Build

All components of the application are build using Docker.

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

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`

The stacked 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