/kahootali

DevOps-Week 2-Github

M. Ali Kahoot

DevOps Course By M. Ali Kahoot - Dice Analytics

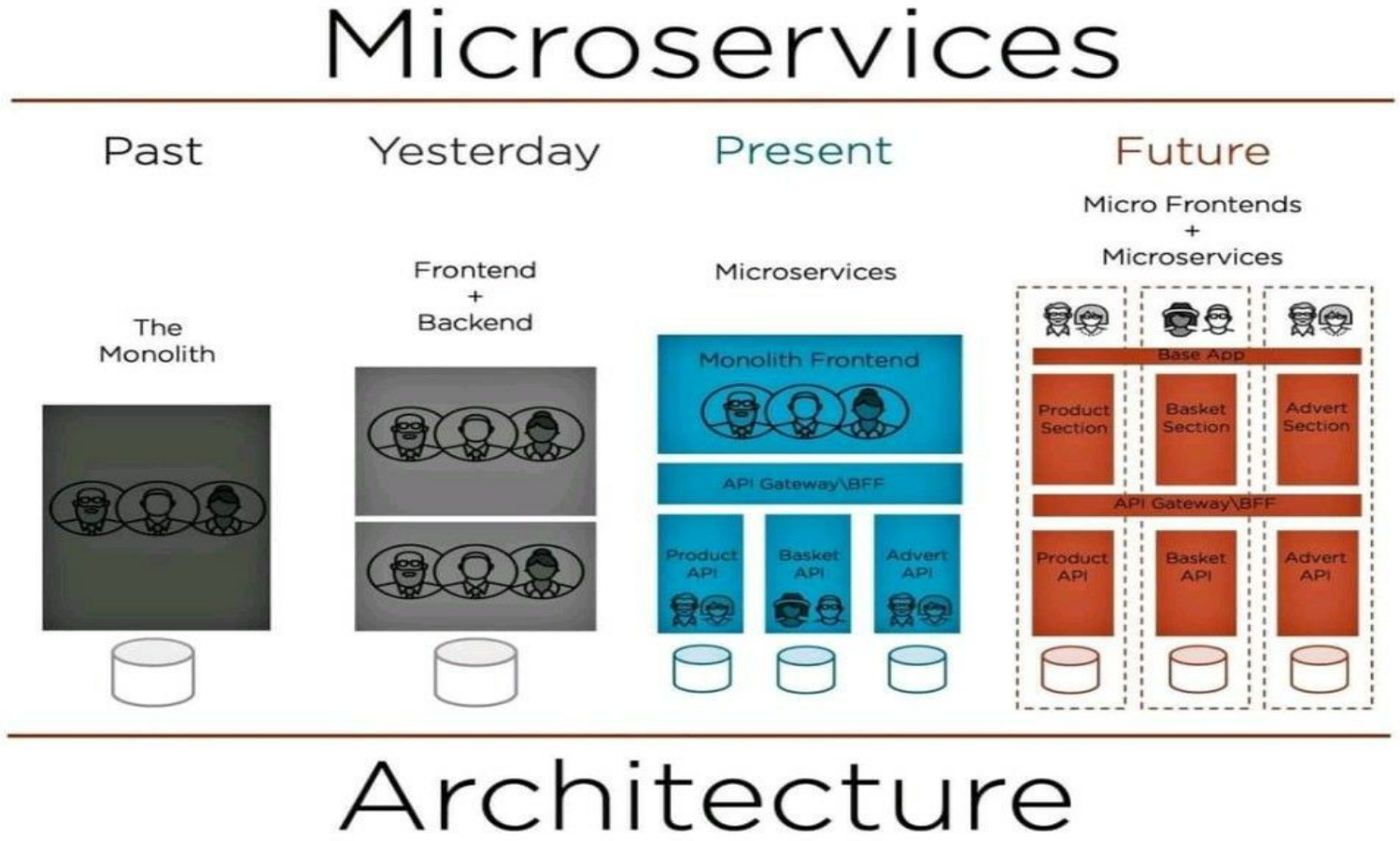
Disclaimer

These slides are made with a lot of effort, so it is a humble request not to share it with any one or reproduce in any way.

QUIZ

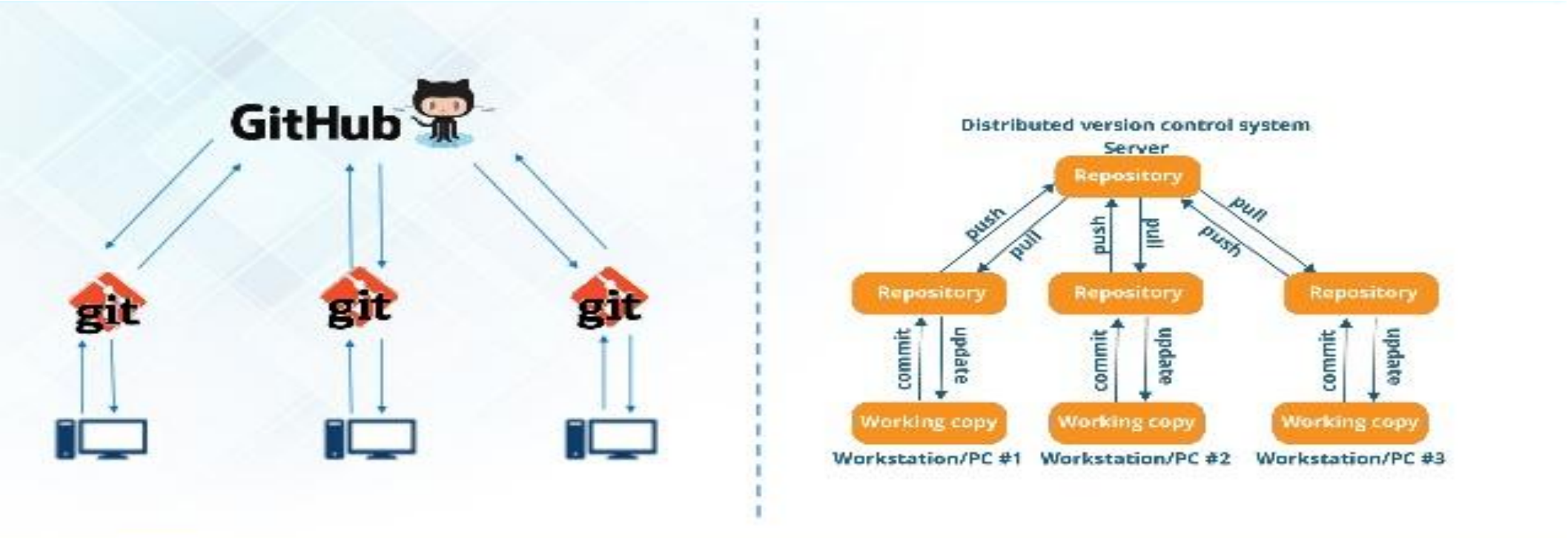
- What is DevOps
- CI vs Continuous Delivery vs Continuous Deployment
- Stages of Git
- Different commands for different stages of Git
- 3 methods of git reset
- How does ssh work

Microservices



Git & Github

From: https://www.slideshare.net/EdurekaIN/what-is-git-what-is-github-git-tutorial-github-tutorial-devops-tutorial-edureka

