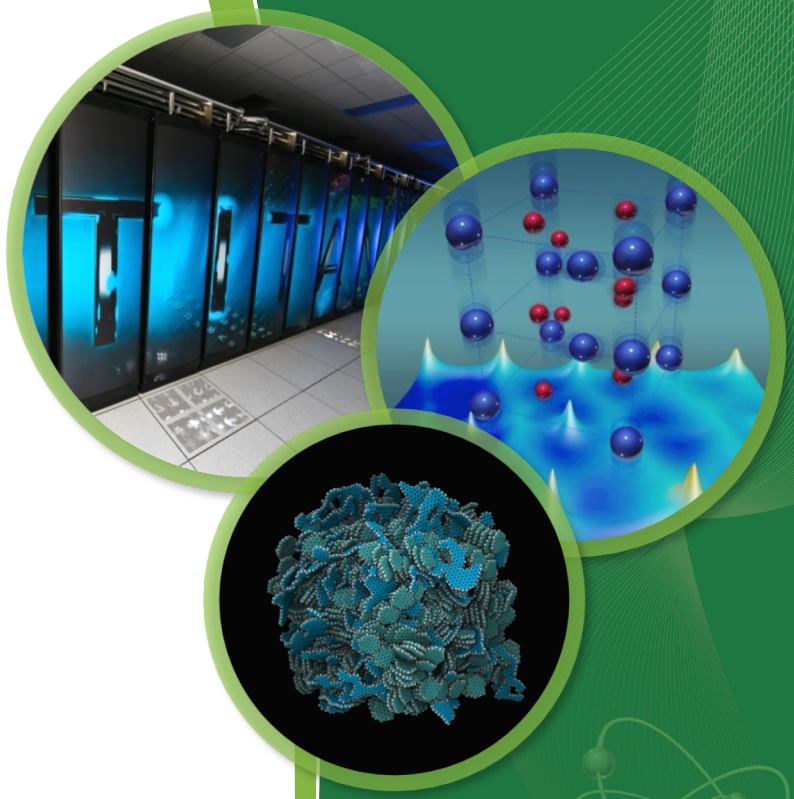
Using Git for Version Control

Jack Morrison & James Wynne

Introduction to HPC Workshop (June 27, 2018)



Agenda

- What is Git?
  (who, what, when, where, why?)
- Getting started with Git
  - Installation
  - Basic concepts
  - Using the Git CLI
- Demo collaboration using Git and “GitHosts”
- Using Git in HPC environments
- Additional Git resources

What is Git?

- Free, open-source, distributed version control system that tracks changes to files
- Ideal for multi-collaborator projects with disparate teams


1520+ contributors


1200+ contributors


280+ contributors

- Also ideal for individuals
- Fast, flexible, light-weight, safe

What is Git?

- Free, open-source, distributed version control system that tracks changes to files
- Ideal for multi-collaborator projects with disparate teams

1520+ contributors

1200+ contributors

280+ contributors

- Also ideal for individuals
- Fast, flexible, light-weight, safe

Getting Started with Git

Installation
Git Concepts
Using the CLI

Getting Started with Git

Installation
Git Concepts
Using the CLI

Installation

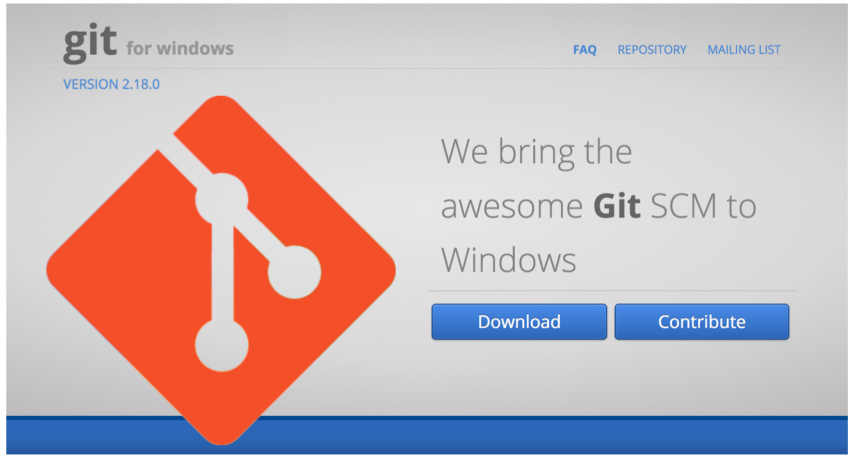
Linux

```
bash-3.2$ sudo apt install git-all
```

