**Diyotta Integration**

**with GitHub**

Contents

[Document Overview 3](#_Toc30516941)

[1.1 Purpose of this Document 3](#_Toc30516942)

[Prerequisites 3](#_Toc30516943)

[Creation of GitHub account 3](#_Toc30516944)

[Git Setup in Local Machine 6](#_Toc30516945)

[Process setup for export and Import script 11](#_Toc30516946)

[Export script Configuration 12](#_Toc30516947)

[Import script Configuration 16](#_Toc30516948)

# Document Overview

# 1.1 Purpose of this Document

This document explains about the creation of GIT HUB account, GIT HUB setup process in local machine, Migration of job flows from diyotta to GITHUB repository and vice-versa using the export and import scripts.

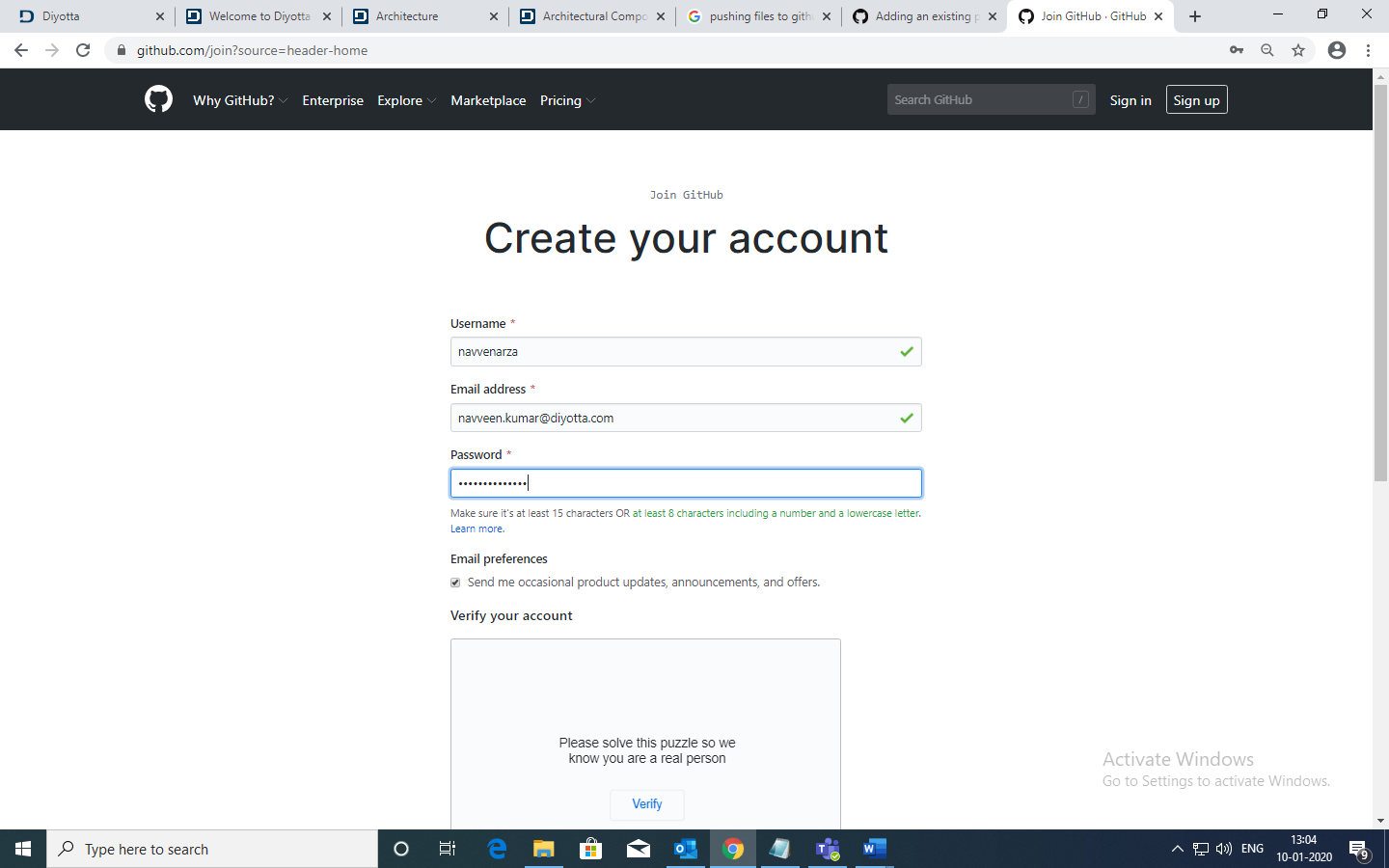
# Prerequisites

* Git Hub access.
* Server access.
* Diyotta suite access.

