Git Hub Flow

Crash Course

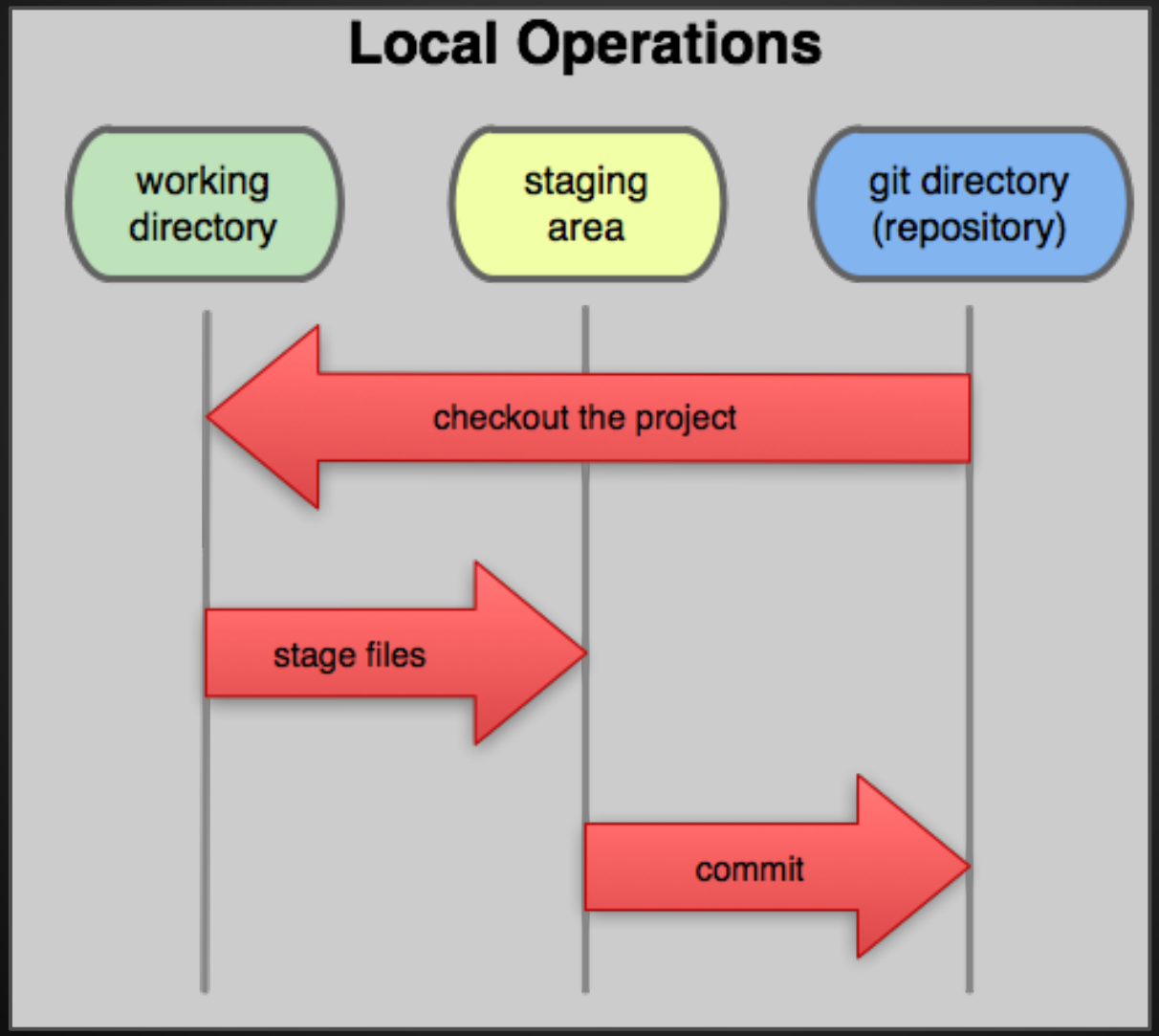
Overview

- Git Concept/Commands
- Overview of GitHub
- Getting Started

What is Git

- Decentralized Version Control
- Easily store, manage, update, share code
- Three Areas:
  - Working Directory
    - This is the actual files you are working with at the current moment
  - Staging
    - After running git add x, git stores the changes/files you've made in a preview area
  - Local Repository
    - After running git commit, git stores the changes into your local repository

