

## 1. Java instalacja i konfiguracja

Ściągamy i instalujemy Jave - download java

Najlepiej ostatnią wersję 8.162

(wersja 9 wprowadza modularyzację i nie wszystkie biblioteki są już dostosowane do wersji 9)

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

### Java SE Development Kit 8u162

Zaznaczamy licencję: Accept License Agreement

I pobieramy odpowiednią wersję;

Po ściągnięciu uruchamiamy i instalujemy ściągnięty pakiet

I instalujemy według własnego uznania, możemy użyć defaultowych ustawień i ścieżek.

Java SE Development Kit 8u161

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☐ Accept License Agreement ☐ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.92 MB	<a href="#">jdk-8u161-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	74.88 MB	<a href="#">jdk-8u161-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	168.96 MB	<a href="#">jdk-8u161-linux-i586.rpm</a>
Linux x86	183.76 MB	<a href="#">jdk-8u161-linux-i586.tar.gz</a>
Linux x64	166.09 MB	<a href="#">jdk-8u161-linux-x64.rpm</a>
Linux x64	180.97 MB	<a href="#">jdk-8u161-linux-x64.tar.gz</a>
macOS	247.12 MB	<a href="#">jdk-8u161-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	139.99 MB	<a href="#">jdk-8u161-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	99.29 MB	<a href="#">jdk-8u161-solaris-sparcv9.tar.gz</a>
Solaris x64	140.57 MB	<a href="#">jdk-8u161-solaris-x64.tar.Z</a>
Solaris x64	97.02 MB	<a href="#">jdk-8u161-solaris-x64.tar.gz</a>
Windows x86	198.54 MB	<a href="#">jdk-8u161-windows-i586.exe</a>
Windows x64	206.51 MB	<a href="#">jdk-8u161-windows-x64.exe</a>

Java SE Development Kit 8u162

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☐ Accept License Agreement ☐ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.93 MB	<a href="#">jdk-8u162-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	74.88 MB	<a href="#">jdk-8u162-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	169.01 MB	<a href="#">jdk-8u162-linux-i586.rpm</a>
Linux x86	183.81 MB	<a href="#">jdk-8u162-linux-i586.tar.gz</a>
Linux x64	166.13 MB	<a href="#">jdk-8u162-linux-x64.rpm</a>
Linux x64	181.02 MB	<a href="#">jdk-8u162-linux-x64.tar.gz</a>
macOS	247.12 MB	<a href="#">jdk-8u162-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	139.98 MB	<a href="#">jdk-8u162-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	99.3 MB	<a href="#">jdk-8u162-solaris-sparcv9.tar.gz</a>
Solaris x64	140.68 MB	<a href="#">jdk-8u162-solaris-x64.tar.Z</a>
Solaris x64	97.03 MB	<a href="#">jdk-8u162-solaris-x64.tar.gz</a>
Windows x86	198.57 MB	<a href="#">jdk-8u162-windows-i586.exe</a>
Windows x64	206.76 MB	<a href="#">jdk-8u162-windows-x64.exe</a>

Java SE Development Kit 8u161 Demos and Samples Downloads

You must accept the [Oracle BSD License](#) to download this software.

☐ Accept License Agreement ☐ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	9.93 MB	<a href="#">jdk-8u161-linux-arm32-vfp-hflt-demos.tar.gz</a>
Linux ARM 64 Hard Float ABI	9.96 MB	<a href="#">jdk-8u161-linux-arm64-vfp-hflt-demos.tar.gz</a>
Linux x86	52.69 MB	<a href="#">jdk-8u161-linux-i586-demos.rpm</a>

Developer Training

[Tutorials](#)

[Java.com](#)

## 1.a Ustawiamy zmienne środowiskowe

Panel Sterowania > System i zabezpieczenia > System > Zaawansowane ustawienia systemu > Zmienne Srodowiskowe

<http://www.wiedzanaplus.pl/systemy-operacyjne/39-windows/95-zmienne-srodowiskowe-win10.html>

Dodajemy nowe dwie zmienne JAVA\_HOME oraz JRE\_HOME

JAVA\_HOME = C:\Program Files\Java\jdk1.8.0\_162

-> wskazujemy ścieżkę do głównego katalogu zainstalowanej przez nas javy.  
(Najczęściej jest to c:\Program Files\java\

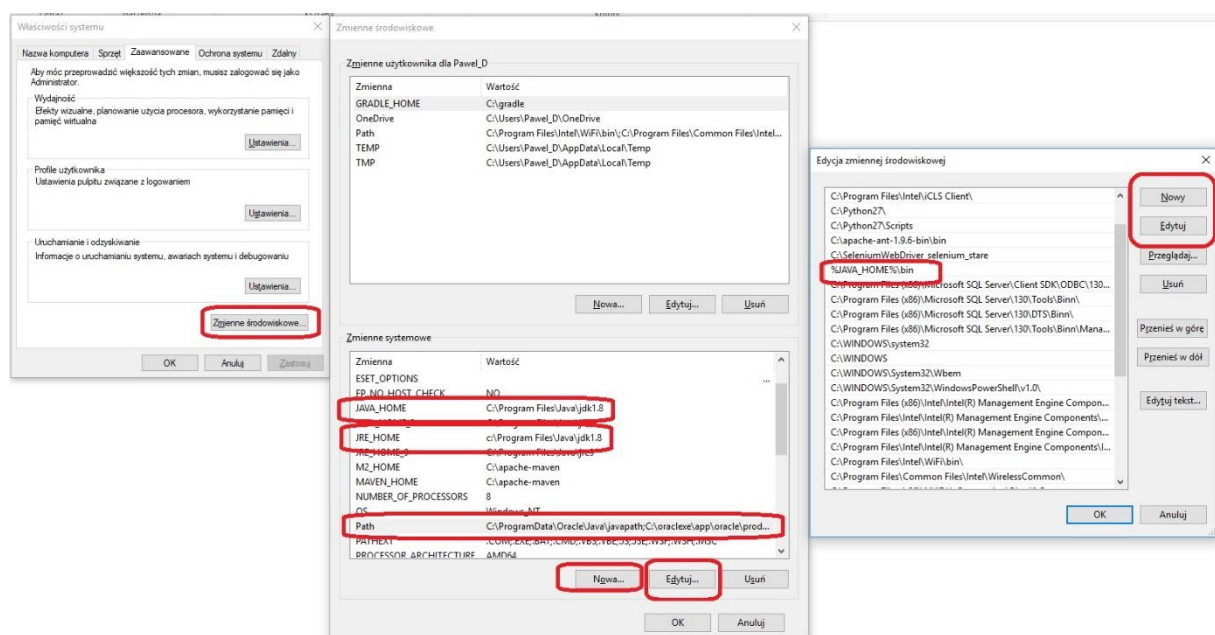
Ustawiamy również ścieżkę do JRE który również się zainstalowało.

JRE\_HOME = C:\Program Files\Java\jre1.8.0\_162

A następnie w PATH dodajemy ścieżkę do katalogu bin naszego instalowanego pakietu javy.

PATH = %JAVA\_HOME%\bin

Ja zawsze ustawiam w Path – ie systemowym , a jeżeli nie mamy dostępu to zostaje nam dodać do Path użytkownika



Otwieramy konsolę cmd

I sprawdzamy na konsoli

Java -version

Powinno nam wyświetlić aktualną wersję java 1.8.0\_162

”

java version "1.8.0\_162"

Java(TM) SE Runtime Environment (build 1.8.0\_152-b16)

Java HotSpot(TM) 64-Bit Server VM (build 25.152-b16, mixed mode)

”

oraz

javac -version

Również powinno nam wyświetlić aktualną wersję

”

Javac 1.8.0\_162

”

Trzeba pamiętać o restarcie konsoli

Czym jest Java Development Kit -> jest to pakiet narzędzi wspierających tworzenie i rozwój oprogramowania z wykorzystaniem języka Java, zawiera w sobie JRE (Java Runtime Environment)

JRE – jest środowiskiem uruchamiającym zawierającym w sobie m.in. JVM (Java Virtual Machine)

## 2. Maven instalacja i konfiguracja

Ściągamy mavena i instalujemy

<http://maven.apache.org/download.cgi>

- download Maven -> Files -> Binary zip archive

rozpakować i wrzucić cały rozpakowany katalog gdzieś na dysk

ja polecam bezpośrednio

c:\apache-maven-3.5.2-bin

Tworzymy nowe zmienne środowiskowe, tak samo jak przy instalacji javy

M2\_HOME = c:\apache-maven-3.5.2-bin\apache-maven-3.5.2 - główny katalog mavena

MAVEN\_HOME = c:\apache-maven-3.5.2-bin\apache-maven-3.5.2 - główny katalog mavena

Tutaj podaje przykłady bezpośrednio bez ingerencji w strukturę katalogową ale warto wyciągnąć katalog apache-maven-3.5.2 bezpośrednio na c:\ aby skrócić ścieżkę o jeden poziom

I dodajemy nowy wpis do zmiennej systemowej PATH

PATH = %MAVEN\_HOME%\bin -> ze skierowaniem bezpośrednio do katalogu bin

Otwieramy konsolę cmd

Sprawdzamy: mvn -v

Wynik powinien zwrócić podobny wynik

”

Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T17:41:47+01:00)

Maven home: C:\apache-maven

Java version: 1.8.0\_152, vendor: Oracle Corporation

Java home: C:\Program Files\Java\jdk1.8\jre

Default locale: pl\_PL, platform encoding: Cp1250

OS name: "windows 10", version: "10.0", arch: "amd64", family: "dos"

”

Tutaj link do tutorialu

<http://cezarywalenciuk.pl/blog/programing/post/jak-zainstalowac-maven-w-windows>

### **3. Git instalacja i konfiguracja**

Ściągamy gita i instalujemy

<https://git-scm.com/downloads>

Uruchamiamy i instalujemy wszystko defaultowo

instaluje się do

c:\Program Files\Git

Zmienne środowiskowe powinny dodać się automatycznie

Należy sprawdzić czy wpis się pojawił w zmiennej PATH systemowej

Path= C:\Program Files\Git\cmd

Uruchamiamy konsolę cmd i sprawdzamy poprawność instalacji

git version

Powinien podać aktualną wersję gita

”

git version 2.16.0.windows.2

”

Uruchamiamy konsolę i ustawiamy

Email i użytkownika

```
git config --global user.name „nazwa_użytkownika”
```

```
git config --global user.email „email_użytkownika”
```

git config -global -l -> komenda do sprawdzenia ustawień które właśnie wprowadziliśmy

#### **4. IntelliJ Instalacja i konfiguracja**

Ściągamy i instalujemy IntelliJ IDE

Wersja community jest darmowa ale uboższa , wersja ultimate jest wersja pełną ale płatną, wersja ultimate jest darmowa przez pierwsze 30 dni od instalacji.

<https://www.jetbrains.com/idea/download/#section=windows> - download IntelliJ

IntelliJ IDE instalujemy defaultowo nic nie musimy ustawiać po drodze

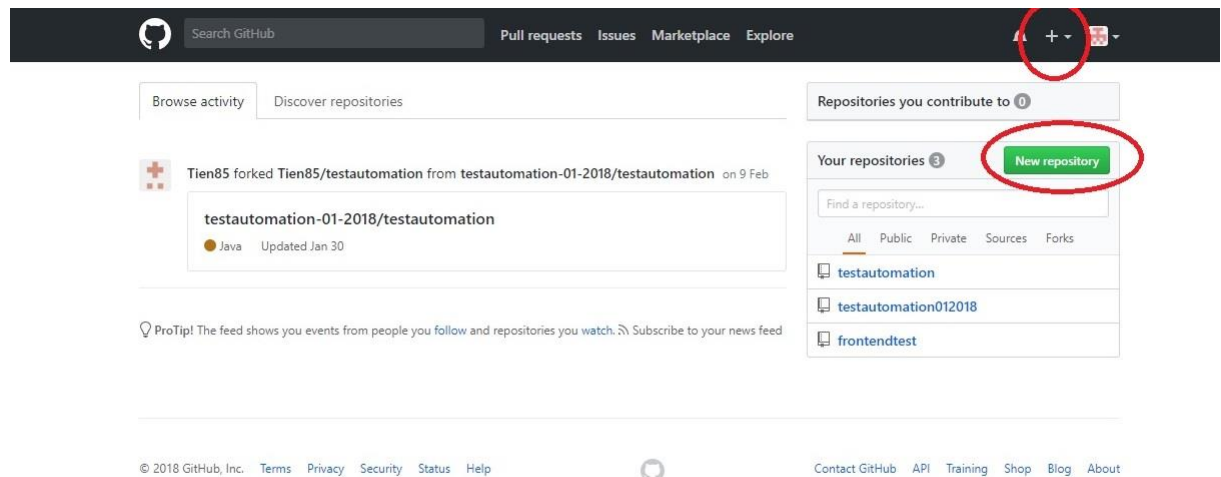
Next Featured plugins -> Start IntelliJ IDE

#### **Uruchomienie i skonfigurowanie nowego projektu w IntelliJ IDE**

Projekt możemy uruchomić w intellij albo utworzyć na repozytorium zdalnym np. github i sklonować (pobrać).

1. Pierwsza opcja klonowanie projektu

Stwórz nowy projekt na githubie poprzez „+” New Repository



Repository name: moja\_nazwa

Na potrzeby szkolenia zostawiamy public.

