INTRODUCTION TO

GITHUB AND GIT

WHAT IS GIT AND GITHUB?

- Is it the same as Github?
- A version control system used for collaborating and managing projects and work.
- What is version control?
- How does Github come into picture? It helps host Git repositories. Other example is Atlassian’s Bitbucket
- Git - Terminal and Github - Browser

WHY IS VERSION CONTROL USED SO EXTENSIVELY?

- It is used in large organisations and startups.
- Easy to manage multiple collaborators across multiple projects
- Keeps track of revisions and code changes
- One question - Why to use git when there is Github?
- Initial easy functions can be done in Github, but as we work extensively across multiple projects Git commands are easier.

LETS GET STARTED

- Create a Github account
- Setup Git on your terminal
- Command - **git**
- Create a repository
- Initialise with a README

CLONING A REPOSITORY

- You download a local copy of your online repository
- Command - ***git clone your_rep_name***

ADD FILES

- Command -
  - ***echo “Hello 1” > hello1.txt***
  - ***echo “Hello 2” > hello2.txt***
- Checking the status of repository.
- Command -
  - ***git status***

ADD FILES AND COMMIT

- Command -
  - ***git commit***
- Command -
  - ***git add -A***
  - ***git status***
  - ***git commit -m "Add new hello text files"***
- Always give meaningful commit names and multiple commits.

LOOKING AT YOUR COMMIT HISTORY

- Command -
    - ***echo "Hello 3" > hello3.txt***
    - ***git add -A***
    - ***git commit -m "Add hello3 text file"***
- Command -
    - ***git log***
    - ***git log -p -2***

THE MOST IMPORTANT STEP - PUSHING THE CHANGES

- Pushing - from Git to Github
- Command -
    - ***git status***
    - ***git push***
- All files are added to your online Github repo.

THE CONCEPT OF HEAD

- Github tracks your changes using a variable called HEAD
- It is a pointer which points to your current version of files/project
- All files are accessed with respect to HEAD

CLONING SOMEONE ELSE'S REPOSITORY

- Most frequent usage of GitHub - cloning a project.
- Command -
    - ***git clone https://github.com/aditya1702/github_workshop***
- Multiple collaborators = multiple versions of code = increased complexity

COMMIT YOUR CHANGES

- Command -
    - ***echo "Hello your_name" > hello_your_name.txt***
- Commit this file and push it using the previous commands.
- Were you successful?
- And this leads us to the concept of branches….

BRANCHES – THE MAGIC OF GIT

- Branches allow multiple collaborators to work on a single project.
- Command -
    - ***git checkout -b branch_your_name***
    - ***git branch***
    - ***git push***
- Were you able to push your changes?

THE CONCEPT OF REMOTES

- Your repository has 2 copies - a local copy and a remote copy on Github.
- Your default remote repository name is ***origin***
- You can have multiple remote repositories with different names and pull from them.
- And the master branch of your remote is referred to as ***origin/master***

SET UP REMOTE TRACKING AND PUSH CHANGES

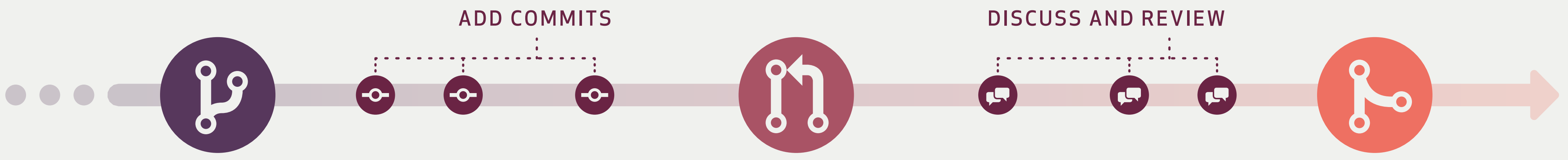
- Your local Git has to know which remote branch is it tracking.
- Command -
    - ***git push --set-upstream origin branch_your_name***
- Now you can push changes easily.
- Create and push a new file *hello_your_name_again.txt*

SUBMIT A PULL REQUEST

- Go to GitHub and on your branches.
- Assign reviewers
- View the diff.
- Give a meaningful pull request name and submit.
- It is reviewed by the code reviewers and then merged into master.

WORK FAST
WORK SMART
THE GITHUB FLOW

The GitHub Flow is a lightweight, branch-based workflow that's great for teams and projects with regular deployments. Find this and other guides at **http://guides.github.com/**.



CREATE A BRANCH

Create a branch in your project where you can safely experiment and make changes.

OPEN A PULL REQUEST

Use a pull request to get feedback on your changes from people down the hall or ten time zones away.

MERGE AND DEPLOY

Merge your changes into your master branch and deploy your code.

GitHub is the best way to build software together.

GitHub provides tools for easier collaboration and code sharing from any device. Start collaborating with millions of developers today!

PULLING CHANGES AND MERGE CONFLICTS

- You keep your branch up-to date with the master branch
- Command -
    - ***git pull origin master***
- However, this can lead to problems when same files are changed in master and in your branch
- Create a file conflict.txt.
- And now pull from master.

STASHING CHANGES

- Sometimes you make lots of changes and then want to commit.
- But you want to merge in new changes from master and then continue on.
- You stash your changes in a temporary stack
- Command -
    - ***git stash***

RESET CHANGES

- Sometimes you have to make changes midway. You can reset your commits
- Command -
  - ***echo “Trying out reset” > reset1.txt***
  - ***echo “Trying out reset again” > reset2.txt***
  - ***git add -A***
  - ***git reset***
- And sometimes you can do a hard reset
  - ***git reset –hard***

REBASING AND SQUASHING

- Used to combine multiple commits into a single commit.
- Command -
  - ***git rebase -i HEAD~n***
- Change first commit to ***r*** and subsequent commits to ***s.***
- In the second editor, change the commit name. Type Esc, : and w, q

CHERRY PICKING

- Cherry picking allows you to apply particular commits to your branch rather than the whole set of changes.
- Command -
    - ***git cherry-pick commit_hash***
- Add only some commits to your branch