Master in High Performance Computing

An introduction to modern tools for collaborative science

Lecture 2 -Advanced topics in GIT

Luca Heltai <luca.heltai@sissa.it>

Mathematical Analysis, Modeling, and Applications (math.sissa.it)
Theoretical and Scientific Data Science (datascience.sissa.it)

sissa mathLab innovating with mathematics

The Abdus Salam International Centre for Theoretical Physics

SISSA

MHPC
Master in High Performance Computing

Branching & Merging

Isolate Experiments

Isolate Work Units

Parallelize

Long Running Topics

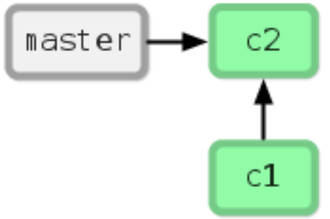
Hot Fix

MHPC
Master in High Performance Computing

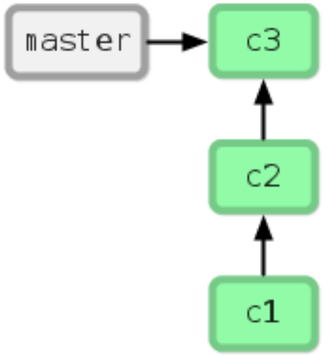
Merge vs. Rebase

Merge vs. Rebase

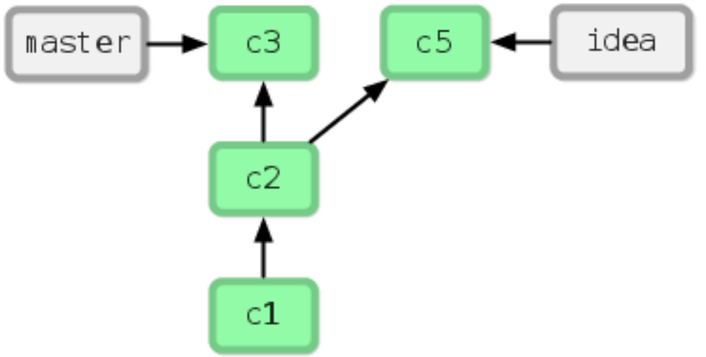
Merge vs. Rebase



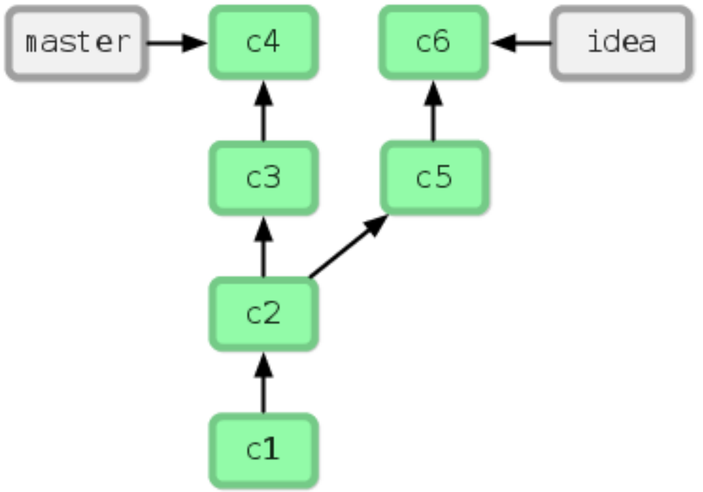
Merge vs. Rebase



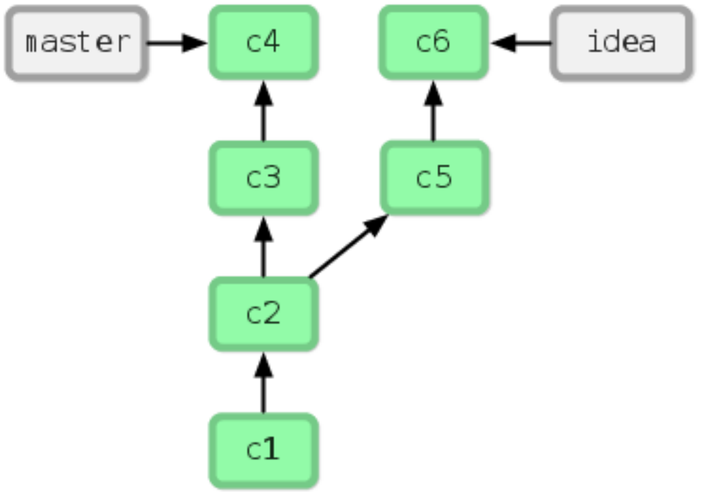
Merge vs. Rebase



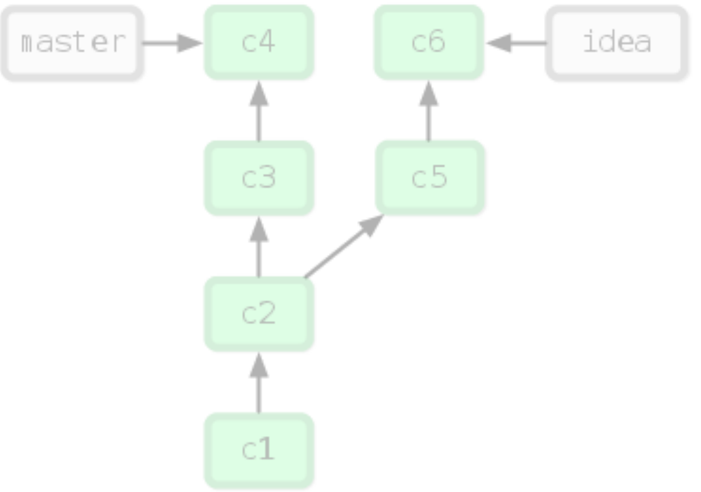
Merge vs. Rebase



Merge vs. Rebase

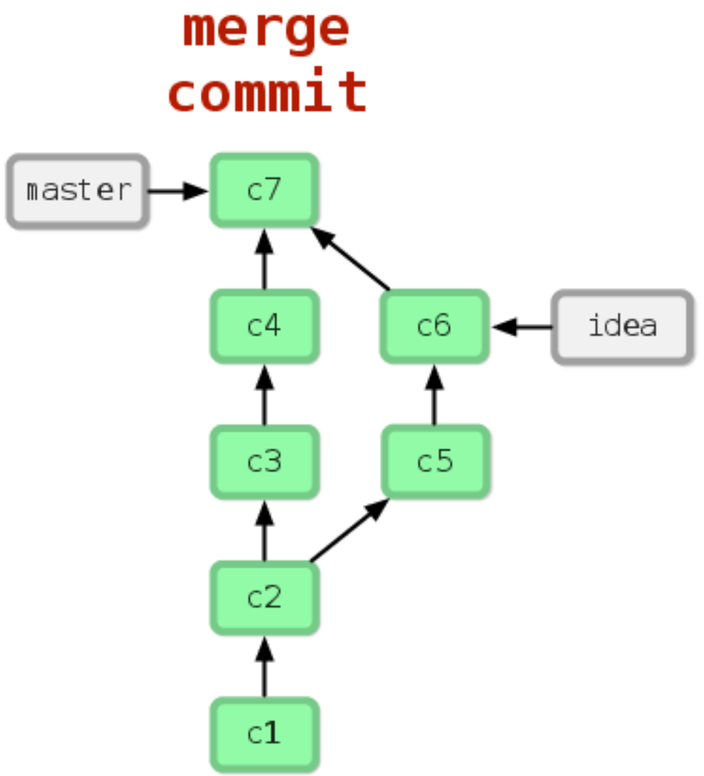


merge

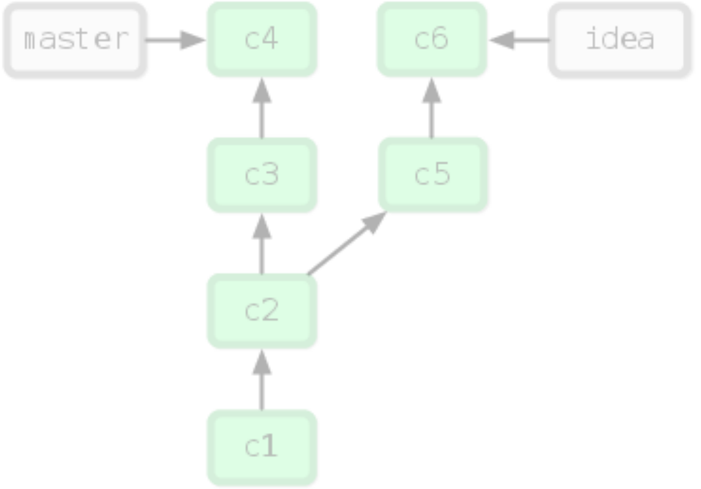


rebase

Merge vs. Rebase

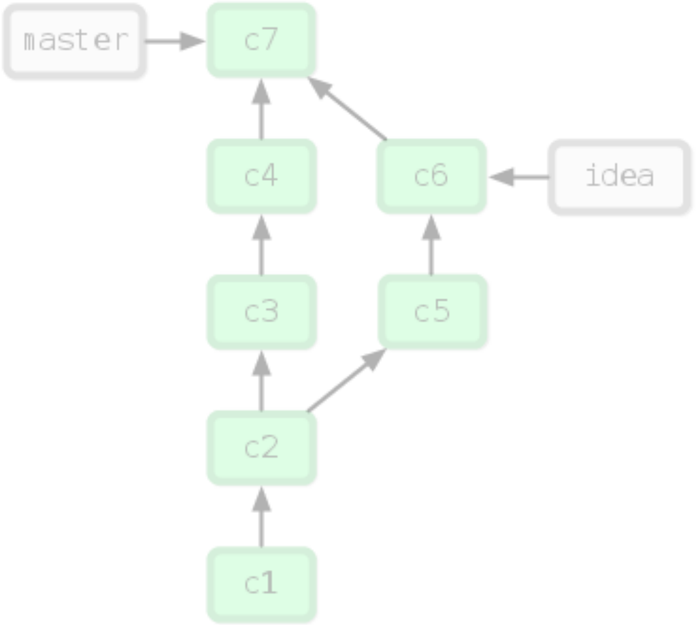


merge

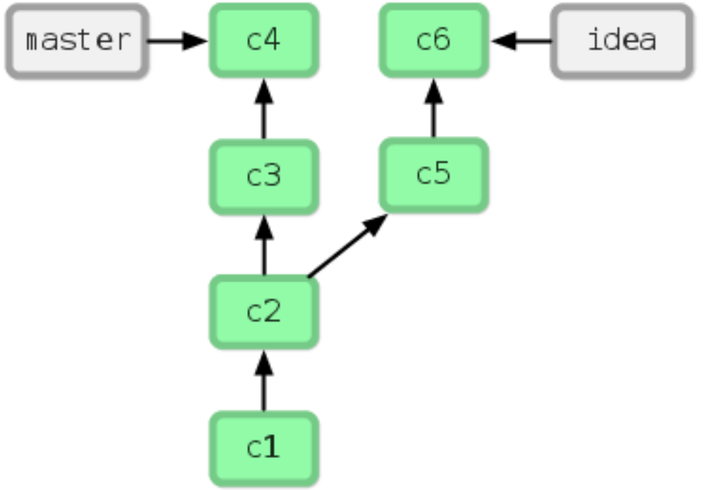


rebase

Merge vs. Rebase

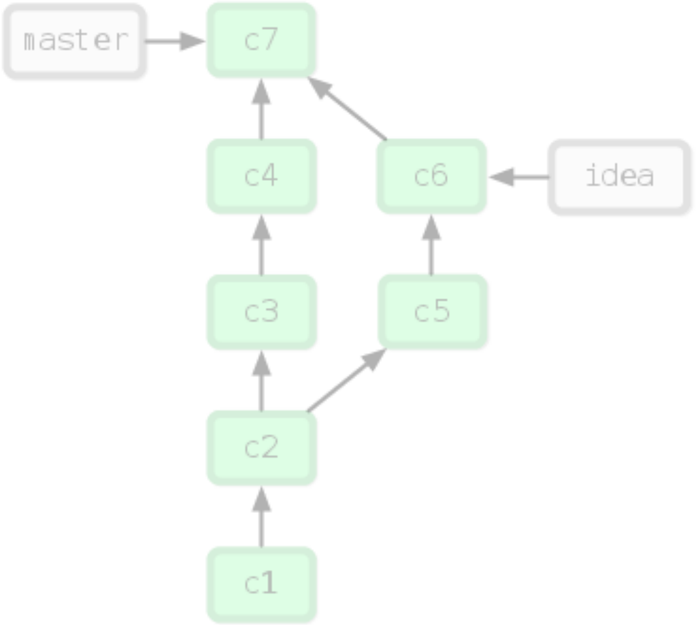


merge

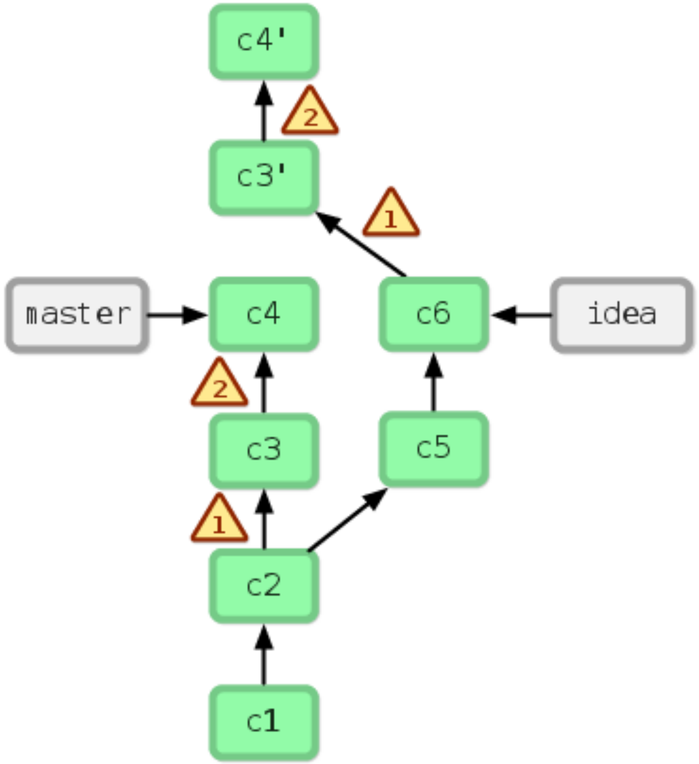


