Selected Issues of Network Applications
2023/2024

Containerization and DevOps – let's automate IaaS

Group members
Wiktor Pawlak, 236621
Szymon Ziemecki 236707
Szymon Zakrzewski
Piotr Skonieczny
Weronika Tutkaj
Mateusz Strzelecki
Oskar Trela

1. Problem description

Manually deploying applications is a cumbersome process that drains time and resources while introducing many issues. Developers and operations teams face several challenges when they rely on manual deployment:

- Time-Consuming
- Error-prone
- Developer Overhead
- Inconsistent Environments
- Dependency on Developer Knowledge
- Lack of Version Control
- Scalability Challenges

Solution

The solution to these deployment challenges lies in automation – **CI/CD**.
Upon pushing code changes to remote repository, a pipeline should be triggered.
It should consist of several stages: build application, test, build application image, push to repository, deploy to test environment.

- Increased efficiency
- Consistency and reliability
- Continuous process
- Developer changes are easy to deploy
- Cost reduction

Manual integration and deployment give developer control, there is no initial setup overhead and it is faster and simple for small projects.
CI/CD usually means that there must be a special ops team responsible for maintenance and updating tools & pipelines.

2. Selected tools

Choice:

- Github actions – automation of development process by compiling, running tests and code deployment directly from GitHub repository.
- OpenShift TUL - cloud platform where application and database will be hosted which will take away our responsibility for infrastructure
- OpenShift - managing containerized services, will help with management of multiple cloned instances of application
- Docker hub – Docker image repository, will be used to persist the built image

3. Functional requirements

Automation of the following steps (GitHub Actions):

- Tests execution (JUnit & Testcontainers)
- Application image preparation (Docker image)
- Push application image to remote image repository (Docker Hub)
- Application deployment (OpenShift)

4. Schedule

(6th week) - Preparation of application and tests
(7th week) - GitHub Actions pipelines with image building
(8th week) - Configuration and manual application deployment on OpenShift
(9th week) - Application deployment on OpenShift with GitHub Actions pipeline
(10th week) - Testing

5. Work division

Wiktor Pawlak

- Preparation of application and tests,
- GitHub Actions pipelines with image building,
- Testing

Szymon Ziemecki

- Preparation of application and tests,
- GitHub Actions pipelines with image building,
- Testing

Oskar Trela

- Preparation of application and tests,
- GitHub Actions pipelines with image building,
- Testing

Mateusz Strzelecki

- Configuration and manual application deployment on OpenShift,
- Testing

Szymon Zakrzewski

- Configuration and manual application deployment on OpenShift
- Testing

Piotr Skonieczny

- Application deployment on OpenShift with GitHub Actions pipeline
- Testing

Weronika Tutkaj

- Application deployment on OpenShift with GitHub Actions pipeline
- Testing

6. Insights

Indeed, automation makes development much easier. Despite the initial overhead of configuring CI/CD and OpenShift, much time is saved later. The process of testing code changes is satisfying.

7. Challenges

Downloading application image from DockerHub. The symptom was an unauthorized HTTP response.
The solution was to add DockerHub credentials as secrets and configure the use of them in the deployments config.
Secret must be in the same namespace as the deployments config file.
In addition, the configured image tag was different from the one created during pipeline which further prevented correct deployment.

7. Further work

Further potential work could involve refinement of GitHub Actions pipelines to incorporate more robust testing suites and deployment strategies.
Exploration of container orchestration tools other than OpenShift, such as Kubernetes or Docker Swarm.
Conducting performance and scalability testing, optimizing application resources, and implementing continuous monitoring apart from Grafana dashboards like APM and logging mechanisms in ELK stack or distributed tracing with Grafana prometheus, loki and tempo.