**GitHub- Training **

## - Agenda

* Introduction
* GitHub Administration (Org/Repo/Users/Team Creation)
* Personal Access Token Generation
* GIT Commands
* Pull Request Creation and Merging
* Create Tag / Releases

- Introduction

### Pre Requisite Software Download/ Registration :

* Register @ https://github.com/
* Install git bash @ https://git-scm.com/downloads

## Introduction

**What is git and why do we use it ?**

* Git is an open-source version control system that was started by Linus Trovalds – the same person who created Linux.
* Git is similar to other version control systems – like SVN (Subversion), CVS (Concurrent Versions System),

and TFS (Team Foundation Server), it is Distributed model.

* In the distributed approach, each developer works directly with his or her own local repository, and changes are shared between repositories as a separate step.

**Popular Open sources**

Git : written in a collection of Perl, C, and various shell scripts. Mercurial : Written in Python.

## Administration

* Org Creation
* Repository Creation
* Team Creation
* Adding Users to Team
* Provide the access to Team to Repository
* Importing Repository

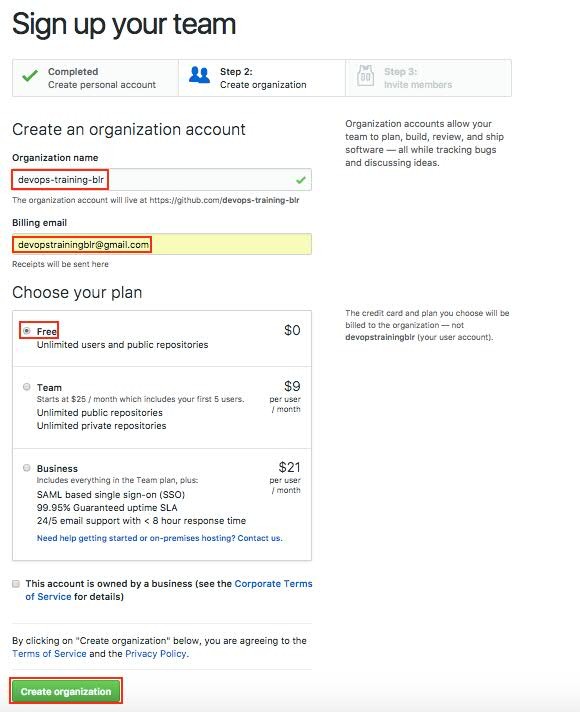
## Administration - Org

### Steps to create Organization

As soon as you login into GitHub, in the upper-right corner, click on + symbol, then click on **New Organization**,

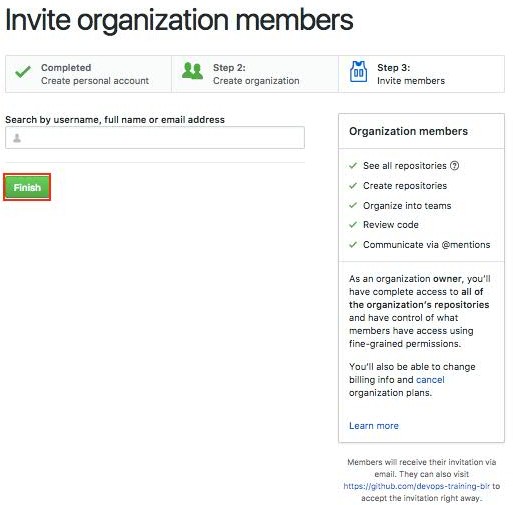
Under "**Organization name**", give the organization a name,

Under "**Billing email**", type the email where receipts for your organization's paid plan should be sent, then click on **Create organization** button.