rebase

Merge vs. Rebase

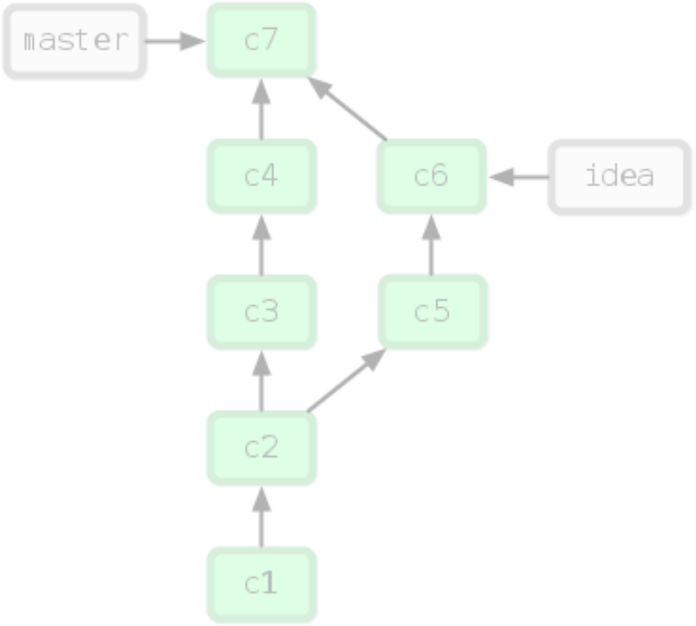


merge

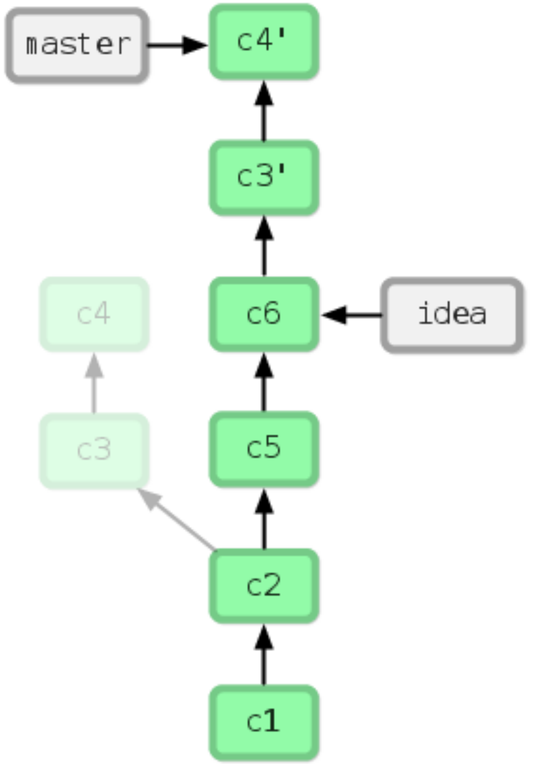


rebase

Merge vs. Rebase

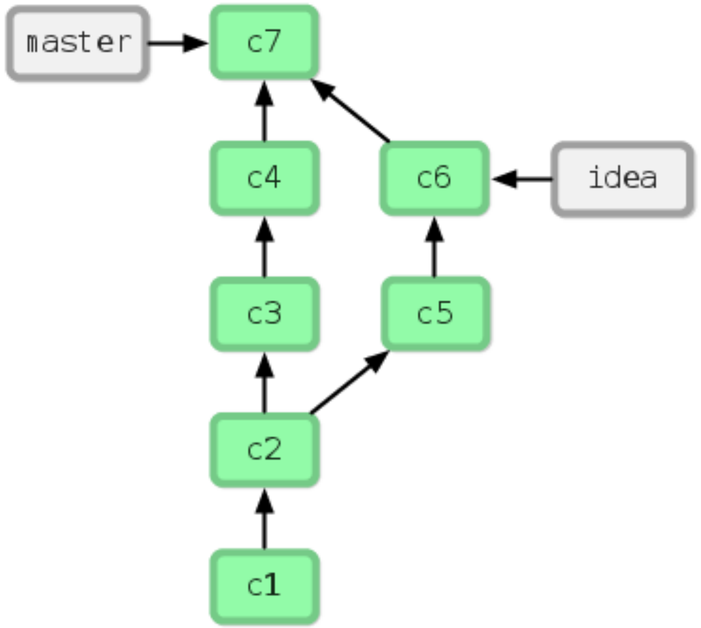


merge

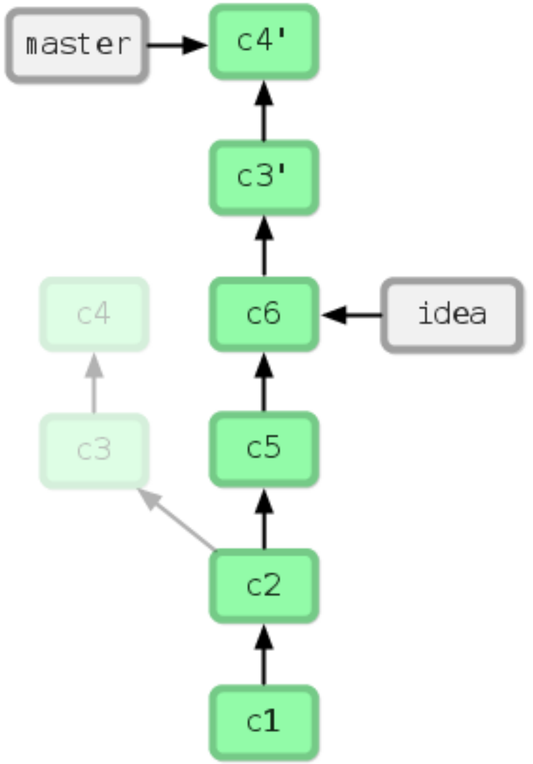


rebase

Merge vs. Rebase



merge



rebase

MHPC
Master in High Performance Computing

Remotes

Remotes

```
remote == URL
```

MHPC
Master in High Performance Computing

Protocols

ssh://

http[s]://

git://

file://

rsync://

ftp://

MHPC
Master in High Performance Computing

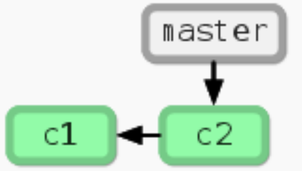
Protocols

push	ssh://	pull
	http[s]://	pull
	git://	pull
push	file://	pull
	~~rsync://~~	
	~~ftp://~~	

Ann's Computer

[redacted]

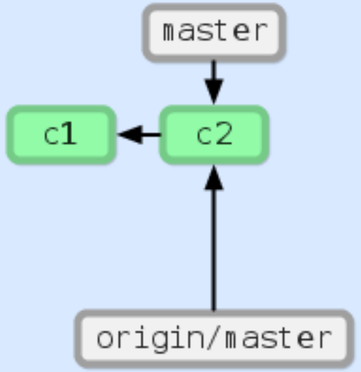
Company Server



Bob's Computer

[redacted]

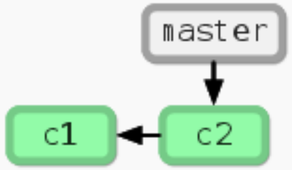
Ann's Computer

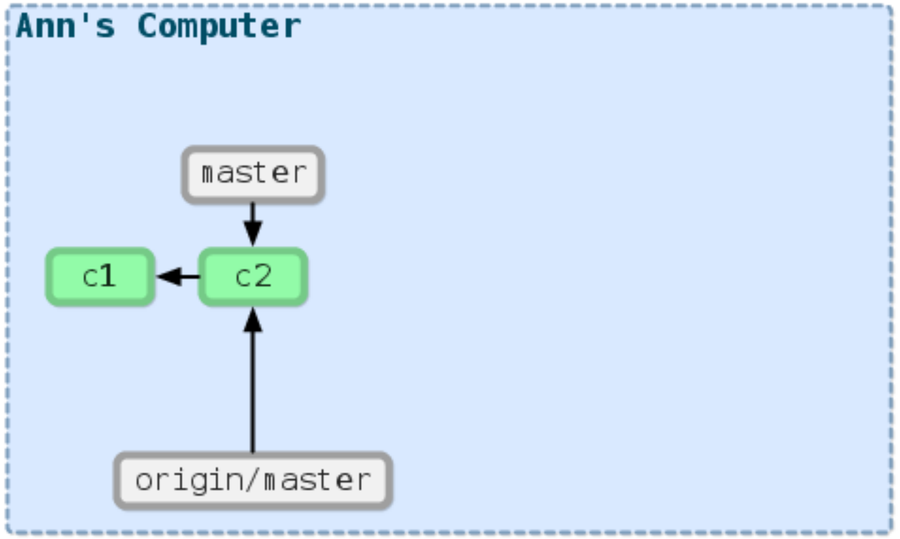


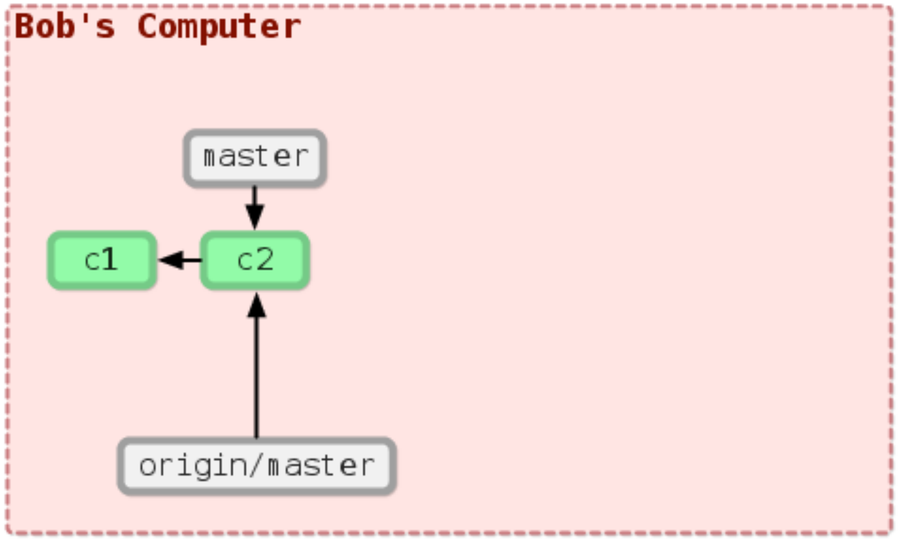
Bob's Computer

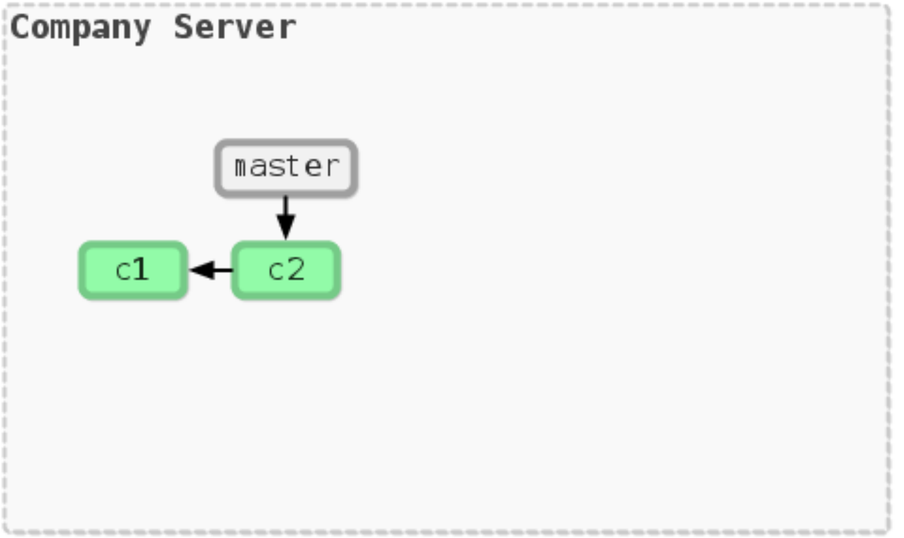
```
git clone ann@git.company.com:project.git
```

Company Server



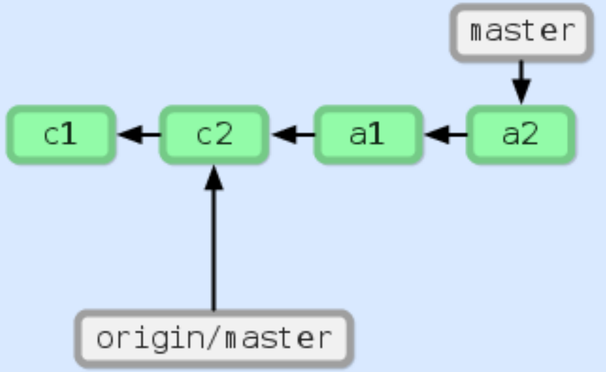






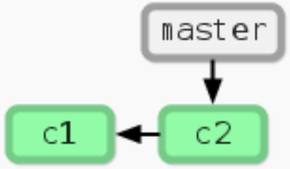

```
git clone bob@git.company.com:project.git
```

Ann's Computer

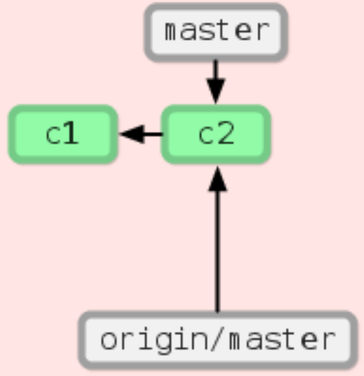



```
git commit -a -m "Ann's new feature"
```

