

Git

Ściągawa

Git

Aby rozpocząć pracę z repozytorium należy najpierw utworzyć konto na bitbucket.org, oraz скачать aplikację git (for windows).

Git – generowanie klucza SSH

Podczas pracy z repozytorium za każdym razem gdy ubiegamy się o dostęp do niepublicznego repozytorium, otrzymujemy zapytanie o dane logowania (tj. Login/email oraz hasło). Można pozostać przy tej formie autoryzacji i wpisywać za każdym razem dane, oraz skorzystać mechanizmu wymiany kluczy ssh.

Po zainstalowaniu git'a, uruchamiamy terminal (git bash) i wpisujemy komendę:

ssh-keygen

Która w rezultacie zapyta nas kolejno o:

- Ścieżkę w której ma stworzyć klucz ssh (klik enter- pozostanie domyślna)
- Hasło dla klucza (tutaj jeśli klikniemy enter, to nie będziemy za każdym razem proszeni o hasło – to hasło nie jest w żaden sposób powiązane z hasłem logowania do bitbucket.org)
- Potwierdzenie hasła

Git – generowanie klucza SSH

Przykład użycia ssh-keygen:

```
[amen@warnencyk ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/amen/.ssh/id_rsa):
Created directory '/home/amen/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/amen/.ssh/id_rsa.
Your public key has been saved in /home/amen/.ssh/id_rsa.pub.
The key fingerprint is:
a1:c4:06:be:84:c9:d7:b3:7e:21:44:e7:d2:d3:53:2a amen@warnencyk
The key's randomart image is:
+--[ RSA 2048]----+
|   . . .
| . + = + . o
| ++ O E +
| o = = + .
| . + S
| ...
| ..
| .
+-----+
```

Git – generowanie klucza SSH

Teraz możemy użyć naszego klucza który znajduje się w katalogu użytkownika. Aby się do niego dostać musimy uruchomić wyświetlanie ukrytych plików i folderów:

<https://support.microsoft.com/pl-pl/help/14201/windows-show-hidden-files>

W katalogu głównym użytkownika(C:/Użytkownicy/NAZWA_UŻYTKOWNIKA/) znajduje się katalog **.ssh**. Wchodzimy do niego i powinniśmy tam znaleźć plik ***id_rsa.pub***. Otwieramy ten plik dowolnym edytorem tekstowym, a następnie kopujemy zawartość, która powinna wyglądać podobnie do :

ssh-rsa

```
AAAAB3NzaC1yc2EAAAQABAAQDBezG1sMoI67CqPmzyUcXt7UQFxdD/MpAB5Ei4G6
pflWy3Noq5YuUnmahYIaaFvsg92YWzqUjwLYio+XVTo79eZHynfpIXB1/z6I9ObvNp035lihhxjG2
VKpLfIN8WArAzAWzK1szjB7t0TfKhB6FfATNgDy32/NfoA5qQY+5jRnrjCjRZH4nx9NfnpDPIArtc
TfEei09O8RcLHHRVI1BNi09E0thCg2KF5v4BAd1bF3v+hk8fOXn01IXaCpCotLlc0BNQFpKhqWn2
t2mQ6i5R8fzizKA24INx4ImNaic4k+rDRMVeNPws9K7VFIRiOe+xY1wKqCmVUtcZGnKgsKI
amen@warnencyk
```

Git – generowanie klucza SSH

Następnie przechodzimy do serwisu bitbucket na którym posiadamy konto, logujemy się na nie i po prawej stronie interfejsu klikamy na **Bitbucket Settings**, a następnie na **SSH Keys**.

The screenshot shows the Bitbucket Settings page for a user named Paweł Reclaw. The left sidebar contains a navigation menu with sections like General, Account settings, Plans and Billing, Access Management, Security, and Integrations and Features. The 'Account settings' section is currently selected. The main content area displays 'Account settings' with fields for Avatar (a blue placeholder icon), Full name (Paweł Reclaw), and a note to update details through the Atlassian account. Below this is the 'Bitbucket profile settings' section, which includes fields for Username (saintamen), Website (empty field), Language (English), and a Private profile checkbox. There are also links for Help, Hide details, and Delete account. At the bottom, the 'Preferences' section is shown with two sections: Shortcuts (checkbox for Enable keyboard shortcuts) and Console (checkbox for Enable console messages). A large blue 'Update' button is at the bottom right.