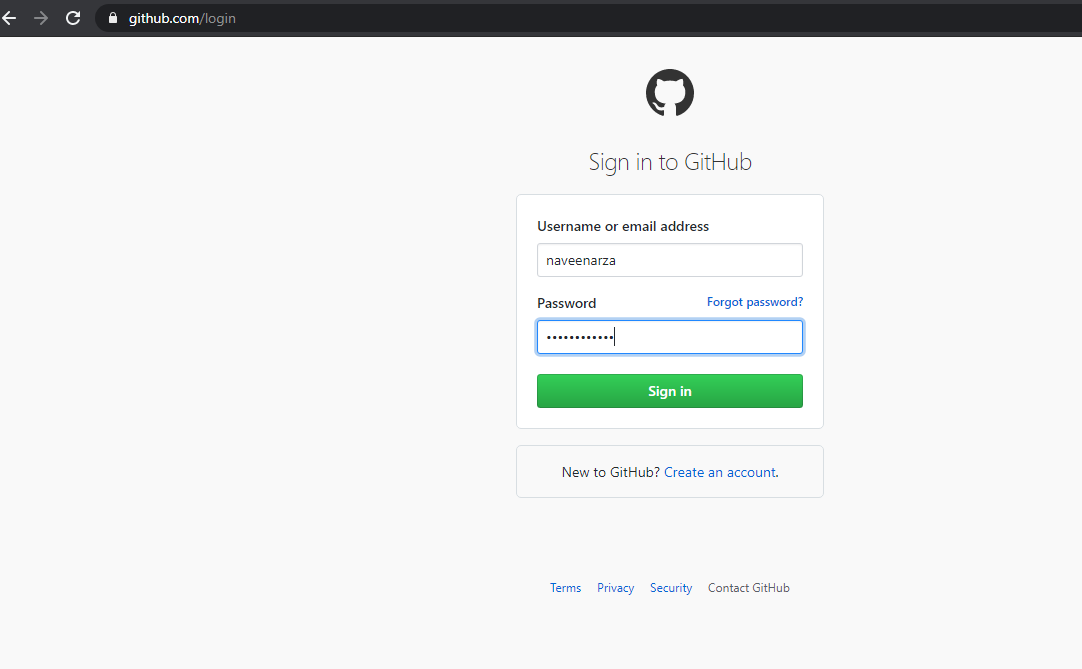
# Creation of GitHub account

**Step 1:** Navigate to<https://github.com/>

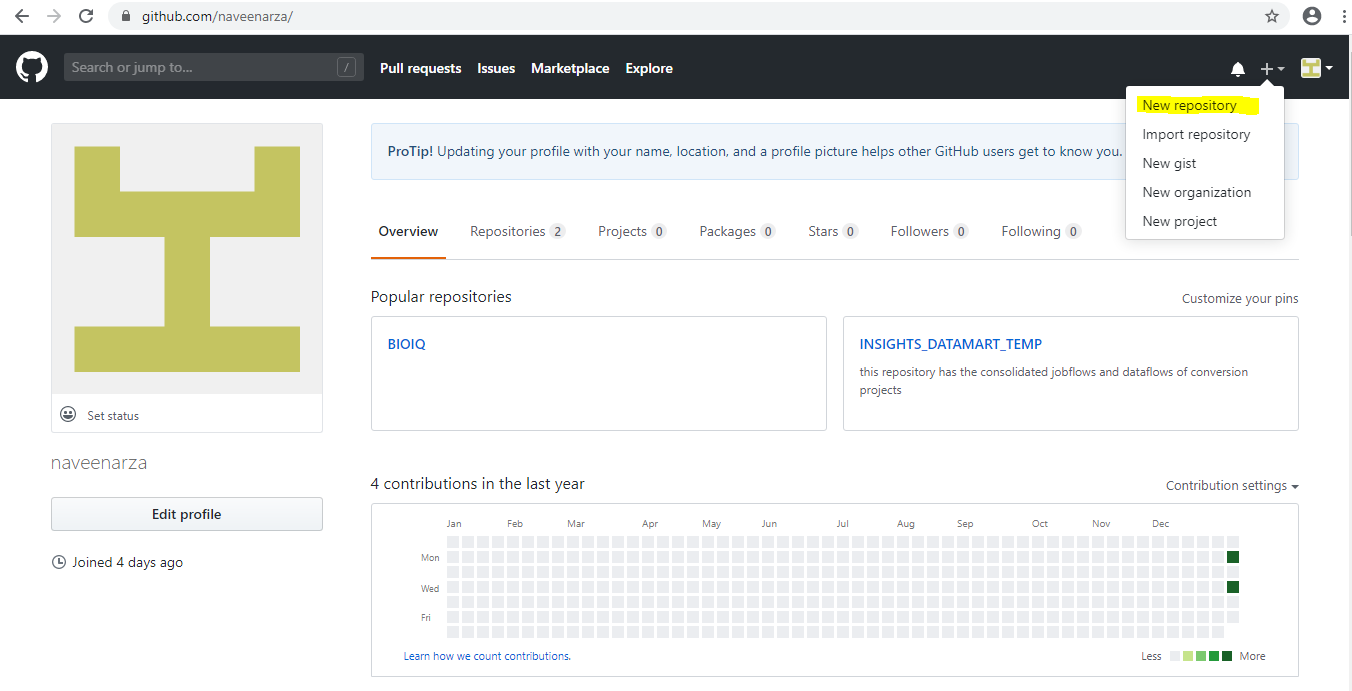
**Step 2**: Create GitHub account using emailid.



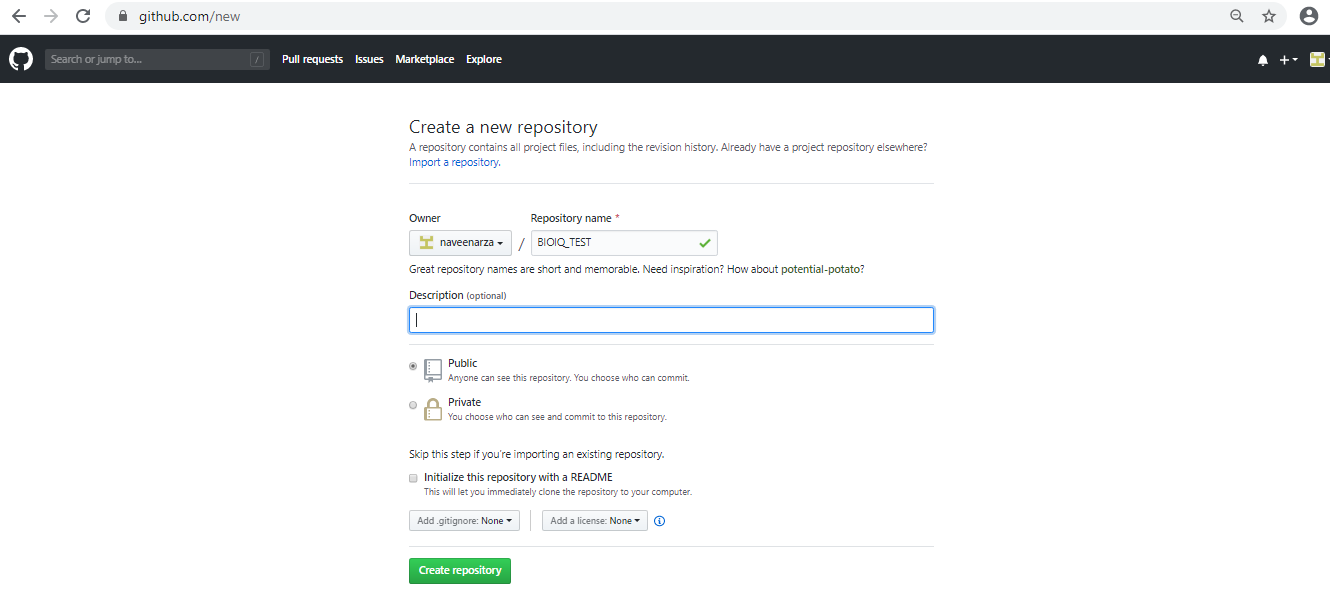
**Step 3:** Login to your GitHub account