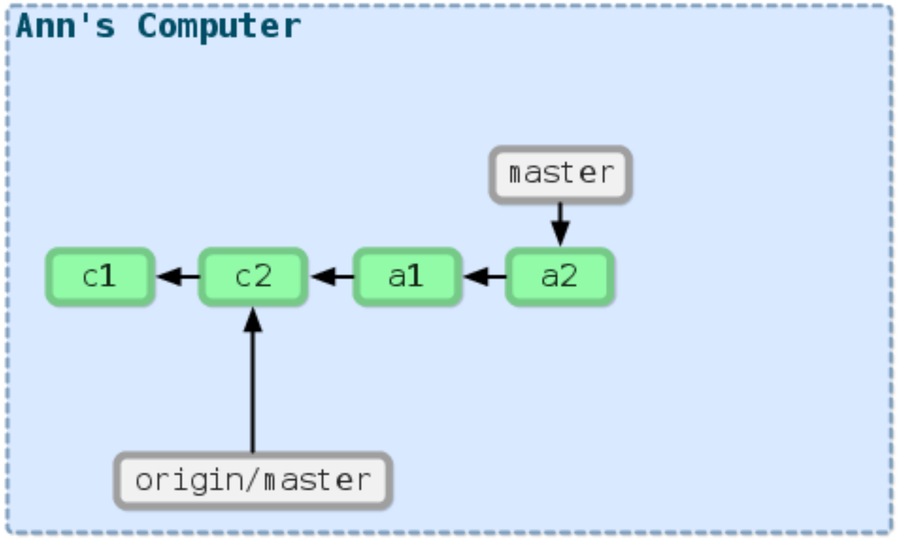
Company Server

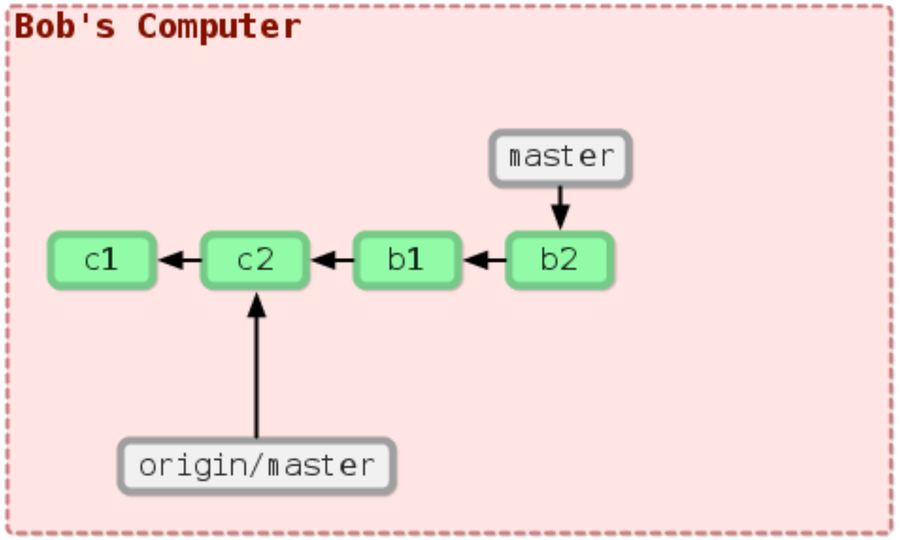


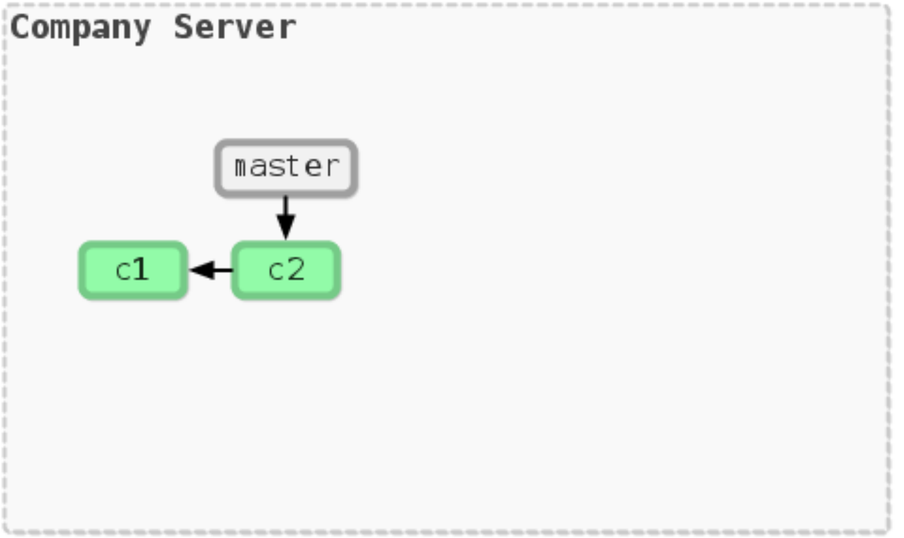
Bob's Computer

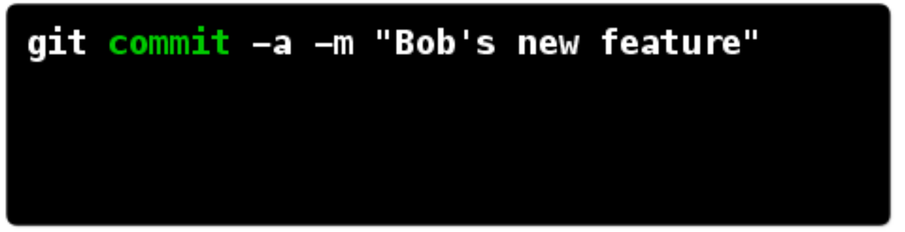


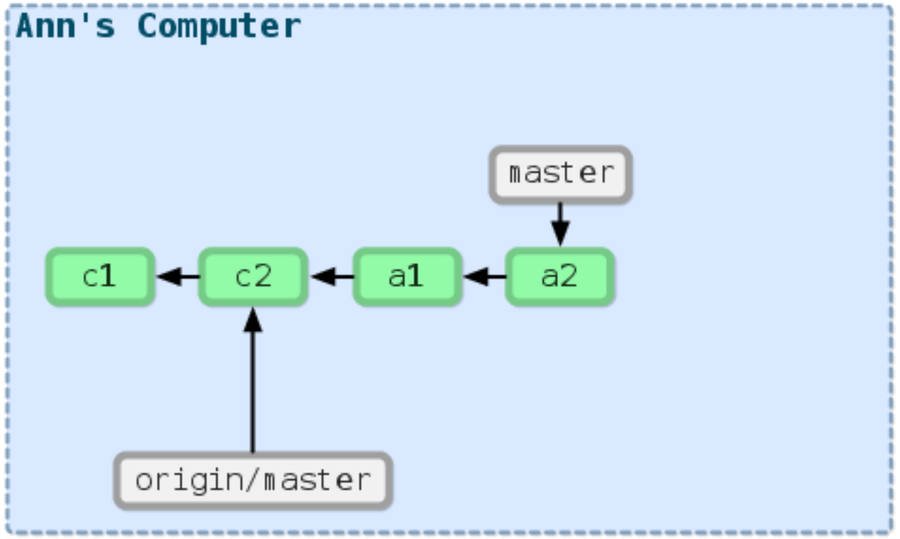

```
```

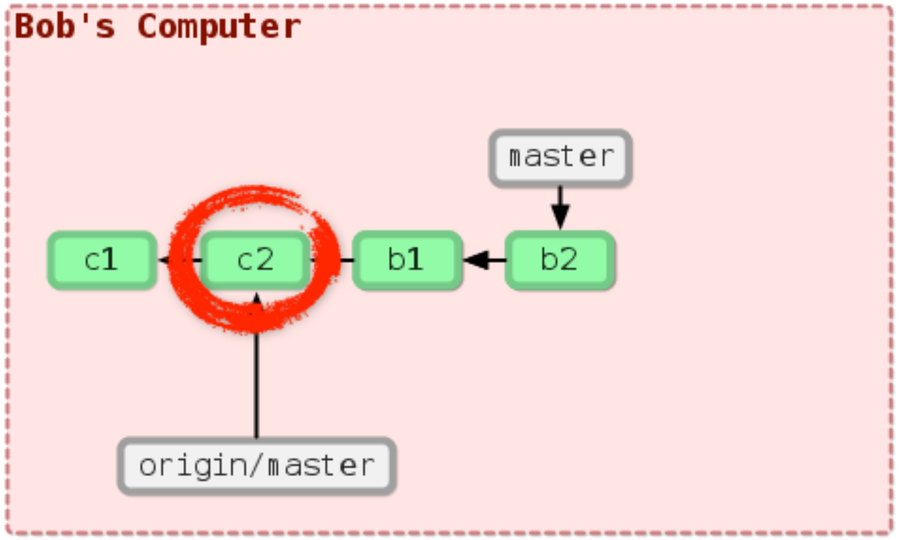

Ann's Computer
master
c1
c2
a1
a2
origin/master

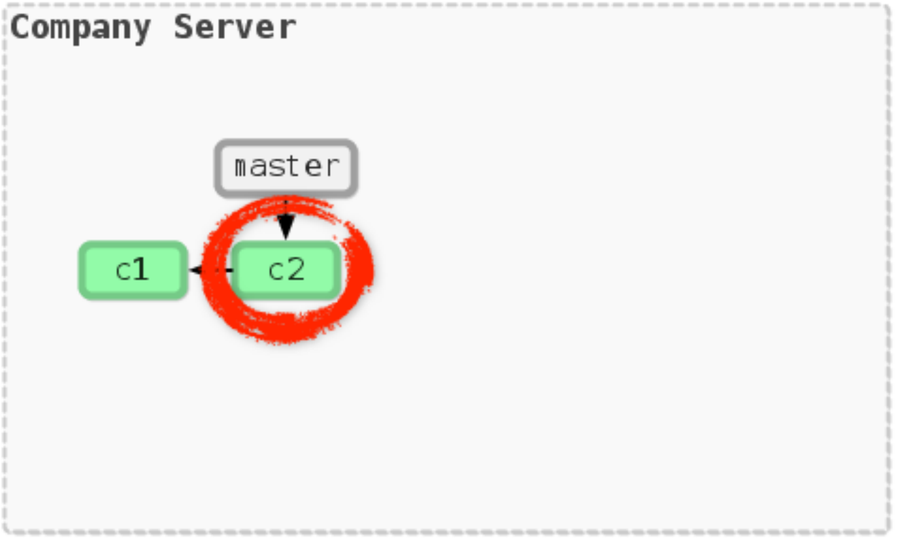

Bob's Computer
master
c1
c2
b1
b2
origin/master


Company Server
master
c1
c2

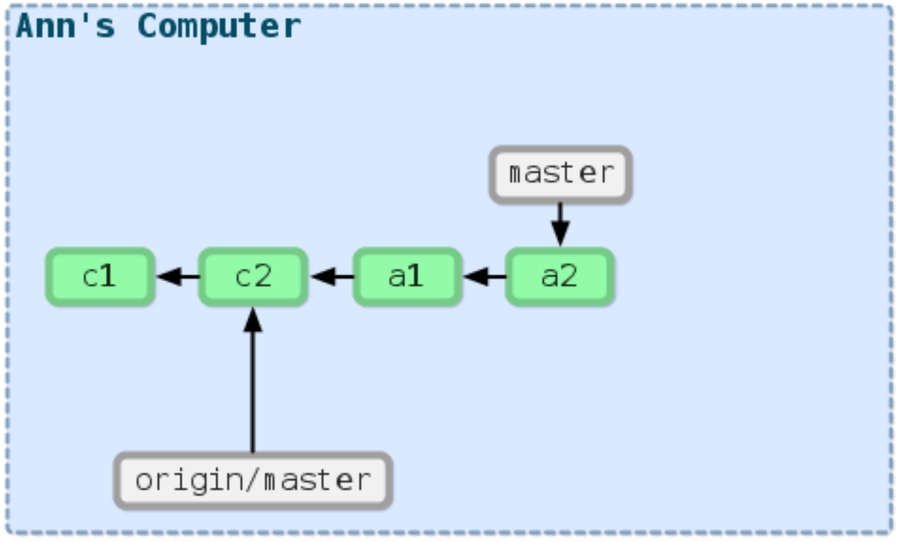

git commit -a -m "Bob's new feature"

