Link to Wiki-Page:
https://github.com/amosproj/amos2025ws01-opensearch-load-tester/wiki/Build-Documentation

TLDR

Run the **interactive deployment** and find the results in `${PWD}/reports`.

```
make loadtest
```

Requirements

The following tools must be installed

- docker
- docker-compose
- make

Deployment

The deployment process brings up all components of the load-testing environment, builds the required images, and starts the full OpenSearch stack. The steps below describe how to initiate and control this process.

Interactive setup

First ensure that all required tools are installed. To deploy the full stack start the interactive setup `make loadtest`

The setup prompts for

- how many load generators should be started
- how many test files should be uploaded to the OpenSearch instance

During this interaction several tasks run automatically in the background

```
make clean
make build
make run
```

These commands clean up the environment build the Docker images and start the complete system including OpenSearch.

Viewing logs

All container logs can be viewed with

```
make logs
```

This attaches to the logs of every container defined in `docker-compose.yaml`.

Interacting with OpenSearch

To send HTTP requests to OpenSearch open an interactive shell inside a helper container

```
make curl
```

This container runs in the same Docker network and allows direct interaction with the OpenSearch API over HTTP.
You can reach the instance using the DNS name `http://test-target-opensearch:9200` on port 9200.

Resource usage

To display CPU and memory consumption of all running containers use

```
make status
```

Test results

After the load test finishes the metrics reporter writes all results into `${PWD}/reports`
Both CSV and JSON reports are generated for later analysis.

Available Make Commands

Command	Description
make build	Builds all Docker images
make clean	Stops and removes containers

make curl	Opens a shell in the curl container
make help	Shows all available commands
make loadtest	Starts the interactive load-test setup
make logs	Streams logs from all containers
make run	Starts all containers in the background
make status	Shows CPU and memory usage of all containers
make stop	Stops all running containers

Build Process

The project contains three Spring Boot applications located in

- ./testdata-generator
- ./load-tester
- ./metrics-reporter

Dockerfiles

Each module has its own Dockerfile. All applications use a version pinned multi stage Docker build.
The build stage uses `maven:3.9-eclipse-temurin-25` as base image and the runtime stage uses a minimal image based on `eclipse-temurin:25-jre-alpine`

The application is placed in the `/app` directory and executed using the non-root user `user` to reduce privilege escalation risks.

Container orchestration

All three images are managed through a single docker-compose.yaml file and are configured via a .env file. The stack includes a bare-bones OpenSearch instance.
All containers run inside the Docker bridge network `opensearch-loadtester-network` and communication takes place over HTTP using Docker DNS resolution.

The metrics reporter is the only container with a host-mounted volume. It writes all test reports into the local reports directory to make them accessible outside the container environment.

Convenience commands

For common tasks such as building images stopping containers or cleaning up the environment use

```
make build
make stop
make clean
```