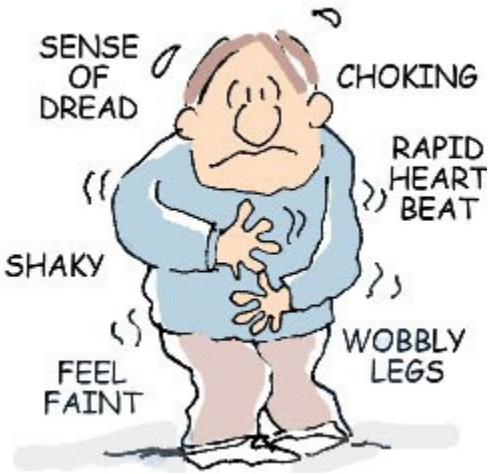
Not avoidable?

5:30pm



Too heavy for us, we can't take it further.

1:00pm

Doesn't feel like food!

Oh no! A few hours left.

1:00pm – 2:00pm


So what
I do?


Collaboration
Dining table.

Why
Care?

Human Resource Team

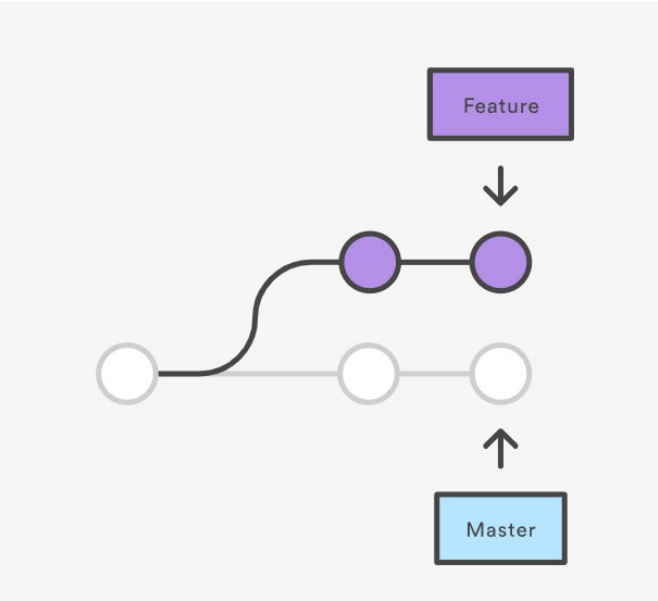
The workflow of an organization determines the kind of personnel(employees) available to hire.

The same is about the talented people in the marketplace, they are most likely to accept offers from organizations where personal growth is guaranteed.

For developers; Version Control Systems and agile principals are one of the major interests.

Most brands choose and recommend a Git to drive collaboration. Git is free regardless of the level of organization, it is secure, easy to learn and has many inexhaustible features; some of which we shall see and master today.

Developer Team



99% of developers who involve in collaboration use version control systems to live an efficient production lifestyle.

A notable feature about VCS is branching.

Branching enables developers to maintain isolated copies of the same code technically known as branch(es).

Usually the branches are Master(Production), Feature A, Feature B etc...

As developer A, works on branch labelled Feature A, it doesn’t affect the Master branch or Feature B branch.

Once developers complete and are happy with what they’ve done, they will send a signal to the production team about their progress. If the production team likes, they will pull the branch and merge it in the master branch. The marketing team will always be kept in sync to make regular announcements to the market.

Since the developers maintain their own copy of the code, they can actually seek support from other developers to give them support. So collaboration with Version Control Systems is one way help others and be helped.

Product Management Team

The benefits of Git for product management is much the same as for marketing. More frequent releases means more frequent customer feedback and faster updates in reaction to that feedback. Instead of waiting for the next release 8 weeks from now, you can push a solution out to customers as quickly as your developers can write the code.

The feature branch workflow also provides flexibility when priorities change. For instance, if you’re halfway through a release cycle and you want to postpone one feature in lieu of another time-critical one, it’s no problem. That initial feature can sit around in its own branch until engineering has time to come back to it.
This same functionality makes it easy to manage innovation projects, beta tests, and rapid prototypes as independent codebases.

Who else can benefit from Version Control Systems to collaborate are?

The marketing.

The designer.

Customer support

Anyone managing a budget product.

Some practice? The vocabulary

1. **Repository:** A repository is a store for track of changes about files of a project. They are two types of repositories (Local on your computer and Remote on internet servers).
2. **Stage:** This is a location where files prepared before they are finally committed.
3. **Commit:** The "commit” is an action is used to save your **changes** to the local repository.
4. **Contributor:** Someone who works together with you on the code
5. **Fork:** A fork is a copy of a project.
   Forking a repository allows you to make changes without affecting the original project.
6. **Pull:** Notify leader to take a look at your contribution and make a decision.
7. **Push:** Upload your local code to the an online codebase(repository).
8. **Remote:** Refers to a repository which is located somewhere else?
9. **Branch:** In Git, the repository refers to your entire project. Within a single Git repository, you have at least one branch. You can use git branch newFeature to create a new branch within your repository to track your **changes** to the **changes** to your code base that pertain to a particular new feature.

