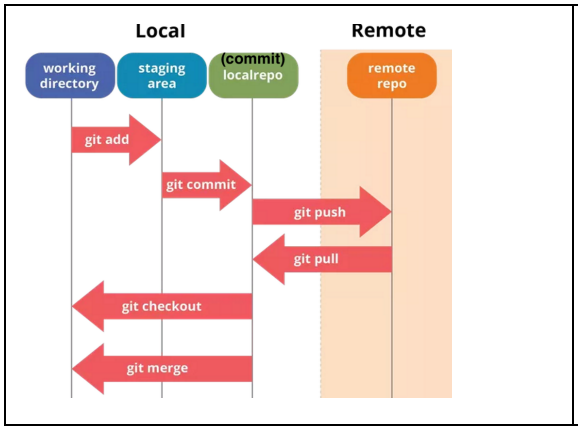
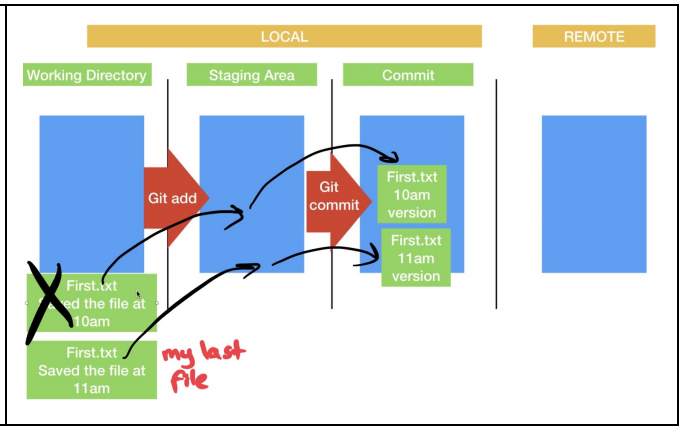
GIT - Version Control System

- ***Version***?
  - Any modification, new file, new feature, deletion
- ***Why do we need version/source control system?***
  - Backup / history / versioning / Archive
  - Undo changes
  - Collaboration and teamwork (to work each other's files)
  - Blame (who did **break the build**, who did mistake)
  - Experiment
  - Code review
- ***Repository***: storage, collection
  - Github is where we put the repository
    - Track any type of text files (.doc, .js, .java …)
- ***Version Control Systems***
  - ***Distributed*** (decentralized) version control system
    - Most operations are local, no need a network connection, central server is required
    - Everyone has the copy of the repository.
  - ***Centralized*** version control system like TFS
    - Requires a **network connection** between your computer and central repository
    - If we cannot reach the central repository we cannot work/see the files that we are working on
- **GIT vs GITHUB**
  - ***Git*** is a version control software, whereas ***Github*** is a hosting service like google drive. Cloud repository / Storage.

GIT WORKFLOW





- https://git-scm.com/ → install appropriate version for your system

GIT COMMANDS

- **sudo install git**
  - **xcode-select --install** (for mac because installation starts from terminal)
- **git --version** → how to learn if we have git in our comp
- **git init** → initiate
- **git status** → to see the latest info about the project
- **git add** filename → adds one file
- **git commit -m "Message"** → commits the repository
- **git config --global user.name "Username"** → introduce yourself to git
- **git config --global user.email "Your email"**
- **git log** → gives the history of commits
- **git log --graph --oneline --decorate** → history in a better format
- **git config --global alias.s "status"** → **git s** (create shortcut)
- **git commit -am "Express Commit"** → adds and commits already tracked files
- **git clone [url]** → create a local repo and pulls everything on remote repo
- **git push origin master** → if we are behind remote repo, downloading everything from remote repo
- **git push -u origin master** → uploads all commits on local to remote
- **git remote add origin [url]** → connects remote repo with local repo

