Building Startup Systems

Objective

The goal of the Building Startup Systems series is to develop the skills necessary to turn great ideas into digital products.

The first course in the series is focused on mastery of key foundational skills by developing a pre-selected project using pre-selected tools and frameworks.

The second course in the series creates the experience of designing, implementing and deploying a workable system that is suitable for demo and early users.

Audience

Take this class if any of these statements are false:

- I can explain the difference between **git pull** and **git fetch**
- I use git rebase -i without fear of losing my code
- I have set up continuous integration and have written a circle.yml file
- I am can code and deploy an endpoint that accepts HTTP POST requests and calls to 3rd party APIs
- I know how to build a single-page-application (SPA) that can initiate two simultaneous HTTP requests and update a portion of the page only after both requests have completed.
- I know how to deploy an application in the cloud collect logs in a central location like CloudWatch

Logistics

- Class here every Weds 3:10 to 5:50
- Use Slack
- Pay attention + ask questions
- Adam Office Hours after class on Weds
- Rahul Office Hours before class on Weds 12:30-2:30pm

Grading

- Do the Homework
- Crush the Project
- No points for attendance

00. The Goal

01. The Tools

It takes software to make software

Install software	https://brew.sh/ or https://chocolatey.org/
Write code	Intellij IDEA Ultimate
Don’t lose your code	git
Compile code	Java 1.8
Build my project	gradle
Test every commit	CircleCI
Package my project for deployment	docker
Run my project in the cloud	AWS

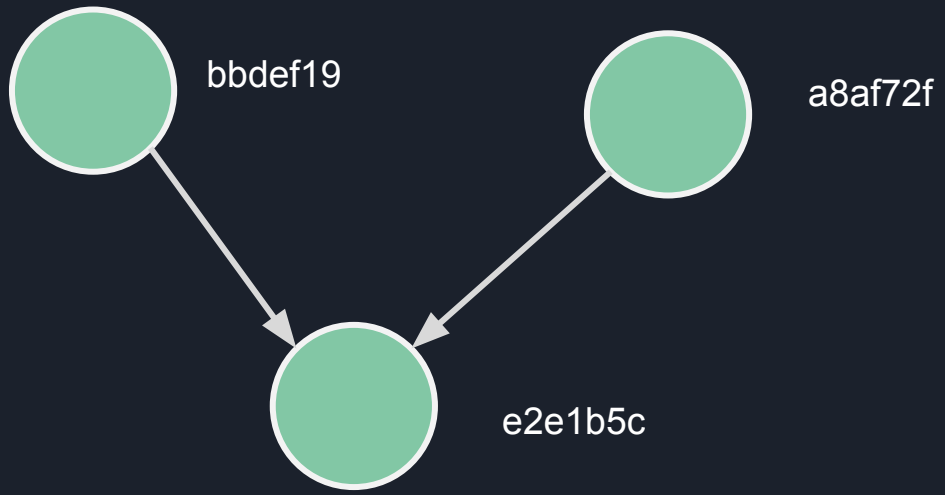
03. Git

Setup git

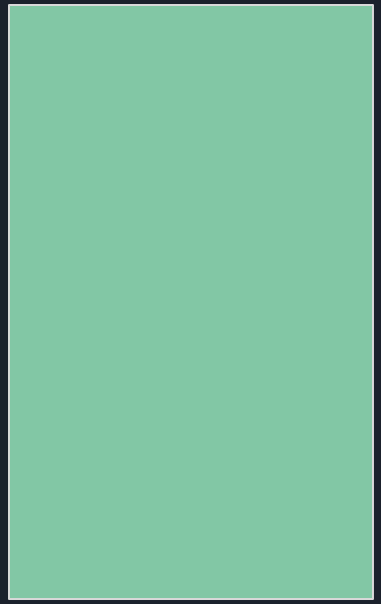
```
Put this in your ~/.gitconfig:     (http://bit.ly/2wEQhap)
--------------------------------


[alias]
l = log --color --graph --pretty=format:'%Cred%h%Creset
-%C(auto)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset'
--abbrev-commit
ff = fetch --all
```

