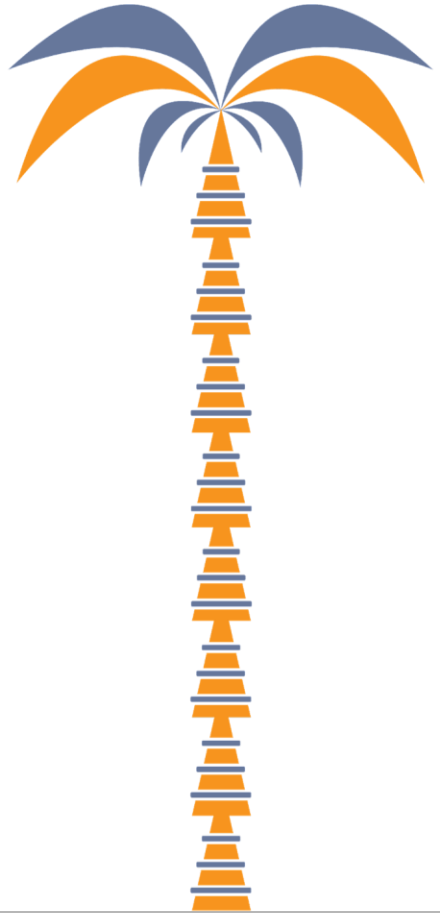


DevOps Bootcamp

Version Control with Git - Lecture #2

Dan Morgenstern

danm@sela.co.il

22/05/2022

(Please write "I am here (your name)" in the zoom chat to register your attendance)

Agenda

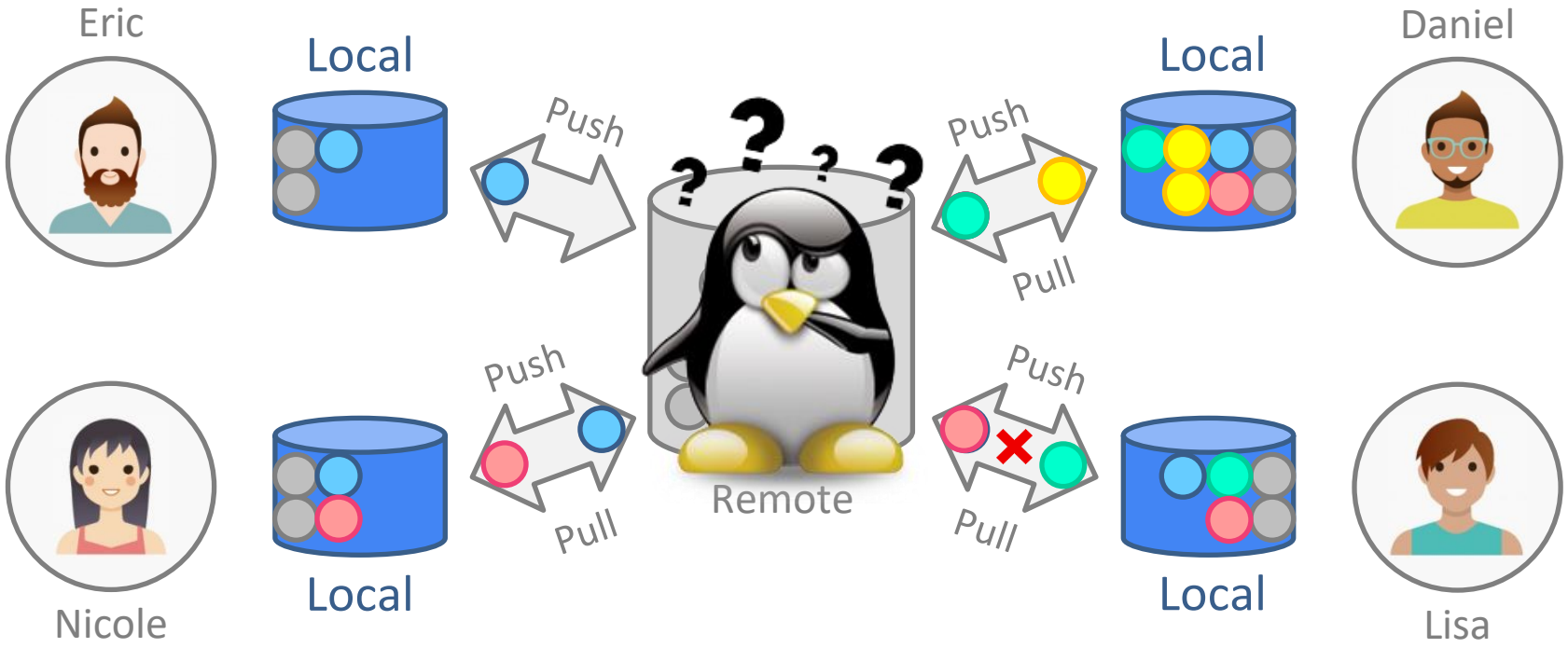
- Bootcamp News
- Git Introduction Recap
- Working with Remotes
- Git Workflows

Git Introduction Recap

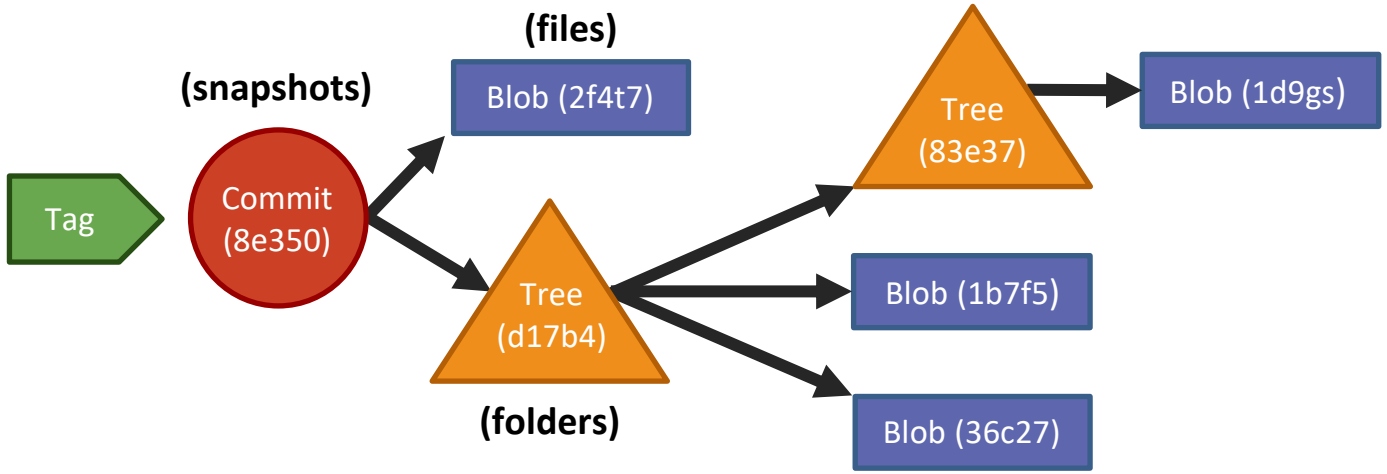
Introduction Summary

Git is a free and open source distributed version control system designed with performance, security and flexibility in mind