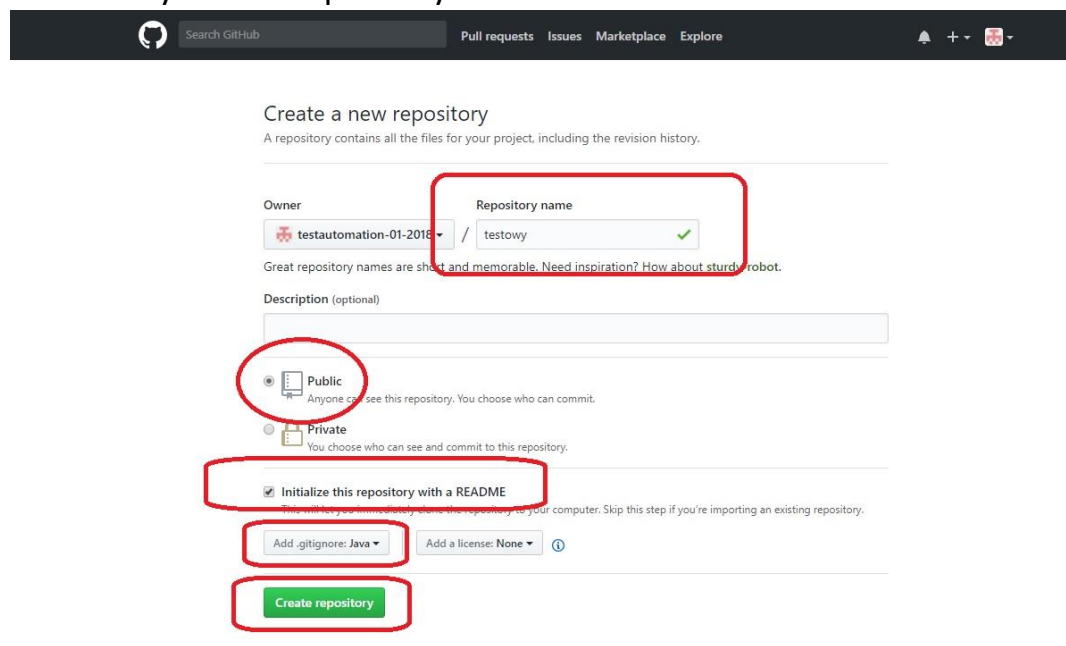
(Ale należy mieć na uwadze, iż projekty wykonywane zawodowo są zazwyczaj tajne i dostępne tylko dla uprawnionych użytkowników.)

Zaznaczamy „Initialize this repository with a README”

Oraz

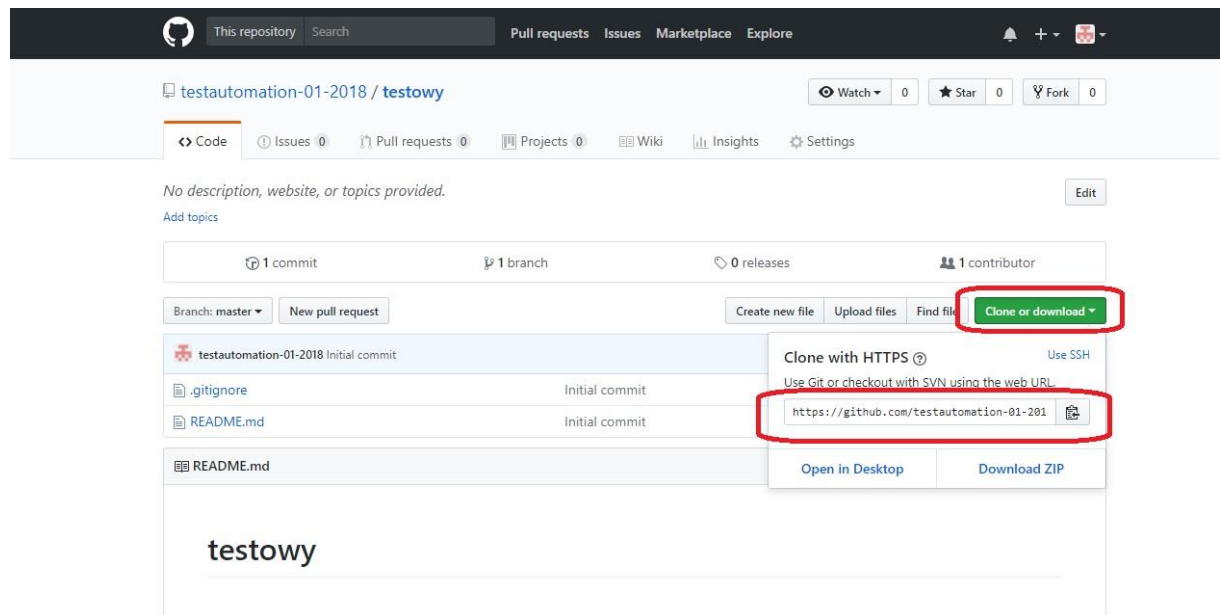
Add .gitignore: Java - ustawiamy

I wciskamy Create repository



Następnie klonujemy nasz projekt

Clone or download

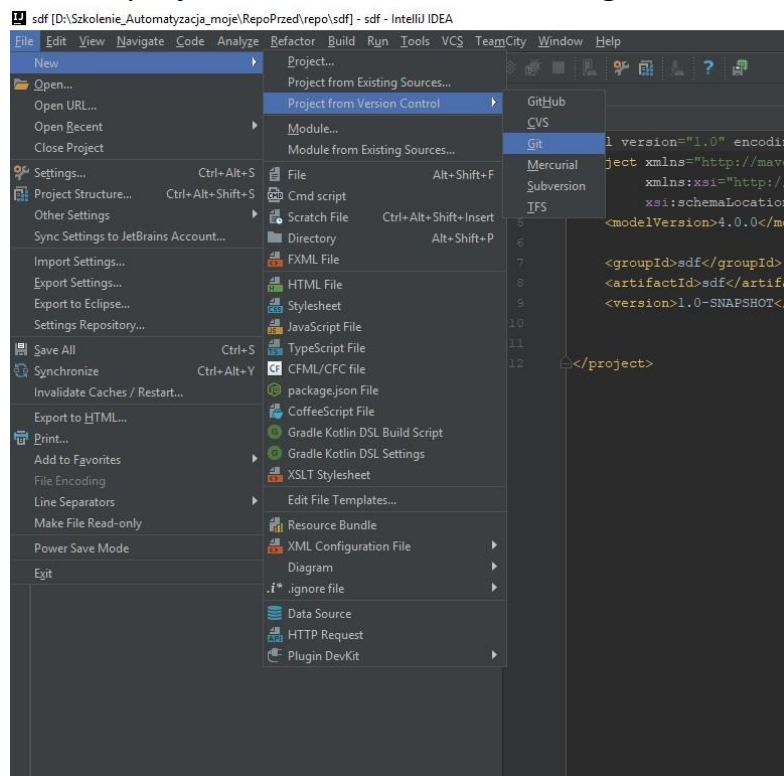


I kopiujemy link do repozytorium

<https://github.com/testautomation-01-2018/testowe.git>

Następnie w IntelliJ wybierz opcje

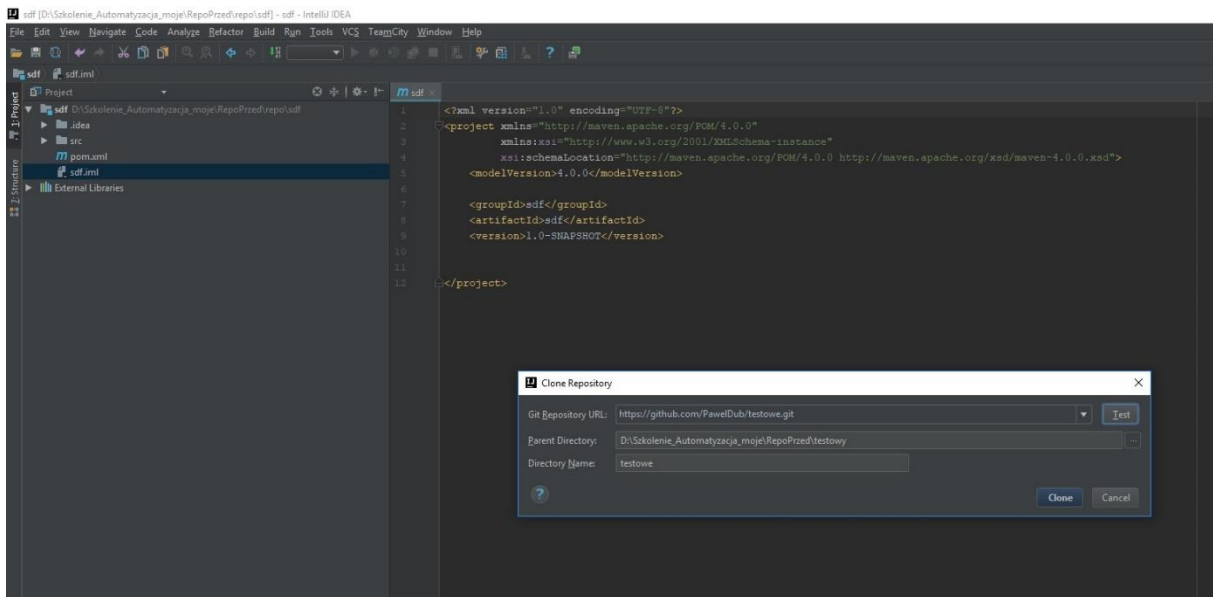
New -> project from version control -> git





I wskazujemy ścieżkę

<https://github.com/testautomation-01-2018/testowe.git>



Wskazujemy lokalizację dla projektu na dysku lokalnym oraz nazwę lokalną  
Możemy wykonać test połączenia

## 2. Druga opcja to stworzenie projektu i wykonanie share projekt on github

Tworzenie nowego projektu jeżeli nie mieliśmy wcześniej żadnej wersji to od razu po instalacji pojawi się okno z wyborem jakiego sposobu chcemy wykorzystać do utworzenia nowego projektu.

W przypadku tworzenia nowego projektu z poziomu uruchomionego IntelliJ wybieramy

Create New Projekt

Następnie

Na oknie: New Project

Projekt SDK -> wskazujemy gdzie mamy java zainstalowaną i wskazujemy główny katalog z jdk -> ten sam główny katalog javy który wskazaliśmy w zmiennej JAVA\_HOME

(C:\Program Files\Java\jdk1.8.0\_162)

Jeżeli chcemy aby projekt zarządzany był przez Mavena to na bocznej listwie wybieramy jako projekt Maven-owy

Nie zaznaczamy żadnych archetypów

Next

Następne okno to opis

groupId – unikatowy identyfikator projektu „com.jsystems.selenium, całego projektu

artifactId – nazwa dla pliku jar bez wersji jsystems.selenium , to pojedynczy moduł całego projektu

Version - możemy zostawić defaultową

GroupId: com.jsystems

ArtifactId: test-automation

Version: 1.0-SNAPSHOT

Next

Następne okno to nazwa projektu , defaultowo jest to nazwa pobrana z groupId

Oraz wskazujemy lokalizację na dysku lokalnym

Finish

Uruchomi się na pliku pom

Poczekajmy aż się całość importów wykona

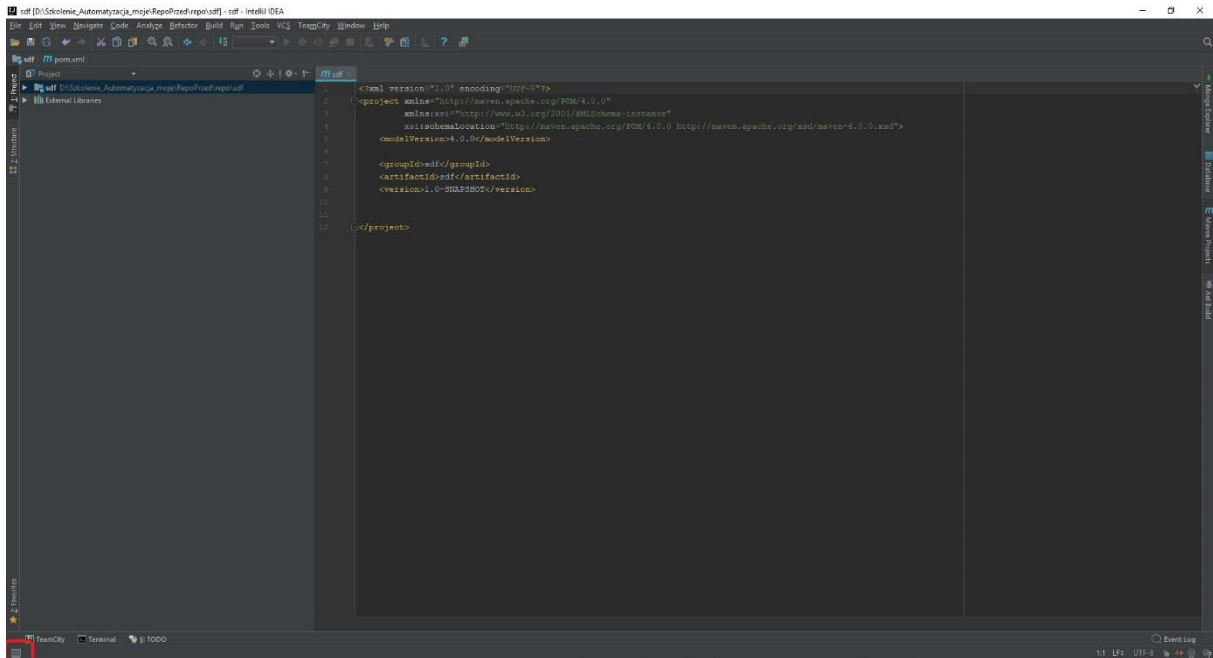
U dołu w prawym rogu powinien wyskoczyć mały popup

Maven project need to be imported

I klikamy Enable Auto-Import

Jeżeli nie włączymy auto-importu to będziemy musieli pamiętać aby po każdej zmianie wykonać odświeżenie w module maven

W lewym dolnym rogu jest ikona która włącza i wyłącza szybki dostęp do modułów w tym mavena.