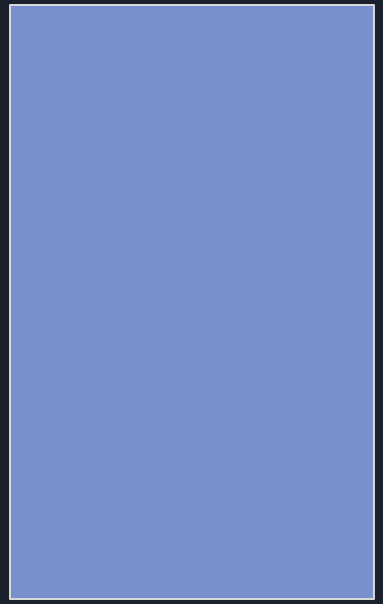
A commit is the fundamental unit of git



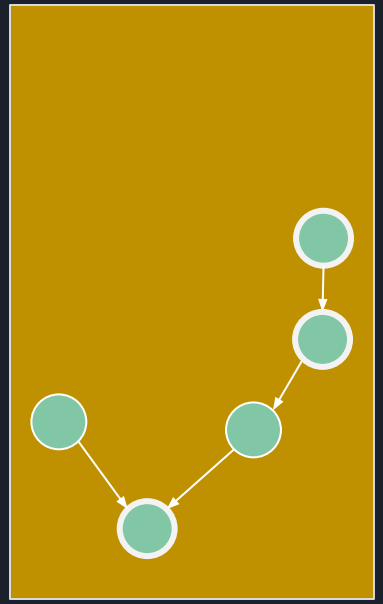
