



# Intro to Git/GitHub/GitLab

NoVaLUG - July 2019

John Kennedy



# Topics

- What is git?
- Why do I need git?
- Git Glossary
- Create an empty git repo
  - What happens when I create a git repo?
- Create a git repo from existing data
- Manage your git repo
- What if I don't want EVERYTHING sync'd in my git repo?
- Sharing your git repo



## Topics (cont - GitHub)

- How does GitHub relate to git?
- How do I create a GitHub repo?
- Share my GitHub repo
- Other GitHub features



## Topics (cont - GitLab

- How does GitLab relate to git?
- How do I create a GitLab repo?
- Share my GitLab repo
- Other GitLab features
- I can run my own GitLab instance???



## Topics (cont)

- Getting Apache to NOT serve your .git directory
- Some things NOT covered you can do with git
  - Branches
  - Hooks
  - Sharing your git repo with remote users (hint - requires ssh to your machine)
- Questions



# What is git?

- git is a distributed version-control system
  - Tracks changes in files
  - Often used in software development to track changes in code
  - Created in 2005 by Linus Torvalds to track development of the Linux Kernel replacing BitKeeper
  - Every git (project) directory is a full copy of the repo
- Why is it called git?
  - "I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'." - Linus Torvalds
- According to the man page - “git - the stupid content tracker”
- More info on git and how it differs from previous version-control systems
  - <https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>



# Why do I need git?

git tracks changes to your files. This way if a mistake is made in your code, you can easily roll back your changes to the previous working version.

- Practical uses for git
  - Tracking changes in source code/scripts
  - Track changes in configuration files
  - Track changes in configuration management files



# git glossary

- **Branch** - A line of development - Allows you to test code in a “development” environment
- **Commit** - A single point in git history. Roughly same as revision or version in other version controls
- **Master** - Default branch of your repo
- **Merge** - To bring the contents of another branch into the current branch
- **Origin** - Default upstream repository





## git glossary (cont)

- **Pull** - To fetch a branch and merge it
- **Push** - To add your changes to a remote repository
- **Rebase** - To reapply changes from a branch to a base
- **Repo/Repository** - A collection of objects with a database showing which are reachable
  - **Remote Repository** - A repository that resides somewhere other than local
- **Stash** - To temporarily store changes to a repo for future use (after a rebase, for example)

- <https://git-scm.com/docs/gitglossary>



## Side note

git has a config file that keeps your name and email. Until we set this up, you will get nagged when you do commits. We will do this globally now.

Set your username:

```
git config --global user.name "FIRST_NAME LAST_NAME"
```

Set your email address:

```
git config --global user.email "MY_NAME@example.com"
```



## Side note (cont)

You can also set this per repository, change directory to the repo (after created - coming up):

Set your username:

```
git config user.name "FIRST_NAME LAST_NAME"
```

Set your email address:

```
git config user.email "MY\_NAME@example.com"
```

Same commands as before without "--global"



# Create an empty git repo

To create an empty git repo type the command *git init*.

That's it. Your directory is now a git repo.



# What did git do here?

- Created a .git directory with 3 files and 5 directories
  - Files - config, description, HEAD
  - Directories - branches, hooks, info, objects, refs
- Files
  - config - Covered later
  - description - Has the name of the repository
  - HEAD - Points to the reference of the current branch - See glossary web page for more info



# What did git do here? (cont)

- Directories
  - branches - Empty until you create branches
  - **hooks** - Contains sample hook scripts/files - A great resource to learn about hooks
  - info - Contains exclude file - Not covered today
  - objects - Contains info and pack directories - Not covered
  - refs - Contains heads and tags directories - Not covered



# Create a git repo from existing data

Go to the directory you want to work with type `git init.` - Note the dot (".") at the end is required

Type `git add.` - Note the dot (".") at the end is not required but preferred

Type `git commit -am "Initial commit"`

We will talk about these commands in a bit

Same files plus a few others as git is now tracking data. The `.git/index` file contains the git database, `COMMIT_EDITMSG` contains the last commit comment ("Initial commit" in our case). None of your existing files are altered



# Manage your git repo

git add . or git add <filename> -- This will add files to your repo. The . will add all files recursively

git rm <file> or git rm -f <directory> -- This will remove file(s) from the repo

git commit -am "Comment" -- Commits changes made to your local repo

git push -- Sends changes in your local repo to a remote repo

git pull -- Pulls down changes from remote repo - Good to run before working so you don't have to stash





## Manage your git repo - git stash (cont)

git stash -- Temporarily store changes outside the current repo to be applied later

```
git stash save "Comment on work or need to save"
```

```
git stash pop (stash@{#}) -- (optional) applies changes to the current repo/branch
```

```
git stash list -- list existing stashes
```

```
git stash apply -- Like git stash pop but the stash remains in the list of stashes
```



## Manage your git repo - logs/status (cont)

git log - Shows version history for current branch

git diff <first-branch> <second-branch> - Shows differences between branches

git status - Shows status of your repo (if there are any un-committed changes or if remote repo differs)

git blame - Shows what revision and author last modified each line of a file



# What if I don't want **EVERYTHING** sync'd in repo

`/path/to/repo/.gitignore` file tells git to leave any files listed out of the repo and not tracked

Examples:

- Temp data used by apps in the repo that you don't want to share
- Files with sensitive info such as user names/passwords (an altogether BAD IDEA to keep in files)
- Files not crucial to the data in your repo



# Sharing your git repo

Sharing used to be a bit of a pain. You could use the git daemon but that would make your repo public with no user control - although your repo would be read only. You could also use an app like gitosis to set up a shared repo. This was what most people did (and likely still do if they don't want to involve a third party).

Now GitHub and GitLab are mostly used, along with some other services like BitBucket (Atlassian). We will cover GitHub next followed by GitLab.