1. **Tags:** Tags give the ability to mark specific points in history as being important.
2. **Issues:** Issues can be bugs, tasks or ideas to be discussed. The Issue Tracker is the place to add things that need to be improved or solved in a project

12. **Merge:** Incorporates changes from the named commits (since the time their histories diverged from the current branch) into the current branch. This command is used by **git**pull to incorporate changes from another repository and can be used by hand to **merge** changes from one branch into another.

13. **Fetch:** In the simplest terms, **git** pull does a **git fetch** followed by a**git** merge . You can do a **git fetch** at any time to update your remote-tracking branches under refs/remotes/<remote>/ . This operation never changes any of your own local branches under refs/heads , and is safe to do without changing your working copy.

14. Status

Introduction to Git

WHAT IS Git?

- Distributed (Decentralized) version control system.

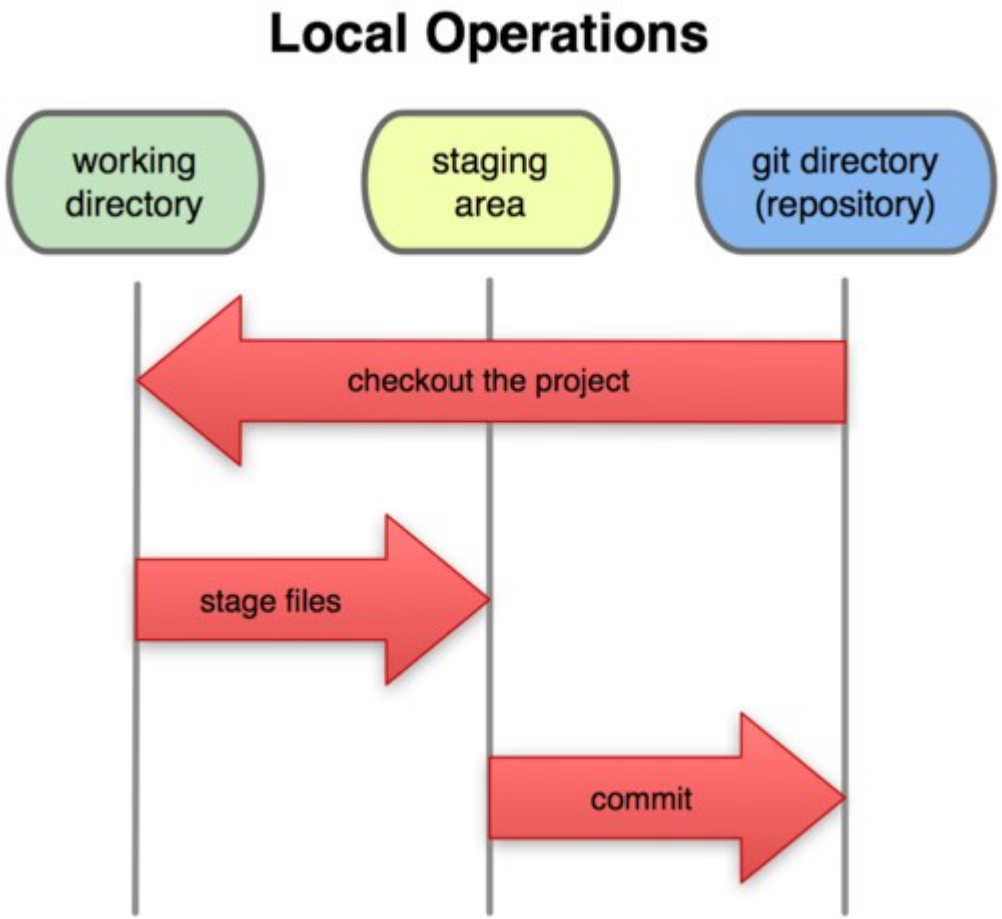
GIT DESIGN GOALS

- Speed
- Simple design
- Strong support for thousands of parallel branches
- Fully distributed
- Able to handle larges projects like Linux kernel effectively

GIT DOESN'T DELETE

Git generally only adds data If you mess up, you can usually recover your stuff.
Recovery can be tricky though

WorkFlow


Local Operations
working directory
staging area
git directory (repository)
checkout the project
stage files
commit

Getting started

Download and install Git on your on workstation. Git can be installed on Linux, MacOS and Windows computers.

