# Integration Testing Setup Documentation for Movimingle Application

## Introduction

This document details the process for setting up integration tests for the Movimingle application, which includes the `favorites-management-service` and `voting-service`. Integration testing is crucial to ensure that the separate services of the application work cohesively before deployment. This setup utilizes Docker Compose for orchestrating an environment that mimics production systems and automates tests via GitHub Actions.

## Repository Creation and Configuration

The first step in setting up the integration tests is to create a new GitHub repository. This repository, named `movimingle-integration-tests`, is specifically dedicated to handling the Docker Compose configurations and the integration tests themselves. It serves as a central point to manage and trigger integration tests across the microservices which are maintained in separate repositories.

### Adding Submodules

Given that the microservices are developed and stored in separate repositories, they are integrated into the testing repository as submodules. This approach allows for managing the versioning of each service independently while still being able to test them together in an integrated environment.

To include each microservice as a submodule, the following commands are executed:

**git submodule add https://github.com/kalopresli/movimingle-favorites-management-service.git favorites-management-service**

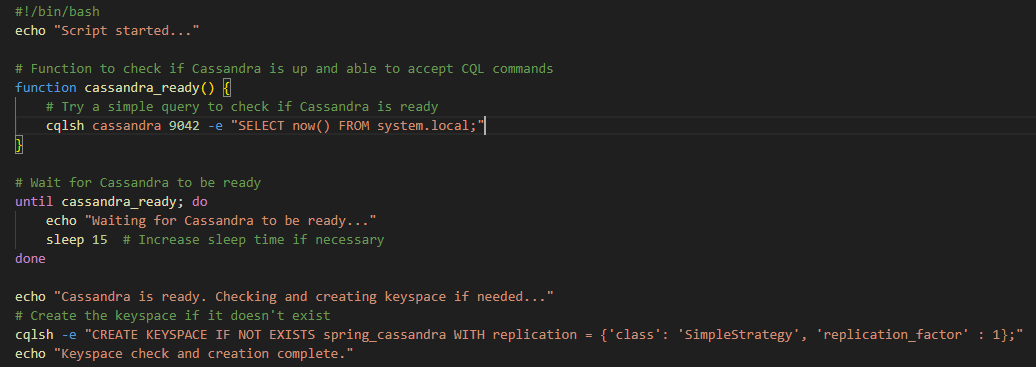
**git submodule add https://github.com/kalopresli/movimingle-voting-service.git voting-service**

## Docker Compose Configuration

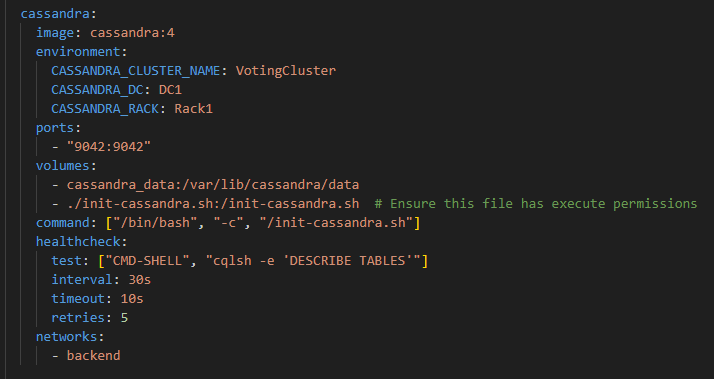
The Docker Compose file is a critical component of the setup, defining how the microservices and their dependencies are orchestrated. The Docker Compose configuration includes definitions for building the services from their respective Dockerfiles, setting up necessary networks, and managing dependencies like databases and message queues.

I spent a lot of time trying to figure out how to fix one problem with the Cassandra database, and after many trials I did not succeed in creating a successful script that creates a keyspace when starting the docker container in case the keyspace is not already created. The problems encountered there were related to the database being unable to connect to the correct network once I add a step for the execution of the bash script.

TO-DO: Include the creation of second (init-cassandra) container

Script:  


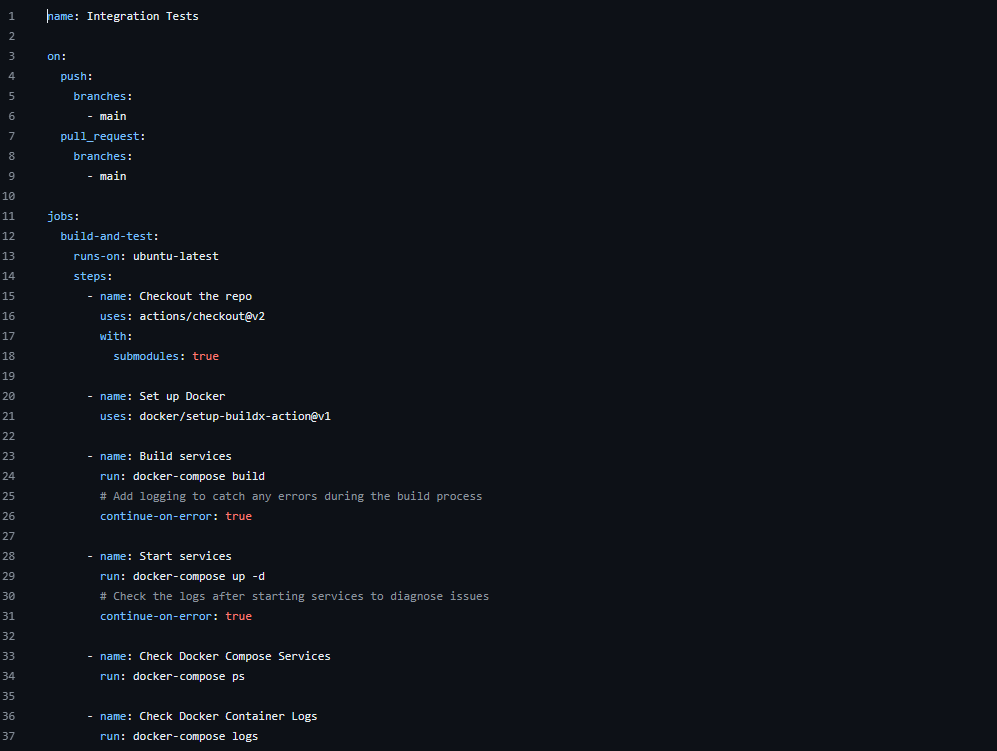
Cassandra call for the script:



This script is needed in order for the voting-service to be able to start, as with no connection to the DB, it is not possible to start the service. The rest of the components are working properly and sending a request to the favorites-management-service is sending a message to the RabbitMQ instance. However, my initial idea was to show the logs of voting-service, where the message would be received, thus showing that there is a successful connection between the two services, through the message broker.

## GitHub Actions for Automated Testing

To facilitate continuous integration and ensure that all components work together seamlessly, GitHub Actions are configured to automate the integration tests. The `.github/workflows/integration-tests.yml` file defines the workflow that GitHub follows upon code pushes to the main branch or pull requests.



## Running and Monitoring Tests

For manual execution, the Docker Compose file can be used to run the services locally:

**docker-compose up -d --build  
docker-compose ps**

## Updating Submodules

Regular updates are necessary to ensure that the latest changes in each microservice are included in the integration tests. This can be achieved by updating the submodules:



## Conclusion

The integration testing setup described here provides a robust framework for ensuring that the `favorites-management-service` and `voting-service` of the Movimingle application work together as expected. By automating the testing process and utilizing Docker Compose for service orchestration, the setup helps in maintaining high quality and reliability of the application prior to production deployment.