

Ochrona Centrów Danych

Raport 2



Politechnika Wrocławska

Anna Płecha, 241446

Wrocław, 2021

Spis treści

1. Budowa projektu GitHub przy pomocy Jenkinsa.	3
2. Kompilacja do postaci binarnej. Tryb potokowy (Jenkins pipeline)	5
3. Publikacja artefaktów binarnych w testowym repozytorium GitHub	8
4. Źródła.	12

1. Budowa projektu GitHub przy pomocy Jenkinsa

W tej części należało ustawić automatyczny dostęp do repozytorium GitHub dla serwisu Jenkins. Przykładowy kod źródłowy umieszczony w stworzonym na potrzeby repozytorium testowym powinien być pobierany automatycznie.

Na początku stworzono testowe repozytorium GitHub.

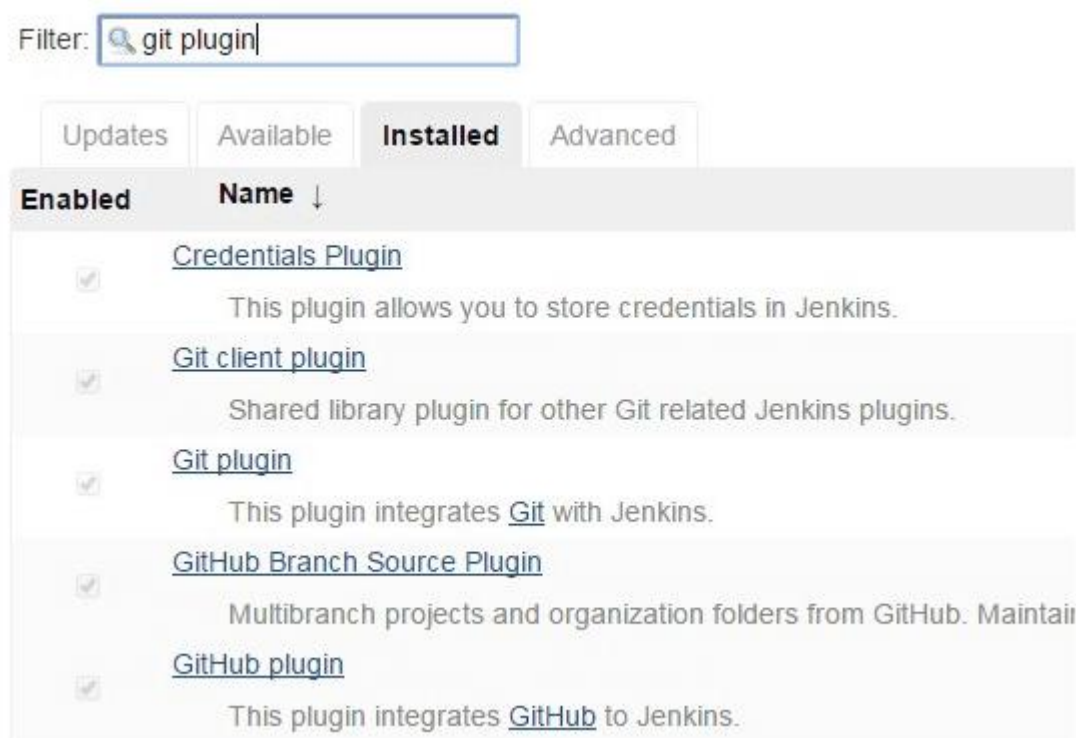


Następnie zmieniono jego widoczność na prywatny.