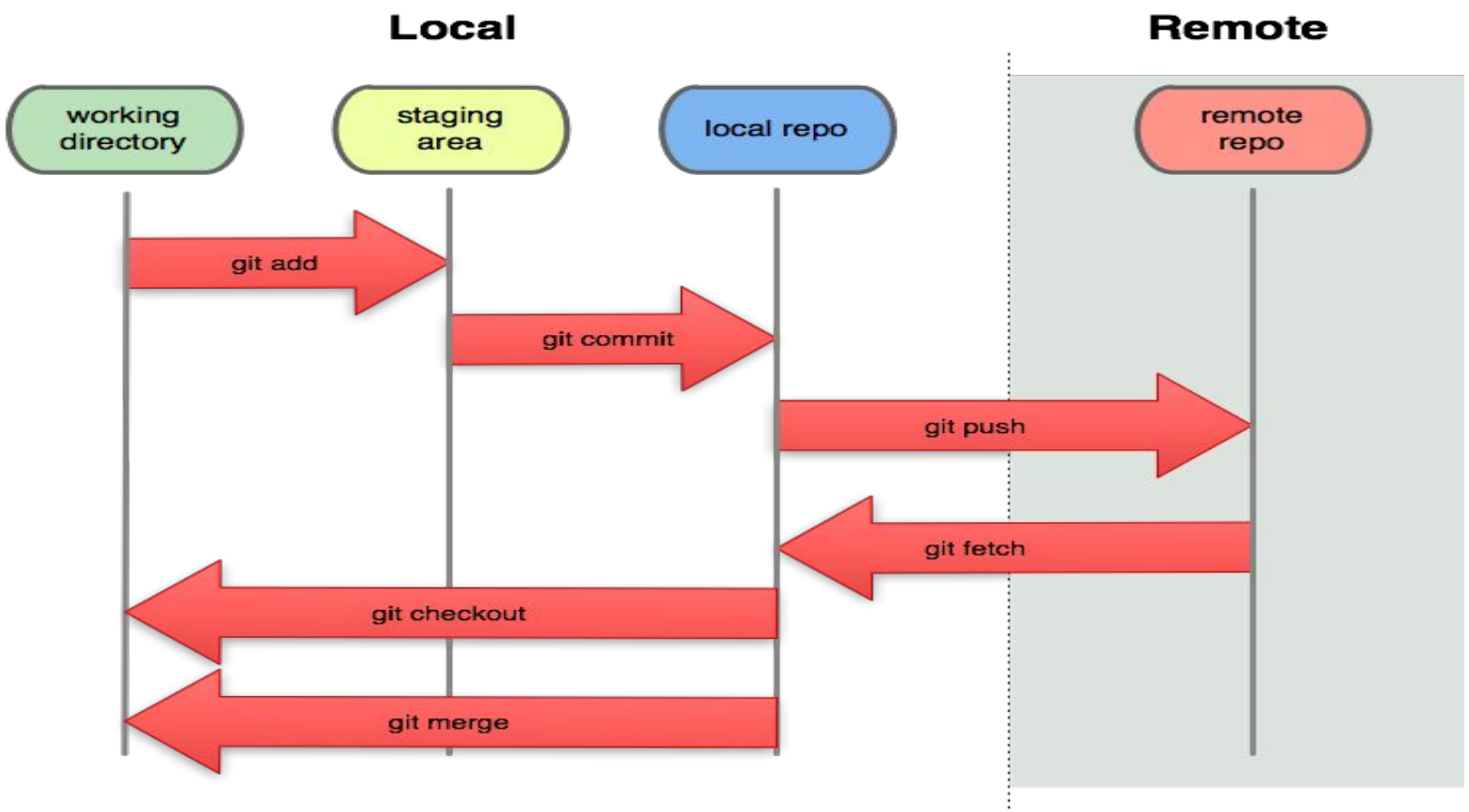


Git & Github

From: https://andersenlab.org/dry-guide/latest/github/

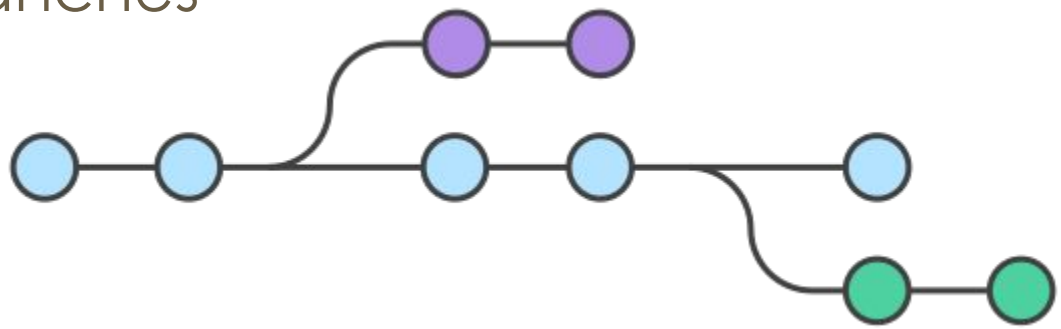
git	GitHub
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the **Git** repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

Stages of Git



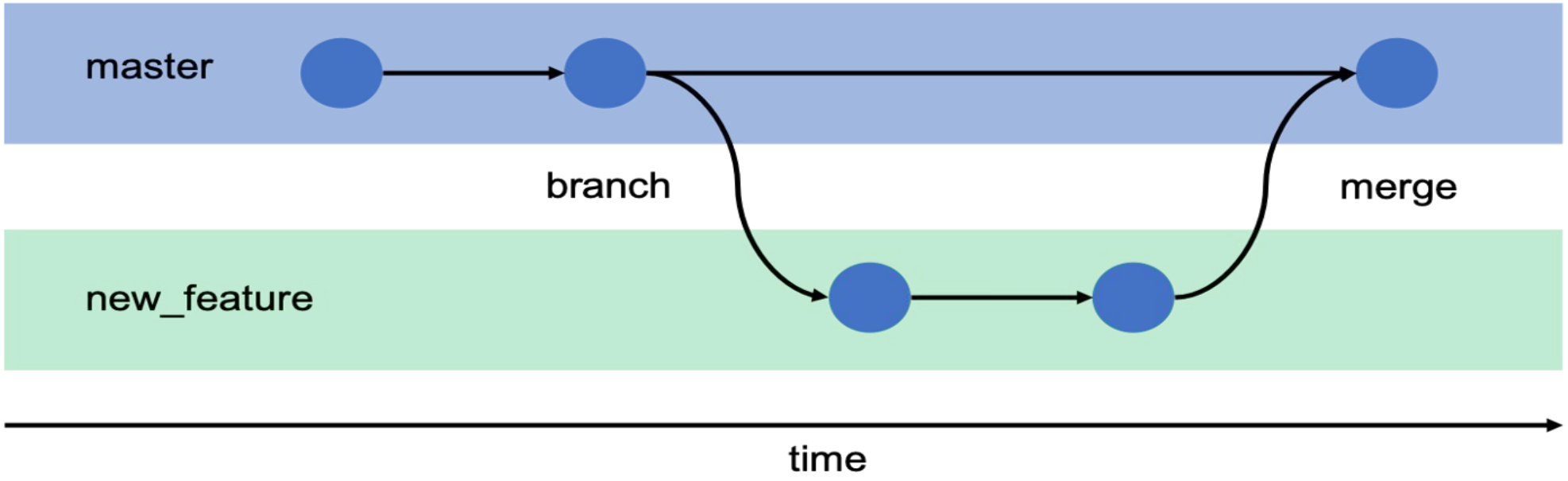
Branching Basics

- Independent/isolated line of development
- Inexpensive/lightweight as compared to other VCS
- Way to work on different version of a repository
- Work in parallel
- Main/Master is the default branch
- Branches are just pointers to commits
- Protected branches



