

# Introduction to Git

Ashley Roach -- Principal Engineer Evangelist -- DevNet

Twitter: [@aroach](#)

Email: [asroach@cisco.com](mailto:asroach@cisco.com)

# Agenda

- Introduction
- Quick overview of Git
- Workshop
- Resources

# How is this relevant to networking?

- Programmability
- Scripts
- Infrastructure as code
  - Ansible
  - Terraform
- Even if you're solo, version control is critical

# WHY ARE WE HERE?



# Terraform: Infra layer

```
103 /*
104   Public Subnet
105 */
106 resource "aws_subnet" "us-east-1-public" {
107   vpc_id = "${aws_vpc.default.id}"
108
109   cidr_block = "${var.public_subnet_cidr}"
110   availability_zone = "us-east-1a"
111
112   tags {
113     Name = "Public Subnet"
114   }
115 }
116
117 resource "aws_route_table" "us-east-1-public" {
118   vpc_id = "${aws_vpc.default.id}"
119
120   route {
121     cidr_block = "0.0.0.0/0"
122     gateway_id = "${aws_internet_gateway.default.id}"
123   }
124
125   tags {
126     Name = "Public Subnet"
127   }
128 }
129
130 resource "aws_route_table_association" "us-east-1-public" {
131   subnet_id = "${aws_subnet.us-east-1-public.id}"
132   route_table_id = "${aws_route_table.us-east-1-public.id}"
133 }
134
135 /*
136   Private Subnet
137 */
138 resource "aws_subnet" "us-east-1-private" {
139   vpc_id = "${aws_vpc.default.id}"
140
141   cidr_block = "${var.private_subnet_cidr}"
142   availability_zone = "us-east-1a"
143
144   tags {
145     Name = "Private Subnet"
146   }
147 }
148
```

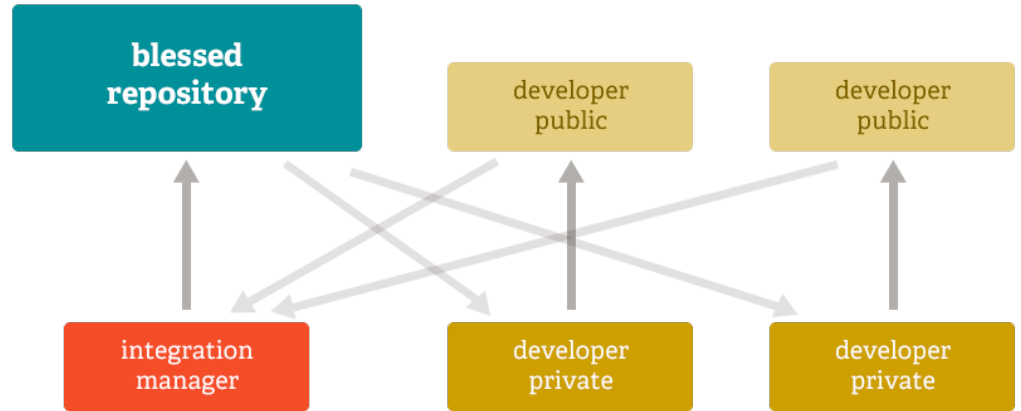
# Ansible: OS/App Layer

```
1 ---
2 # inspiration from https://github.com/belmarca/caddy-ansible
3
4 - name: Create Caddy user
5   become: yes
6   user:
7     name=caddy
8     system=yes
9     createhome=yes
10
11 - name: Get all Caddy releases
12   become: yes
13   get_url:
14     url=https://api.github.com/repos/mholt/caddy/git/refs/tags
15     dest=/home/caddy/releases.txt
16     force=yes
17   when: caddy_update
18   register: caddy_releases_cache
19
20 - name: Set Caddy features
21   become: yes
22   copy:
23     content="{{caddy_features}}"
24     dest=/home/caddy/features.txt
25   when: caddy_update
26   register: caddy_features_cache
27
28 - name: Download new Caddy version or build
29   become: yes
30   get_url:
31     url=https://caddyserver.com/download/linux/amd64?plugins=tlsls.dns.gandi
32     dest=/home/caddy/caddy.tar.gz
33     force=yes
34   when: caddy_releases_cache.changed or caddy_features_cache.changed
35   register: caddy_binary_cache
36
37 - name: Download Caddy
38   become: yes
39   get_url:
40     url=https://caddyserver.com/download/linux/amd64?plugins=tlsls.dns.gandi
41     dest=/home/caddy/caddy.tar.gz
42
43 - name: Extract new Caddy version or build
44   become: yes
45   unarchive:
46     src=/home/caddy/caddy.tar.gz
47     dest=/usr/bin/
48     copy=no
49   when: caddy_binary_cache.changed
```