Projekt otwiera się na pliku pom.xml

Wchodzimy do ustawień

File ->

Projekt Struktura :

-Project – ustawiamy SDK i Projekt language

- Module – ustawiamy source i moduły:

Source

Ustawiamy

Language level na 8.0 – java

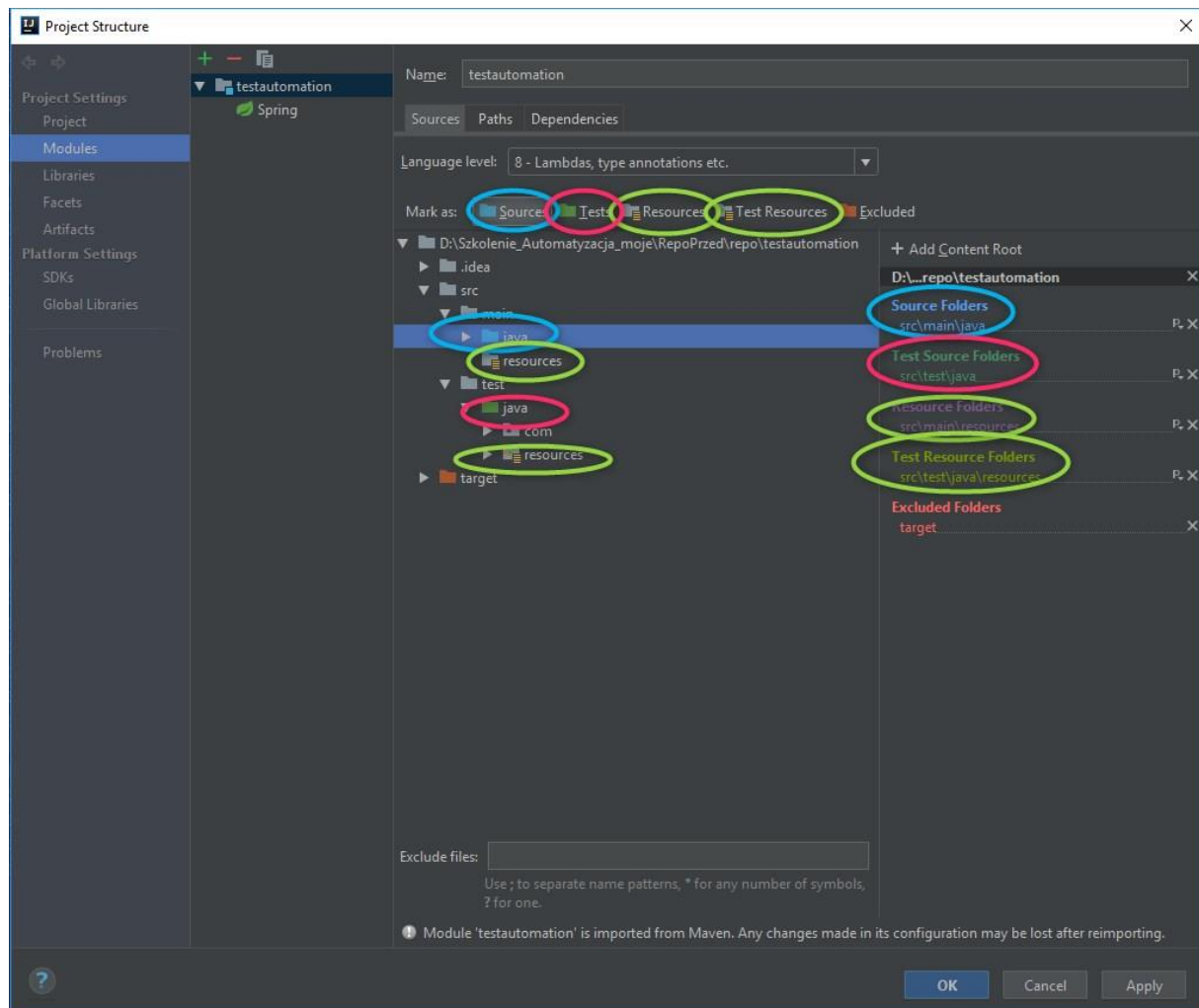
Oraz struktura

Dla Main -> java : Source,

Test -> java : Test,

Resources : Resources

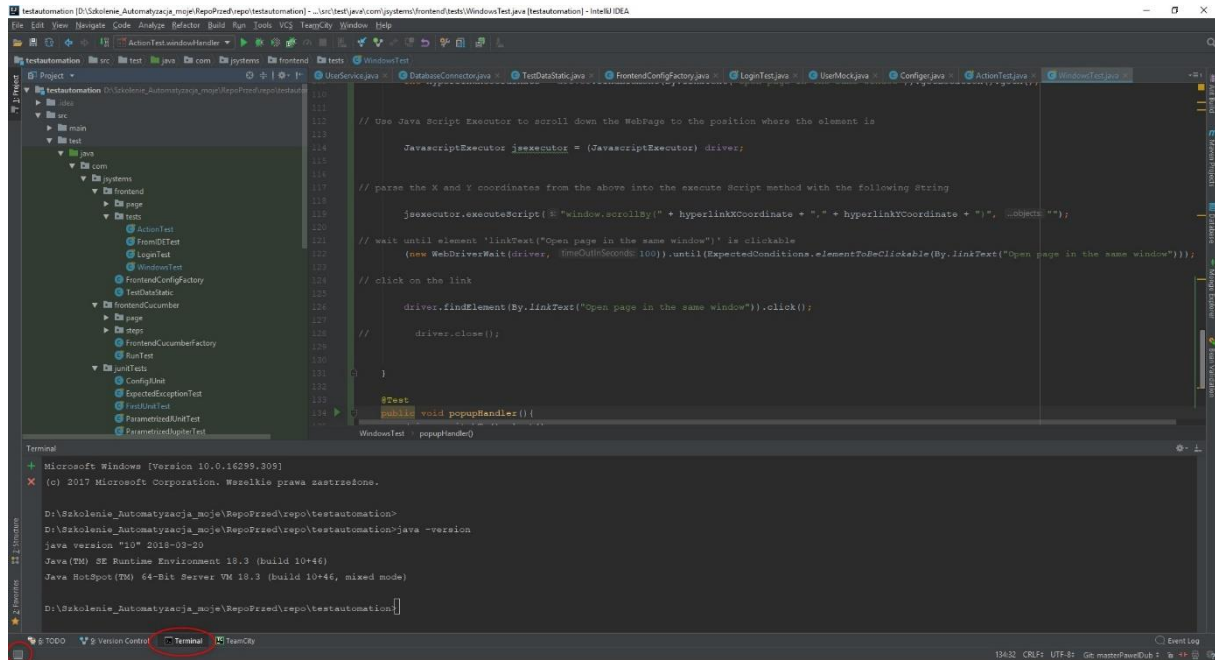
test -> resources : Test Resources



Otwieramy konsolę w IntelliJ-u

ALT + F12 uruchomienie konsoli

Sprawdzamy wersje Javy i Mavena tak samo jak przy instalacji



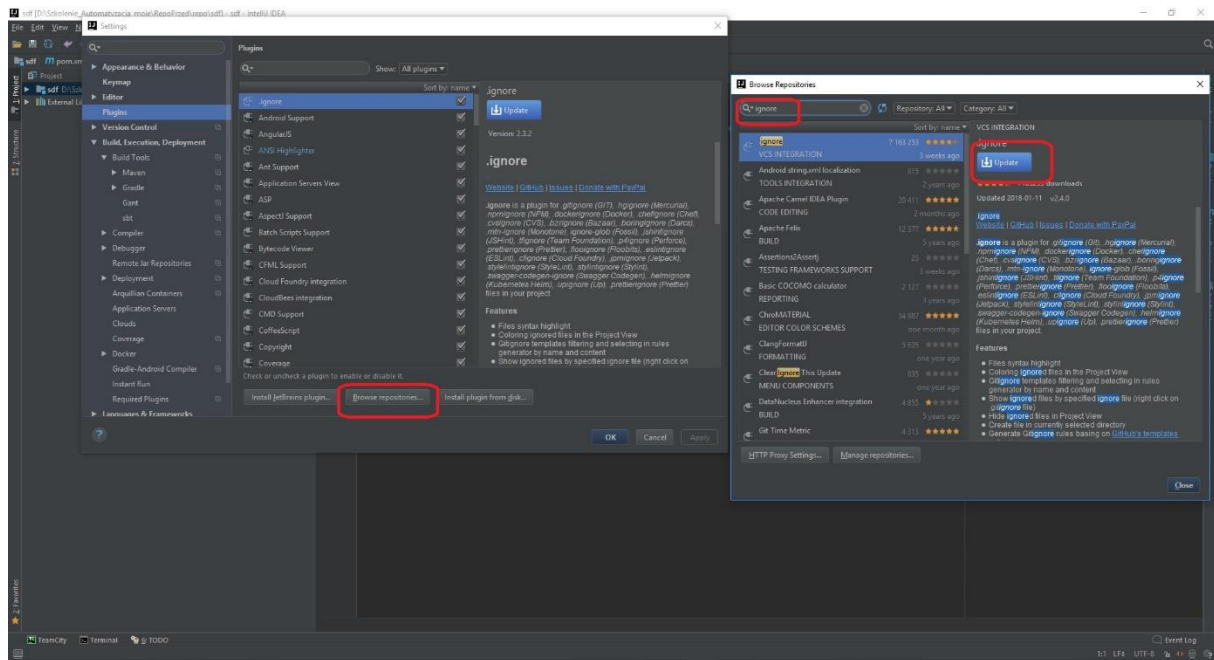
Musimy pociągnąć pluginy

Zaczynamy od ignore

Settings

Plugins

Browse Repositories



Dociągamy plugin .ignore

Wyszukujemy go poprzez Browse Repository

Wpisujemy ignore w polu wyszukiwania

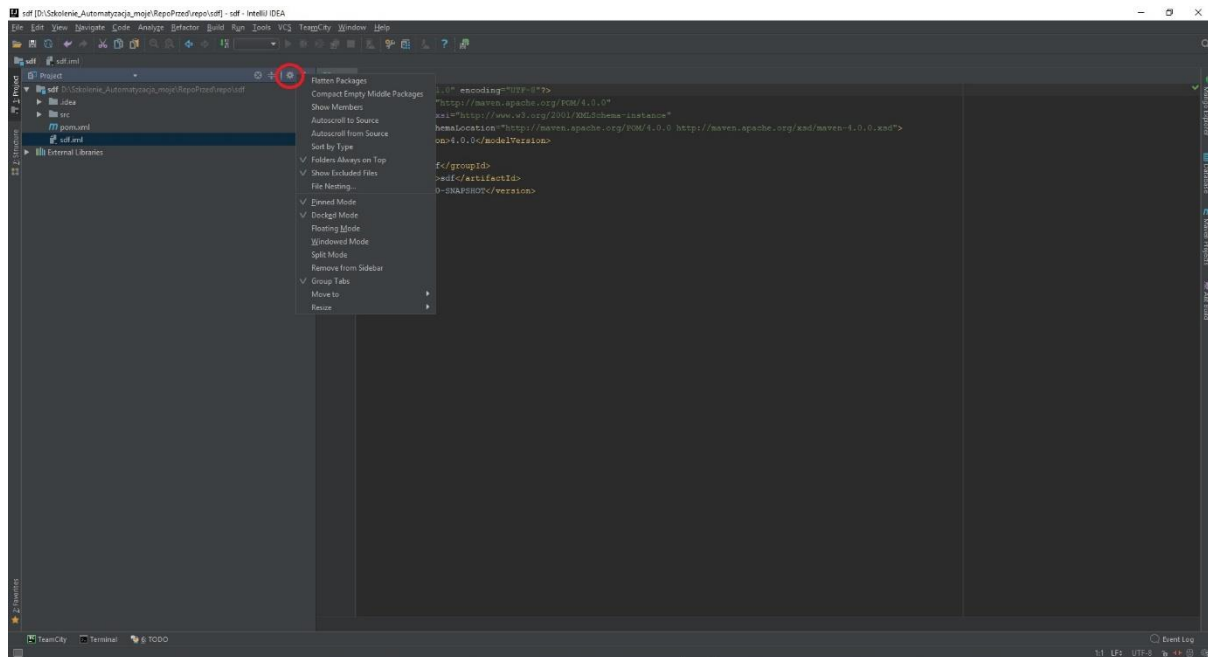
Wskazujemy .ignore i klikamy „Install”

Settings

Keymap – do ustawiania skrótów klawiaturowych

Kilka ustawień na samym Projekcie

Klikamy ikonę ustawień na „Project” i odznaczamy Flatten Packages oraz Hide Empty Middle Packages

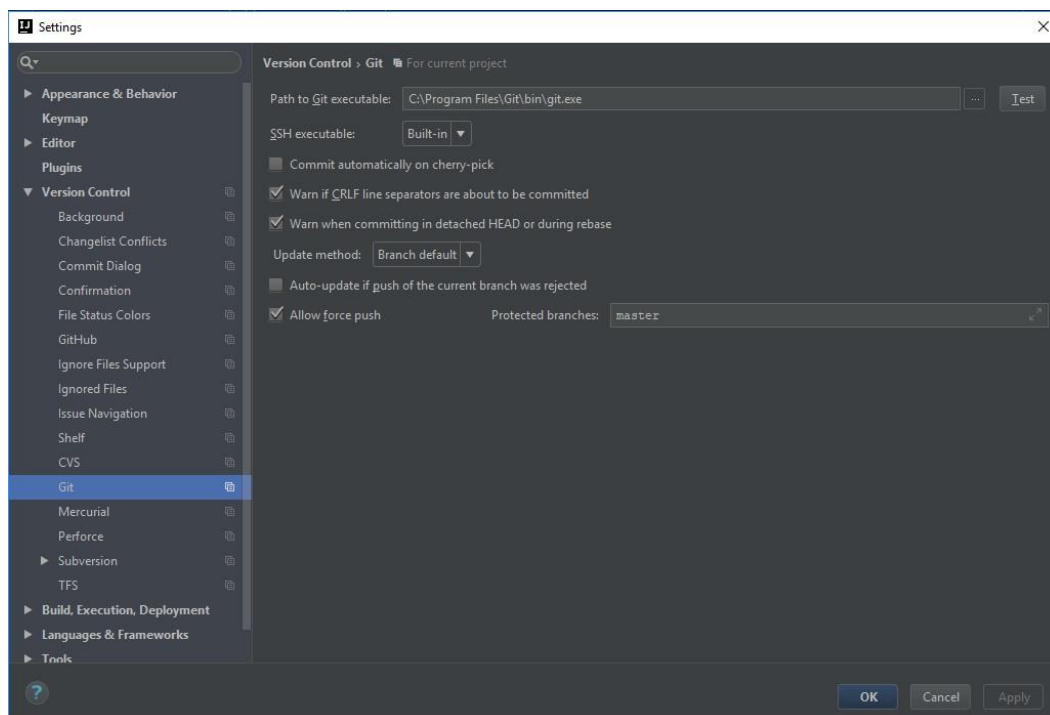


Ok.

Ścieżkę do gita ustawiamy w

Settings

➔ Version Control -> Git



Po założeniu projektu wskazane jest w pierwszej kolejności ustawienie kontroli wersji w naszym przypadku z użyciem Git-a

Otwieramy terminal (powinniśmy się znaleźć w ścieżce na głównym projekcie)

Terminal na głównym projekcie i robimy inicjalizację kontroli wersji

Git init

Plugin ignore mamy już ściągnięty jak nie, to dociągamy

Musimy ustawić plik .gitignore aby nam nie dodawał do repozytorium śmieci (plików których nie chcemy albo nie musimy umieszczać w repozytorium zdalnym)

Tworzymy plik .gitignore

PKM – Prawy klawisz myszy

Na projekcie PKM -> New -> .ignore file -> .gitignore file (Git) -> ustawiamy java -> Generate -> Yes

Dodajemy pliki oraz katalogi do .gitignore

Klikamy na testautomation.iml -> PKM Add to .gitignore file

.idea -> PKM Add to .gitignore file

.gitignore -> PKM -> Add to .gitignore

Dodajemy pliki do kontroli wersji

Wskazujemy plik

Pom.xml -> PKM -> Git -> +add

Przy każdym nowo założonym pliku pojawi się okno z pytaniem czy dodać plik do kontroli wersji -> oczywiście się zgadzamy

Robimy pierwszy commit

Używamy skrótu klawiszowego i narzędzia z IntelliJ

Ctrl+k



Wpisujemy message dla commita

Przeglądamy pliki które dodajemy

Dalej

-> Commit

Lub przy użyciu konsoli

git commit -m „wiadomość”

Następnie musimy wysłać projekt na githuba.