Git Local Basics

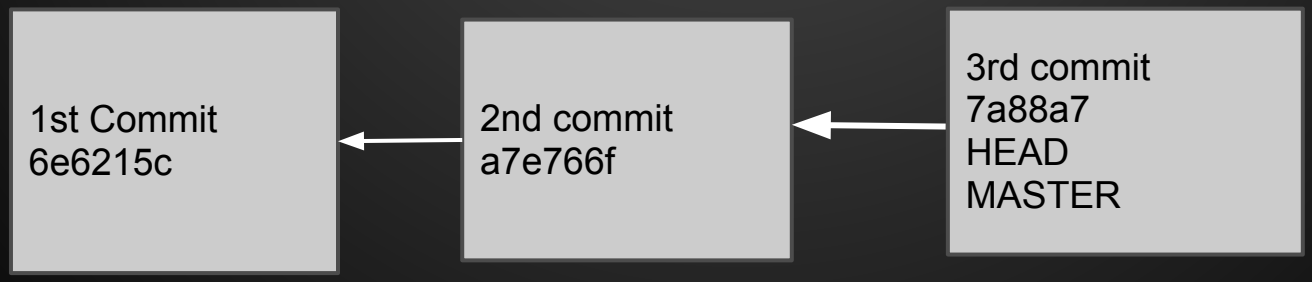


Git Basics

- Create Local Git Repo
  - git init
- If you have existing repo
  - git clone url_address
- Check Status of files in Working Directory, Staging
  - git status
- Working Directory -> Staging
  - git add .
- Staging -> Local Repository
  - git commit -m 'useful commit message'

Git Commits

- Each commit is snapshot and has a reference back to its ancestor
- Each commit has a unique hash
- Useful Commands:
  - Commit with message
    - git commit -m 'message'
  - Commit any modified existing file
    - git commit -a -m 'message'
  - Get List Commits and Hash Code
    - git log
  - Revert back to previous commit, please note all changes since commit will be lost!
    - git reset --hard commit_hash_here



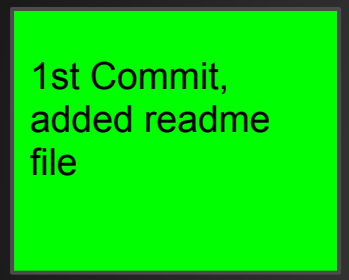
Quick Demo

- Commit
  - git commit -a -m 'test'
- Commit History
  - git log
- Reverting back to previous commit
  - git reset --hard commit_hash_here

Git Branches

- Creates a new tree from current commit
- Every git starts on **master** branch
- To create branch from current commit
  - git checkout -b 'new branch name'
- To Switch Branches
  - git checkout branch_name
- To merge changes from another branch into current branch
  - git merge branch_name
- To See current Branch and List of Branches
  - git branch
- To delete branch
  - git branch -d 'branch_name'
- To push local branch to remote server
  - git push origin branch_name
- To pull from remote branch
  - git pull origin branch_name
- Remove Branch on remote server
  - git push origin --delete branch_name

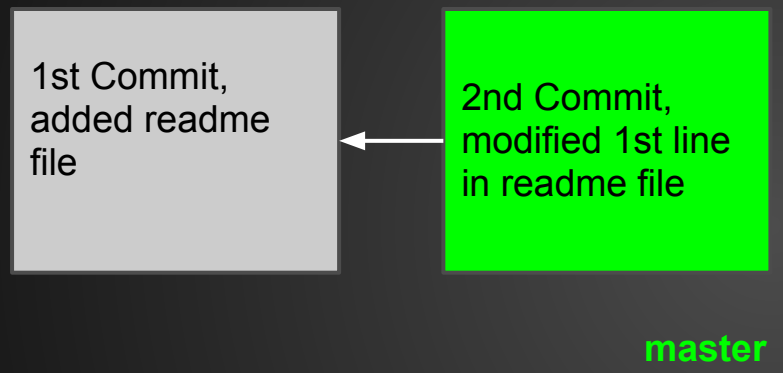
Git Branch Demo



master

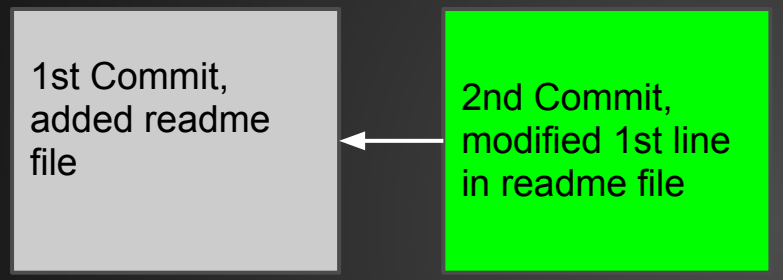
```
git init
vim readme.txt
git add readme.txt
git commit -m '1st commit'
```