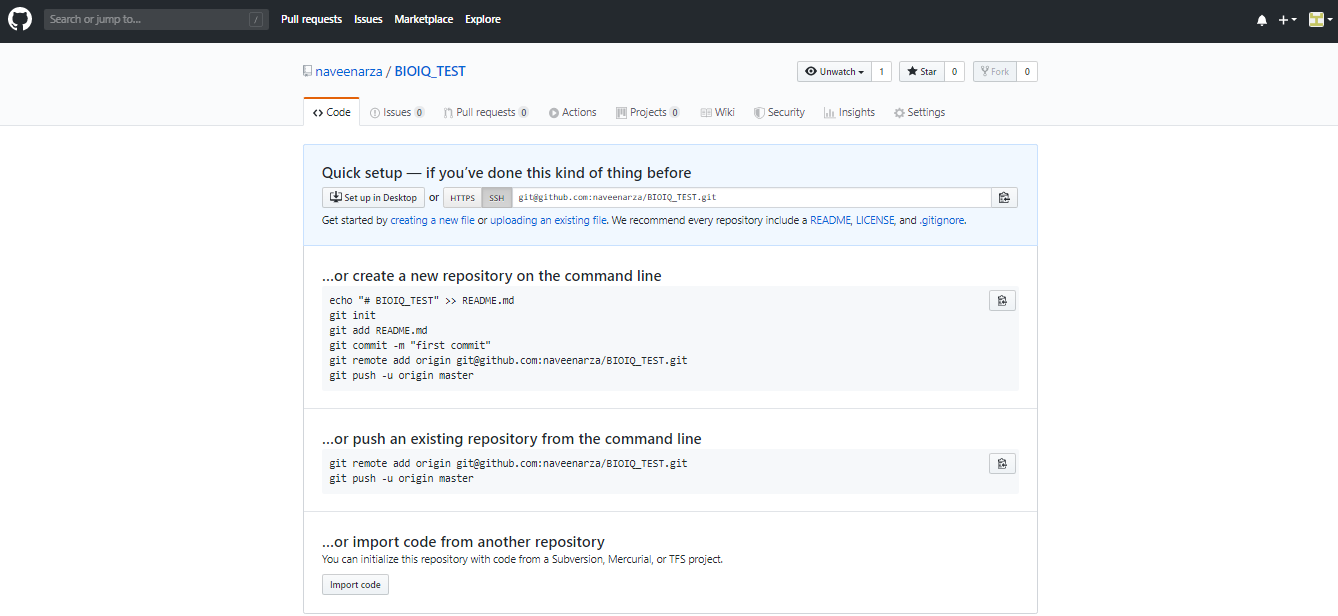
**Step 4:** Create a new repository as shown in the below screenshot.



**Step 5:** Enter the name of the repository and click on Create Repository button.



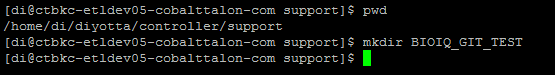
**Step 6:**  It will navigate to the home page as shown in the below screenshot.



# Git Setup in Local Machine

**Step 1:** Create a directory in your local machine where we will place all the code. We will convert this local project to a Git repository in the further steps.

**Syntax: mkdir <directory\_name>**

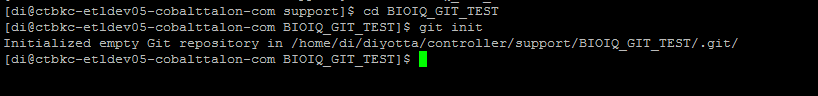


In the above step we have created a directory **BIOIQ\_GIT\_TEST** in the controller path.

**Step 2:** Change the current working directory to the project directory **BIOIQ\_GIT\_TEST** and initialize the local repository as a git repository.

**Syntax: git init**

* command initializes the local repository as a git repository.

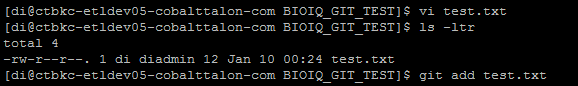


**Note:** If the project directory already exists then ignore first step.

**Step 3:** Create/Place the files which needs to be moved to remote GitHub repository.

**Syntax: git add** **<filename>**

* The above command adds the files to the staging area in the local machine.



**Note:** To move all the files to the staging area use **(git add .**)command.

**Step 4:** Setup the global username and email configuration for the local git repository.

**Syntax:**

**git config user.name “name”**

**git config user.email “@Email.com”**

**A close up of a screen

Description automatically generated**

**Note**:

Below commands are used to check the Global username and email which is existing.

1.To check the username of the local git repository

**git config user.name**

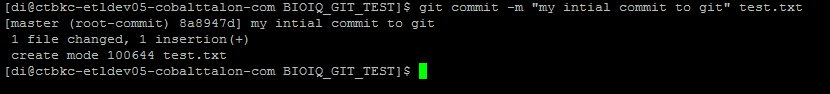
**2**.To check the email of the local git repository

**git config user.email**

**Step 5:** Commits the staging files in the local repository.

**Syntax:** **git commit -m “Commit Message” <filename>**

* Commits the tracked changes and prepares them to be pushed to a remote repository.