Download Git for your OS from the web link below

https://git-scm.com/downloads

Git Commands

IDENTITY
$git config --global user.name "John Doe"
$git config --global user.email johndoe@example.com

CREATING A REPOSITORY
$git init

CLONING A REPOSITORY
$git clone https://github.com/user/project.git

CONNECT LOCAL TO REMOTE
$git remote add origin https://github.com/user/project.git

ADD NEW FILE
$git add README.rst

REMOVE FILE
$git rm file.py

COMMIT CHANGES
$git commit -am 'First commit’

SHOW LOG
$git log

SHOW COMMITS
$git show

SHOW DIFFS
$git diff

UNDOING THINGS

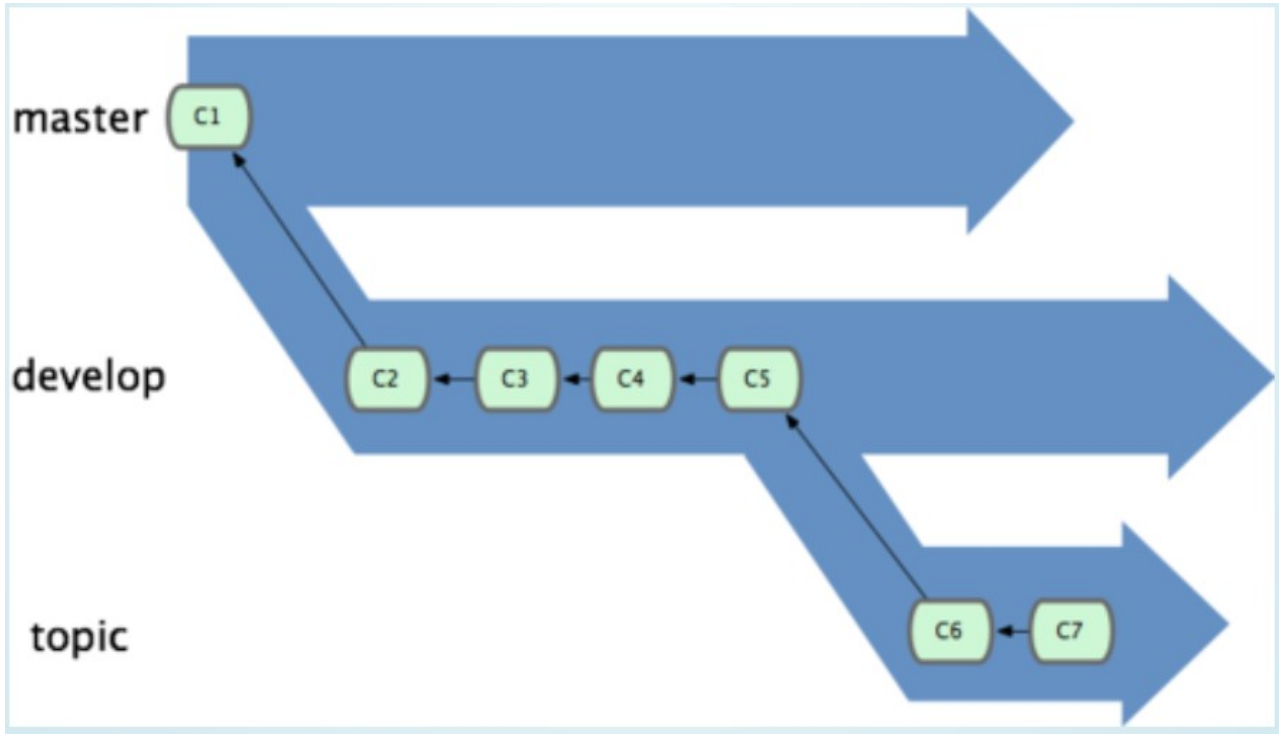
UNMODIFY MODIFIED FILE

$git checkout -- file.py

REVERT A COMMIT

$git revert 1776f5

BRANCH MANAGEMENT



Create new Branch
$git branch iss53
$git checkout -b iss53
master

Switch Branch
$git checkout iss53

Delete Branch
$git branch -d iss53

Show All Branch
$git branch -d iss53

Repositories

Remote Commands

Push To remote

$git push origin
branch_name

Fetch & Merge

$git pull

Fetch
$git fetch origin

Merge
$git merge origin
master

Merge conflicts

$git merge iss53 Auto-merging index.html CONFLICT (content): Merge conflict in index.html Automatic merge failed; fix conflicts and then commit the result.

Home work

Study about the different Git workflows such as feature branch, gitflow etc.

Resources

- http://slides.com/moji3000/git_presentation/fullscreen#/
- https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud
- Several YouTube Tutorials