

GAME PROGRAMMING 2

GIT & GITHUB

GIT & GITHUB

GIT

- VERSION CONTROL SYSTEM
- FREE & OPEN SOURCE
- DISTRIBUTED
- BY FAR, THE MOST COMMON VERSION CONTROL SYSTEM USED BY SOFTWARE COMPANIES

GITHUB

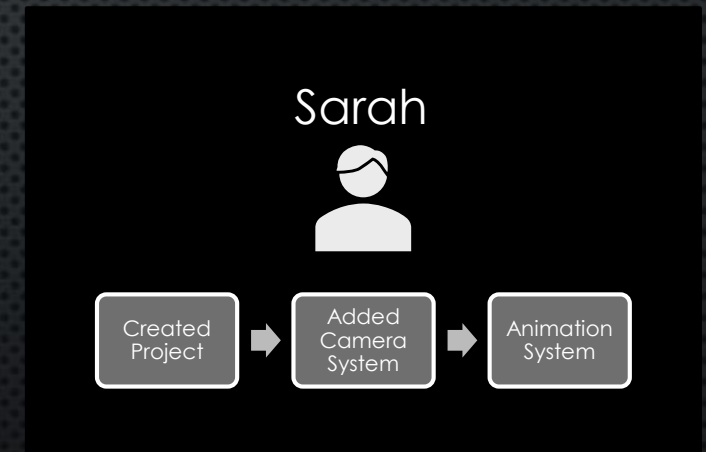
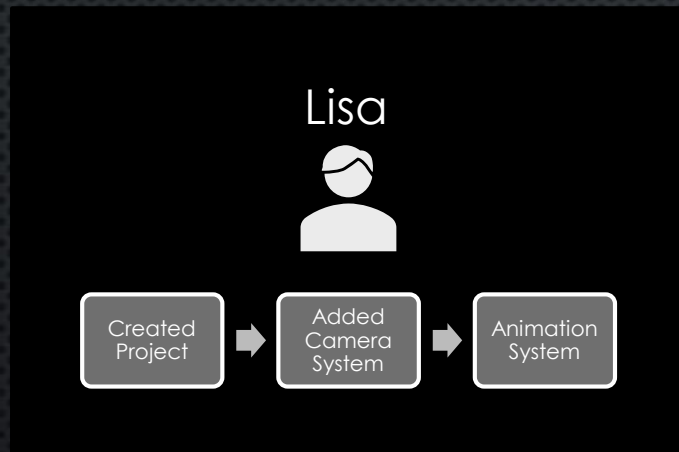
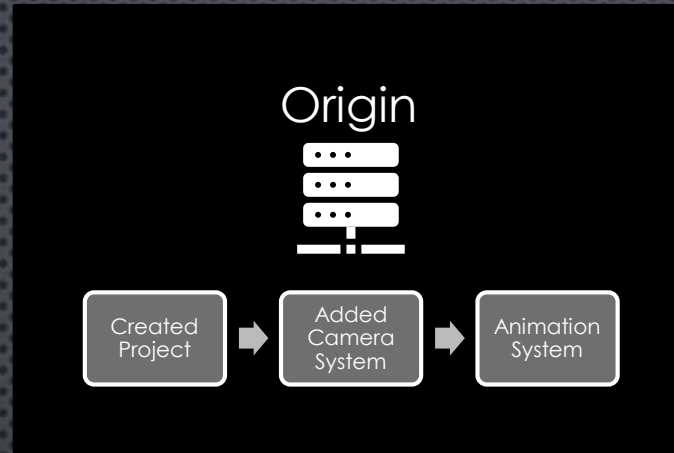
- PROVIDES HOSTING FOR GIT-REPOSITORIES
- OWNED BY MICROSOFT

GIT, COMMITS

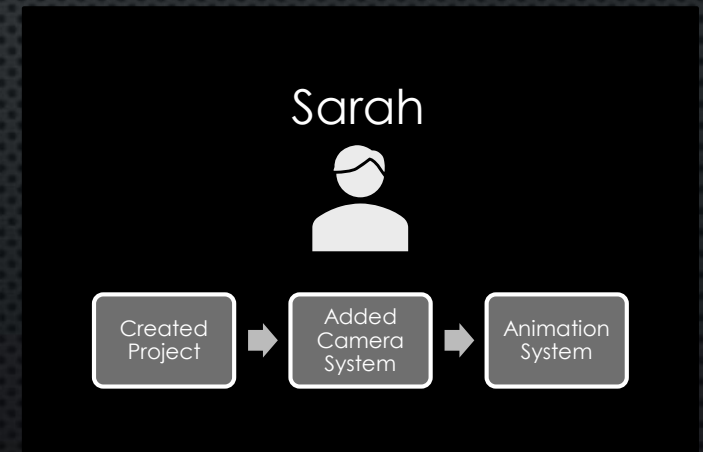
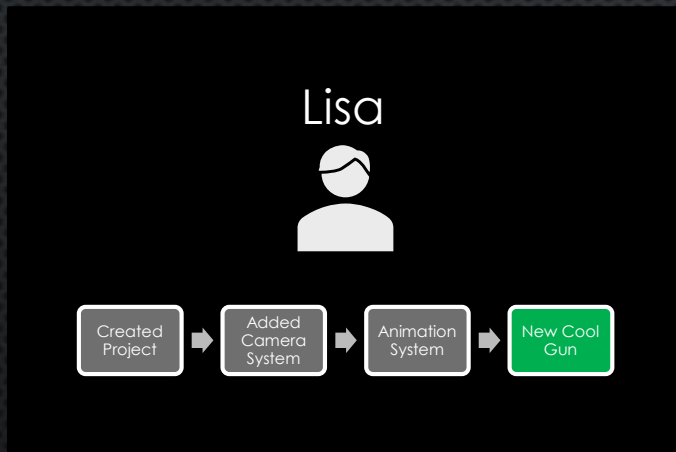
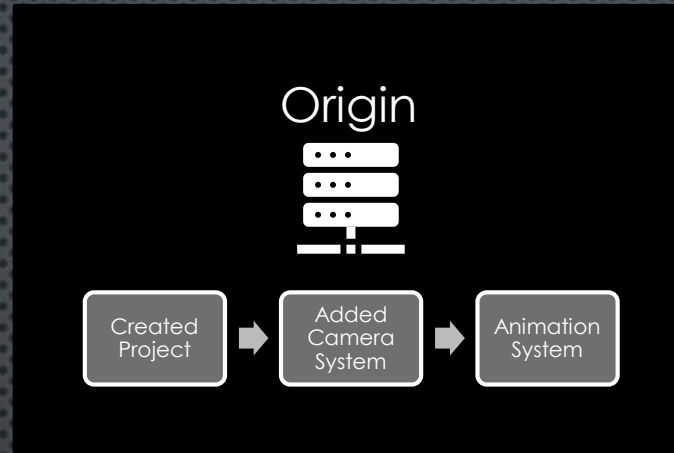


- GIT IS A VERSION CONTROL SYSTEM
- LIST OF COMMITS (SNAPSHOTS) OF THE PROJECT
- CAN GO BACK TO ANY COMMIT AT ANY TIME
- EVERYBODY HAS A COPY OF THE COMPLETE HISTORY

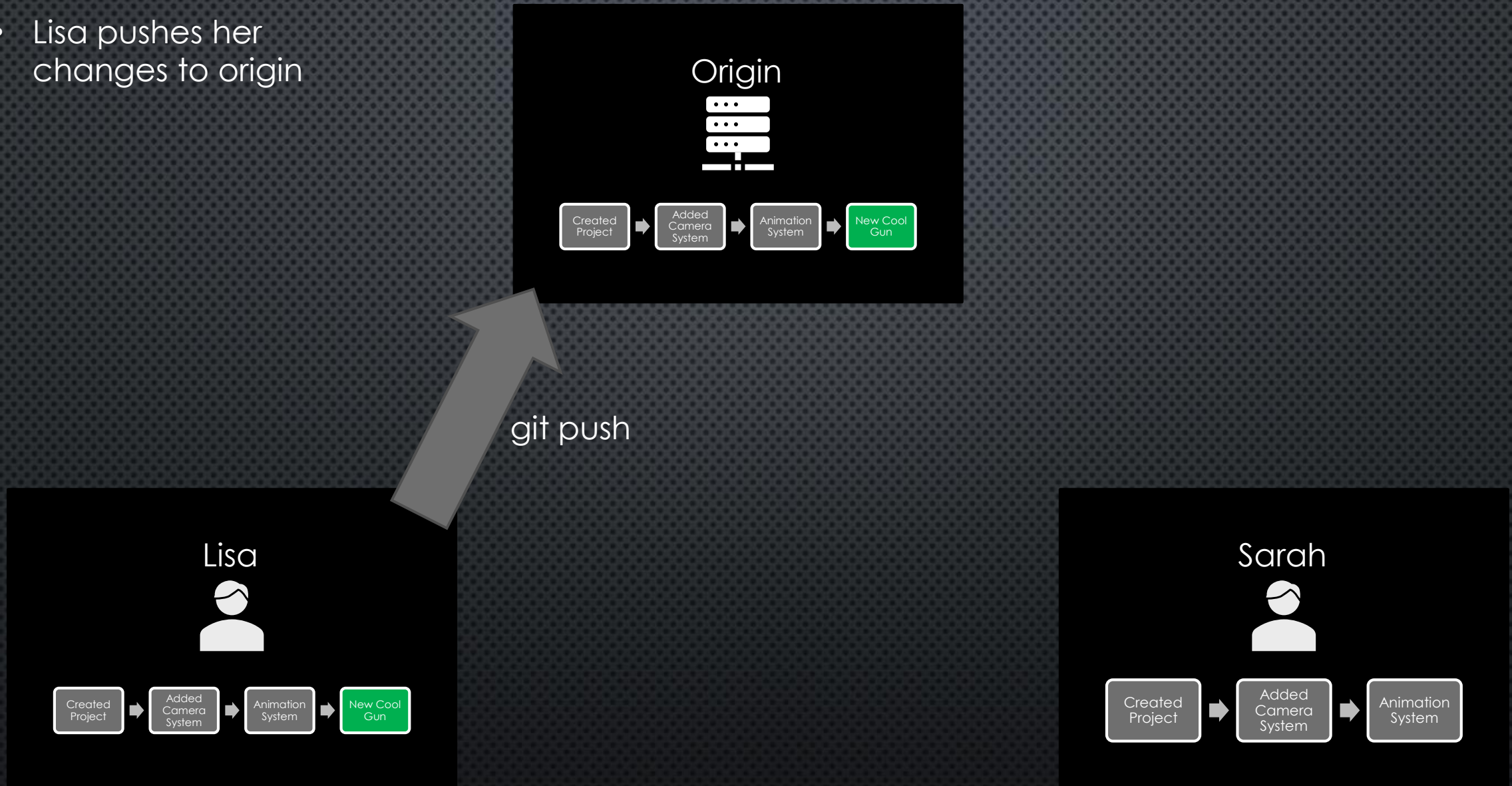
- Lisa & Sarah are developers on a new game
- They both have a copy of the complete version history
- The git repository is also hosted at Origin (Could be GitHub for example)