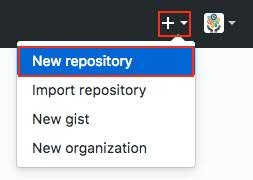


## Administration

* + Administration

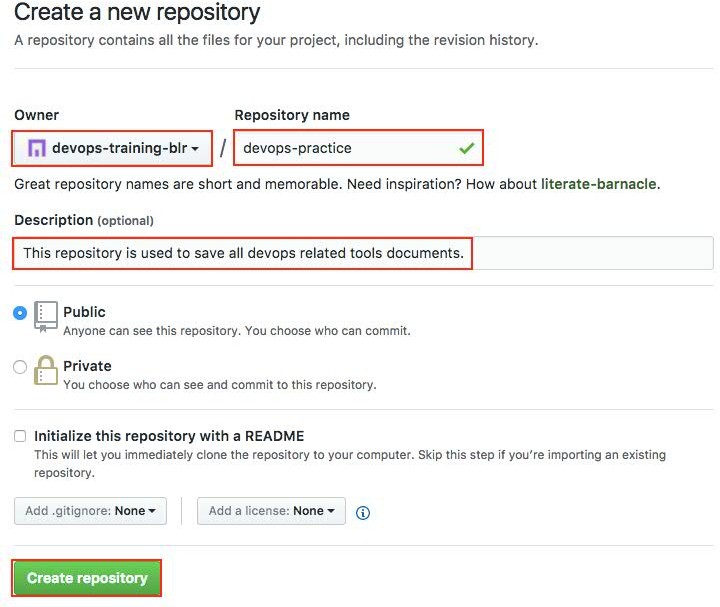


* + Administration - Repo

As soon as you login into GitHub, in the upper-right corner, click on + symbol, then click on **New repository.**

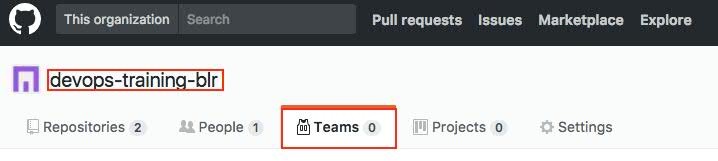
## Administration - Repo

Provide the details as below and click on Create repository.

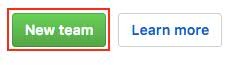


## Administration - Team

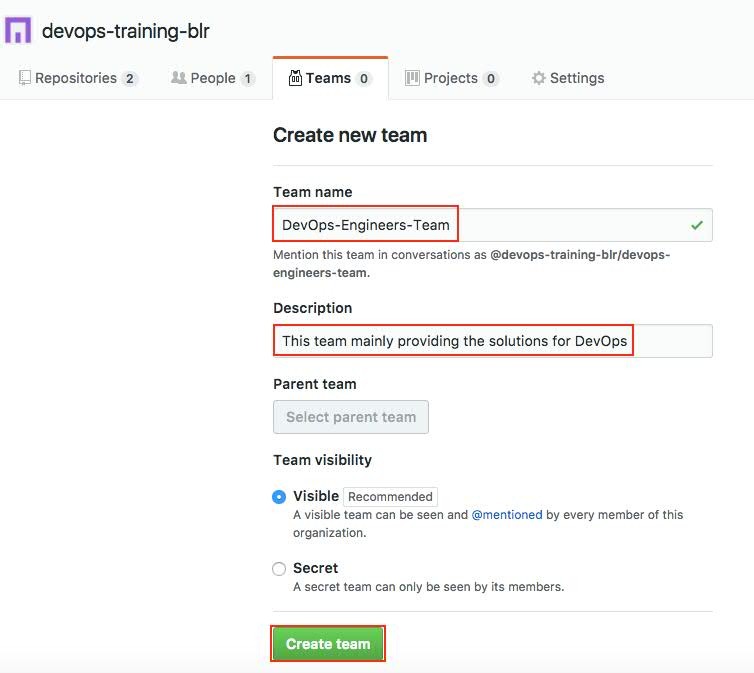
Select the organization where you need to create the team and click on Team.



Click on New team.