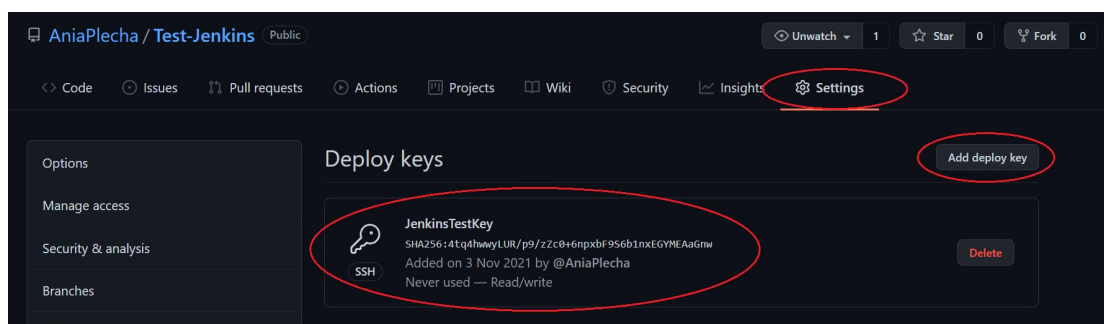
Następnie w repozytorium utworzono plik z testowym kodem źródłowym - „Hello World” w języku C.



Na maszynie wirtualnej z zainstalowanym Jenkinsem na początku odpowiednio skonfigurowano Jenkinsa - utworzono konto administratora, domyślny adres serwisu zmieniono na adres maszyny wirtualnej. Następnie zainstalowano wszystkie wymagane wtyczki:

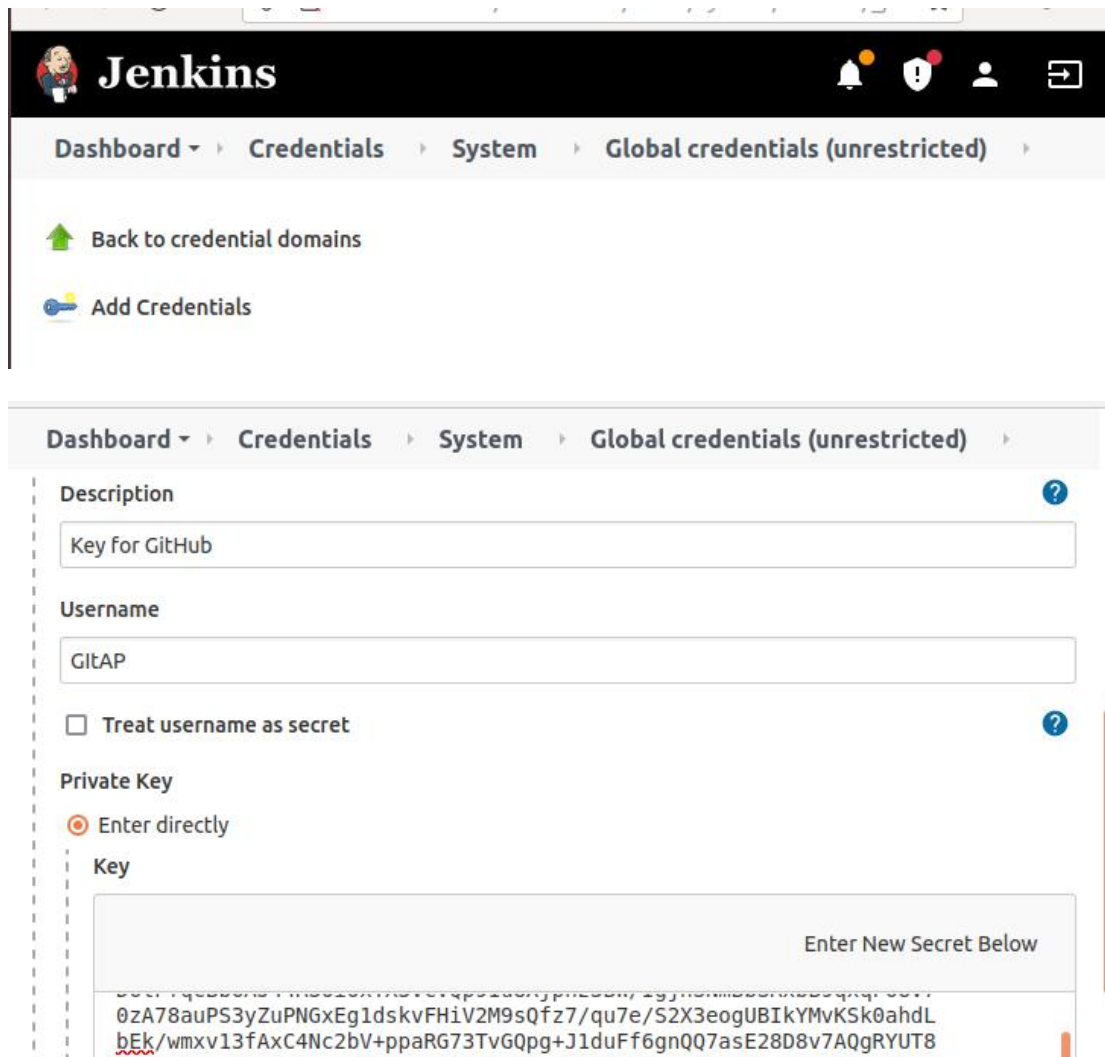


By dostęp do testowego repo był automatyczny, łączność powinna odbywać się za pomocą pary kluczy. Do łączności użyto kluczy wgenerowanych za pomocą polecenia „ssh-keygen -t rsa -b 4096 -f github_jd_rsa. Klucz publiczny został umieszczony na serwisie GitHub za pomocą funkcji *Add deploy key*.



Klucz prywatny z kolei został umieszczony na Jenkinsie w następujący sposób. Na Jenkins należało wejść kolejno w ścieżkę jak na screenie i za

pomocą opcji „Add credentials” utworzyć nowe uwierzytelnianie kluczem prywatnym.



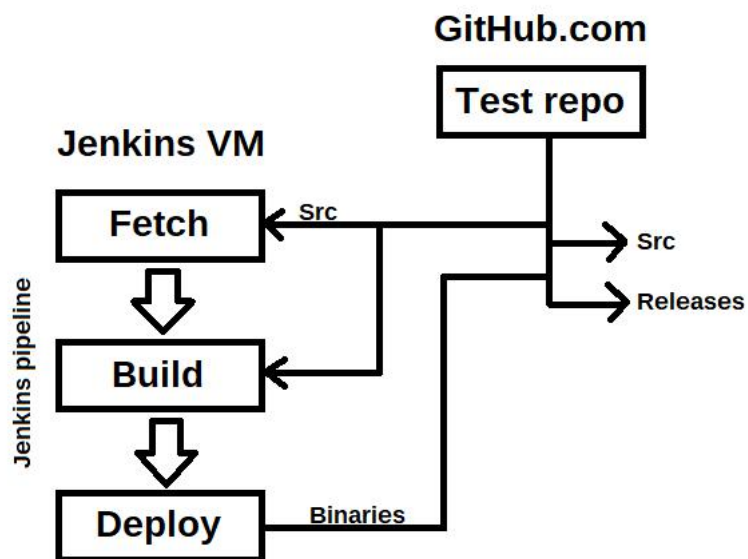
The screenshot shows the Jenkins web interface. At the top, the Jenkins logo and navigation links (Dashboard, Credentials, System, Global credentials (unrestricted)) are visible. Below the navigation bar, there are two buttons: 'Back to credential domains' and 'Add Credentials'. The 'Add Credentials' button is highlighted. Below this, the 'Add Credentials' form is shown. The form has a title bar with the same navigation links. The form fields are: 'Description' (with a value 'Key for GitHub'), 'Username' (with a value 'GitAP'), and 'Private Key' (with a radio button selected for 'Enter directly'). The 'Private Key' field is a large text area containing a long, base64-encoded string. A vertical orange bar is on the right side of the form.

W ten sposób utworzono automatyczny szyfrowany dostęp Jenkinsa do GitHub.