The change model is complex



Working Directory

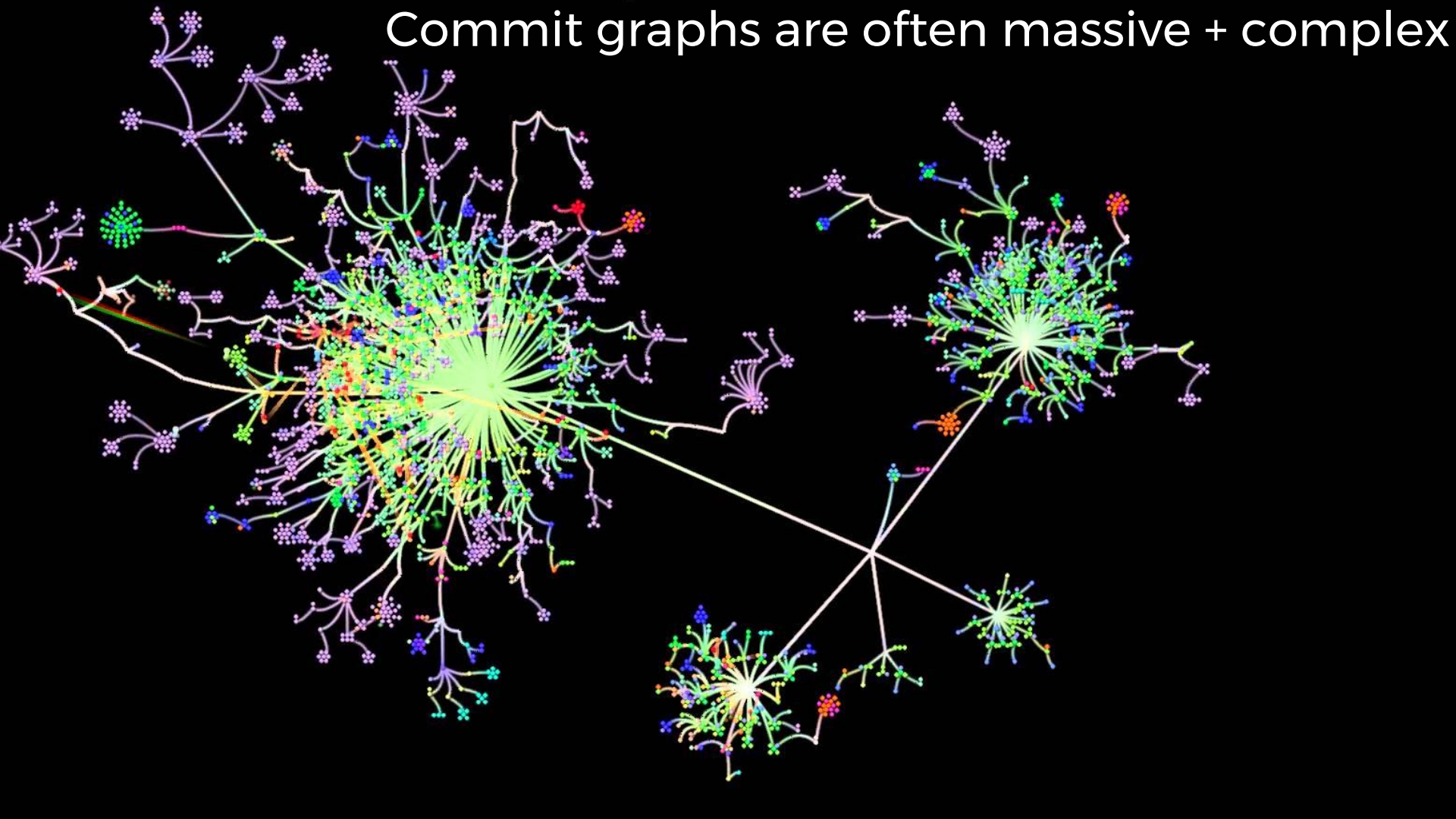


Index

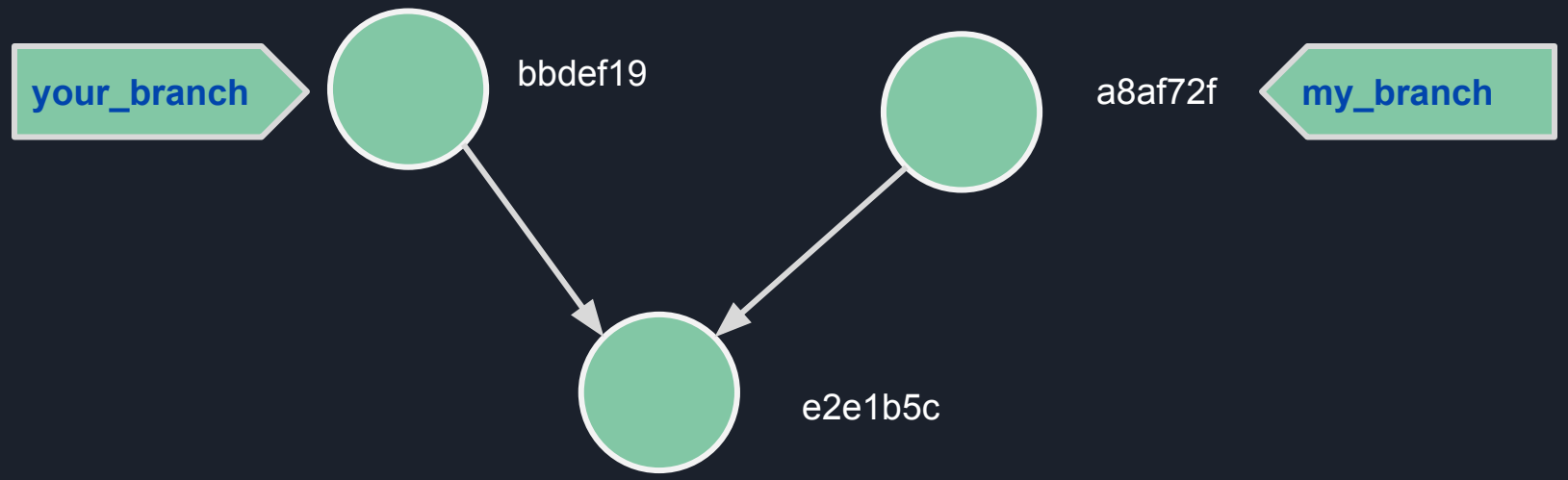


