Git Deep Dive

Topics

- Intro
  - Setup and configuration of git
- Foundational Concepts
  - commits, branches, 3 states of files
- Sharing and Integrating content across branches
  - merge vs rebase vs cherry-pick
- Distributed Git Repos
  - git clone, local vs remote branches, git push
- Changing history
  - re-ordering commits, squashing, fixing

What is Git?

Git is a distributed version control system

Git is used for

- tracking changes to files
- coordinating changes from multiple people

Basic Git Setup

- Download and install
- Setup a .gitconfig file
    - configure user
    - configure some basic usefull shortcuts

.gitconfig

```
[user]
        name = Monica
        email = mraj@floof.com
[core]
        editor = vim
[alias]
    graph = log \
            --date=local \
            --graph \
            --format=format:'%C(bold blue)%h%C(reset) - %C(bold
cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold
yellow)%d%C(reset)%n%C(white)%w(110,10,10)%s%C(reset) %C(bold
yellow)- %an %n%C(red)%b%C(reset)' --abbrev-commit

    graphall = log \
            --date=local \
            --graph \
            --all \
            --format=format:'%C(bold blue)%h%C(reset) - %C(bold
cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold
yellow)%d%C(reset)%n%C(white)%w(110,10,10)%s%C(reset) %C(bold
yellow)- %an %n%C(red)%b%C(reset)' --abbrev-commit
    co       = checkout

[color]
    branch  = auto
    diff    = auto
    status  = auto
    ui      = auto

...
```

```
...
[push]
    default = simple

[pull]
    rebase = true

[blame]
    date = short

[diff]
    tool = vimdiff

[merge]
    tool = vimdiff
    conflictstyle = diff3
```

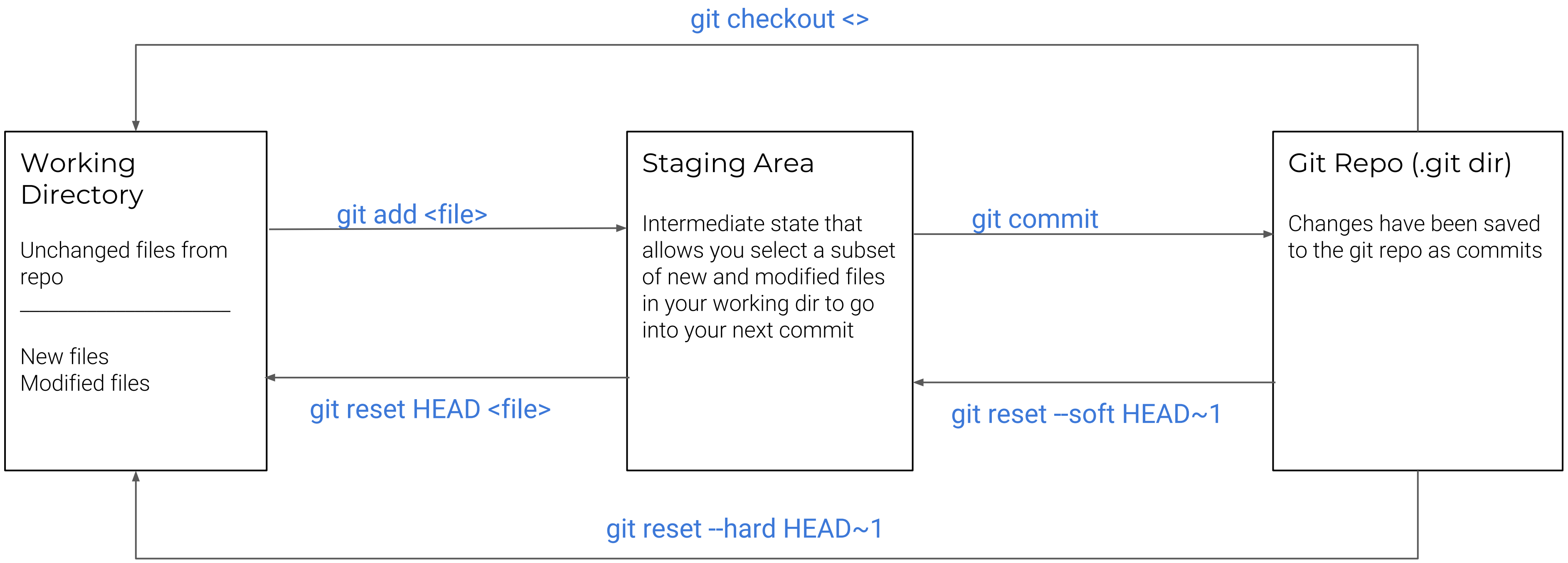
Foundational Git Concepts

What is a git repo ?