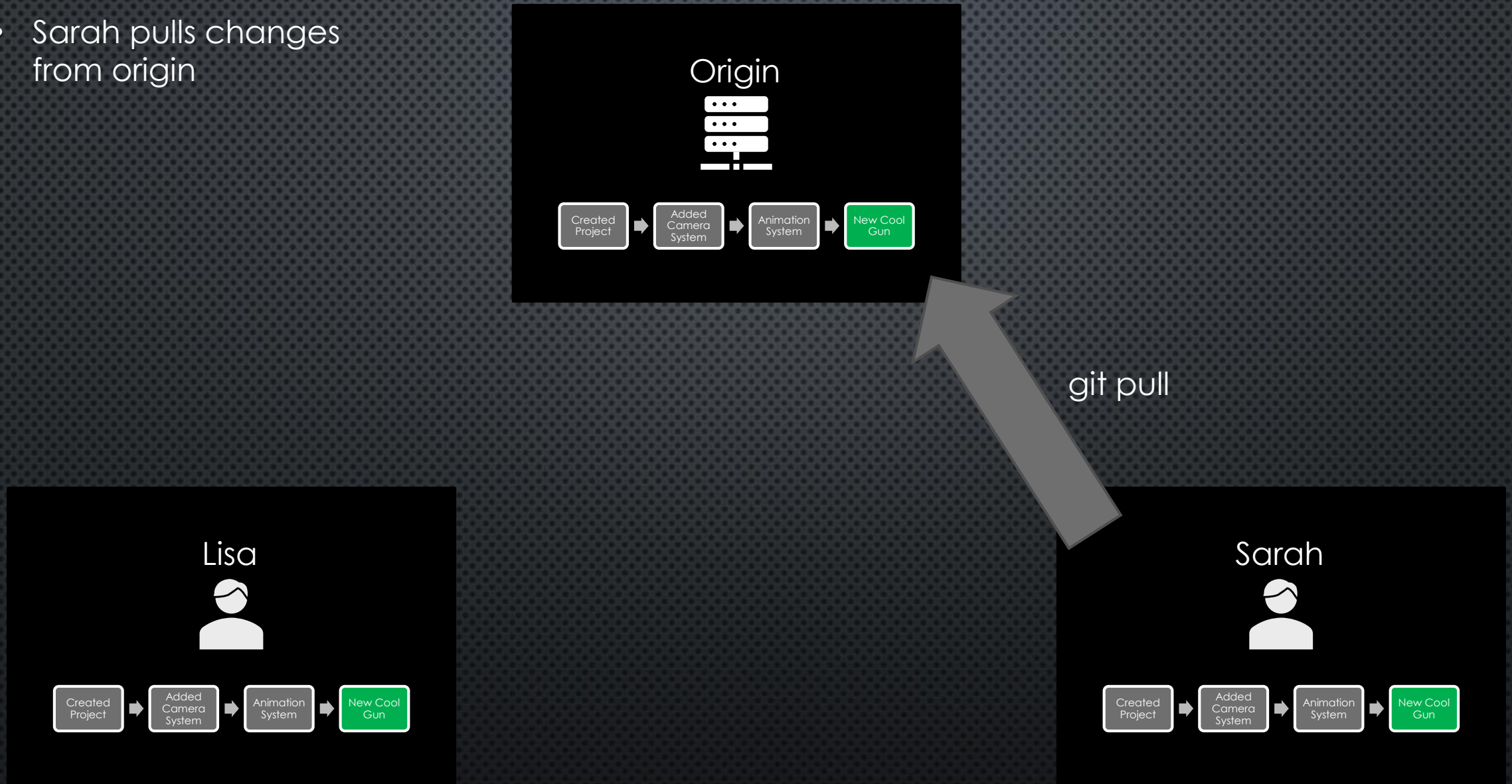
- Lisa creates a new feature “New Cool Gun” and commits it to her local repository