Git – Distributed but Centralized

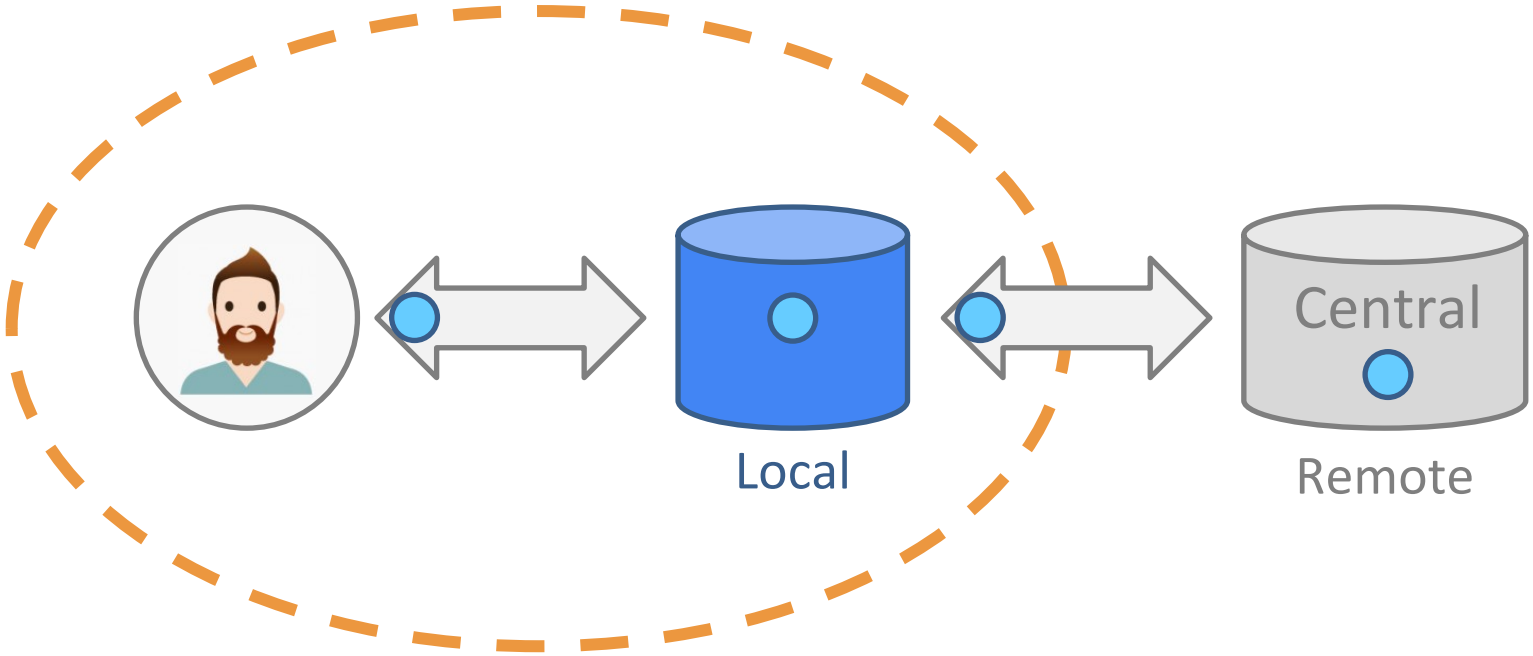


Git Structure - Objects

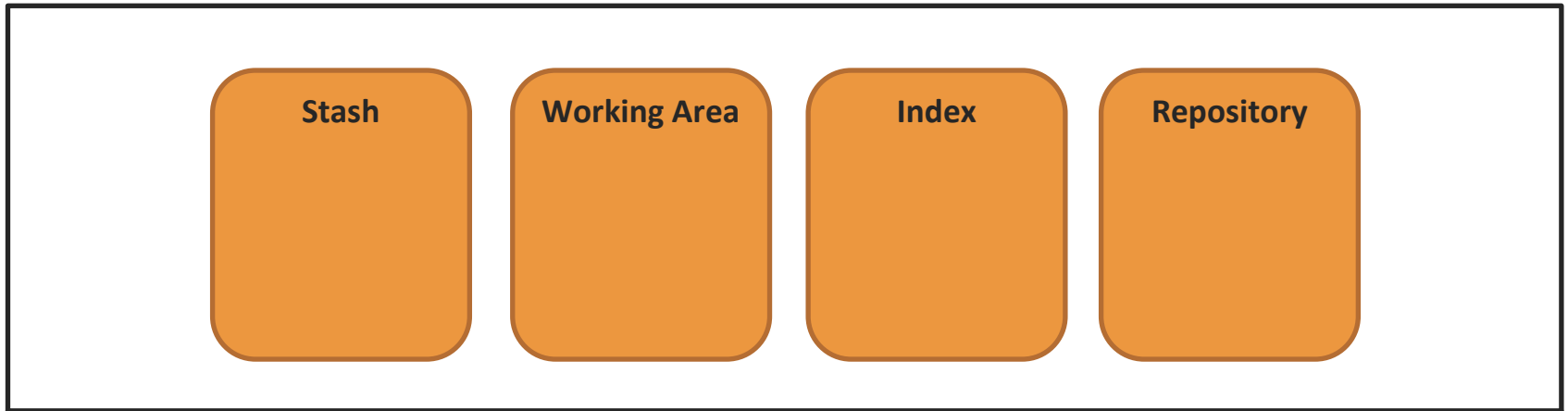


Commit = Snapshot at some point in time | **Tag** = Reference to a commit

Summary

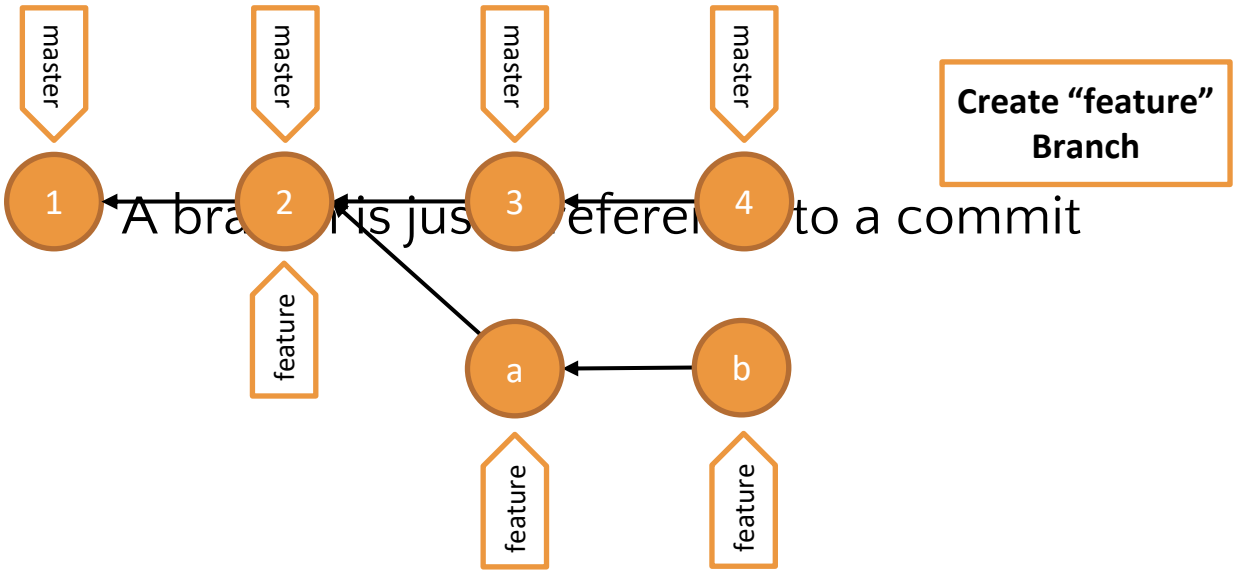


Working Locally (`git init`)



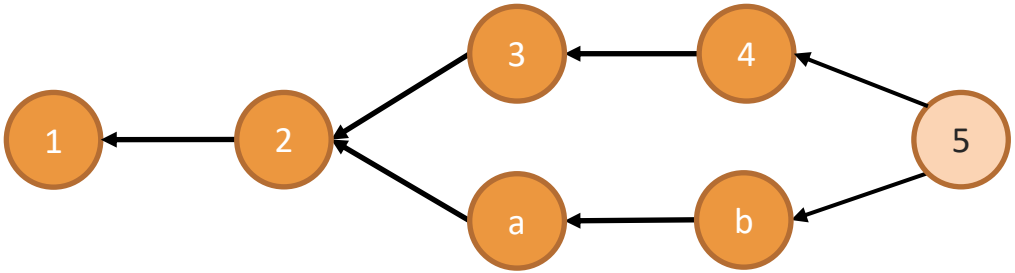

```
$ git init

Initialized empty Git repository in
C:/Users/leonj/Desktop/MyRepo/.git/
```

Working Locally – What Branches really are?



Working Locally - Merging Simplified

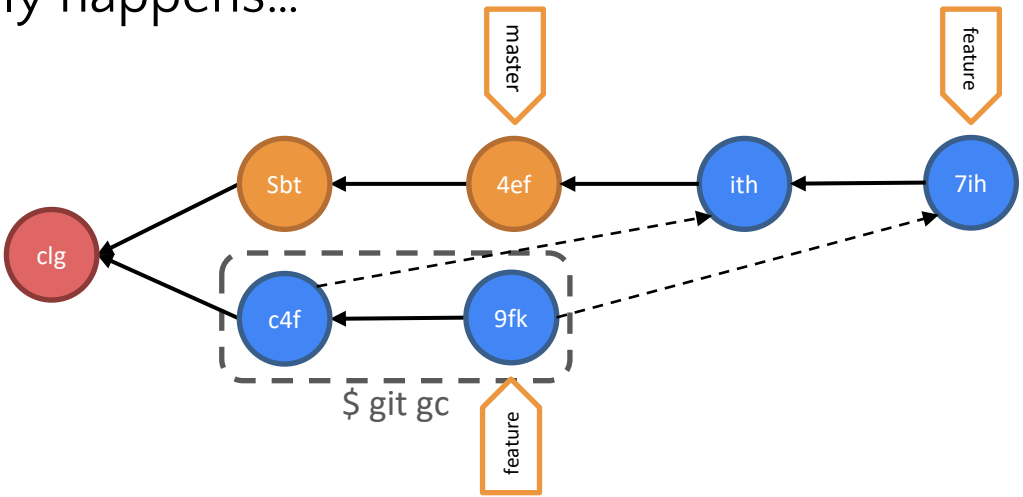
- Merge is just a commit with two parents



