Automated Build Process

The backend and frontend of the application is run through a pipeline of building the application and executing unit tests seperatly on each push on every branch.

Component	Workflow	Actions
Backend	`build-backend.yml`	Build application, run unit tests
Frontend	`build-frontend.yml`	Build application, run unit tests

Building / Deploy the Application Setup locally

Requirements

- Docker / Docker Desktop (incl. Docker Compose)

Starting the Setup

Once the repository is checked out, on the project root directory execute:

```
docker compose --profile test up --build
```

This will compile, build and start the application(s) in a test setup manner that represents all aspects of the application (database, frontend and multiple containers of the backend representing the different roles/parts of the application).

Under **http://localhost:4200** you can view the UI of the application and interact with it as a user.

Component	Container	Port
Frontend	Angular + Nginx	4200
Cluster Manager	Spring Boot	8080
Backup Nodes	Spring Boot (simulated)	—
Database	PostgreSQL	5432
Prometheus	Metrics collection	9090
Grafana	Metrics visualization	3000

Accessing the Application

Service	URL	Credentials

Service	URL	Credentials
Frontend (UI)	http://localhost:4200	—
Backend API	http://localhost:8080	—
Grafana	http://localhost:3000	admin / admin
Prometheus	http://localhost:9090	—

Alternative Profiles

Command	Description
`docker compose up --build`	Frontend + Cluster Manager + Database only
`docker compose --profile test up --build`	Full stack with simulated Backup Nodes

Stopping the Application

```
docker compose --profile test down
```

To remove volumes (clean database):

```
docker compose --profile test down -v
```

Building Individual Components

Backend

```
cd backend
mvn clean install
mvn spring-boot:run -Dspring-boot.run.profiles=cluster_manager,dev
```

Frontend

```
cd frontend
npm install
ng serve
```

Stress Testing

Setup

In `docker-compose.yml`, change the cluster-manager profile:

```
- SPRING_PROFILES_ACTIVE=cluster_manager,test-runner
```

Running a Test

```
docker compose --profile test-backup-volume-create up --build
```

Results are written to `/stress-test/test-backup-volume-create/results/`

Available Tests

Test	Description	Config
`test-backup-volume-create`	Concurrent backup creation across nodes	`MAX_VUS`, `ITERATIONS`
`test-backup-volume-delete`	Create + delete backups concurrently	`MAX_VUS`, `ITERATIONS`
`test-backup-volume-read`	Stress `GET /api/v1/cm/backups` with ramping users	`INITIAL_BACKUPS`, `BACKUP_INCREMENT`, `ITERATIONS`

Configuration

Configure tests via environment variables in `docker-compose.yml`:

Variable	Description
`MAX_VUS`	Maximum number of concurrent virtual users
`ITERATIONS`	Number of test iterations to perform
`INITIAL_BACKUPS`	Initial backups to create (read test only)
`BACKUP_INCREMENT`	Backups added per iteration (read test only)

Each test folder (`/stress-test/test-*/`) contains its own `README.md` with detailed instructions.