Git Branch




```
git add .
git commit -m '2nd commit'
```

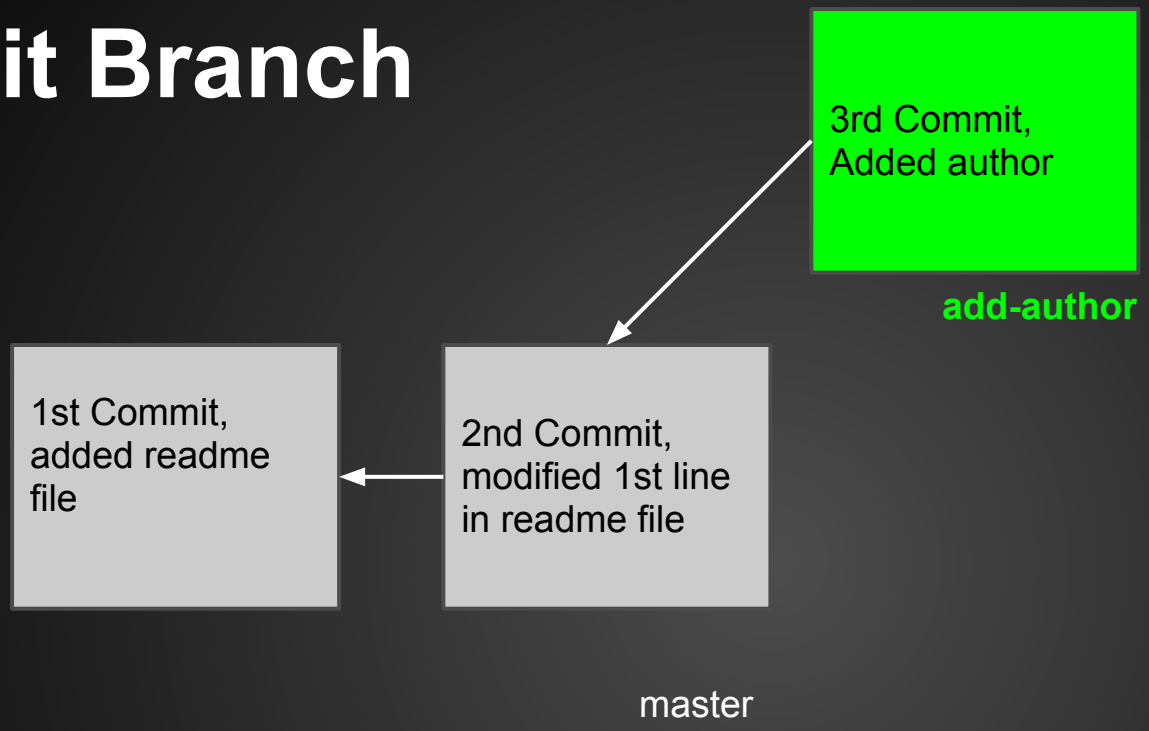
Git Branch



master
add-author

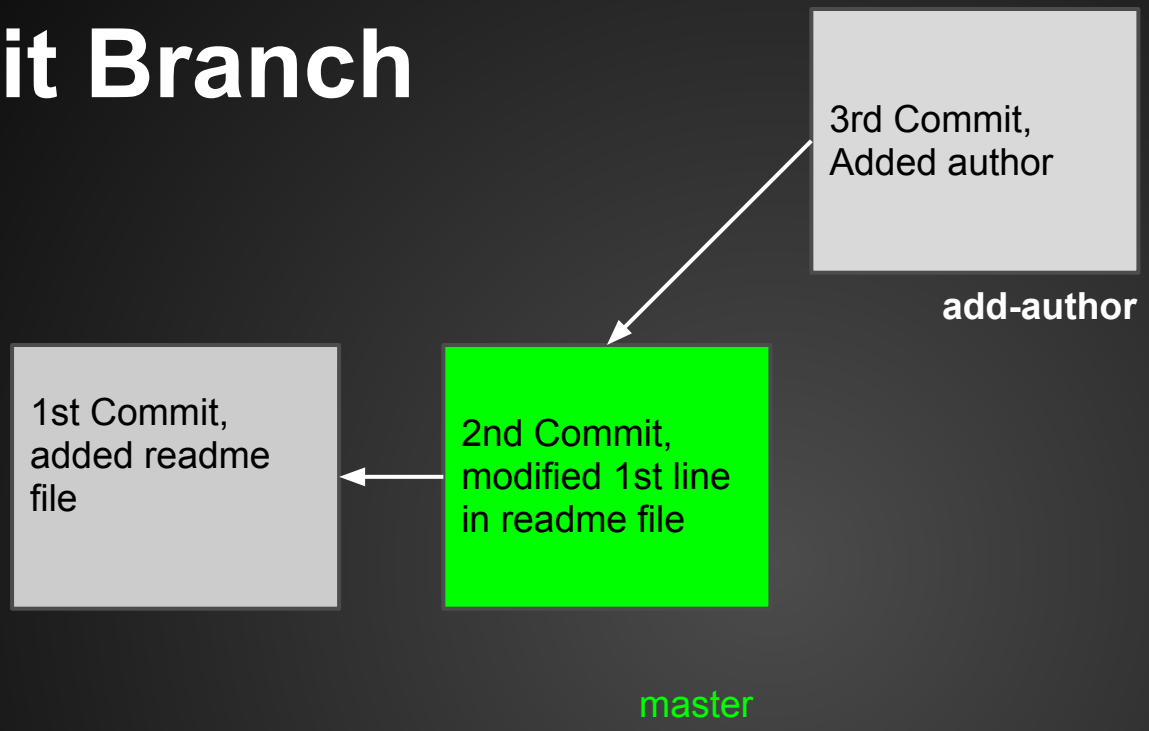