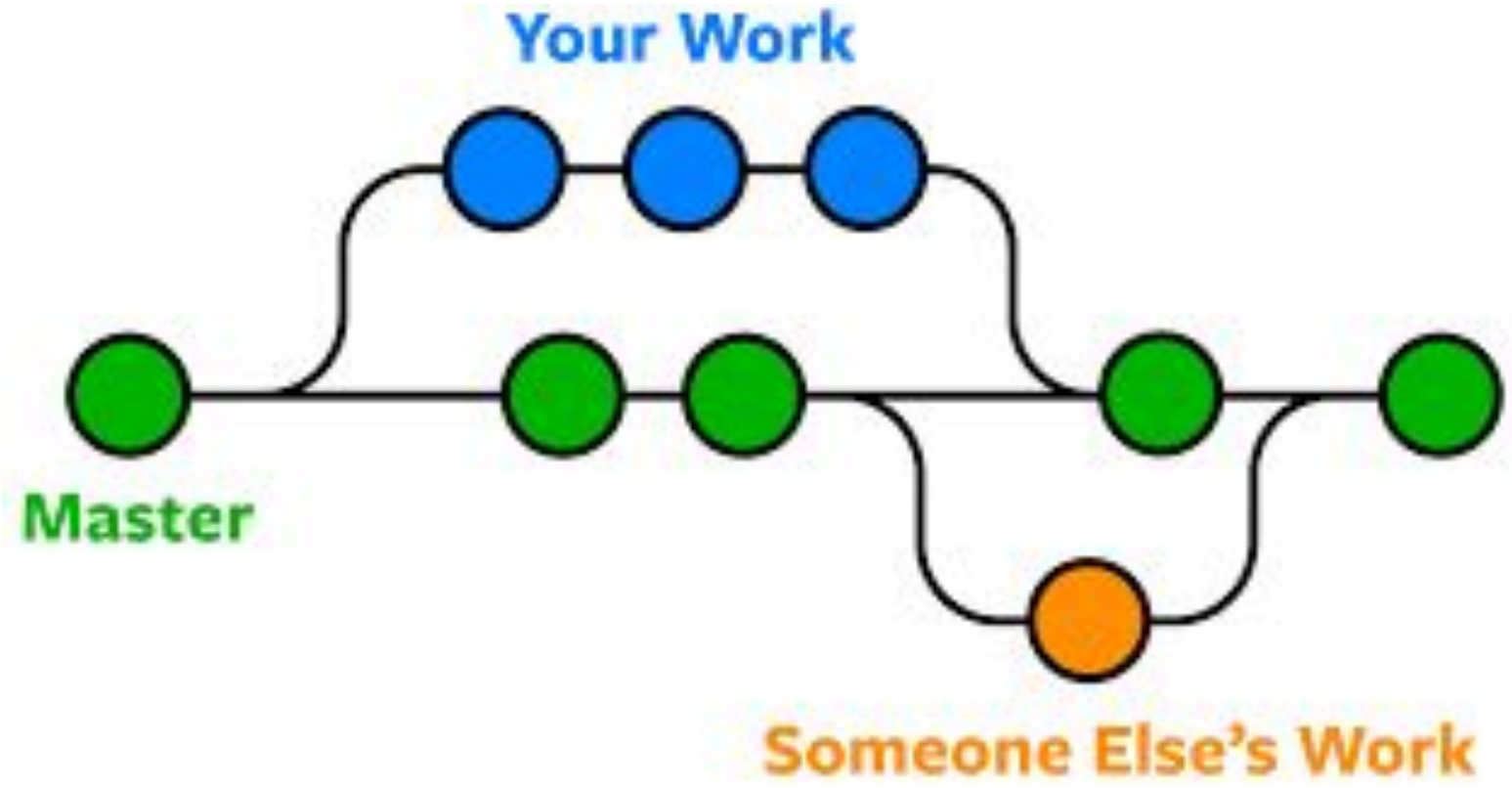
Branching

From: https://medium.com/@natetadesse4991/git-branches-and-merging-overview-17810959c28a



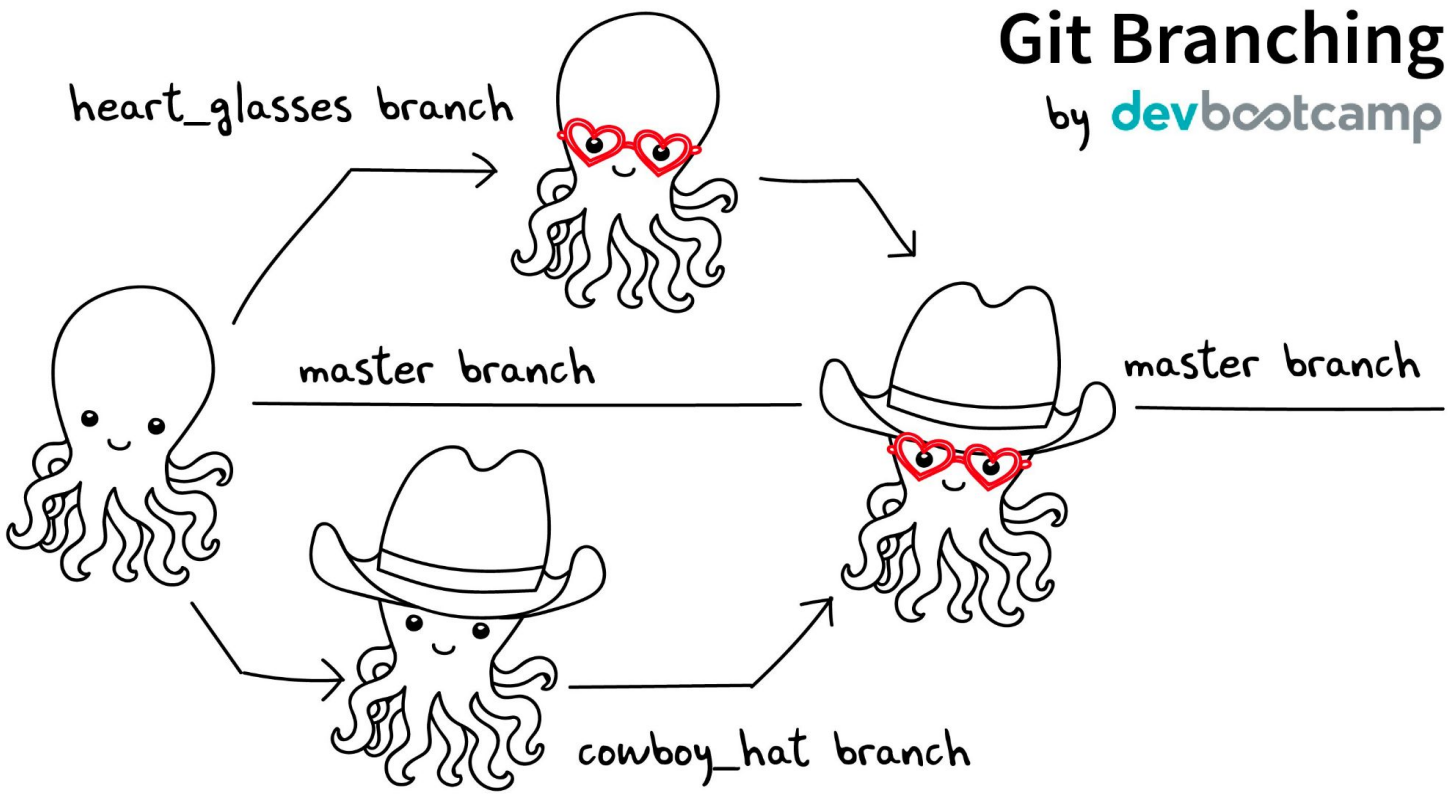
Branching

From: http://blog.alwawee.ru/?p=2768



Branching

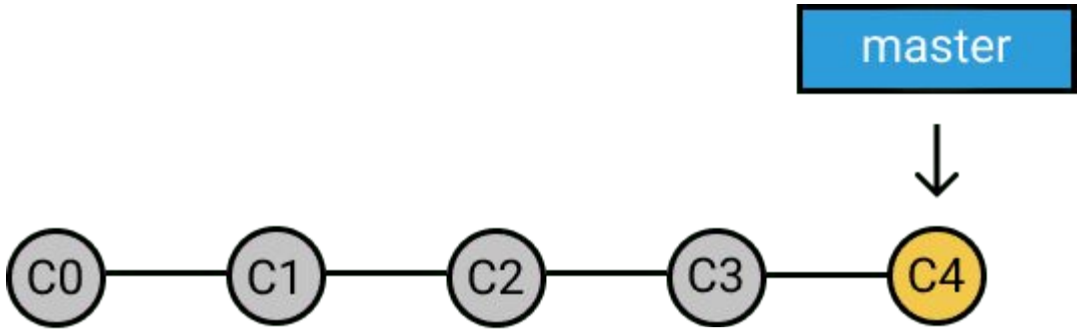
From: https://twitter.com/jay_gee/status/703360688618536960/photo/1