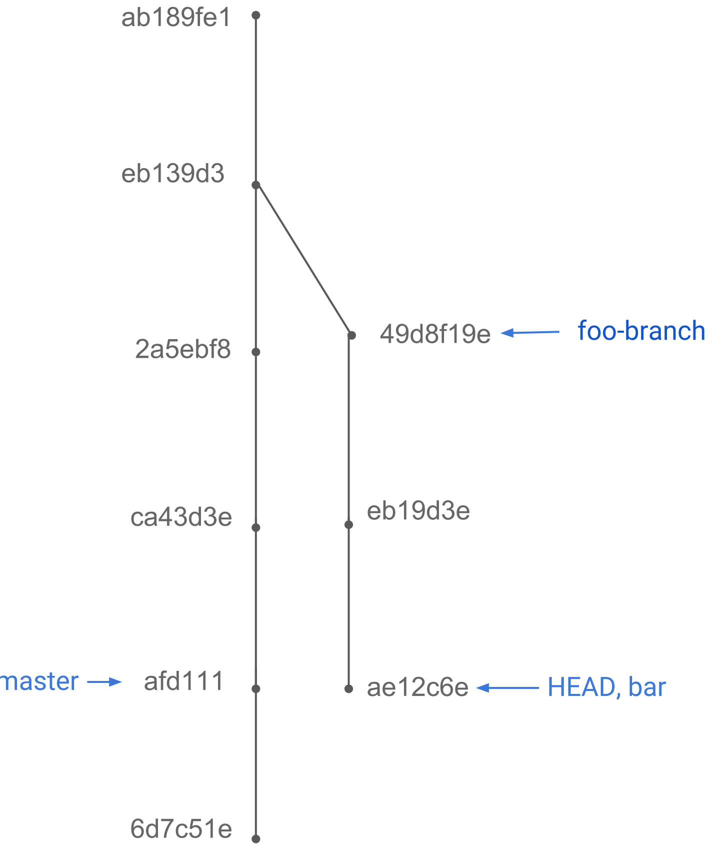
- A git repo (repository)
  - Repository: “a place to gather and store something”
  - A git repo includes a set of directories and files and all their history and versions
    - This concept will be fleshed out in more detail through the remaining slides

Working Directory, Staging Area, Git Repo

All the files in your git controlled directory exist in one of these 3 states:


git checkout <>
Working Directory
Unchanged files from repo
New files
Modified files
git add <file>
git reset HEAD <file>
Staging Area
Intermediate state that allows you select a subset of new and modified files in your working dir to go into your next commit
git commit
git reset --soft HEAD~1
Git Repo (.git dir)
Changes have been saved to the git repo as commits
git reset --hard HEAD~1

Visualizing Git Commits and Branches



Git Commit

A git commit is a set diffs

A record of the incremental changes relative to the previous commit

A commit is identifiable by the commit hash

Every commit has a commit message.

Git Branch

A reference or pointer to a commit

HEAD

A pointer to the most recent commit in the currently checkedout branch

Tag

A reference to a commit

Integrating content across branches

Why?

- Contribute your local changes to the shared public branch for everyone to use

- Pull in other peoples changes so you can integrate your changes with other concurrent changes
  - Do this as often as possible for the repos you are working on.
  - Multiple times a day or at least once a day

merge vs rebase vs cherry-pick

- Merge:
  - Integrate all the changes from one branch into another branch
  - Non-destructive, will not change any existing commits
  - Use this when integrating private changes back into a shared/public branch

- Rebase:
  - Integrate all the changes from one branch into another branch
  - Will result in changing git history
  - Will result in changing git commit hashes
  - Use this when keeping your branch up-to-date with work from other developers
  - Do not do this public branches, private (aka topic) branches only