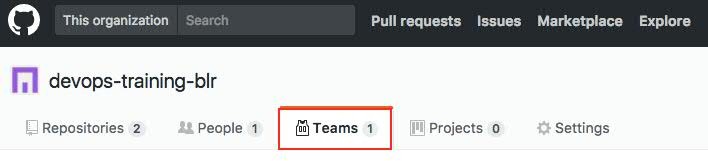
## Administration - Team

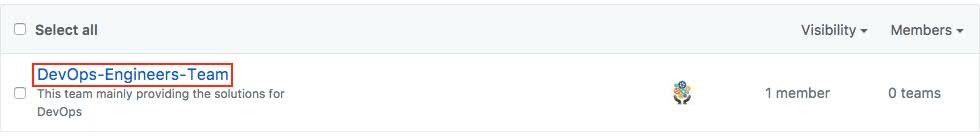


* + Administration – Add Users to Team

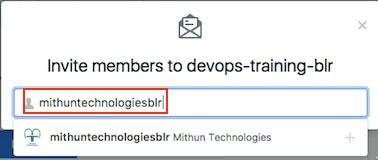
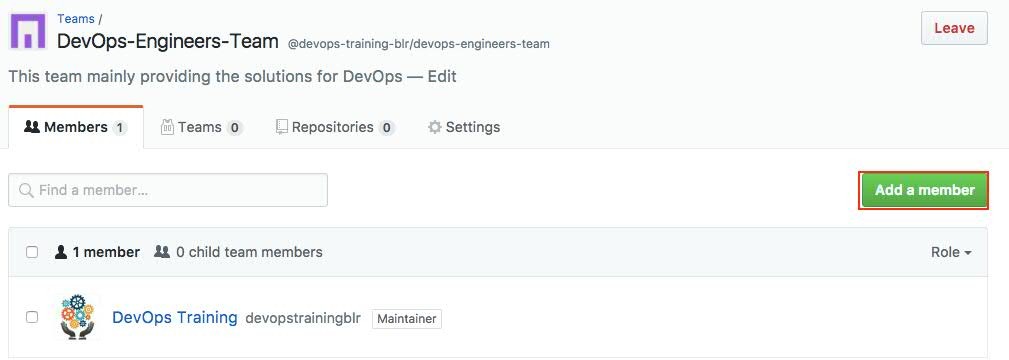
Go to the organization and Click on Teams and select the team name to which you want to add users.

https://github.com/devops-training-blr

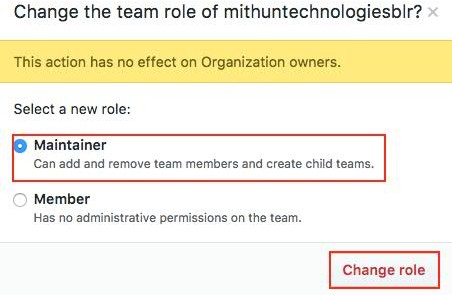
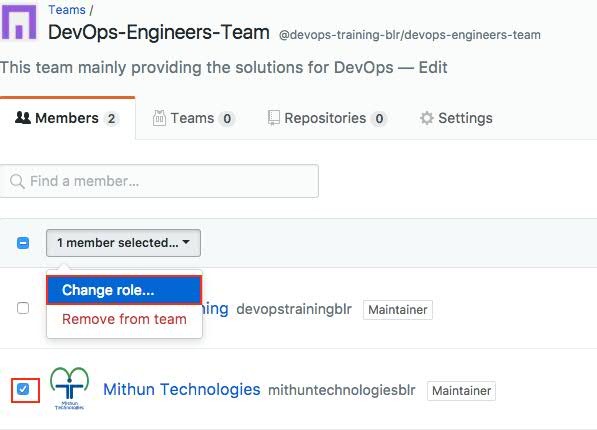