UNIX COMMANDS

- **pwd** → print/present working directory
- **ls** → list the folders and files inside the folder
  - **ls -a** → show hidden files
  - **ls -al** → show all files in a list format
- **cd** → change directory
  - **cd..** → go one up directory
  - **cd ~** → go to top folder
  - **cd** +drag&drop the folder which you want
- **mkdir** foldername: create a folder on pwd with a name
- **clear** → clears the screen

Start using Git with Projects by 3 ways

1. **FRESH PROJECT**

- Configuration (Configure user information for all local repositories)
  - **$ git config --global user.name "Eyup"** → introduce yourself to git
  - **$ git config --global user.email "eyupaydinusa@gmail.com"** → introduce yourself to git
  - **$ git config --global color.ui auto** → beautiful appearance
  - **$ git config --global --list** → list all username etc on the screen
- Create a Folder and named `git-projects` (**mkdir git-projects**) and create another folder `myFirstRepo` inside it
- Go to terminal and reach your `myFirstRepo` like ⇒ macs-MBP:myfirstrepo mac$
- **git init** → (Initiate) it creates repository inside the folder and it is hidden file (to see the hidden file → ⇧ ⌘ .)
  - then **git status**

```
macs-MBP:myfirstrepo mac$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

- Create a `first.txt` file and put it inside the `myFirstRepo` folder
  - then **git status**

```
macs-MBP:myfirstrepo mac$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        first.txt

nothing added to commit but untracked files present (use "git add" to track)
```

- **git add** first.txt → Snapshots the file in preparation for versioning
  - then **git status**

```
macs-MBP:myfirstrepo mac$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   first.txt
```

- Create a new file second.txt and modified the first.txt
  - then **git status**

```
macs-MBP:myfirstrepo mac$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   first.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working

        modified:   first.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        second.txt
```

- **git add . || git add * || git add --all**
  - then **git status**

```
macs-MBP:myfirstrepo mac$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   first.txt
        new file:   second.txt
```

  - I lost first version of first.txt because I did not add it to the commit part
- **git commit -m 'Adding new lines'** → like **$ git commit -m "My first commit"**
  - then **git status** → give me commit summary
- **git log** → gives the history of commits

```
macs-MBP:myfirstrepo mac$ git log
commit 32df7ff23e51e4e5fccec74f6e5f16f20c4e23e4 (HEAD -> master)
Author: Eyup <eyupaydinusa@gmail.com>
Date:   Sat Dec 1 14:05:52 2018 -0500

    Adding new lines
```

  - **git log --oneline** →
  - **git log --graph --oneline** → more beautiful showing
  - **git log --graph --oneline --decorate** → history in a better format
- If we use a comment a lot we can change its shortcut like
  - **git config --global alias.s "status"** → **git s**
  - **git config --global alias.hist "log --graph --oneline --decorate --all"** → **git hist**

- After changing any line from a file
  - **git commit -am "Express Commit"** (The file already must be added before to use this command)

```
macs-MBP:myFirstRepo mac$ git commit -am "Express Commit"
warning: unable to access '/Users/mac/.config/git/attributes': P
[master f72e43c] Express Commit
warning: unable to access '/Users/mac/.config/git/attributes': P
 1 file changed, 2 insertions(+), 1 deletion(-)
macs-MBP:myFirstRepo mac$ git status
On branch master
nothing to commit, working tree clean
macs-MBP:myFirstRepo mac$ git log
commit f72e43c9368c890027ce75294ea1ccf84946d008 (HEAD -> master)
Author: Eyup <eyupaydinusa@gmail.com>
Date:   Wed Dec 5 12:01:20 2018 -0500

    Express Commit

commit 6adb98027a327db37434922bb82ff2e20c663cf3
Author: Eyup <eyupaydinusa@gmail.com>
Date:   Wed Dec 5 11:35:04 2018 -0500

    My first commit
macs-MBP:myFirstRepo mac$
```

2. EXISTING PROJECT

- Copy your project folder (backend-testing) inside **git-projects** folder
- **git init** → to initiate
- Create **.gitignore** file (it is hidden) with Sublime and save it into the backend-testing folder → I don't want to see all version of all my documents