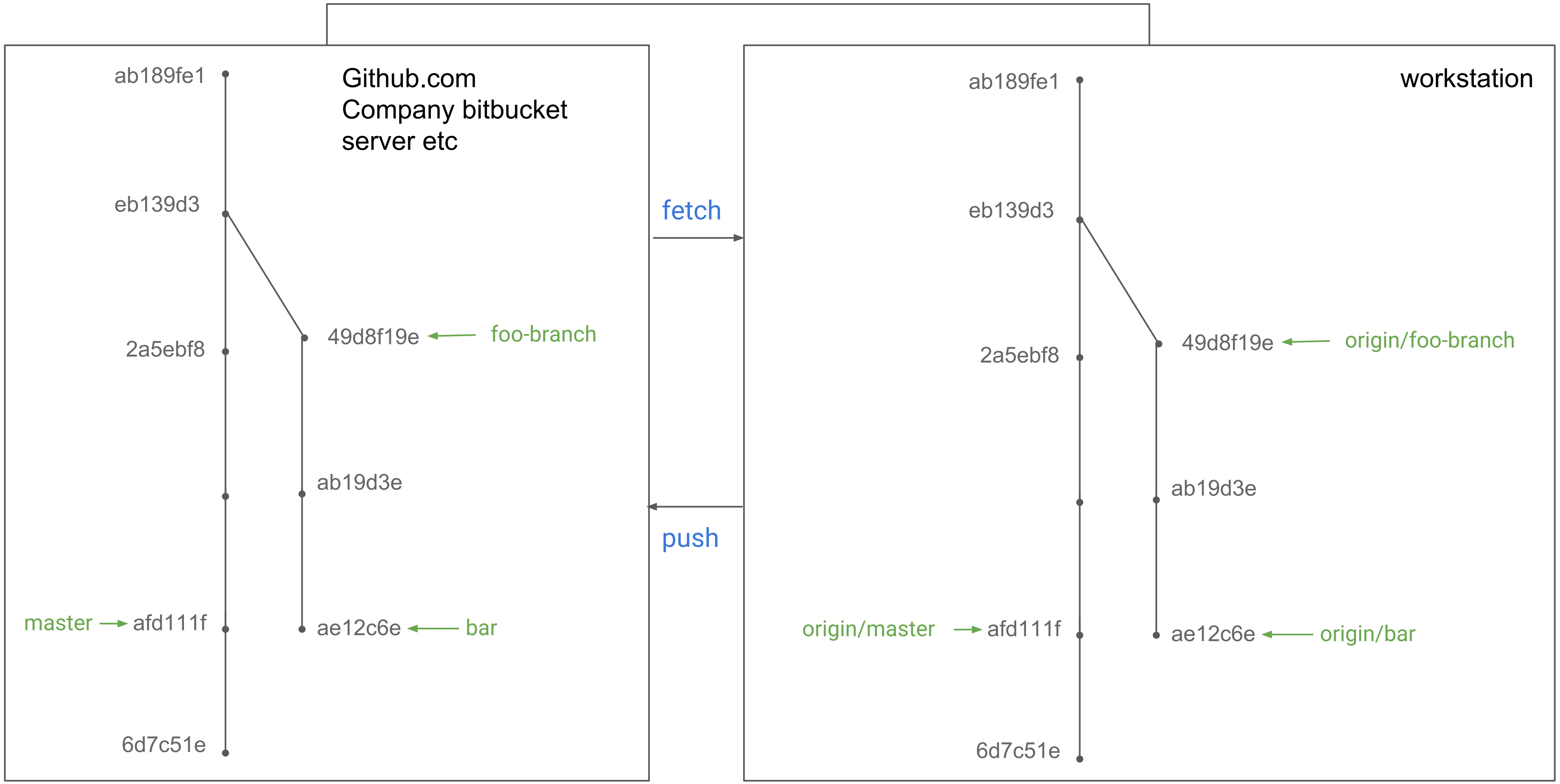
- Cherry-pick:
  - Selectively apply the changes from a single commit from one branch onto another branch

Discuss remote git repos before coming back to discussing merging/rebasing

Distributed Git Repos

Clone

```
git clone <URL-of-remote-repo>/<path-to-repo-on-remote-host>
git clone ssh://git@remote-repo-server/floof
```



