Kurs DevOps

Lista 12

21, 28 stycznia 2026

Zadanie 1. (14 p.)
Using Jenkins and the tools covered in class, set up a CI/CD environment that verifies and deploys a sample application of your choice.

- Place the source code of the application in a remote Git repository (e.g., on GitHub).
- Configure a local instance of Jenkins so that it periodically polls the remote repository,[1] to see if a new commit has been pushed.[2]
- In Jenkins configure the use of a Jenkinsfile from the repository as the pipeline definition.
- Define building and verification steps for the application in the Jenkinsfile.
- Prepare a Docker image that contains all dependencies needed to build and verify the application, plus a Jenkins agent.
- Using Kubernetes run the prepared image with its embedded Jenkins agent.
- Define an agent in the Jenkins.
- Show that Jenkins detects changes in the repository and triggers a build.
- Store the artifact built by Jenkins on a remote server (e.g., locally running `miniserve`).
- Define a new pipeline in Jenkins that will deploy the application (e.g., copying it into a container, creating a new Docker image).
- After successful tests in the verification pipeline with no errors, automatically trigger the deployment pipeline.
- Expose metrics from Jenkins to Prometheus[3] and based on those prepare an example dashboard in Grafana.

Scoring:

- Polling changes from the remote repository: 1p.
- Build and verification pipeline: 4p.
- Connecting the agent: 2p.
- Deployment pipeline: 4p.
- Prometheus + Grafana: 3p.

[1] Besides periodic polling, you can define hooks in Git/GitHub that would trigger Jenkins automatically. However, given the local nature of our Jenkins instance, polling will be easier to set up.

[2] With GitHub you'll likely need to generate a token and store it as a secret in Jenkins. In my case, a fine-grained token stored as "Username with password" was sufficient.

[3] e.g., using `https://plugins.jenkins.io/prometheus/`

Zadanie 2.
What mistakes were made in:
a)

```
pipeline {
    agent any
    environment {
        EXAMPLE_CREDS = credentials('example-credentials-id')
    }
    stages {
        stage('Example') {
            steps {
                sh("curl -u ${EXAMPLE_CREDS_USR}:${EXAMPLE_CREDS_PSW} https://example.com/")
            }
        }
    }
}
```

b)

```
pipeline {
  agent any
  parameters {
    string(name: 'STATEMENT', defaultValue: 'hello; ls /', description: 'What should I say?')
  }
  stages {
    stage('Example') {
      steps {
        sh("echo ${STATEMENT}")
      }
    }
  }
}
```

Zadanie 3. (2 p.)
Using the resources provided by GitHub in its free tier, show how to define a verification pipeline with GitHub Actions that:

- Compiles and runs a simple program written in C++.
- Executes twice: once on an x86 machine and once on an ARM64 machine.
- Uses the `matrix` strategy to specify the machines.

Zadanie 4. (3 p.)
Demonstrate how to use GitHub Actions to generate a website (e.g., using mdBook or Sphinx) that can then be hosted with GitHub Pages.