

## GITHUB alternative setup using GIT Bash

- Download and install GIT BASH from : <https://git-scm.com/downloads> (for windows 64 bit)

- Once git bash is installed, **check for existing SSH keys** using the following steps:

1. Open Git Bash.
2. Enter ( `ls -al ~/.ssh` ) to see if existing SSH keys are present:

```
$ ls -al ~/.ssh
# Lists the files in your .ssh directory, if they exist
```

3. Check the directory listing to see if you already have a public SSH key.

By default, the filenames of the public keys are one of the following:

- `id_dsa.pub`
- `id_ecdsa.pub`
- `id_ed25519.pub`
- `id_rsa.pub`
- If you don't have an existing public and private key pair, or don't wish to use any that are available to connect to GitHub, then generate a new SSH key.
- If you see an existing public and private key pair listed (for example `id_rsa.pub` and `id_rsa`) that you would like to use to connect to GitHub, you can add your SSH key to the ssh-agent.

- If ssh key does not exists, follow the next **steps to create SSH key** :

- Open Git Bash.
- Paste the text below, substituting in your GitHub Enterprise email address.
  - `$ ssh-keygen -t rsa -b 4096 -C "your_email@in.ibm.com"`
  - This creates a new ssh key, using the provided email as a label.
- Generating public/private rsa key pair.
- When you're prompted to "Enter a file in which to save the key," press Enter. This accepts the default file location.
- Enter a file in which to save the key (`/c/Users/you/.ssh/id_rsa`):*[Press enter]*
- At the prompt, type a secure passphrase.
  - `> Enter passphrase (empty for no passphrase):` *[Type a passphrase]*
  - `> Enter same passphrase again:` *[Type passphrase again]*

➤ **Adding ssh keys to ssh agent :**

- Ensure the ssh-agent is running:
  - If you are using the Git Shell that's installed with GitHub Desktop, the ssh-agent should be running.
  - If you are using another terminal prompt, such as Git for Windows, you can start it manually:

```
○ # start the ssh-agent in the background
○ $ eval $(ssh-agent -s)
○ Agent pid 59566
```

- Add your SSH private key to the ssh-agent. If you created your key with a different name, or if you are adding an existing key that has a different name, replace *id\_rsa* in the command with the name of your private key file.

```
○ $ ssh-add ~/.ssh/id_rsa
```

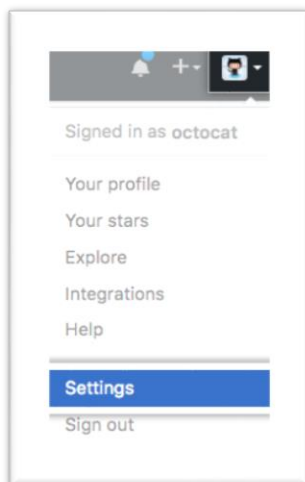
➤ **Adding the SSH key to your GitHub account :**

Copy the SSH key to your clipboard.

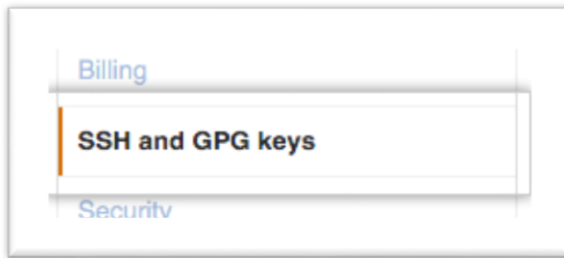
- If your SSH key file has a different name than the example code, modify the filename to match your current setup. When copying your key, don't add any newlines or whitespace.

```
○ $ clip < ~/.ssh/id_rsa.pub
○ # Copies the contents of the id_rsa.pub file to your clipboard
```

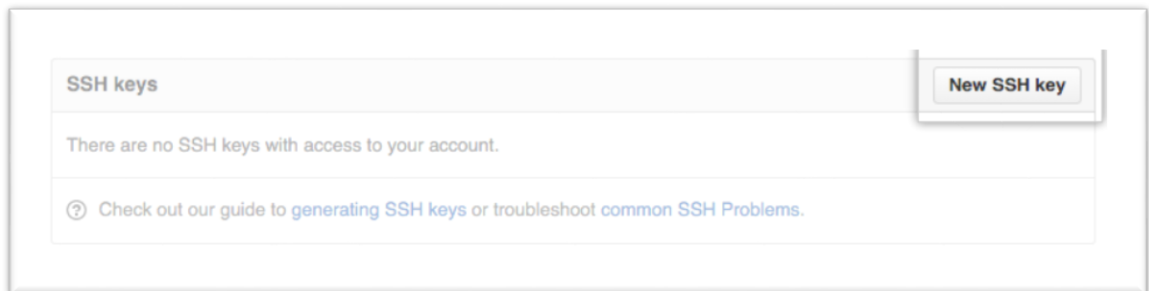
1. In the upper-right corner of any page, click your profile photo, then click **Settings**.