Jeżeli mamy projekt sklonowany z githuba to wystarczy

W konsoli wpisać

Git push

Jeżeli założyliśmy projekt w intelliJ to musimy wysłać nasz nowy projekt na githuba

VCS -> Import into Version Control -> Share project on github

Host: github.com ,

Auth Type: Password

Login: [podajemy login do githuba](#)

Password: tutaj hasło do githuba

➔ Login

New repository name: nadajemy\_nazwe (testowy)

Remote name: nadajemy\_nazwe (można obydwie nazwy pozostawić takie same) (testowy)

➔ Share

I projekt mamy wysłany na githuba

Następnie warto sprawdzić czy zmiany/projekt na githubie się pojawił.

Możemy zmienić login i hasło użytkownika githuba

Settings > Version Control > Github – ustawiamy użytkownika githuba / jeżeli tego nie ustawimy to zapyta się o to przy pierwszym ustawieniu

Występują problemy z poświadczeniem przez windows przy zmianie usera na githubie wtedy musimy ustawić nowego użytkownika do poświadczenia

[ Panel sterowania\Konta użytkowników\Menedżer poświadczeń ] - ustawienia na Windowsie do poświadczenia usera po protokole https zamiast ssh

Sprawdzamy ścieżkę projektów zdalnych oraz możemy zarządzać ścieżkami do projektów zdalnych

VCS -> Git -> Remotes -

Powinna być ustawiona

Name: testautomation

URL: <https://github.com/testautomation-01-2018/testowy>

<https://try.github.io/levels/1/challenges/1> - do potrenowania gita

---

## 5. TeamCity instalacja i konfiguracja

Pobieramy z jetbrainsa

<https://www.jetbrains.com/teamcity/download/>

Uruchamiamy instalację

Next ...

License Agreement -> I Agree ...

Najlepiej w domyślnej lokalizacji (ale możemy wybrać własną)

Next ...

Wybór komponentów pozostawiamy bez zmian (domyślnie)

Next ...

TeamCity server port – 9191 - (port ustawiamy dowolny taki aby się nie mieszał z innymi rzeczami)

Next ...

Configure Build Agent Properties – zatwierdzamy bez zmian

Save ...

Potwierdzamy zapis

Ok ...

Select Service Account for Server – zaznaczamy :

Run TeamCity Server under the System account (możemy na swoje potrzeby ustawić ...under a user account , według własnego uznania)

Next ...

TeamCity First Start

Okno pierwszego uruchamiania – zostawiamy bez zmian

Proceed ...

Select the database type: HSQLDB (Możemy sobie ustawić inną według własnych potrzeb)

Proceed

Na dole licencja zgadzamy się

i

Continue ...

Create Administrator account

User

Password

Confirm password

Create Account

(

przy drugiej instalacji jeżeli już mieliśmy zainstalowane TeamCity , pojawi nam się tylko zaznaczanie Run dla Servera i Agenta

Select Service Account for Agent – zaznaczamy :

Run TeamCity Agent under the System account (możemy na swoje potrzeby ustawić ...under a user account , według własnego uznania)

Next ...

Setup Services – zostawiamy zaznaczone :

Start Build Agent service

Start TeamCity Server service

Next ...

Completing the JEtBrains teamCity ... - zostawiamy zaznaczone:

Open TeamCity Web UI after Setup is completed

Finish ...

)

Team City uruchomi nam się na przeglądarce domyślnej

Na adresie (który podaliśmy):

localhost:9191

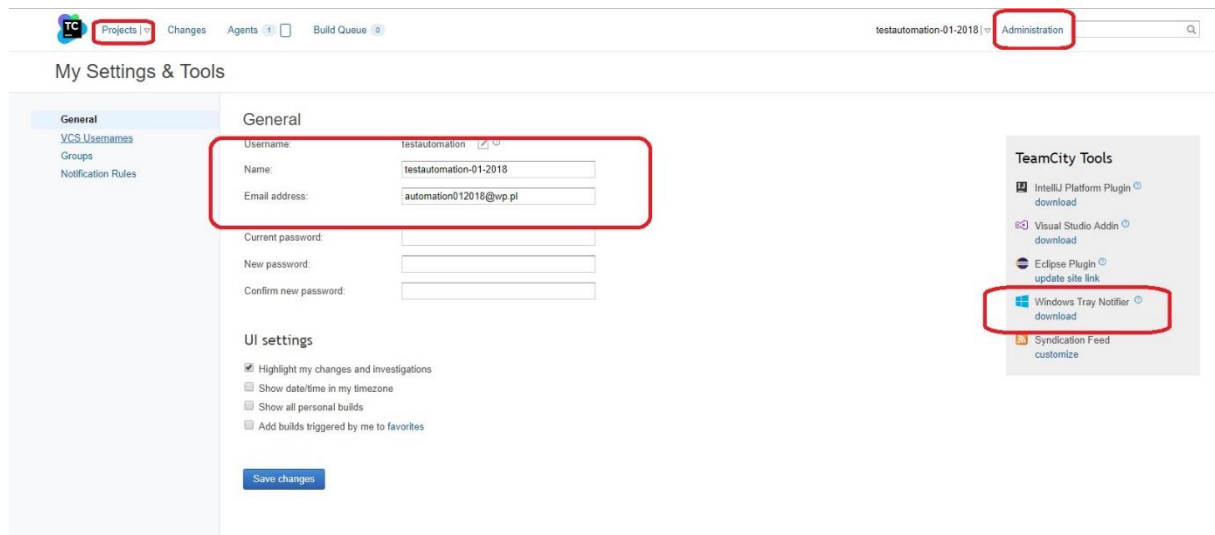
Łączymy na stronie

My settings & Tools

Wpisujemy

Name: (najlepiej takie jak używane jest w git)

Email address: (najlepiej takie jak używane jest w git)



Ściągamy Windows Tray Notifier – jest na listwie z prawej strony

Instalujemy Tray Notifier - defaultowo

Klikamy na ikonę u dołu w ikonach ukrytych

Pokaże nam się okno

TeamCity Tray Notifier

TeamCity URL: (podajemy url do TeamCity) http://localhost:9191

I klikamy Ok.

Pojawi się okno do logowania podajemy dane logowania do TeamCity

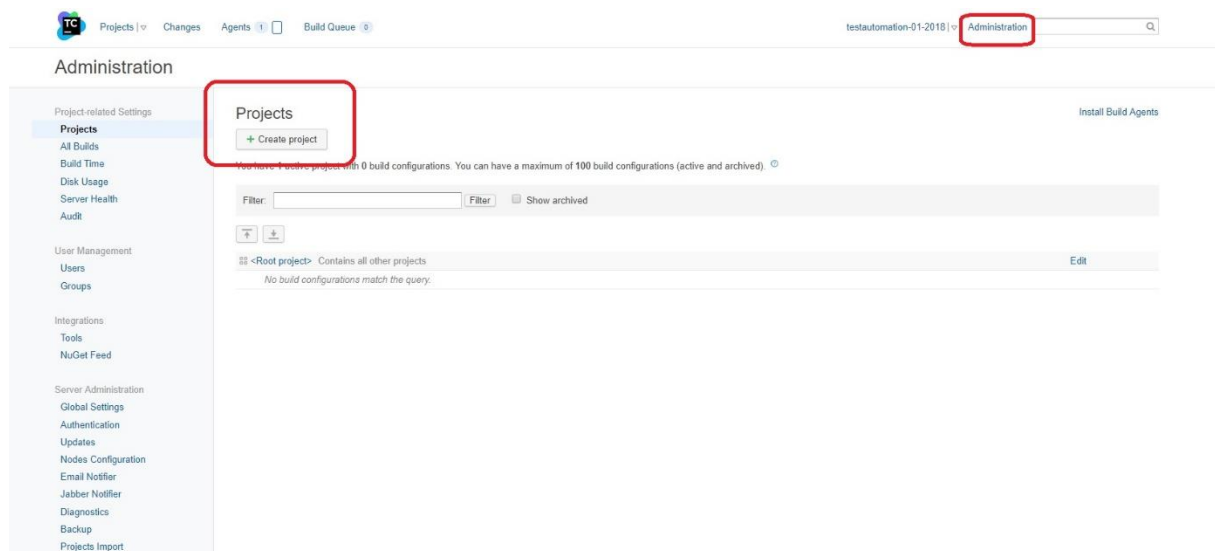
<https://confluence.jetbrains.com/display/TCD10/Configure+and+Run+Your+First+Build>

## Podłączamy projekt

Administration

Projects

Create project



From a repository URL (nie GitHub) bo przez githuba chce access tokena

(to jest konfiguracja na potrzeby nauki, na potrzeby zawodowe lepiej używać wszędzie autoryzowanych konfiguracji z użyciem klucza ssh lub/oraz access tokena)

Repository wprowadzamy URL do naszego projektu na githubie

<https://github.com/testautomation-01-2018/testautomation>

wprowadzamy username i password do githuba

Proceed

Administration / Projects / Changes / Agents / Build Queue

testautomation-01-2018 Administration

Administration / <Root project>

Create Project

From a repository URL

From GitHub Set up connection

From Bitbucket Cloud Set up connection

From Visual Studio Team Services Set up connection

Manually

Parent project: <Root project>

Repository URL:  A VCS repository URL. Supported formats: http(s)://, svn://, git://, etc. as well as URLs in Maven format.

Username:  Provide username if access to repository requires authentication.

Password:  Provide password if access to repository requires authentication.

Proceed

Create Project -> From URL > nic nie zmieniamy

Proceed ...

Zaznaczamy tylko checkboxa przy Maven

Administration / Projects / Changes / Agents / Build Queue

testautomation-01-2018 Administration

Administration / <Root project> / Testautomation

Run Actions Build Configuration Home

Build

General Settings

Version Control Settings

Build Steps

Triggers

Failure Conditions

Build Features

Dependencies

Parameters

Agent Requirements

Created moments ago by testautomation-01-2018 (view history)

New project "Testautomation", build configuration "Build" and VCS root "https://github.com/testautomation-01-2018/testautomation#refs/heads/master" have been successfully created.

Auto-detected Build Steps

Build steps and their settings are detected automatically by scanning VCS repository. You can configure build steps manually if auto-detect did not find relevant build steps.

Build Step

Maven

Parameters Description

Path to POM: pom.xml

Goals: clean test

Use selected Refresh

Use selected ...

Administration ->

Projects ->

Wybieramy nasz project

Administration

Projects | Changes | Agents | Build Queue

testautomation-01-2018 | Administration

Administration

Projects

+ Create project

You have 2 active projects with 1 build configuration. You can have a maximum of 100 build configurations (active and archived).

Filter: Filter Show archived

+ -

<Root project> Contains all other projects Edit

Testautomation Edit

General Settings

Projects

All Builds

Build Time

Disk Usage

Server Health

Audit

User Management

Users

Groups

Integrations

Tools

NuGet Feed

Server Administration

Global Settings

Authentication

Updates

Nodes Configuration

Email Notifier

Jabber Notifier

Diagnostics

Backup

Projects Import

Install Build Agents

## Version Control Settings > VCS Roots >

Administration / <Root project>

Testautomation

General Settings

VCS Roots

Report Tabs

Parameters

Builds Schedule

Connections

Shared Resources

Meta-Runners

Maven Settings

Issue Trackers

Cloud Profiles

SSH Keys

Clean-up Rules

Versioned Settings

Created 15 minutes ago by testautomation-01-2018 (view history)

Name: \* Testautomation

Project ID: \* Testautomation Regenerate ID

Description:

Save Cancel

Build Configurations (99 left)

Build configurations define how to retrieve and build sources of a project.

+ Create build configuration + Create composite build configuration

Name	Build Steps	
Build	Maven	Edit

Build Configuration Templates

Build configuration templates define settings that can be reused by different build configurations.

+ Create template

Subprojects

Subprojects can be used to group build configurations and define projects hierarchy within a single project.

+ Create subproject

## Edit VCS Root

## Edit > Link do projektu

Administration / <Root project>

Testautomation

General Settings

VCS Roots

Report Tabs

Parameters

Builds Schedule

Connections

Shared Resources

Meta-Runners

Maven Settings

Issue Trackers

Cloud Profiles

SSH Keys

Clean-up Rules

Versioned Settings

Created 19 minutes ago by testautomation-01-2018 (view history)

VCS Roots

A VCS Root is a set of settings defining how TeamCity communicates with a version control system to monitor changes and get sources of a build.

+ Create VCS root

Filter: Filter Show VCS roots from sub-projects Show unused VCS roots only

Name	checked at	usage	
(git) https://github.com/testautomation-01-2018/testautomation#refs/heads/master	2 67	1 usage	Edit   deletable



## Rozwijamy zaawansowane ustawienia

Administration / <<Root project> / <<<Testautomation> / Edit Project Actions

### Edit VCS Root

Type of VCS: Git

Used in 1 build configuration | [View history](#)

Created 20 minutes ago by testautomation-01-2018 (view history)

VCS Root

VCS root name: https://github.com/testautomation-01-2018/testautomation#refs/heads/  
A unique name to distinguish this VCS root from other roots

VCS root ID: Testautomation\_HttpsGithubComTestautomation012018testautoma [Regenerate ID](#)  
VCS root ID must be unique across all VCS roots. VCS root ID can be used in parameter references to VCS root parameters and REST API.

General Settings

Fetch URL: https://github.com/testautomation-01-2018/testautomation  
It is used for fetching data from the repository.

Push URL:   
It is used for pushing tags to the remote repository. If blank, the fetch url is used.

Default branch: refs/heads/master  
The main branch or tag to be monitored

Authentication Settings

Authentication method: Password  
Authentication methods incompatible with the 'https' protocol are disabled

Username: automation012018@wp.pl  
Specify the username if there is no username in the clone URL. The username specified here overrides the username from the URL.