Mac

```
bash-3.2$ git --version
```

Windows



OLCF Systems

```
jackm@titan-ext7:~/intro_git> git --version
git version 2.13.0
jackm@titan-ext7:~/intro_git> module -t list
Currently Loaded Modulefiles:
eswrap/1.3.3-1.020200.1280.0
craype-network-gemini
pgi/18.4.0
craype/2.5.13
cray-libsci/16.11.1
udreg/2.3.2-1.0502.10518.2.17.gem
ugni/6.0-1.0502.10863.8.28.gem
pmi/5.0.12
dmapp/7.0.1-1.0502.11080.8.74.gem
gni-headers/4.0-1.0502.10859.7.8.gem
xpmem/0.1-2.0502.64982.5.3.gem
dvs/2.5_0.9.0-1.0502.2188.1.113.gem
alps/5.2.4-2.0502.9774.31.12.gem
rca/1.0.0-2.0502.60530.1.63.gem
atp/2.1.1
PrgEnv-pgi/5.2.82
cray-mpich/7.6.3
craype-interlagos
lustredu/1.4
xalt/0.7.5
git/2.13.0
module_msg/0.1
modulator/1.2.0
hsi/5.0.2.p1
DefApps
python/3.6.3
jackm@titan-ext7:~/intro_git>
```

Installation

- Configure user information
  - Metadata will be included in git history

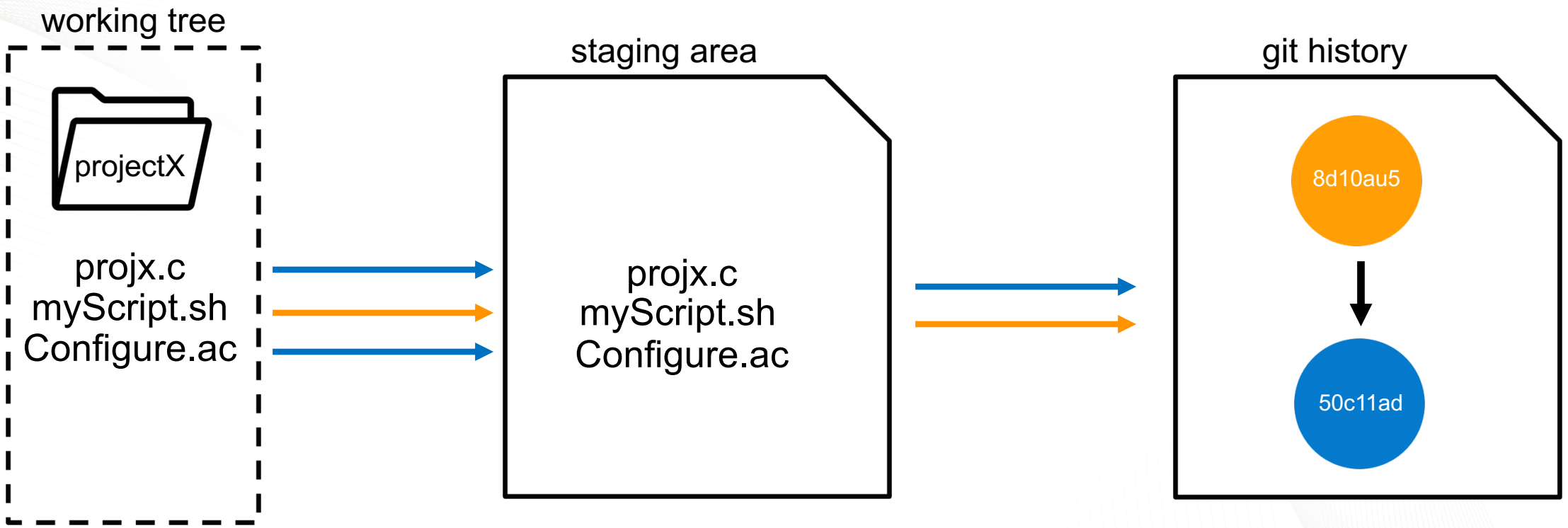
```
bash-3.2$ git config --global user.name "Jack Morrison"
bash-3.2$ git config --global user.email "morrisonjc@ornl.gov"
bash-3.2$ git config --list
credential.helper=osxkeychain
user.name=Jack Morrison
user.email=morrisonjc@ornl.gov
bash-3.2$
```