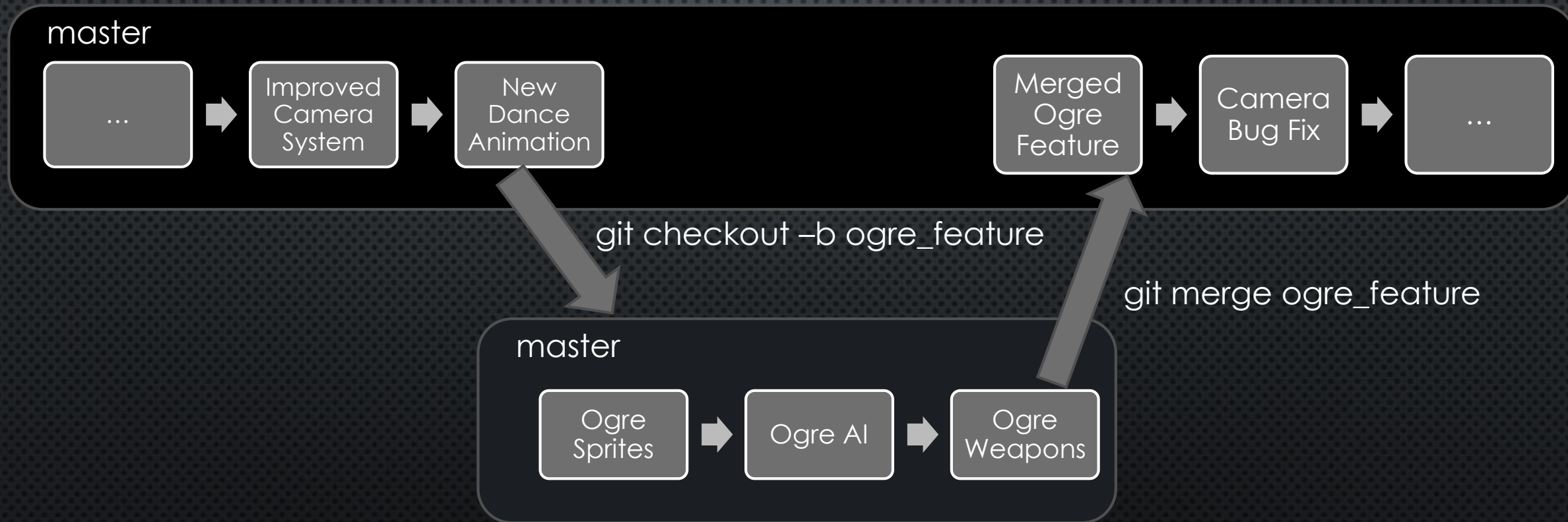
- Lisa pushes her changes to origin



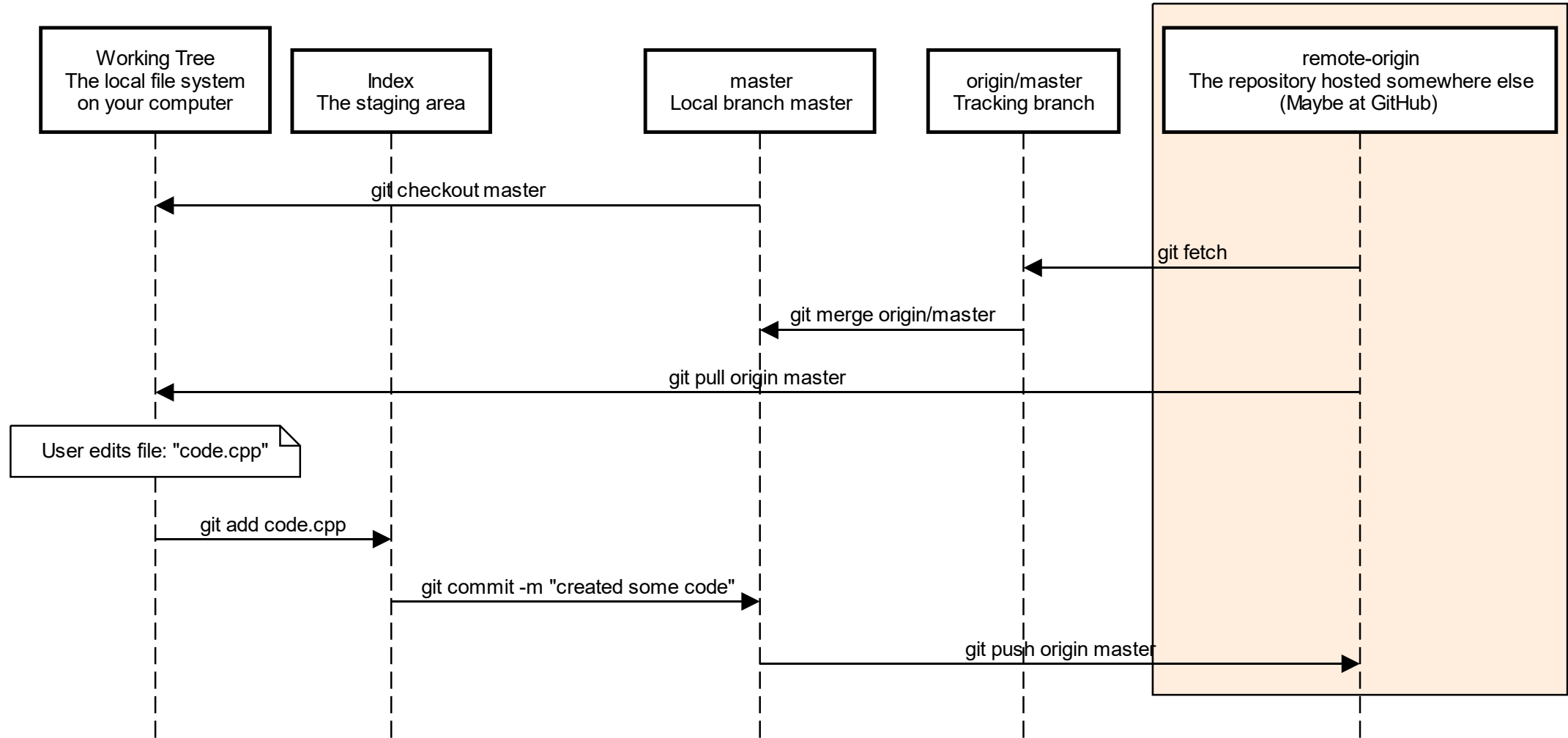
- Sarah pulls changes from origin



BRANCHES



Git Workflow



MOST COMMON AND USEFUL GIT COMMANDS

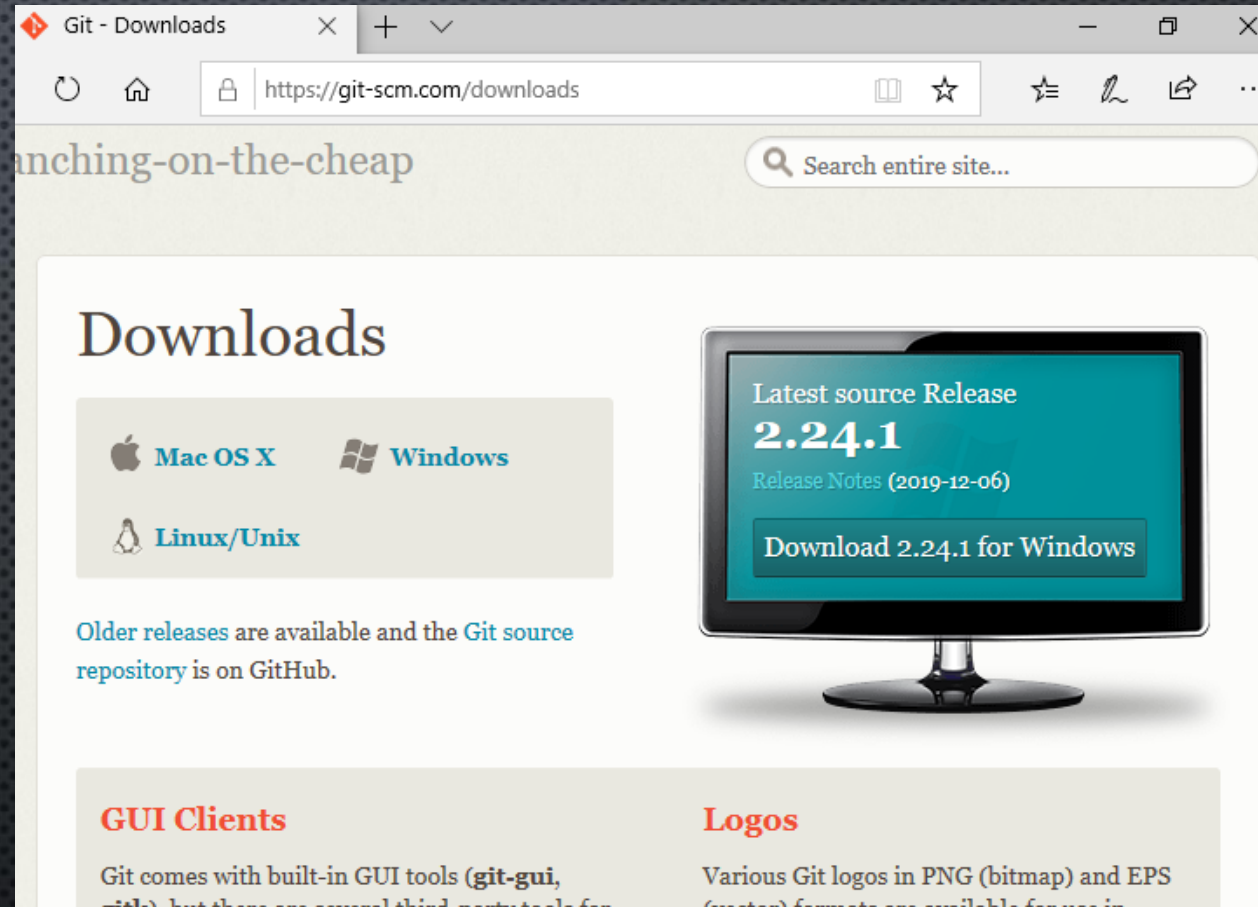
<code>git init</code>	# Inits a new repository locally
<code>git clone <repository></code>	# Clones a repository and creates a new remote "origin" pointing at the <repository>
<code>git checkout -b feature_x</code>	# Creates a new branch called "feature_x"
<code>git push origin feature_x</code>	# Pushes feature_x to the remote "origin"
<code>git push -u origin feature_x</code>	# Pushes feature_x to the remote "origin" and sets up tracking for that branch
<code>git pull origin feature_x</code>	# Get all changes on feature_x from origin.
<code>git checkout master</code>	# Checkouts branch master
<code>git merge feature_x</code>	# Merges feature_x into current branch
<code>git stash -u</code>	# Stashes all changes you've made so that your working tree is clear
<code>git stash pop</code>	# Remove a single stashed state from the stash list and apply it on top of the current working tree state
<code>git clean -xdf</code>	# Cleans your working tree (not safe)
<code>git reset branch_or_commit</code>	# Sets HEAD to commit, relatively safe in its default form
<code>git revert <commit></code>	# Reverts a single commit (Commonly reset would be used
<code>git gui</code>	# Opens git GUI where staging, commit and push can be done in a GUI
<code>gitk --all</code>	# Opens commit viewer GUI. Very useful for reviewing changes, resetting, tagging and reverting.
<code>git status</code>	# View the current branch, remote tracking branch, lists any staged and unstaged files
<code>git log</code>	# Quick look at history (normally gitk is used)