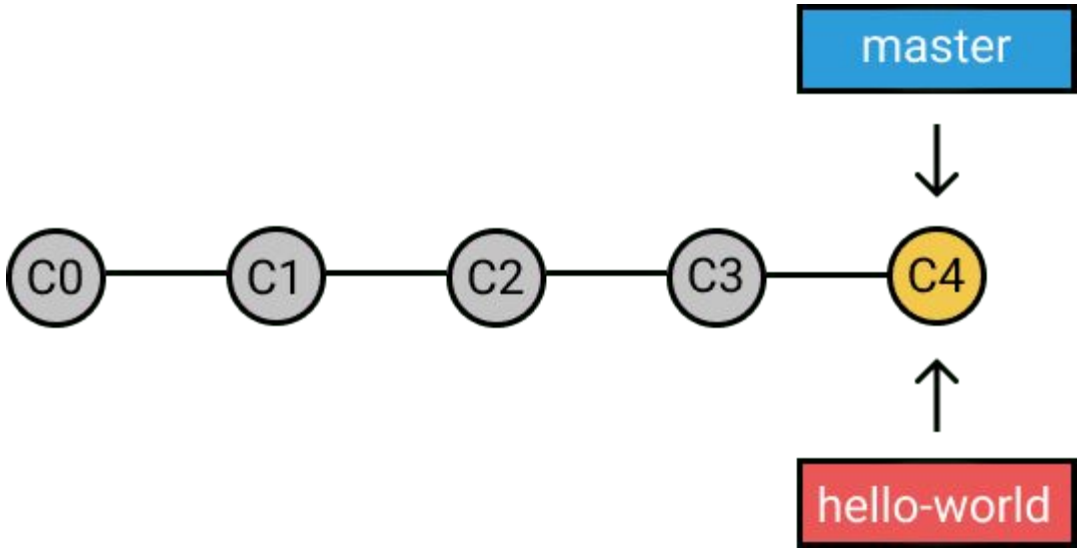
Branching Commands

- List Branches: git branch
- List all Branches: git branch -a
- Create Branch: git branch [branch_name]
- Switch Branch: git checkout [branch_name]
- Delete Branch: git branch -d [branch_name]
- Delete remote Branch: git push origin --delete [branch_name]
- Create Branch and Switch: git checkout -b [branch_name]

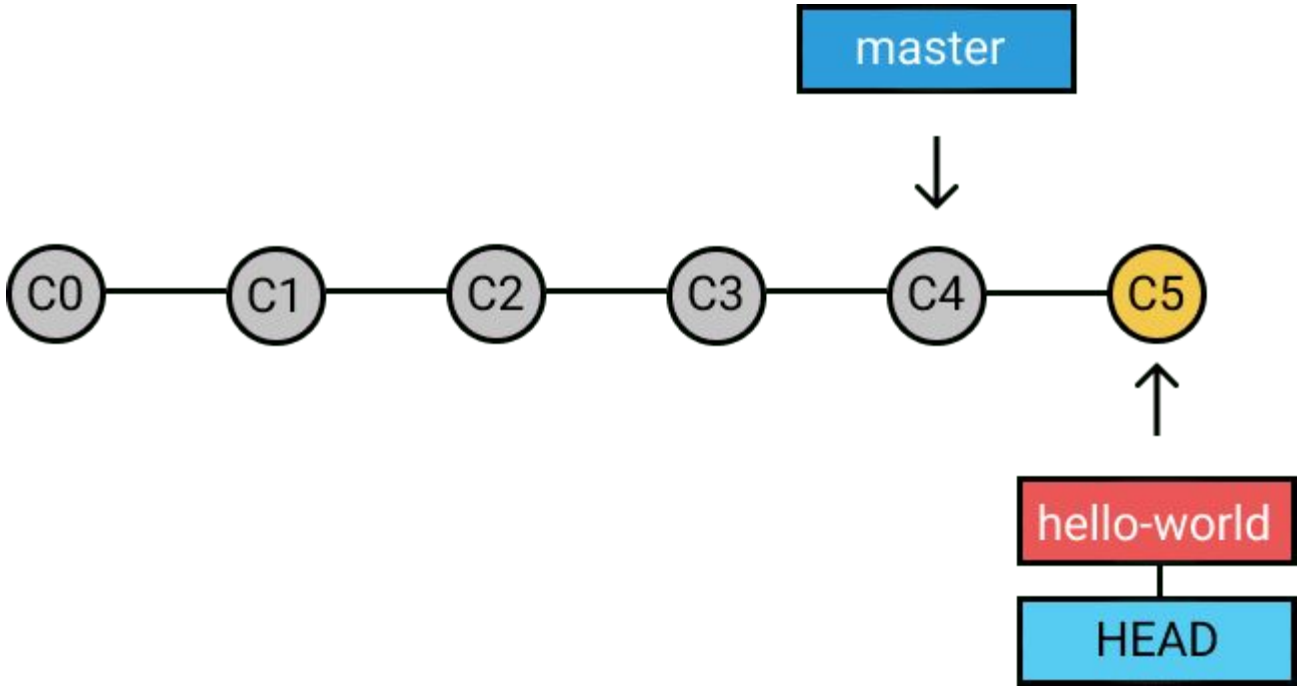
Branching



Branching



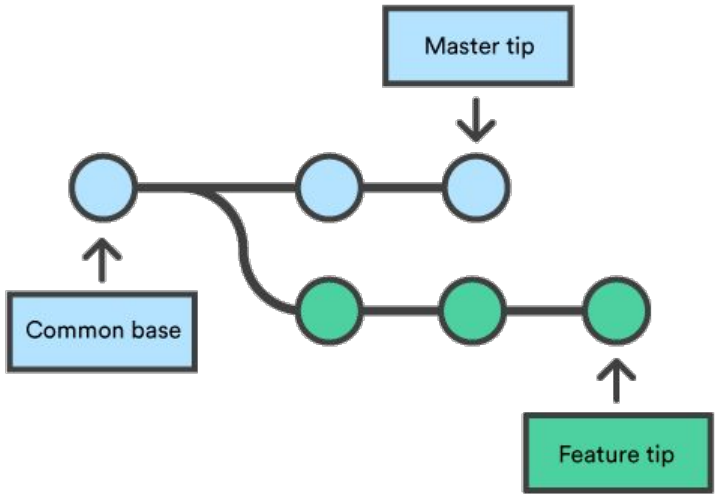
Branching

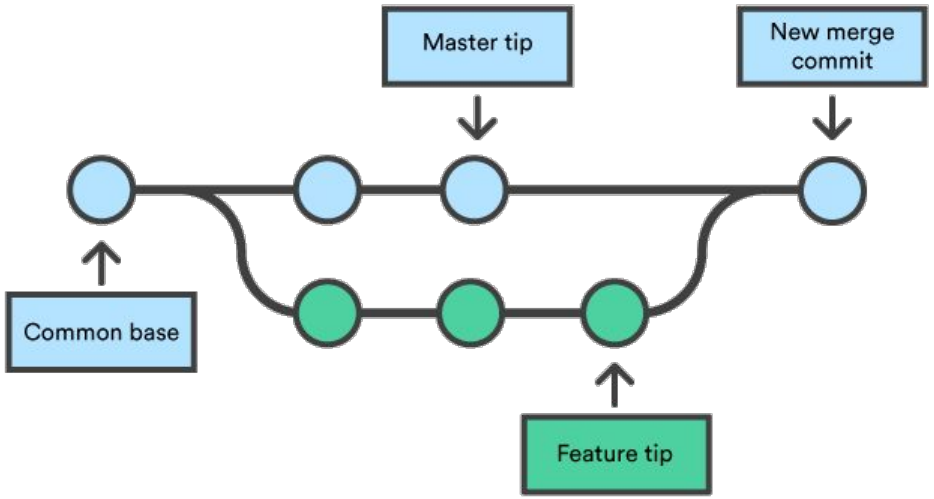


Branching

Merging

- Combine multiple branches into one
- Combines multiple sequence into unified history
- Commands
  - Merge branch into active branch: git merge [branch name]
  - Merge branch into target branch: git merge [source branch] [target branch]