Getting Started with Git

Installation
Git Concepts
Using the CLI

A simple git workflow

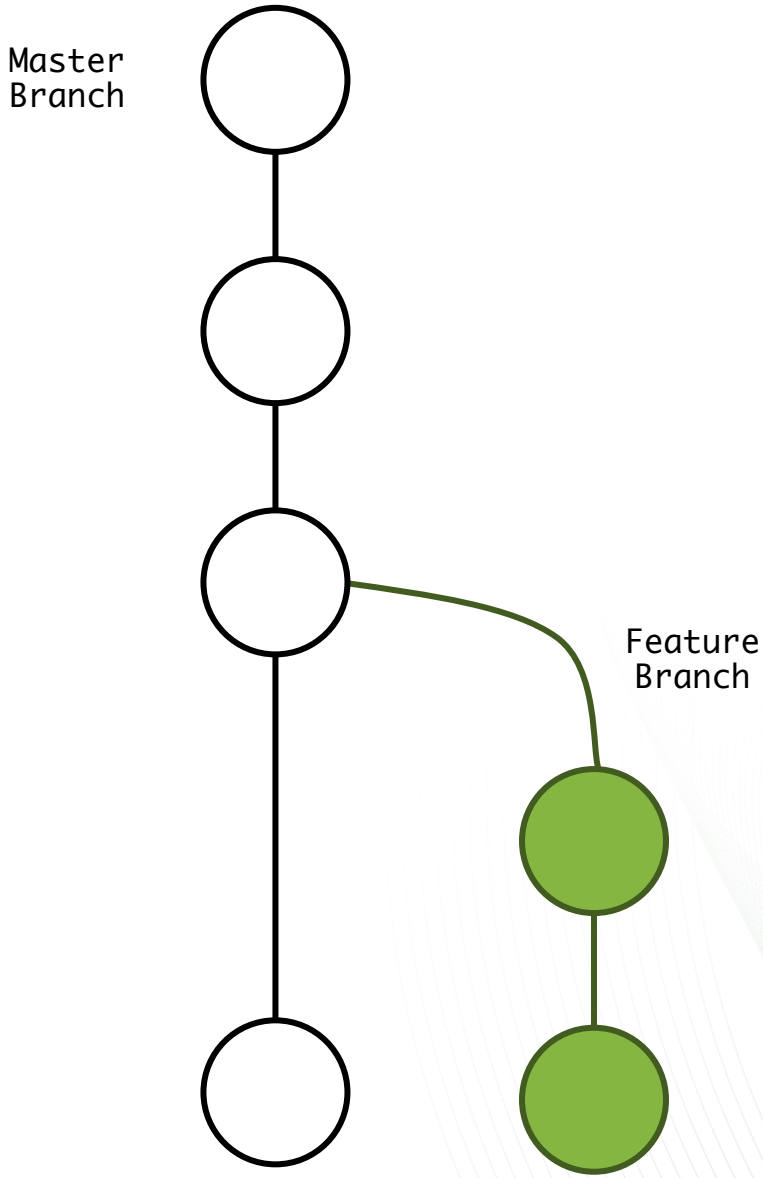
local



Folder by Arijit Adak from the Noun Project

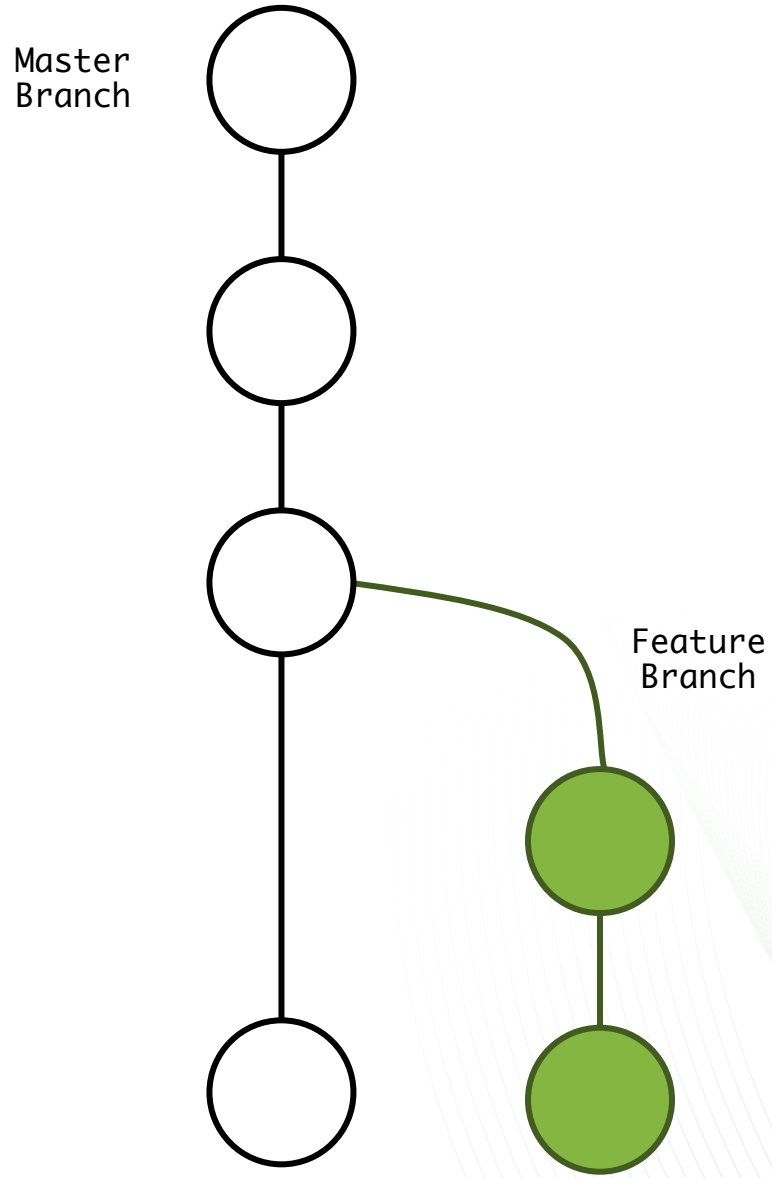
What is a commit?

- Better question: What is a commit like?
  - Savepoint
  - Snapshot
  - Backup
- What are commits for?
  - Capture the smallest meaningful addition of new features
  - Recover working versions of the code
  - Build a historical narrative of your work
- What do commits contain?
  - Each commit is identified uniquely by a SHA-1 hash
    (40 digits hexadecimal, generally abbreviated to the first 7 characters)
  - Deltas/diffs of changes made
  - Meaningful commit message
  - Pointer to parent commit