Password: \*\*\*\*\*

VCS Root Project

Belongs to project: Testautomation [Move](#)

[Show advanced options](#)

Musimy ustawić ścieżkę do gita

Administration > projekt > itd.....

Edit VCS ROOT

Path to Git -> ustawić ścieżkę do gita

C:\\"Program Files"\Git

The screenshot shows the TeamCity configuration page for a project. Several settings are highlighted with red rectangular boxes:

- Fetch URL:** `https://github.com/testautomation-01-2018/testautomation`
- Push URL:** (Empty field)
- Default branch:** `refs/heads/master`
- Branch specification:** (Empty field)
- Use tags as branches:** ☒ **Enable to use tags in the branch specification**
- Username style:** `Userid`
- Submodules:** `Checkout`
- Username for tags/merge:** (Empty field)
- Authentication Settings:**
  - Authentication method:** `Anonymous`
- Server Settings:** (Empty field)
- Convert line-endings to CRLF:** ☒
- Agent Settings:** (Empty field)
- Path to Git:** `C:\Program Files\Git`

Na dole sprawdzamy czy połączenie jest poprawne

I dajemy

Save ...

W teamCity są trigery – które ustawiamy przy użyciu wyrażeń crone

Możemy ustawić trigery np. aby się testy uruchamiały co godzinę.

Można jeszcze ustawić aby się testy odpalały po każdym pullu do mastera

Na githubie Webhooks i Integrations & services

Konfiguracja na githubie aby buildy wykonywały się przy pull-u do mastera

# Github Project

## Settings

### Integrations & Services

testautomation-01-2018 / testautomation

Watch 0 Star 0 Fork 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights **Settings**

Options Collaborators Branches Webhooks **Integrations & services** Deploy keys

Services / Manage TeamCity Test service

**Note:** GitHub Services are being deprecated. Please contact your integrator for more information on how to migrate or replace a service to webhooks or GitHub Apps.

TeamCity is a continuous integration server that automates building and testing of your software.

Note that since TeamCity 10.0 using the [TeamCity Commit Hooks](https://plugins.jetbrains.com/plugin/9179?pr=teamcity) plugin is recommended. The GitHub TeamCity service can be used in two ways: \* to trigger builds after code has been pushed to your git repository; (Default) \* to enforce checking for changes after code has been pushed to your git repository. (see 7)

#### Install Notes

1. Your TeamCity server must be accessible from the internet.
2. "Base url" is the URL to your TeamCity server  
Example: <https://teamcity.example.com/> or <http://teamcity.example.com/teamcity/>
3. "Build type" is the identifier of the build configuration you want to trigger.  
Multiple configurations can be triggered by specifying a comma-separated list of identifiers.  
Example: "bt123", which can be found in URL of the build configuration page ("...viewType.html?buildTypeId=bt123&...")
4. "Username" and "Password" - username and password of a TeamCity user that can trigger a manual build of the TeamCity build configuration defined in "Build type"
5. "Branches" (Optional) is an space-separated list of branches to watch commits for.  
Commits to other branches will be ignored.  
If no branches are specified, TeamCity will be notified of all commits.
6. "Full branch ref" if enabled full branch reference (e.g. 'refs/heads/master') will be send to TeamCity server. Otherwise 'refs/heads' will be omitted.
7. "Check for pending changes" if enabled, service will force TeamCity server to check for pending changes. Service will not trigger new build.
8. Since TeamCity uses BASIC authentication, it is highly recommended that the

### Add Service – wyszukuje teamCity

This repository Search Pull requests Issues Marketplace Explore

testautomation-01-2018 / testautomation Watch 0 Star 0 Fork 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights **Settings**

Options Collaborators Branches Webhooks **Integrations & services** Deploy keys

#### Installed GitHub Apps

GitHub Apps augment and extend your workflows on GitHub with commercial, open source, and homegrown tools.

#### Services

**Add service**

**Note:** GitHub Services are being deprecated. Please contact your integrator for more information on how to migrate or replace a service to webhooks or GitHub Apps.

Services are pre-built integrations that perform certain actions when events occur on GitHub.

**TeamCity** Edit Delete

Dodajemy Base url <http://localhost:9191>

Username i password do teamcity

4. "Username" and "Password" - username and password of a TeamCity user that can trigger a manual build of the TeamCity build configuration defined in "Build type"

5. "Branches" (Optional) is a space-separated list of branches to watch commits for. Commits to other branches will be ignored. If no branches are specified, TeamCity will be notified of all commits.

6. "Full branch ref" if enabled full branch reference (e.g. 'refs/heads/master') will be send to TeamCity server. Otherwise 'refs/heads' will be omitted.

7. "Check for pending changes" if enabled, service will force TeamCity server to check for pending changes. Service will not trigger new build.

8. Since TeamCity uses BASIC authentication, it is highly recommended that the TeamCity server use HTTPS/SSL to protect the password that is passed unencrypted over the internet.

Base url

Build type

Branches

☐ Full branch ref

Username

Password  
\*\*\*\*\* — Edit

☐ Check for changes only

☒ Active  
We will run this service when an event is triggered.

Build Type: Build configuration ID z team city pobrane -> z

Administration > (MyProject) > build (Maven) Edit >

TC Projects | ▾ Changes Agents | 1 ▾ Build Queue ▾ testautomation-01-2018 | Administration

Administration / <<Root project>>

Testautomation 1 Info item

General Settings

VCS Roots 1

Report Tabs 1

Parameters

Builds Schedule

Connections

Shared Resources

Meta-Runners

Maven Settings

Issue Trackers

Cloud Profiles

SSH Keys

Clean-up Rules

Versioned Settings

Created 37 minutes ago by testautomation-01-2018 (view history)

Name: \*

Project ID: \*  Regenerate ID

Description:

Build Configurations (99 left)

Build configurations define how to retrieve and build sources of a project.

Name	Build Steps
Build	Maven

▾

Build Configuration Templates

Build configuration templates define settings that can be reused by different build configurations.

Subprojects

Subprojects can be used to group build configurations and define projects hierarchy within a single project.

TC Projects | Changes Agents 1 Build Queue 8 testautomation-01-2018 | Administration

Administration / <Root project> / Testautomation Run Actions Build Configuration Home

Build 1 info item

**General Settings**

- Version Control Settings 1
- Build Step: Maven
- Triggers 1
- Failure Conditions
- Build Features
- Dependencies
- Parameters
- Agent Requirements

Last edited 35 minutes ago by testautomation-01-2018 (view history)

Name: Build

Build configuration ID: Testautomation\_Build Regenerate ID

Description:

Build configuration type: Regular Builds of a regular build configuration can have build steps and are executed on agents.

Build number format: %build.counter% The format may include "%build.counter%" as a placeholder for the build counter value, for example, 1.%build.counter%. It may also contain a reference to any other available parameter, for example, %build.vcs.number.VC.RootName%. Note: The maximum length of a build number after all substitutions is 256 characters.

Build counter: 1 Reset

Artifact paths:

Build options:

- ☒ enable hanging builds detection
- ☒ allow triggering personal builds
- ☐ enable status widget

Limit the number of simultaneously running builds (0 — unlimited) 0

Hide advanced options

I klikamy

Update Service

Mamy skonfigurowane

Jeżeli teraz spuszczamy zmiany do mastera nastąpi automatyczne wywołanie builda na TeamCity

Możemy przejść na Team City i sprawdzić

Należy poczekać chwilę zanim się build uruchomi

# Jak ustawić triggera do automatycznego wykonywania buildów

## Administration ->

The screenshot shows the Jenkins Administration interface. The top navigation bar includes 'Projects', 'Changes', 'Agents', and 'Build Queue'. The 'Administration' link is highlighted in the top right. The left sidebar lists various administration settings. The main content area shows the 'Projects' section with a 'Create project' button and a list of projects. The '<Root project>' and 'Testautomation' projects are highlighted with a red box.

## Root project : Mój projekt (testautomation) ->

## Build Configuration : maven : Edit ->

The screenshot shows the Jenkins Build Configuration page for the 'Testautomation' project. The 'Build Configurations' section is highlighted with a red box. The 'Build' configuration is selected, and the 'Edit' button is highlighted with a red box.

Administration / <Root project> / Testautomation

Build Queue

testautomation-01-2018 Administration

Run Actions Build Configuration Home

Build 1 info item

General Settings

Version Control Settings

Build Step: Maven

Triggers 1

Failure Conditions

Build Features

Dependencies

Parameters

Agent Requirements

Last edited 59 minutes ago by testautomation-01-2018 (view history)

Name: Build

Build configuration ID: Testautomation\_Build Regenerate ID

Description:

Build configuration type: Regular

Build number format: %build.counter%

Build counter: 0 Reset

Artifact paths:

Build options:

enable hanging builds detection

allow triggering personal builds

enable status widget

Limit the number of simultaneously running builds (0 — unlimited) 0

Hide advanced options

Administration / <Root project> / Testautomation

Build Queue

testautomation-01-2018 Administration

Run Actions Build Configuration Home

Build 1 info item

General Settings

Version Control Settings

Build Step: Maven

Triggers 1

Failure Conditions

Build Features

Dependencies

Parameters

Agent Requirements

Last edited one hour ago by testautomation-01-2018 (view history)

Triggers

Triggers are used to add builds to the queue either when an event occurs (like a VCS check-in) or periodically with some configurable interval.

+ Add new trigger

Trigger	Parameters	Description	Edit	
VCS Trigger	Branch filter: +*			

+Add new trigger

Wybieramy z listy rodzaj triggera, jeżeli chcemy zrobić cykliczne uruchomienie

To wybieramy

Schedule Trigger

I z listy:

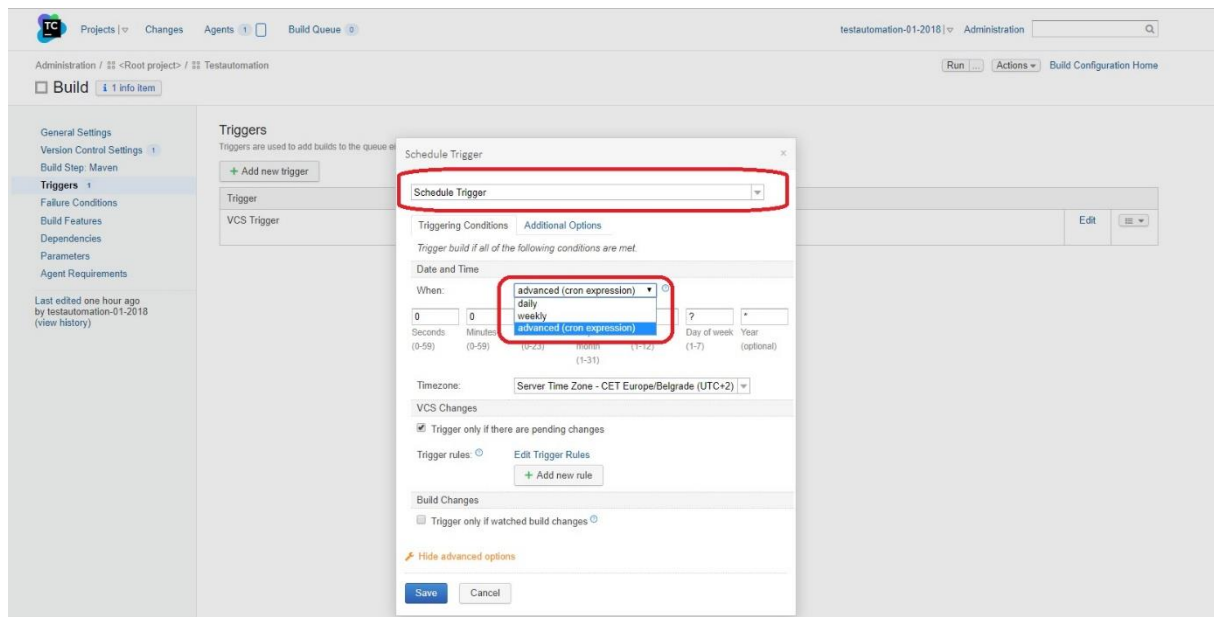
When

Wybieramy cykliczność

Lub

advanced (cron expression)

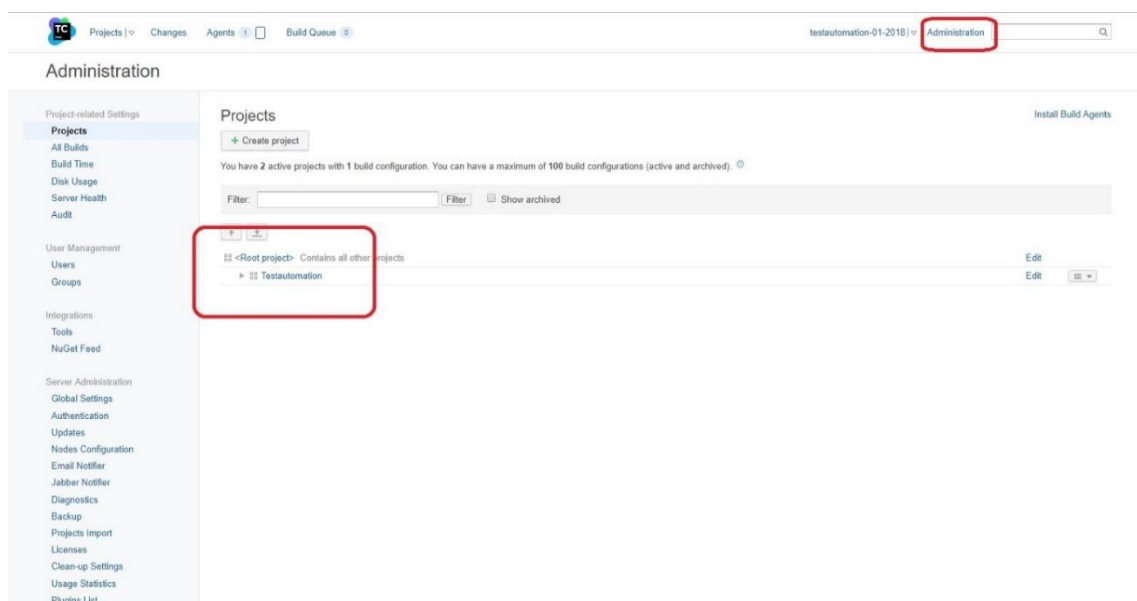
i tam możemy zdefiniować cykliczność z użyciem crone wyrażen



Można również użyć Parameters aby sparametryzować uruchamianie builda  
 Np. można ustawić tam zmienne systemowe, zmienne środowiskowe,  
 parametry konfiguracyjne z którymi będą się uruchamiały buildy , można tam  
 ustawić nazwy zmiennych, wartości defaultowe, listy wyboru i wiele innych  
 przydatnych konfiguracji.