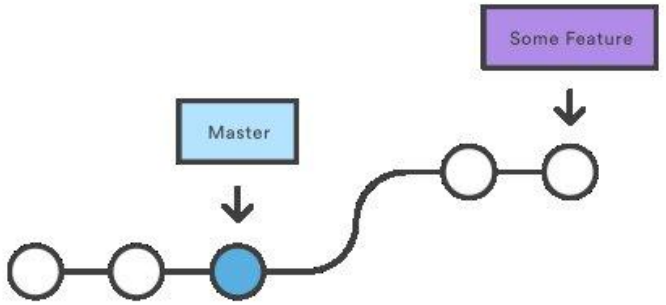




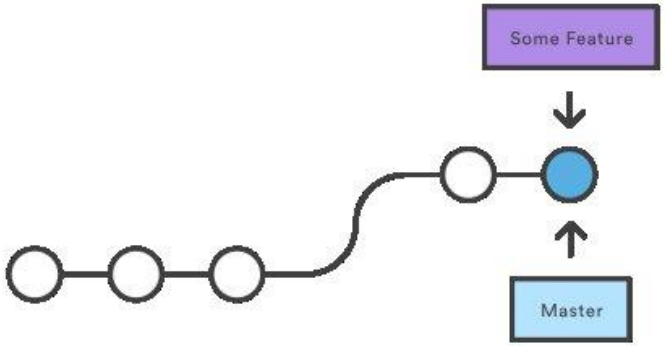
Merging - Fast Forward

/kahootali

Before Merging



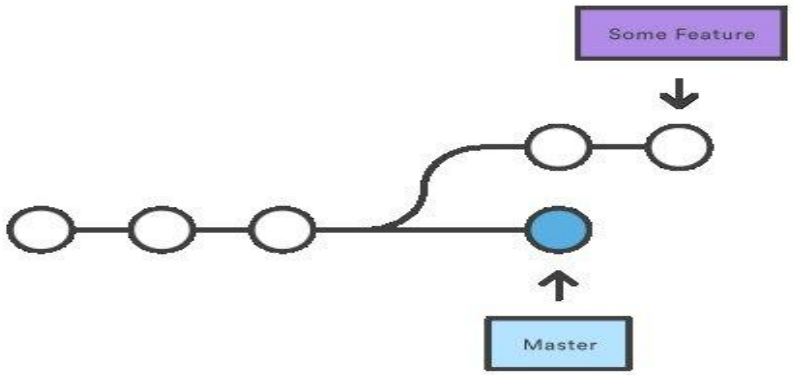
After a Fast-Forward Merge



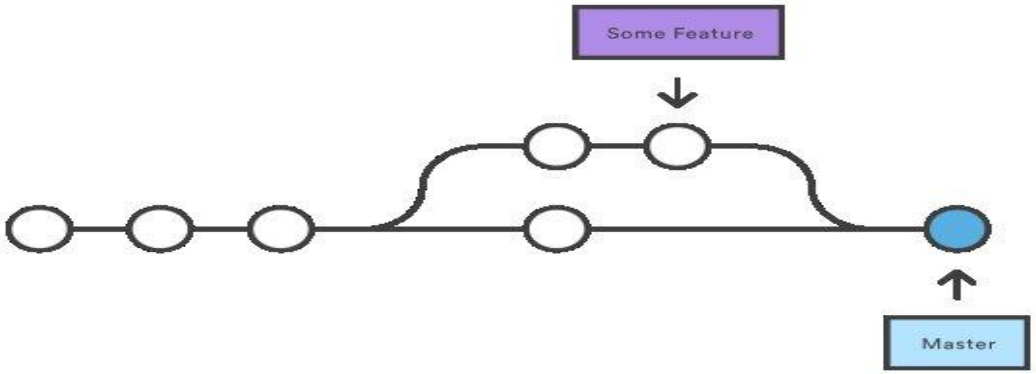
Merging - Three Way

/kahootali

Before Merging



After a 3-way Merge



Merging - Resolving Conflict

/kahootali

- Git automatically merge commits unless there are changes that conflict

```
On branch master
Unmerged paths:
(use "git add/rm ..." as appropriate to mark resolution)
both modified: hello.py
```

```
here is some content not affected by the conflict
<<<<<<< master
this is conflicted text from master
=======
this is conflicted text from feature branch
```

======= marker is the receiving branch and the part after is the merging branch

Lab - Branching

/kahootali

- Create Branch

```
git branch test-branch
```

- List Branches

```
git branch
git branch --list
Remote:   git branch -r
All:      git branch -a
```

- Switch Branch

```
git checkout test-branch
```

Lab - Branching

/kahootali

- Commit to new Branch

  Switch to new branch & create a new file test-branch.txt and edit contents

```
vi test-branch.txt
git add test-branch.txt
git commit -m "create test-branch.txt file"
```

- Rename Branch

```
git branch -m test-branch test-new-branch
```

- Delete Branch

```
git checkout main
git branch -d test-new-branch
```

Lab - Fast Forward Merge

/kahootali

```
Create new branch & checkout
    git branch merge-ff
    git checkout merge-ff
Add new file & commit to branch
    vi merge-ff.txt
    git add merge-ff.txt
    git commit -m "create merge-ff.txt file"
Merge new branch to Main
    git checkout main
    git merge merge-ff
    git log --oneline --graph --decorate
    git branch -d merge-ff
```

Lab - Three Way Merge

/kahootali

```
Create new branch & checkout
    git checkout -b merge-tw main
Add new file & commit to branch
    vi merge-tw.txt
    git add merge-tw.txt
    git commit -m "create merge-tw.txt file"
Move to Main & make a new commit there
    git checkout main
    vi main-merge-ex.txt
    git add main-merge-ex.txt
    git commit -m "create main-merge-ex file"
```

Lab - Three Way Merge

/kahootali

```
Merge the new branch to Main
    git merge merge-tw
    git log --oneline --graph --decorate
    git branch -d merge-tw
```

Lab - Resolving Conflict

/kahootali

Create new branch & checkout

```
git checkout -b conflict main
```

Edit existing file, append “Hello from branch” at end & commit to branch

```
vi main-merge-ex.txt
git add main-merge-ex.txt
git commit -m "edit main-merge-ex.txt file"
```

Move to Main, edit the same file, append “Hello from main” & make a new commit there

```
git checkout main
vi main-merge-ex.txt
git add main-merge-ex.txt
git commit -m "edit main-merge-ex.txt file on main branch"
```

Lab - Resolving Conflict

/kahootali

```
Merge the branch and you will get conflict
    git merge conflict
    git status
Open the file main-merge-ex.txt, resolve the conflict
    git add main-merge-ex.txt
    git commit -m "resolve merge conflict"
```

Rebasing

- Alternative to integrate changes from one branch to another
- Changes the base of your branch from one commit to another
- Integrates upstream changes into your branch
- Linear history
- Command: git rebase

Merging vs Rebasing

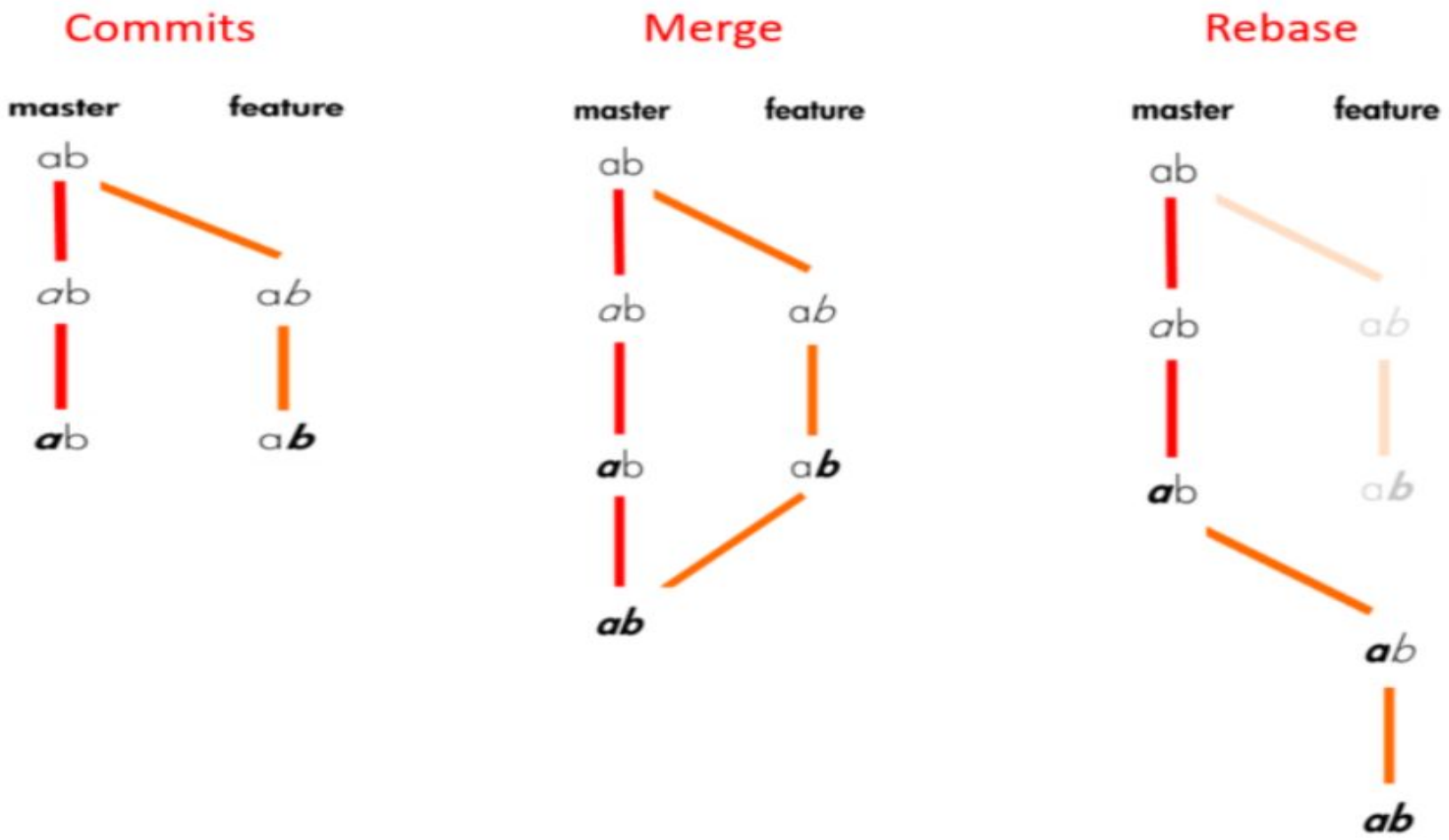
Merging: https://miro.medium.com/max/4000/0*fQxnxhrYC6Exwn8d.gif

