Team Activity 1

- Meet your team!
- What are motivations?
- What are current beliefs?
- Skills? Gaps in skills?
- git, chef, docker, continuous integration, continuous delivery
- http://goo.gl/forms/9MnAO2bYw97Jp7VF3

After this activity you should:

- Understand some of the different motivations as to why folks are taking this class.
- Understand the skills and capabilities your team has to accomplish the goals of the class.

Team Activity 2

Careless Conversations (inspired by Alan Cyment)

- Pair up
- Select one person to go first.
- For one minute, speaker talks about something passionate about. Listener stays seated, quiet, and acts disinterested.
- Switch roles and repeat.
- Repeat until both people have done this twice.

After this activity you should:

- Understand what it feels like to be ignored.
- Understand the difference in communication when distracted.
- Importance of listening in communication.

Team Activity 3

Discuss with your team:

- What is the difference between a task and a project?
- Do you have interrupts? What are they?
- How do you determine when work is done?

Additional Background:

- Properties of Tasks
- Name
- Start Date

- End Date
- Current state
- Description
- Priority
- Owner
- Interrupts - non planned work that comes in
- customer request
- incident
- request for help from coworker
- single point of knowledge (you) work
- high priority task push from manager

After this activity you should:

- Have a better understanding of how individuals can classify tasks versus projects.
- Have a better understanding of interrupts.
- Understand the importance of defining done.

Team Activity 4

Talk through workflow for tasks for your team. How will you figure out what work needs to be done, who will work on the work, and when it is done. Use postits to mock up a legend for types of work items. Use pad to mock up your work items board. Use blue tape to mark off lanes.

- Discuss objectives.
- Describe work.
- Define lanes.
- Define a task.
- Define a project.

Kanban as a system is a visual process management tool that helps you to see:

- what to do
- when to do it
- how much of it to do

Kanban Principles

- Current process.
- Incremental, evolutionary change.
- Respects current processes, roles, responsibilities and titles.

Kanban Core Practices

- Visualize the work you do.
- Limit the amount of WIP.
- Manage the flow of work.
- Make policies explicit through documentation.
- Improve collaboratively.

Your states of work should reflect your current process, not the ideal process of what you want it to be. Incremental change can help you to get to a better set of processes and practices.

After this activity you should:

- Have an understanding of how to have this conversation with your own teammates.
- Built a primative kanban board.
- Know the fundamentals of kanban.

Verify Local Environment

Overview

In this exercise each participant configures and tests a local development environment.

Connect to Node (Everyone)

- ssh chef@NODE

If you don't have ssh available please downlowd an ssh client. For Windows, a good option is putty.

Introduction to Git

Customize your workstation (Everyone)

```
git config --global user.name "YOUR NAME"
git config --global user.email "YOUR EMAIL ADDRESS"
```

Example:

```
$ git config --global user.name "Jennifer Davis"
$ git config --global user.email "sparklydevops@gmail.com"
```

Verify .gitconfig creation (Everyone)

```
cat ~/.gitconfig
```

Example:

```
   cat ~/.gitconfig
[user]
   name = Jennifer Davis
   email = sparklydevops@gmail.com
```

Set your preferred git editor (Everyone)

If you don't set your preferred editor, it will use the default text editor for the system.

- emacs
- nano
- vi/vim

```
git config --global core.editor EDITORNAME
```

Example:

```
$ git config --global core.editor nano
```

Verify the configuration (Everyone)

```
git config --list
```

Example Output:

```
[chef@ip-172-31-11-246 hello_world]$ git config --list
user.name=Jennifer Davis
user.email=sparklydevops@gmail.com
core.editor=nano
core.repositoryformatversion=0
core.filemode=true
core.bare=false
```

Create a project Directory (Everyone)

```
mkdir wd
cd wd
```

Create initial project (Everyone)

```
mkdir hello_world
cd hello_world
git init
git status
```

Expected Output

```
[chef@ip-172-31-11-246 hello_world]$ git status
On branch master

Initial commit

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

Create hello.txt file:

```
echo "hello world" >> hello.txt
git status
git add hello.txt
git status
git commit -m "creation of hello.txt file"
git status
```

Example of first status output:

```
 git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

   hello.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Example of second status output:

```
[chef@ip-172-31-11-246 hello_world]$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
```

```
   new file:   hello.txt
```

Example of third status output:

```
[chef@ip-172-31-11-246 hello_world]$ git status
On branch master
nothing to commit, working directory clean
```

Create a GitHub identity (Everyone)

- If you don't already have a github account, create one.
- Browse to http://github.com. Supply a username, email address, and password.
- Free plan is fine. Other plans allow you to have private repositories.