Administration ->

Root project : Mój projekt (testautomation) ->





TC

Projects | ▾ | Changes | Agents | 1 | Build Queue | 0

testautomation-01-2018 | ▾ | Administration |

Administration / <Root project>

Testautomation | ▾

General Settings

VCS Roots | 1

Report Tabs | 1

**Parameters**

Builds Schedule

Connections

Meta-Runners

Shared Resources

Maven Settings

Issue Trackers

Cloud Profiles

SSH Keys

Clean-up Rules

Versioned Settings

Last edited 15 hours ago by testautomation-01-2018 (view history)

Name: \*

Testautomation

Project ID: + ⓘ

Testautomation

Regenerate ID

This ID is used in URLs, REST API, HTTP requests to the server, and configuration settings in the TeamCity Data Directory.

Description:

Save Cancel

Build Configurations (59 left)

Build configurations define how to retrieve and build sources of a project ⓘ

+ Create build configuration + Create composite build configuration

Name	Build Steps
Build	Maven

Edit | ▾

Build Configuration Templates

Build configuration templates define settings that can be reused by different build configurations. ⓘ

+ Create template

Subprojects

Subprojects can be used to group build configurations and define projects hierarchy within a single project. ⓘ

+ Create subproject

TC

Projects | ▾ | Changes | Agents | 1 | Build Queue | 0

testautomation-01-2018 | ▾ | Administration |

Administration / <Root project>

Testautomation | ▾

General Settings

VCS Roots | 1

Report Tabs | 1

**Parameters**

Builds Schedule

Connections

Meta-Runners

Shared Resources

Maven Settings

Issue Trackers

Cloud Profiles

SSH Keys

Clean-up Rules

Versioned Settings

Last edited 15 hours ago by testautomation-01-2018 (view history)

+ Add new parameter

Configuration Parameters

Configuration parameters are not passed into build, can be used in references only ⓘ

None defined

System Properties (system.)

System properties will be passed into the build

None defined

Environment Variables (env.)

Environment variables will be added to the env

None defined

Add New Parameter

Name: \*

Kind:

Configuration parameter

Configuration parameter

System property (system.)

Environment variable (env.)

Value:

Spec:

Edit...

Show raw value

Defines parameter control presentation and validation.

Save Cancel

-----

# Jenkins

## Instalacja

Ściągnąć zipa z Jenkinsem

<https://jenkins.io/download/>

Download Jenkins 2.176.1 for:

Windows

Rozpakować i uruchomić instalator jenkins.msi

Począć na uruchomienie Jenkinsa

Odpali się na przeglądarce na porcie 8080

Hasło będzie w pliku

C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword

wprowadzić hasło i zainstalować sugerowane pluginy

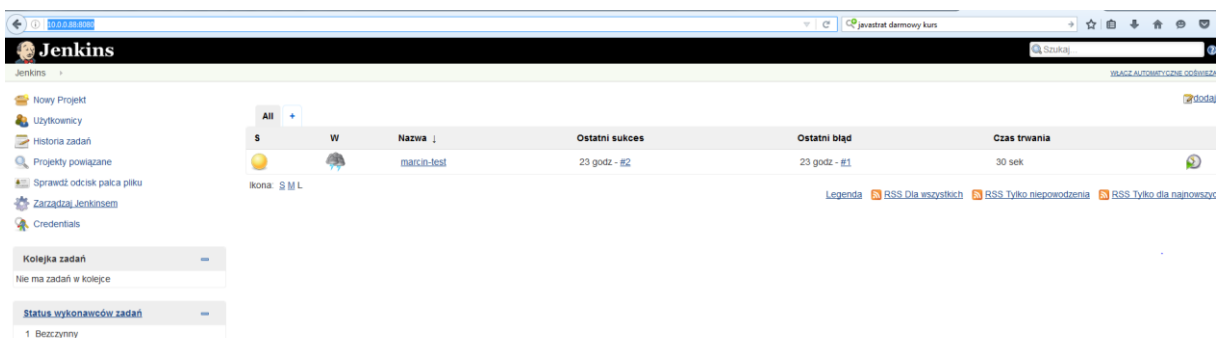
ważne aby się pluginy Git oraz GitHub zainstalowały

wprowadzić nowego usera i password

uruchomić

Uruchamiamy z domyślnym URL-em

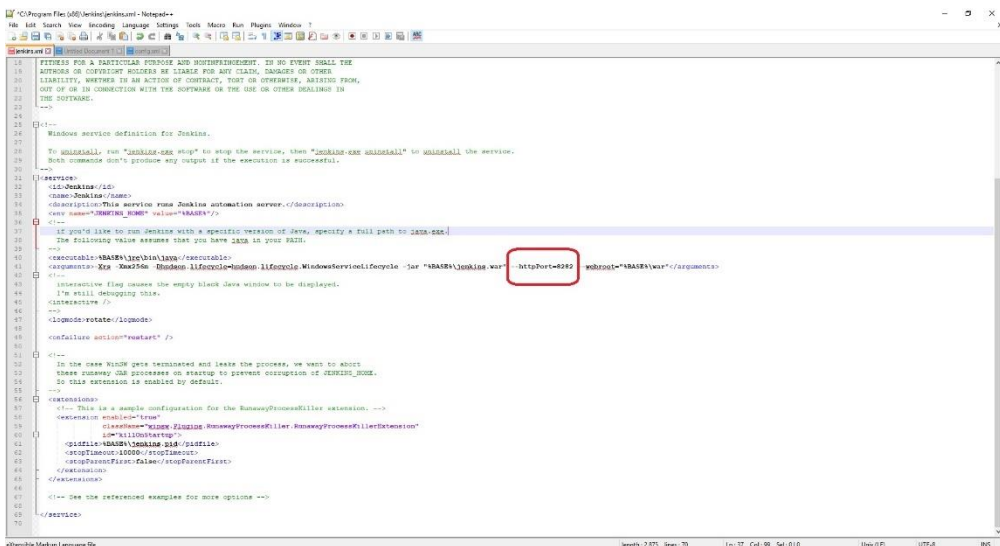
Jenkins uruchomi się na <http://localhost:8080/>



Możemy zmienić port dla Jenkinsa jeżeli nam koliduje

Wchodzimy do pliku C:\Program Files (x86)\Jenkins\jenkins.xml

i zmieniamy zmienną --httpPort=8282



oraz w zmieniamy ustawienie URL-a w Jenkinsie

Wchodzimy na Jenkinsa -> Zarządzaj Jenkinsem -> Skonfiguruj System

Jenkins

Nowy Projekt  
Użytkownicy  
Historia zadań  
**Zarządzaj Jenkinsem**  
Moje widoki  
Lockable Resources  
Credentials  
Nowy widok

Kolejka zadań  
Nie ma zadań w kolejce

Status wykonawców zadań  
1 Bezczynny  
2 Bezczynny

S	P	Nazwa	Ostatni sukces
		gł	23 godz - #4

Ikona: [Średnia](#) [Mała](#) [Duża](#)

Jenkins

Nowy Projekt  
Użytkownicy  
Historia zadań  
**Zarządzaj Jenkinsem**  
Moje widoki  
Lockable Resources  
Credentials  
Nowy widok

Kolejka zadań  
Nie ma zadań w kolejce

Status wykonawców zadań  
1 Bezczynny  
2 Bezczynny

### Zarządzaj Jenkinsem

- Skonfiguruj system**  
Konfiguruj ustawienia globalne i ścieżki.
- Konfiguruj ustawienia bezpieczeństwa**  
Zabezpiecz Jenkinsa: decyduj, kto ma do niego dostęp.
- Configure Credentials**  
Configure the credential providers and types
- Globalne narzędzia do konfiguracji**  
Konfiguruj narzędzia, ścieżki do nich i automatyczne instalatory
- Odczytaj ponownie konfigurację z dysku**  
Porzuć wszystkie dane załadowane w pamięci i załaduj wszystko z systemu plików. Opcja użyteczna gdy zmodyfikowano pliki konfig.
- Zarządzaj wtyczkami**  
Dodaj, usuń, wyłącz lub włącz wtyczki które mogą rozszerzyć funkcjonalność Jenkinsa.
- Informacje o systemie**  
Wyświetla wiele środowiskowych informacji pomocnych przy rozwiązywaniu problemów.
- Dziennik systemowy**  
Dziennik systemowy gromadzi wywołania java.util.logging powiązane z Jenkinsem.

Jenkins > konfiguracja

Global properties

☐ Disable deferred wipeout on this node  
☐ Lokalizacja narzędzi  
☐ Zmienne środowiskowe

Pipeline Speed/Durability Settings

Pipeline Default Speed/Durability Level:

Usage Statistics

☒ Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project.

Timestamp

System clock time format:

Elapsed time format:

Enabled for all Pipeline builds: ☐

Administrative monitors configuration

Jenkins Location

Jenkins URL:

System Admin e-mail address:

Lockable Resources Manager

Lockable Resources:

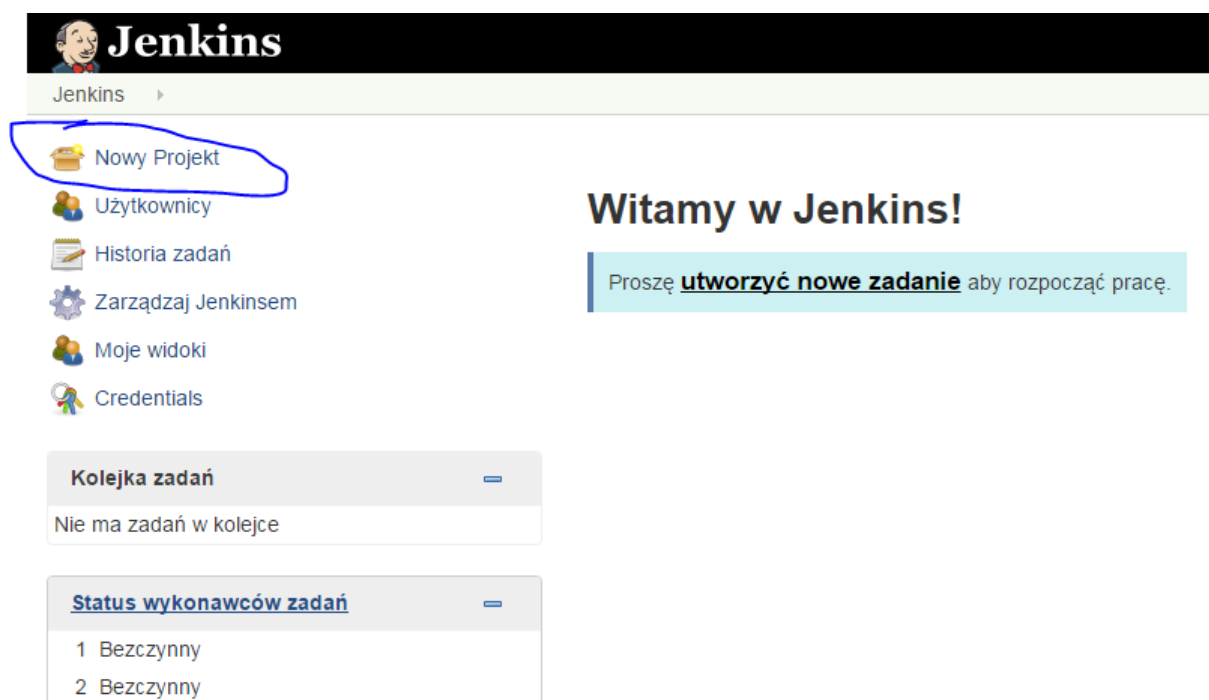
GitHub

GitHub Servers:

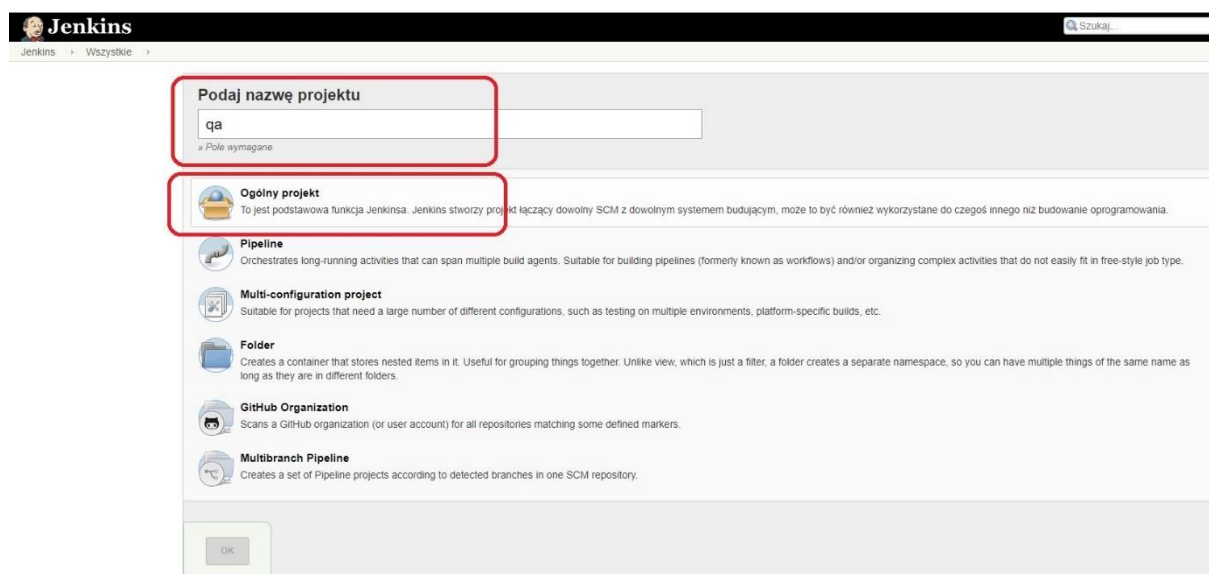
## Nowy Projekt

Wracamy na stronę główną Jenkinsa

Nowy Projekt ->



Podajemy nazwę dla naszego projektu i zaznaczamy <Ogólny Projekt> i wciskamy <Ok>



## Konfigurujemy:

### Repozytorium Kodu źródłowego

Wskazujemy repozytorium Git

Oraz podajemy ścieżkę do projektu (do katalogu .git)

Np. <https://github.com/testautomation-06-2019/qa.git>

The screenshot shows the 'Repository' configuration page in Jenkins. The 'Git' radio button is selected. Under 'Repositories', the 'Repository URL' is set to 'https://github.com/streser/automatyzacja-grudzien-2016.git' and 'Credentials' is set to '- none -'. There are buttons for 'Zaawansowane...', 'Add Repository', and 'Add Branch'. Under 'Branches to build', the 'Branch Specifier (blank for \'any\')' is set to '\*/mmajewska'. There is an 'Add Branch' button. Under 'Repository browser', it is set to '(Automatyczny)'. Under 'Additional Behaviours', there is an 'Add' button. At the bottom, the 'Subversion' radio button is also visible.