Local Repository

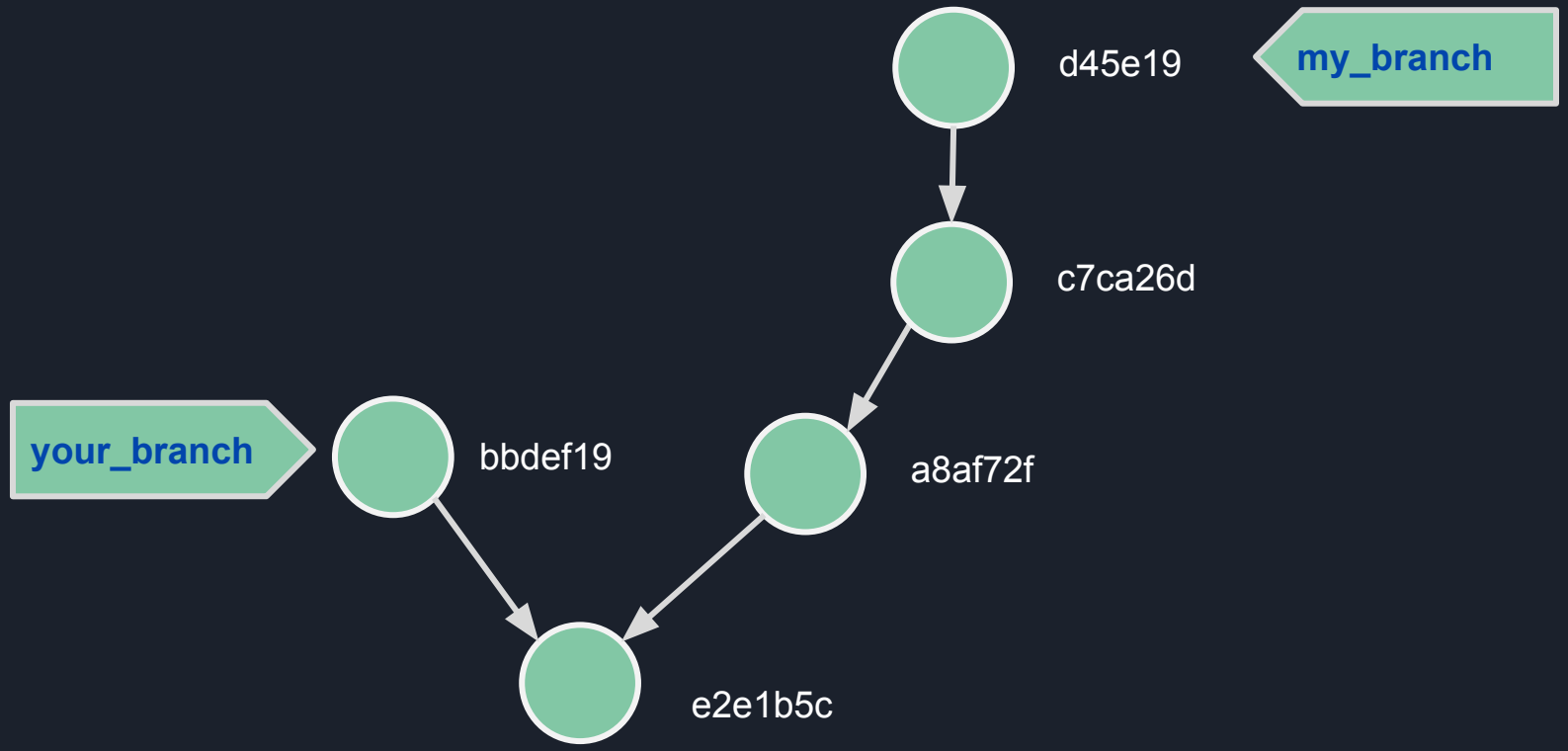
Commit graphs are often massive + complex