git checkout -b 'add-author'

Git Branch



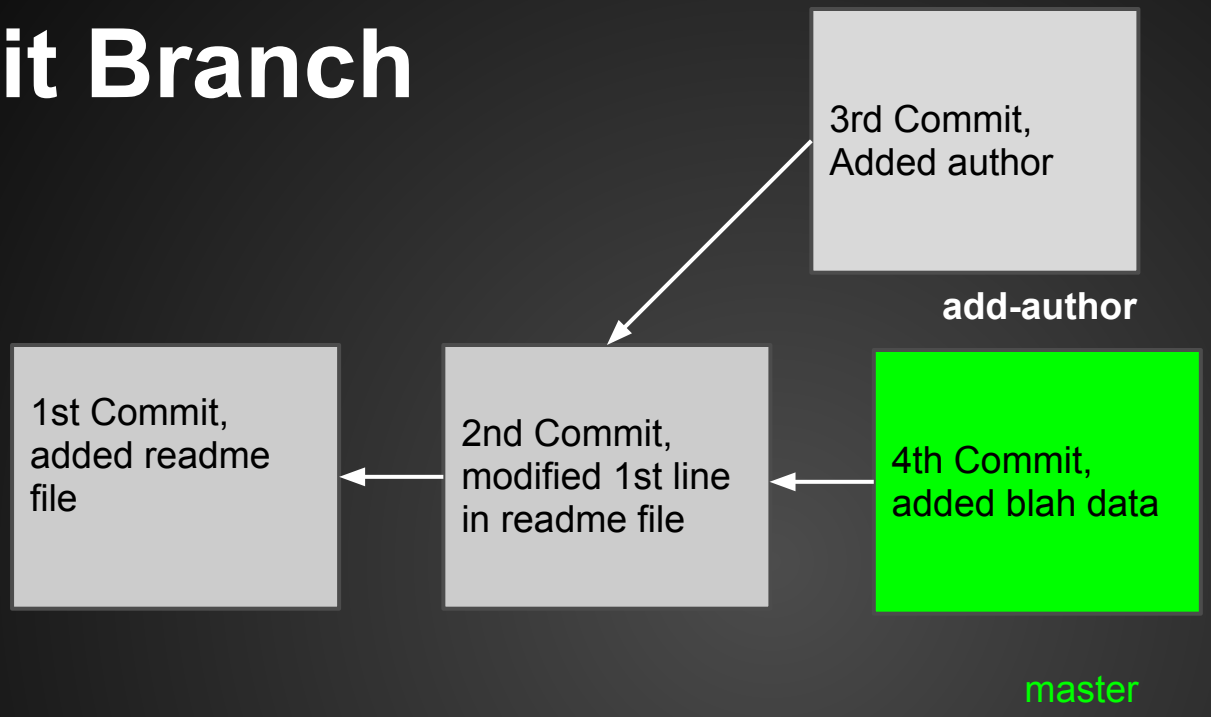

```
vim readme.txt (add author name)
git add .
git commit -m 'added author'
```