# Why version control? To Protect yourself and others

# DISTRIBUTED VERSION CONTROL

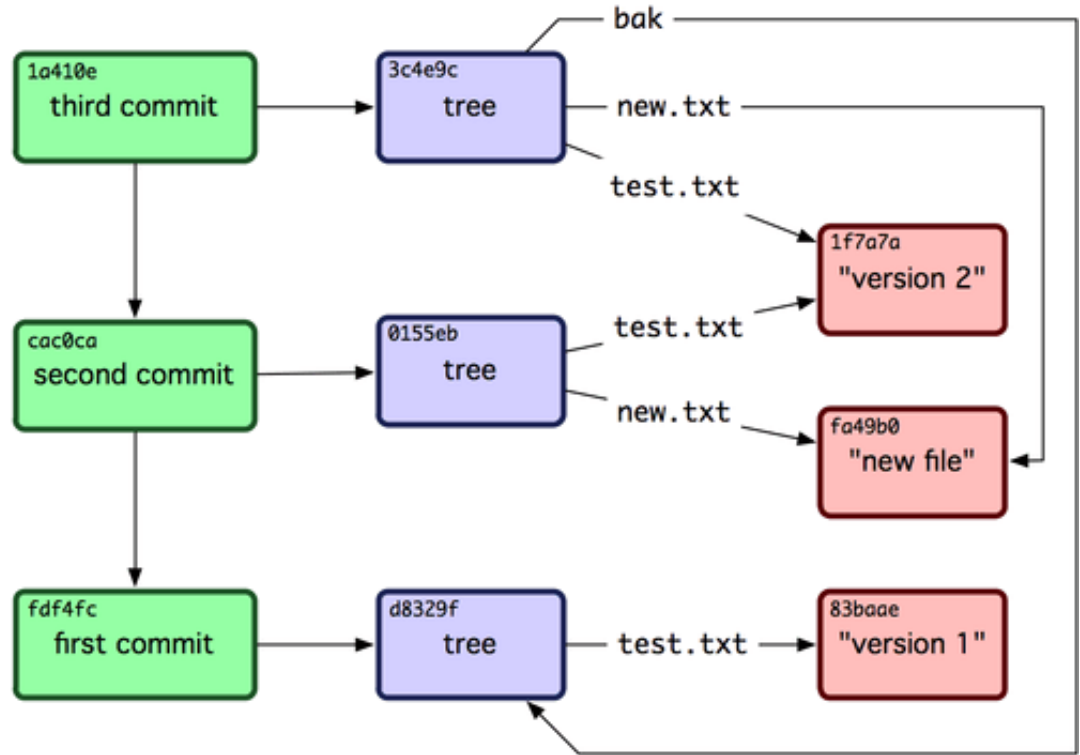
- Opens up to new workflows: git flow
- Each system has an exact replica of the repo as other collaborators.



<https://git-scm.com/images/about/workflow-b@2x.png>

# Under the hood

- Changes are stored in trees
- Trees contain changed files
- Commits contain trees



<http://git-scm.com/figures/18333fig0903-tn.png>



# GIT CONFIG

- So you can be held accountable, configure git

```
$ git config --global user.name "Your Name Comes Here"  
$ git config --global user.email you@yourdomain.example.com
```

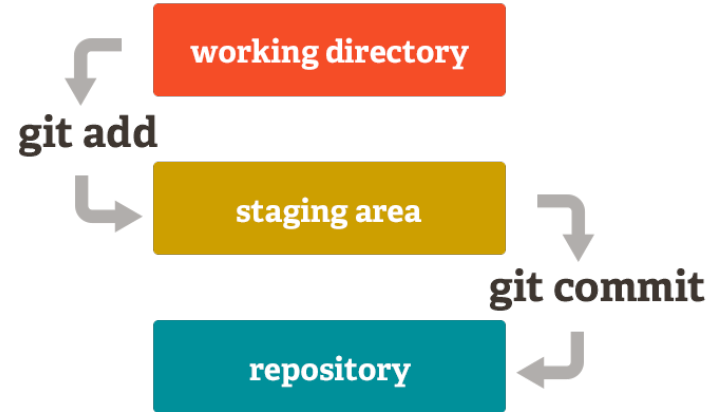
# GIT INIT

- Initializes a project directory with a hidden directory `/.git/`

```
$ tar xzf project.tar.gz  
$ cd project  
$ git init
```