Rebasing: https://imgur.com/aowdZq7

https://www.atlassian.com/git/tutorials/merging-vs-rebasing

https://www.edureka.co/blog/git-rebase-vs-merge/

Merging vs Rebasing



Lab - Rebasing

/kahootali

```
Create new branch & checkout
    git checkout -b rebase-test
Create new file
    vi rebase-file.txt
    git add rebase-file.txt
    git commit -m "create rebase-file in rebase-test branch"
    git log --oneline --graph --decorate
Switch to main branch, edit a file and commit
    git checkout main
    vi main-merge-ex.txt
    git add main-merge-ex.txt
    git commit -m "edit main-merge-ex.txt file"
```

Lab - Rebasing

/kahootali

```
Rebase
    git checkout main
    git rebase rebase-test
    git log --oneline --graph --decorate
    git branch -d rebase-test
```

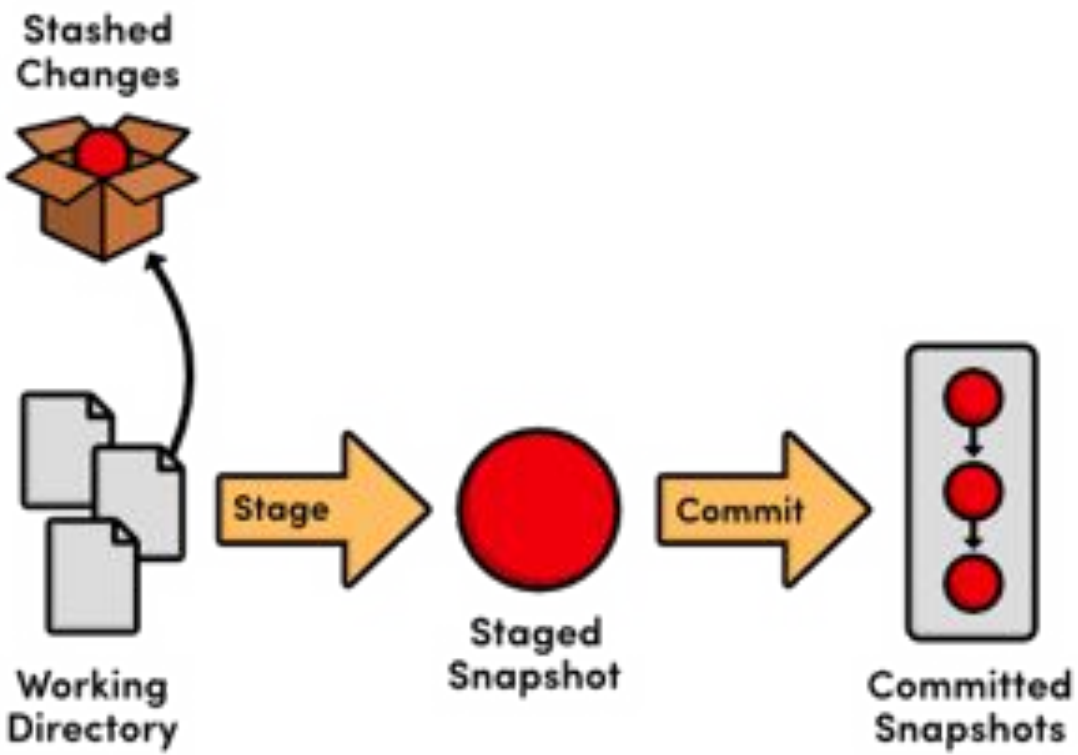
Stashing

- Why do we need Stashing?
- Saves you uncommitted code for later use
- Reverts changes from your working copy
- You can make changes, create new commits, switch branches, and perform any other Git operations and then re-apply your stash when you're ready
- Local to your Git Repo
- Commands:
  - List Stashes: git stash list
  - Stash Changes: git stash
  - Apply Stash: git stash apply or git stash pop
  - Clear all Stashes: git stash clear
  - Stash with description: git stash save "message"

Stashing

/kahootali

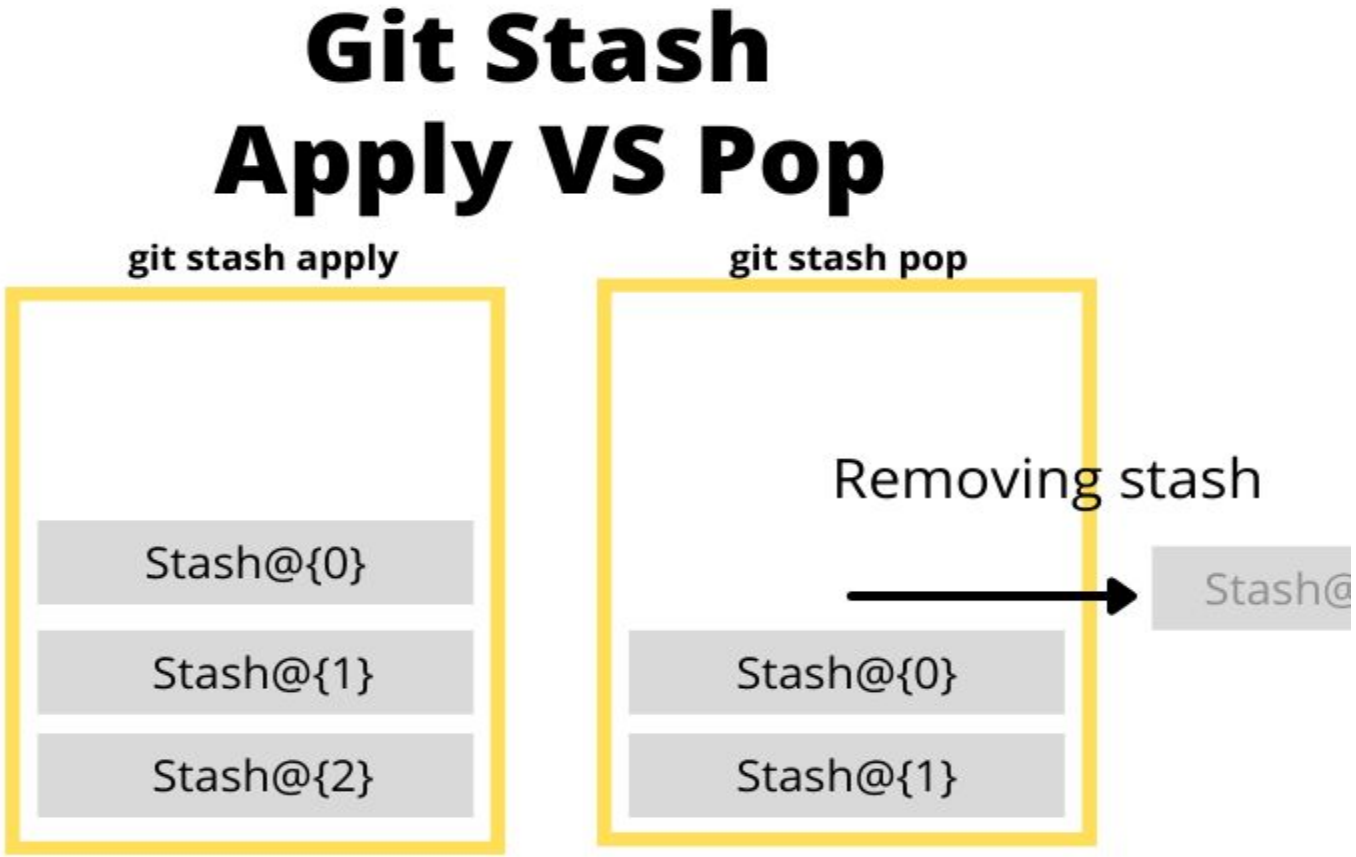
From: https://code.tutsplus.com/tutorials/quick-tip-leveraging-the-power-of-git-stash--cms-22988