2. Kompilacja do postaci binarnej. Tryb potokowy (Jenkins pipeline)

Następnie kod źródłowy z GitHub powinien zostać skompilowany do postaci binarnej i umieszczony na nim powrotnie. Kroki potoku powinny wyglądać w sposób następujący:

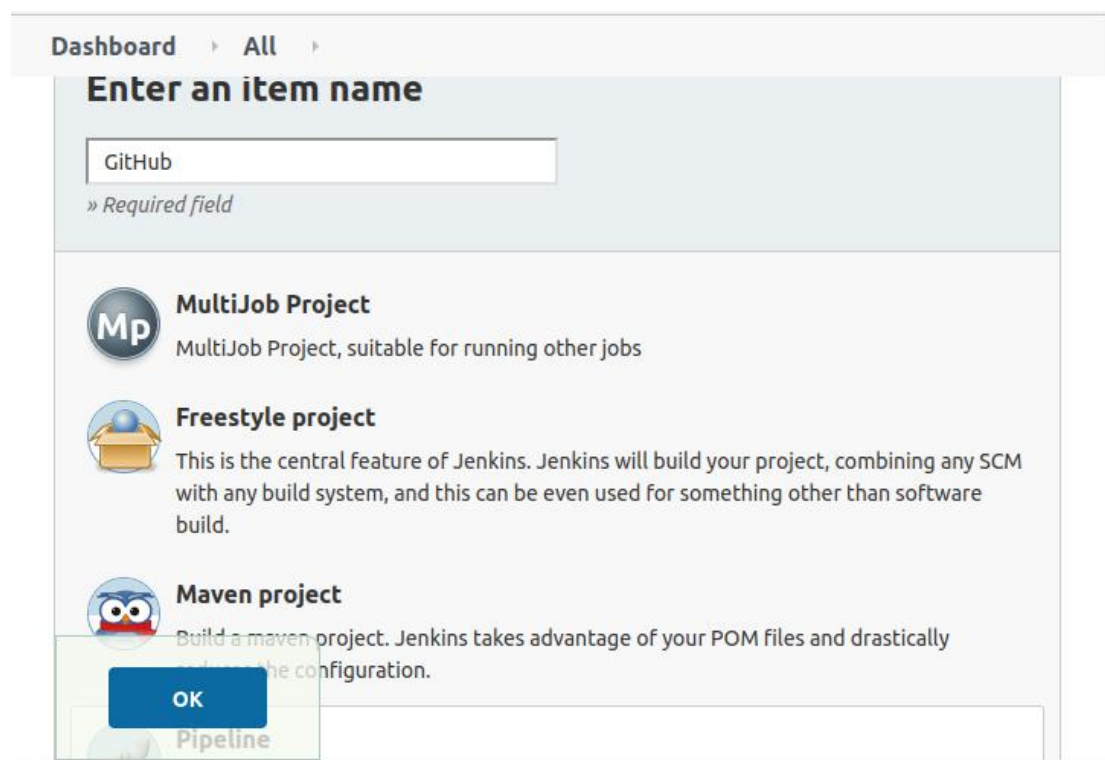
Pobranie kodu z repozytorium -> kompilacja -> testowe uruchomienie
-> zapisanie artefaktów na GitHub