# How does GitHub relate to git?

GitHub was released in February 2008 and created using Ruby on Rails

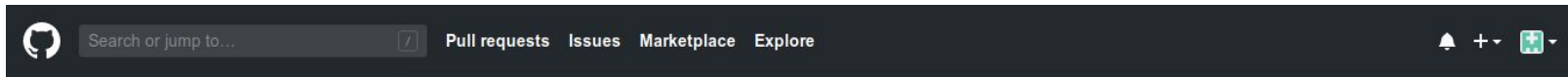
GitHub builds on the functionality by hosting git repositories and adding features such as bug tracking, feature requests, and wikis, among other services

GitHub has free, professional, and team (enterprise) accounts - see <https://github.com/pricing#feature-comparison>

Repositories can be public (public has read only access to the repo) or private (users/collaborators given specific access). In January 2019 free accounts were allowed unlimited private repos from the original 1 - likely in response to the upsurge of GitLab which allows unlimited private repos.



# How do I create a GitHub repo?



## Repositories

Your most active repositories will appear here. [Create a repository](#) or [explore repositories](#).

## Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#)[Start a project](#)

## Discover repositories

### timvideos/litex-buildenv

An environment for building LiteX based FPGA designs. Makes it easy to get everything you need!

Python ★ 51

### square/okhttp

An HTTP client for Android, Kotlin, and Java.

Java ★ 33.4k

### opencv/cvat

Powerful and efficient Computer Vision Annotation Tool (CVAT)

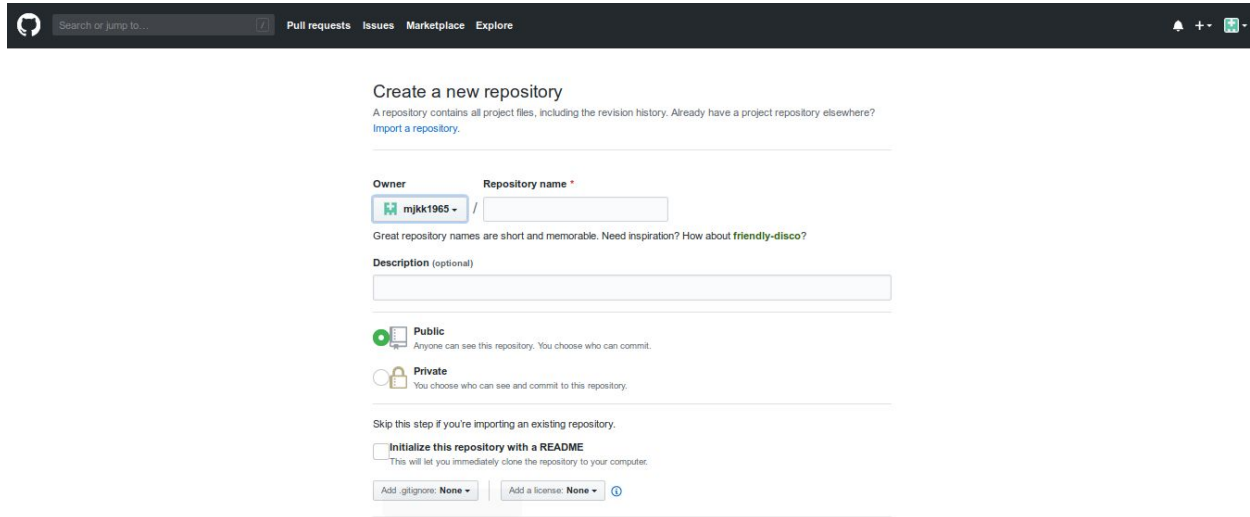
JavaScript ★ 2.1k

[Explore more](#) →

## Discover interesting projects and people to



# How do I create a GitHub repo? (cont)




Search or jump to... Pull requests Issues Marketplace Explore

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Owner **Repository name \***

 mjkk1965 - /

Great repository names are short and memorable. Need inspiration? How about [friendly-disco?](#)

Description (optional)

**Public**  
Anyone can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

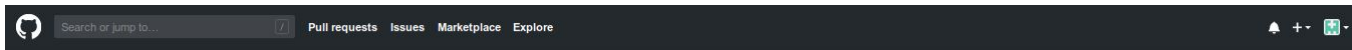
Skip this step if you're importing an existing repository.

**Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** - | Add a license: **None** - ⓘ



# How do I create a GitHub repo? (cont)



## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Owner **Repository name \***

 mjkk1965 /

Great repository names are short and memorable. Need inspiration? How about [friendly-disco](#)?

**Description** (optional)

**Public**  
Anyone can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

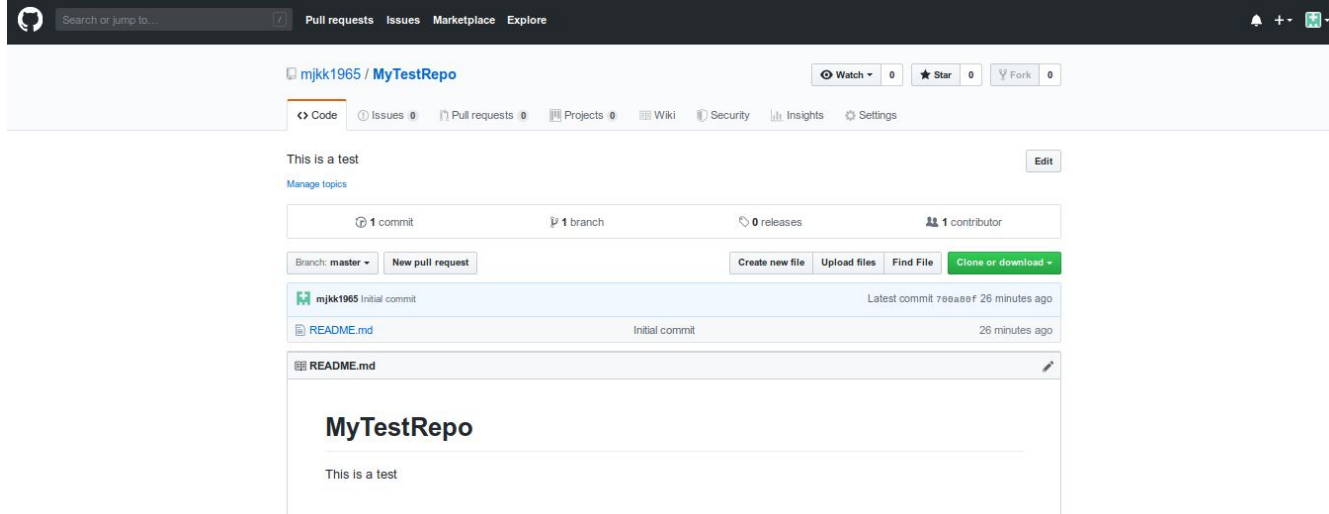
**Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** | Add a license: **None** ⓘ





