

Politechnika Wrocławska
Wydział Elektroniki

Kierunek: Cyberbezpieczeństwo

Ochrona centrów danych

Lab 2

Krzysztof Bocian 241183

1. Wstęp

Celem tych zajęć było zbudowanie projektu GitHub przy pomocy Jenkinsa. Jenkins miał za zadanie pobranie kodu źródłowego z repozytorium Githuba, kompilacja kodu i zapisanie wyniku spowrotem na Githubie.

2. Konfiguracja połączenia Jenkinsa i GitHuba.

Do skonfigurowania połączenia pomiędzy usługami wygenerowałem parę kluczy SSH za pomocą komendy ssh-keygen. W ten sposób wygenerowane zostały dwa pliki: id_rsa i id_rsa.pub. Zawartość klucza publicznego przeniosłem do repozytorium GitHuba w zakładce “Deploy Key”. Za pomocą klucza prywatnego stworzyłem “credential” w Jenkinsie. Jako Username oraz ID ustawiłem “git”.

3. Konfiguracja Pipeline

W Jenkinsie stworzyłem nowy projekt “pipeline” o nazwie “JobOne”. Jako tryb wybrałem “Pipeline script from SCM” i wcześniej utworzone Credentiale. W polu “script path” wybrałem Jenkinsfile.

The screenshot shows the Jenkins configuration interface for a pipeline named "Pipeline script from SCM". The configuration is as follows:

- SCM:** Set to "Git".
- Repositories:**
 - Repository URL:** git@github.com:KrzysztofBocian/Project-test.git
 - Credentials:** A dropdown menu shows "git (example)" with an "Add" button next to it.
 - Buttons:** "Advanced..." and "Add Repository".
- Branches to build:**
 - Branch Specifier (blank for 'any'):** */main
 - Buttons:** "Add Branch" and a red "X" icon.
- Repository browser:** Set to "(Auto)".
- Additional Behaviours:** An "Add" button is present.
- Script Path:** Jenkinsfile

Następnie utworzyłem 3 pliki w repozytorium: Jenkinsfile, Makefile oraz plik z kodem do wykonania hello.c

```
1 pipeline {
2     agent any
3
4     stages {
5         stage('Test') {
6             steps {
7                 echo 'Testing...'
8             }
9         }
10        stage('Build') {
11            steps {
12                echo "${pwd}"
13                sh "make"
14            }
15        }
16        stage('Run') {
17            steps {
18                sh './HelloWorld'
19            }
20        }
21    }
22 }
23 }
```

Zdj. 1. Jenkinsfile

```
1 # Makefile
2 compile: hello.c
3     gcc hello.c -o HelloWorld
```

Zdj. 2. Makefile

```
6 lines (6 sloc) | 127 Bytes

1  #include <stdio.h>
2  int main() {
3      // printf() displays the string inside quotation
4      printf("Hello, World!");
5      return 0;
6  }
```

Zdj. 3. hello.c

Powyższe zdjęcia przedstawiają wykonywany kod. Jenkinsfile początkowo wykonuje test za pomocą komendy “echo”, następnie wykonuje plik Makefile przy pomocy komendy “make”. W tym momencie plik Makefile tworzy z pliku hello.c plik możliwy do wykonania. Następnie Jenkinsfile wykonuje utworzony plik HelloWorld. Cały proces przedstawiony jest poniżej.

```
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
Testing...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
/var/lib/jenkins
[Pipeline] sh
+ make
gcc hello.c -o HelloWorld
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Run)
[Pipeline] sh
+ ./HelloWorld
Hello, World!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Zdj. 4. Logi jenkinsa przedstawiające wynik

4. Podsumowanie

Niestety nie udało się zrealizować wszystkich celów na to laboratorium. Problemy wynikały z braku doświadczenia w pracy z Jenkinsem. Mimo wszystko udało się zrealizować pobieranie i budowanie projektu z githuba za pomocą Jenkinsa. Dzięki tym zajęciom udało się znacznie poszerzyć wiedzę na temat Jenkinsa, pliku Makefile oraz Jenkinsfile.