**Note:** To commit all the files in the local repository use (**git commit -m “Commit Message”**).

**Step 6:** Adding Remote GIT URL to the local repository. This can be done either using **SSH** or **HTTPS** URL**.**

**To get the HTTP/SSH URL:**

We can get the **HTTP/SSH URL** in two ways.

1. When the files are not present in the GIT repository, use the below path.

Go to repository 🡪 Click on code 🡪Click on SSh/http as shown in screenshot below.

**A screenshot of a cell phone

Description automatically generated**

1. When files are present in the GIT repository, use the below path.

GOTO repository 🡪 Click on code 🡪Click on Clone or download option in the right side as shown in the below screenshot.

A screenshot of a computer

Description automatically generated

**Example for SSH, HTTPS URL’s**

**SSH URL:** git@github.com:naveenarza/BIOIQ\_TEST.git

**HTTPS URL :** <https://github.com/navvenarza/BIOIQ_TEST>

**Note:**

If we use **SSH** URL then we don’t need to type password every time we push code to GitHub.

If we use the **HTTPS,** then GitHub prompts for password every time.

**To Check the SSH Key whether it Is Existing or Not.**

1. Type cd command to change the directory to .ssh folder.

**cd ~/.ssh**

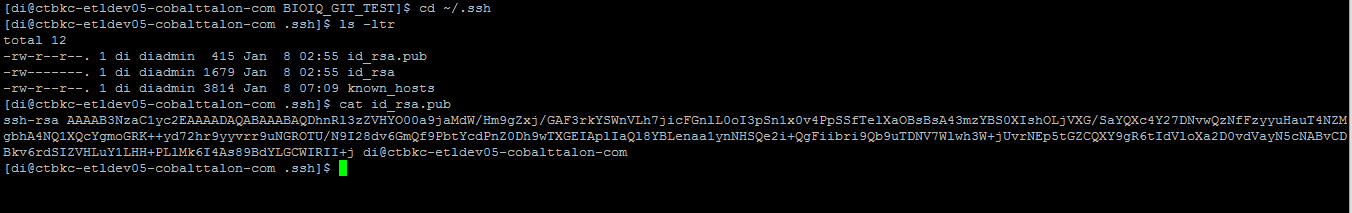
1. To View the files in the .ssh directory.

**ls -ltr**

1. Display the content of the file.

**cat id\_rsa.pub**

1. Copy the public key which is present in id\_rsa.pub



**Generation of new SSH Key if it doesn’t exist:**

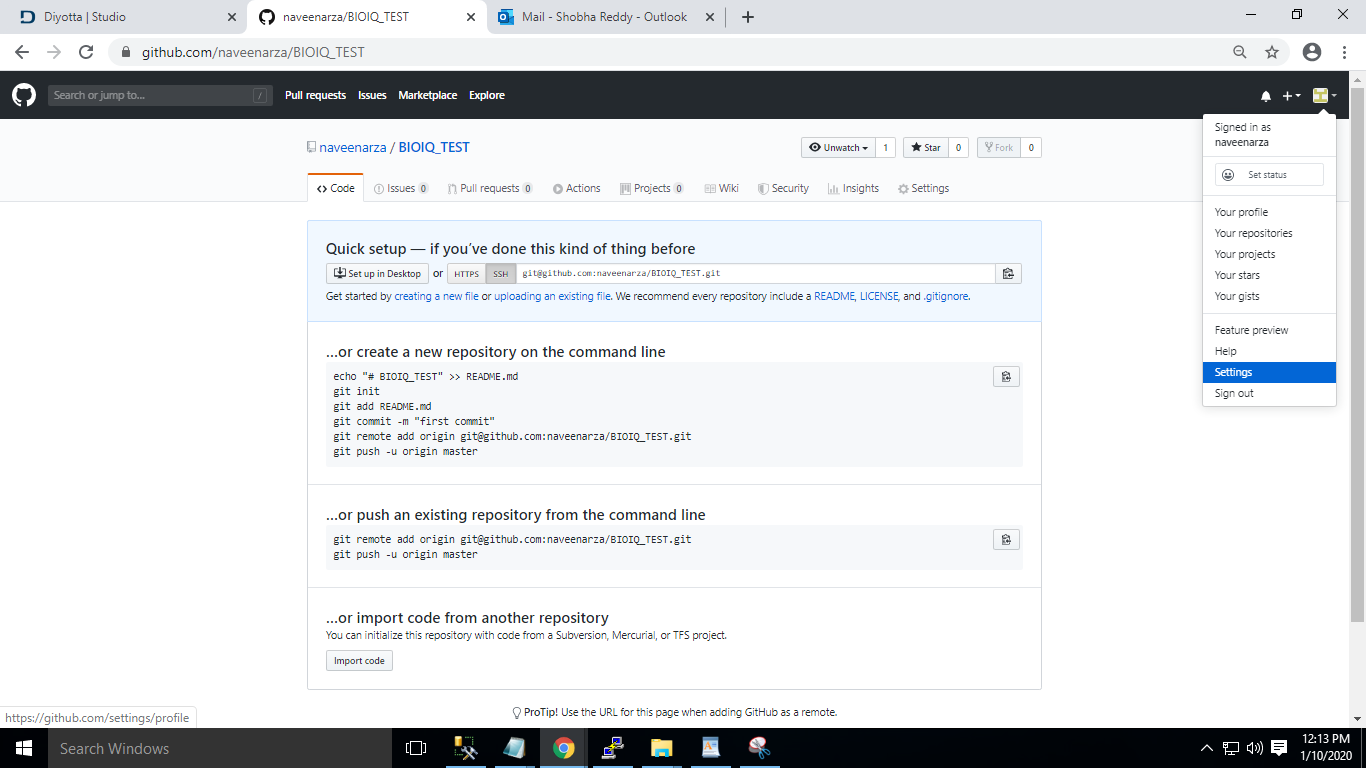
1. Run the below command to generate the new SSH key.