# How do I create a GitHub repo? (cont)



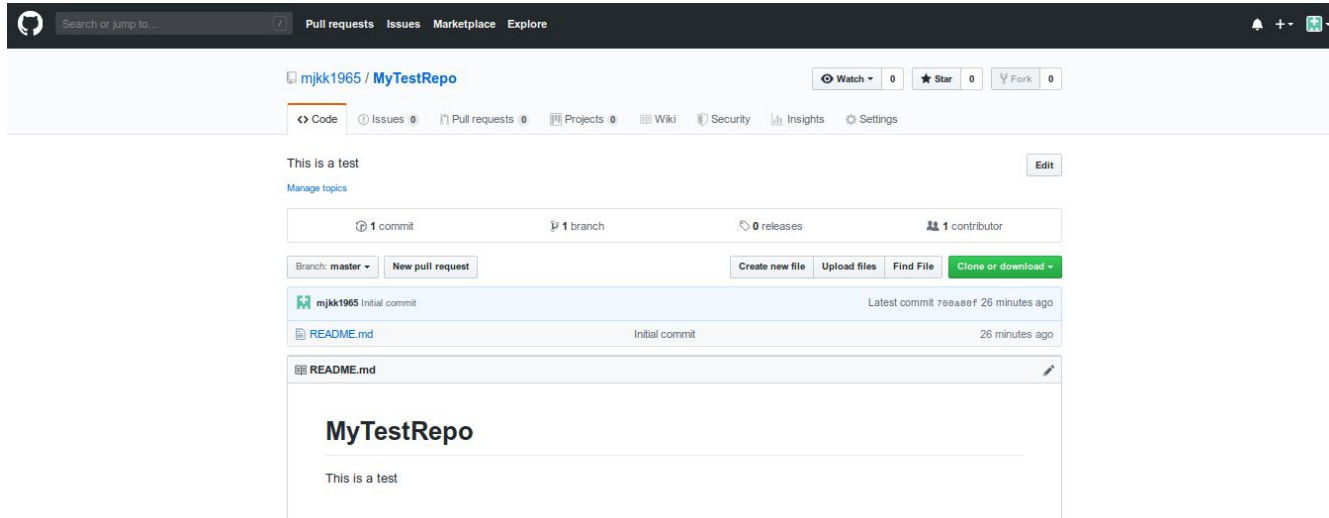
The screenshot shows a GitHub repository page for 'mjjk1965 / MyTestRepo'. The repository is currently on the 'master' branch and has 1 commit, 1 contributor, and 0 releases. The repository description is 'This is a test'. The repository contains a single file named 'README.md', which is the initial commit, created 26 minutes ago. The content of the README.md file is as follows:

```
MyTestRepo

This is a test
```



# Share my GitHub repo



The screenshot shows a GitHub repository page for user 'mjjk1965' and repository 'MyTestRepo'. The page includes a navigation bar with 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the repository name, there are buttons for 'Watch', 'Star', and 'Fork'. The main content area shows a commit history table with one commit by 'mjjk1965' titled 'Initial commit' from 26 minutes ago. Below the commit, the 'README.md' file is displayed, containing the text 'MyTestRepo' and 'This is a test'.

Search or jump to... Pull requests Issues Marketplace Explore

mjjk1965 / MyTestRepo Watch 0 Star 0 Fork 0

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

This is a test Edit

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find File Clone or download

Author	Commit message	Latest commit
mjjk1965	Initial commit	Initial commit 26 minutes ago

README.md Initial commit 26 minutes ago

## MyTestRepo

This is a test



# Share my GitHub repo (cont)

## Clone with HTTPS

[Use SSH](#)

Use Git or checkout with SVN using the web URL.

`https://github.com/mjkk1965/MyTestR`



[Download ZIP](#)

## Clone with SSH

[Use HTTPS](#)

You don't have any public SSH keys in your GitHub account. You can [add a new public key](#), or try cloning this repository via [HTTPS](#).

Use an SSH key and passphrase from account.