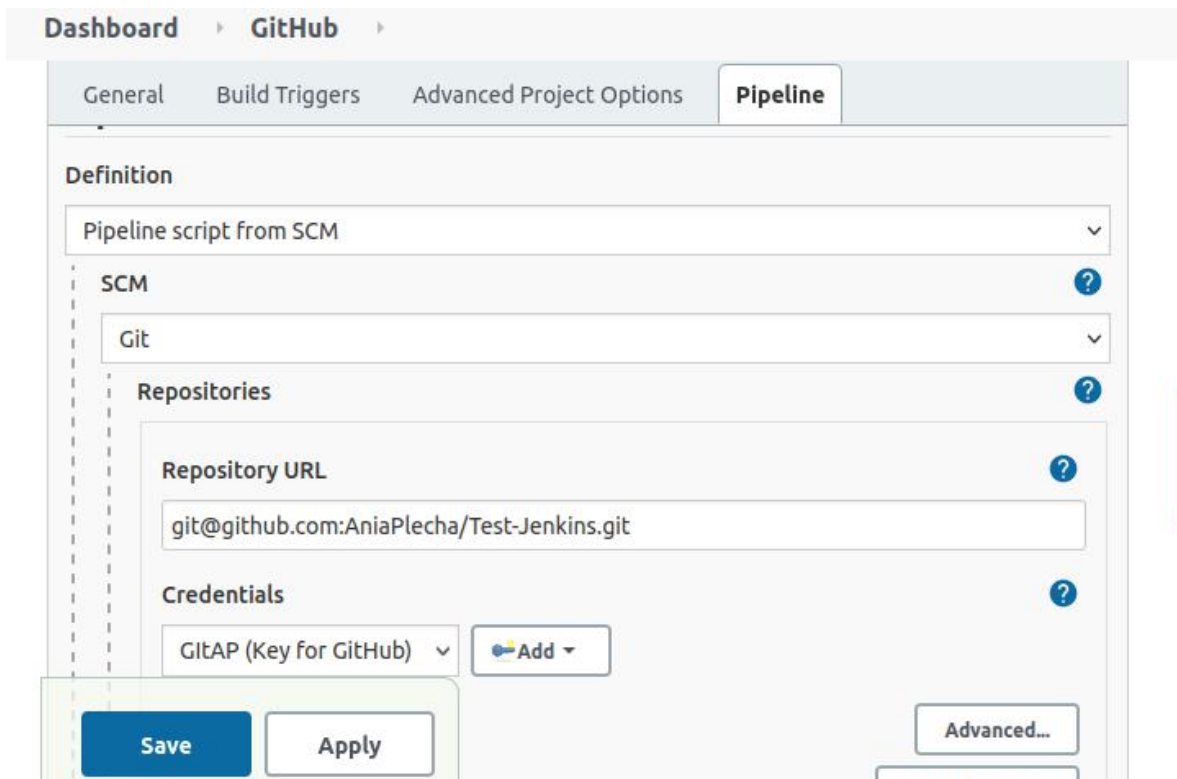
Rys 1. Schemat kolejnych kroków potoku.

Wykonano następujące kroki:

Utworzono nowy projekt w Jenkinsie w kategorii Pipeline



Następnie w edycji projektu, w zakładce Pipeline ustawiono pobieranie kodu z repozytorium testowego na GitHub i połączenie za pomocą zdefiniowanego wcześniej uwierzytelniania kluczem.



Dashboard > GitHub >

General Build Triggers Advanced Project Options **Pipeline**

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

git@github.com:AniaPlecha/Test-Jenkins.git

Credentials

GitAP (Key for GitHub) Add

Save Apply Advanced...



Additional Behaviours

Add

Script Path

helloworldind

☒ Lightweight checkout

Help Pipeline Syntax

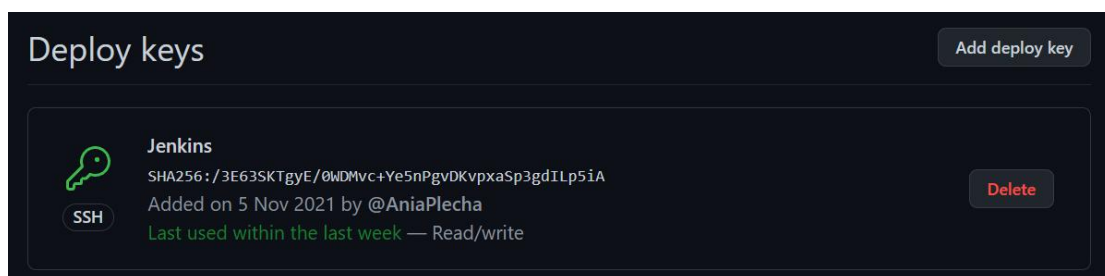
Po uruchomieniu projektu powinny wykonać się określone zadania w ramach pipeline, czego efekt powinien być widoczny w logach. Niestety na moment obecny ten etap kończy się poniższym błędem:

Console Output

```
Uruchomiono przez użytkownika Admin
hudson.plugins.git.GitException: Command "git fetch --tags --progress --prune -- origin +refs/heads
/master:refs/remotes/origin/master" returned status code 128:
stdout:
stderr: fatal: Couldn't find remote ref refs/heads/master

    at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandIn(CliGitAPIImpl.java:2681)
    at
org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandWithCredentials(CliGitAPIImpl.java:2102)
    at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.access$500(CliGitAPIImpl.java:86)
    at org.jenkinsci.plugins.gitclient.CliGitAPIImpl$1.execute(CliGitAPIImpl.java:624)
    at jenkins.plugins.git.GitSCMFileSystem$BuilderImpl.build(GitSCMFileSystem.java:366)
    at jenkins.scm.api.SCMFileSystem.of(SCMFileSystem.java:197)
    at jenkins.scm.api.SCMFileSystem.of(SCMFileSystem.java:173)
    at org.jenkinsci.plugins.workflow.cps.CpsScmFlowDefinition.create(CpsScmFlowDefinition.java:114)
    at org.jenkinsci.plugins.workflow.cps.CpsScmFlowDefinition.create(CpsScmFlowDefinition.java:68)
    at org.jenkinsci.plugins.workflow.job.WorkflowRun.run(WorkflowRun.java:310)
    at hudson.model.ResourceController.execute(ResourceController.java:99)
    at hudson.model.Executor.run(Executor.java:431)
Finished: FAILURE
```

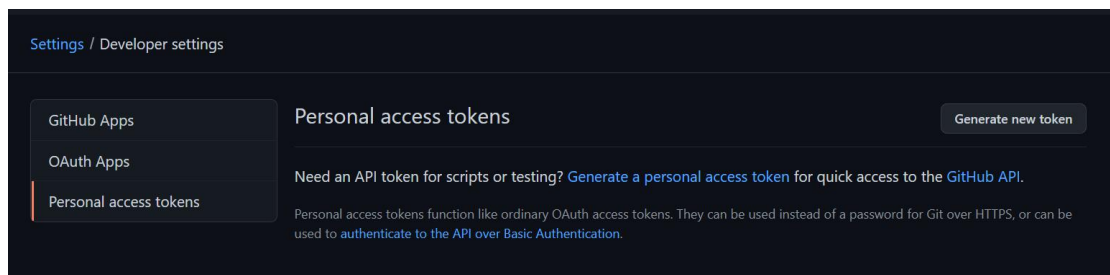
Łączność Jenkinsa z GitHub za pomocą klucza jest prawidłowa, ponieważ na GitHub widzimy, że klucz został użyty:



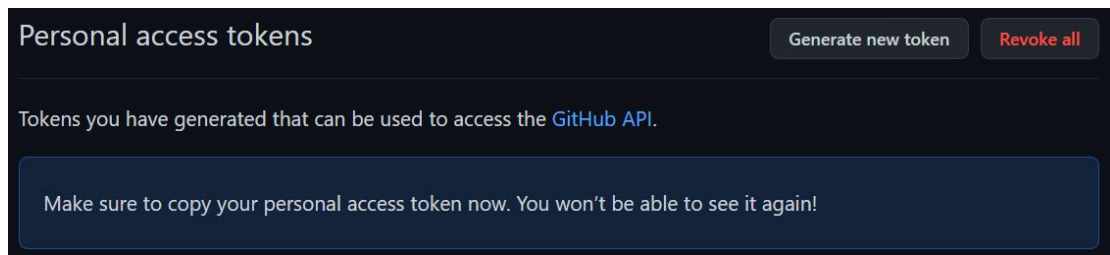
3. Publikacja artefaktów binarnych w testowym repozytorium GitHub

W tym punkcie należało dodatkowo ustawić token dostępowy do funkcji GitHub Releases.