```
.gitignore
1  node_modules
2  package-lock.json
3  *.log
```

- After ignore this file we will not see on **git status**

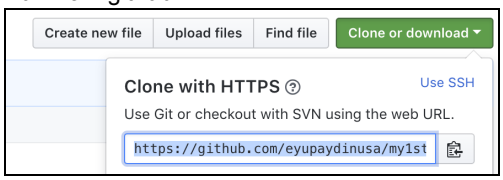
```
macs-MBP:backend-testing mac$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        Configuration/
        Pages/
        TestData/
        Tests/
        db.js
        dbAmazon.js
        level1/
        package.json
        report/

nothing added to commit but untracked files present (use "git add
```

- **git add .** → then **git status**
- **git commit -m "Initialize the project"** → then **git status** (If I see this below, I am good)

```
macs-MBP:backend-testing mac$ git status
On branch master
nothing to commit, working tree clean
```

3. Starting on Github by cloning your project

- Create new repository (write a name with all lower case) and inside it create new file
- Go to terminal and go to git-projects folder with **cd ..** command
- Copy your url from github

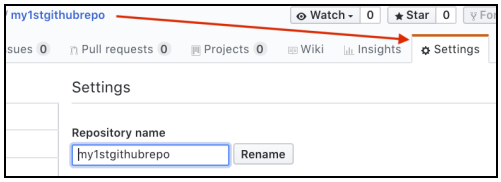


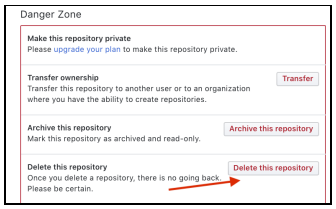
- **git clone https://github.com/eyupaydinusa/my1stgithubrepo.git**
  To get the file from github to my computer
- Create another file in your **my1stgithubrepo** folder with named second.txt and then **git status**

```
macs-MBP:my1stgithubrepo mac$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        second.txt

nothing added to commit but untracked files present (use "git ad
```

- **git commit -m "Initialize the project"** → to add it to commit
- **git push origin master** → To push our file from computer to github

- **Deleting a repository from a Github Account**

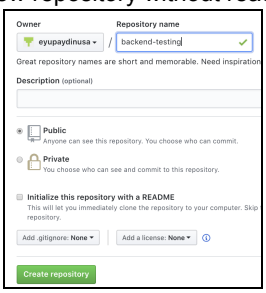




- **Creating Local Repository with Remote Repository (we have project in our local computer)**

```
macs-MBP:backend-testing mac$ git s
On branch master
nothing to commit, working tree clean
```

  - Create new repository without readme and the **same name** folder in your computer



  - **git remote add origin https://github.com/eyupaydinusa/backend-testing.git**
  - **git status**
  - **git push -u origin master** → take this repo and upload it from computer (you had already those file name)

  - Create new file **readme.txt** with sublime and save it in **backend-testing** folder

```
macs-MBP:backend-testing mac$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        readme.txt

nothing added to commit but untracked files present (use "git add" to track)
```

  - **git add** → **git status**

```
macs-MBP:backend-testing mac$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   readme.txt
```

  - **git commit -m "Adding readme file"** → then **git status**

```
macs-MBP:backend-testing mac$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

  - **git push origin master** → **then git status**

```
macs-MBP:backend-testing mac$ git s
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

  - **git hist** → we create before → git log --graph --oneline --decorate

```
macs-MBP:backend-testing mac$ git hist
* 703a8b2 (HEAD -> master, origin/master) Adding readme file
* 90e8d00 Initialize the project
```

- **How to change the username and password on Mac from Terminal**
  - **git -credential- osxkeychain erase** → Enter
  - **Host = github.com** → Enter
  - **Protocol = https** → Enter
  - Enter
  - Enter