**ssh-keygen -C "naveen.kumar@diyotta.com" -t rsa**

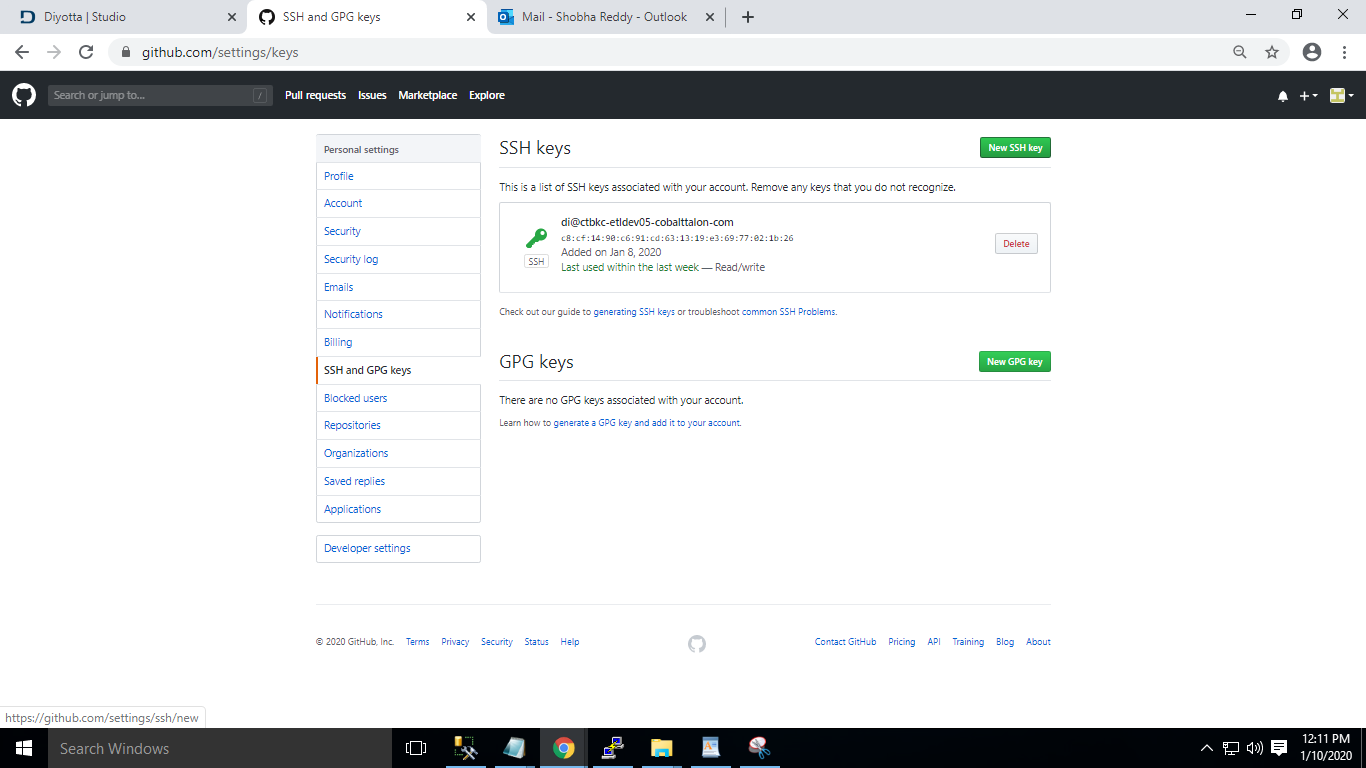
1. When you're prompted to "Enter a file in which to save the key," press Enter. This accepts the default file location. Press enter key for 3 times.
2. Navigate to the .ssh directory in the local machine as shown above and copy the public key from the file **id\_rsa.pub**.

Follow the below steps to add the SSH Key to the Remote Git repository.

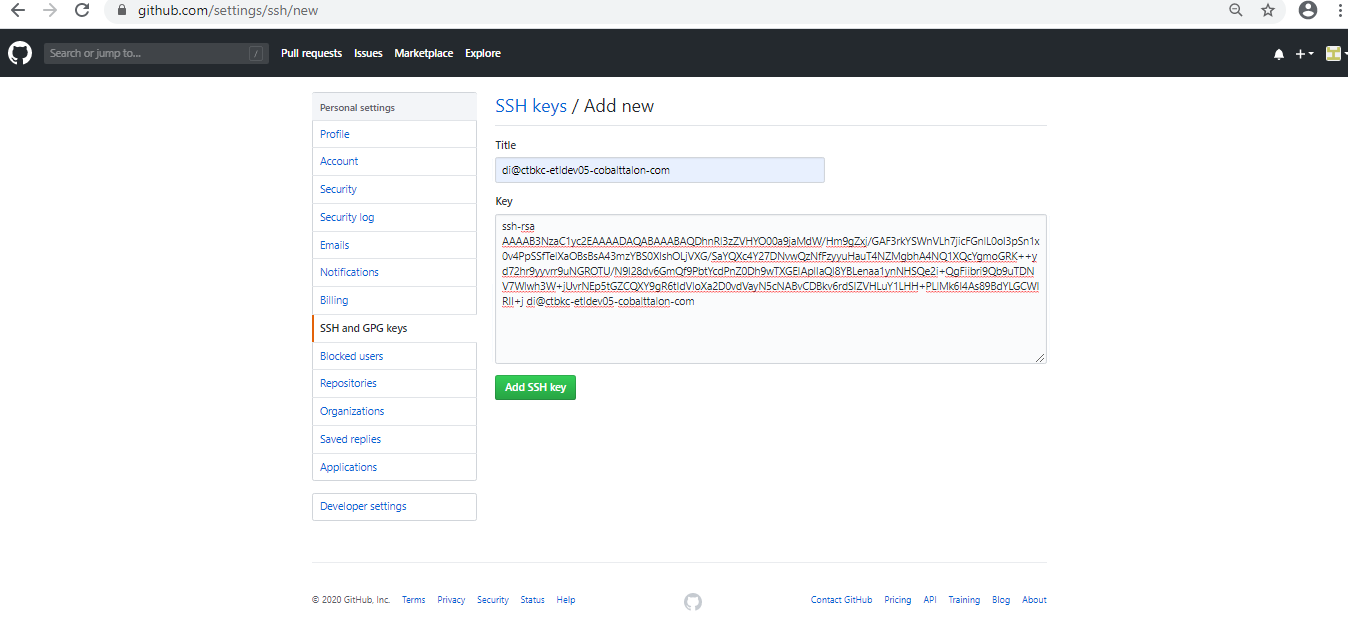
1. Go to the settings as shown in the below screenshot.