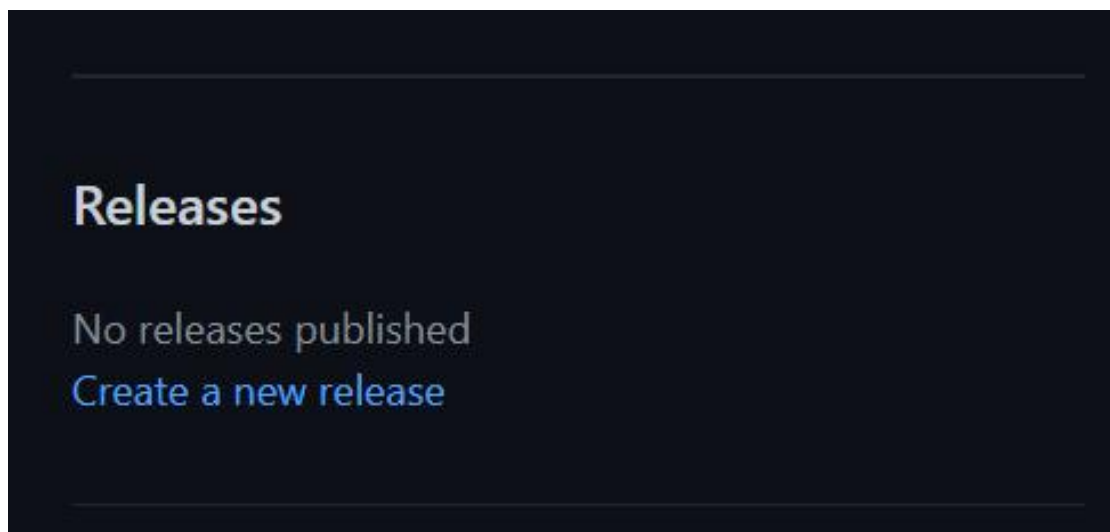
Tokeny utworzyć można w zakładce Settings -> Developer settings -> Personal Access Tokens -> Generate new token. Następnie należy ustawić mu uprawnienia do repozytorium.