SETUP & GETTING STARTED

INSTALL GIT AND CREATE A GITHUB ACCOUNT

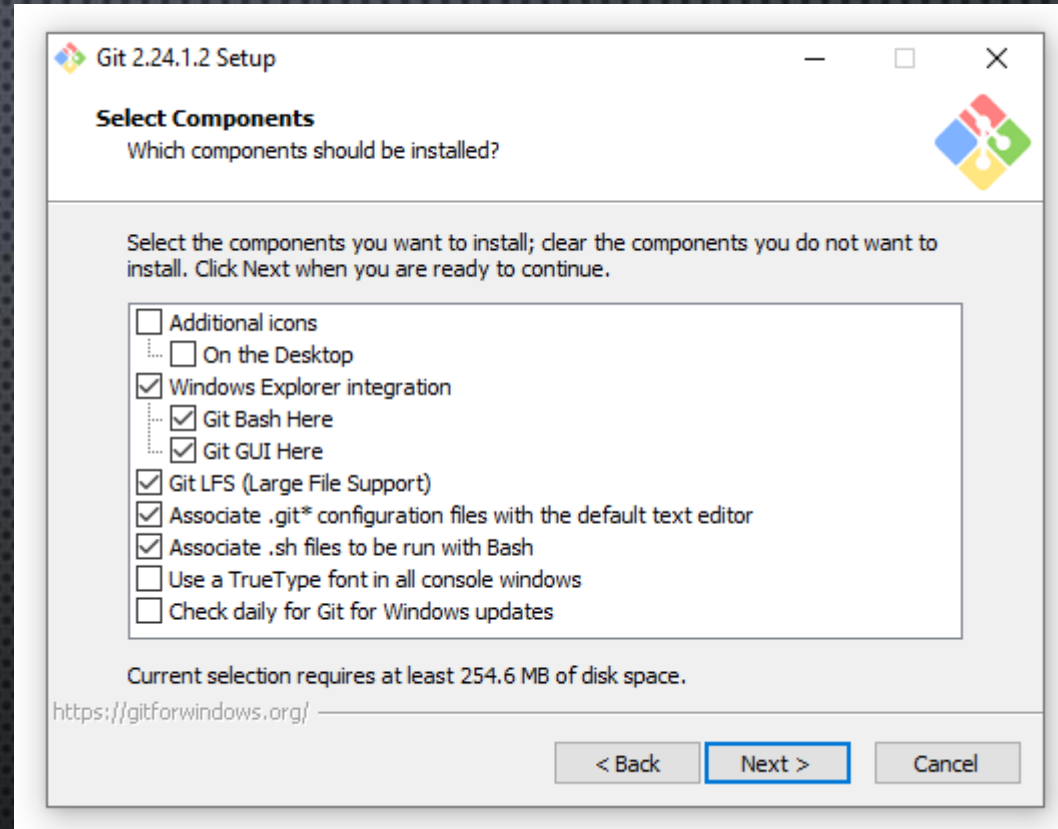
DOWNLOAD

- GIT-SCM.COM/DOWNLOADS
- DOWNLOAD THE LATEST STABLE RELEASE



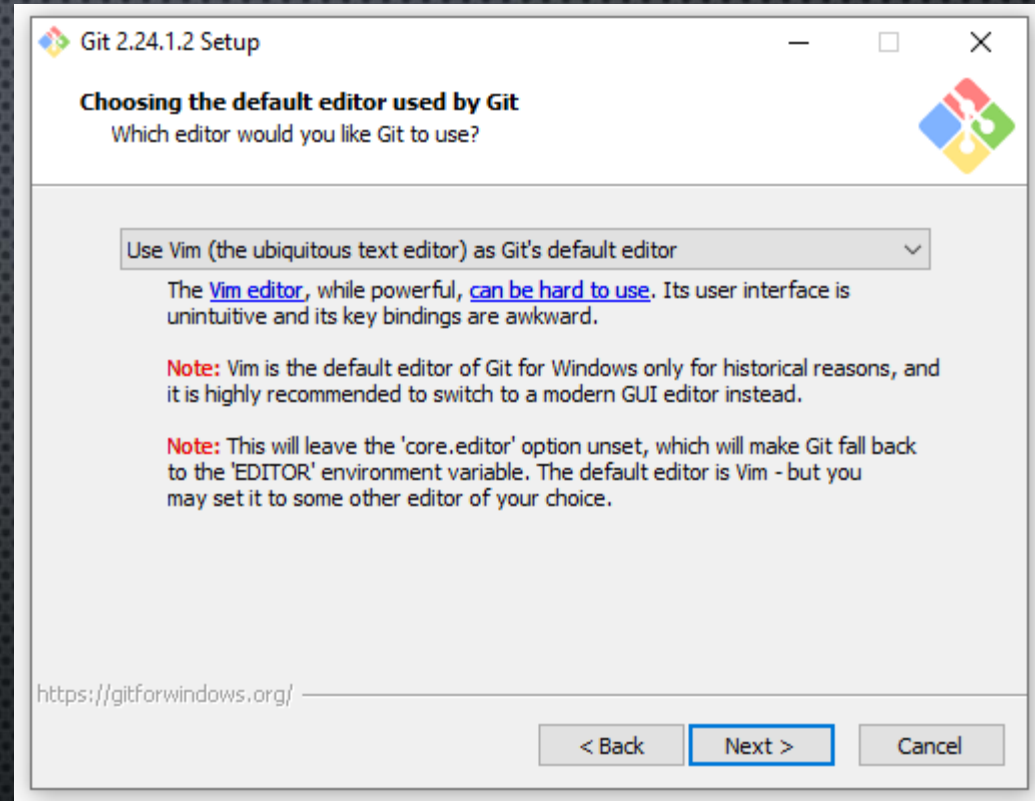
SETUP

- FOLLOW THE INSTRUCTIONS



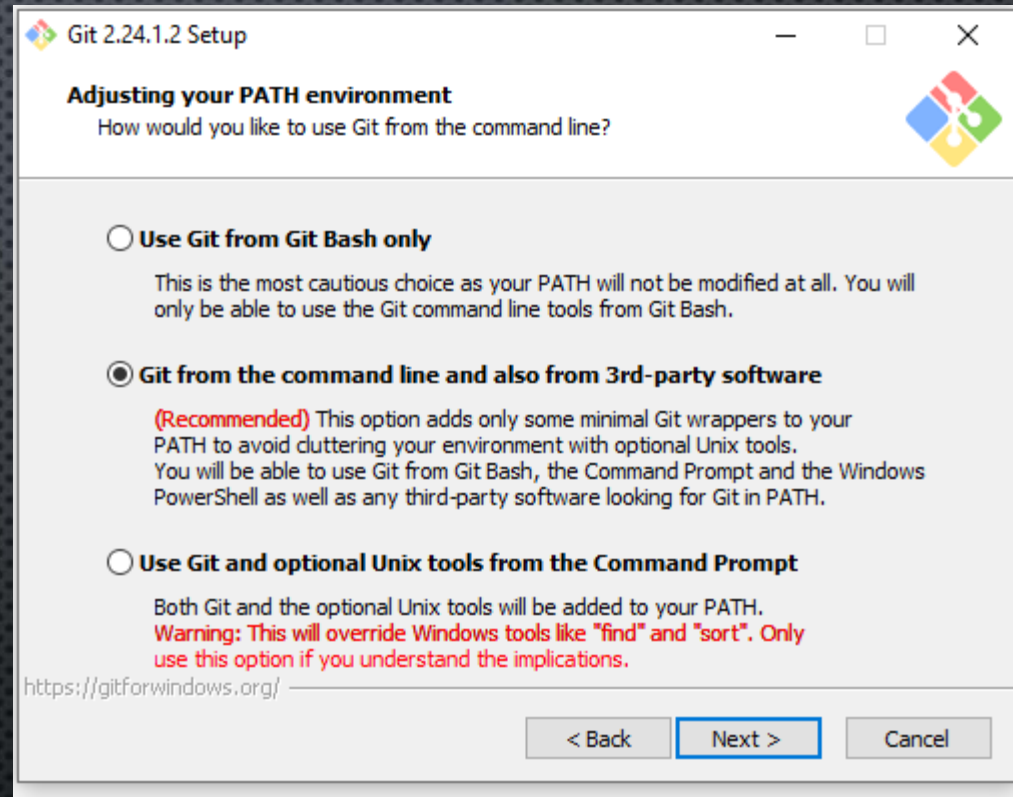
SETUP

- CHOOSE THE TEXT EDITOR OF YOUR CHOICE (OR JUST GO WITH VIM)



