**Getting started**

Detailed setup instructions can be found in each application's README file. To conveniently start all applications, the run\_all\_applications scripts can be used. Alternatively, to start a minimal subset of applications, i.e., the Customer Management Applications and the Customer Core, use the run\_customer\_management\_applications script.

1. Make sure you have [Java 8, 11 or 15](https://adoptopenjdk.net/) installed.
2. Install [Node](https://nodejs.org/en/). Version 8.2 or later is required. You can check the currently installed version by running node --version.
3. Install Python. We don't use Python ourselves, but some Node.js packages require native addons that are built using node-gyp, which requires Python. See the [node-gyp README for details on which Python version to install](https://github.com/nodejs/node-gyp#on-unix).
4. Install Maven (see [https://maven.apache.org](https://maven.apache.org/) for installation instructions).
5. Run the run\_all\_applications script suitable for your platform. Note that the frontend applications might be running before the backends are ready. In that case, just reload the page in the browser.

If the script exits, one of the applications could not be started. For troubleshooting, we recommend to start the applications individually. Note that you don't need to start all applications. The overview diagram above can be used to figure out the dependencies of each service.

The following table lists all the ports that have to be free for each component to work correctly. If you need to change any of these ports, please consult the README of the corresponding component:

| **Component** | **Ports** |
| --- | --- |
| [Customer Self-Service Backend](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/customer-self-service-backend) | 8080 (HTTP resource API) |
| [Policy Management Backend](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/policy-management-backend) | 8090 (HTTP resource API) 61613 (ActiveMQ broker) 61616 (ActiveMQ broker) |
| [Customer Management Backend](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/customer-management-backend) | 8100 (HTTP resource API) |
| [Customer Core](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/customer-core) | 8110 (HTTP resource API) |
| [Customer Self-Service Frontend](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/customer-self-service-frontend) | 3000 (Web server) |
| [Policy Management Frontend](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/policy-management-frontend) | 3010 (Web server) |
| [Customer Management Frontend](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/customer-management-frontend) | 3020 (Web server) |
| [Risk Management Server](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/risk-management-server) | 50051 (gRPC server) |
| [Risk Management Client](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/risk-management-client) | - (CLI Client) |
| [Eureka Server](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/eureka-server) | 8761 (Admin web frontend) |
| [Spring Boot Admin](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/spring-boot-admin) | 9000 (Web server) |

**Docker**

All projects come with Dockerfiles that can be used to run the services as Docker containers. The [docker-compose.yml](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/docker-compose.yml) builds and starts all applications in a single command, just like the run\_all\_applications scripts mentioned above. See the [docker-compose.yml](https://github.com/Microservice-API-Patterns/LakesideMutual/blob/master/docker-compose.yml) for more information.

Note that the Dockerfiles make use of [BuildKit](https://docs.docker.com/develop/develop-images/build_enhancements/) features. To be able to build the Docker images, you have to set the DOCKER\_BUILDKIT environment variable accordingly.

The easiest way to build the individual images is by running the following command in the corresponding project directory:

DOCKER\_BUILDKIT=1 docker build

To build and run all applications with Docker Compose, use the following command:

COMPOSE\_DOCKER\_CLI\_BUILD=1 DOCKER\_BUILDKIT=1 docker-compose up

**Data Stores**

Each backend service has its own data store. The Spring-JPA based applications all use the H2 relational database. By default, all data will be lost during restarts, please see the individual README files to enable durable persistency. The backend services also contain the H2 Console to browse the database. It can be found at /console. For example, for the Customer Core, the address is <http://localhost:8110/console>.