Git Branch



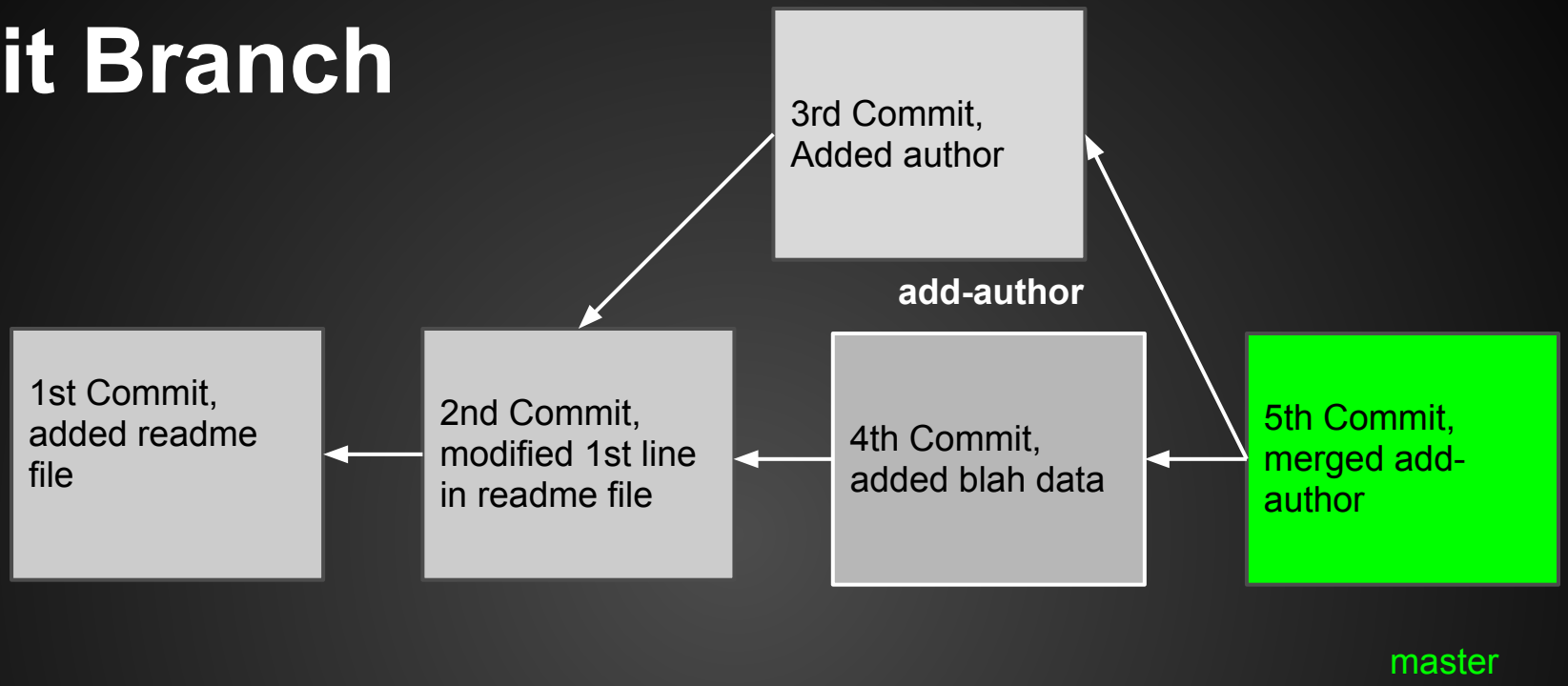
git checkout master

Git Branch