Stashing

/kahootali

From: https://www.becomebetterprogrammer.com/git-stash-with-name/



Lab - Stashing

/kahootali

```
STASH CHANGES
    git branch stash-example
    git checkout stash-example
Change any file
    vi master-merge-ex.txt
    git status
    git stash
    git stash list
    git status
After stashing the changes your working directory will be clean
```

Lab - Stashing

/kahootali

```
APPLY STASHED CHANGES
    git status
    git stash pop
    git status
Commit the file
    git add master-merge-ex.txt
    git commit -m "edit master-merge-ex file for stash example"
    git checkout master
    git branch -d stash-example
```

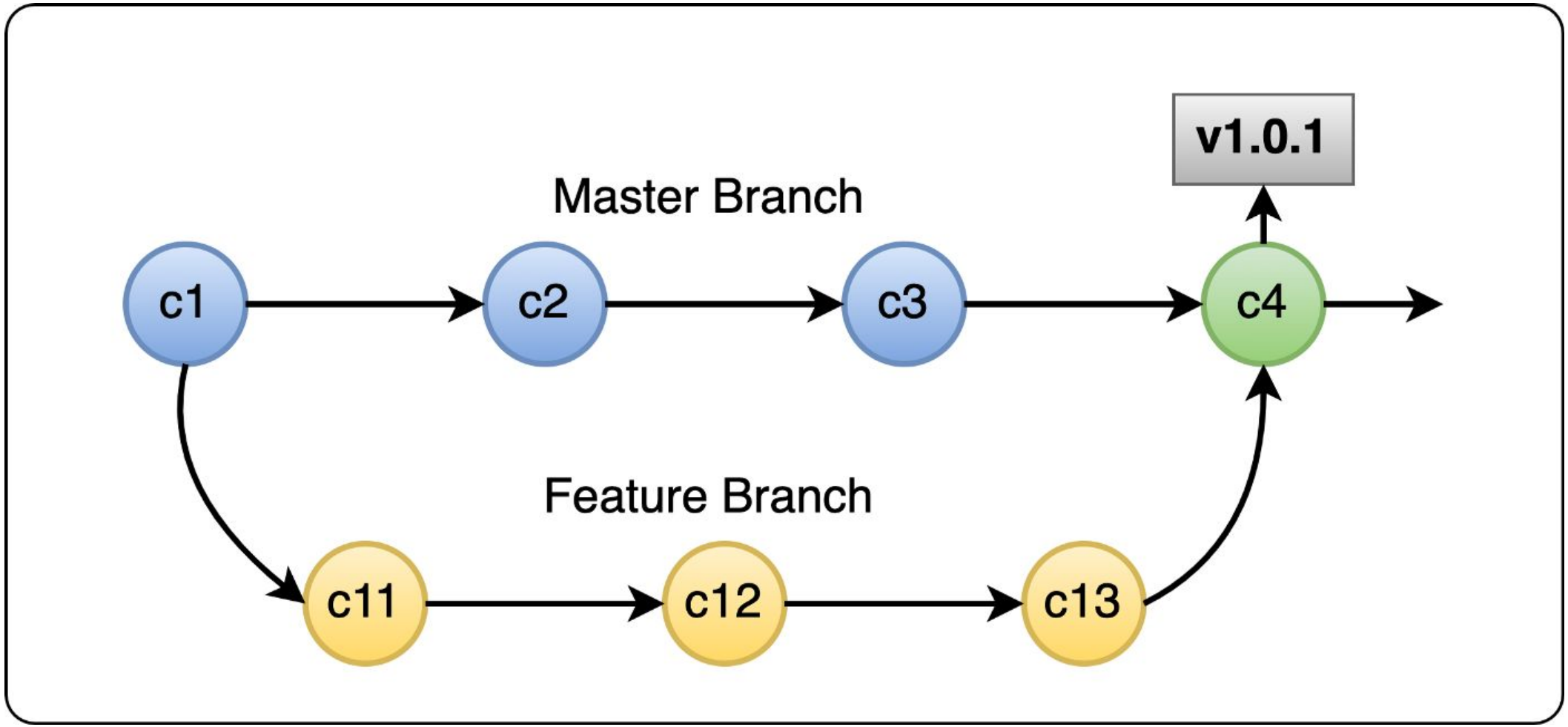
Tagging

- Allows you to give commit a name
- Mark important checkpoint in the project
- Tags
  - Annotated: extra metadata e.g. tagger name, email, and date.
  - Lightweight: only tag name
- Branch that doesn't change
  - You can checkout to them
- Commands
  - List tags: git tag -l
  - Lightweight tag: git tag <tag-name>
  - Annotated tag: git tag -a <tag-name> -m <message>
  - Tag Details: git show <tag-name>
  - Delete tag: git tag --delete <tag-name>

Tagging

/kahootali

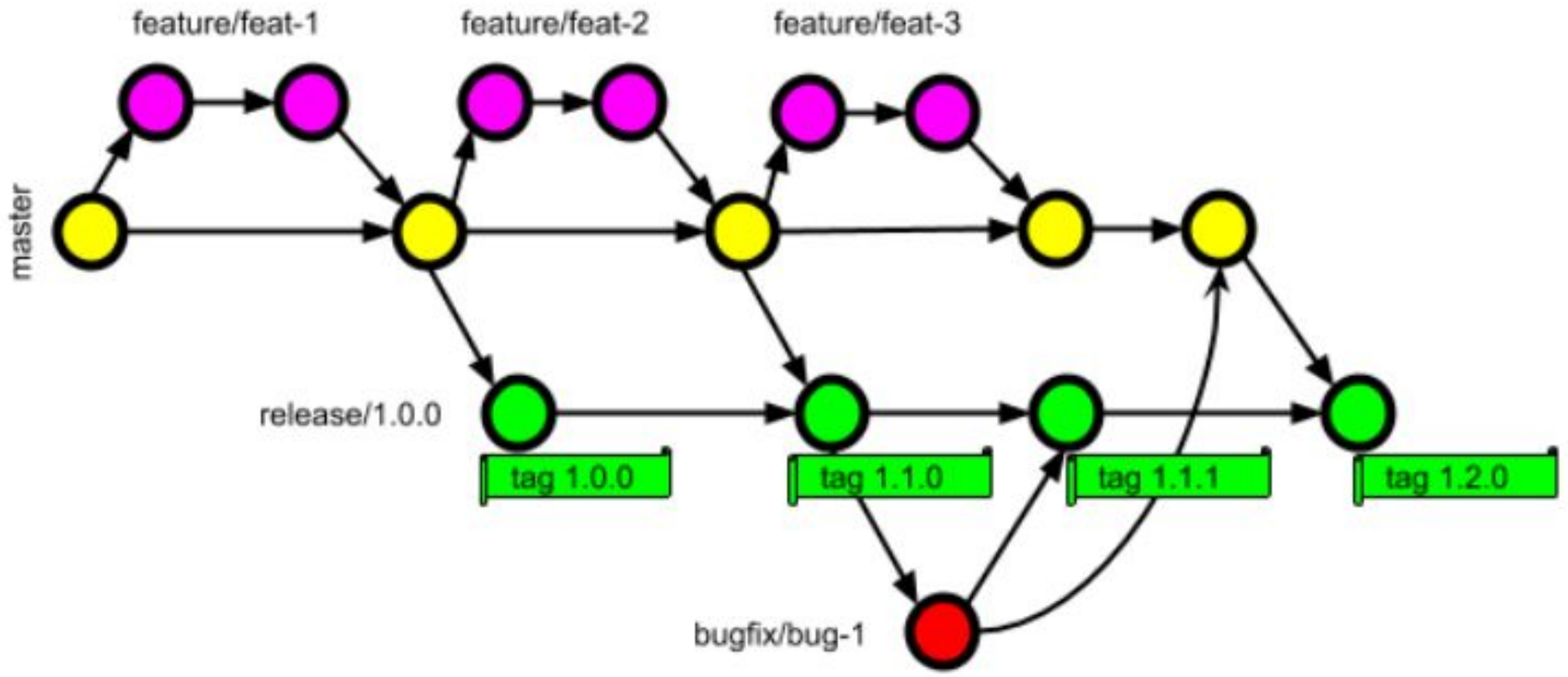
From: https://medium.com/javarevisited/git-tag-all-you-need-to-know-8d859b5bb506



