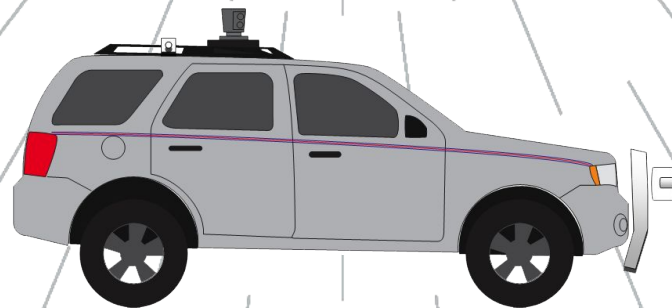




# CAT VEHICLE REU 2019

Introduction to project management with Git

Rahul Bhadani <[rahulbhadani@email.arizona.edu](mailto:rahulbhadani@email.arizona.edu)>



# What is version control?

- ❖ Records changes to file or set of files with timestamp and remembers what changes you made.
- ❖ Helpful when you want to roll back your project to a certain time,
- ❖ Or you did something wrong and everything stops working and you wish you had not made those changes
- ❖ Or, your computer crashed !
  
- ❖ Also helpful when you are working in team, so that you know who made what changes

# Publicly available version control

- ❑ Based on Git:

- ❑ GitHub: <https://github.com>
- ❑ Gitlab: <http://gitlab.com>
- ❑ Bitbucket: <https://bitbucket.org/product/>

- ❑ Other version control:

- ❑ Subversion
- ❑ Mercurial

- ❑ U of A Engineering version control:

- ❑ Gitlab: <http://gitlab.engr.arizona.edu>
- ❑ We will use this, you login with your netid and password.

# Engr Git

- ❑ CAT Vehicle REU group will create projects in their designated group: <https://git.engr.arizona.edu/reu2019>
- ❑ Create separate projects inside the group: probably one for each individual to log and record your daily research journals, papers, etc. that you will read, any summary, code, etc; and one group project that you will be working in collaboration with other team members.

# Cloning a sample repo

- ❑ `git clone https://git.engr.arizona.edu/reu2019/git-tutorial`
- ❑ This will clone [copy in the language of git] the repository in your local system

# Adding your git username

- ❑ `git config --global user.name <yournet id>`
- ❑ `git config --global user.email <yournetid>@email.arizona.edu`

Note: everything inside angular bracket is a placeholder, you should replace them with appropriate texts: e.g.

```
git config --global user.name wilmawildcat
```

```
git config --global user.email wilmawildcat@email.arizona.edu
```

# Creating your own repository

- ❑ On Website:
  - ❑ Choose **new project** option in the gitlab website.
  - ❑ Enter the name you want to give.
  - ❑ Add description and enable readme that will tell any visitor what this project is about
- ❑ In your local system:
  - ❑ **git clone** <the name of your repository>

# Adding a file to your git project

- ❑ Check if there is unversioned files in your git folder on your local system
  - ❑ Type **git status**
  - ❑ Untracked files means they are unversioned
  - ❑ Modified means, you have made some changes to it
  - ❑ To see the difference between current version you have locally and one in the git, type **git diff <name of the file>**
  - ❑ To add files: **git add < name of the file>**
  - ❑ To add all files: **git add --all**



# Committing your changes

- ❑ At this point, you have told the git software that these files are ready to be added
- ❑ Now commit these changes to the local database with a descriptive message:
  - ❑ `git commit -m "This is my first git commit" <filenames>`
- ❑ You can do multiple commits to different files with different commit messages
- ❑ It is a good practice to commit with a meaningful message about changes you are going to commit: it will save a lot of trouble in the future! Trust me!

# Pushing changes to git server

- ❑ `git push -u origin master`
- ❑ Here, `master` is the name of the branch you are committing to.
- ❑ There can be multiple branches [more on [this](#) later].