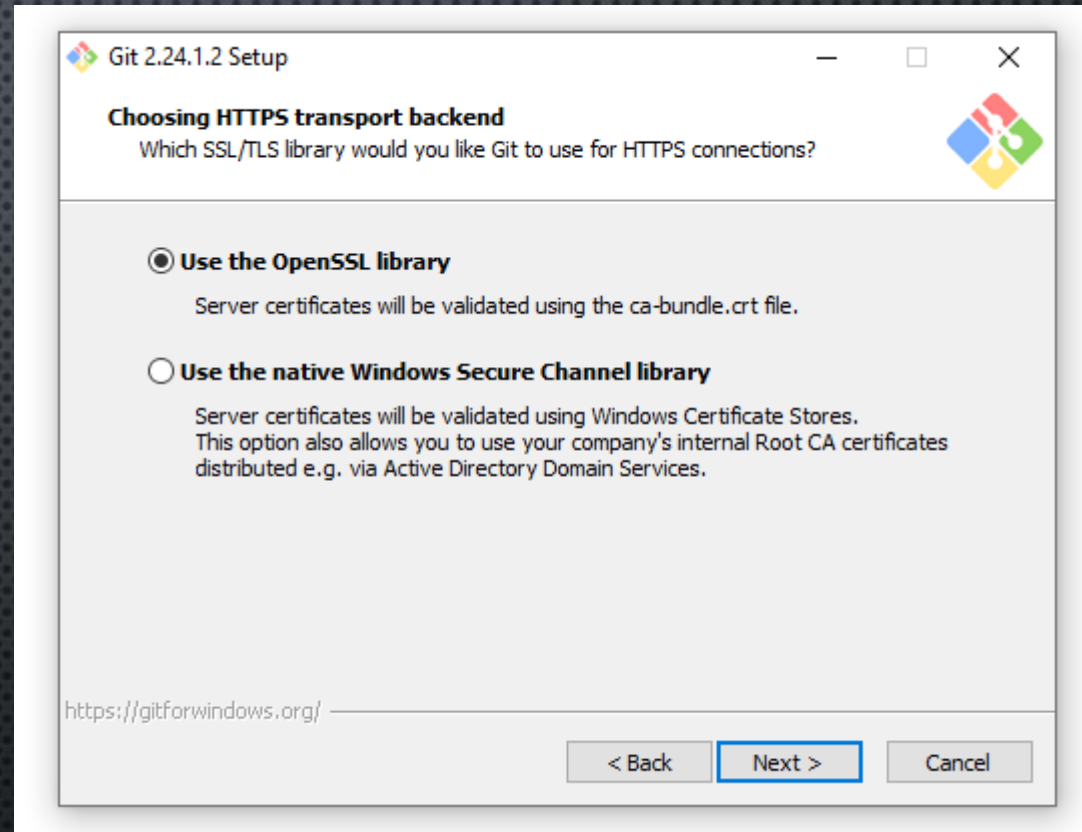
SETUP

- IN THIS SCREEN, CHOOSE THE **(RECOMMENDED)** OPTION



OPENSSL

- USE THE OPENSSL LIBRARY

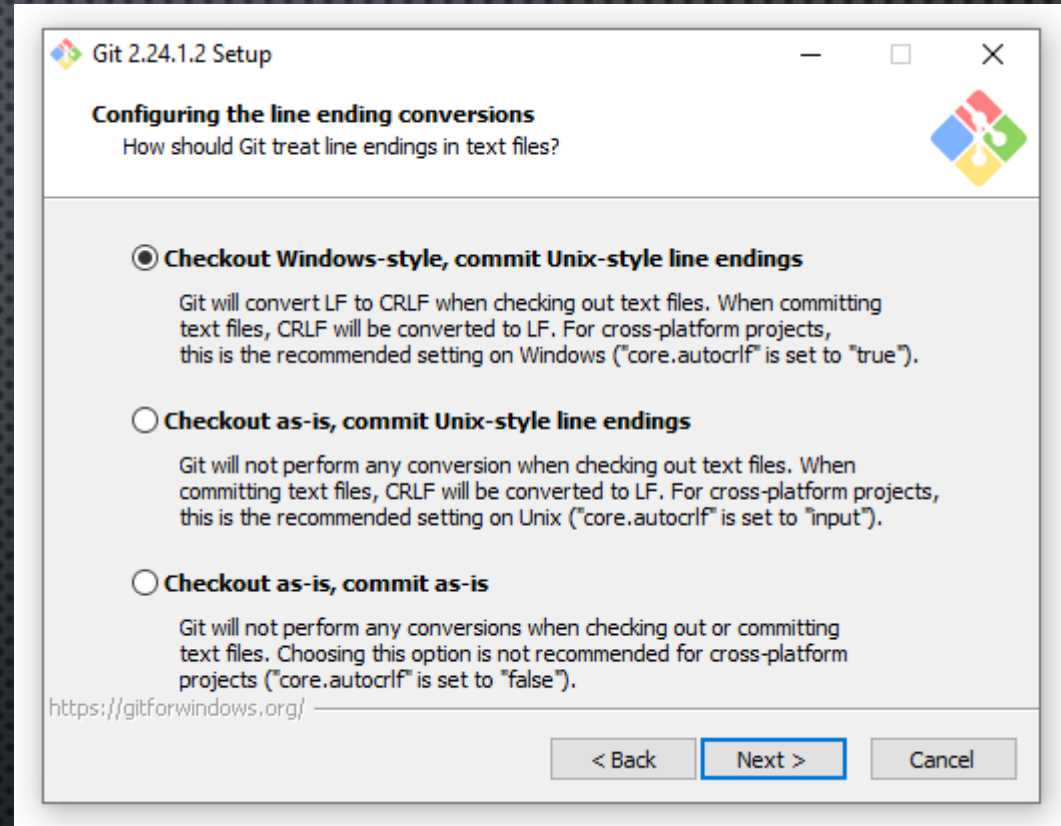


LINE ENDINGS

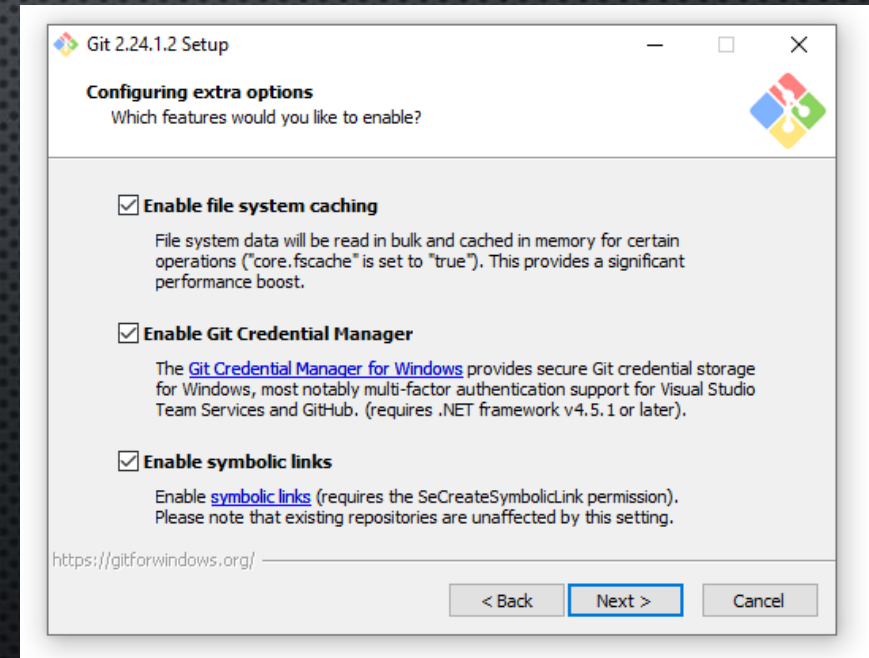
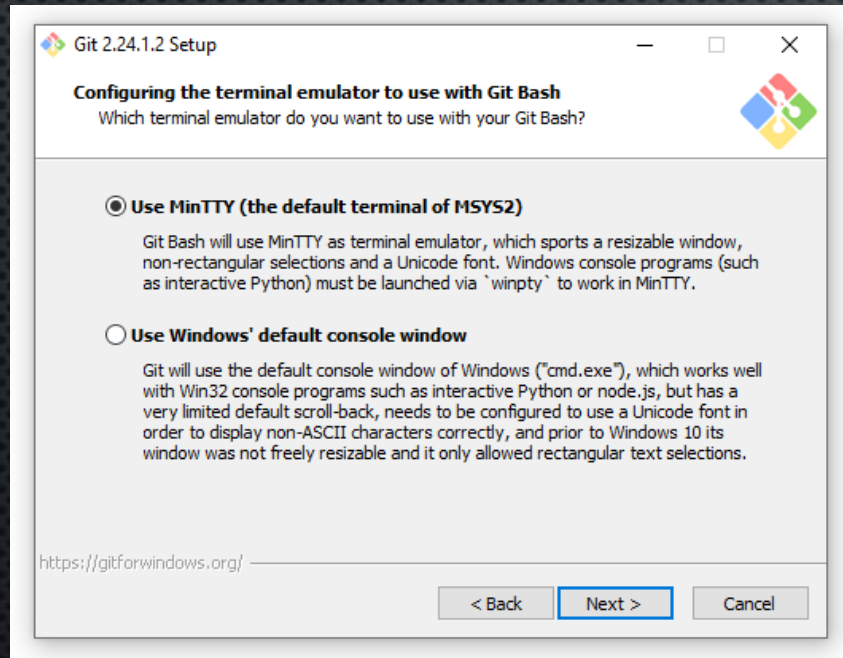
- IT IS GENERALLY RECOMMENDED TO USE "CHECKOUT WINDOWS-STYLE, COMMIT UNIX-STYLE LINE ENDINGS"