(Optional) Setting up your Github keys (Everyone)

If you want to skip the added burden of entering your username and password each time at the prompt with git, you can follow the steps here to set up your ssh keys:

https://help.github.com/articles/generating-ssh-keys/

Example output of successful setup of keys:

```
 [chef@ip-172-31-11-246 wd]$ ssh -T git@github.com
Hi sparklydevops! You've successfully authenticated, but GitHub does not provide shell access.
```

Create a new remote repo

In a web browser, after logging into Github, you'll see an option to create a "New Repository" on the lower right.

Click on New Repository:

In the repository name field, enter `ed-lab1` and click the Initilize README option.:

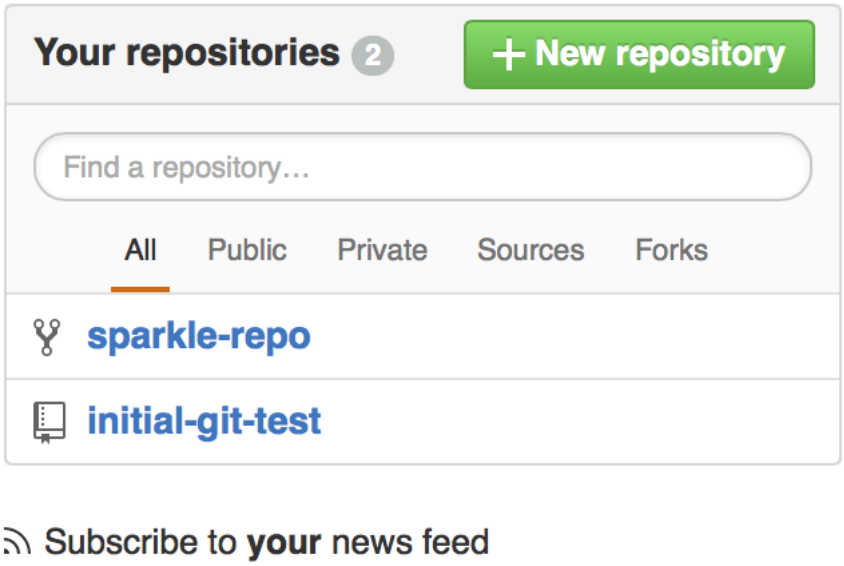
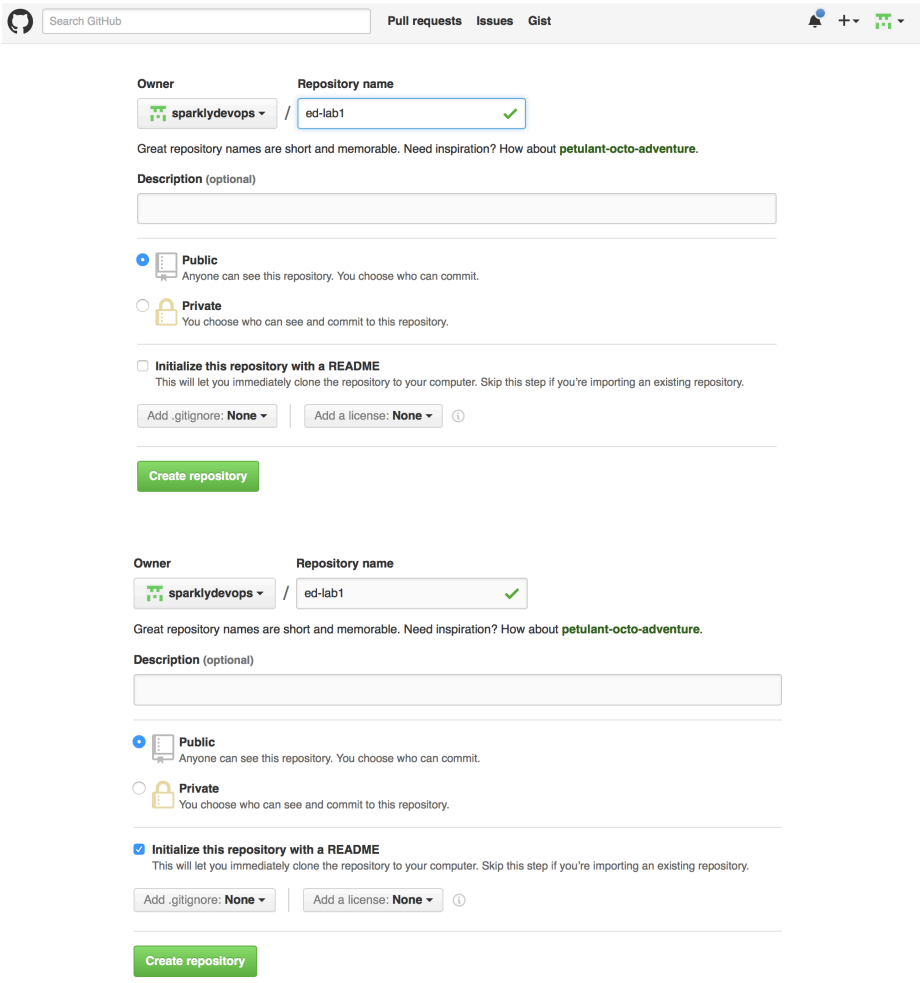