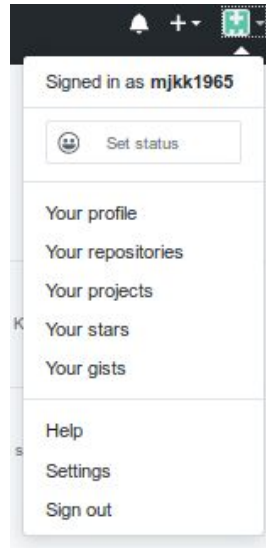
`git@github.com:mjkk1965/MyTestRepo.`



[Download ZIP](#)

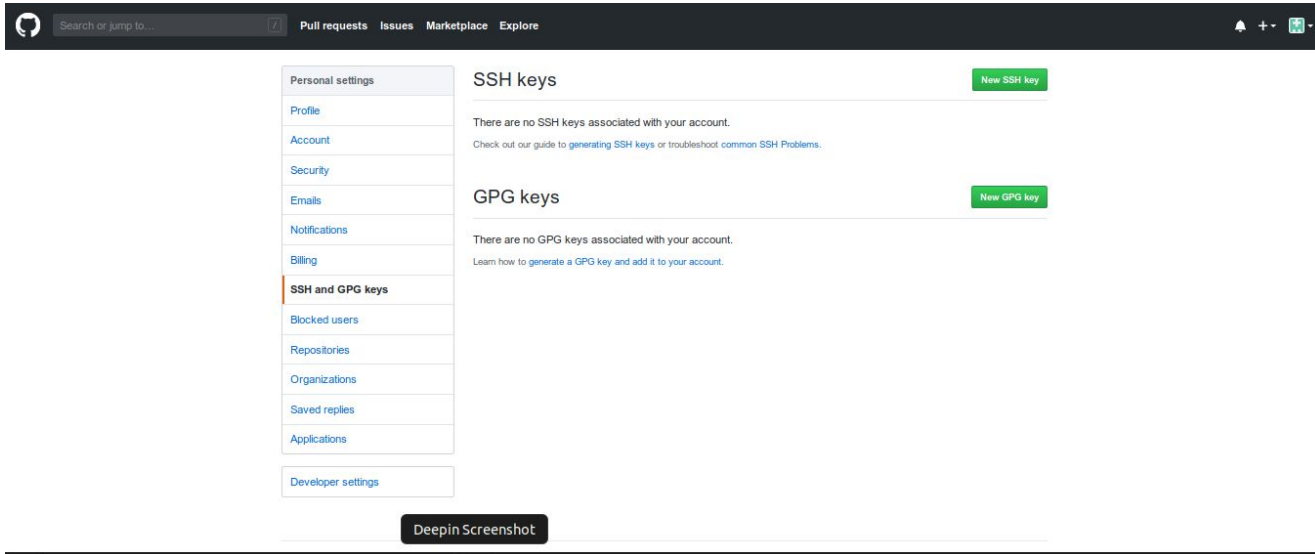


# Share my GitHub repo (cont)





# Share my GitHub repo (cont)



The screenshot shows the GitHub account settings page. The top navigation bar includes a search field, navigation links for Pull requests, Issues, Marketplace, and Explore, and notification icons. The left sidebar lists various settings categories, with 'SSH and GPG keys' highlighted. The main content area is divided into two sections: 'SSH keys' and 'GPG keys'. Both sections indicate that no keys are currently associated with the account and provide links to guides for generating and adding keys. A 'Deepin Screenshot' watermark is visible at the bottom of the screenshot.

Search or jump to... Pull requests Issues Marketplace Explore

Personal settings

- Profile
- Account
- Security
- Emails
- Notifications
- Billing
- SSH and GPG keys**
- Blocked users
- Repositories
- Organizations
- Saved replies
- Applications

Developer settings

## SSH keys New SSH key

There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH Problems](#).

## GPG keys New GPG key

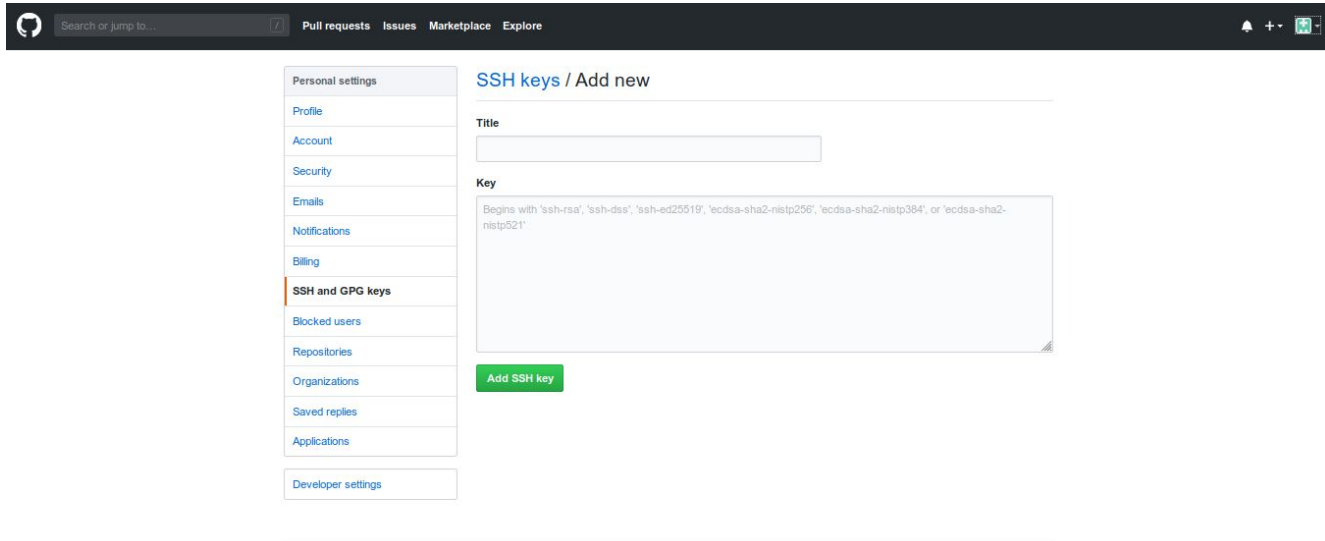
There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and add it to your account.

