ__USERNAME=${RABBITMQ_DEFAULT_USER}
      - RABBITMQ__PASSWORD=${RABBITMQ_DEFAULT_PASS}
      - PROVIDERS__LOCAL__STRATEGY=LocalStrategy
      - APP__HEALTH_ACCESS_KEY=${OPENCTI_HEALTHCHECK_ACCESS_KEY}
    ports:
      - "8080:8080"
    depends_on:
      redis:
        condition: service_healthy
```

```
      elasticsearch:
        condition: service_healthy
      minio:
        condition: service_healthy
      rabbitmq:
        condition: service_healthy
    restart: always

  worker:
    image: opencti/worker:6.7.17
    environment:
      - OPENCTI_URL=http://opencti:8080
      - OPENCTI_TOKEN=${OPENCTI_ADMIN_TOKEN}
      - WORKER_LOG_LEVEL=info
    depends_on:
      opencti:
        condition: service_healthy
    deploy:
      mode: replicated
      replicas: 3
    restart: always

volumes:
  esdata:
  s3data:
  redisdata:
  amqpdata:
```

6 Step 5: Start OpenCTI

From inside the project folder, run:

```
docker compose up -d
```

First startup may take 5–10 minutes.

7 Step 6: Verify and Log In

Check containers:

```
docker compose ps
```

Open the UI in a browser:

`http://localhost:8080`

Log in using the admin email and password from the `.env` file.

8 Step 7: Add Connectors Safely (docker-compose.override.yml)

OpenCTI connectors should be added **only after the base platform is running and healthy**. To keep the core stack stable and upgradeable, connectors must be defined in a separate file named `docker-compose.override.yml`.

This approach allows you to:

- Keep the base `docker-compose.yml` unchanged
- Add or remove connectors without destabilizing OpenCTI
- Upgrade OpenCTI safely without merging connector changes
- Isolate connector failures from core services

8.1 Why Use docker-compose.override.yml

Docker Compose automatically merges `docker-compose.override.yml` with `docker-compose.yml` when both are present. This means:

- Base services (OpenCTI, Redis, Elasticsearch, MinIO, RabbitMQ) remain untouched
- Connector services are layered on top

8.2 Example: docker-compose.override.yml

Below is an example override file that adds external and internal connectors. All connector images are pinned to the same OpenCTI platform version to prevent compatibility issues. The file should be located in the same directory as your docker-compose.yml.

```
services:

  connector-mitre:
    image: opencti/connector-mitre:6.7.17
    environment:
      - OPENCTI_URL=http://opencti:8080
      - OPENCTI_TOKEN=${OPENCTI_ADMIN_TOKEN}
      - CONNECTOR_ID=mitre
      - CONNECTOR_TYPE=EXTERNAL_IMPORT
      - CONNECTOR_NAME=MITRE ATT&CK
      - CONNECTOR_SCOPE=identity,attack-pattern,malware,tool,course-of-action
      - CONNECTOR_CONFIDENCE_LEVEL=80
      - CONNECTOR_LOG_LEVEL=info
    restart: always
    depends_on:
      - opencti
```

```
  connector-alienvault:
    image: opencti/connector-alienvault:6.7.17
    environment:
      - OPENCTI_URL=http://opencti:8080
      - OPENCTI_TOKEN=${OPENCTI_ADMIN_TOKEN}
      - CONNECTOR_ID=${ALIENVAULT_CONNECTOR_ID}
      - CONNECTOR_TYPE=EXTERNAL_IMPORT
      - CONNECTOR_NAME=AlienVault OTX
      - CONNECTOR_SCOPE=indicator
      - CONNECTOR_CONFIDENCE_LEVEL=70
      - CONNECTOR_LOG_LEVEL=info
      - ALIENVAULT_BASE_URL=https://otx.alienvault.com
      - ALIENVAULT_API_KEY=${ALIENVAULT_API_KEY}
      - ALIENVAULT_REPORT_STATUS=${ALIENVAULT_REPORT_STATUS}
      - ALIENVAULT_FILTER_SUBSCRIPTION=${ALIENVAULT_FILTER_SUBSCRIPTION}
      - ALIENVAULT_PULSE_START_TIMESTAMP=${ALIENVAULT_PULSE_START_TIMESTAMP}
      - ALIENVAULT_SSL_VERIFY=${ALIENVAULT_SSL_VERIFY}
    restart: unless-stopped
    depends_on:
      - opencti
```

8.3 Bringing the Stack Up with Connectors

Once the base OpenCTI platform is running, bring up the full stack (base + connectors) using:

```
docker compose -f docker-compose.yml -f docker-compose.override.yml up -d
```

Docker Compose will:

- Start all base OpenCTI services (if not already running)
- Start each connector container
- Automatically connect connectors to the OpenCTI API

8.4 Important Connector Rules

- **Always pin connector images** to the same OpenCTI version
- Use a **unique CONNECTOR_ID** for every connector (UUID recommended)
- Use **dedicated API tokens** per connector (do not reuse admin tokens)
- Add connectors only after confirming OpenCTI is healthy

9 Operational Commands

Stop services (keeps volumes):

```
docker compose down
```

Start again:

```
docker compose up -d
```

Follow logs:

```
docker compose logs -f --tail 200
```