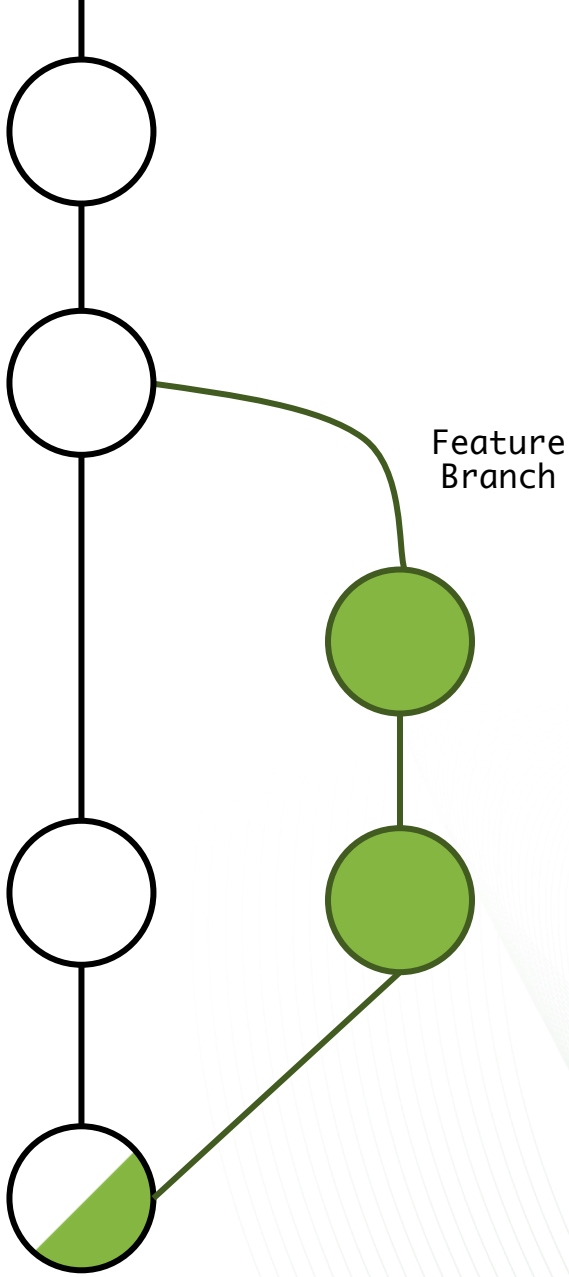
What is a branch?

- Better question: How does branching help me work?
  - Isolated, alternative timeline for evaluating hypotheses
  - A branch addresses an idea/experiment/issue
  - Branches are disposable
- References to commits
- Master branches vs. feature branches
  - “authoritative and stable” vs. “under development”



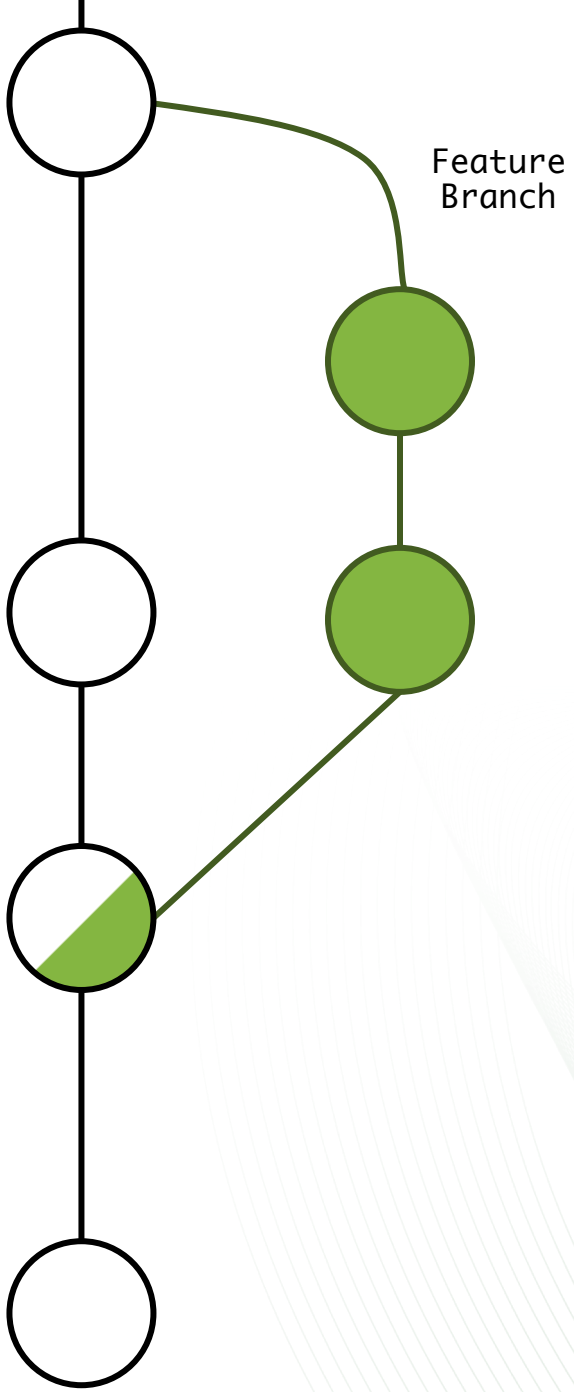
Merging

- Incorporating new features to known-good code
  - Generally between 2 branches
- If multiple branches make changes to the same pieces of code, “merge conflicts” can arise, and will require resolving.
- New commit will include changes made on both source branches
- Merge often and stay away from long-running branches!