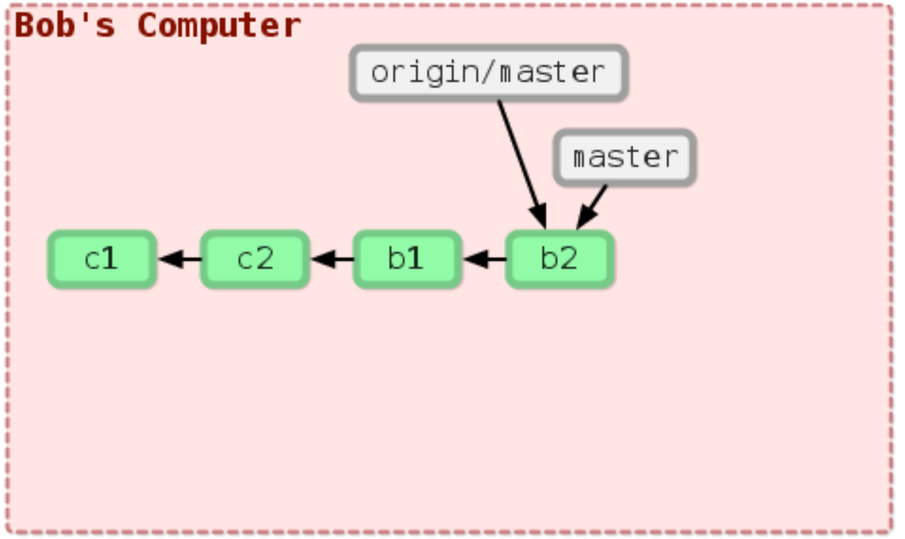

Ann's Computer
master
c1
c2
a1
a2
origin/master

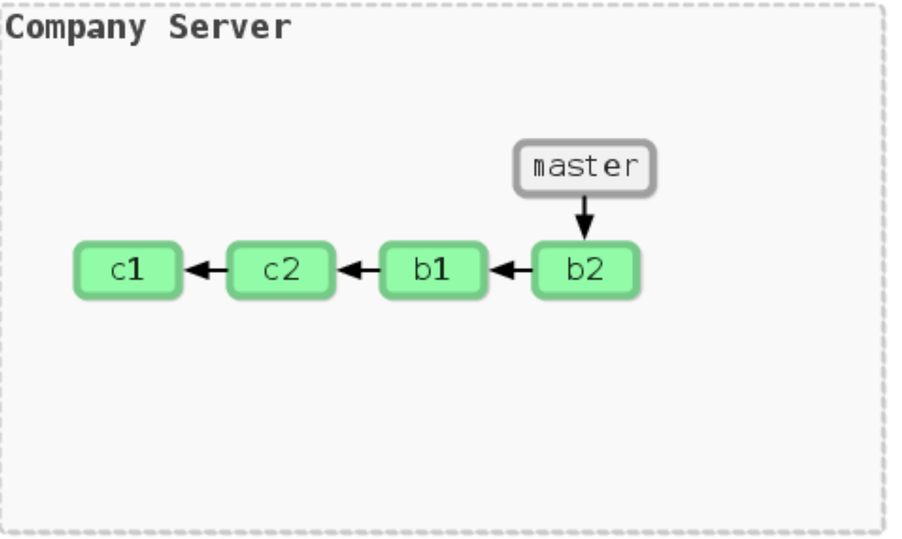

Bob's Computer
master
c1
c2
b1
b2
origin/master


Company Server
master
c1
c2

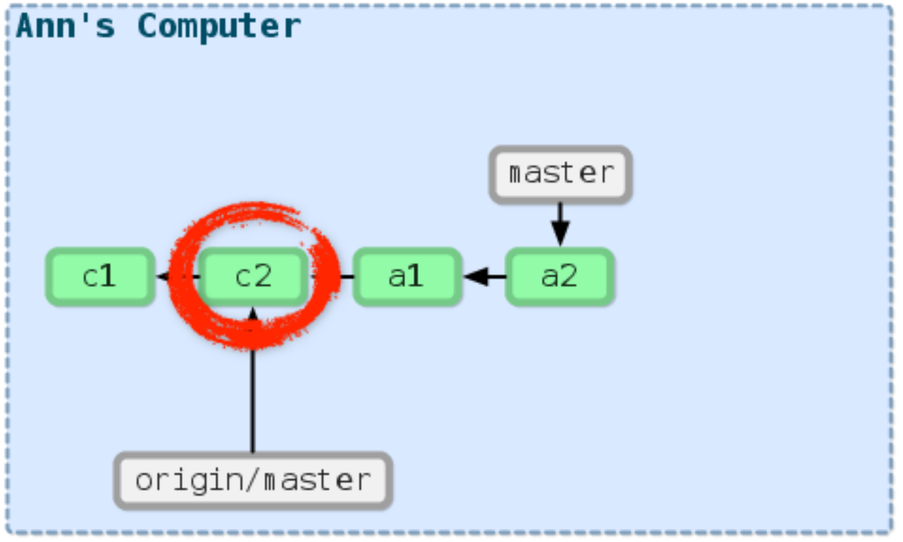

git push origin master


Ann's Computer
master
c1
c2
a1
a2
origin/master

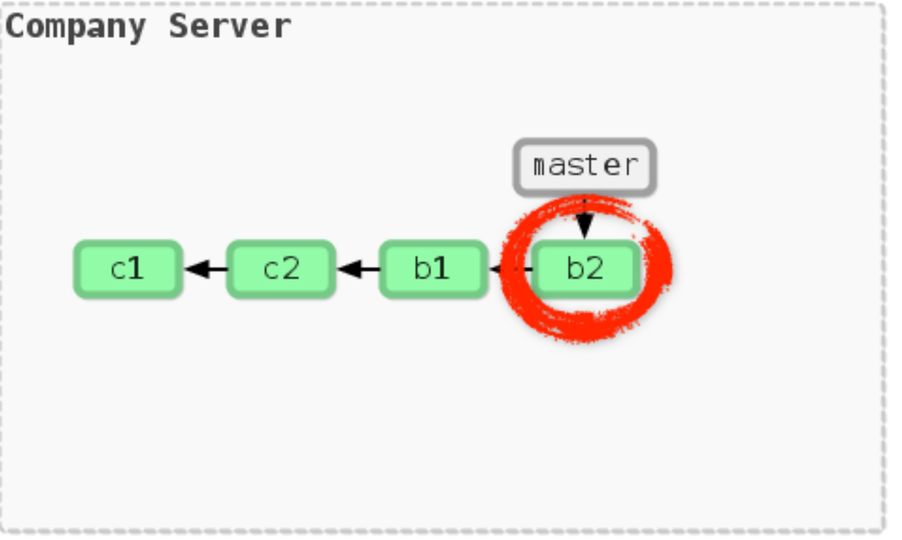

Bob's Computer
origin/master
master
c1
c2
b1
b2

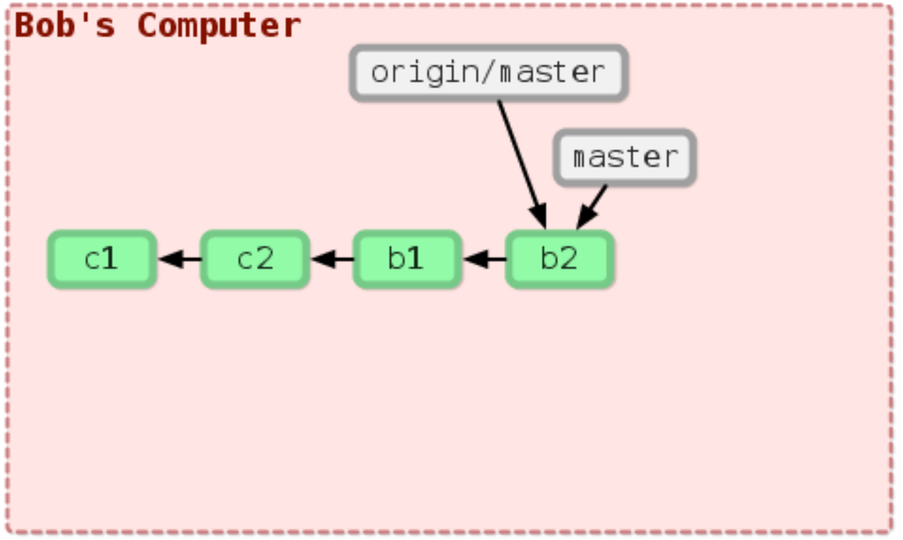

Company Server
master
c1
c2
b1
b2


git push origin master

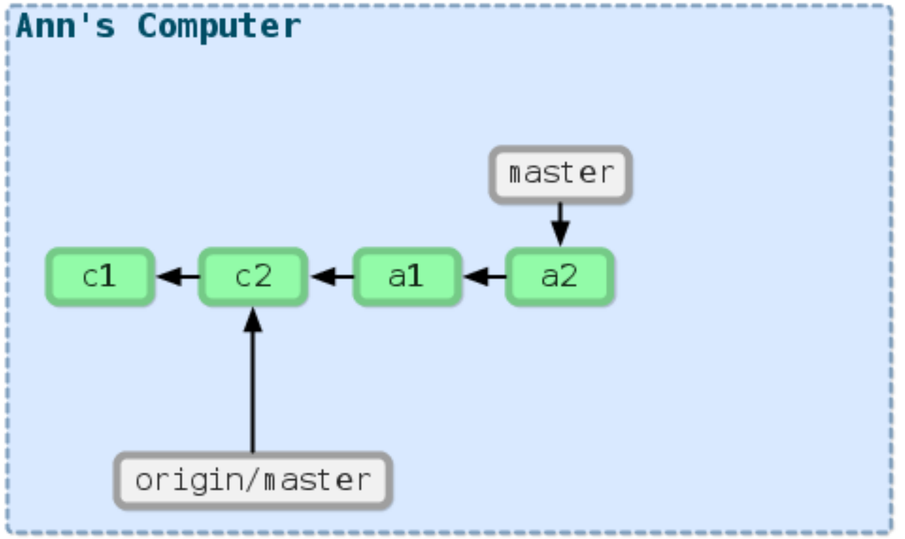



```
git push origin master
```

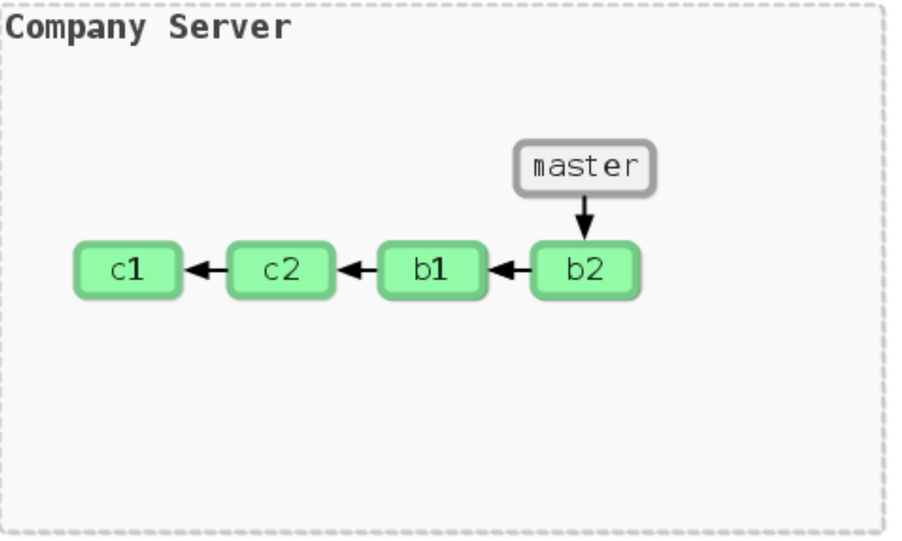


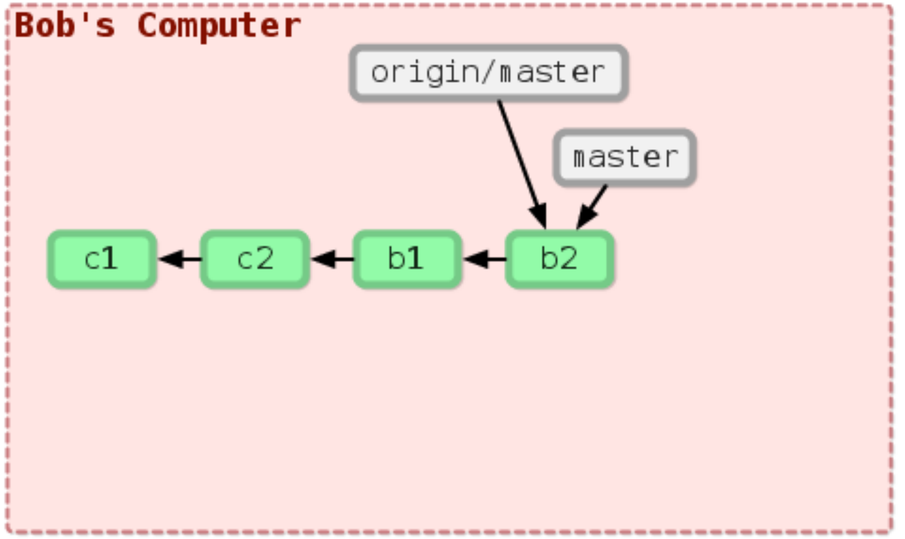



```
```

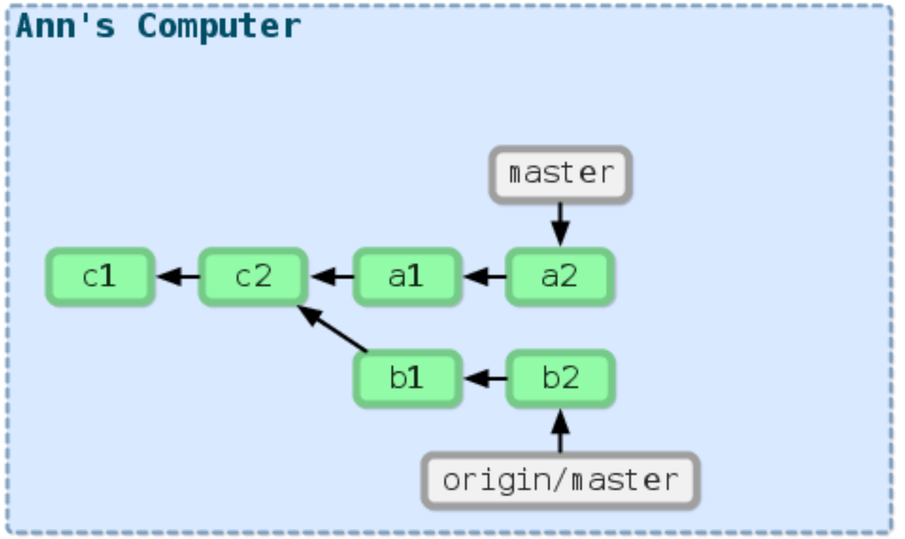

Ann's Computer
master
c1
c2
a1
a2
origin/master

```
git fetch
```

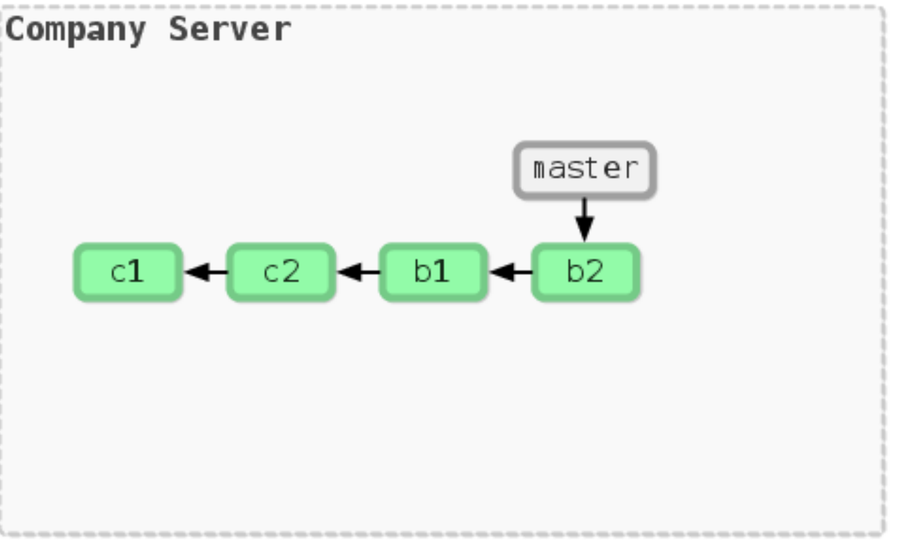

Company Server
master
c1
c2
b1
b2

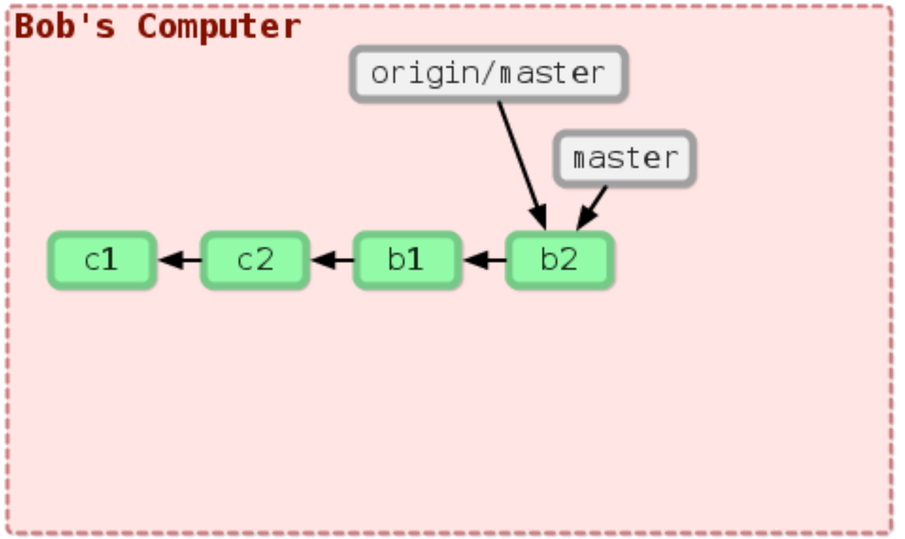

Bob's Computer
origin/master
master
c1
c2
b1
b2

```
```

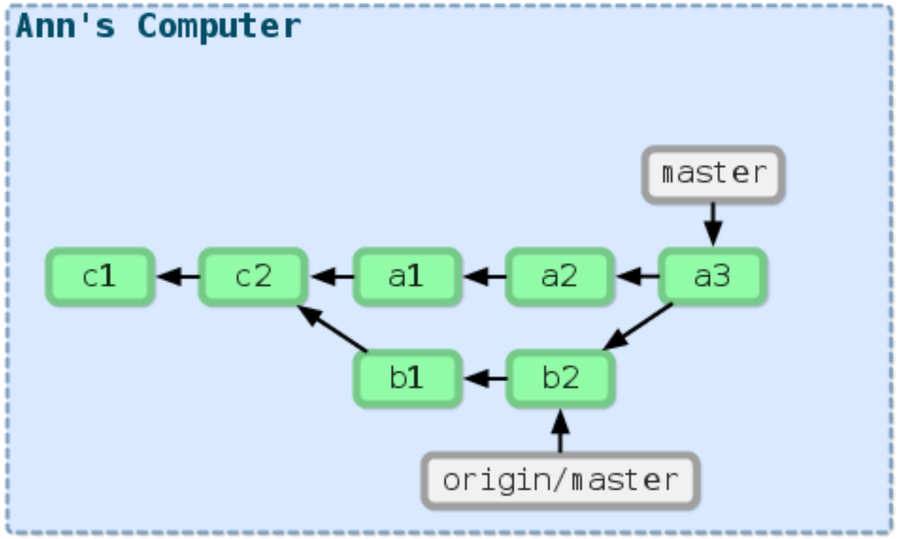