Settings Paweł Reclaw

GENERAL

Account settings

Email aliases
Notifications

PLANS AND BILLING

Plan details
Git LFS

ACCESS MANAGEMENT

User groups
OAuth
App passwords
Access controls Premium

SECURITY

SSH keys
Two-step verification
Connected accounts
Sessions
Audit log

INTEGRATIONS AND FEATURES

Labs
Find integrations
Manage integrations

Find a repository...

Paweł Reclaw
pawel.reclaw@gmail.com

Manage Atlassian account
View profile
Bitbucket settings
Integrations
Log out

Account settings

Avatar

Full name Paweł Reclaw

Update these details through your Atlassian account.

Bitbucket profile settings

Username [saintamen \(change\)](#)

Website

Language English

Help translate Bitbucket into your language.
 Private profile
Hide the details on your profile page from all other users.

Delete account

Preferences

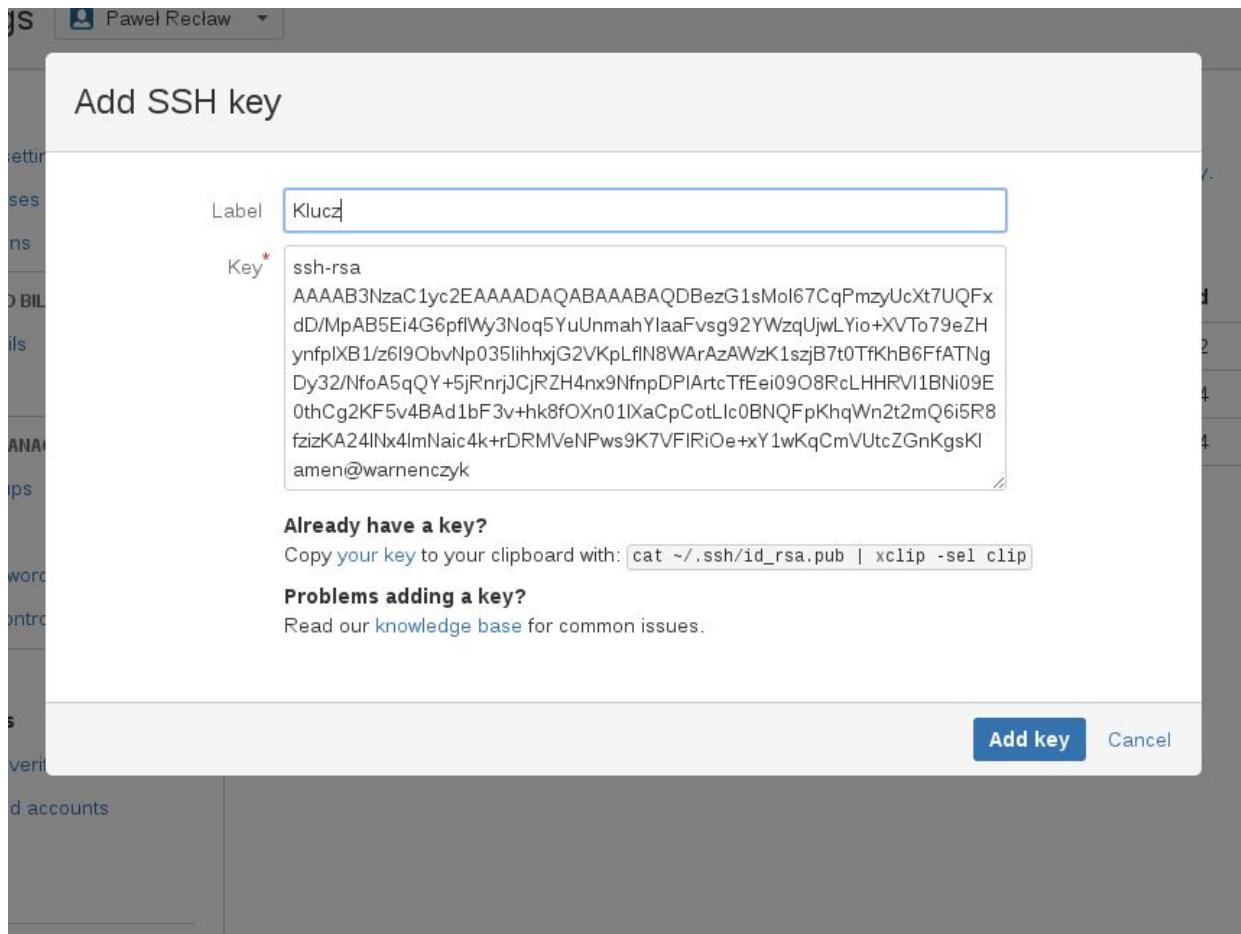
Shortcuts Enable keyboard shortcuts
Press your ? key for a list of shortcuts.