- Merge command says: “merge the branch X into the current branch”

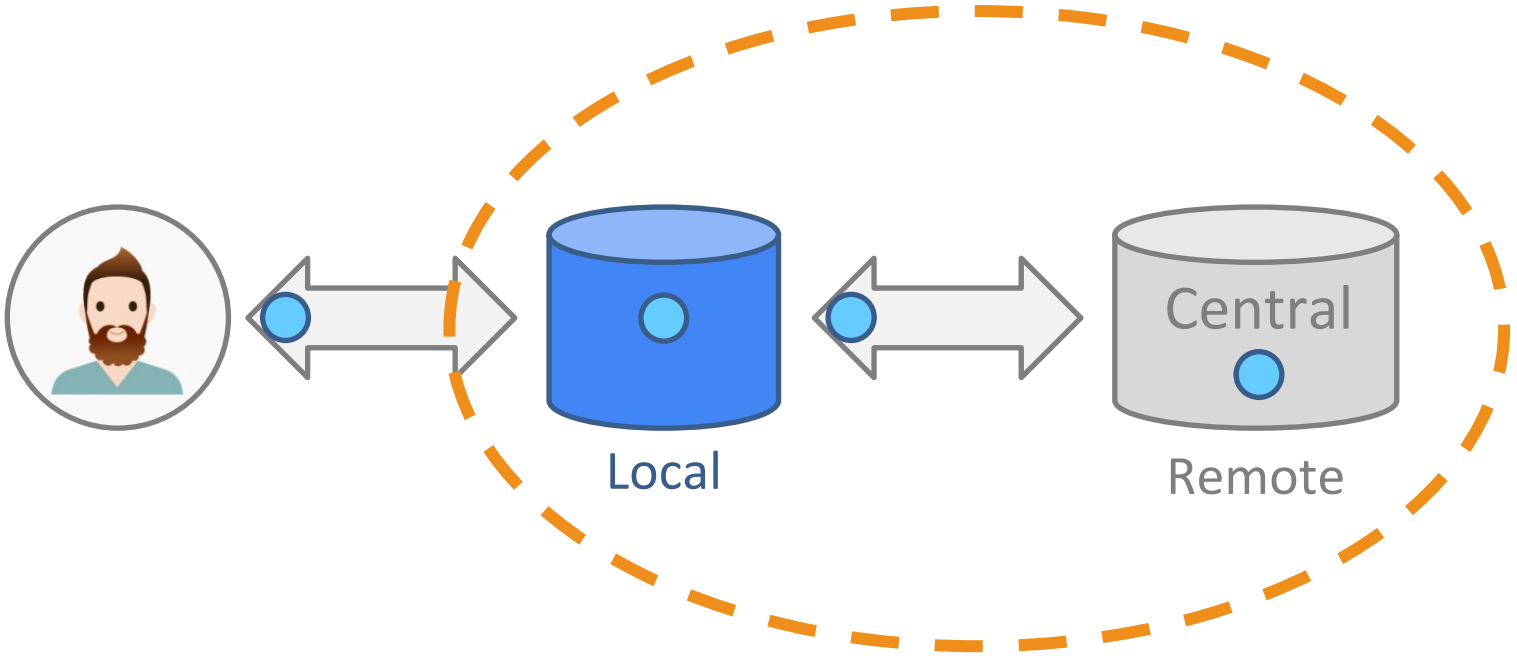
Working Locally – Rebasing Explained

- What really happens…



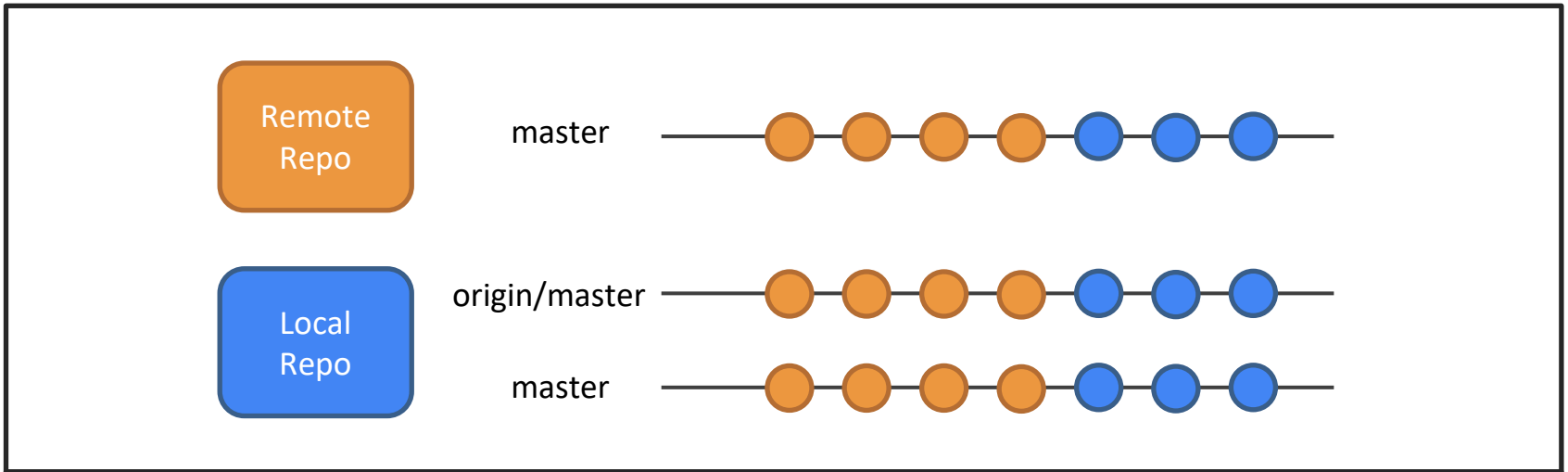
- Rebase command says: “rebase the current branch into the branch X”

Summary



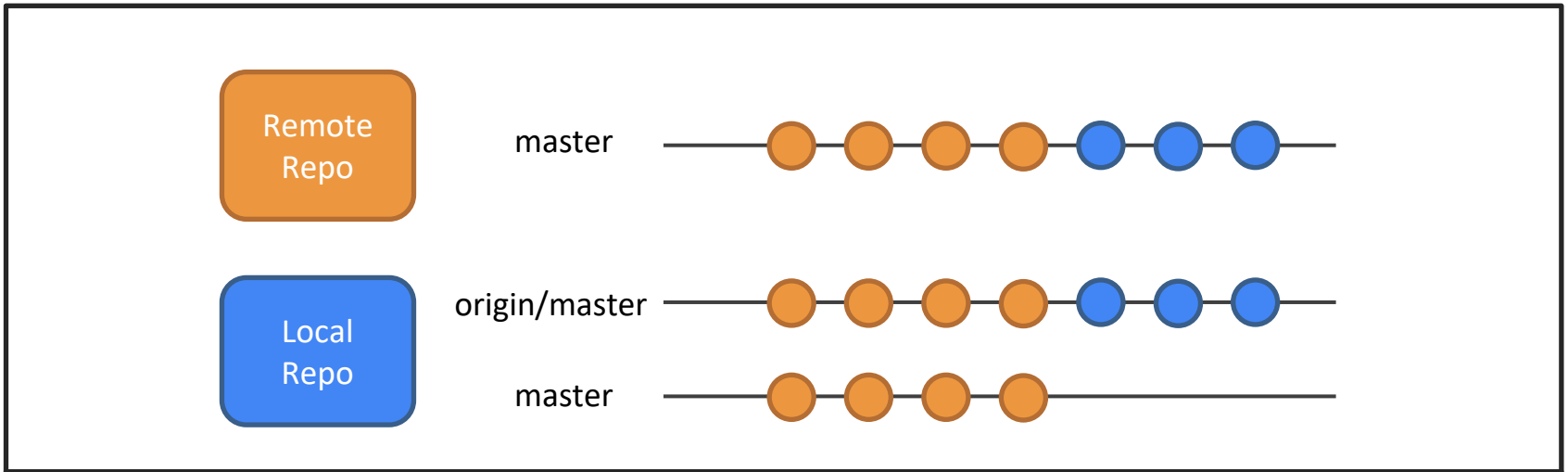
Working with Remotes

Working Locally (`git push`)




```
$ git push origin master
   Counting objects: 3, done.
   Writing objects: 100% (3/3), 243 bytes | 0 bytes/s, done.
   52938bc..a330c52  master -> master
```