2. In the user settings sidebar, click **SSH and GPG keys**.



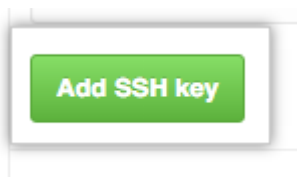
3. Click **New SSH key** or **Add SSH key**.



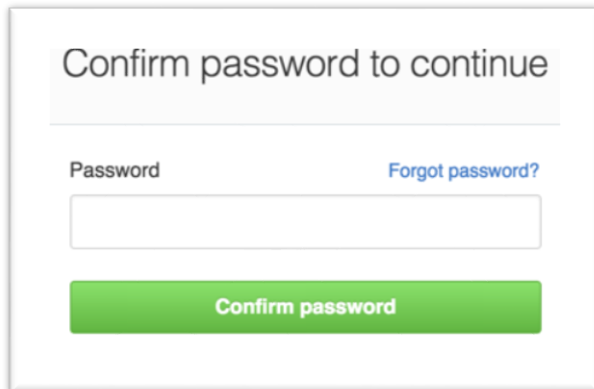
4. In the "Title" field, add a descriptive label for the new key. For example, if you're using a personal Mac, you might call this key "Personal MacBook Air".
5. Paste your key into the "Key" field.

A screenshot of the 'SSH keys' form. At the top left is the text 'SSH keys' and at the top right is a button 'New SSH key'. Below this is a message: 'There are no SSH keys with access to your account.' The form has two main sections. The first section is labeled 'Title' and contains an empty text input field. The second section is labeled 'Key' and contains a large text area with a placeholder text: 'Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521''. Below the 'Key' field is a green button labeled 'Add SSH key'. At the bottom, there is a link with a question mark icon: 'Check out our guide to generating SSH keys or troubleshoot common SSH Problems.'

6. Click **Add SSH key**.



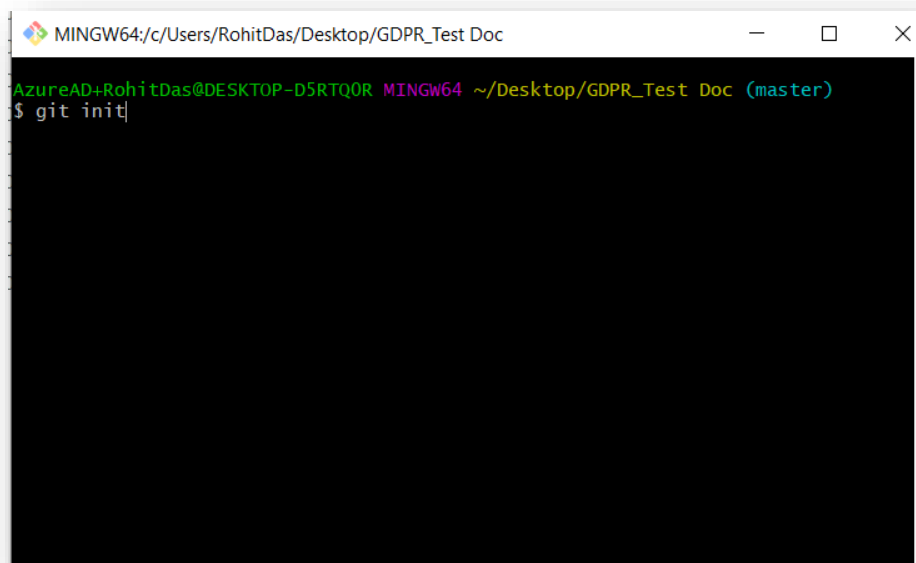
7. If prompted, confirm your GitHub Enterprise password.

A light gray rectangular form with a white border. At the top, it has a title "Confirm password to continue" in a light gray box. Below the title, the word "Password" is followed by a blue link "Forgot password?". Underneath is a white rectangular input field. At the bottom of the form is a green button with the text "Confirm password" in white.