BUT

- IT CAN IN SOME CASES BE VERY ANNOYING



CONTINUE THE SETUP USING THE DEFAULT SETTINGS

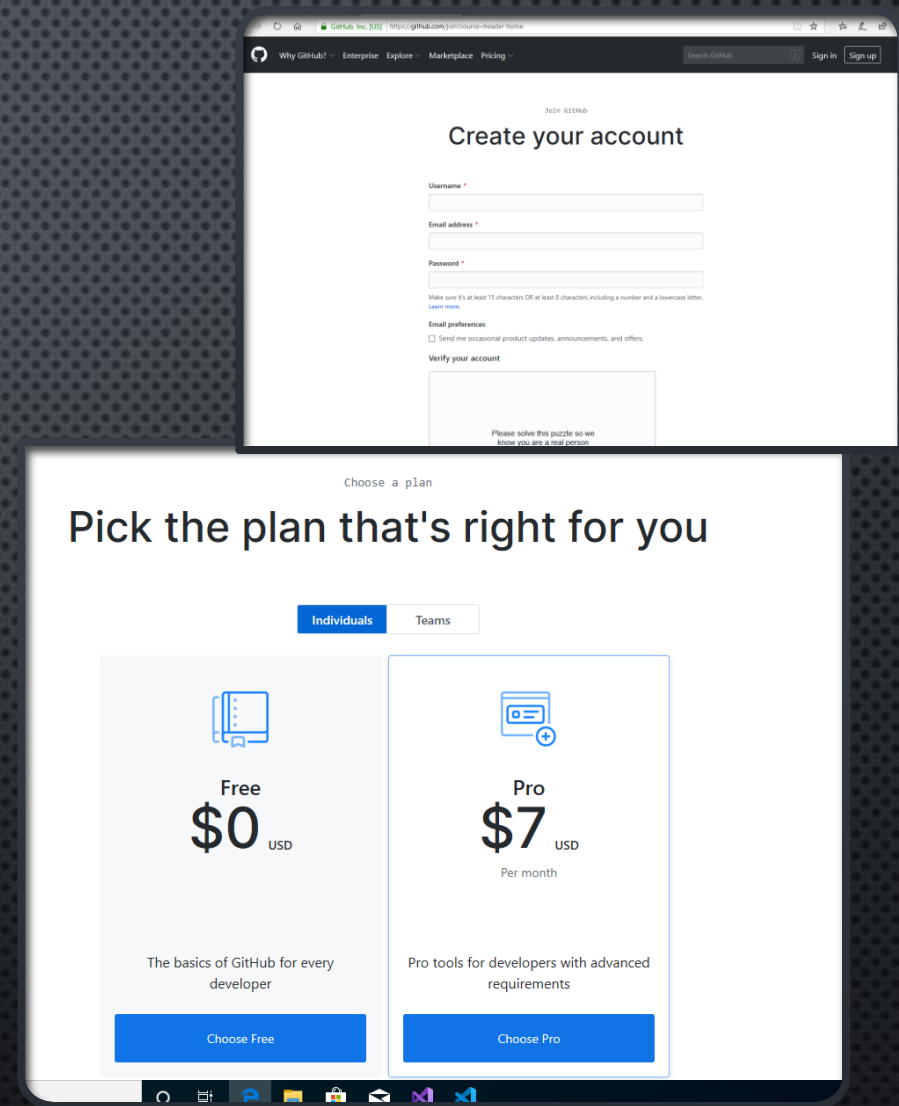


GITHUB

- CREATE GITHUB ACCOUNT
- SETUP SSH KEY FROM YOUR COMPUTER TO YOUR GITHUB ACCOUNT

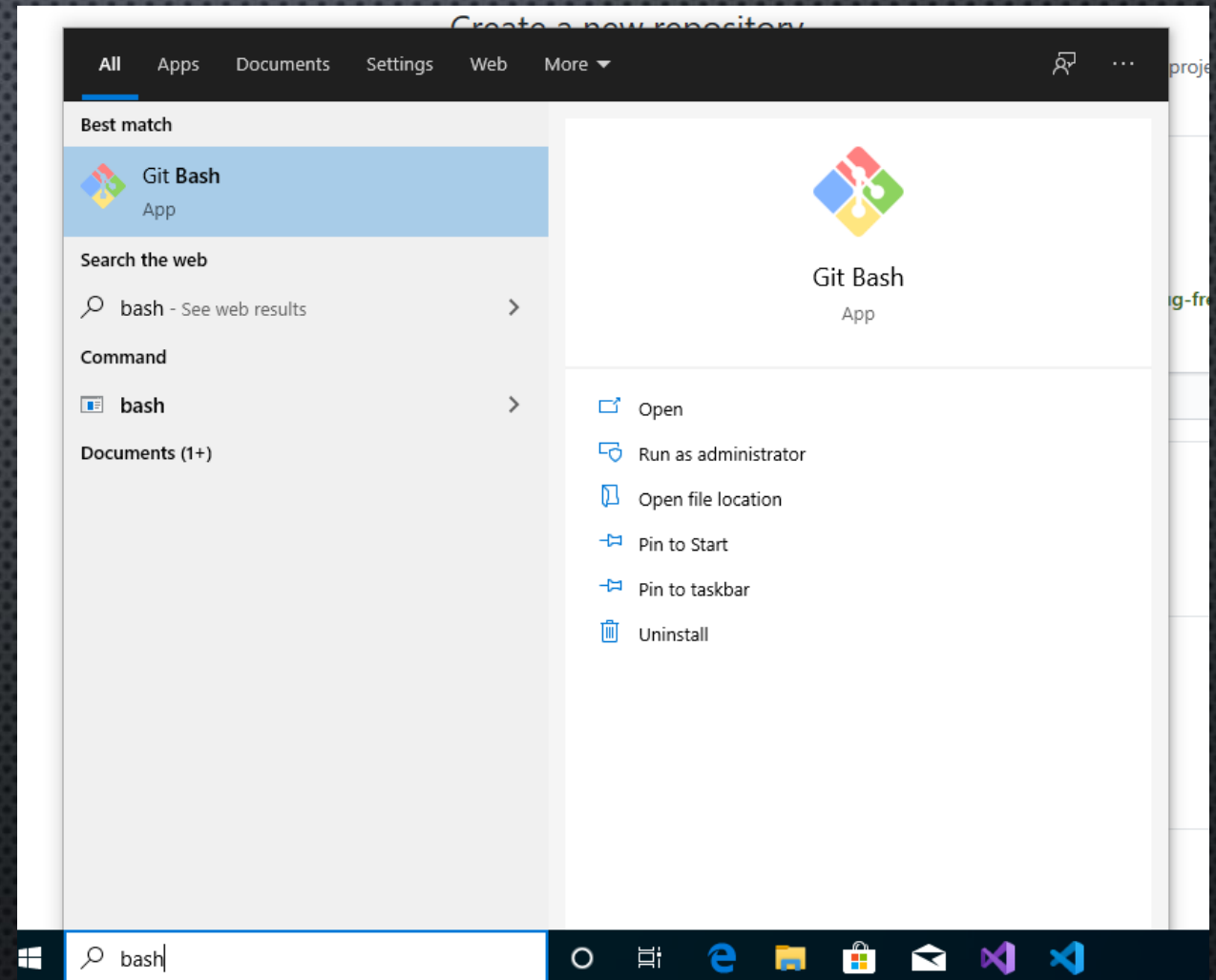
CREATE ACCOUNT

- GO TO GITHUB.COM
- CREATE AN ACCOUNT
- SELECT THE FREE PLAN



SSH KEY

OPEN BASH

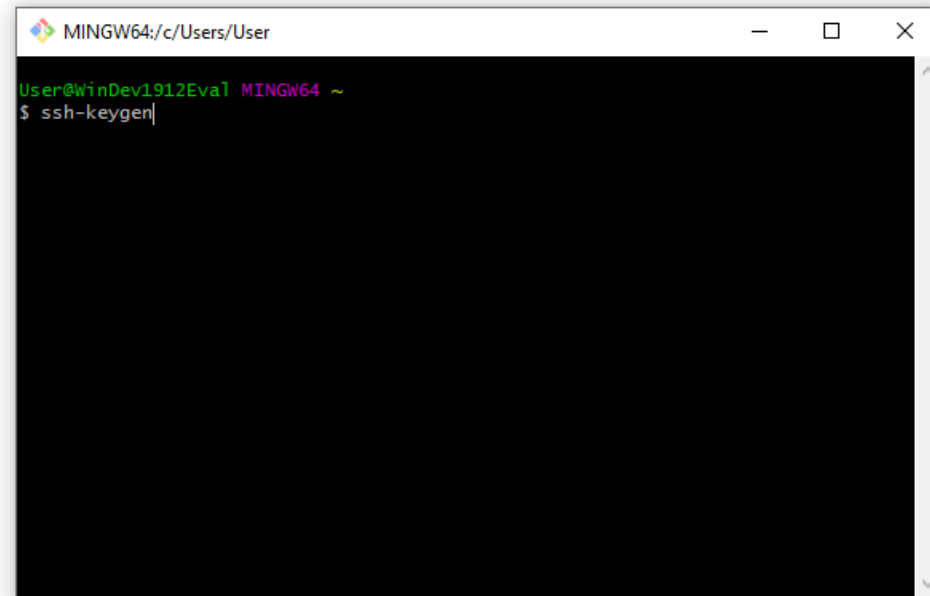


SSH-KEYGEN

TYPE:

SSH-KEYGEN

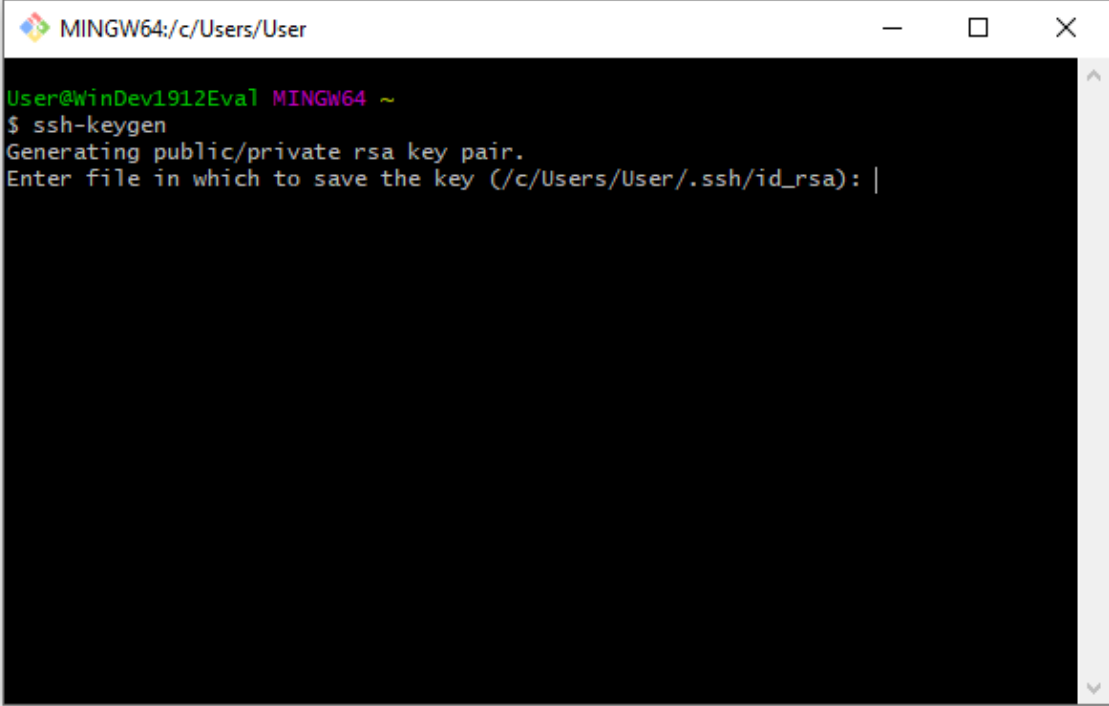
AND PRESS ENTER



```
MINGW64:/c/Users/User
User@WinDev1912Eval MINGW64 ~
$ ssh-keygen|
```


