Build & Deployment Documentation

All components of the application are normally build using Docker. There are also configuration files for a execution without Docker for faster development cycles, but we do not recommend to run production environments without Docker.

Environment Variables

Here you can find a short description for all environment variables which are set via the docker-compose file.

Generator:

Variable	Default	Description
HOST	0.0.0.0:34254	The host of the generator. Should use 0.0.0.0 for accepting all incoming connection requests and specifies the published port
TARGET	backend:34255	The target for the UDP socket. Default uses docker DNS.
PPS	10000.0	The packages per second or sample rate
SIQ_FREQ	1.0	The signal frequency

Backend:

Variable	Default	Description
UDP_PORT	34255	Bind address for the UDP port. Should be equal to mapped port used in generator:TARGET
WS_PORT	9000	Bind port for the websocket. Should be equal to port used in frontend .env (VITE_BACKEND_WEBSOCKET) config

Variable	Default	Description
REST_PORT	8080	Bind port for rest api inside the container. Mapped port should be equal to port used in frontend .env (VITE_BACKEND_BASE_URL) config
HOST_ADDRESS	0.0.0.0	Sepcifies the host address. Use 0.0.0.0 to accept all incoming connections.
FRONTEND_ORIGIN	http://localhost:5000	Specifies the frontend origin to allow CORS.
API_KEY	sosci-local	Api key used for authentication at rest api.

Frontend:

The frontend mainly manages the configuration ion .env files. There are 5 configuration files for different scenarios.

File	Description
.env	For local npm execution
.env.localhost	For local docker execution
.env.develop	For develop deployments
.env.integration	For integration deployments
.env.production	For production deployments

Those files declare the following variables:

Variable	Description	Example
VITE_BASE_URL	The base url of the frontend	http://localhost:5000 for local default docker execution.

Variable	Description	Example
VITE_BACKEND_BASE_URL	The base url of the backend service.	http://localhost:8081 for local default docker execution.
VITE_BACKEND_WEBSOCKET	The address of the backend websocket	ws://localhost:9000 for local default docker execution.
VITE_REST_API_KEY	The api key for the backend rest api.	sosci-local

Keep in mind that secrets like the API_KEY should not be inside the source code. SOSCI's web environments use the GitHub Action Secrets to pass that information securly to the docker containers.

Local Build

Prerequisites

- Docker installed (Tutorial)
- docker-compose installed (Tutorial)

Build Tasks

1. Enter the Apps directory: `cd Apps/`
2. Build & run all images: `docker-compose build`
3. Run detached the container: `docker-compose up -d`
4. Rebuild specific container: `docker-compose up -d --build <generator/backend/frontend>`

Web Build

Officially provided images are managed via CICD. * Closed pull requests to dev branch update the nightly tag * Closed pull requests to int branch create a new sprint-XX-release-candidate image tag * Closed pull requests to main branch create a new sprint-XX-release image tag and update the latest tag ### CICD Steps

Related Github Actions workflows:

Workflow	Description
build-generator-automation.yml	Builds the generator image using docker/build-push-action@v3 Does not push the image to DockerHub Used for compile & build testing
build-backend-automation.yml	Builds the backend image using docker/build-push-action@v3 Does not push the image to DockerHub Used for compile & build testing
build-frontend-automation.yml	Builds the frontend image using docker/build-push-action@v3 Does not push the image to DockerHub Used for compile & build testing
push-generator-automation.yml	Builds & pushes the generator image using docker/build-push-action@v3 Pushes the image to DockerHub
push-backend-automation.yml	Builds & pushes the backend image using docker/build-push-action@v3 Pushes the image to DockerHub
push-frontend-automation.yml	Builds & pushes the frontend image using docker/build-push-action@v3 Pushes the image to DockerHub
test-frontend-automation.yml	Runs frontend tests
test-backend-automation.yml	Runs backend tests, fails if the backend code coverage is below 70%
test-generator-automation.yml	Runs generator tests
test-generator-codecov.yml	Fails if generator unit test code coverage is below 70%
ci-codeql.yml	Runs CodeQL
ci-sonarqube.yml	Runs a SonarScanner and reports to Sonarqube for SAST.
release-automation.yml	Creates a new release or release-candidates of pull request with release or release-candidate tags based on the PRs title and description.

DockerHub Repositories

The DockerHub user is owned by the development team. - Generator - Backend - Frontend

Image tags

Image Tag	Description
nightly	Used for develop deployments. Continuously updated. Not stable.
sprint-XX-release-candidate	Used for integration deployments. New tag for each sprint (example: sprint-01-release-candidate) Stable.
sprint-XX–release	Used for production deployments. New tag for each sprint. (example: sprint-01-release) Stable
latest	Not used Latest stable release Stable

Deployment

Local Deployment

Prerequisites

- Docker installed (Tutorial)
- docker-compose installed (Tutorial)

Deployment Tasks

1. Enter the Apps directory: `cd Apps/`
2. Run all containers: `docker-compose up`
3. Run detached the container: `docker-compose up -d`
4. Rebuild specific container: `docker-compose up -d --build <generator/backend/frontend>`

The frontend should be available in the browser of your choice at http://localhost:5000

Web Deployment

Deployments to web environments are managed via CICD * Closed pull requests to dev branch update the develop environment * Closed pull requests to int branch update the integration environment * Closed pull requests to main branch updatethe production environment

---

The deployment itself is hosted using Portainer and creates a Docker stack based on the `Deployments/web_deployment/<environment>/docker-compose.yml` This stack is linked to the repository and provides a webhook which will trigger a redeployment of the stack. The application is served via Traefik Container reverse proxy which is configured using labels on the Docker containers.

Environments

- Develop - Nightly Builds
- Integration - Latest Release Canidate
- Production - Latest Release

CICD Steps

Related Github Actions workflows:

Workflow	Description
push-generator-automation.yml	Builds & pushes the generator image using docker/build-push-action@v3 Pushes the image to DockerHub Triggers the webhook linked to Degen Hosting environment
push-backend-automation.yml	Builds & pushes the backend image using docker/build-push-action@v3 Pushes the image to DockerHub Triggers the webhook linked to Degen Hosting environment
push-frontend-automation.yml	Builds & pushes the frontend image using docker/build-push-action@v3 Pushes the image to DockerHub Triggers the webhook linked to Degen Hosting environment