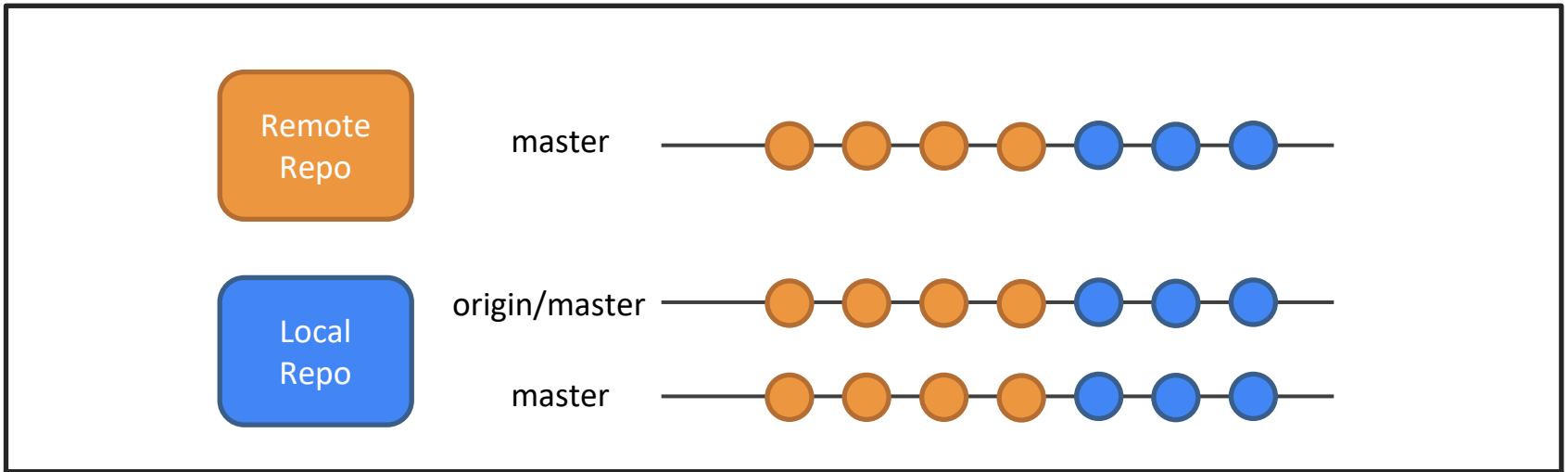
Working Locally (git fetch)




```
$ git fetch origin master
   From https://leonj.visualstudio.com/DefaultCollection/Repo/_git/Test
    * branch            master     -> FETCH_HEAD
      14a2e05..a2e111a  master     -> origin/master
```

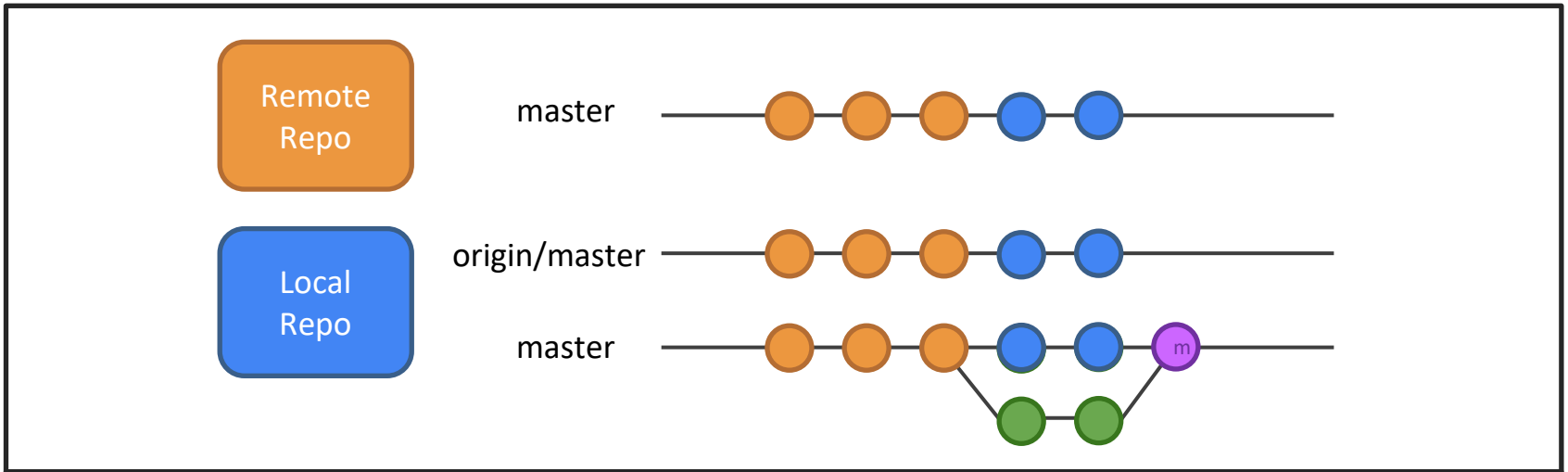
Working Locally (`git pull`)

(fetch + merge)




```
$ git pull origin master
   * branch            master     -> FETCH_HEAD
  Updating 60e69bb..a2e111a
  Fast-forward
```

Working Locally (git pull + conflicts)



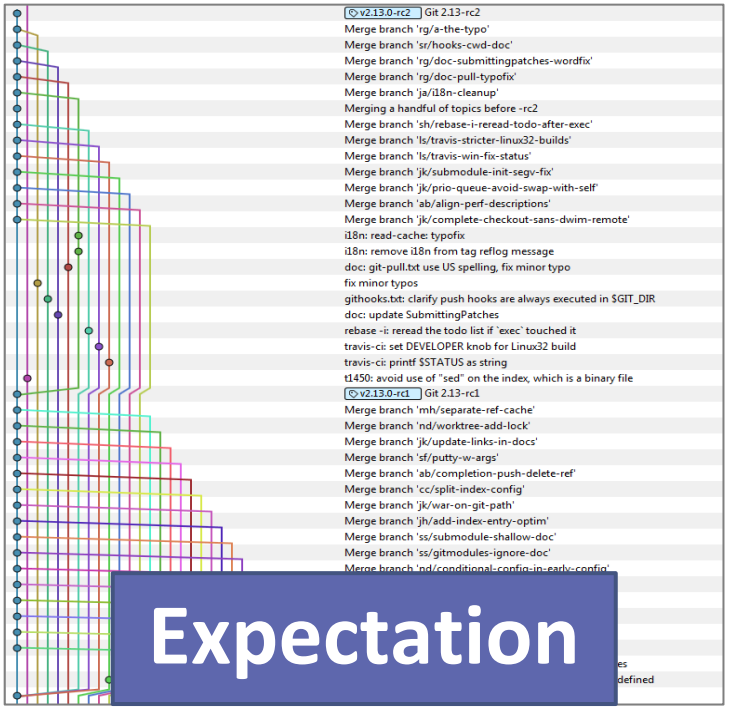

```
$ git pull origin master
   a2e111a..0532902  master     -> origin/master
Auto-merging File.txt
Merge made by the 'recursive' strategy.
```

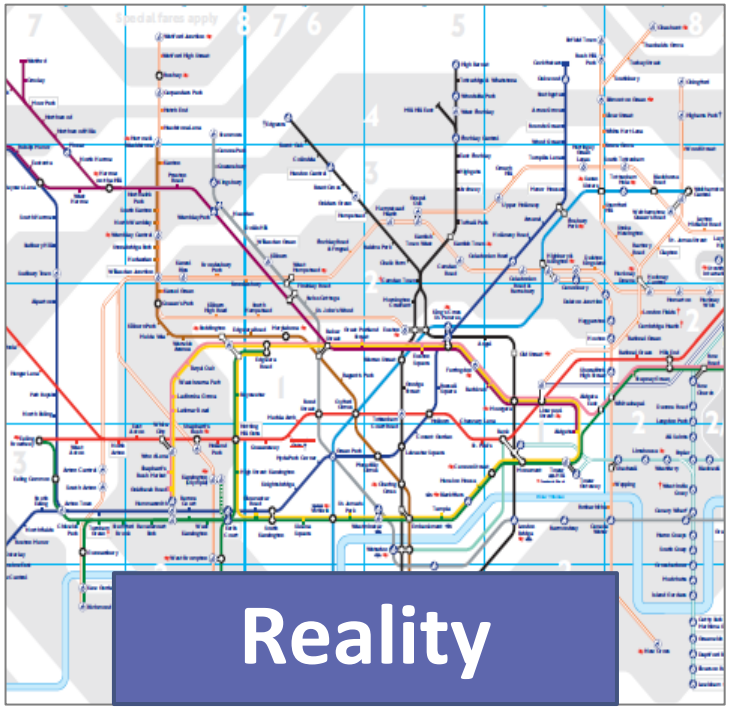
Working Locally (Basics)

Demo

Git Workflows

Introduction – Expectation VS Reality





Git Workflows

- A workflow is a set of conventions that defines how the team should use git, generally composed by:
  - Distribution Model
  - Branches Model
  - Constraints
- The goal of a workflow is to facilitate teamwork and keep the repository clean and organized