SSH-KEYGEN

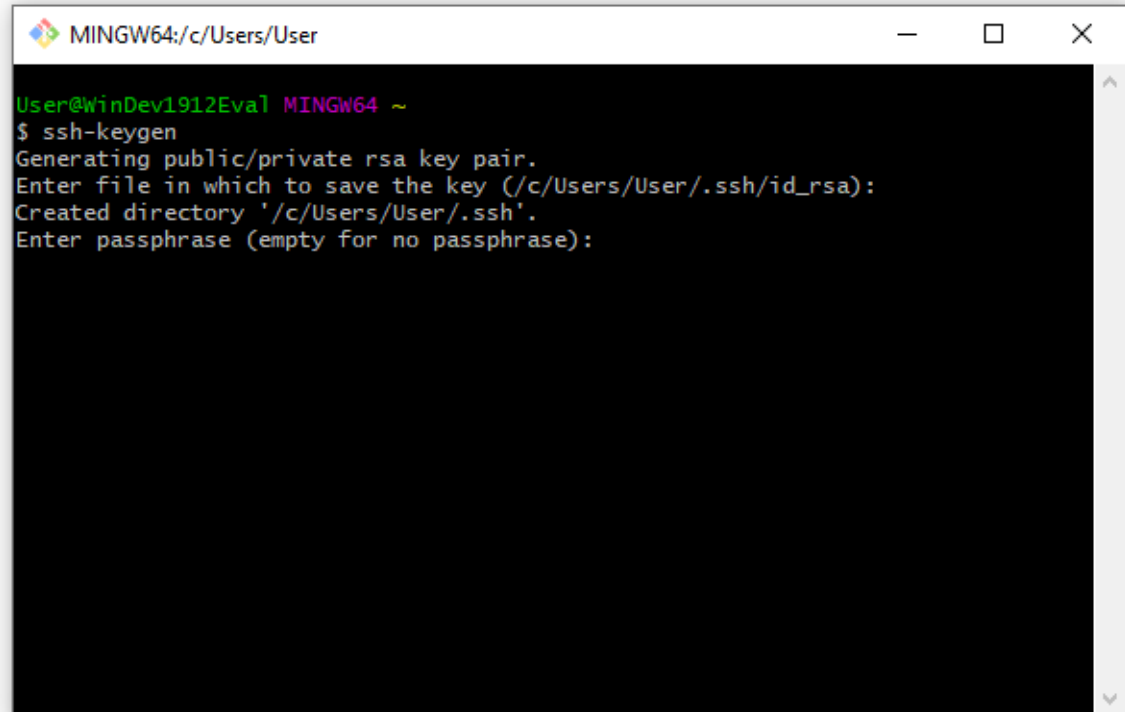
- PRESS ENTER TO SAVE THE KEY IN THE DEFAULT LOCATION



```
MINGW64:/c/Users/User
User@WinDev1912Eval MINGW64 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/User/.ssh/id_rsa): |
```

SSH-KEYGEN

- PRESS ENTER FOR NO PASSPHRASE
- TWO FILES ARE NOW CREATED, ID_RSA & ID_RSA.PUB
- ID_RSA.PUB HOLDS THE PUBLIC KEY THAT YOU CAN DISTRIBUTE

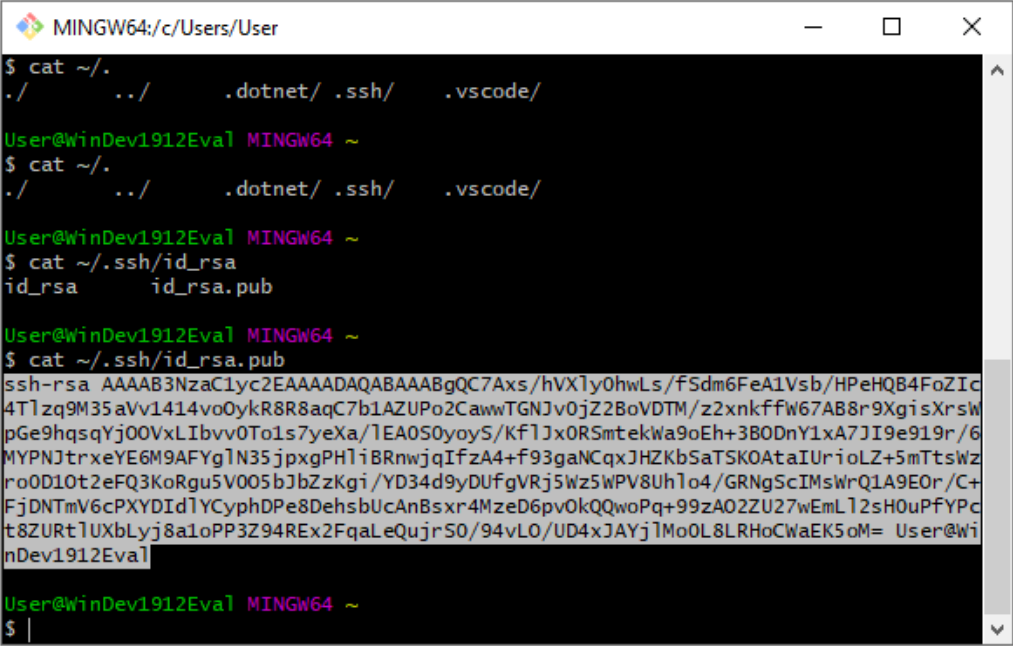


```
MINGW64:/c/Users/User

User@WinDev1912Eval MINGW64 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/User/.ssh/id_rsa):
Created directory '/c/Users/User/.ssh'.
Enter passphrase (empty for no passphrase):
```


COPY THE SSH KEY

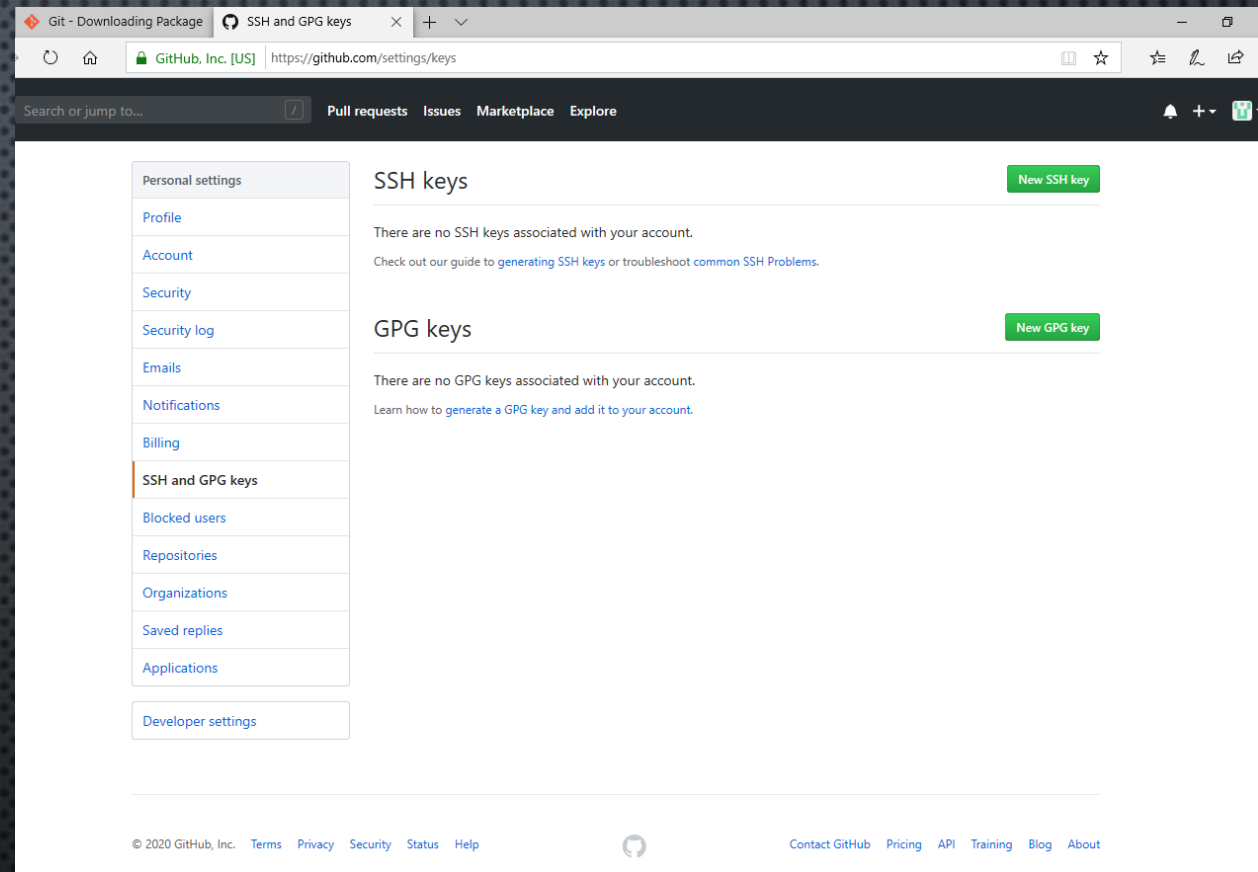
- CAT ~/.SSH/ID_RSA.PUB
- COPY THE TEXT (TRIPPEL CLICK WITH LEFT MOUSE TO SELECT THE TEXT & CTRL+INS TO COPY



```
MINGW64:/c/Users/User
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAxsfhVXly0hwLs/fSdm6FeA1Vsb/HPeHQB4FoZIC
4TlZq9M35aVv1414voOykR8R8aqC7b1AZUPo2CawwTGNJv0jZ2BoVDTM/z2xnkffw67AB8r9XgisXrsw
pGe9hqsqYj00VxLIbvv0To1s7yeXa/1EA0S0yoyS/Kf1Jx0R5mteKwa9oEh+3B0DnY1xA7JI9e919r/6
MYPNJtrxeYE6M9AFYg1N35jpxgPH1iBRnwjqIfzA4+f93gaNCqxJHZKbSaTSK0AtaIurioLZ+5mTtsWz
ro0D10t2eFQ3KoRgu5V005bJbZzKgi/YD34d9yDUfgVRj5Wz5WPV8Uhl04/GRNgScIMswrQ1A9E0r/C+
FjDNTmV6cPXyDIIdlYCyphDPe8DehsbUcAnBsxr4MzeD6pv0kQQwoPq+99zA02ZU27wEmL12sH0uPfYPc
t8ZURtlUXbLyj8a1oPP3Z94REx2FqaLeQujrS0/94vLO/UD4xJAYj1Mo0L8LRHoCwaEK5oM= User@Wi
nDev1912Eval
$
```