Deepin Screenshot



# Share my GitHub repo (cont)



The screenshot shows the GitHub user interface for managing SSH keys. At the top, there is a dark navigation bar with the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. On the left, a sidebar lists various settings categories, with 'SSH and GPG keys' highlighted. The main content area is titled 'SSH keys / Add new' and contains a form with a 'Title' input field and a 'Key' text area. The key area includes a placeholder text indicating the expected format. A green 'Add SSH key' button is positioned below the key input field.

Personal settings

- Profile
- Account
- Security
- Emails
- Notifications
- Billing
- SSH and GPG keys**
- Blocked users
- Repositories
- Organizations
- Saved replies
- Applications

Developer settings

SSH keys / Add new

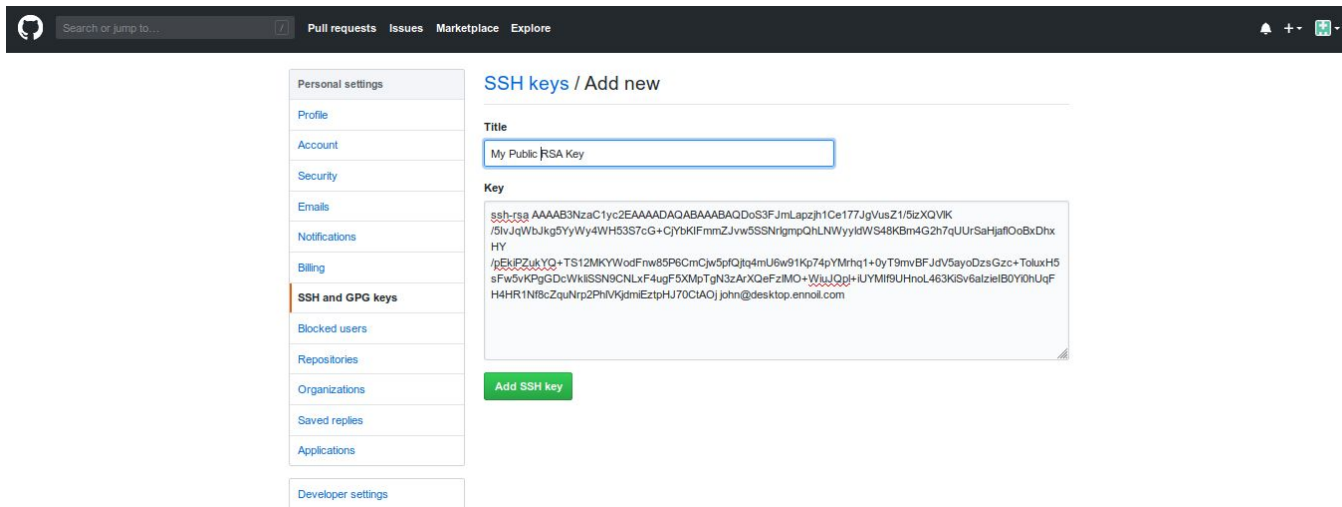
Title

Key

Begin with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

# Share my GitHub repo (cont)



The screenshot shows the GitHub 'SSH keys / Add new' page. On the left is a sidebar with navigation links: Personal settings, Profile, Account, Security, Emails, Notifications, Billing, SSH and GPG keys (highlighted), Blocked users, Repositories, Organizations, Saved replies, Applications, and Developer settings. The main content area has a dark header with 'Search or jump to...', 'Pull requests', 'Issues', 'Marketplace', 'Explore', and notification icons. Below the header, the 'SSH keys / Add new' section contains a 'Title' input field with the text 'My Public RSA Key', a 'Key' input field containing a long SSH public key, and a green 'Add SSH key' button.

Personal settings

Profile

Account

Security

Emails

Notifications

Billing

**SSH and GPG keys**

Blocked users

Repositories

Organizations

Saved replies

Applications

Developer settings

SSH keys / Add new

Title

My Public RSA Key

Key

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDcS3FJmLapzjh1Ce177JgVusZ1/5izXOVik  
/8vJqWbJkg5YyWy4WH53S7cG+CjYbKIFmmZJvw5SSNrgmpQhLNWyykdWS48Bm4G2h7qUUrSaHjalOoBxDhx  
HY  
/rEhPZkYQ+TS12MKY1WodFmw85P6CmCjw5pQjg4mU6w91Kg74pYMrh1+OyT9mvBFJuv5ayoDzsGzc+ToluxH5  
sFw6vKPyGdcWkISSN9CNLxF4ugF5XmpTgN3zArXQeFzIMO+WuUQg+IUyMf9UHnoL463KSVsbatzeIB0Y0hUqf  
H4HR1NBcZquNrp2PhIVKjmiEztpHJ70CIAOjJohn@desktop.ennol.com
```

Add SSH key



# Share my GitHub repo (cont)

## Clone with HTTPS

[Use SSH](#)

Use Git or checkout with SVN using the web URL.

`https://github.com/mjkk1965/MyTestR`



[Download ZIP](#)

## Clone with SSH

[Use HTTPS](#)

You don't have any public SSH keys in your GitHub account. You can [add a new public key](#), or try cloning this repository via [HTTPS](#).

Use an SSH key and passphrase from account.

`git@github.com:mjkk1965/MyTestRepo.`



[Download ZIP](#)





## Share my GitHub repo (cont)

On your computer, in your ~/git folder run the command:

```
git clone <ssh or https string from last slide>
```

```
git clone git@github.com:mjkk1965/MyTestRepo.git
```

Your new repo will be found in ~/git/MyTestRepo



# Other GitHub features

README.md uses Markdown and allows you to give basic info about your project

Wiki for more detailed documentation

Allow Teams and other contributors (will all but free version)

Info on your project such as the number of commits, contributor contribution, repo traffic, Forks, and Pull requests



# How does GitLab relate to git?

GitLab was released in 2011 and created using Ruby

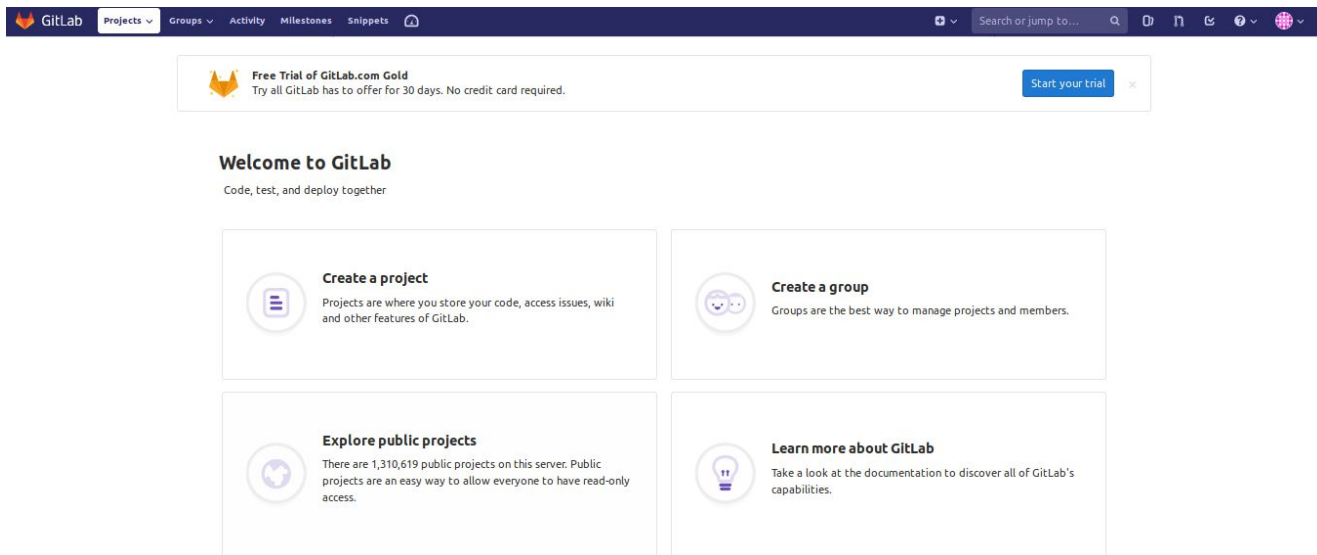
GitLab builds on the functionality by hosting git repositories and adding features such as bug tracking, feature requests, and wikis, among other services including integrated CI/CD

GitLab has 4 levels of accounts from free to "Gold" at \$99/user/year - see <https://about.gitlab.com/pricing/>

Repositories can be public (public has read only access to the repo) or private (users/collaborators given specific access).



# How do I create a GitLab repo



The screenshot shows the GitLab homepage with a dark blue navigation bar at the top. The navigation bar includes the GitLab logo, a search bar, and several menu items: Projects, Groups, Activity, Milestones, and Snippets. A notification banner for a 'Free Trial of GitLab.com Gold' is visible below the navigation bar. The main content area is titled 'Welcome to GitLab' and features four cards: 'Create a project', 'Create a group', 'Explore public projects', and 'Learn more about GitLab'.


**GitLab** Projects Groups Activity Milestones Snippets

Search or jump to...

**Free Trial of GitLab.com Gold**  
Try all GitLab has to offer for 30 days. No credit card required. [Start your trial](#)


## Welcome to GitLab

Code, test, and deploy together




### Create a project

Projects are where you store your code, access issues, wiki and other features of GitLab.




### Create a group

Groups are the best way to manage projects and members.



### Explore public projects

There are 1,310,619 public projects on this server. Public projects are an easy way to allow everyone to have read-only access.



### Learn more about GitLab

Take a look at the documentation to discover all of GitLab's capabilities.

# How do I create a GitLab repo? (cont)

**New project**

A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

To only use CI/CD features for an external repository, choose **CI/CD for external repo**.

Information about additional Pages templates and how to install them can be found in our [Pages getting started guide](#).

**Tip:** You can also create a project from the command line. [Show command](#)

**Blank project** | Create from template | Import project | CI/CD for external repo

**Project name**  
MyTestRepo

**Project URL**  
https://gitlab.com/ennoil/

**Project slug**  
mytestrepo

Want to house several dependent projects under the same namespace? [Create a group](#).

**Project description (optional)**  
Test

**Visibility Level**

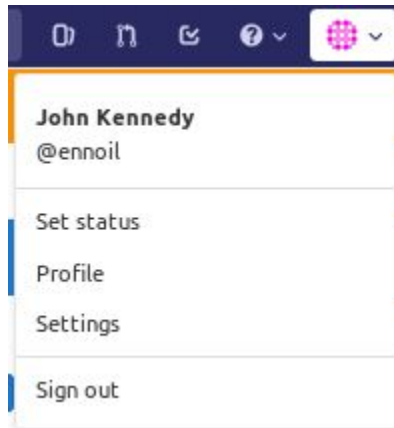