1. Navigate to SSH and GPG Keys section and click on New SSH Key button.



1. Click on Add SSH key enter the title and Paste the copied public key in the key section as below.



**Step 5**: Adding Remote GIT URL to the local repository.

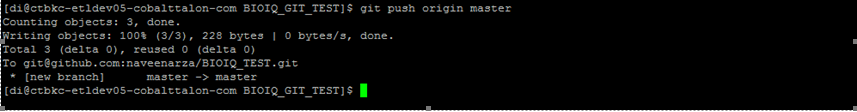
**Syntax:** git remote add origin <git URL(SSH/HTTPS)>

**git remote add origin** [**git@github.com:naveenarza/BIOIQ\_TEST.git**](mailto:git@github.com:naveenarza/BIOIQ_TEST.git)

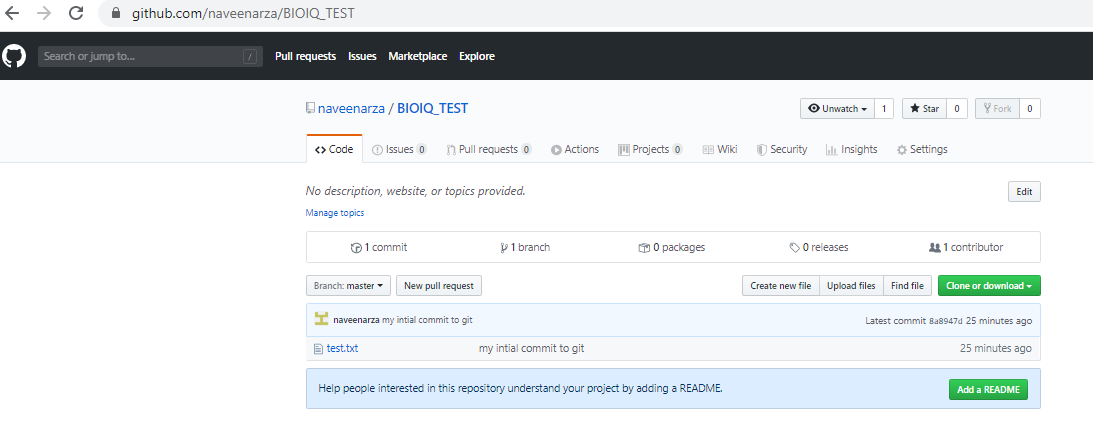


**Step 6:** Pushing the changes from local to the remote repository.

**Syntax: git push origin master**



**Step 7:** Validate whether the code is pushed to the remote Git repository.



# Process setup for export and Import script

Place the below files in the mentioned path.

**diy\_export.sh Location:**

/home/di/diyotta/controller/support/git\_scripts/

**diy\_import.sh Location:**

/home/di/diyotta/controller/support/git\_scripts/

**Git Location:**

/home/di/diyotta/controller/support/BIOIQ\_GIT/

**Env.properties file location:**

/home/di/diyotta/controller/support/conf/

**diy\_functions.sh location:**

/home/di/diyotta/controller/support/lib/migration/

****

**Note 1**: Update the local git location in **env.properties** file and make sure that all the paths mentioned in the properties file exists in your setup.

**Note 2:** Update the temporary path in the script **diy\_export.sh** at **224** line accordingly to temp directory location (/home/di/diyotta/controller/support/tmp ).

**Below are the jobflows which will run the GIT Import and Export scripts. We have to run/schedule the jobs to catchup the changes every time.**

Import the attached json files to the environment you are going to automate the export and import the GitHub process.

1. **Jf\_git\_export** :

This jobflow is used to export all the jobflows in the mentioned project**.**

1. **Jf\_git\_export\_jobflows** :

This jobflow is used to export the specific jobflows which are mentioned in **exportlistfile.dat** file.

1. **Jf\_git\_import :**

This jobflow is used to import the jobflows which are mentioned in **importlistfile.dat** file.

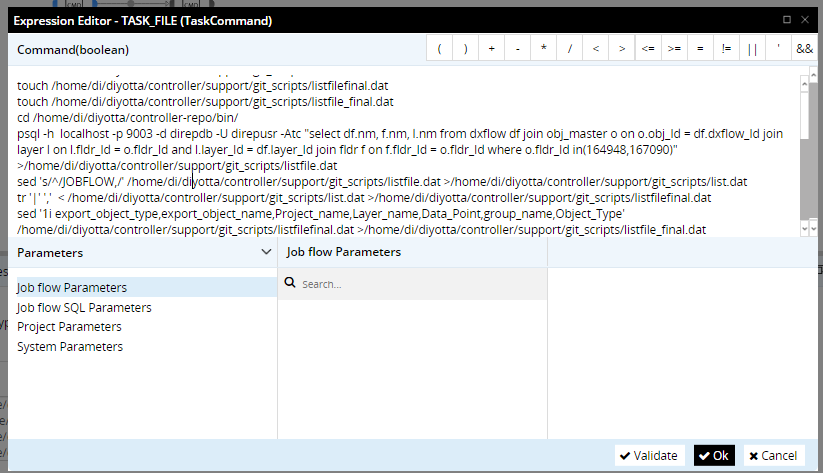
# Export script Configuration

The export script will get all json files of the jobflows from Diyotta metadata and pushes them to the GIT remote repository.