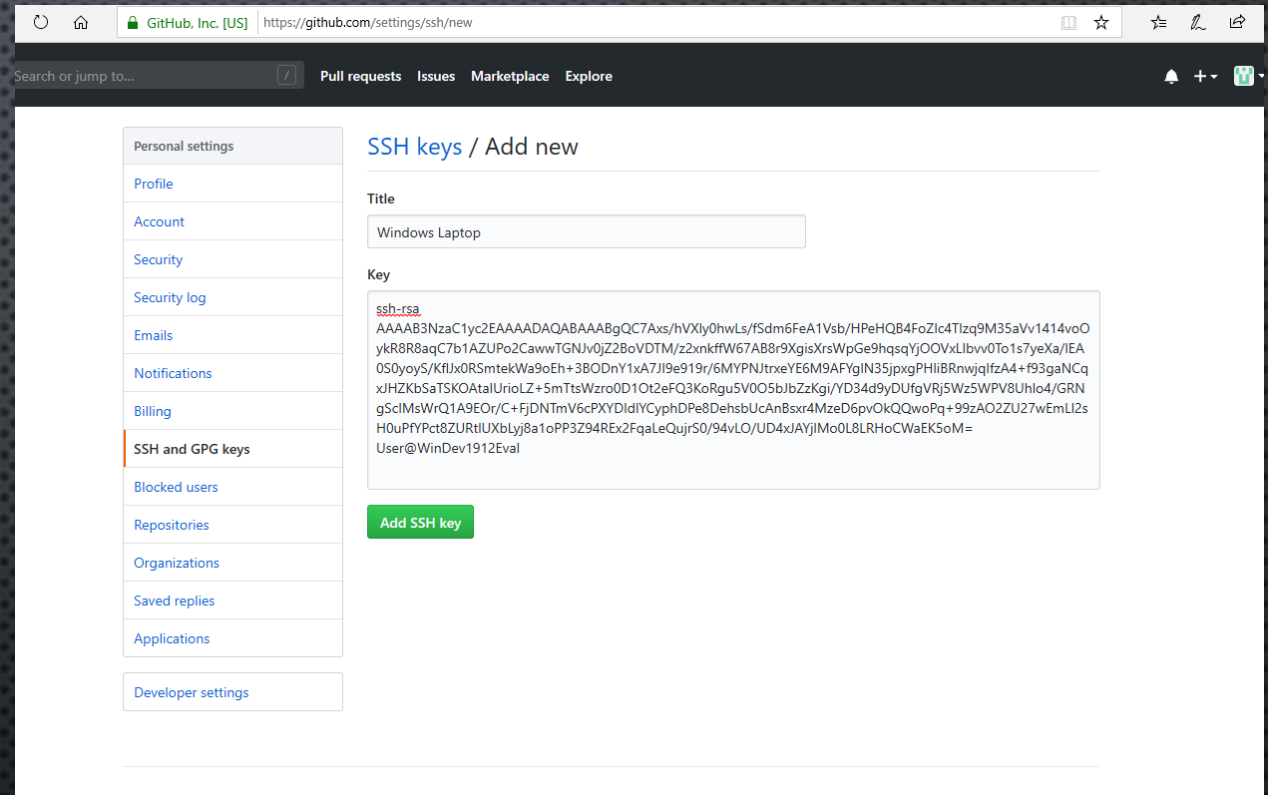
REGISTER SSH KEY

- GITHUB.COM/SETTINGS/KEYS
- CLICK “NEW SSH KEY”



REGISTER SSH KEY

PASTE THE SSH-KEY, GIVE IT A NAME AND
CLICK “**ADD SSH KEY**”.



The screenshot shows the GitHub 'SSH keys / Add new' page. On the left is a sidebar with navigation links: Personal settings, Profile, Account, Security, Security log, Emails, Notifications, Billing, SSH and GPG keys (highlighted), Blocked users, Repositories, Organizations, Saved replies, Applications, and Developer settings. The main content area has the title 'SSH keys / Add new'. Below the title is a 'Title' field containing 'Windows Laptop'. Underneath is a 'Key' field containing a long SSH key starting with 'ssh-rsa'. At the bottom of the key field is a green 'Add SSH key' button.

Personal settings

Profile

Account

Security

Security log

Emails

Notifications

Billing

SSH and GPG keys

Blocked users

Repositories

Organizations

Saved replies

Applications

Developer settings

SSH keys / Add new

Title

Windows Laptop

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGC7Axs/hVXly0hwLs/fSdm6FeA1Vsb/HPeHQB4FoZlc4TlZq9M35aVv1414voO
ykR8R8aqC7b1AZUPo2CawwTGNJv0jZ2BoVDTM/z2xnkfW67AB8r9XgisXrsWpGe9hqsqYjOOVxLlbvv0To1s7yeXa/EA
0S0yoyS/KfIJx0RSmtekWa9oEh+3BODnY1xA7JI9e919r/6MYPNJtrxeYE6M9AFYgIN35jpxgPHliB8rnwjqIfzA4+f93gaNCq
xJHZKbSaTSKOAtalUrioLZ+5mTtsWzro0D1Ot2eFQ3KoRgu5V0O5bJbZzKgi/YD34d9yDUfgVRj5Wz5WPV8Uhlo4/GRN
gScIMsWrQ1A9EOrr/C+FjDNTmV6cPXYDIdlYCyphDPe8DehsbUcAnBsxr4MzeD6pvOkQQwoPq+99zAO2ZU27wEmLI2s
H0uPfYPct8ZURtlUXblyj8a1oPP3Z94REx2FqaLeQujrS0/94vLO/UD4xJAfjiMo0L8LRHoCWaEK5oM=
User@WinDev1912Eval
```

Add SSH key

USER NAME AND EMAIL

- YOU NEED TO CONFIGURE YOUR USERNAME AND EMAIL

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

```
git config --global user.name "Your Name"
```


SOME POPULAR GIT GUI CLIENTS...

- VISUAL STUDIO
 - VISUAL STUDIO HAS BUILT IN SUPPORT FOR MOST COMMON GIT COMMANDS
- SOURCE TREE
 - VERY POPULAR
- GITHUB DESKTOP
 - GITHUB'S OWN CLIENT. SIMPLE TO USE.
- GIT KRAKEN
 - POPULAR AT UPPSALA UNIVERSITY

REFERENCE

- [HTTPS://GIT-SCM.COM/DOCS](https://git-scm.com/docs)
 - GIT REFERENCE
- [HTTPS://ROGERDUDLER.GITHUB.IO/GIT-GUIDE/](https://rogerdudler.github.io/git-guide/)
 - VERY GOOD, VERY SIMPLE GIT TUTORIAL
- [HTTPS://GITHUB.COM/](https://github.com/)
- [HTTPS://GIT-SCM.COM/DOWNLOAD/GUI/WIN](https://git-scm.com/download/gui/win)
 - LIST OF WINDOWS GIT GUI CLIENTS