# GIT ADD

- Add any files in your repository to git “stage”



<https://git-scm.com/images/about/index1@2x.png>

```
$ git add .
```

# STAGING AREA

## Working Directory

Changes not staged for commit:

modified: files.txt  
modified: have.txt

Untracked files:

been.txt  
changed.txt

## Staging Area

Changes to be committed:

modified: files.txt  
modified: have.txt

newfile: been.txt  
newfile: staged.txt

## Repository

[master (root-commit)  
3325sldd] "initial commit"

4 files changed,  
2 insertions(+)  
files.txt  
have.txt  
been.txt  
committed.txt

Hat tip: @GeorgiaReh

# GIT COMMIT

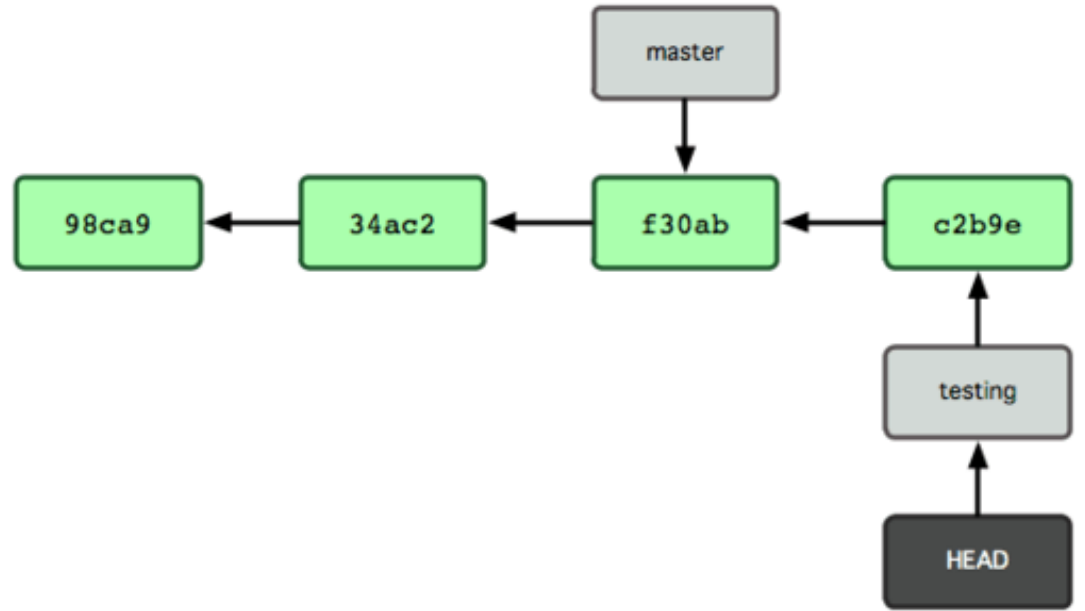
- Store your changes into a commit
- Saves all of your changes together / save point
- Commit does NOT push

```
$ git commit -m 'Initial commit'
```

# BRANCHING: Your safe place

- Makes a pointer to your code
- Moves HEAD around

```
$ git branch <name>  
$ git branch testing  
$ git commit -m "new"  
$ git checkout master
```



<http://git-scm.com/figures/18333fig0307-tn.png>

# MERGING

- `git merge <topic>`
- You must be on the branch you want to merge INTO when you execute this command (e.g. master)

```
$ git merge <branch>
```

# GOING BACKWARDS

- Generate a new commit that undoes all of the changes introduced in <commit>, then apply it to the current branch.

```
$ git revert <commit>
```



# SHARE YOUR CHANGES

- `git push <destination> <branch>`
- `git push origin master`

```
$ git remote add <name> <url>  
$ git push <name> <branch>  
$ git push origin master
```