Ann's Computer
master
c1
c2
a1
a2
b1
b2
origin/master

```
git fetch
```

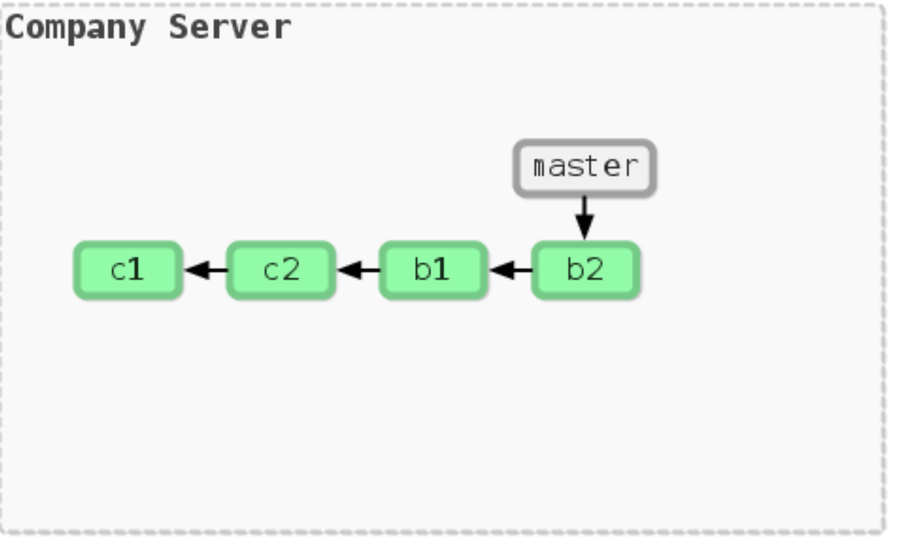

Company Server
master
c1
c2
b1
b2

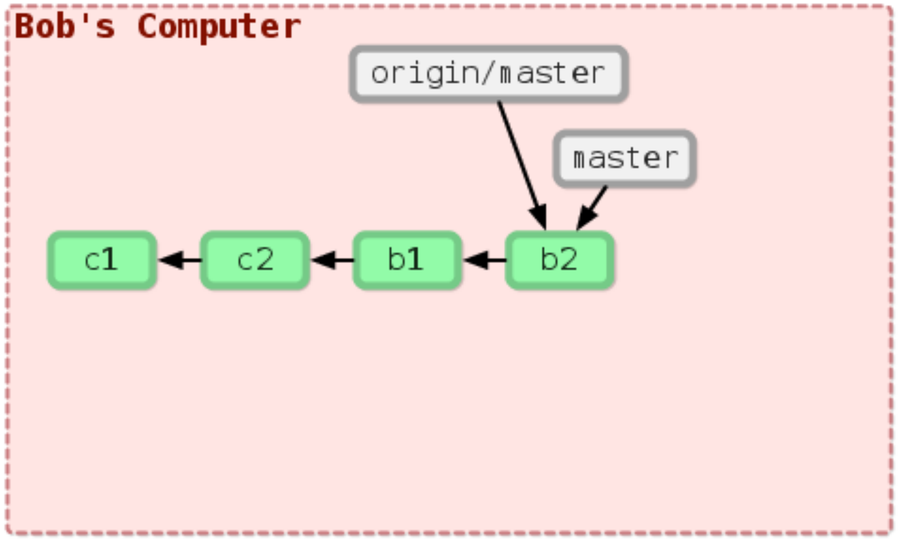

Bob's Computer
origin/master
master
c1
c2
b1
b2

```
```

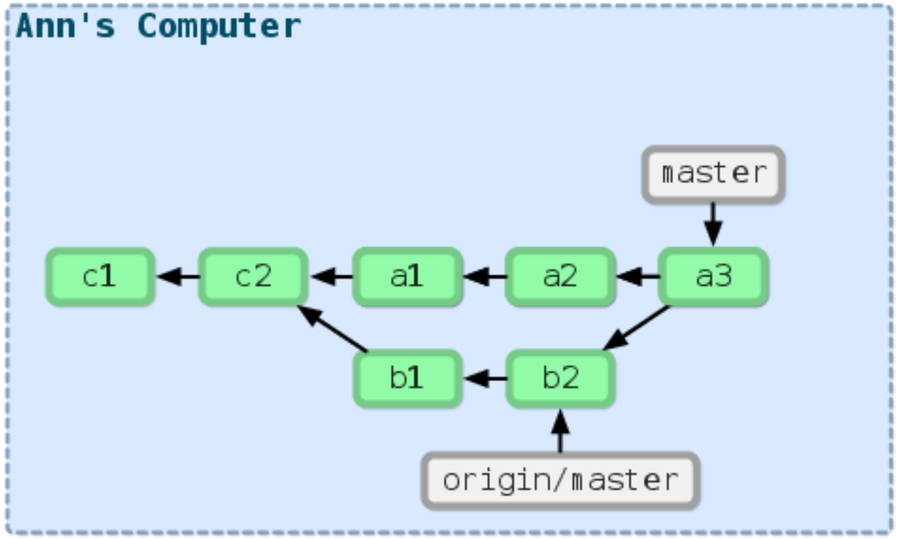



```
git merge origin/master
```

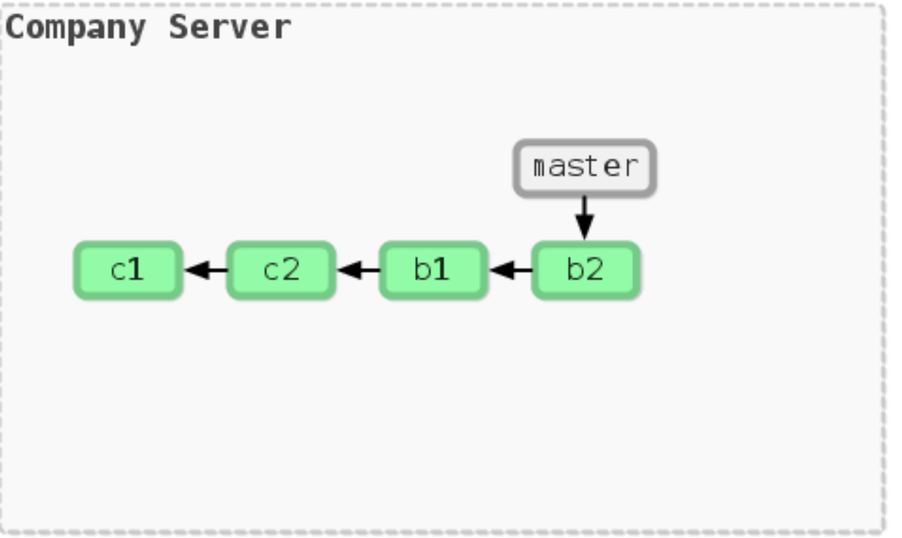


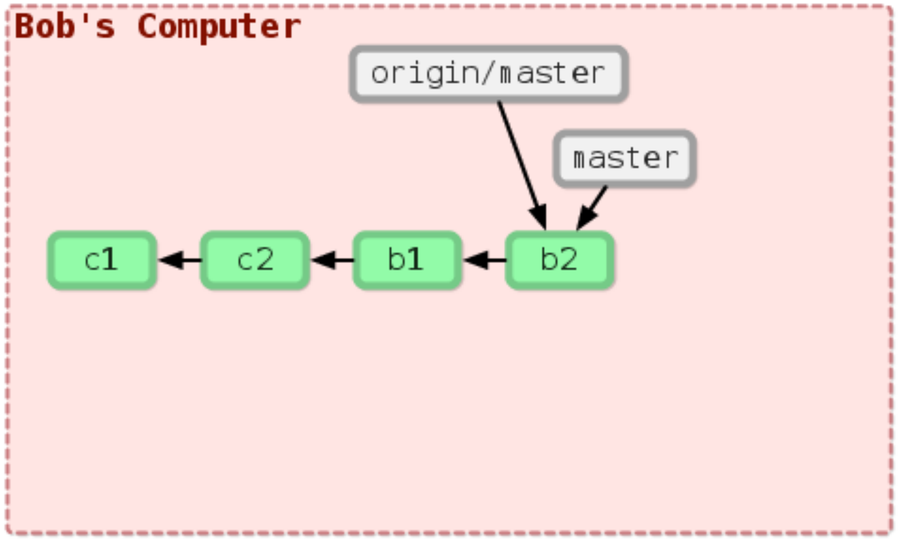



```
```

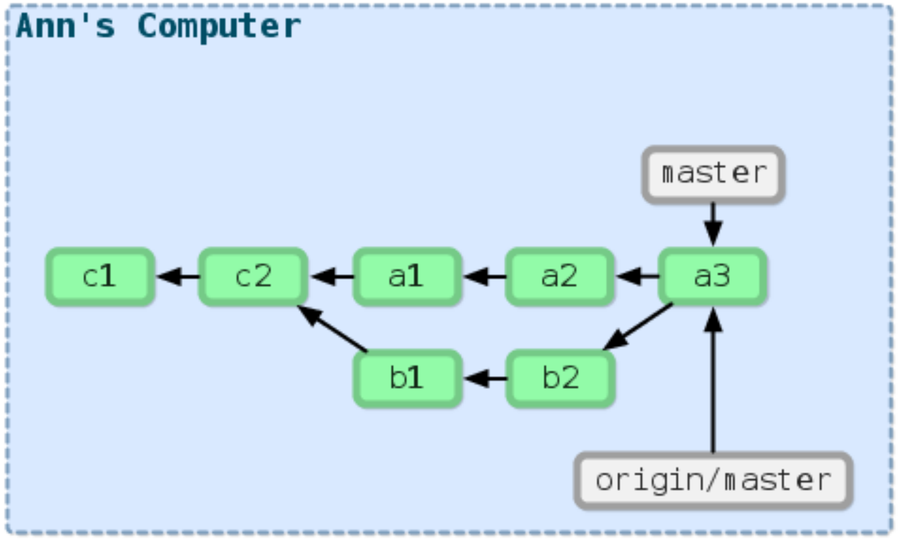



```
git push origin master
```

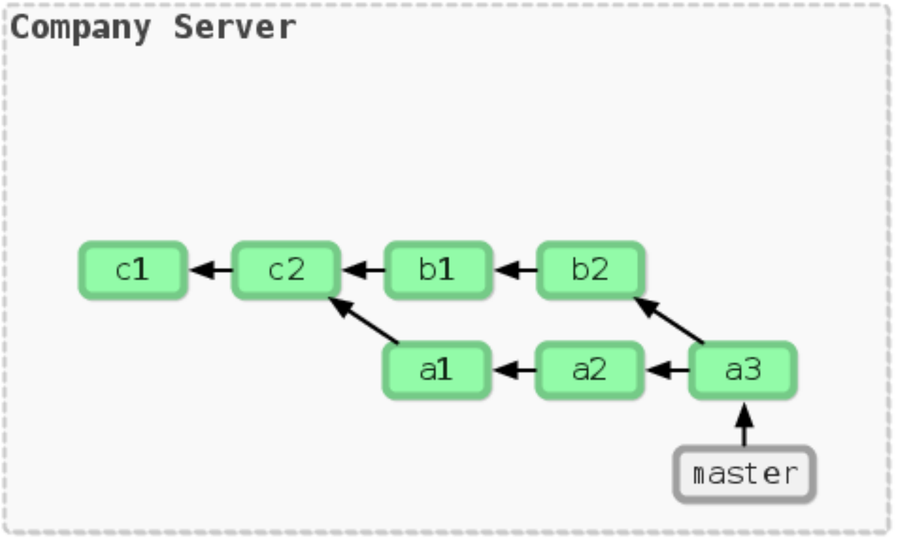


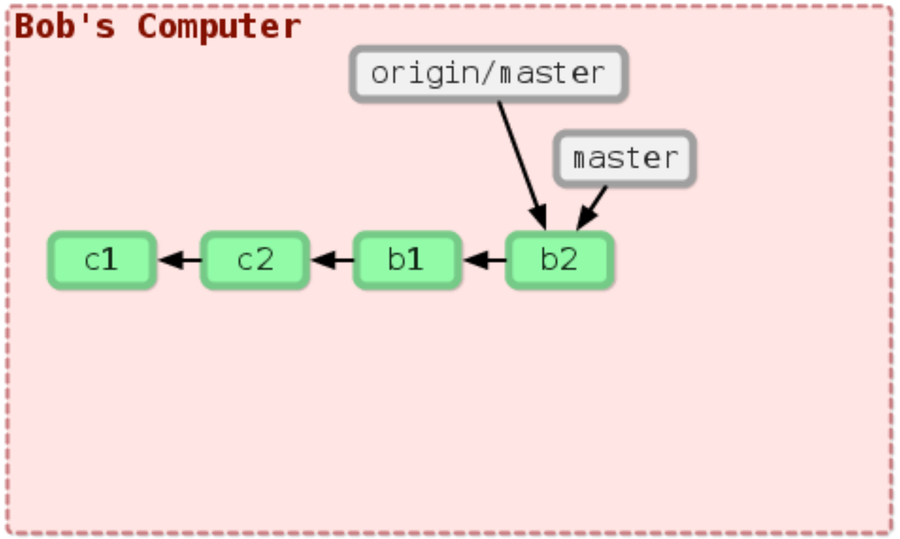



```
```

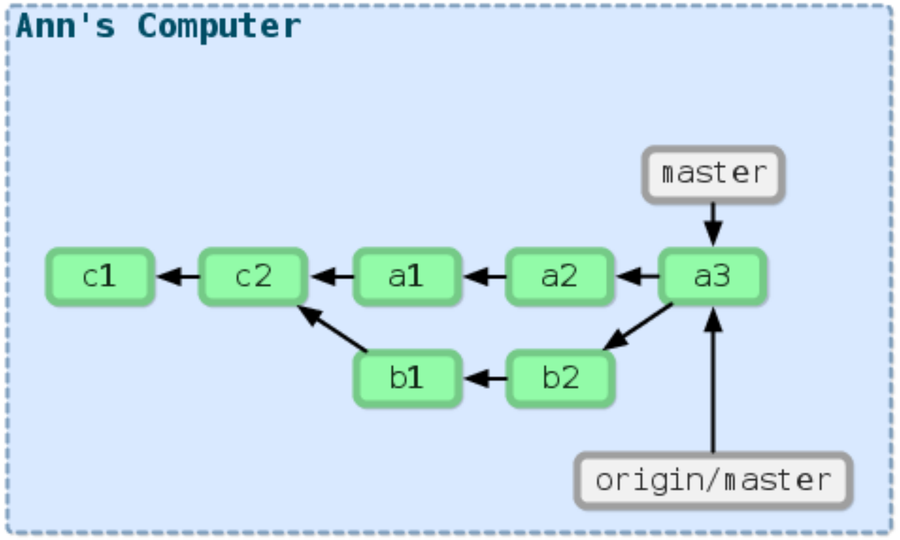


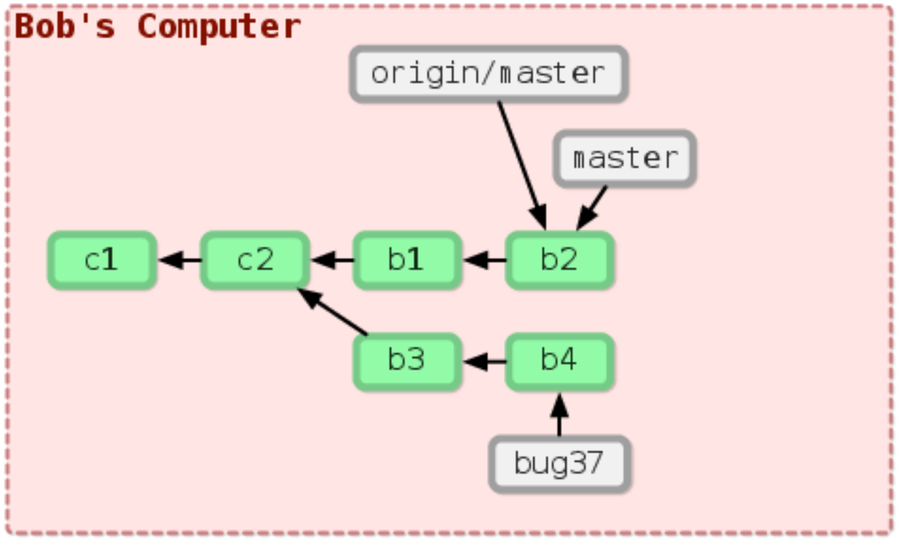

```
git push origin master
```

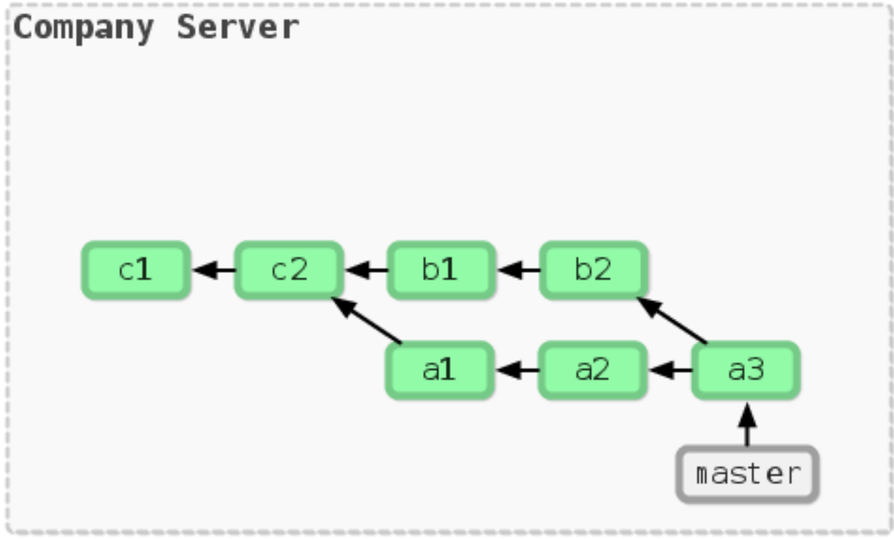




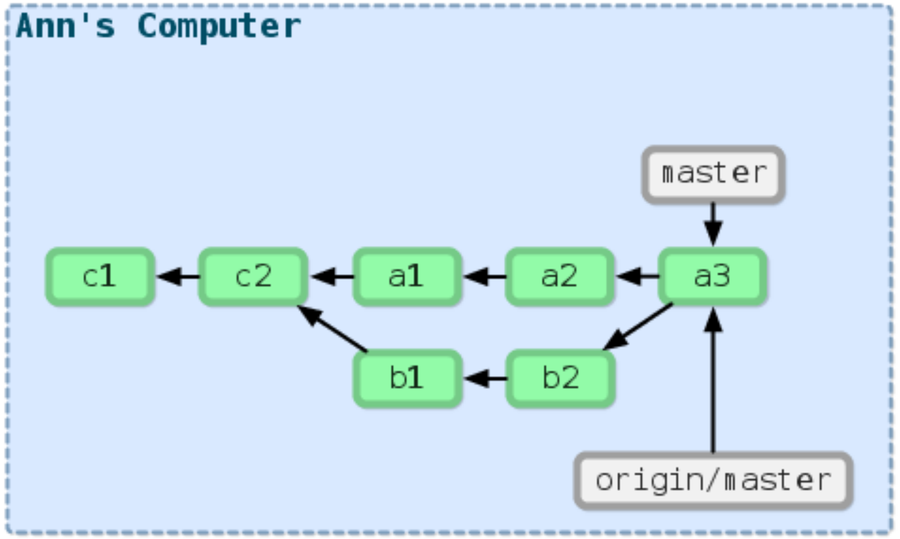

```
```

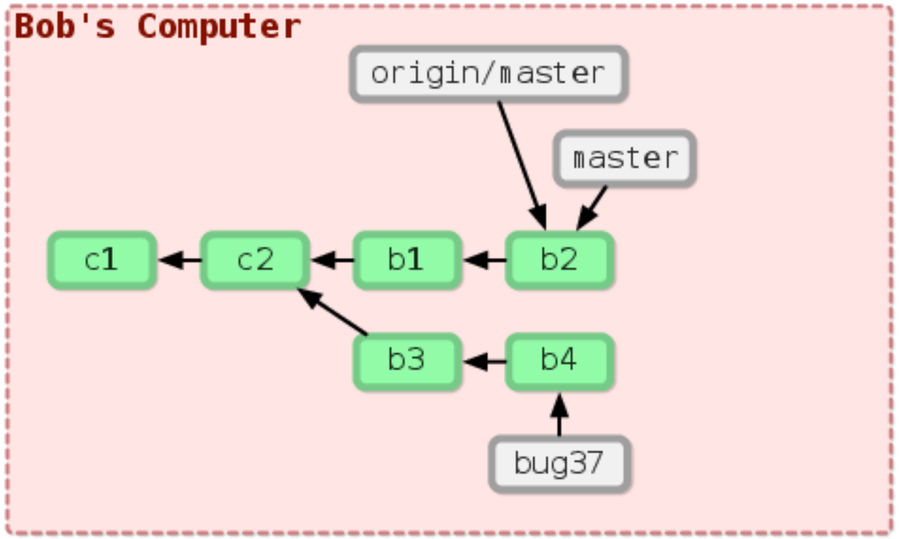


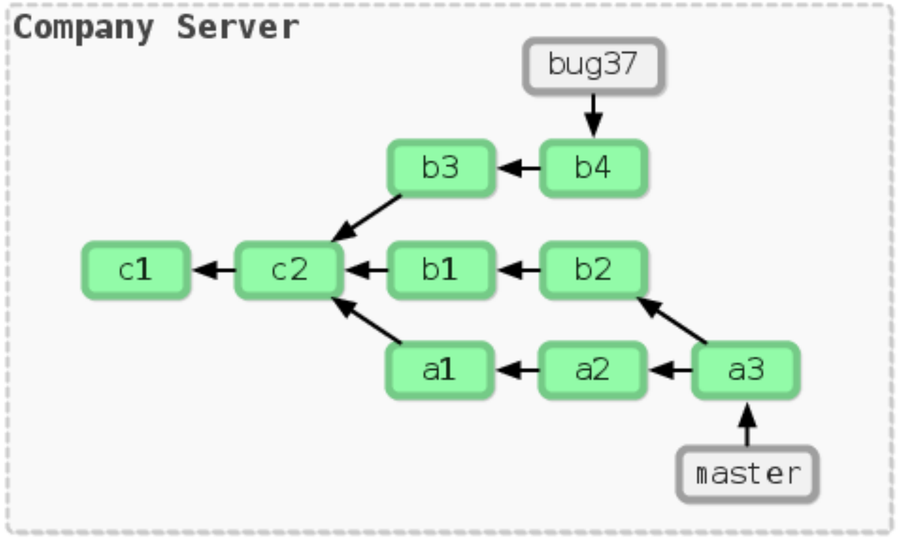




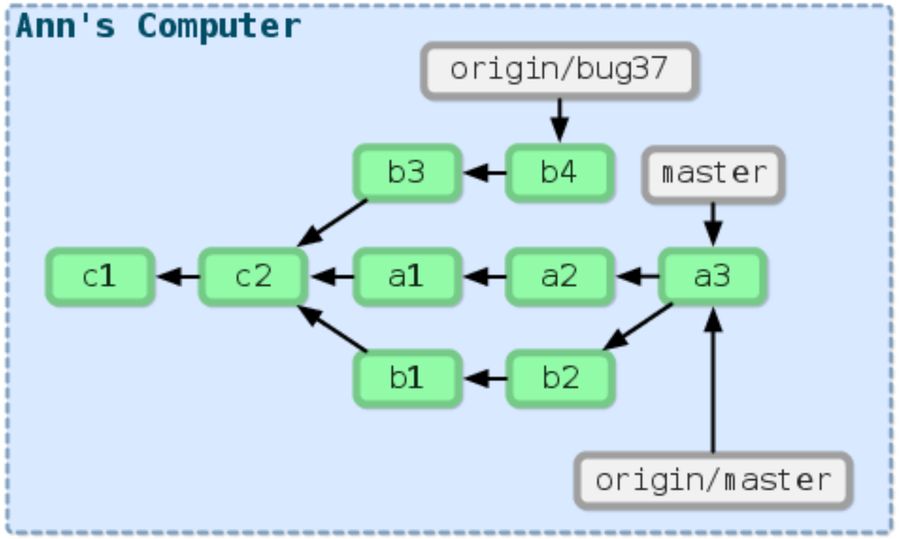

```
git checkout -b c2 bug37
git commit
git commit
```