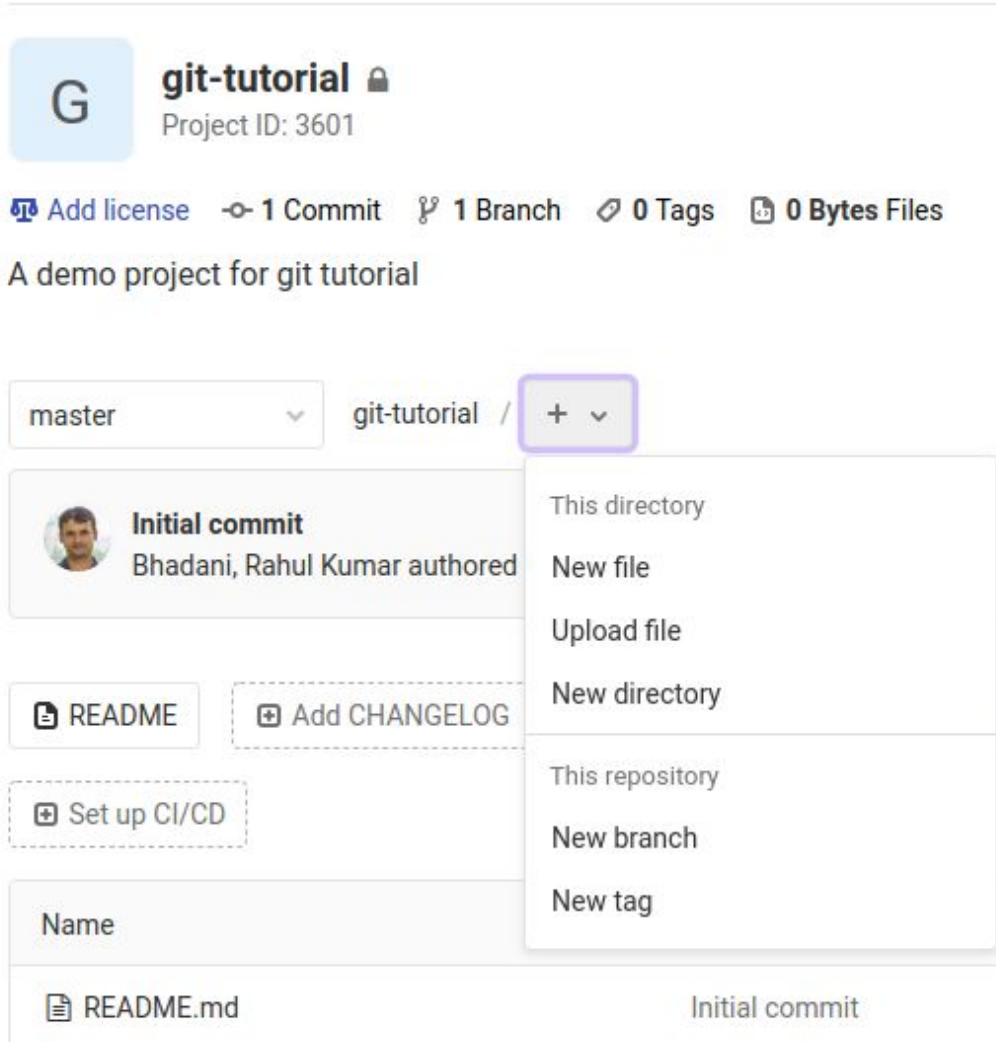
# Retrieving changes from git

- ❑ You are working in a team and your teammates made some commit and pushed to git. You want to retrieve that changes before you commit.
- ❑ **git pull**
- ❑ Always pull to check to see if there is some update in the repo you before push.

# Teach git some exception rules

- ❑ There are certain files we don't need to commit to the git server
  - ❑ E.g. executables, log files, auto-generated files, intermediate files, etc. created during compile time
- ❑ You will specify those in `.gitignore`
  - ❑ In your project folder, create a new file `.gitignore` (if it doesn't exist)
    - ❑ `touch .gitignore`
    - ❑ `gedit .gitignore`  
Type `*.log` in `.gitignore` file.
    - ❑ `save`
  - ❑ This will teach git not to add any file ending with `.log`.
  - ❑ You can do this for a lot of other type of files

# Working with branches



The screenshot shows a GitHub repository page for 'git-tutorial'. The repository is owned by 'git-tutorial' (Project ID: 3601) and has 1 commit, 1 branch, 0 tags, and 0 bytes of files. A dropdown menu is open, showing options like 'This directory', 'New file', 'Upload file', 'New directory', 'This repository', 'New branch', and 'New tag'. The 'New branch' option is highlighted. Below the dropdown, there is a 'Name' input field and a 'README.md' file listed with an 'Initial commit' label.

- ❑ Branches are useful when we want to create different versions:
  - ❑ E.g version 1.01, beta version, etc.
- ❑ You can create a new branch from existing branch
- ❑ To clone a specific branch:
  - ❑ `git clone -b <name of branch> <repo address>`
- ❑ To switch a branch:
  - ❑ `git checkout <name of branch>`

# Reverting changes

- ❑ There will be a point where you find out that you didn't like changes you made to a file for some reason and you want to revert back to what is there in git repo on server
  - ❑ **git checkout** <name of file>
- ❑ Be careful with **git checkout** as once you execute this, you will lose any changes you made to that file.

# Deleting a file from a git

- ❑ `git rm <name of the file>`
- ❑ This will delete the file locally as well as from git.
- ❑ You are yet to commit and push this deletion request to the server
- ❑ If you change your mind before committing:
  - ❑ `git checkout <name of the file>`

# Retrieve a particular version of a file

- ❑ `git checkout c27bod33 <fileName>`
- ❑ Here `c27bod33` is the history number [or revision number], it can be anything for you. Check history to know which version you want to checkout





# That's it folks.

You will learn more as you move forward with the project