Git Workflows

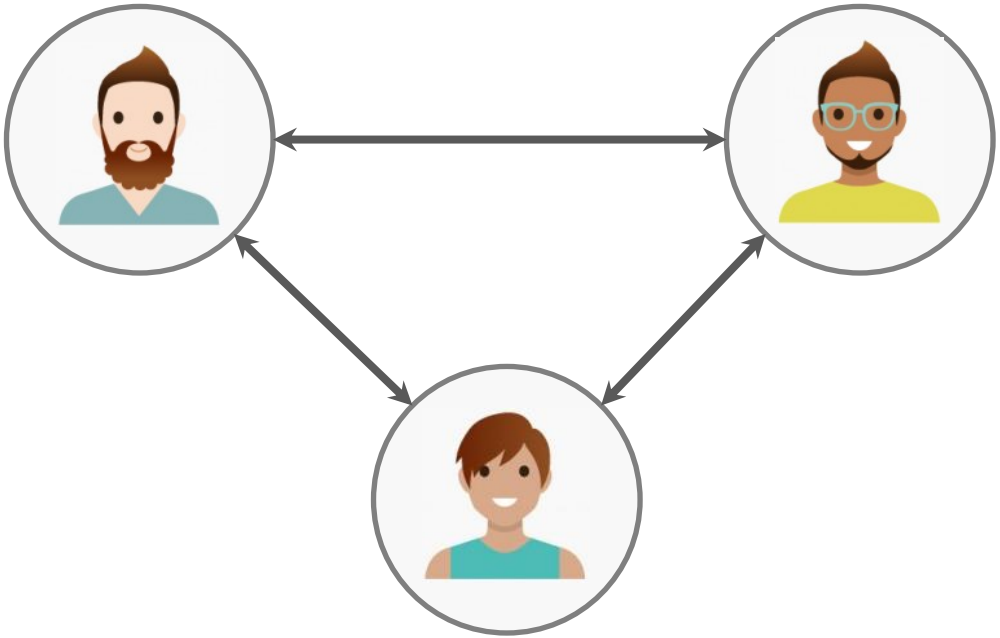
- Let's analyze the pieces that can be used to design our own workflow...

- However decide how to put the pieces together is up to you...

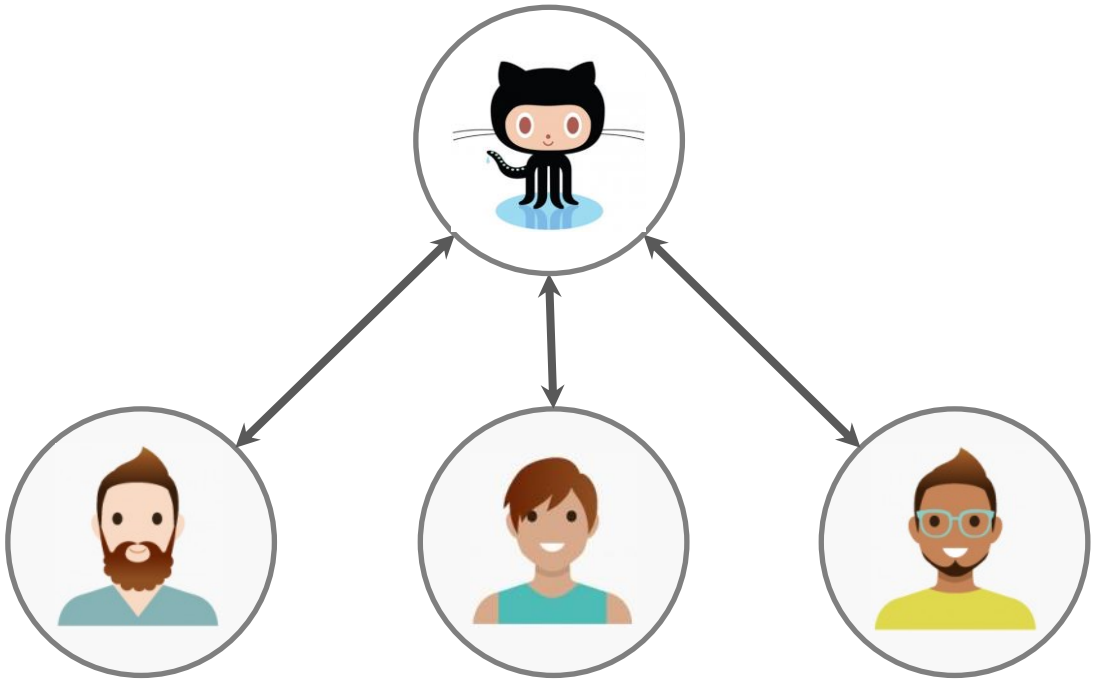
Distribution Models

- How many repositories do you have?
- What will each repository be used for?
- Where will each repository be hosted?
- Who can read/write in each repository?
- Who will manage each repository?