Console Enable console messages
Disable to prevent Bitbucket from sending nonurgent console messages.

Update

Git – generowanie klucza SSH

Następnie klikamy **Add key** i wklejamy wcześniej skopiowaną zawartość pliku:



Git – generowanie klucza SSH

Klikamy **Add key** i od tego momentu cieszymy się, że nie ma konieczności wprowadzania loginu i hasła do konta przy każdym pull i clone repozytorium.

Git - clone

Komenda klonowania repozytorium to :

git clone git@dres.com:NazwaRepozytorium.git

Domyślnie git pobiera/klonuje repozytorium do katalogu o nazwie ***NazwaRepozytorium***. Dodatkowo do komendy można dodać jeszcze jeden parametr tym parametrem jest nazwa katalogu do którego git ma sklonować repozytorium (zamiast utworzyć katalog o nazwie ***NazwaRepozytorium*** stworzy katalog o podanej przez nas nazwie)

Przykład:

git clone git@dres.com:NazwaRepozytorium.git Katalog

Spowoduje wykonanie klonu repozytorium do nowego folderu o nazwie ***Katalog***.

Git - status

Komenda sprawdzania stanu repozytorium:

git status

Ta komenda wyświetla nam stan repozytorium. Musimy się jednak znajdować w katalogu który jest śledzony przez git. Przykładowy status:

```
[amen@warnenczyk Repository]$ git status  
# On branch AD-68  
nothing to commit, working directory clean
```

Po zmodyfikowaniu plików git status informuje nas o zmianach na plikach w repozytorium. Przykład wyjścia po zmianie pliku README.md:

```
# On branch AD-68  
# Untracked files:  
#   (use "git add <file>..." to include in what will be committed)  
#  
#       README.md  
nothing added to commit but untracked files present (use "git add" to track)
```

Git - status

Komenda sprawdzania stanu repozytorium:

git status

Po otrzymaniu komunikatu że zmiany niektórych plików nie są śledzone (Untracked files) należy dodać je komendą **git add** (o tym za chwilę).

Kiedy dodamy plik i jest śledzony git status zwróci nam następujący komunikat:

On branch AD-68

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

#

new file: README.md

#

W tym przypadku mamy informację o nowym pliku. Git zwraca nam również listę plików zmodyfikowanych oraz usuniętych (jeśli takie były).

Git - add

Komenda dodawania plików do git'a (aby zaczął je śledzić).

git add nazwaPliku

Komenda spowoduje, że git zacznie śledzić zmiany wybranego przez nas pliku. Przed wykonaniem commita konieczne jest dodanie listy plików (poprzez tą komendę) do git'a – aby je śledził. Tylko śledzone pliki zostaną potraktowane jako zmiany które będą zapisane przez commit.

Aby commit zapisał stan pliku, musisz go dodać komendą **add**.

Opcjonalnie, zamiast dopisywać wszystkie pliki które chcielibyśmy śledzić, możemy dodać wszystkie zmiany, używając komendy:

git add --all

Git - config

Komenda konfiguracyjna:

git config --global user.name "imie nazwisko"
git config --global user.email "nasz@email"

Jednorazowo po zainstalowaniu gitu konieczne jest jego skonfigurowanie. Powyższe komendy są potrzebne do wykonania commita (należy najpierw wykonać config, następnie commit). Te informacje zapisują się w katalogu gitu na naszych komputerach. Dane służą do identyfikacji autora commita na serwerze:

Author	Commit	Message
 Paweł Reclaw	9499781b222	First implementation, only created files.

Git - commit

Komenda tworzenia commita:

git commit –am “Komentarz do commita”

Komenda spowoduje utworzenie snapshota/checkpoint'a stanu kodu. Dzięki temu możliwy jest powrót do tego stanu kodu w przyszłości. Każdy commit generuje unikalne ID po którym będziemy mogli się odwołać w celu resetu(cofnięcia zmian) kodu.