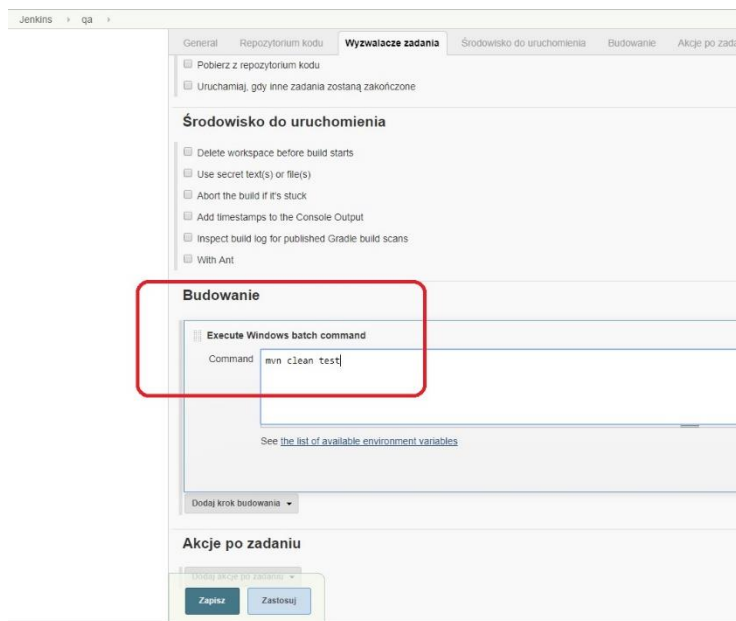
## Budowanie

Ustawiamy jak ma się budować projekt, w naszym przypadku mamy Jenkinsa na Windowsie zainstalowanego więc użyjemy windowsowego cmd

Dodaj krok budowania -> Execute Windows batch command  
(jeżeli by był na linuxie to możemy użyć (Uruchom powłokę))

The screenshot shows the 'Build' configuration page in Jenkins. The 'Wyzwalacz zadania' tab is active. Under 'Wyzwalacz zadania', there are several checkboxes: 'Wyzwalaj budowanie zdalnie (np. przez skrypt)', 'Buduj cyklicznie', 'GitHub hook trigger for GITScm polling', 'Pobierz z repozytorium kodu', and 'Uruchamiaj, gdy inne zadania zostaną zakończone'. Under 'Środowisko do uruchomienia', there are checkboxes: 'Delete workspace before build starts', 'Use secret text(s) or file(s)', 'Abort the build if it's stuck', 'Add timestamps to the Console Output', 'Inspect build log for published Gradle build scans', and 'With Ant'. Under 'Budowanie', there is a dropdown menu 'Dodaj krok budowania' which is open, showing options: 'Execute Windows batch command', 'Invoke Ant', 'Invoke Gradle script', 'Invoke top-level Maven targets', 'Run with timeout', 'Set build status to \'pending\' on GitHub commit', and 'Uruchom powłokę'. The 'Execute Windows batch command' option is highlighted. At the bottom, there are 'Zapisz' and 'Zastosuj' buttons.

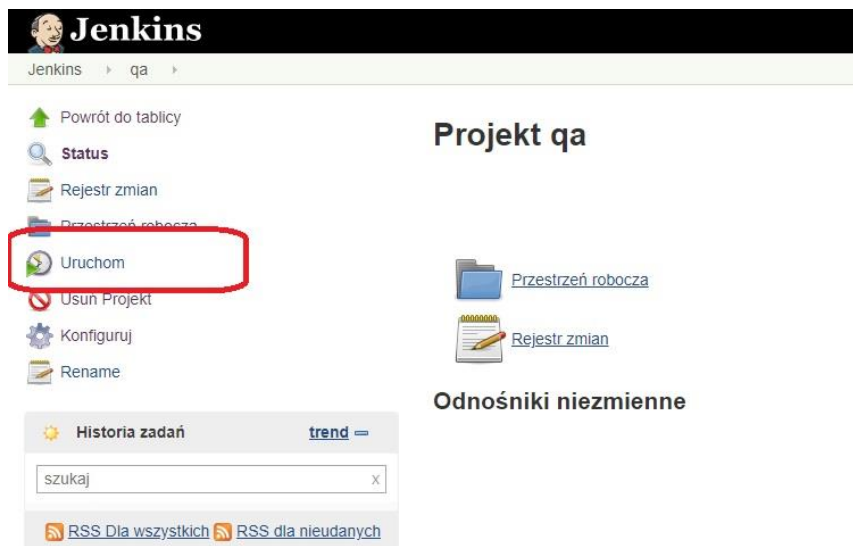
I wprowadzamy komendę: mvn clean test



Zastosuj i Zapisz  
Sprawdzamy czy działa.

**Uruchom**

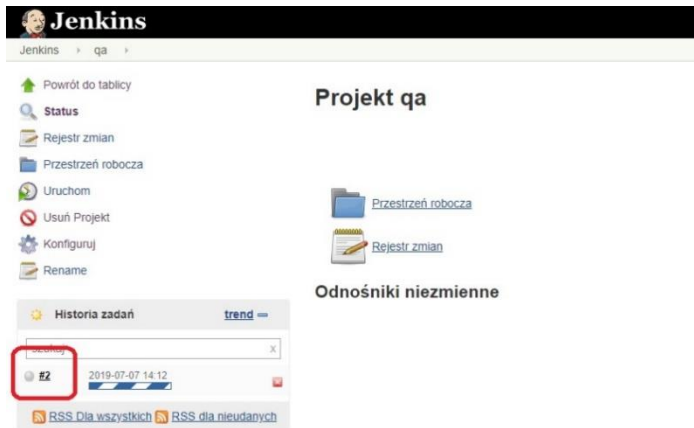
Klikamy: Uruchom



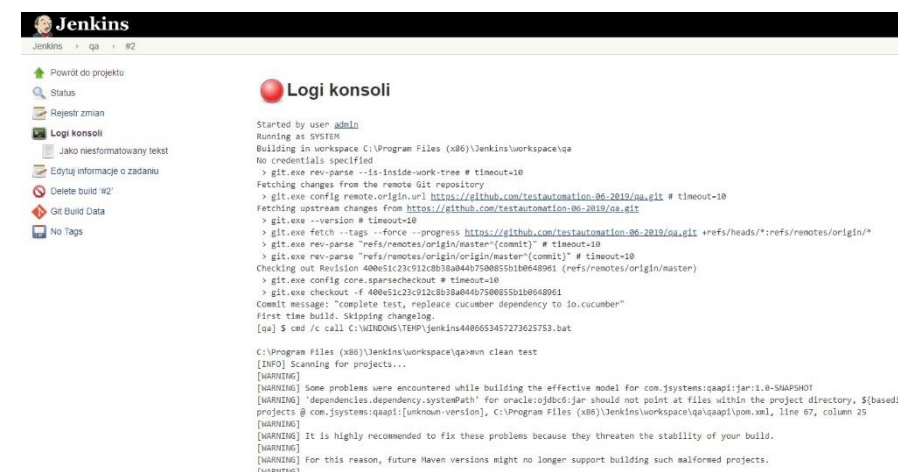
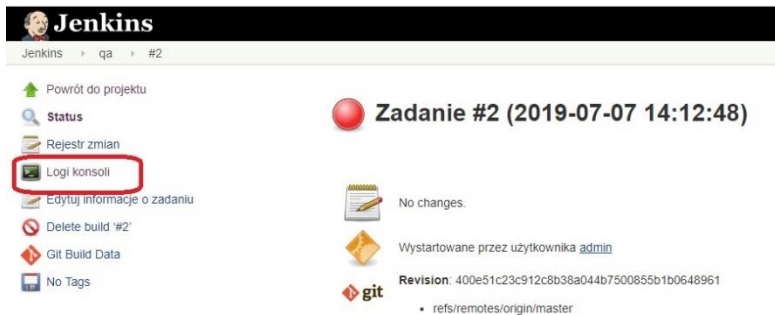
## Logi z Konsoli

Możemy podglądać logi w trakcie budowania

Klikamy na nowy build

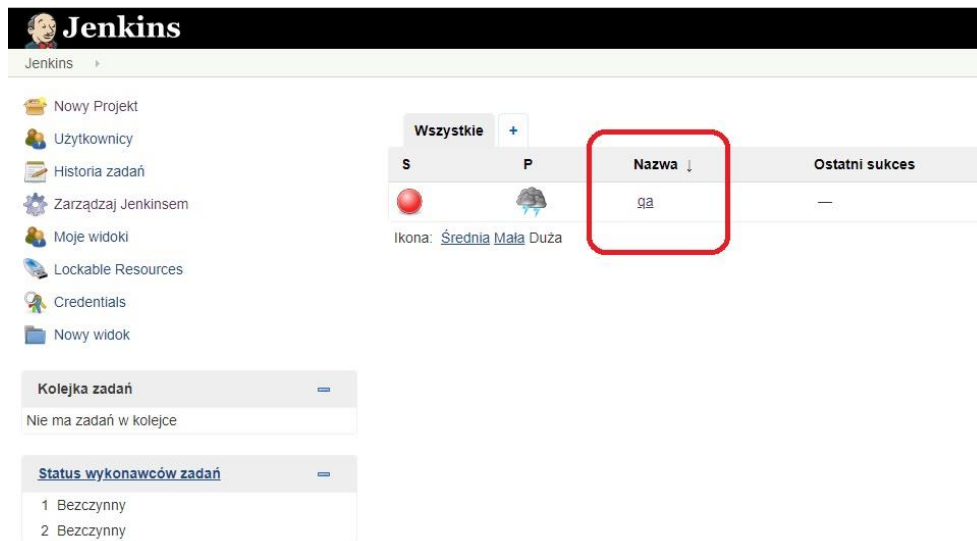


Klikamy: Logi z Konsoli



## Parametryzacja

Przechodzimy do jenkinsa i wybieramy projekt

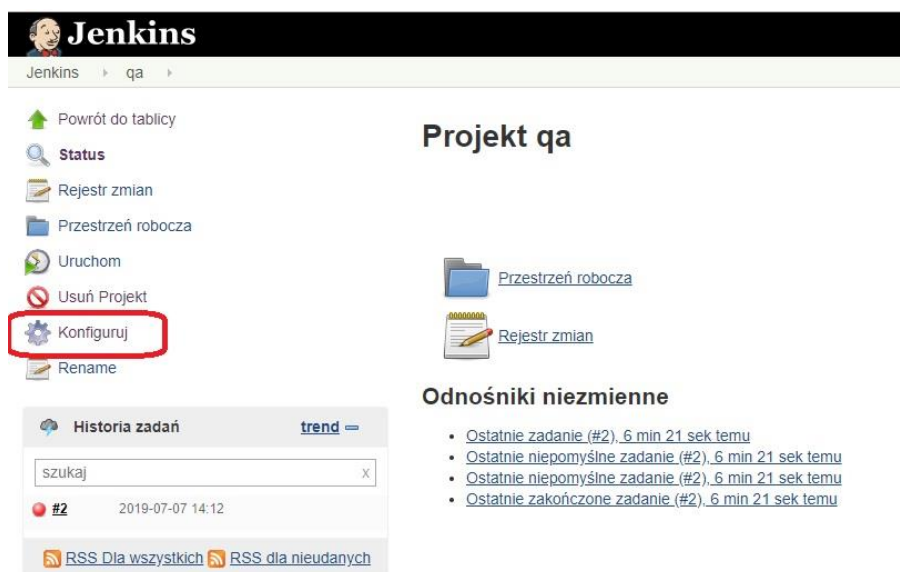


The screenshot shows the Jenkins dashboard. On the left is a sidebar with navigation links: Nowy Projekt, Użytkownicy, Historia zadań, Zarządzaj Jenkinsem, Moje widoki, Lockable Resources, Credentials, and Nowy widok. Below this is a 'Kolejka zadań' section showing 'Nie ma zadań w kolejce' and a 'Status wykonawców zadań' section showing two 'Bezczynny' (Idle) executors. The main area displays a table of projects. The first project is 'qa'. The 'Nazwa' column header is highlighted with a red box. Below the table, there are links for 'Ikona: Średnia Mała Duża'.

S	P	Nazwa ↓	Ostatni sukces
		qa	—

Ikona: [Średnia](#) [Mała](#) [Duża](#)

Wybieramy: Konfiguruj



The screenshot shows the Jenkins configuration page for the 'qa' project. The left sidebar contains links: Powrót do tablicy, Status, Rejestr zmian, Przestrzeń robocza, Uruchom, Usuń Projekt, Konfiguruj (highlighted with a red box), and Rename. The main area is titled 'Projekt qa' and contains links for 'Przestrzeń robocza' and 'Rejestr zmian'. Below this is a section 'Odkazy na ostatnie zadania' (Links to last jobs) with a list of links. At the bottom, there is a 'Historia zadań' (Job history) section with a search bar and a table of job history.