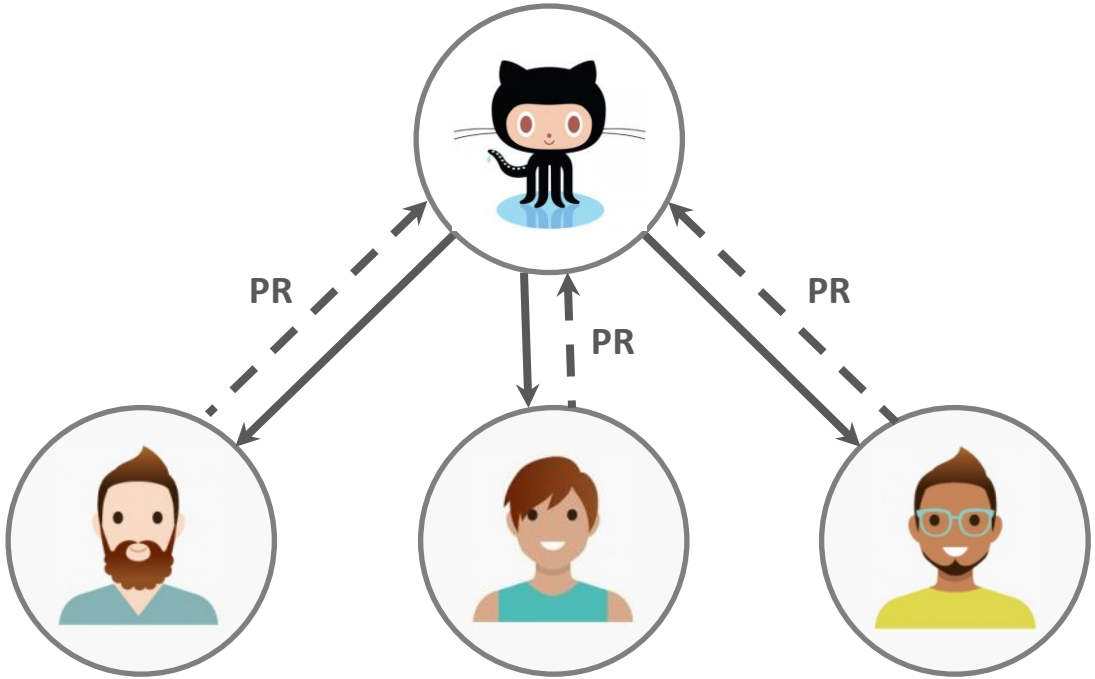
Peer to Peer Model



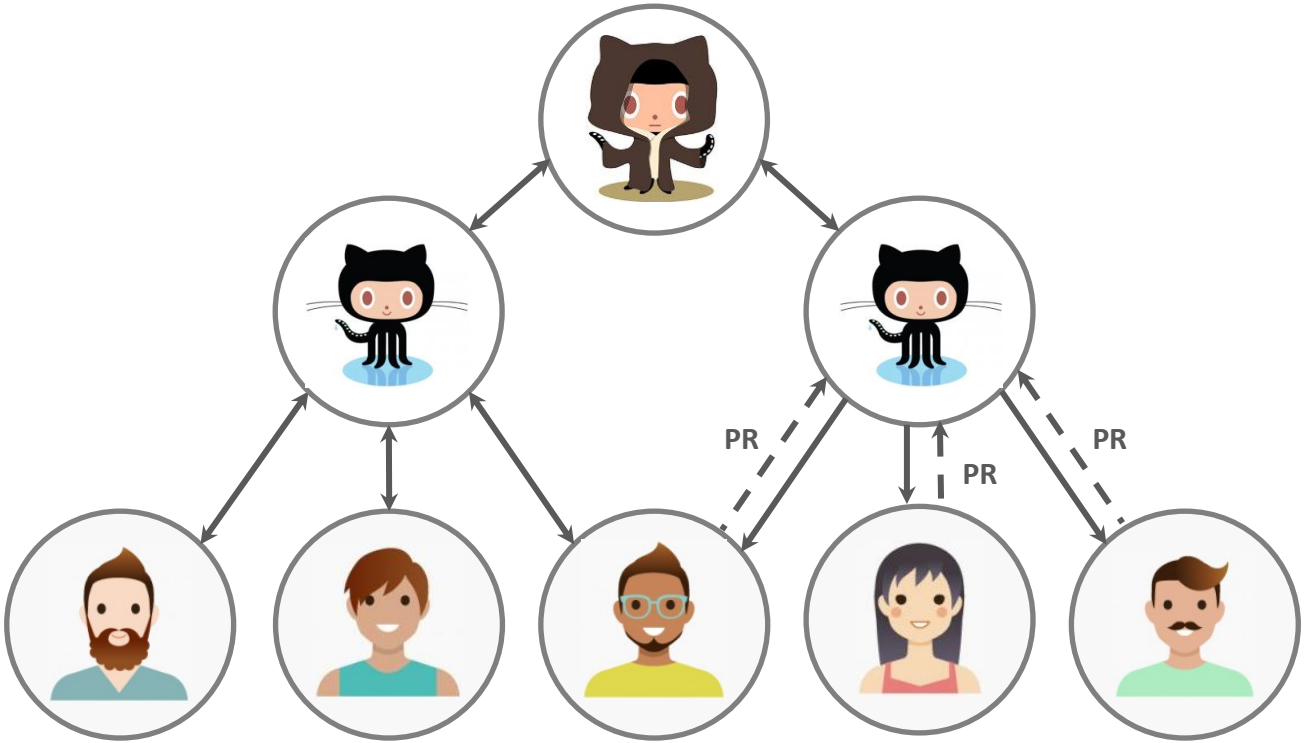
Centralized Model



Pull Request Model



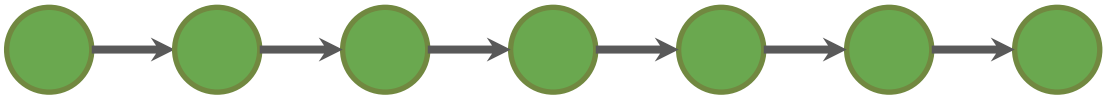
Dictator and Lieutenants Model



Branching Models

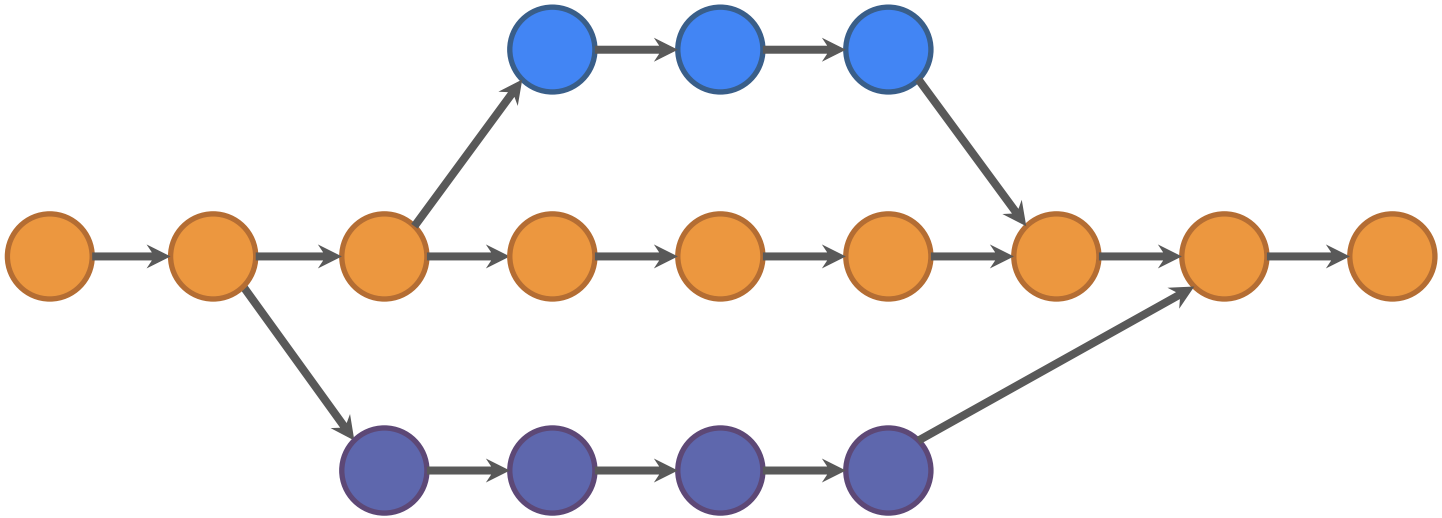
- Which branches do you have?
- How do you use them?
- Who can access each branch?
- Which branches should be merged and when?