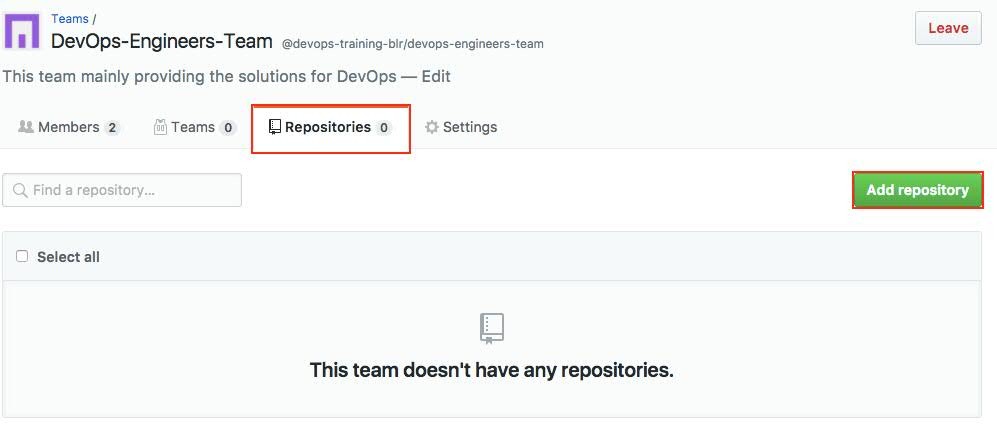
* + Administration – Add Users to Team



* + Administration – Give the admin access to user

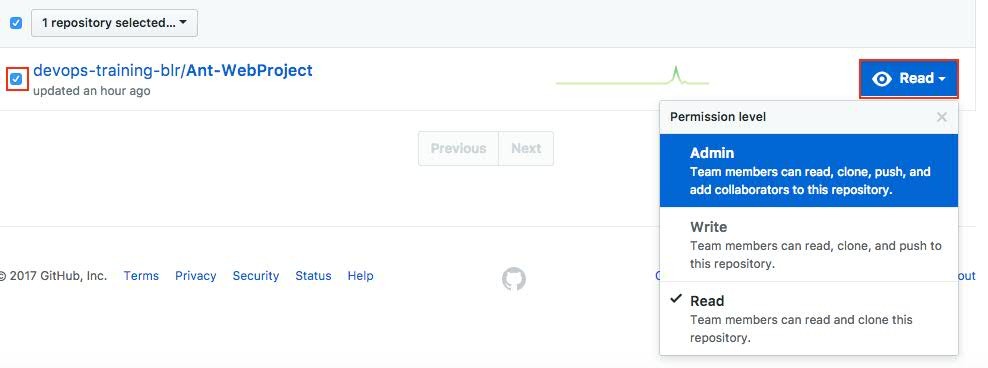


* + Administration – Give the repo access to Team



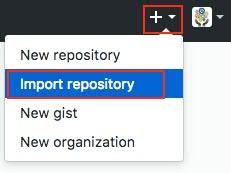
* + Administration – Give the repo access to Team



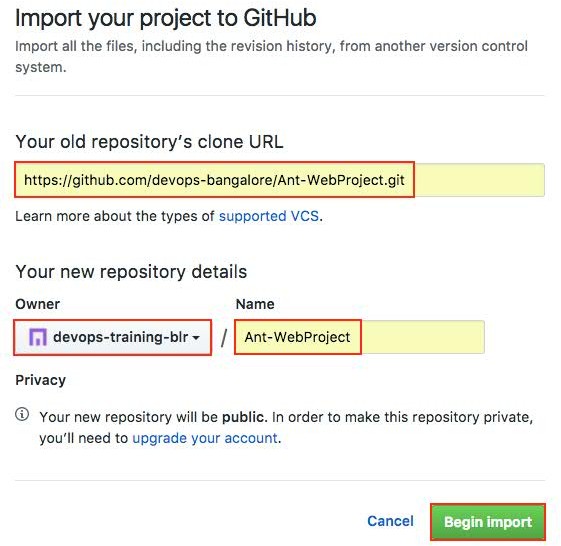


* + Administration - Repo

As soon as you login into GitHub, in the upper-right corner, click on + symbol, then click on **Import repository.**



## Administration - Repo



* + Introduction

**What is Version Control System?**

When developers are creating something (an application, for example), they are making constant changes to the code and releasing new versions, up to and after the first official (non-beta) release.