Branches are simply bookmarks in the graph



Branches should move with you as you code

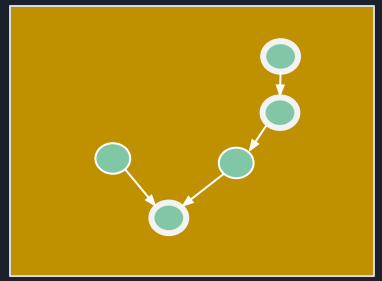


Important git state

Working Directory

Index

Local Repository



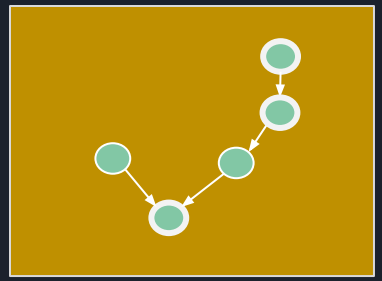
`HEAD`	The reference commit for diffs
`branch`	A bookmark. You may or may not be “on” a branch
`git diff`	Compare WD + Index vs. HEAD
`git add`	“Stage” a change in the index
`git commit`	Create a commit from index, move HEAD and attached branch

Important git state

Working Directory

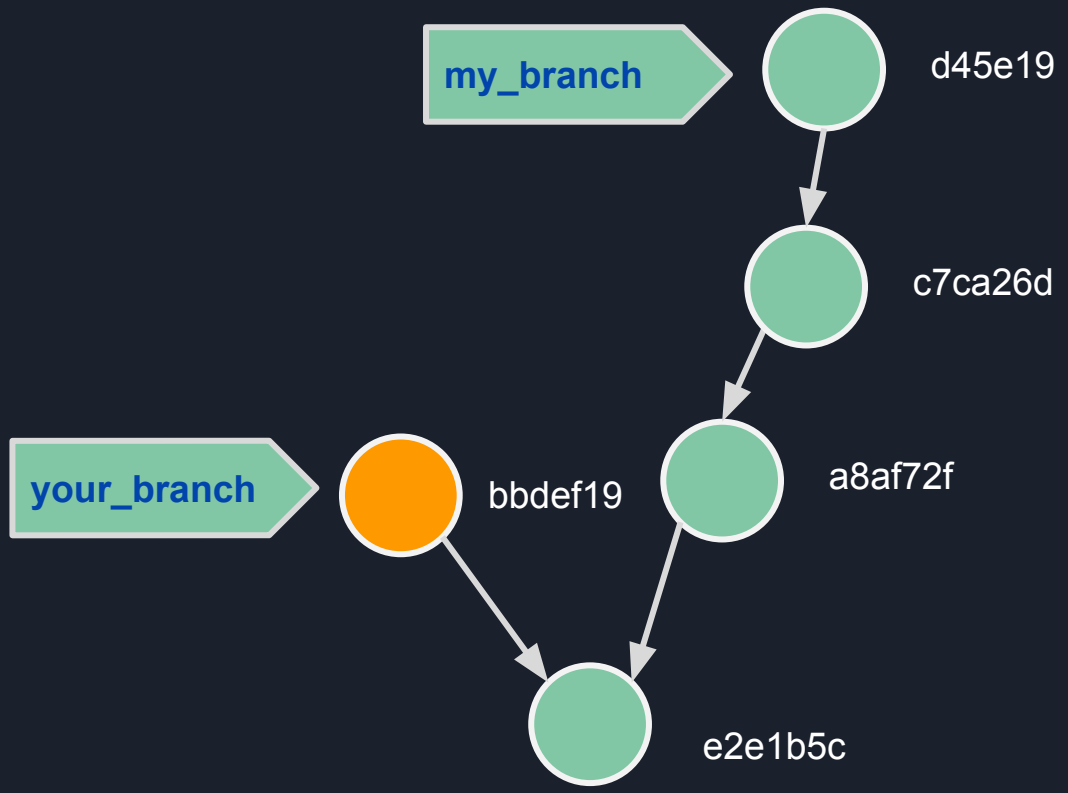
Index

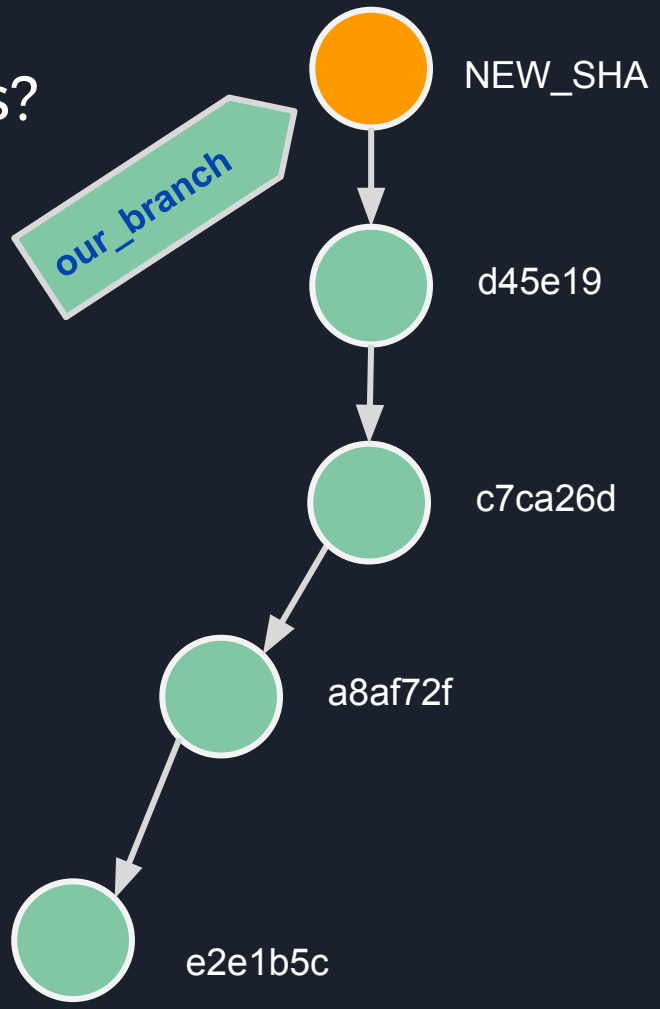
Local Repository



`git checkout`	Move HEAD, attach to new branch, update WD
`git reset`	Move HEAD + attached branch. DO NOT update WD
`git reset --hard`	Move HEAD + attached branch. DO update WD
`git fetch`	Update the local repository
`git pull`	Fetch and merge !DANGER!