- Private  
Project access must be granted explicitly to each user.
- Internal  
The project can be accessed by any logged in user.
- Public  
The project can be accessed without any authentication.

**Initialize repository with a README**  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

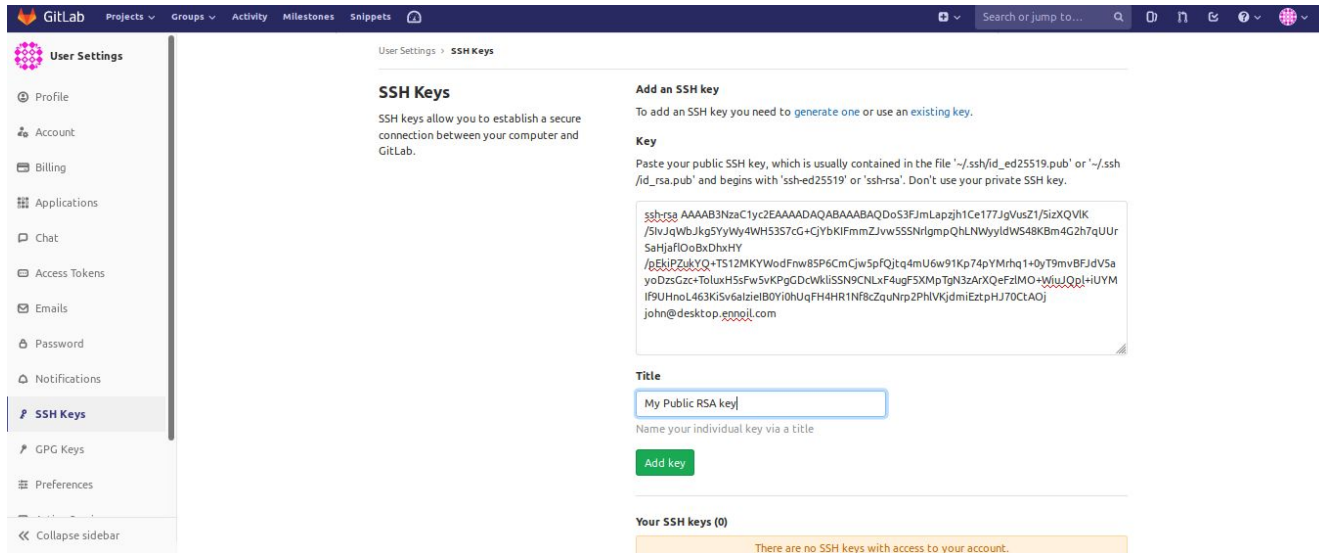
# How do I create a GitLab repo? (cont)

The screenshot displays the GitLab web interface for a newly created repository named "MyTestRepo". The interface includes a top navigation bar with "GitLab" and various menu items like "Projects", "Groups", "Activity", "Milestones", and "Snippets". A search bar is also present. Below the navigation bar, a blue banner indicates that the project "MyTestRepo" was successfully created. The main content area shows the repository details, including the name "MyTestRepo", the project ID "13297371", and a "clone" button. There are also statistics for stars (0), forks (0), and bytes (0). A section for "Auto DevOps" is visible, explaining that it will automatically build, test, and deploy applications based on predefined CI/CD configurations. Below this, there are buttons for "Add license", "1 Commit", "1 Branch", "0 Tags", and "0 Bytes Files". The repository is currently on the "master" branch, and the commit hash is "b178f158". A "Deepin Screenshot" watermark is visible at the bottom of the image.

# How do I create a GitLab repo? (cont)



# How do I create a GitLab repo? (cont)



The screenshot shows the GitLab user interface. On the left is a sidebar with 'User Settings' selected, containing options like Profile, Account, Billing, Applications, Chat, Access Tokens, Emails, Password, Notifications, SSH Keys (highlighted), GPG Keys, and Preferences. The main content area is titled 'SSH Keys' and includes a description: 'SSH keys allow you to establish a secure connection between your computer and GitLab.' Below this is a section 'Add an SSH key' with instructions to generate or use an existing key. A 'Key' field contains a long public key string. A 'Title' field has the text 'My Public RSA key'. At the bottom, it shows 'Your SSH keys (0)' and a message: 'There are no SSH keys with access to your account.'

**SSH Keys**

SSH keys allow you to establish a secure connection between your computer and GitLab.

**Add an SSH key**

To add an SSH key you need to [generate one](#) or use an [existing key](#).

**Key**

Paste your public SSH key, which is usually contained in the file `~/.ssh/id_ed25519.pub` or `~/.ssh/id_rsa.pub` and begins with `ssh-ed25519` or `ssh-rsa`. Don't use your private SSH key.

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDoS3FJmLapzh1Ce177JgVusZ1/SizXQVik  
/SivJqWbJkg5YyWy4WHS3S7cG+cJybKIFmmZjww5SSNrlgmpQhLNWyyldW548KBm4G2h7qUUr  
SaHjaFIo8xHxHY  
/gEkPZukYQ+TS12MKYWodFmw85P6CmCjw5pFQjtq4mU6w91Kp74pYMrhq1+0yT9mvBFJdV5a  
yoDzsGzc+ToluxH5sFw5vKpGdcWkllSSN9CNLx4ugF5XMpTgN3zArXQeFzIMO+WuUQpl+IUYM  
lF9UHnoL463KlSv6alzleB0Yi0HUqFH4HR1Nf8cZquNrp2PHLVKjdmieZtpHJ7OCLAOJ  
john@desktop.eggoil.com
```