Ann's Computer
master
c1
c2
a1
a2
a3
b1
b2
origin/master

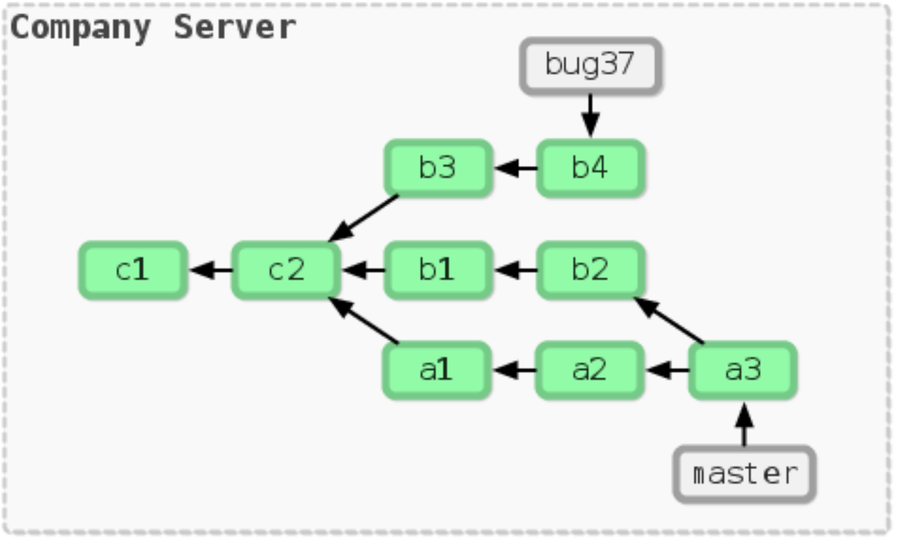

Bob's Computer
origin/master
master
c1
c2
b1
b2
b3
b4
bug37

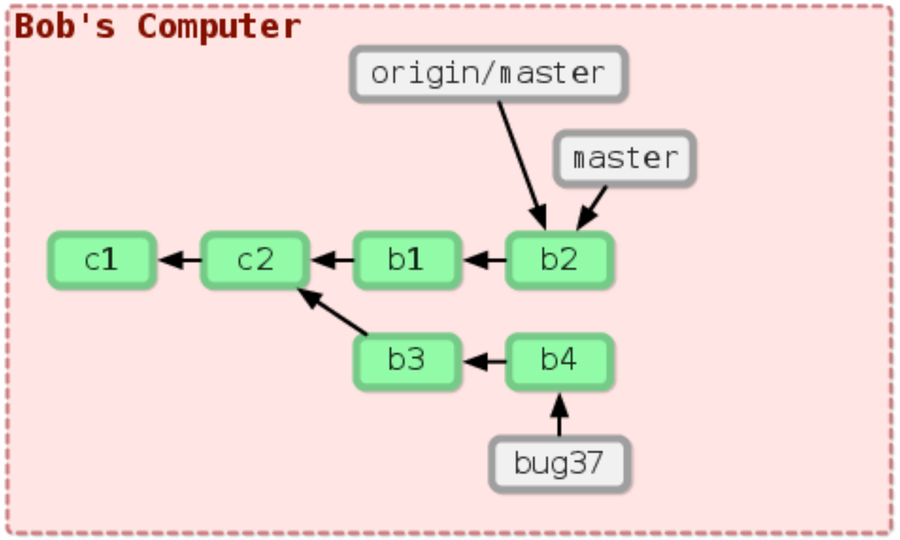

Company Server
bug37
b3
b4
c1
c2
b1
b2
a1
a2
a3
master


git push origin bug37


Ann's Computer
origin/bug37
b3
b4
master
c1
c2
a1
a2
a3
b1
b2
origin/master

```
git fetch
```


Company Server
bug37
b3
b4
c1
c2
b1
b2
a1
a2
a3
master


Bob's Computer
origin/master
master
c1
c2
b1
b2
b3
b4
bug37

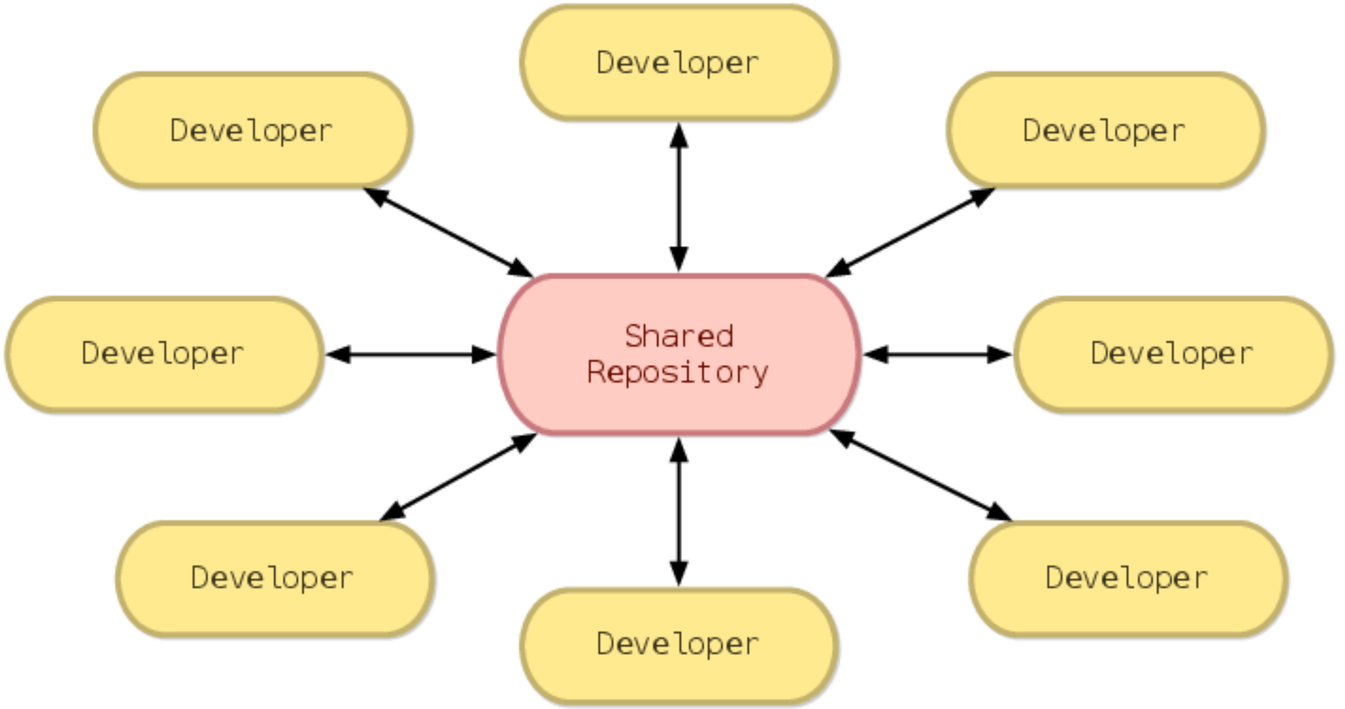
```
```

```
git pull
==
fetch + merge
```

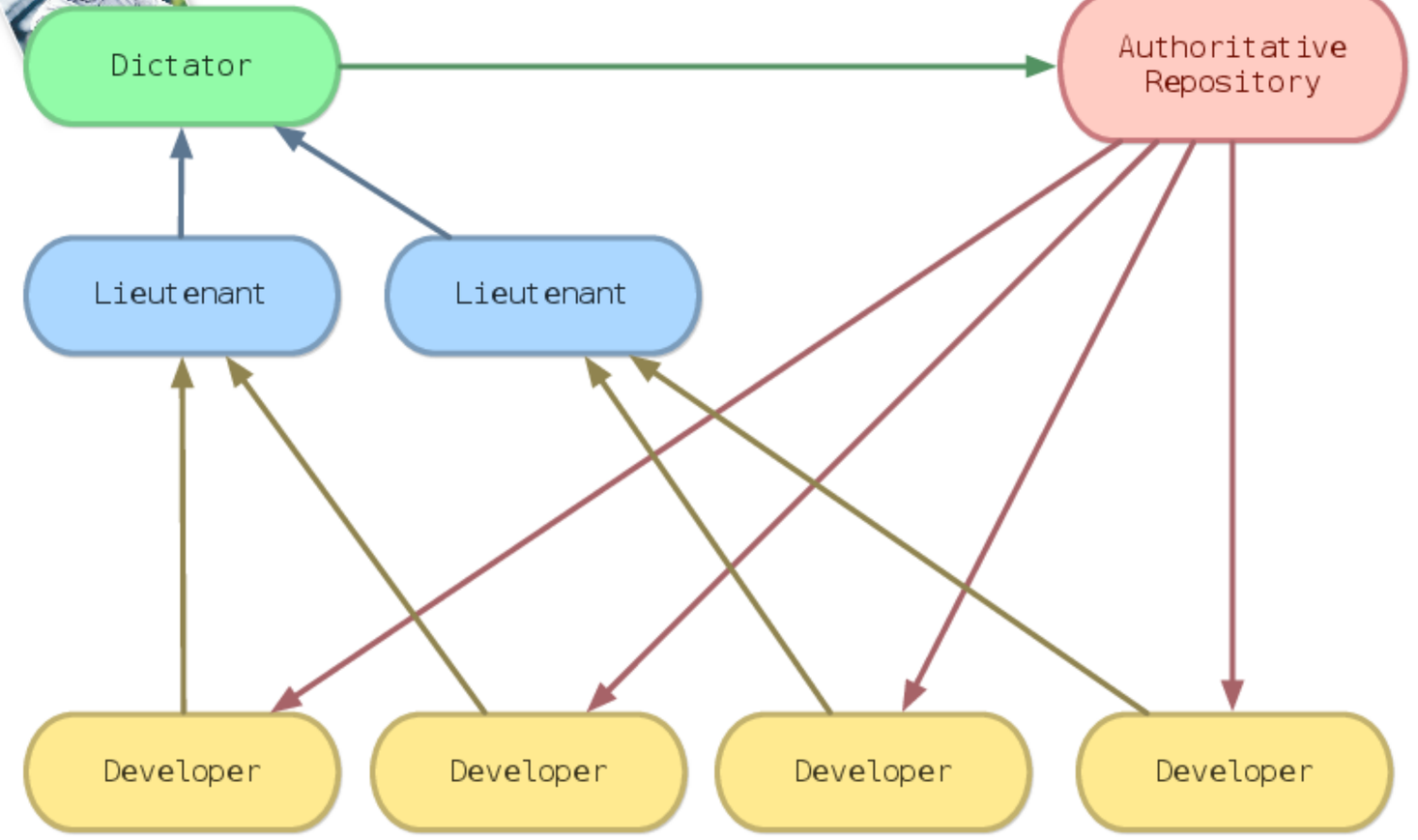
MHPC
Master in High Performance Computing

Collaboration

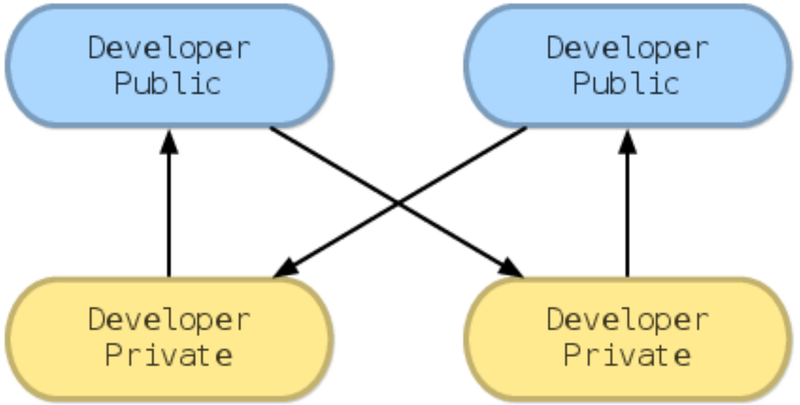
Centralized Model



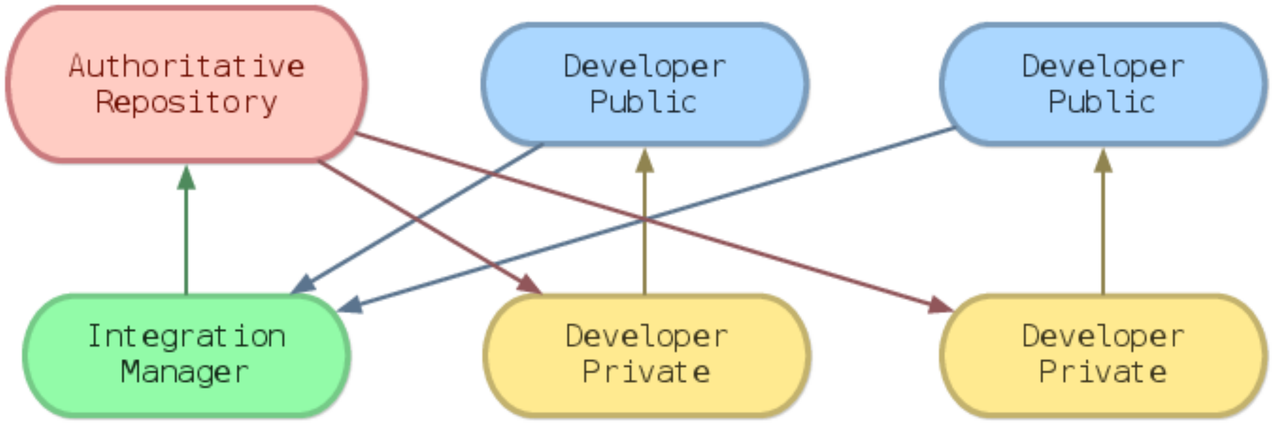
Dictator Model



Peer to Peer Model



Integration Model



Bottom Line

Many good collaboration models

Customize for **your** project / team

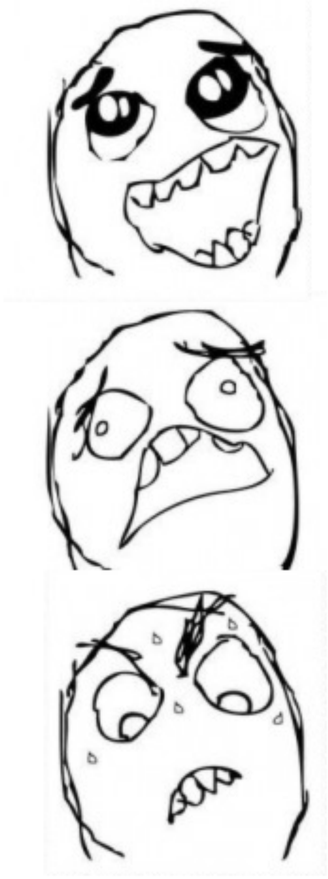
Distributed VCS gives you **options**

MHPC
Master in High Performance Computing

Workflow

MHPC
Master in High Performance Computing

Non Workflow



Make Changes

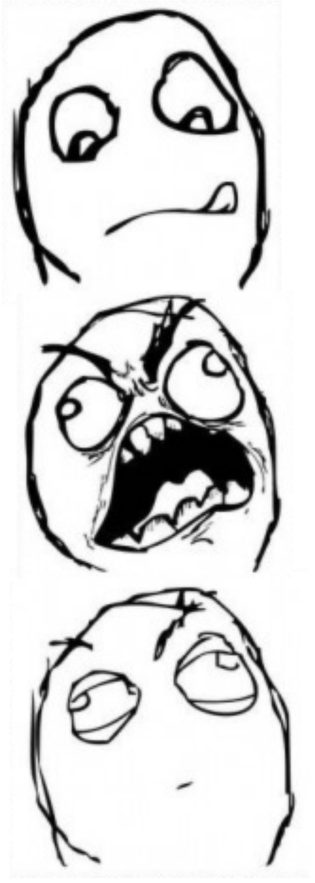
More Changes

Break Codebase

RAGE!!!