Stable Branch

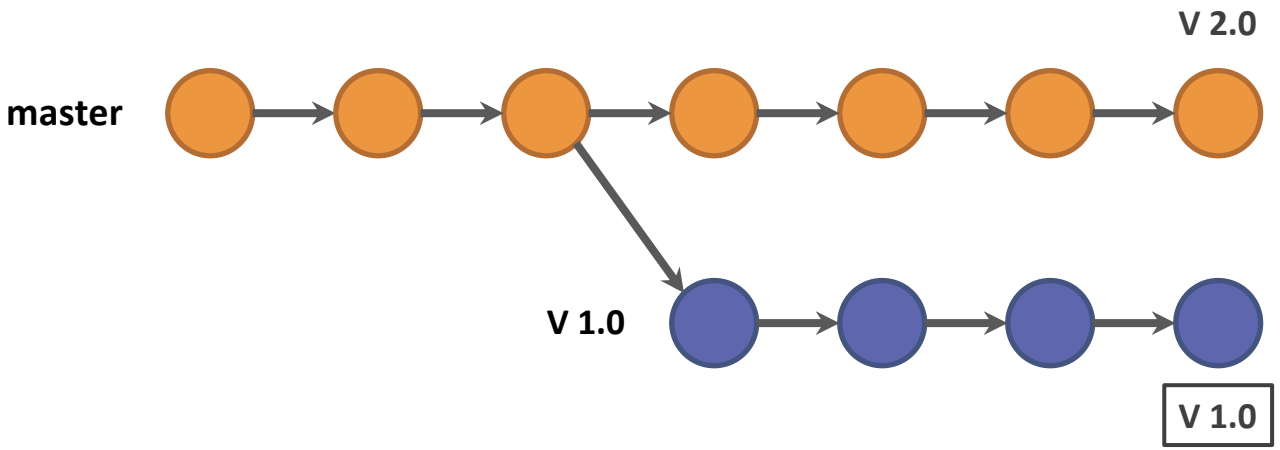


Unstable Branch

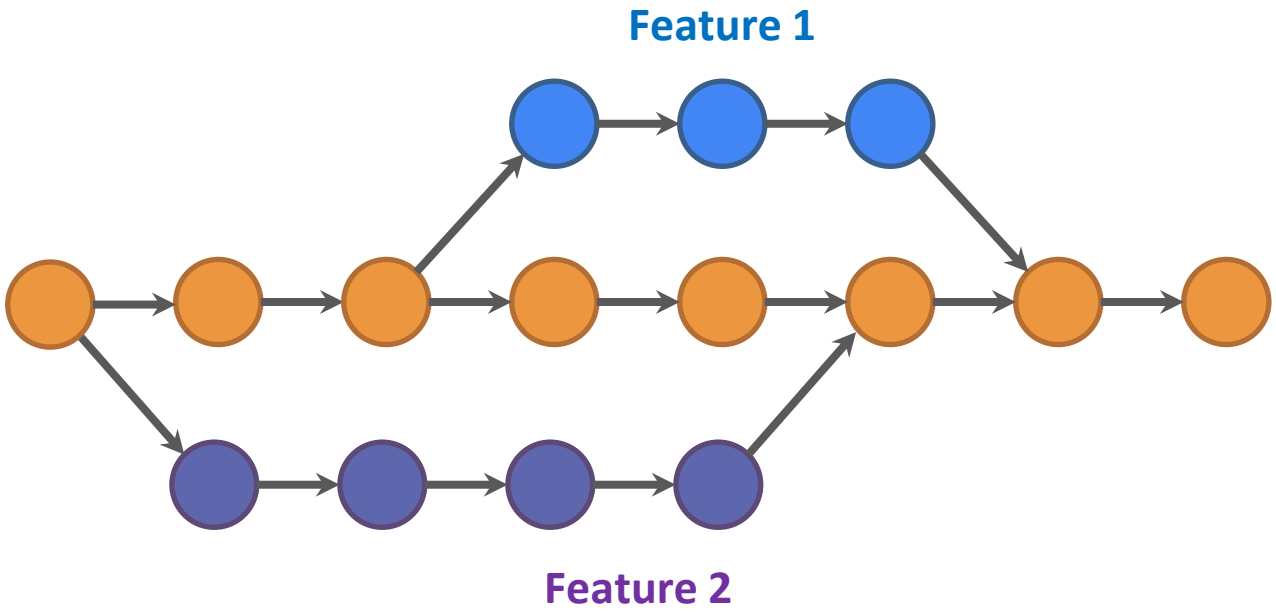
Integration Branch



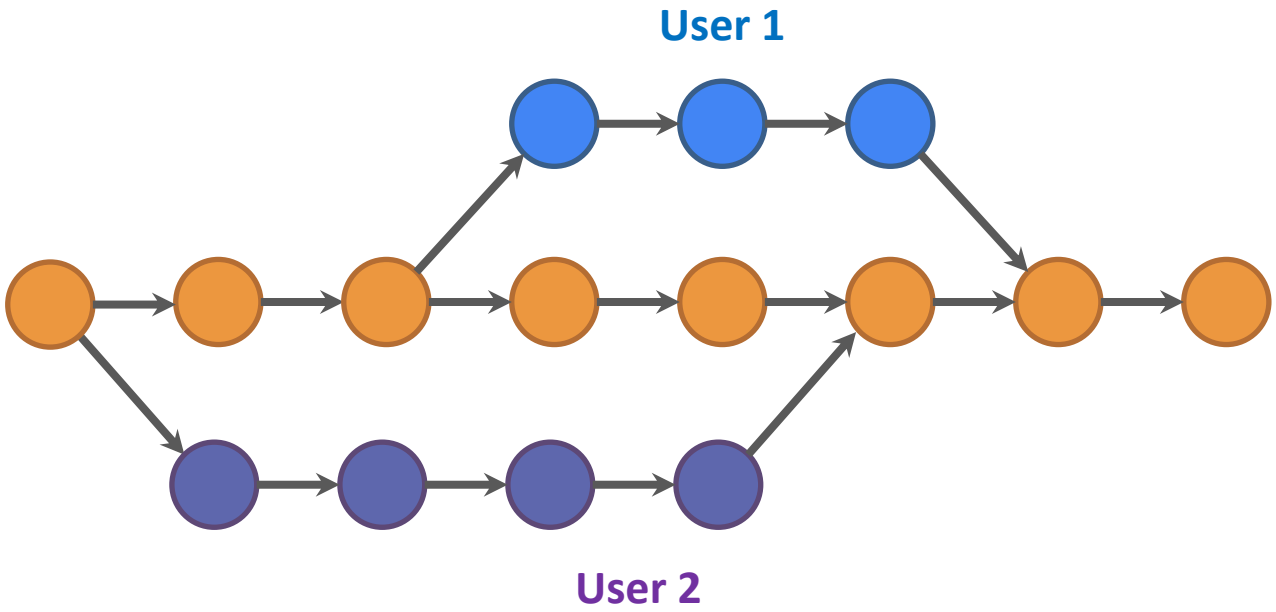
Release Branch



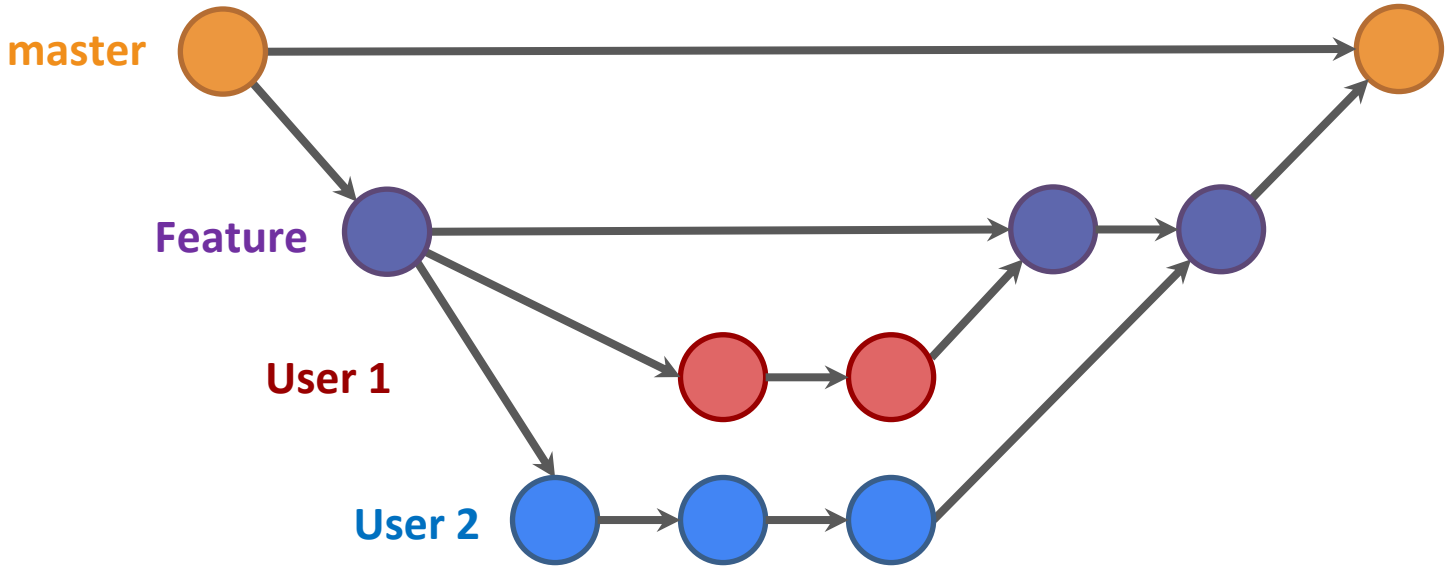
Feature Branch



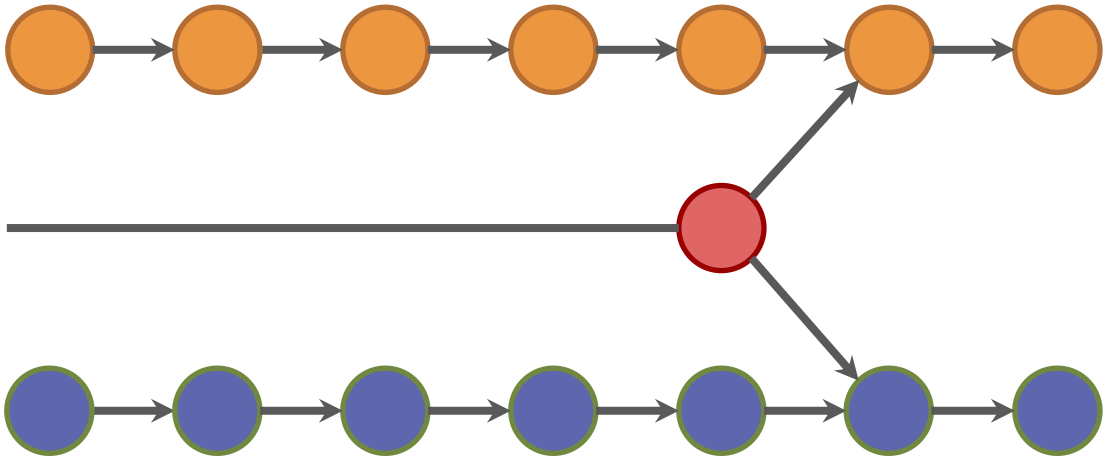
User Branch



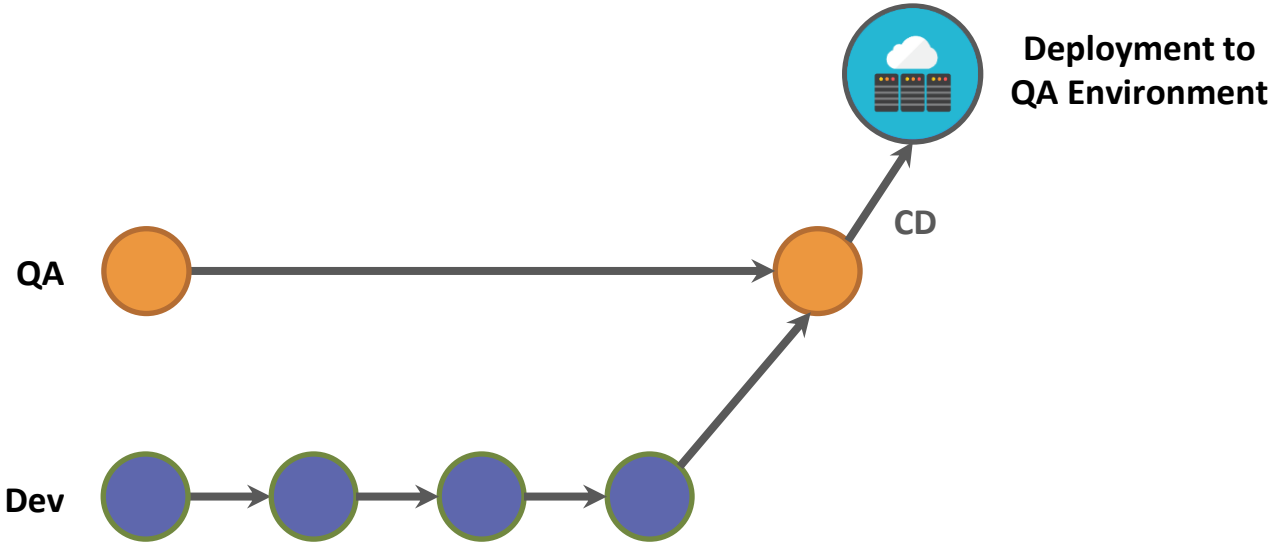
Feature + User Branch



Hotfix Branch



Environment Branch



Constraints

- Do you merge or do you rebase?
- Can you push unstable code? Where yes and where not?
- You will use .gitignore file? For which files?
- Which merge strategy you will use?
- Where and when should tags be used?