**Title**

My Public RSA key

Name your individual key via a title

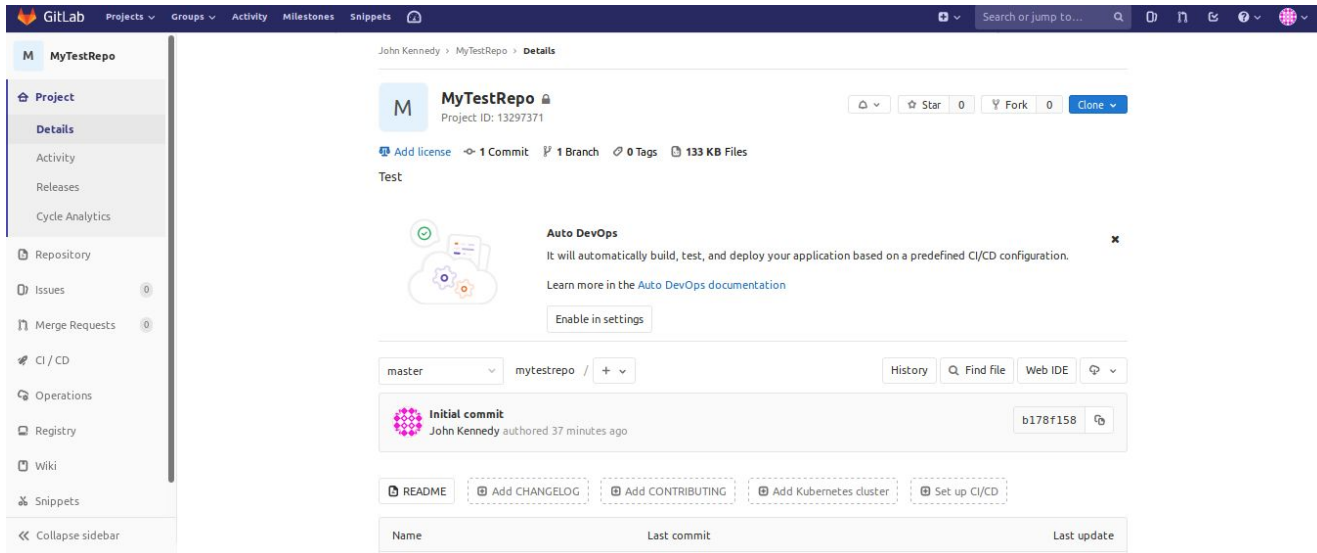
**Add key**

**Your SSH keys (0)**

There are no SSH keys with access to your account.

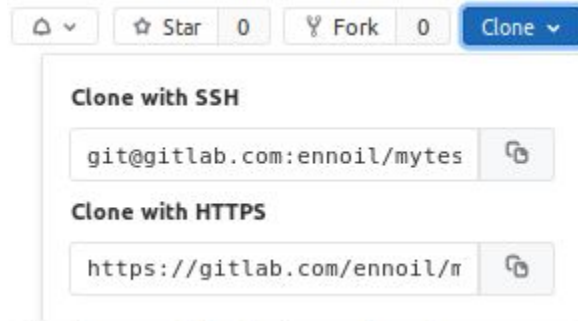


# How do I create a GitLab repo? (cont)



The screenshot displays the GitLab web interface for a newly created repository named "MyTestRepo". The interface is dark-themed and includes a sidebar on the left with navigation options: Project, Details, Activity, Releases, Cycle Analytics, Repository, Issues (0), Merge Requests (0), CI / CD, Operations, Registry, Wiki, Snippets, and Collapse sidebar. The main content area shows the repository details for "MyTestRepo" (Project ID: 13297371) by John Kennedy. It features a "Clone" button, "Add license", "1 Commit", "1 Branch", "0 Tags", and "133 KB Files". A prominent "Auto DevOps" notification is displayed, stating it will automatically build, test, and deploy applications based on predefined CI/CD configurations, with a link to documentation and an "Enable in settings" button. Below the notification, the repository's current state is shown as "master" with a "mytestrepo / +" path, and buttons for "History", "Find file", and "Web IDE". The "Initial commit" by John Kennedy is listed, authored 37 minutes ago, with commit hash "b178f158". At the bottom, there are buttons for "README", "Add CHANGELOG", "Add CONTRIBUTING", "Add Kubernetes cluster", and "Set up CI/CD". A table header with columns "Name", "Last commit", and "Last update" is visible at the very bottom.

## How do I create a GitLab repo? (cont)





## How do I create a GitLab repo? (cont)

On your computer, in your ~/git folder run the command:

```
git clone <ssh or https string from last slide>
```

```
git clone git@github.com:mjkk1965/MyTestRepo.git
```

Your new repo will be found in ~/git/MyTestRepo



# Other GitLab features

README.md uses Markdown and allows you to give basic info about your project

Wiki for more detailed documentation

Allow Teams and other contributors (will all but free version)

Info on your project such as the number of commits, contributor contribution, repo traffic, Forks, and Pull requests

Integrated CI/CD capability



# I can run my own GitLab instance

Why yes, yes you can run your own GitLab instance in your environment.

4 different levels from free with community support to Ultimate with 24/7 uptime support and many other features for \$99/user/month

GitLab recommends at least 4GB of FREE RAM to run - That would translate to a minimum of 8 GB once you take away OS memory. Not likely for individuals to run in cloud

There is an official GitLab Docker image for GitLab Community Edition (free)



# Getting Apache to NOT serve your .git directory

A fun little feature in Apache to NOT serve a directory in DocumentRoot

In your site config file add (in the VirtualHost stanza):

```
<Directorymatch "^.*/\..git/">
```

```
Order deny,allow
```

```
Deny from all
```

```
</Directorymatch>
```



# Some things NOT covered you can do with git

- Branches
  - Allow different environments of same code - Keeps development out of production until you are ready to merge them
  - Allows experimentation with code in a place where it will not affect your stable code
- Hooks
  - Allows you to run scripts based on git actions. See `/path/to/git/repo/.git/hooks` for examples



# What we covered

- What is git
- Why do I need it
- Some glossary terms
- Creating and managing a git repo
- Sharing your git repo
- GitHub and GitLab
  - Create Repo
  - Share Repo
  - Other features
  - Yes, you can run your own GitLab instance





## What we covered (cont)

- How to get Apache to not serve your .git directory
- Some of the things you can do with Git NOT covered today



# Additional Resources

Git official documentation - <https://git-scm.com/docs>

A nice visual reference to git - <http://ndpsoftware.com/git-cheatsheet.html>

A PDF cheat sheet from GitHub -

<https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>

A good tutorial on pull requests -

<https://www.freecodecamp.org/news/a-simple-git-guide-and-cheat-sheet-for-open-source-contributors>



**Questions???**