Fix Codebase

Repeat



Bad Workflow

Create Changes

Commit Changes

MHPC
Master in High Performance Computing

Basic Workflow

Create Changes

Stage Changes

Review Changes

Commit Changes

Workflow Pro Tip

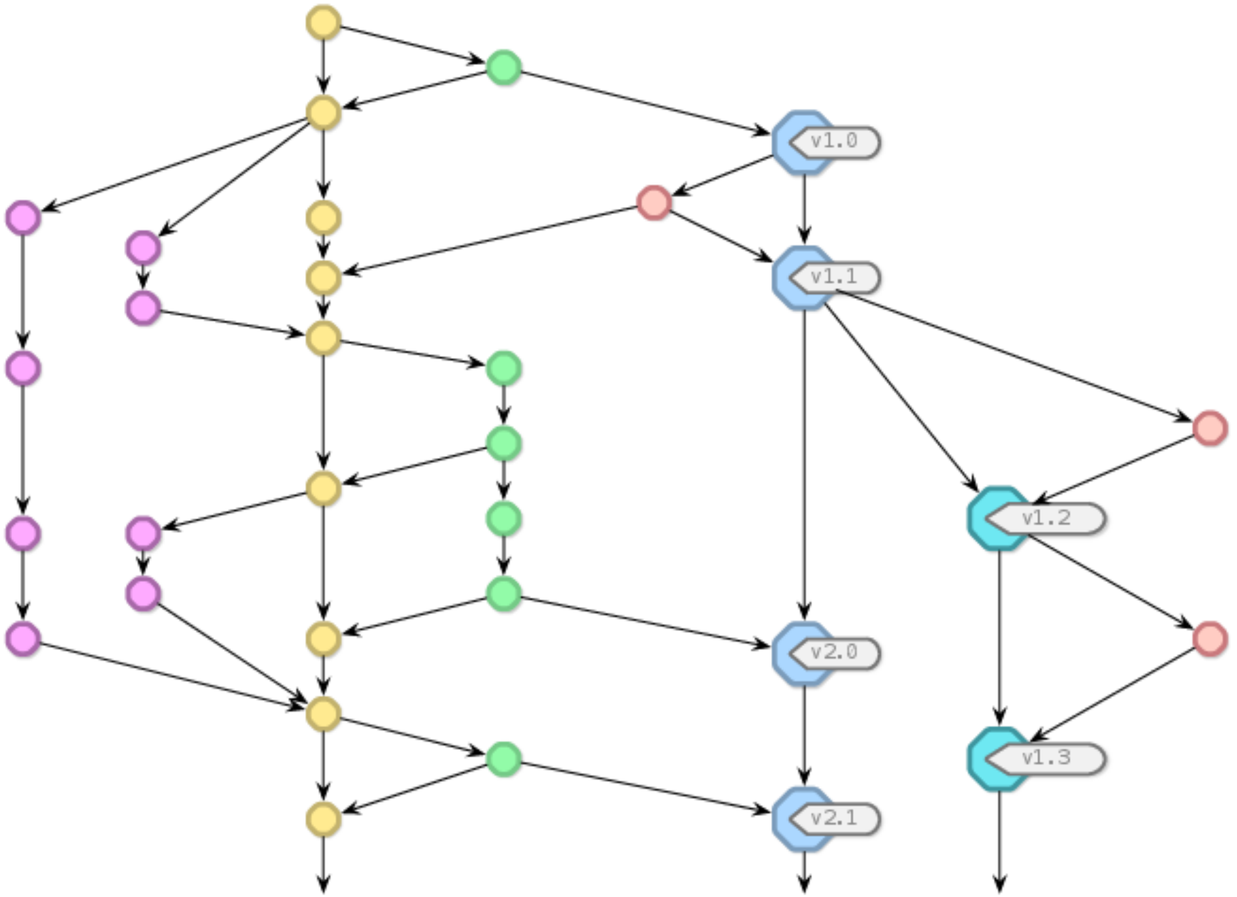
Commit **Early**

Commit **Often**

Commit **Units**

MHPC
Master in High Performance Computing

Good Workflow



Credit: **Vincent Driessen**, http://nvie.com/posts/a-successful-git-branching-model

MHPC
Master in High Performance Computing

DON'T PANIC!

Primary Branches

MHPC
Master in High Performance Computing

Master Branch

Always reflects PRODUCTION-READY state.

Always exists.

Develop Branch

Latest DEVELOPMENT state for next release.

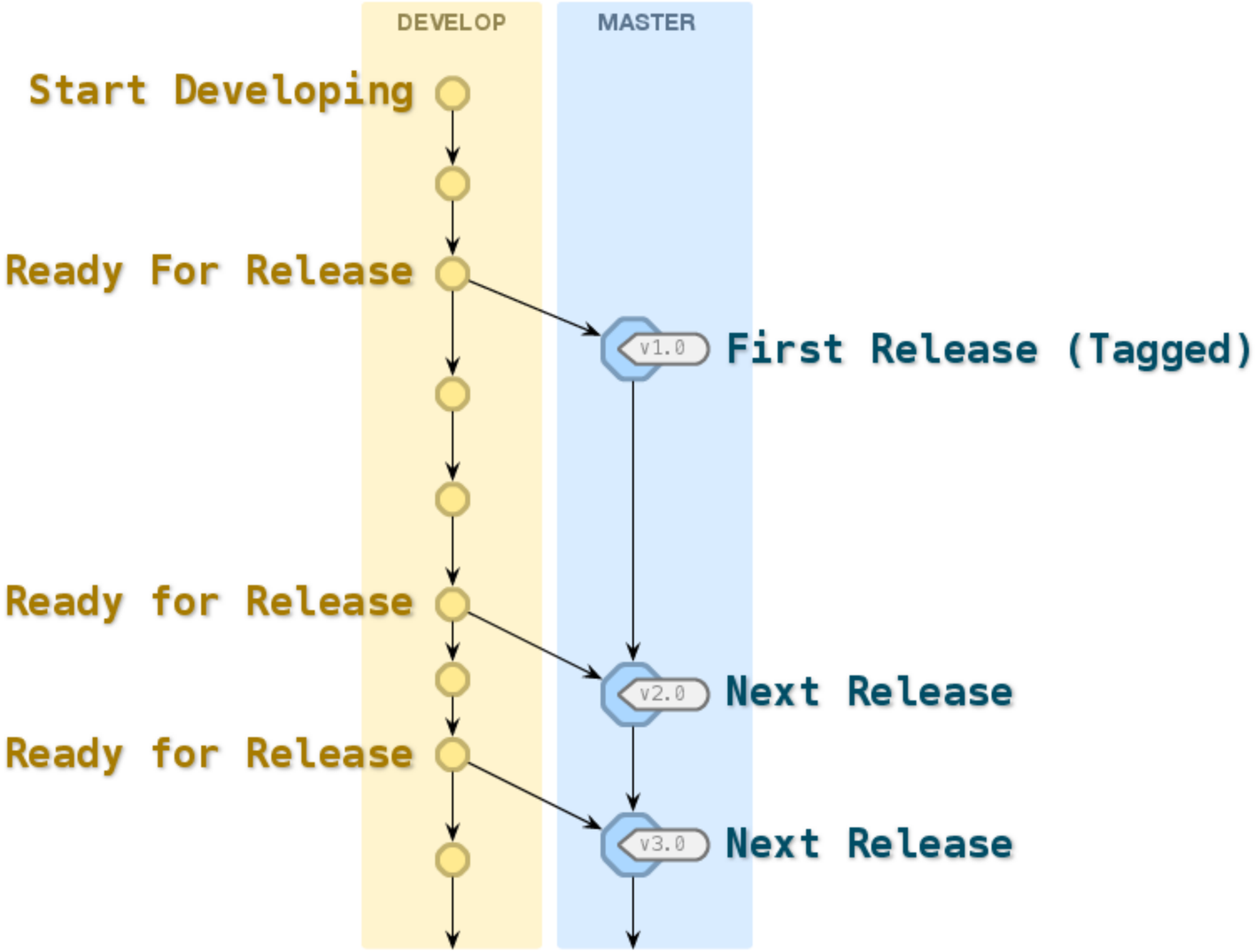
Base your continuous integration on this.

Ultimately ends up in MASTER.

Always exists.

Primary Branches

MHPC
Master in High Performance Computing



Secondary Branches

MHPC
Master in High Performance Computing

Feature Branches

Fine-grained **work-in-progress** for future release.

Branches off latest **DEVELOP**.

Merges back into **DEVELOP** then **discard**.

Or just discard (failed experiments).

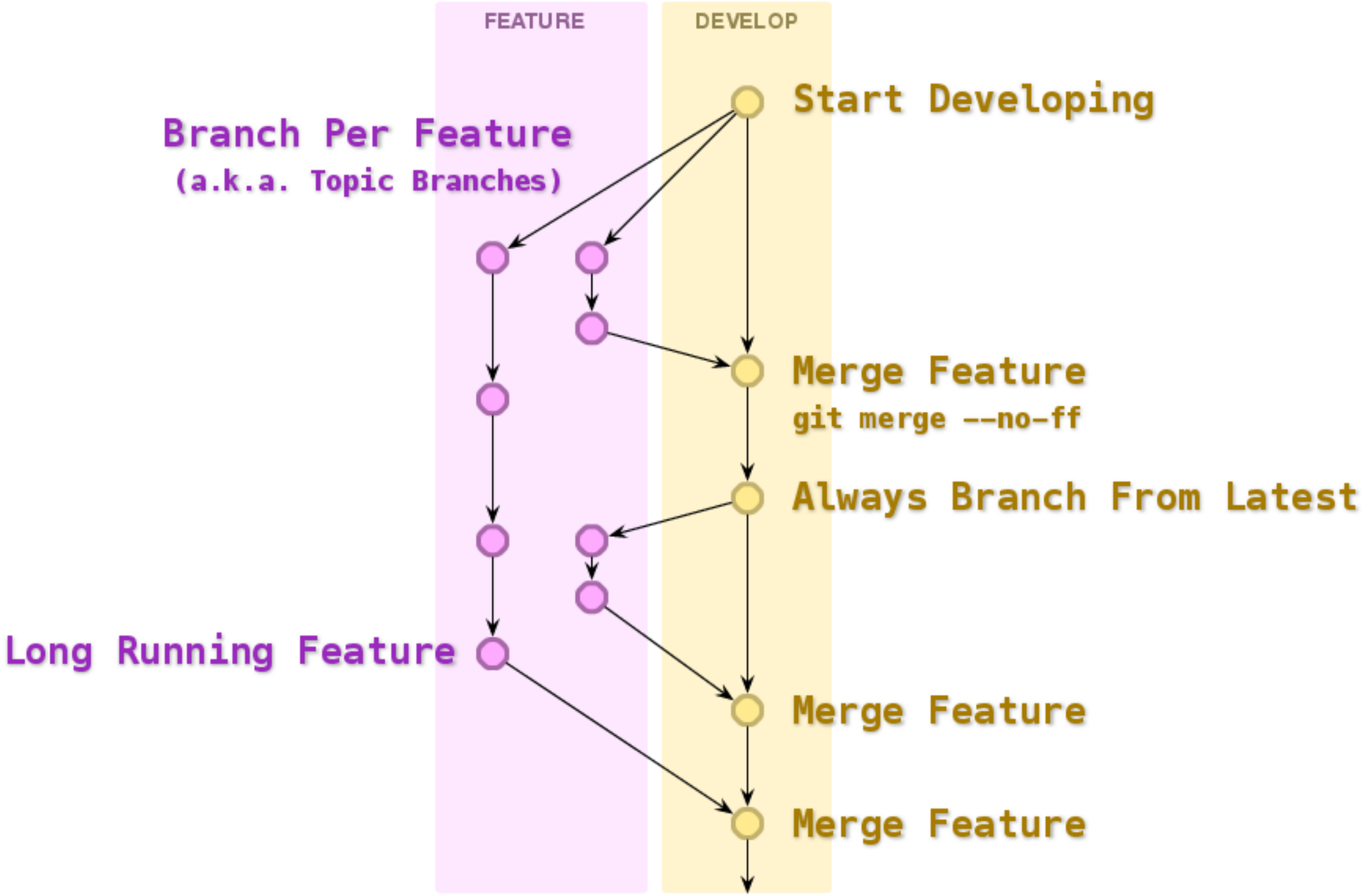
Short or long running.

Typically in developer repositories only.

Naming convention: **feature / cool-new-feature**

Feature Branches

MHPC
Master in High Performance Computing



Secondary Branches

MHPC
Master in High Performance Computing

Release Branches

Latest **RELEASE CANDIDATE** state.

Preparatory work for release.

Last minute QA, testing & bug fixes happens here.

Sits between **DEVELOP** and **MASTER**.

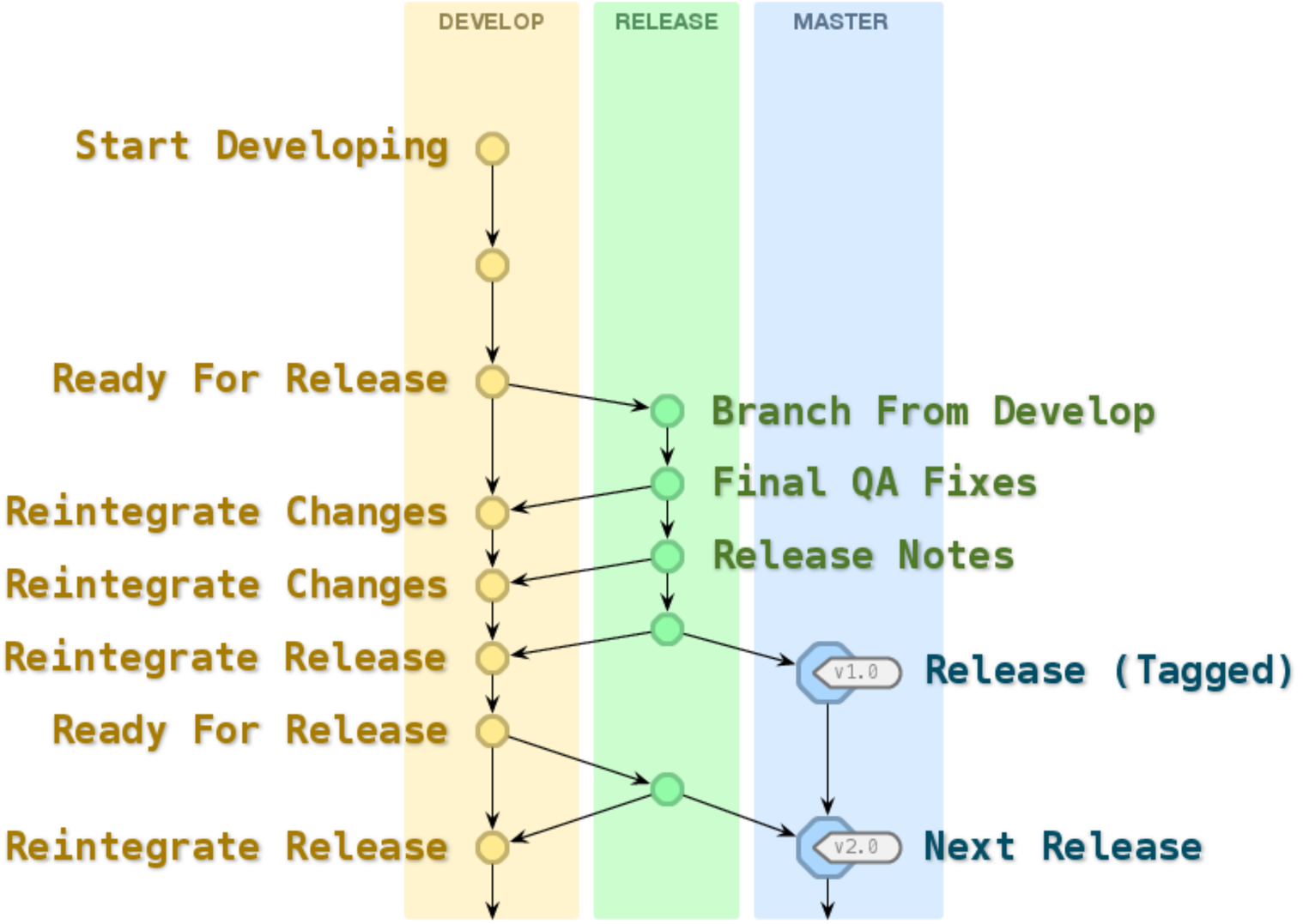
Branch from **DEVELOP**.

Merge back into **both** **MASTER** and **DEVELOP**.

Discard after merging.

Release Branches

MHPC
Master in High Performance Computing



Secondary Branches

MHPC
Master in High Performance Computing

HotFix Branches

Like **RELEASE**, preparing for new release.

Resolve **emergency problems** with **existing** production release.

Branch from **MASTER**.