Merging

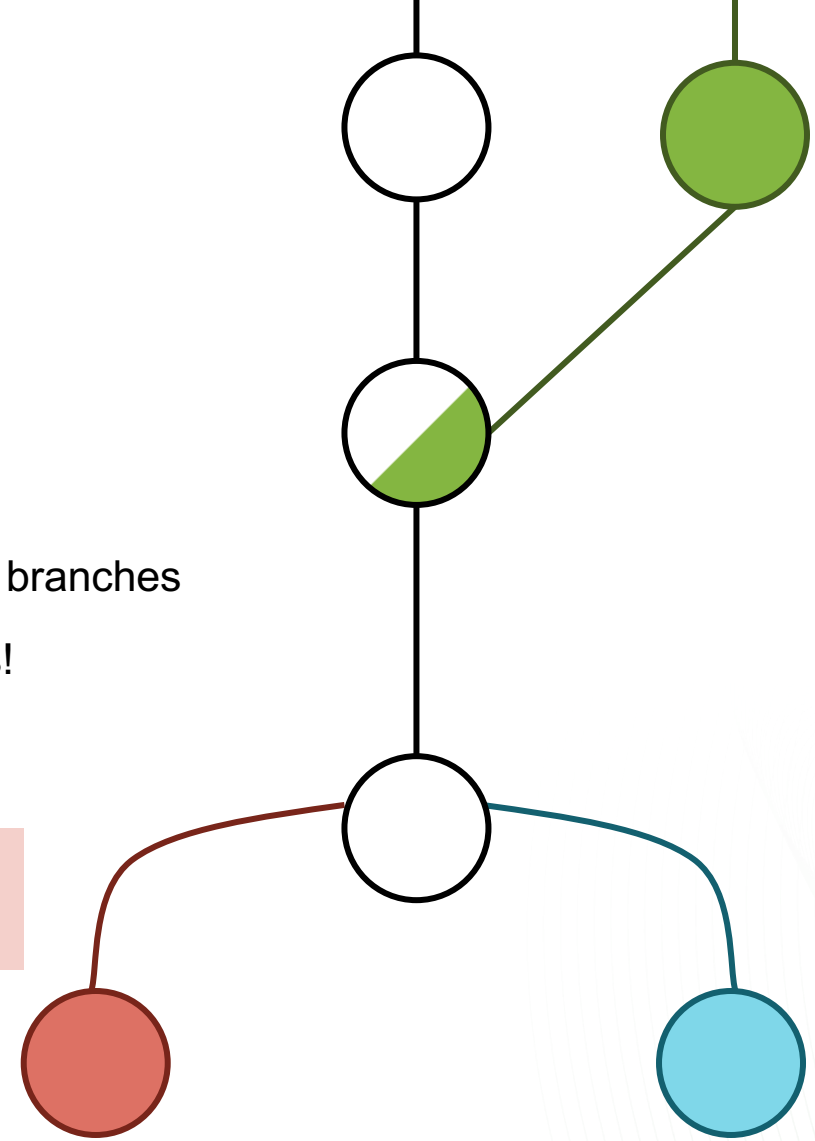
- Incorporating new features to known-good code
  - Generally between 2 branches
- If multiple branches make changes to the same pieces of code, “merge conflicts” can arise, and will require resolving.
- New commit will include changes made on both source branches
- Merge often and stay away from long-running branches!



Merging

- Incorporating new features to known-good code
  - Generally between 2 branches
- If multiple branches make changes to the same pieces of code, “merge conflicts” can arise, and will require resolving.
- New commit will include changes made on both source branches
- Merge often and stay away from long-running branches!