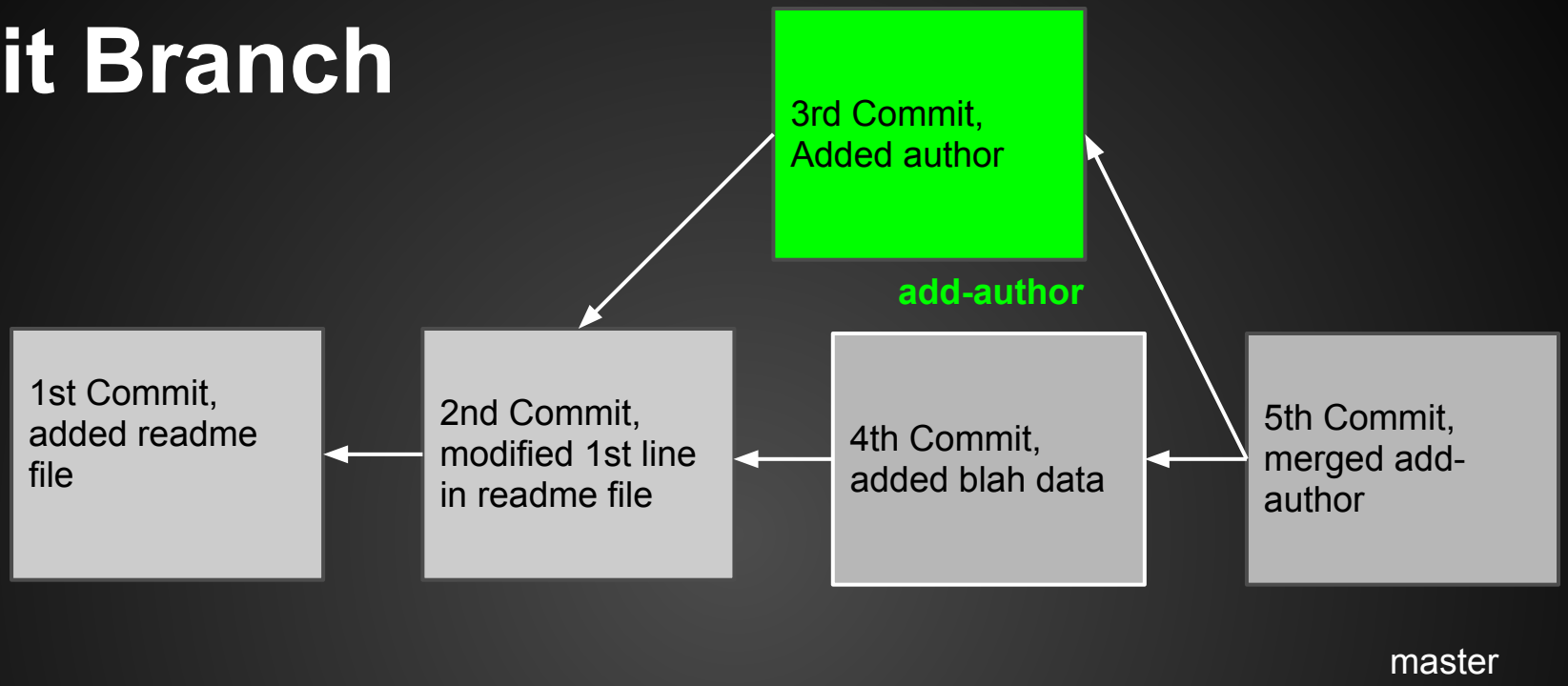

```
vim readme.txt (add blahdata)
git add .
git commit -m 'added blah data'
```