Version control systems keep these revisions straight, and store the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute.

* + Introduction

### Branch

* Branches are used to create another line of development.
* By default, Git has a master branch, which is same as trunk in Subversion (SVN). Usually, a branch is created to work on a new feature.
* Once the feature is completed, it is merged back with the master branch and we delete the branch.

### Tags

* Tags similar to branches, but the difference is that tags are immutable.
* It means, tag is a branch, which nobody intends to modify. Once a tag is created for a particular commit, even if you create a new commit, it will not be updated.
* Usually, developers create tags for product releases.

## Personal Access Token - Generation

### Go to Personal settings ---> Developer settings

**-** **Personal access tokens ---> Generate New Token -** **Provide some info about token in Token description** input box - Select the appropriate the Scopes and Click on **Generate token** button.

## SSH Key Generation

SSH keys are a way to identify trusted computers without involving passwords. You can generate an SSH key and add the public

key to your GitHub account.

#ls -al ~/.ssh ---> To see if existing SSH keys are present in machine.

**Generate SSH Key**

# ssh-keygen -t rsa -b 4096 -C ”JENKINS\_USERNAME@HOSTNAME”

Here-t --> Specifies the type of key to create. The possible values are "rsa1" for protocol version 1 and "rsa" or "dsa" for protocol version 2.

-b --> Specifies the number of bits in the key to create. For RSA (Rivest, Shamir, and Adelman) keys, the minimum size is 768 bits and the default is 2048 bits. Generally, 2048 bits is considered sufficient. DSA (Digital Signature Algorithm)keys must be exactly 1024 bits as specified by FIPS 186-2.

-C --> Provides a new comment. cat ~/.ssh/id\_rsa.pub

**Add SSH key to GitHub**

Click on Setting ---> **SSH and GPG keys** ---> **New SSH key** or **Add SSH key** ---> Provide the name for **Title** and copy SSH key

**Key** field ---> Click on Add SSH key button

## SSH Key Generation



* + Pull Request Creation and Merging

https://help.github.com/articles/creating-a-pull-request/

**Pull requests** let you tell others about changes you've pushed to a GitHub repository. Once a **pull request** is sent, interested parties can review the set of changes, discuss potential modifications, and even push follow-up commits if necessary.

* + Create Tag / Releases

Step 1: On GitHub, navigate to the main page of the repository. Step 2: Under your repository name, click **Releases**.

Step 3: Click **Create a new release/Draft a new release**.

Step 4: Type a version number for your release. Select a branch that contains the project you want to release. Usually, you'll want to release against your master branch, unless you're releasing beta software.

Type a title and description that describes your release.

Step 5: If you're ready to publicize your release, click **Publish release**. Otherwise, click **Save draft** to work on it later.

