DEVSECOPS

Reasoning, Conclusions & Recommendations

Fontys

HBO ICT

OPENING

In this document I will explain why I made certain decisions in my implementations and why I did not go for a different approach. The topics that we will be going over are DevOps & Code Quality

CHALLENGE

Water polo is a niche sport in the Netherlands. Unlike football, hockey, or tennis where each town might have one club of the three sports and big cities might have multiple within the same sport. This results in clubs easily finding opponents to play practice matches outside the main competition. Water polo on the other hand besides having a smaller group of athletes also has the problem that it is restricted to towns and cities with a swimming pool which is also competition approved. This combination leads to difficulty finding practice matches, the idea for this project is an integrated system where teams can find each other to play practice matches, contact each other, find relevant personnel like referees, keep track of matches themselves and create an environment which promotes competitive play among the regions.

REASONING

DEPLOYMENT, DOCKER, AND KUBERNETES

The industry standard is for development to find place in dockerized containers, and to deploy the system to a server or a cloud service. To make use of both the features I tried a deployment method where a MiniKube services pulls an image from Dockerhub on startup and each pod contains a Docker container which scales according to CPU usage. To update the Dockerhub images each service has his own pipeline and when a GitHub page which has each service as a submodule updates a pipeline will run telling each pod to pull a new image from Dockerhub.

This allows for the system to be update all the images at once resulting in more consistency with the drawback of a slightly longer down time. At same time as all images are pushed to Dockerhub separately we don’t need a pipeline to build all images and we can rely on reducing pipeline wait time.

Lastly the reason to combine the two leads to a deployment that resembles the developer environment allowing for an experience where both the Docker container in development looks and acts like the container within the Kubernetes Pods.

LOGGING, QUALITY CONTROL

Quality Control was something that came early in the process, when researching the different frameworks, I came along a paper which explained why Java and old object-oriented language is still popular. Object-oriented language like Java forces the developer to follow rules which leads to less bugs in the long run.

To further assure quality and security different controls can be introduced into the pipeline to ensure quality. Sonarcloud and Jetbrains both provide checks on code quality, security risks and code dept. With JetBrains also checking for package updates and versions to make sure your system is up to date. Logging with Spring is also built in. With parameters such as time to response, and the ability to add objects, date, and custom events to a log file.

CONCLUSION & RECOMMENDATIONS

The DevSecOps section of this project is a combination of other learning outcomes with it combining Scalability, Security and Architecture. This leads to it mainly serving as a testing and development outcome to help with the other outcomes in my experience. An example of this is the logging system helped me identify where a problem was in my sorting and matching algorithm allowing for the system to both work but also create a performance improved.

For next time I would want to try more different deployment methods allowing for more alternatives that are not industry standard yet. Further I would like to do more research into if constant deployment and update is better compared to weekly updates with a regular hot patch if needed.