Tagging

/kahootali

From: https://stackoverflow.com/questions/63575134/are-git-tags-merged



Versioning in Git

- Creating Tags & Releases in Git
- Helpful in not managing the Version through a separate file
- The tagging can be used when creating pipelines
- Creating pipeline/image version

Tags vs Releases

- Tag is given by Git, Releases are supported by Github/Gitlab, etc

- Tag is just pointer, In Releases, you can add description about release, changelogs, or add release artifacts

- Examples:
  - https://github.com/argoproj-labs/argocd-operator/releases
  - https://github.com/stakater/Reloader/releases
  - https://github.com/kubernetes/kubernetes/releases
  - Only tags: https://github.com/nginxinc/docker-nginx

Lab - Tagging

- Create Lightweight Tag

```
git tag "first-tag"
```

- Create Annotated Tag

```
git tag -a "second-tag" -m "tag-test"
```

- List Tags

```
git tag -l
```

- Tag Old Commit

Show list of last three commits

```
git log -n3
Copy Commit Hash
git tag "third-tag" HASH
```

Lab - Tagging

- Get Information On a Tag

```
git show first-tag
git show second-tag
```

- Delete Tags

```
git tag --delete first-tag
git tag --delete second-tag
git tag --delete third-tag
```

Pull Request

- Method of submitting contribution
- Makes collaboration easier
- Workflow
  - Create Branch
  - Make Changes
  - Create Pull Request
  - Merge Once Approved

Lab - Pull Request

/kahootali

- Create Pull Request

```
git checkout -b pr-example main
vi pr-example.txt
git add pr-example.txt
git commit -m "add new file"
git push origin pr-example
```

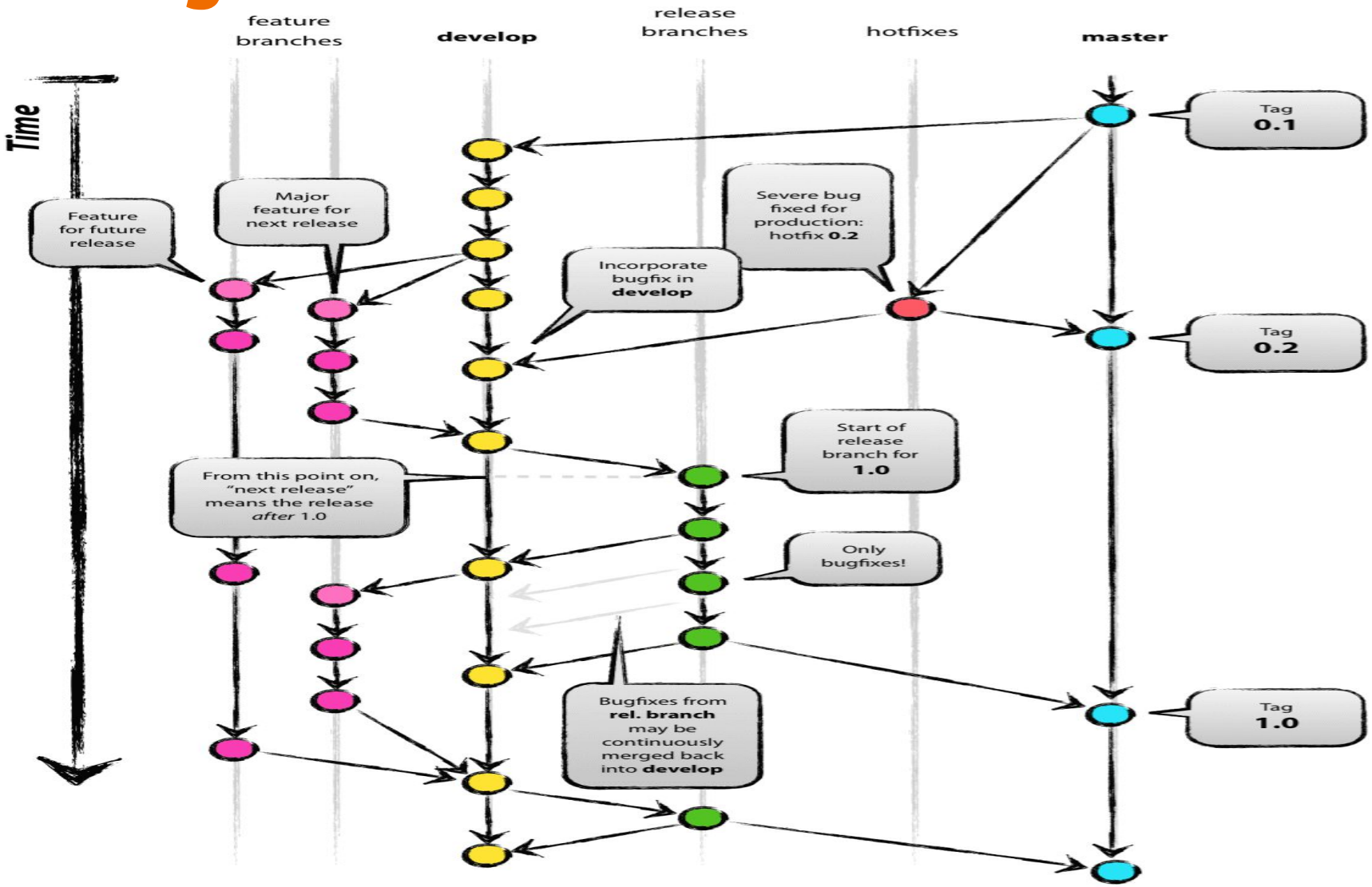
When you push you will get the link to create Pull Request or you can go to the repo, and create Pull Request

Lab - Pull Request

- Can have Branch Protection
- Approvals
- Pipeline workflows
- Include Administrators

Branching Model

/kahootali



DevOps Course By M. Ali Kahoot - Dice Analytics

Forking

- Copy of a repository
- Server-side copy as compared to clone
- Allows to freely experiment without affecting the original project
- Use Cases
  - Propose changes to someone else's project
  - Contribute to open source projects

Contribute to Open Source Projects

/kahootali

- Create Github Issue
- Fork the Project
- Clone your copy of Project
- Do the work
- Push to your remote repository
- Create a new Pull Request in GitHub
- Review by Maintainers
- Respond to any code review feedback

Lab - Forking

- Fork Repository

Best Practices

- Don’t commit directly to main
- Create .gitignore file for your projects
- Don’t store credentials as code/config in GitHub
- Write meaningful commit message
- Test Your Code Before You Commit
- Use Branches
- Always pull the latest updates
- Protect your project/branches
- The more approvals, the better
- Rebase your branches periodically
- Agree on workflow

Git Playground

- http://git-school.github.io/visualizing-git/
- https://learngitbranching.js.org/?NODEMO
- https://learngitbranching.js.org/

Things to do before next class

- Install Docker on the Ubuntu system, you can search how to install Docker on Ubuntu, it's really simple, to verify, you must be able to run the command "docker info" or "sudo docker info"

- Complete all the labs in git slides