## GIT Commands

**#git inint :** Initialize git repo.

**#git status :** Check current status of the project

**#git add ”Mithun.txt"** : Add file/files to staging area also called indexing.

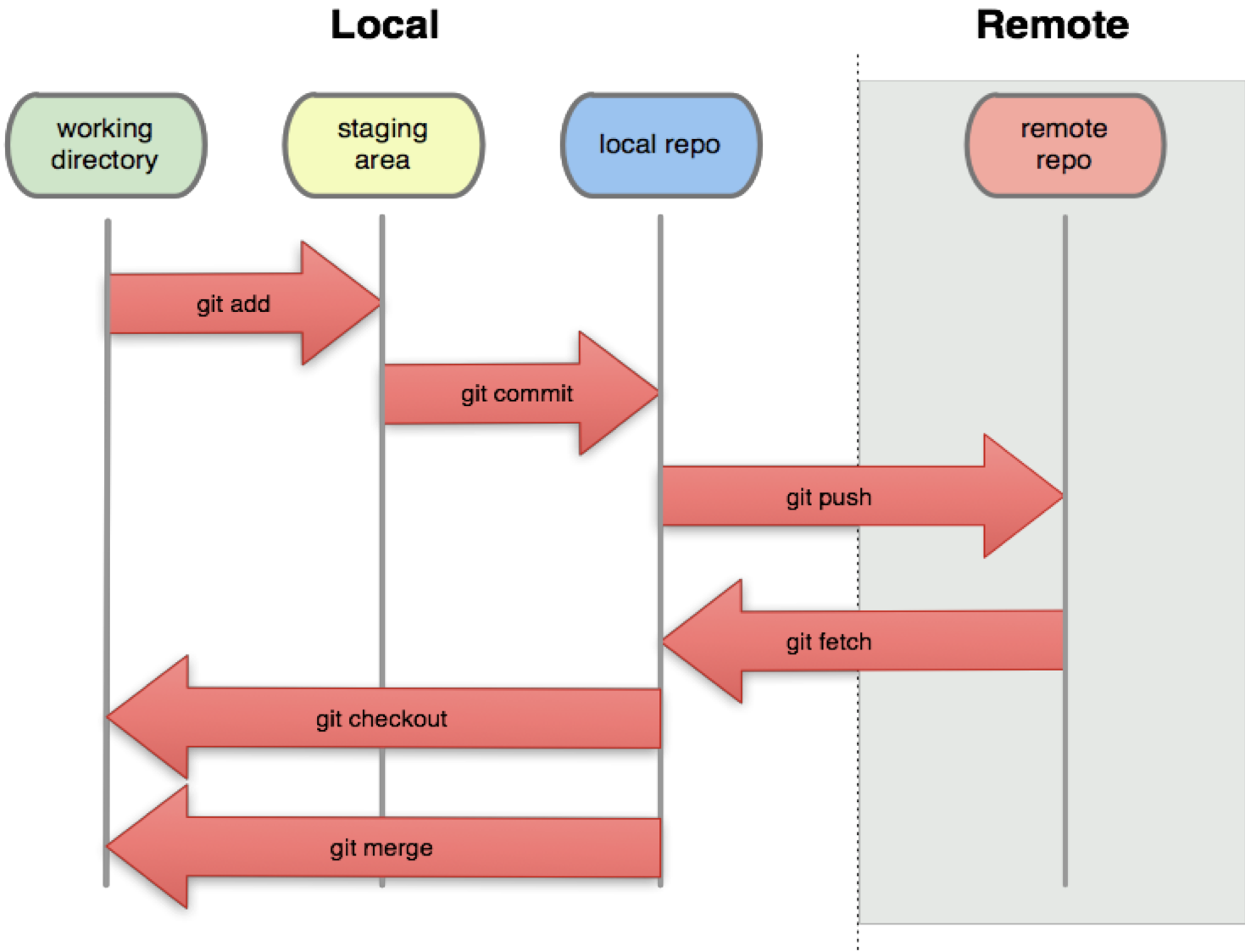
**#git commit -m "Initial commit" :** Commit your indexed files to local git repo.

**#git remote add origin <GIT Remote URL> :** Add Remote repository where you will be pushing your code.

**#git push –u origin master:** Push committed code to remote repository in master branch.

**#git clone <GIT Remote URL> :** Cloning remote git repository.

## Introduction



* + More GIT Commands
* $ git branch <branchName>
* $ git checkout <branchName>
* $ git merge <branchName>
* $ git rebase <branchName>
* $ git log
* $ git branch -d|-D <branchName>
* $ git fetch
* $ git pull
* $ git tag -a v1.0 -m “version 1.0”
* $git push --tags

## Reference URLs

https://help.github.com/articles/creating-releases/ ---> Tag/Releases https://help.github.com/articles/creating-a-pull-request-from-a-fork/ ---> Pull Request https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/ ---> SSH Key https://help.github.com/articles/creating-a-pull-request/ ---> Pull Request

Thanks