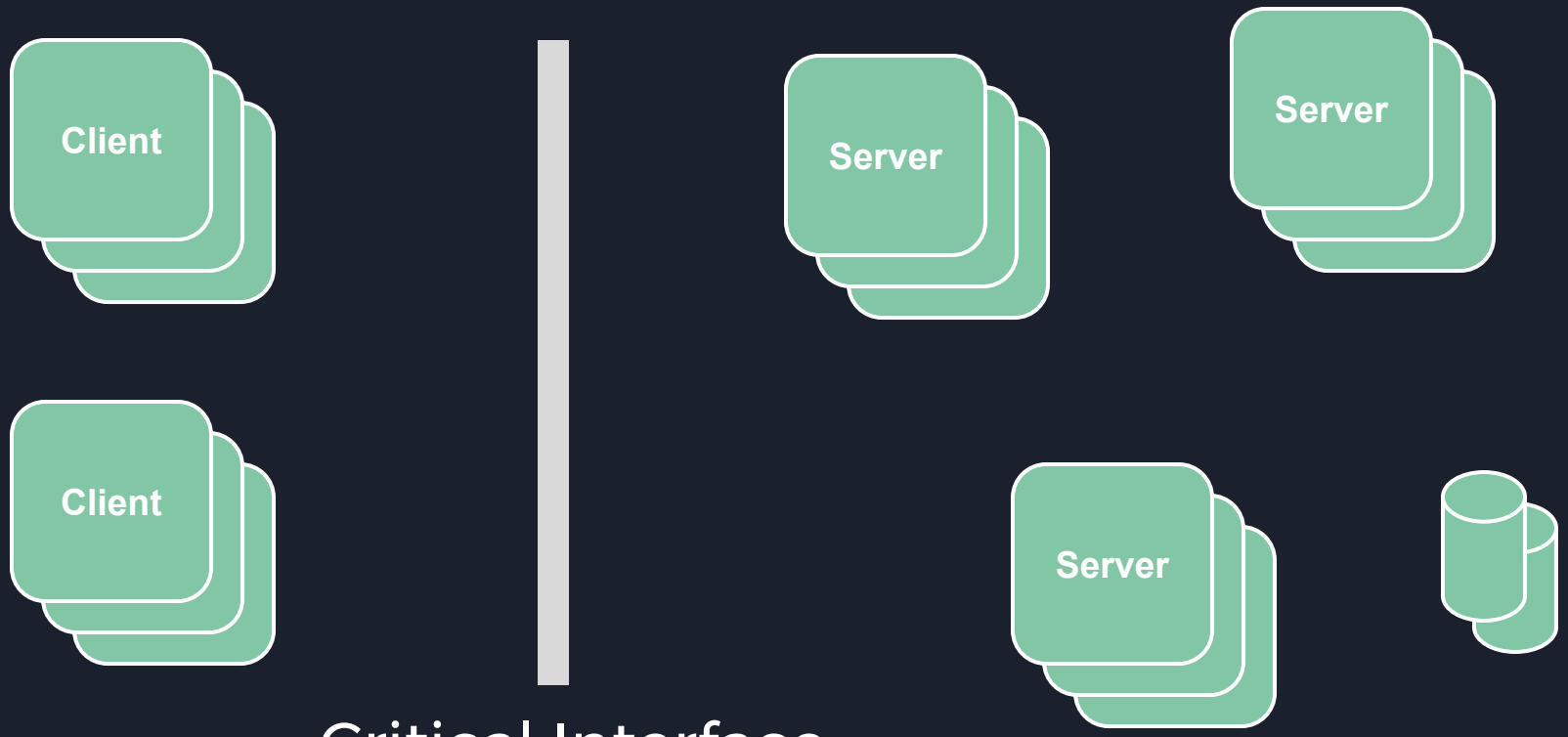
How do we combine branches?





04. Modern Apps

Multiple Front-Ends, Multiple Back-Ends



Critical Interface

REST is the idiomatic interface for modern apps

RESTful systems comply with a number of architectural constraints, which beget some highly desirable system properties: *performance, scalability, reliability, portability*

One constraint is having a *Uniform Interface*.

In modern apps this means using *HTTP verbs* to access and manipulate well-defined data collections. In REST we call each collection a *resource*

REST example

Your interface lives at http://www.mydomain.com/api

You would like your users to be able to access *receipts* so your interface supports manipulation of *receipts* using the /api/receipts endpoint

POST	Create a new receipt	Returns 200 + id of new receipt
PUT	Updates a receipt	Returns 200
GET	Gets the details of the receipt	Returns 200 + receipt data
DELETE	Deletes a receipt	Returns 200

REST has strong language support

JAX-RS is the Java API for Restful Web Services

JAX-RS is not working code, it is a spec that must be implemented

We will use Dropwizard, which is a combination of A+ libraries and utilities for building RESTful Java services

The mapping of REST concepts into your code is straightforward

http://shop.oreilly.com/product/0636920028925.do