### Projekt qa

[Przestrzeń robocza](#)

[Rejestr zmian](#)

#### Odkazy na ostatnie zadania

- [Ostatnie zadanie \(#2\), 6 min 21 sek temu](#)
- [Ostatnie niepowodzenie zadania \(#2\), 6 min 21 sek temu](#)
- [Ostatnie niepowodzenie zadania \(#2\), 6 min 21 sek temu](#)
- [Ostatnie zakończone zadanie \(#2\), 6 min 21 sek temu](#)

#### Historia zadań

szukaj x

#	Time
#2	2019-07-07 14:12

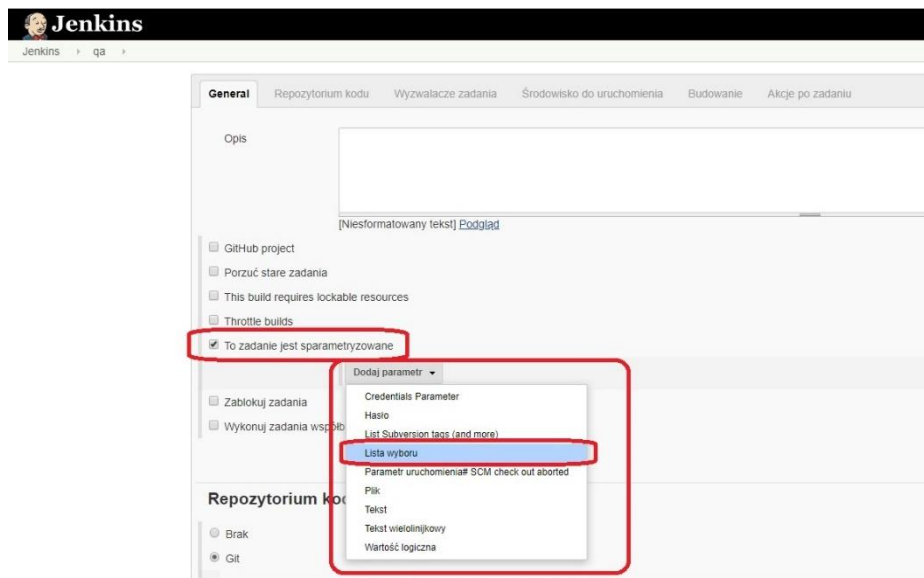
[RSS Dla wszystkich](#) [RSS dla nieudanych](#)

Przechodzimy do sekcji General

Wybieramy: To zadanie jest sparametryzowane ->

Dodaj Parametr -> Lista wyboru



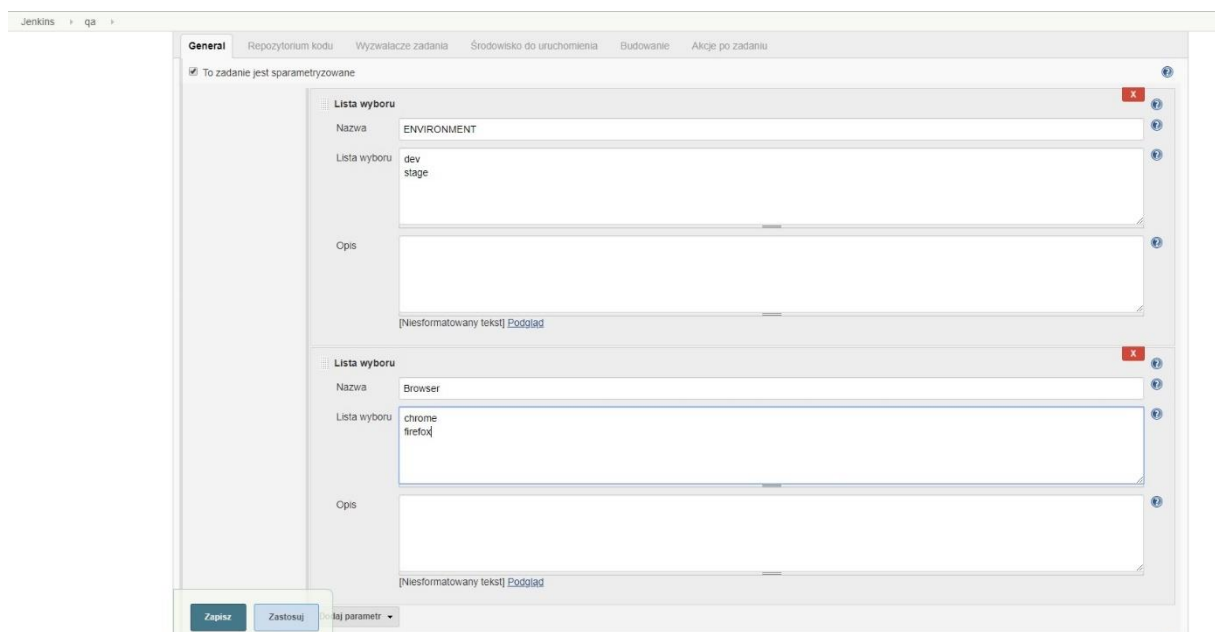


I podajemy zmienną oraz wartości dla wyboru

Dla naszego projektu ustawiamy dwie zmienne: ENVIRONMENT oraz BROWSER

Używamy entera do podania kolejnej

Ta która ma być domyślna znajduje się na samej górze



Zastosuj i Zapisz

I możemy uruchomić builda z naszymi parametrami

Wracamy do Jenkinsa -> wybieramy projekt

I klikamy: Uruchom z parametrami

Wybieramy parametry albo pozostawiamy domyślne i klikamy: Buduj

**Jenkins**

Jenkins > qa >

[Powrót do tablicy](#)

[Status](#)

[Rejestr zmian](#)

[Przestrzeń robocza](#)

**Uruchom z parametrami**

[Usuń Projekt](#)

[Konfiguruj](#)

[GitHub Hook Log](#)

[Rename](#)

**Projekt qa**

To zadanie wymaga parametrów:

ENVIRONMENT

Browser

**Buduj**

**Historia zadań** [trend](#)

szukaj X

#2 2019-07-07 14:12

[RSS Dla wszystkich](#) [RSS dla nieudanych](#)

## WebHook

Ustawiamy WebHook-a aby przy pushu do mastera uruchamiał nam automatycznie builda

Przechodzimy do jenkinsa i wybieramy projekt

**Jenkins**

Jenkins >

[Nowy Projekt](#)

[Użytkownicy](#)

[Historia zadań](#)

[Zarządzaj Jenkinsem](#)

[Moje widoki](#)

[Lockable Resources](#)

[Credentials](#)

[Nowy widok](#)

**Wszystkie** +

S	P	Nazwa ↓	Ostatni sukces
		qa	—

Ikona: [Średnia](#) [Mała](#) [Duża](#)

**Kolejka zadań** —

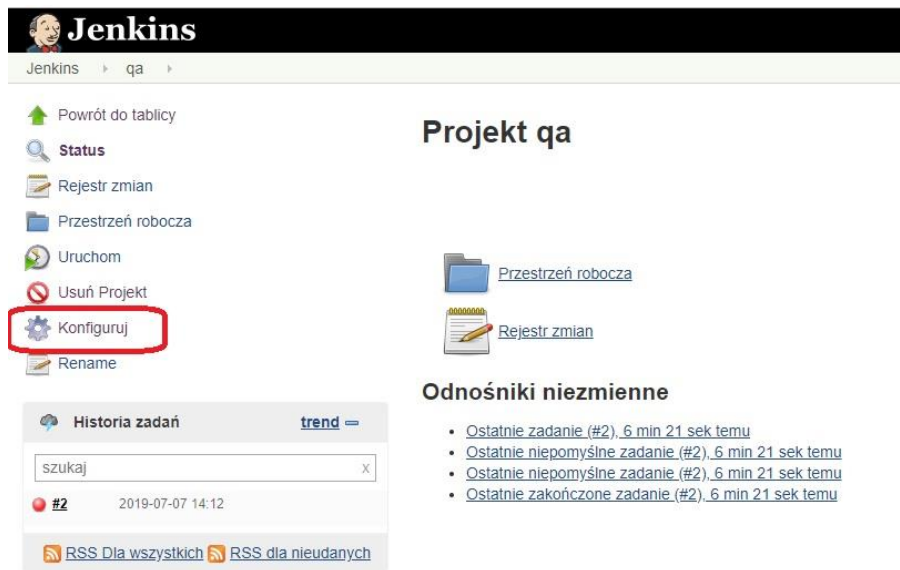
Nie ma zadań w kolejce

**Status wykonawców zadań** —

1 Bezczynny

2 Bezczynny

Wybieramy: Konfiguruj



**Jenkins**

Jenkins » qa »

[Powrót do tablicy](#)

**Status**

[Rejestr zmian](#)

[Przestrzeń robocza](#)

[Uruchom](#)

[Usuń Projekt](#)

**Konfiguruj**

[Rename](#)

**Projekt qa**

[Przestrzeń robocza](#)

[Rejestr zmian](#)

**Odnośniki niezmiennie**

- [Ostatnie zadanie \(#2\), 6 min 21 sek temu](#)
- [Ostatnie niepomyślne zadanie \(#2\), 6 min 21 sek temu](#)
- [Ostatnie niepomyślne zadanie \(#2\), 6 min 21 sek temu](#)
- [Ostatnie zakończone zadanie \(#2\), 6 min 21 sek temu](#)

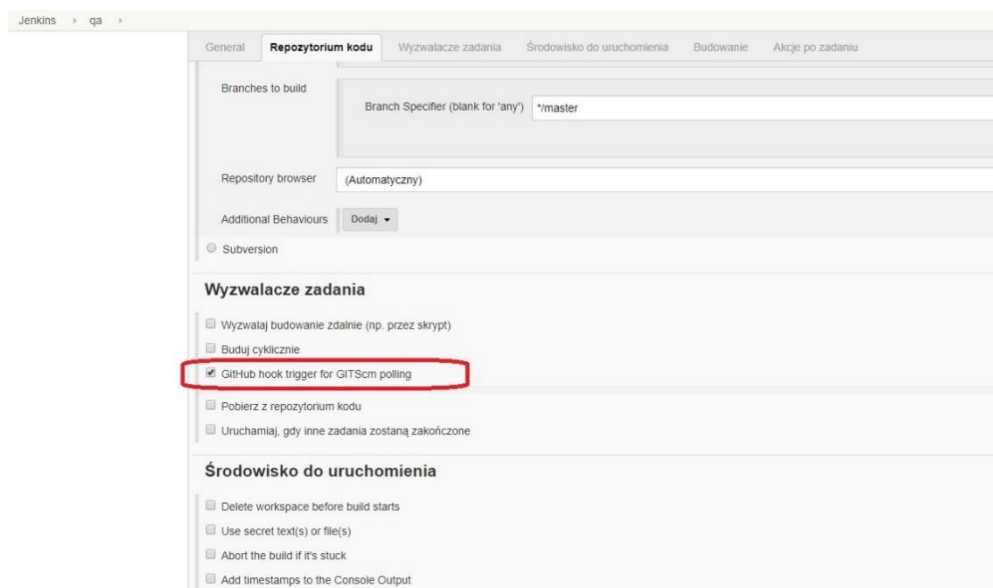
**Historia zadań** [trend](#)

szukaj X

#2 2019-07-07 14:12

[RSS Dla wszystkich](#) [RSS dla nieudanych](#)

Przechodzimy do sekcji Wyzwalacze zadania  
Zaznaczamy opcję: GitHub hook trigger for GITScm polling



Jenkins » qa »

General **Repository kodu** Wyzwalacze zadania Środowisko do uruchomienia Budowanie Akcje po zadaniu

Branches to build

Branch Specifier (blank for 'any') \*/master

Repository browser (Automatyczny)

Additional Behaviours Dodaj

Subversion

**Wyzwalacze zadania**

- ☐ Wyzwalaj budowanie zdalnie (np. przez skrypt)
- ☐ Buduj cyklicznie
- ☒ **GitHub hook trigger for GITScm polling**
- ☐ Pobierz z repozytorium kodu
- ☐ Uruchamiaj, gdy inne zadania zostaną zakończone

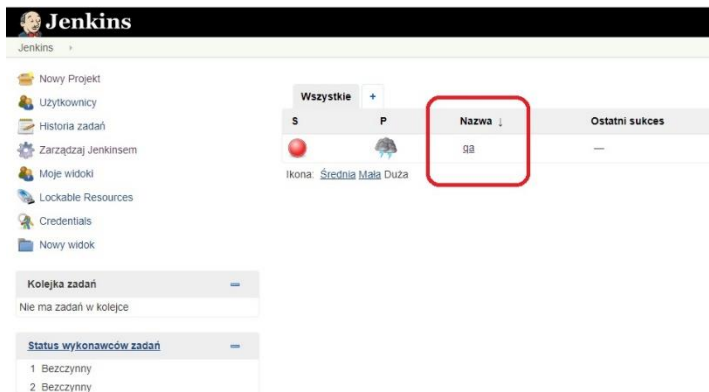
**Środowisko do uruchomienia**

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output

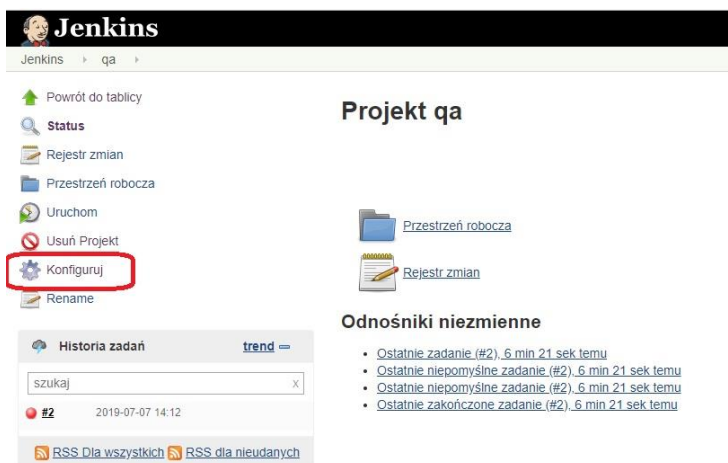
I następnie Zastosuj i Zapisz

## Trigger

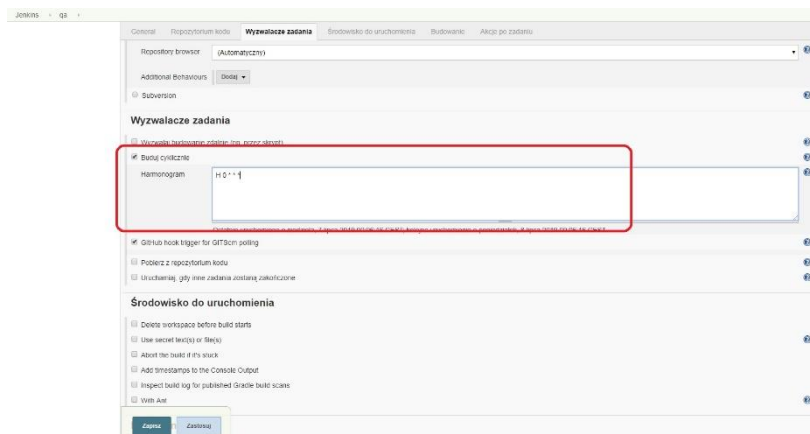
Ustawiamy trigger do automatycznego cyklicznego uruchamiania builda. Przechodzimy do Jenkinsa i wybieramy projekt



Wybieramy: Konfiguruj



Przechodzimy do sekcji Wyzwalacz zadania  
Zaznaczamy opcję: Buduj Cyklicznie



I następnie Zastosuj i Zapisz