Ta komenda nie powoduje wypchnięcia zmian na serwer. Po wykonaniu tej komendy zostaje jedynie stworzony snapshot w lokalnych katalogach.

Flagi użyte to ‘a’ – aby dodać wszystkie zmiany (poza nowo utworzonymi plikami) do śledzonych, oraz ‘m’ – oznacza że komentarz do commita zostanie podany po spacji.

Git - push

Komenda wypchnięcie zmian na serwer:

git push

Komenda spowoduje wypchnięcie wszystkich lokalnych commitów na serwer. Jeśli od czasu ostatniego pusha wystąpił więcej niż jeden commit, to zostaną wypchnięte wszystkie zmiany. Komenda wymaga, aby stan lokalny repozytorium był aktualny (można to zrobić komendą ‘git pull’).

Jeśli na serwerze nie istnieje branch na którym się znajdujemy, konieczne jest użycie dodatkowych parametrów, aby sprecyzować nazwę brancha który ma być utworzony na serwerze:

git push -u origin nazwa_brancha

np.

git push -u origin master

Git - checkout

Komenda przechodzenia między branchami:

git checkout branch_name

Komenda spowoduje przejście z aktualnego branch'a na branch **branch_name**. Skutkiem wykonania komendy będzie podmiana zawartości wszystkich plików na to, co znajduje się na branchu **branch_name**.

Jeśli chcemy przejść na nowy (stworzyć) branch, należy dodatkowo użyć flagi **-b** która spowoduje jego utworzenie (jeśli taki branch nie istnieje). Przykład:

git checkout -b branch_name

Kiedy znajdujemy się na nowo stworzonym branchu należy pamiętać, że tworzymy go jedynie lokalnie, dopóki nie wypchniemy zmian na serwer komendą **git push**. Jeśli na serwerze dany branch nie istnieje, to należy go ustawić używając komendy:

git push -u origin branch_name

Git – tworzenie repozytorium

Po stworzeniu repozytorium na serwerze (zainicjowaniu repozytorium przez przeglądarkę), mamy dwie opcje rozpoczęcia pracy z repozytorium. Jeśli posiadamy kod, który chcemy zacząć śledzić, możemy lokalnie uworzyć w katalogu repozytorium i połączyć je z serwerem zdalnym:

```
cd FOLDER_PROJEKTU
git init
git add --all
git commit -m "Initial Commit"
git remote add origin ssh://git@ADRES_SERWERA:PORT/NAZWA_REPO.git
git push -u origin master
```

Git jest bardzo przyjazny użytkownikowi i w większości przypadków podpowiada nam o komendach jakich musimy użyć.

Git – tworzenie repozytorium

Drugą opcją tworzenia repo jest pobranie gołego repozytorium (pustego folderu) z serwera, czyli nic innego jak po prostu sklonowanie repozytorium. Zostaniemy poinformowani o tym, że repozytorium jest puste:

```
[amen@warnencyk Repository]$ git clone ssh://git@SERWER:PORT/NAZWA_REPO.git  
Cloning into 'ad'...  
warning: You appear to have cloned an empty repository
```

Od tego momentu możemy do niego przenieść nasze źródła lub w tym folderze rozpoczęć pracę.

Git – tworzenie repozytorium

Przykładowy screen z repozytorium bitbucket:

The screenshot shows a Bitbucket repository page for a new, empty repository. The top navigation bar includes 'Bitbucket', 'Projects', and 'Repositories'. On the left, there's a sidebar with icons for a profile picture, download, repository list, and settings.

You have an empty repository

To get started you will need to run these commands in your terminal.

```
git config --global user.name "Paweł Reclaw"  
git config --global user.email "pawel.reclaw@gmail.com"
```

Configure Git for the first time

Working with your repository

I just want to clone this repository

If you want to simply clone this empty repository then run this command in your terminal.

```
git clone ssh://git@aps-drones.com:7999/ad/ad.git
```

My code is ready to be pushed

If you already have code ready to be pushed to this repository then run this in your terminal.

```
cd existing-project  
git init  
git add --all  
git commit -m "Initial Commit"  
git remote add origin ssh://git@aps-drones.com:7999/ad/ad.git  
git push -u origin master
```

My code is already tracked by Git

If your code is already tracked by Git then set this repository as your "origin" to push to.

```
cd existing-project  
git remote set-url origin ssh://git@aps-drones.com:7999/ad/ad.git  
git push -u origin master
```

All done with the commands?

Refresh