Sharing Content Across Git Repos on Different Hosts

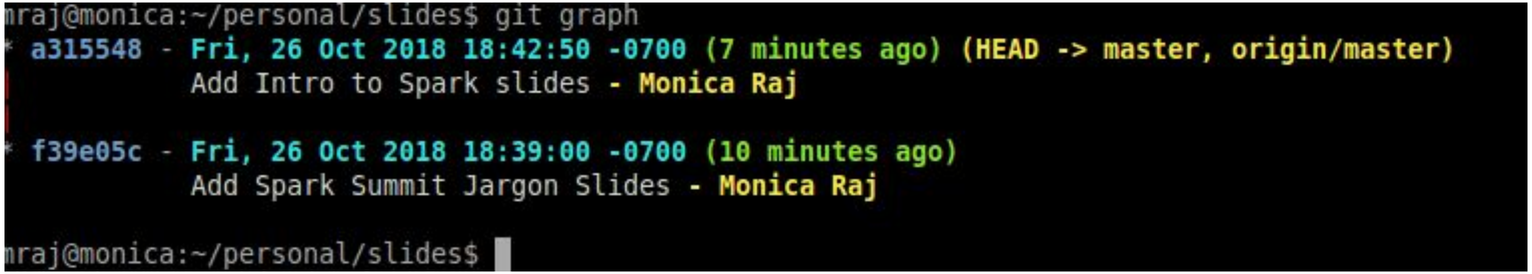
- git fetch
  - git fetch <remote-name>
  - git fetch -a

- git push
  - git push <remote-name> <local-reference>:<remote-reference>

List Remotes

git remote -vv

Local vs Remote Branches



```
nraj@monica:~/personal/slides$ git graph
* a315548 - Fri, 26 Oct 2018 18:42:50 -0700 (7 minutes ago) (HEAD -> master, origin/master)
|           Add Intro to Spark slides - Monica Raj
|
* f39e05c - Fri, 26 Oct 2018 18:39:00 -0700 (10 minutes ago)
            Add Spark Summit Jargon Slides - Monica Raj

nraj@monica:~/personal/slides$
```

Merge and Rebase Gotchas

- Use local develop branch instead of remote develop branch when integrating upstream changes into your git repo
- Move remote branch pointer to point to a private branch
    - How does this happen? Consequences
- Using merge --no-ff when integrating upstream changes into your git repo
- Creating a topic branch off one branch and merging that topic branch into a different branch

Changing History

Changing Git History

Clean-up git history by consolidating commits into logical bundles of changes

- Rebase
  - Squash
  - Fix-up

Reverting commits

```
git revert <commit-hash>
```

- Creates a new commit with inverse changes to the specified commit
- Is a non-destructive (no change to git history) way of undoing a commit
- Use this when you want to undo commits that have been pushed to a public branch
- Be aware that if you are reverting a commit that is not the most recent one, you will likely have to resolve merge conflicts

FIN

How do you get a git repo?

- git init
  - Create a brand new empty repo on your local machine

- git clone <url-of-remote-repo>
  - Copy a remote repo onto your local machine
  - Full repo is copied, this includes the files and all histories and branches (more on this later)

git repo from scratch

```
$ mkdir floof
```

```
$ cd floof
$ git init
   Initialized empty Git repository in /home/mraj/floof/.git/


$ git graph
fatal: your current branch 'master' does not have any commits yet



 $ git branch

 $ git branch -a

 $ git branch -vv



 $ git status
 On branch master

 No commits yet

 nothing to commit (create/copy files and use "git add" to track)
```

```
└── floof
    └── .git
        ├── branches
        ├── config
        ├── description
        ├── HEAD
        ├── hooks
        │   ├── applypatch-msg.sample
        │   ├── commit-msg.sample
        │   ├── post-update.sample
        │   ├── pre-applypatch.sample
        │   ├── pre-commit.sample
        │   ├── prepare-commit-msg.sample
        │   ├── pre-push.sample
        │   ├── pre-rebase.sample
        │   ├── pre-receive.sample
        │   └── update.sample
        ├── info
        │   └── exclude
        ├── objects
        │   ├── info
        │   └── pack
        └── refs
            ├── heads
            └── tags
```