Git Branch



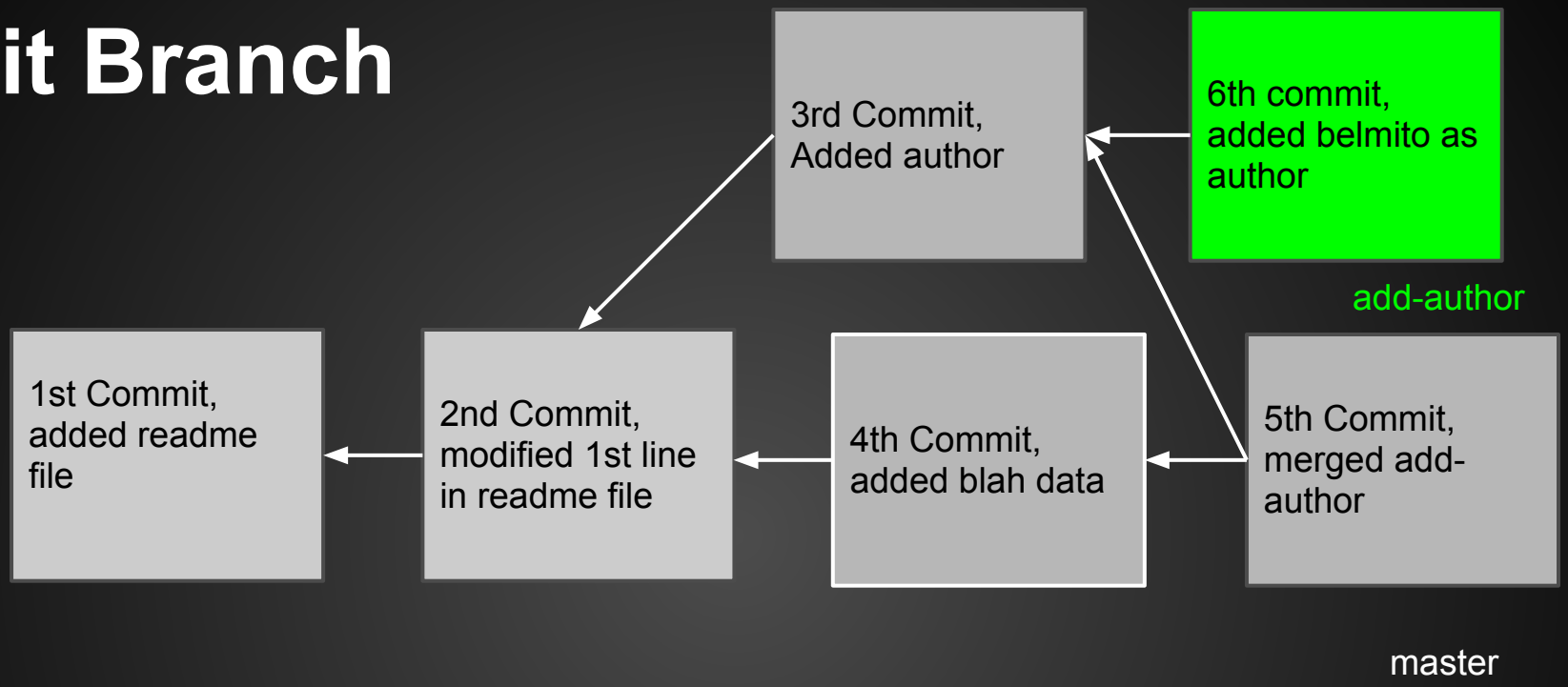
git merge add-author

Git Branch



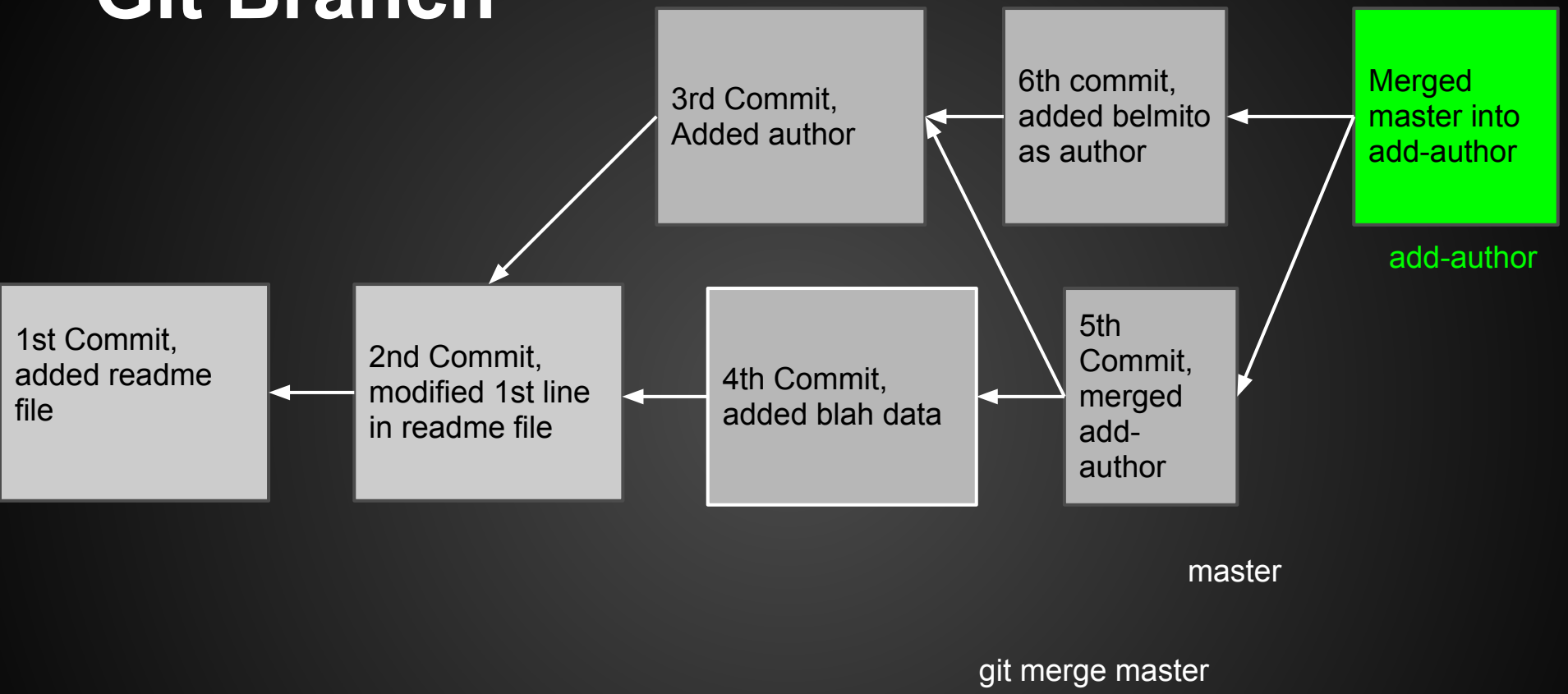
git checkout add-author

Git Branch




```
vim readme.txt (add belmito)
git add .
git commit -m 'added belmito as author'
```

Git Branch



git merge master

Best Practices

- Master branch is always production ready, DO NOT use branch unless your deploying
- Create branch name with following convention: issue#-descriptive-title like:
  - 43-Adding-Readme-File
- Make sure to pull changes from github often, commit changes often, push to github often

Usual Github Workflow

1. New Projects
   - Ask Belmer/Farsheed to Create Repository
   - Run these commands for brand new repo:
     - touch README.md
     - git init
     - git add README.md
     - git commit -m "first commit"
     - git remote add origin git@github.com:drumbi/project_name.git
     - git push -u origin master
   - Run these commands for existing files:
     - git remote add origin git@github.com:drumbi/project_name.git
     - git push -u origin master

Usual Github Worklow

1. Adding Feature/Bugfix to existing Project
    a. if you don't have copy of project
        i. git clone git@github.com:drumbi/project_name.git
    b. Checkout branch you like branch FROM
        i. git checkout branch_name
    c. Create New Branch with issue#-issue-title
        i. git checkout -b '01-descriptive-branch-title'
    d. Make your changes, stage, and commit often (repeat as necessary)
        i. git add .
        ii. git commit -m 'useful commit msg'
    e. Push and Pull Changes back to GitHub often (repeat as necessary)
        i. git push origin branch_name (pushes changes from local to github)
        ii. git pull origin other_branch_name_like_dev (note: this will merge that branch's code into yours)
    f. When code is tested and ready, go to github.com, go to project page, at top, tap Pull Request, and specify destination and source with useful description of changes

Usual Github Workfow

- master: branch is always production deployable, only used by repo maintainer
- staging: branch is deployed to staging servers and used for final testing
- dev: branch is used for active development/integration and deployed to development servers
- dev -> staging -> master

Github Overview

Demo

Getting Started

http://www.github.com -> Create Account

https://help.github.com/articles/set-up-git -> Setup Git Locally

Github does have its own native GUI program, you may use it, but it is highly recommended to use command lines first to get understanding of git

http://try.github.com -> Quick online course that teaches you commands and interactive demonstration

ANY QUESTIONS?

spark or email me at eddy@drumbi.com