1. **Working of Jf\_git\_export** : This jobflow is used to export all the jobflows in mentioned project\_id’s in jobflow parameter.

1. **Task\_FILE :** Modify the Metadata details with respective environment.

Psql -h localhost -p 9003 -d direpdb -U direpusr -ATC “ select df.nm,f.nm,l.nm from dxflow df join obj\_master o on o. obj\_id = df.dxflow\_id join layer l on l.fldr\_id = o.fldr\_id and l.layer\_id = df.layer\_id join fldr f on f. fldr\_id=o.fldr\_id where o.fldr\_id in ($FS\_Project\_id)



1. **Modify the below jobflow parameters accordingly.**

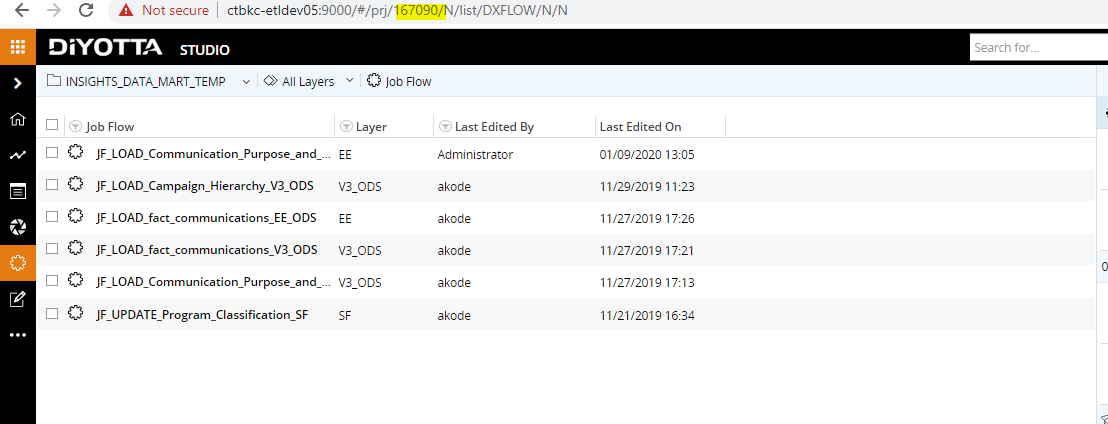
**Project\_ID :** This is unique for each project and we can get the project id as shown in below note.