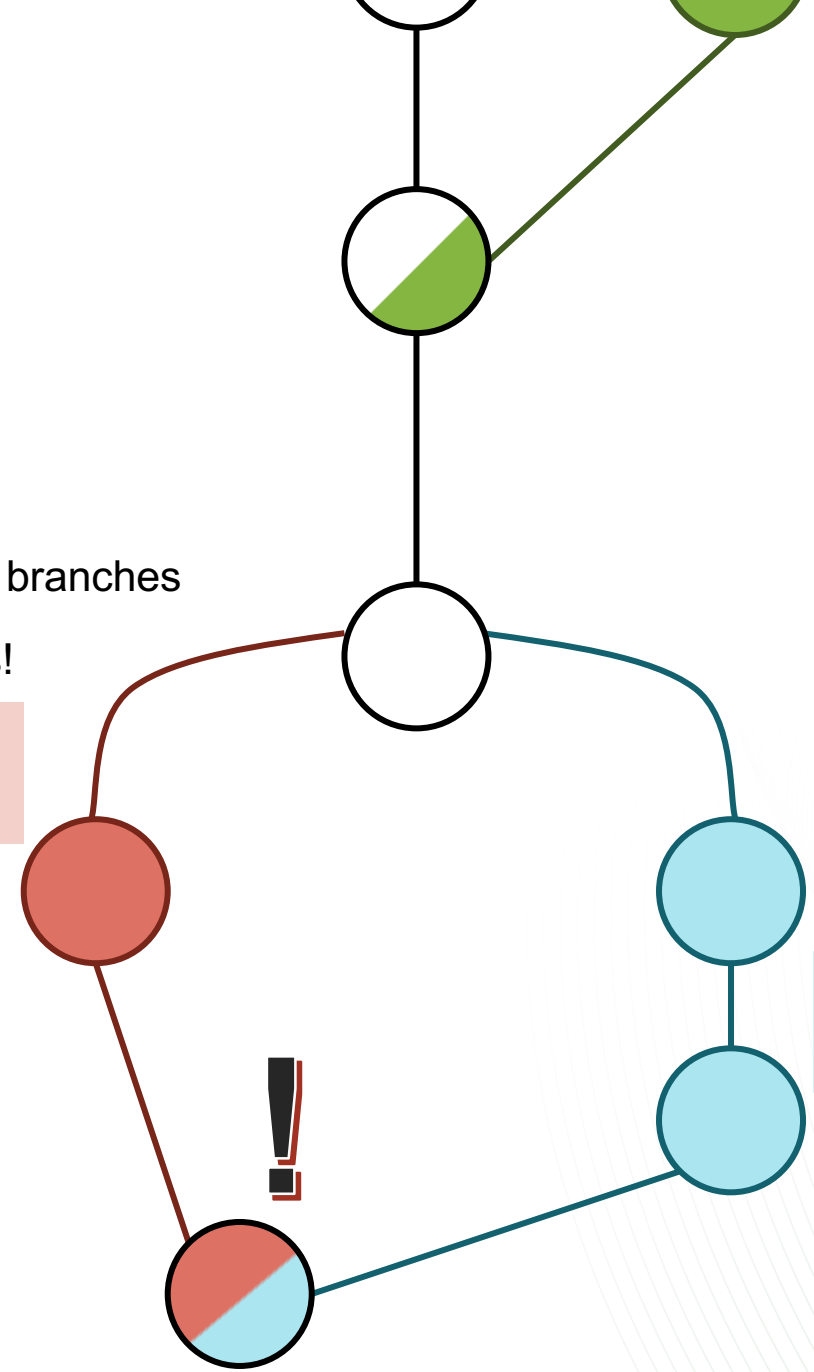
```
...
double niter = 200000;
...
```



Merging

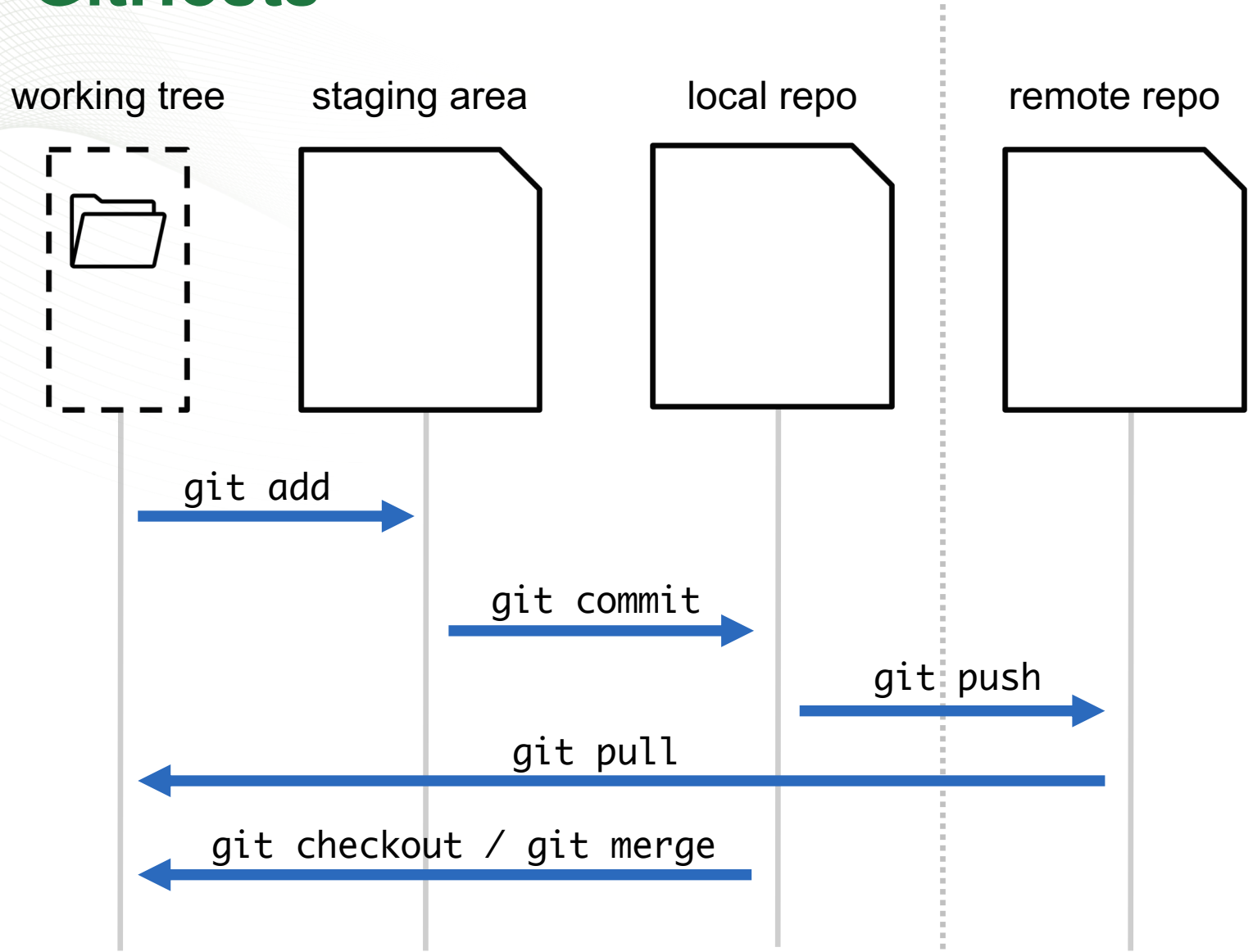
- Incorporating new features to known-good code
  - Generally between 2 branches
- If multiple branches make changes to the same pieces of code, “merge conflicts” can arise, and will require resolving.
- New commit will include changes made on both source branches
- Merge often and stay away from long-running branches!

```
...
double niter = 200000;
...
```




```
...
double niter = 500000;
...
```

“GitHosts”



GitLab

GitHub

Bitbucket

Getting Started with Git

Installation
Git Concepts
Using the CLI

Git CLI

Command	Description
$ git init <directory>	Initialize an empty git repository in <directory>
$ git status	List staged files, unstaged files, and untracked files
$ git add / rm <file/dir>	Add files or directories to the staging index
$ git commit –m “<message>”	Commit the staged changes, and use the commit message <message>
$ git merge <branch>	Merge <branch> into current branch
$ git checkout –b <branch_name>	Checkout a new branch, <branch_name>
$ git log	List the entire git history

Using Git for Version Control

Jack Morrison & James Wynne

Introduction to HPC Workshop (June 27, 2018)