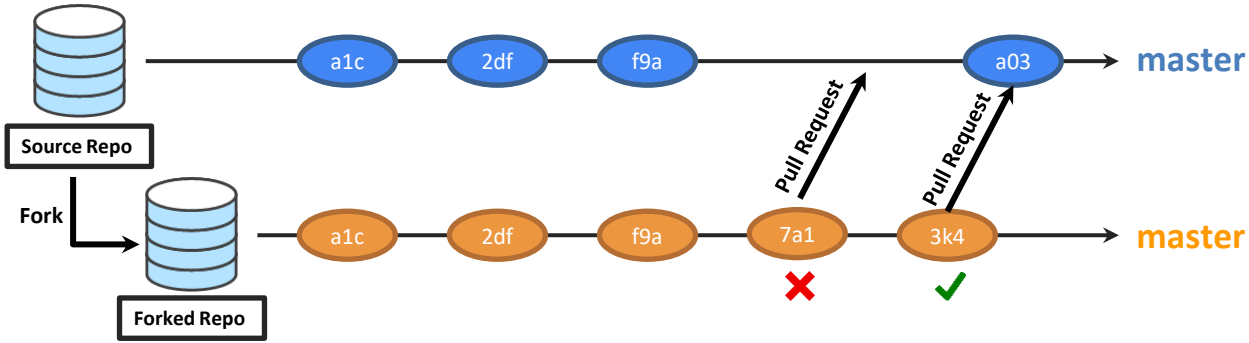
Constraints

- Merge VS Rebase
- Only certain people can do certain things
- Always merge using --no-ff
- Don’t push to unstable branch
- Squash features before merge
- Tag bug fix commits

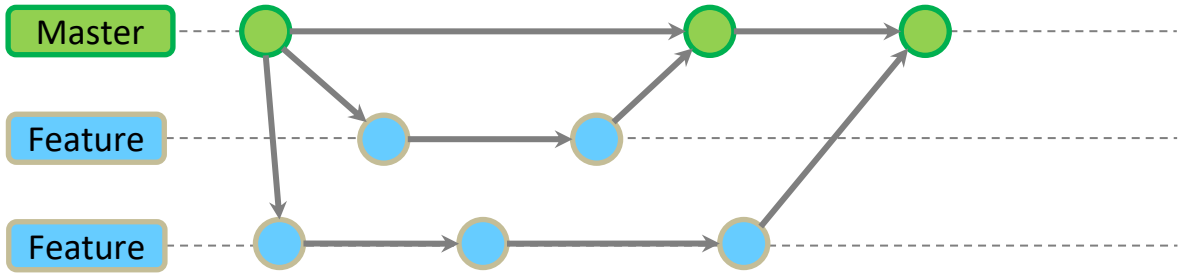
Most Known Workflows

- Forking Workflow
- Feature Branch Workflow
- Environment Workflow
- GitFlow Workflow

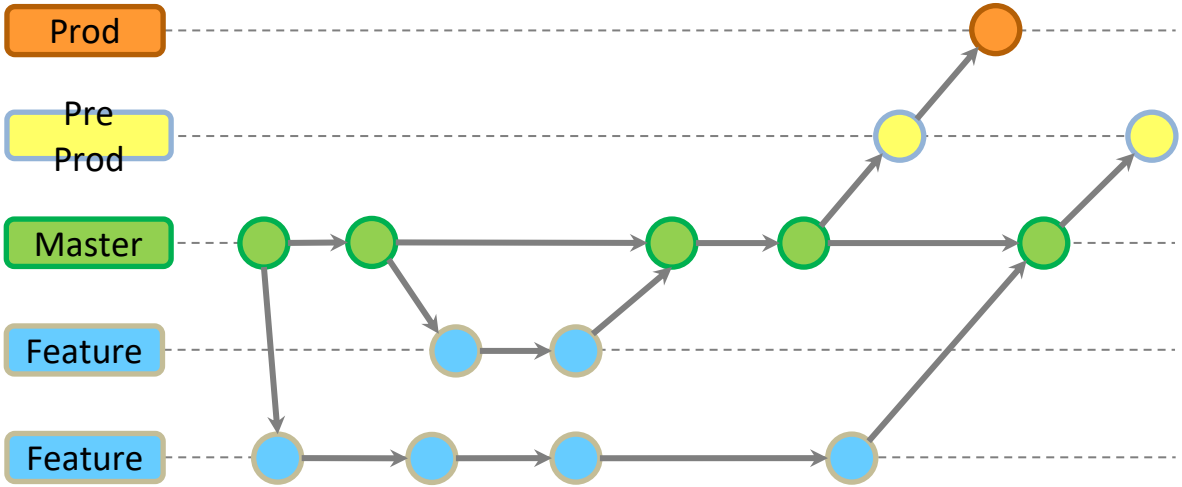
Forking Workflow



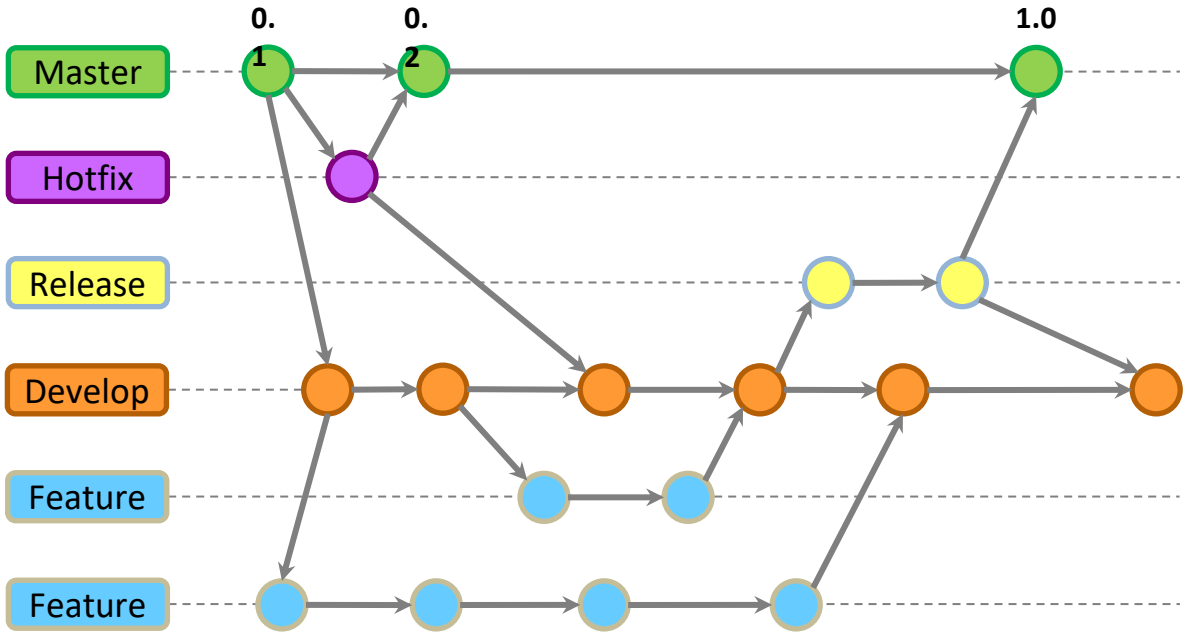
Feature Branch Workflow



Environment Workflow



GitFlow Workflow



Creating a Custom Workflow

- There is no single "correct" workflow
- The most complete workflow is not necessarily the best for you
- The simpler and easy workflow the better
- Don’t look for a workflow that suits your need, create it
- Take into consideration that your needs may change over the time
- Usually the workflow also changes in order to fit your needs

Summary

A Git workflow is the methodology that define the distribution model, the branching model and the constraints for a Git project.

Don’t design a complex workflow

Instead, grow it...

Working with Branches

Demo

Questions

DevOps Bootcamp

Version Control with Git - Lecture #2

Natali Cutic
natalic@sela.co.il

Noam Amrani
noama@sela.co.il

Pushing YOU forward