#### ➤ Syncing Git and Github :

After setting up the ssh key using the above mentioned steps, first initialize an empty folder as git repo by going inside the directory and typing “git init” from git bash :

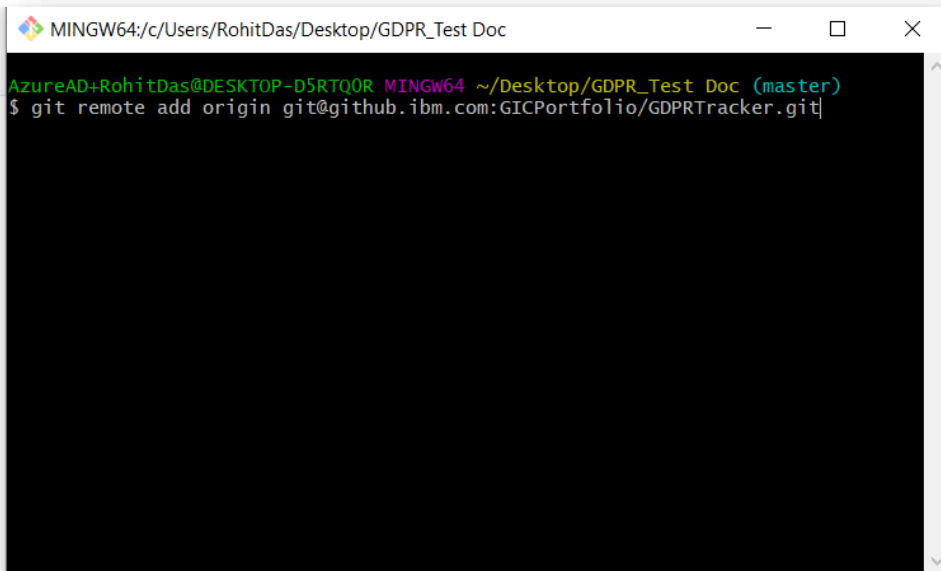
For example :

A screenshot of a Windows command prompt window. The title bar shows the path "MINGW64:/c:/Users/RohitDas/Desktop/GDPR\_Test Doc". The command prompt shows the user "AzureAD+RohitDas@DESKTOP-D5RTQ0R" in a "MINGW64" shell at the directory "~/Desktop/GDPR\_Test Doc" on the "(master)" branch. The command "git init" has been entered and is being executed.

- Then after initialising the folder as git repo run the following command **once** to link the CENTRAL repository to that folder :

- **Git remote add origin <ssh\_link>**

For example :



```

MINGW64:/c/Users/RohitDas/Desktop/GDPR_Test Doc
AzureAD+RohitDas@DESKTOP-D5RTQ0R MINGW64 ~/Desktop/GDPR_Test Doc (master)
$ git remote add origin git@github.ibm.com:GICPortfolio/GDPRTracker.git

```

- Now you can use the following commands for performing pull, push, etc :












- **git remote -v :** To check fetch and pull
- **git pull origin master :** To pull origin's master data branch to local system
- **git push origin master:** To push origin's master data branch from local system
- **git status :** To check the status such as untracked/modified/staged,etc

#### ❖ **NOTE :**

- 🔧 Always perform pull before push to avoid conflicts
- 🔧 Any changes in the local git repo must be staged and committed for reflecting in the central repository

#### ▪ **Other basic git bash commands :**

- 🔧 **git add <filename.extension> :** To place a single file to staging area
- 🔧 **git add -A :** To place multiple (all) files to staging area
- 🔧 **git reset HEAD <filename.extension> :** To unstage file from staging area
- 🔧 **git commit -m "commit\_message" :** To commit the files added to staging area
- 🔧 **git commit -a -m "commit\_message" :** To directly commit modified files without adding them to staging area
- 🔧 **git rm <filename.extension> :** To remove the file from working directory as well as staging area

 <b>git rm - - cached &lt;filename.extension&gt; :</b>	To remove file from staging area only
 <b>git checkout :</b>	Matches the working directory with last commit
 <b>git checkout &lt;filename.extension&gt; :</b>	Matches the working directory with last commit for a particular file
 <b>git checkout -f :</b>	Matches the working directory with last commit for all files
 <b>git log :</b>	To check all commits
 <b>git log -p -5:</b>	To check last 5 commits
 <b>git diff :</b>	To compare working directory with staging area
 <b>git diff - - staged :</b>	To compare staging area with last commit
 <b>git branch &lt;branch_name&gt; :</b>	To create a branch from master
 <b>git branch :</b>	To see the branches present
 <b>git checkout &lt;branch_name&gt; :</b>	To switch to other branches