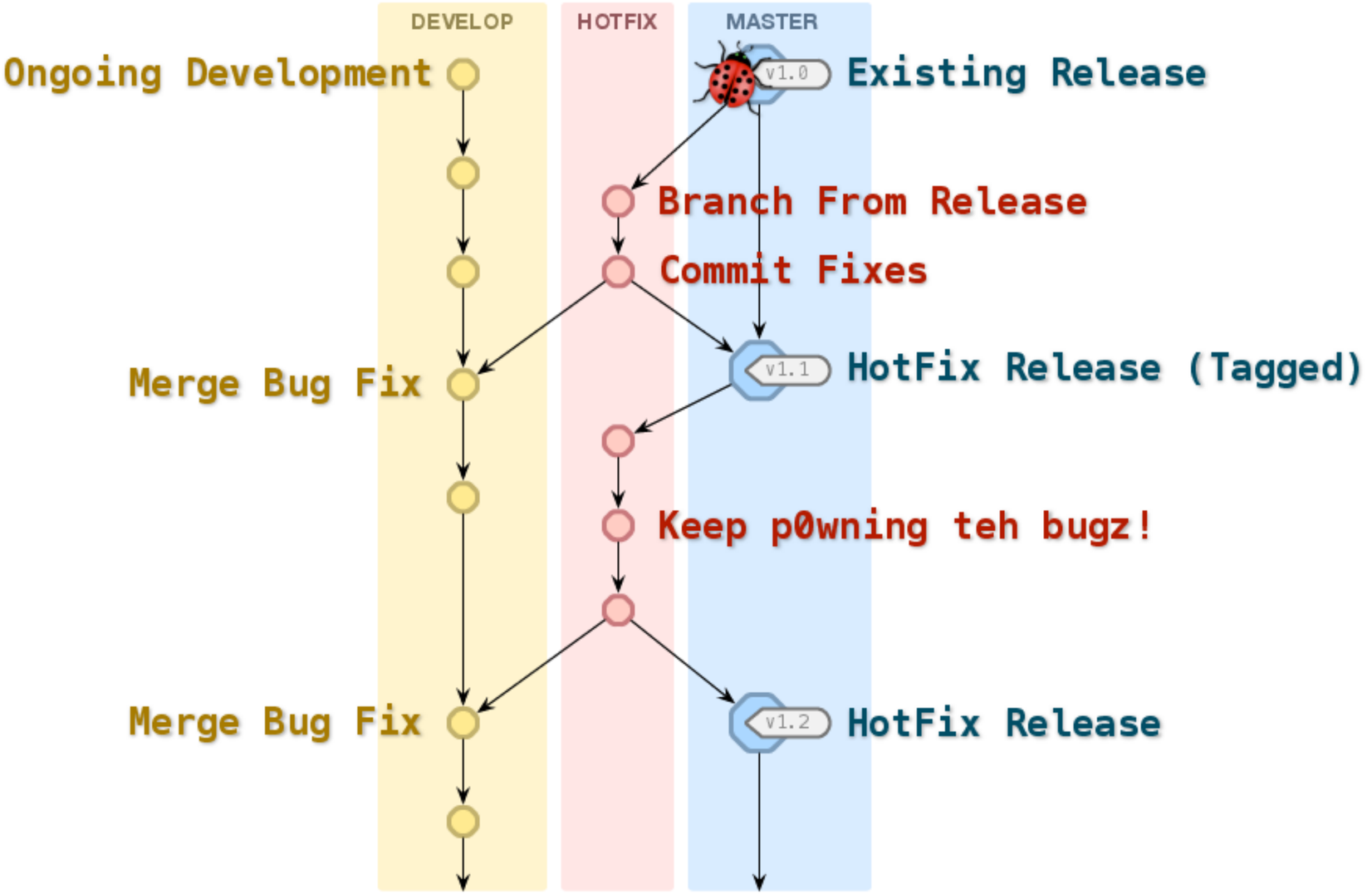
Merge back into both **MASTER** and **DEVELOP**.

Discard after merging.

Naming convention: **hotfix / bug-157**

HotFix Branches

MHPC
Master in High Performance Computing



Secondary Branches

MHPC
Master in High Performance Computing

Support Branches

Similar to **MASTER** + **HOTFIX** for **legacy releases.**

Branches off from **earlier** tagged **MASTER**.

Does not merge back into anything.

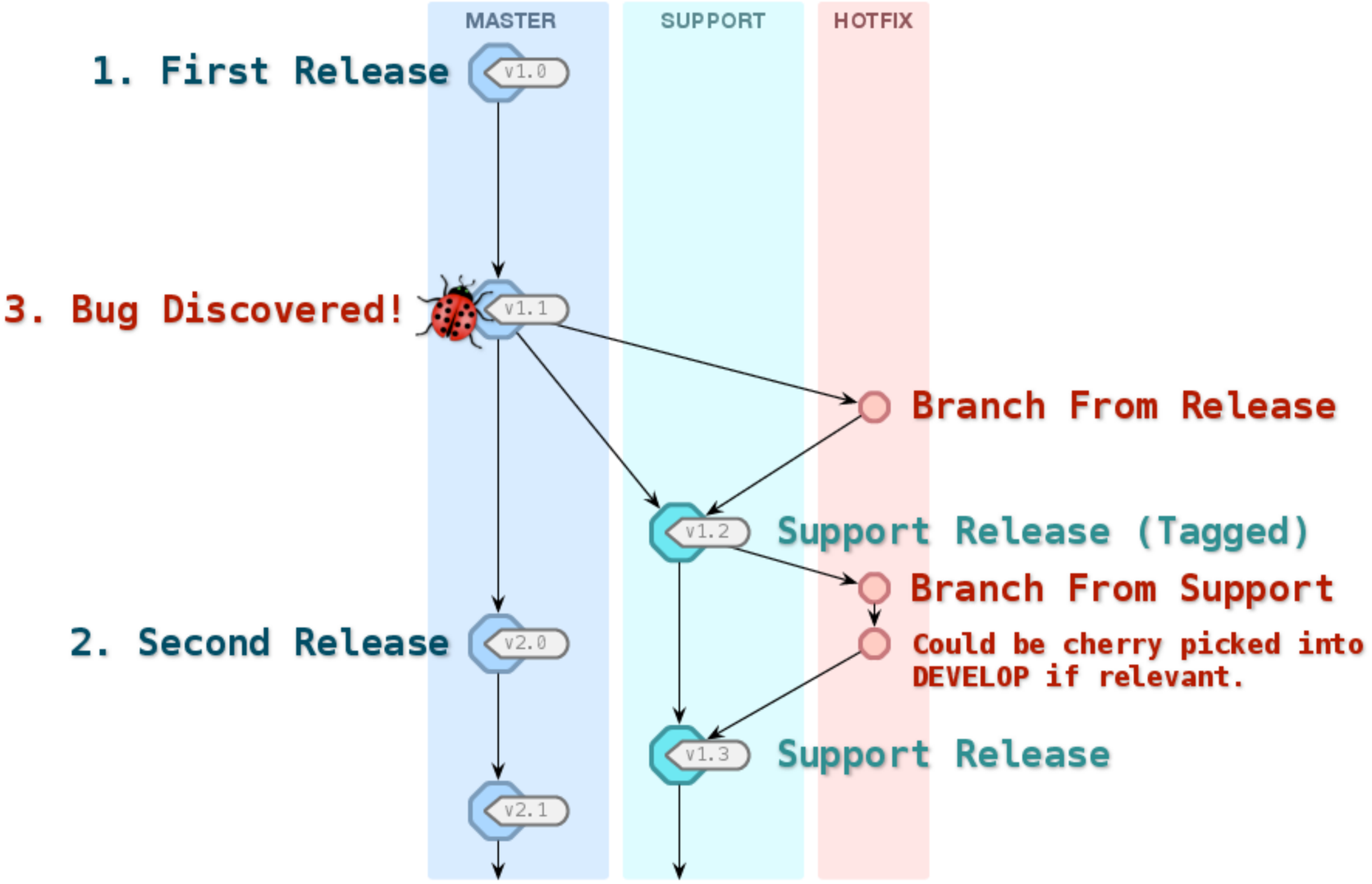
Always exists once created.

Continuing **parallel master branch** for a version series.

Naming convention: **support / version-1**

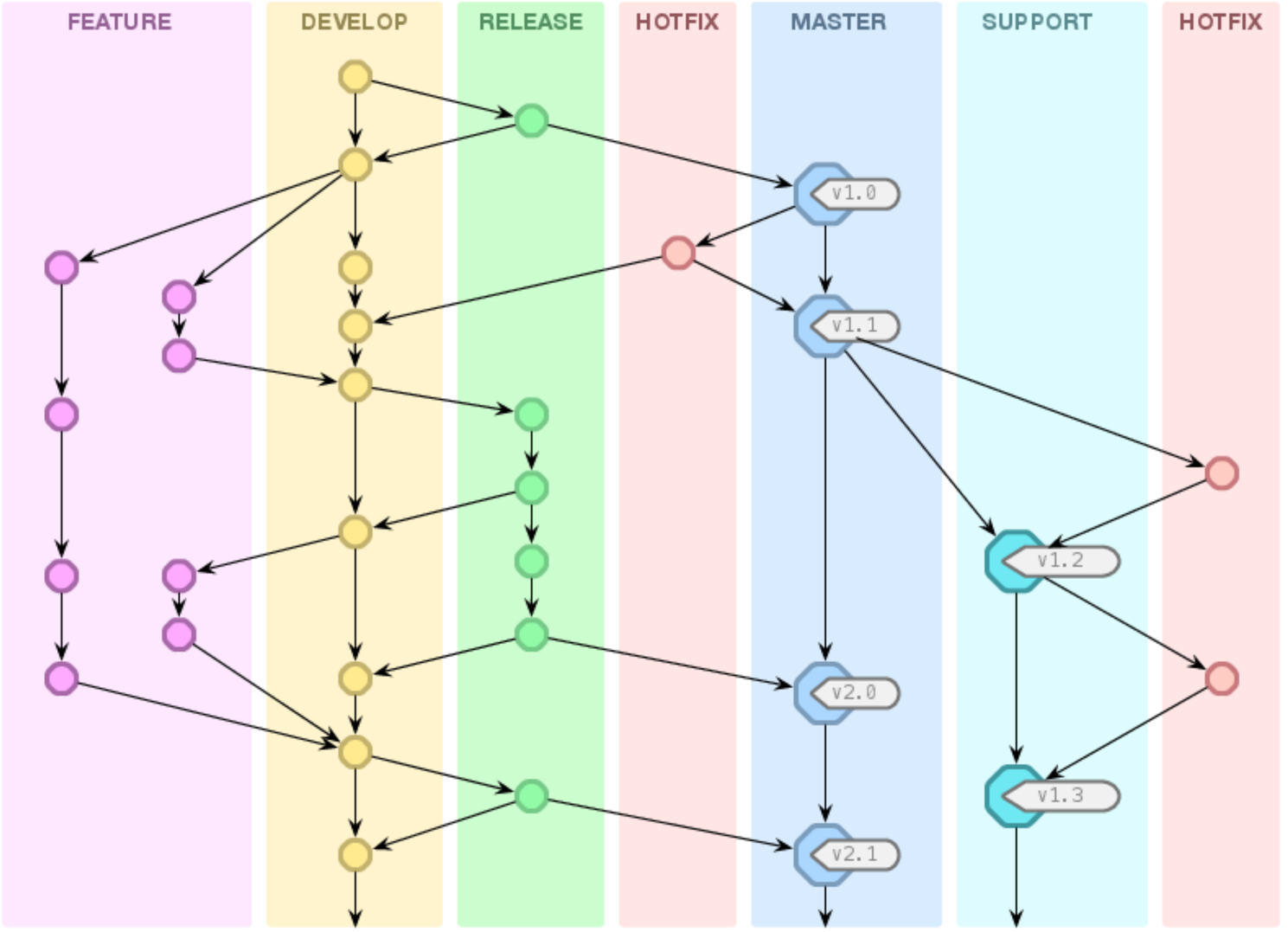
Support Branches

MHPC
Master in High Performance Computing



All Together Now

MHPC
Master in High Performance Computing



Public-Private Workflow

Public Branches

Authoritative history of the project.

Commits are **succinct** and well-documented.

As **linear** as possible.

Immutable.

Credit: **Benjamin Sandofsky**, http://sandofsky.com/blog/git-workflow.html

Public-Private Workflow

Private Branches

Disposable and **malleable.**

Kept in **local** repositories.

Never merge directly into **public**.

First **clean up** (reset, rebase, squash, and amend)

Then merge a **pristine, single commit** into **public**.

Credit: **Benjamin Sandofsky**, http://sandofsky.com/blog/git-workflow.html

Public-Private Workflow

1. Create a private branch off a public branch.
2. Regularly commit your work to this private branch.
3. Once your code is perfect, clean up its history.
4. Merge the cleaned-up branch back into the public branch.

Credit: **Benjamin Sandofsky**, http://sandofsky.com/blog/git-workflow.html

Bottom Line

Every project is **different**.

Custom **design** your workflow.

Branches are your LEGO blocks.

Further Reading

http://nvie.com/posts/a-successful-git-branching-model

https://github.com/nvie/gitflow

http://sandofsky.com/blog/git-workflow.html

Commanding Git

Edit ~/.profile

```
git config --global user.name "Patrick Hogan"
git config --global user.email pbhogan@gmail.com

git config --global core.autocrlf input
git config --global core.safecrlf true

git config --global color.ui true
```

MHPC
Master in High Performance Computing

Commanding Git

Edit ~/.profile

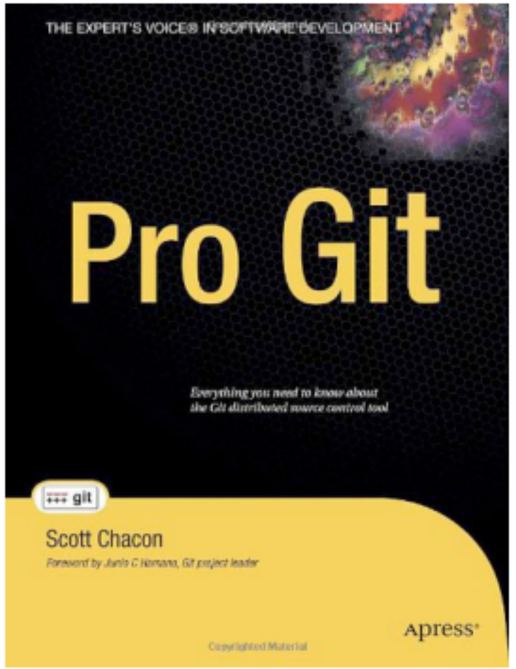
```
source /usr/local/etc/bash_completion.d/git-completion.bash

RED="\[\033[0;31m\]"
YELLOW="\[\033[0;33m\]"
GREEN="\[\033[0;32m\]"
WHITE="\[\033[1;37m\]"
RESET="\[\033[1;0m\]"
GIT='$(__git_ps1 "[%s]")'
PS1="\n$WHITE\u$RESET: \W$YELLOW$GIT $GREEN\$$RESET "
```

```
pbhogan: Swivel[master] $
```

MHPC
Master in High Performance Computing

Learning Git

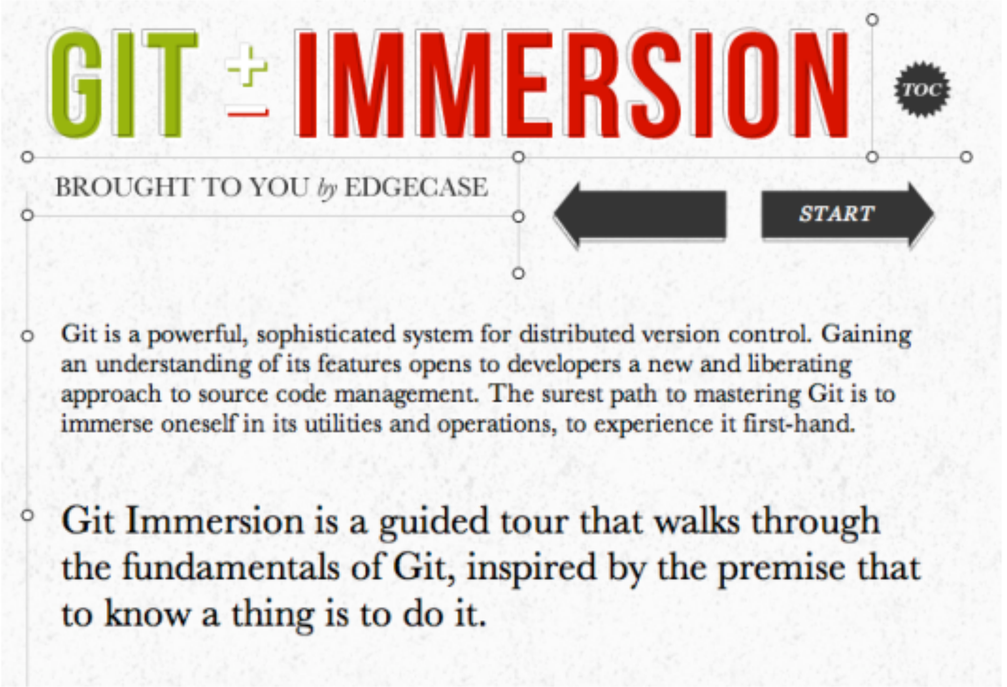


http://progit.org/book/

by Scott Chacon

MHPC
Master in High Performance Computing

Learning Git



http://gitimmersion.com

by the EdgeCase team

MHPC
Master in High Performance Computing

Hosting Git

http://dropbox.com

DIY single user hosting :-)

MHPC
Master in High Performance Computing

Hosting Git


```
$ mkdir -p /Users/pbhogan/Dropbox/Repos/Swivel.git
$ cd /Users/pbhogan/Dropbox/Repos/Swivel.git
$ git init --bare
$ cd /Users/pbhogan/Projects/Swivel
$ git remote add dropbox file:///Users/pbhogan/Dropbox/Repos/Swivel.git
```

http://dropbox.com

DIY single user hosting :-)

The end