# CLONING PROJECTS (SSH OR HTTPS)

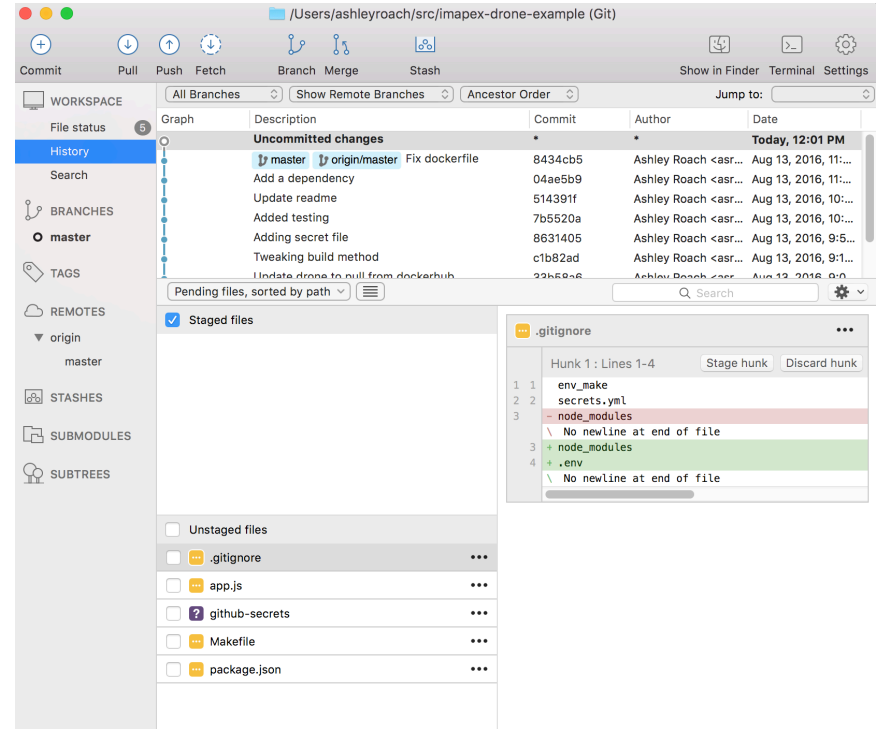
- No password
- Set up SSH key on remote server (e.g. ssh-keygen)

```
$ git clone git@github.com:aroach/upgraded-guacamole.git
```

```
$ git clone https://github.com/aroach/upgraded-guacamole.git
```

# GIT CLIENTS

- CLI Client
- IDE Clients
  - VIM: airblade/vim-gitgutter
  - Emacs: magit
- GUI Clients
  - SourceTree
  - Git Kracken



# Workshop

# Git Workshop Resources

<http://cs.co/git-us2017>

# Let's get started

<http://bit.ly/devnet-git-intro>

DevNet Learning Labs

Tracks Modules Labs Help Feedback

Collaboration Spark Beginner Python JSON Postman Login to Start →

- data (6)
- Data Center (13)
- dCloud (1)
- Deployment (1)
- DLP (5)
- DMo (1)
- DNA (4)
- Featured (2)
- Finesse (2)
- fog (1)
- Fun (2)
- git (3)
- Git (1)

**Git 100: Basics of the git version control system** 30 min

Learn how to use git for version control including configuration, commits, and differences.

Coding git version control Login to Start →

**Git 101: Branching** 30 min

Learn the basics of git branching so that you can diverge from the main code base without messing up the main code base.

Coding git version control Login to Start →

**Git 102: Using git with servers** 30 min

Learn how to use git with servers to share your work with others.

Coding git version control Login to Start →

# Forking and Pull Requests

- Login to GitHub
- Navigate to: <https://github.com/aroach/git-intro-fork-example>
- Click the fork button
- Create a new file with some dummy content
- Save it
- Submit a Pull Request (PR)

# Come find me after today... I'll be waiting

## Ashley Roach

 [asroach@cisco.com](mailto:asroach@cisco.com)

 [@aroach](https://twitter.com/aroach)

 <http://github.com/aroach>

 <http://linkedin.com/in/ashleyroach>

## Cisco DEVNET

 [@CiscoDevNet](https://twitter.com/CiscoDevNet)

 <http://github.com/CiscoDevNet>







**DEVNET**