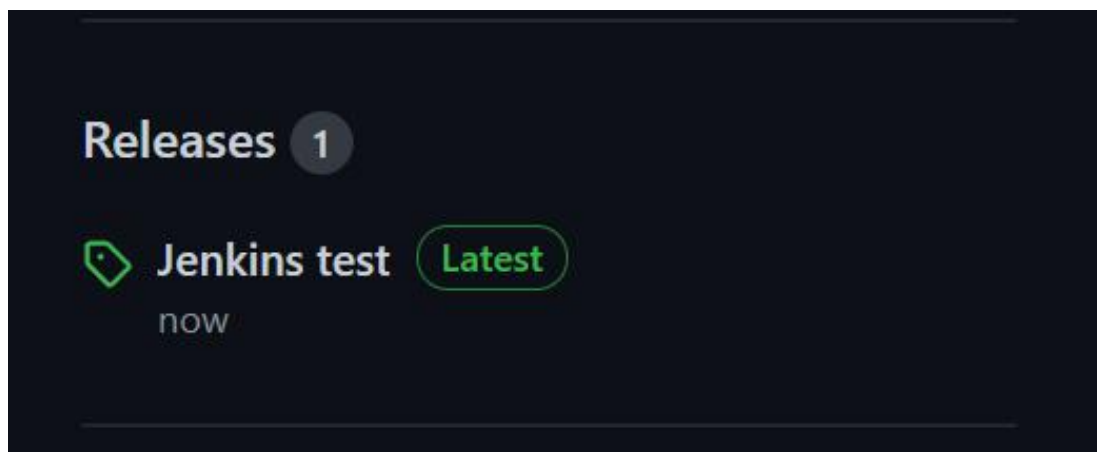
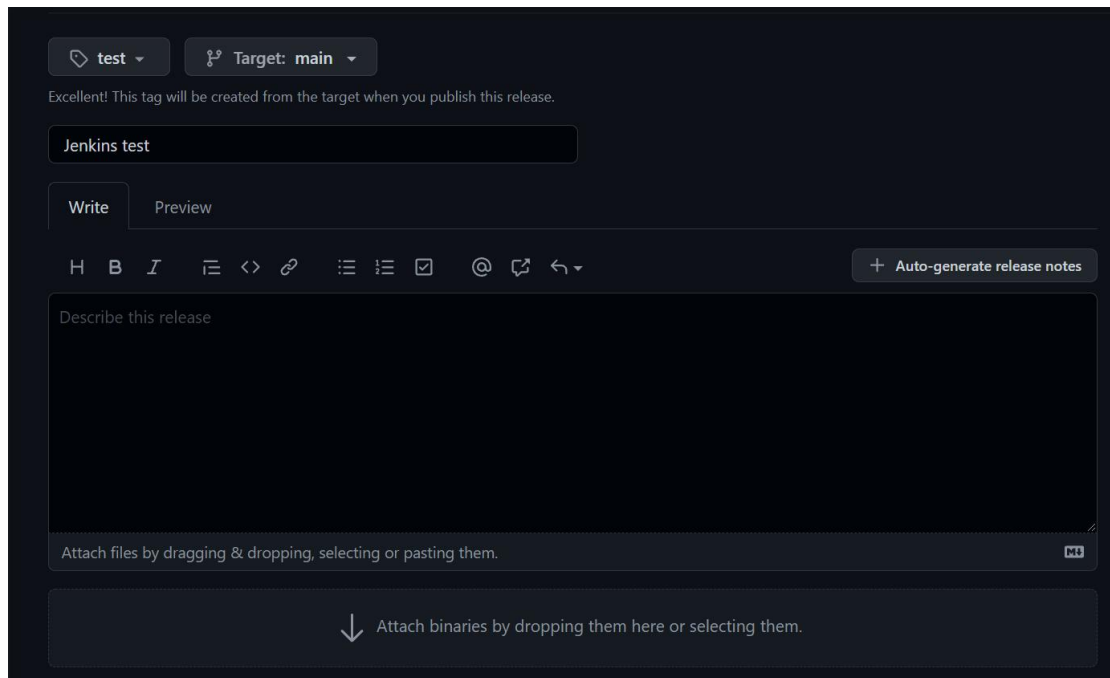


Token należy gdzieś zapisać, ponieważ będzie jedynie raz wyświetlony.



W repozytorium na GitHub utworzono Release





Następnie należało ustawić w Jenkinsie dostęp do repozytorium z wykorzystaniem tokena dostępowego.

W tym celu należało utworzyć nowy sposób uwierzytelniania poprzez opcję „Add Credentials”

Kind

Secret text

Scope

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID

gittoken

Description

Access Token for GitHub

A następnie ustawić ten sposób uwierzytelniania w zarządzaniu utworzonym wcześniej buildem Pipeline.

4. Źródła

1. <https://mohitgoyal.co/2017/02/27/configuring-ssh-authentication-between-github-and-jenkins/>
2. <https://medium.com/appgambit/ssh-authentication-between-github-and-jenkins-d873dd138dbo>
3. <https://www.blazemeter.com/blog/how-to-integrate-your-github-repository-to-your-jenkins-project/>
4. <https://stackoverflow.com/questions/6930147/git-pull-displays-fatal-couldnt-find-remote-ref-refs-heads-xxxx-and-hangs-up/6930399>
5. <https://github.com/jenkinsci/ghprb-plugin/issues/507>
6. <https://www.adoclib.com/blog/jenkins-fails-to-build-when-i-use-a-parameterised-branch-fatal-could-not-find-remote-ref-refs-heads-build-branch.html>
7. <https://stackoverflow.com/questions/61105368/how-to-use-github-personal-access-token-in-jenkins>
8. <https://docs.github.com/en/repositories/releasing-projects-on-github/managing-releases-in-a-repository#creating-a-release>