**Branch name**  : Name of the branch we want to export the jobflows.

**A screenshot of a cell phone

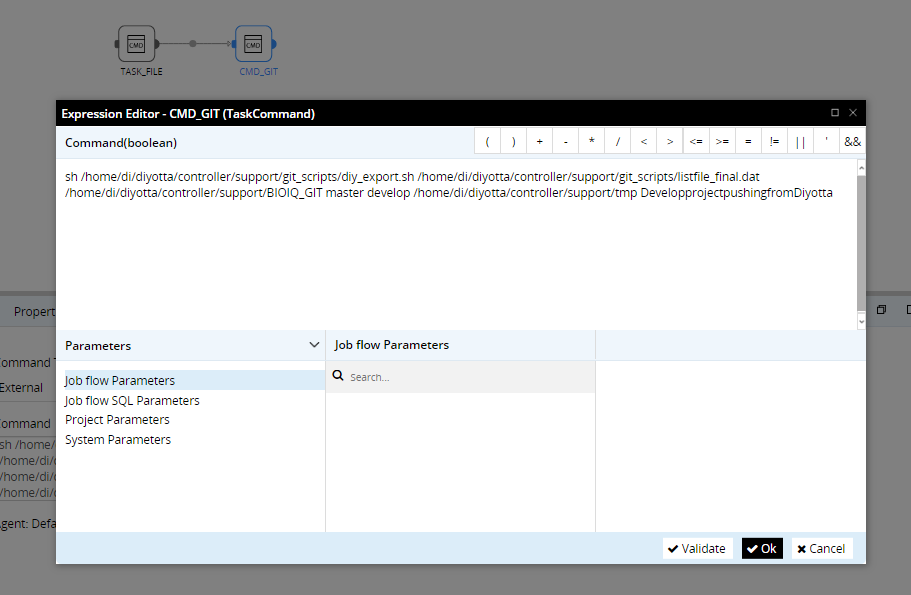
Description automatically generated**

**Note: We can get the project\_ID from the URL as shown in below screenshot which is highlighted.**

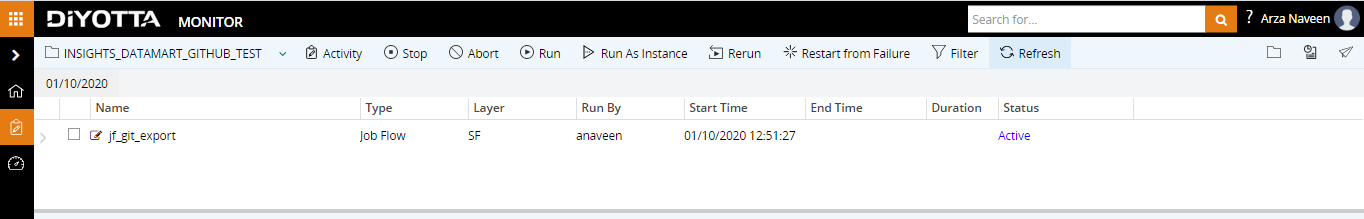


**C) CMD\_GIT:** The Export script requires 1 input.

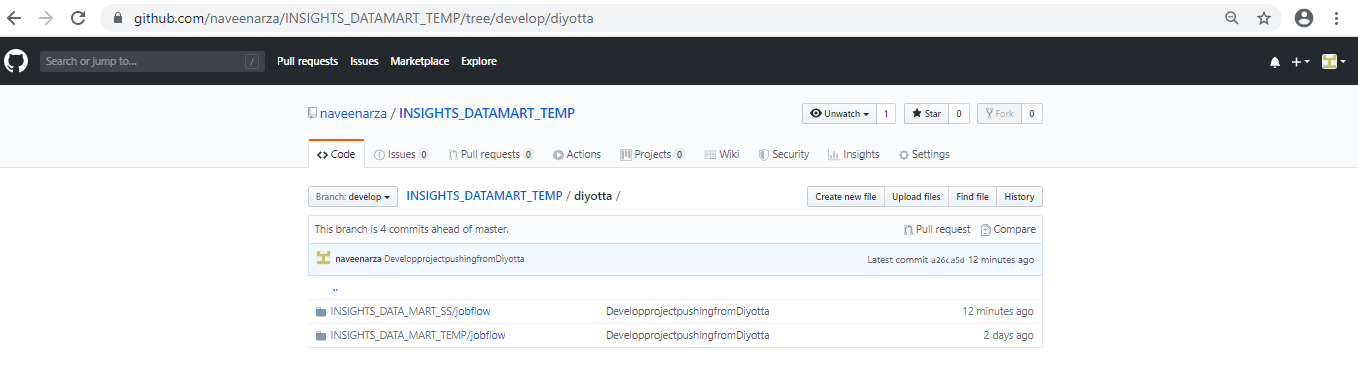
1. **Release Branch:** Name of the Branch You want to export the jobflows.



1. Run the jobflow Jf\_git\_export.



**e)** Validate Git hub remote repository once after the jobflow succeeded.



**2) Working of Jf\_git\_export\_jobflow:**

This jobflow is used to export some specific jobflows list which are mentioned in **exportlistfile.dat** file to remote git repository.

we need to update/modify the below input’s before running this jobflow.

1. Place the list of jobflows in **exportlistfile.dat** which we want to export .
2. Modify the remote git branch name in the jobflow parameters.

A screenshot of a social media post

Description automatically generated

# Import script Configuration

The Import script will get json file from remote repository and import the job flows to the specified project in Diyotta suite.

1. **Working of jf\_git\_import jobflow:**

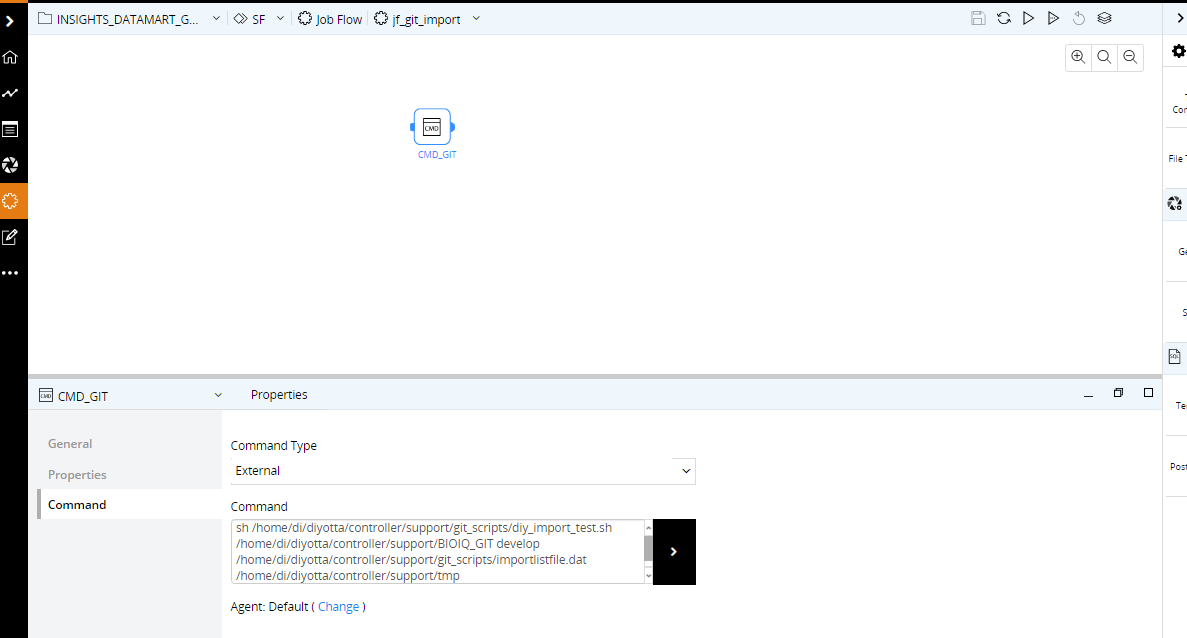
Follow the same steps for creating local git repository in the new environment which we want to import and import the jobflow **jf\_git\_import** in any project of the new environment.

1. **CMD\_GIT in:** Modify the below inputs as per the local setup.

The Import script requires 2 inputs:

**1. Branch\_name:** Name of the Branch You want to get the files from.

**2. Project Name:** Name of the project for importing the jobflows.



1. we need to modify the below parameters according to the requirement.

**Project name :** Name of the project which we want to import the jobflow**.**

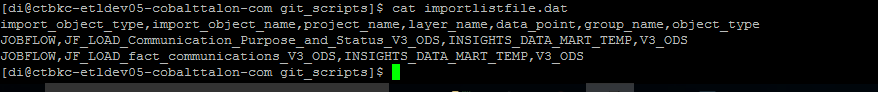
**Branch\_name:** Name of the branch we want to get the json file for importing.

A screenshot of a cell phone

Description automatically generated

**Note**: Below is the format for placing the Jobflows list in **importListfile.dat**

(JOBFLOW,Nameof the jobflow,Project\_name,Layer\_name)



4.Run the Jobflow Jf\_git\_import.

5.Validate whether the jobflows present in the listfile.dat are imported to the respective project folder.