Figure 1: Once logged in, click New repository



Once you have created the repo, you will see a screen like this:

On the lower right you will see the clone url. You can leave it with https and enter your credentials each time, or you can switch to ssh by clicking on SSH if you've set up your ssh keys.

Clone new repo to node

On the node:

```
cd ~/wd
git clone git@github.com:YOURUSERNAME/ed-lab1.git
```

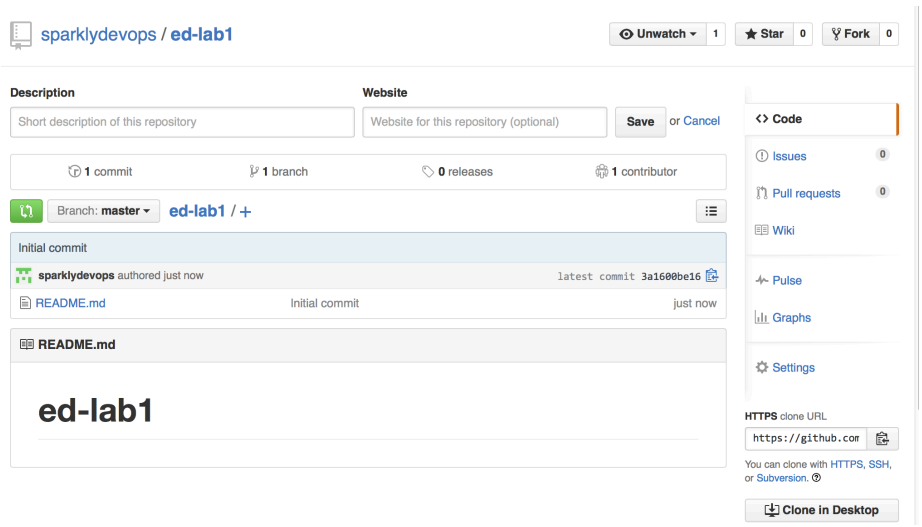


Figure 2: After clicking Create Repo

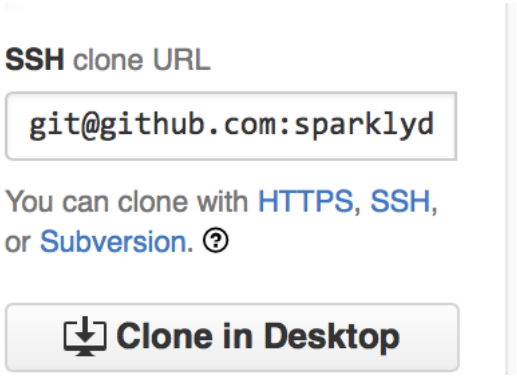


Figure 3: Click on SSH and grab the SSH clone url

Example:

```
[chef@ip-172-31-11-246 wd]$ git clone git@github.com:sparklydevops/ed-lab1.git
Cloning into 'ed-lab1'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
Checking connectivity... done.
```

Check the contents of the repository you just cloned

```
cd ~/wd/ed-lab1
ls
git remote -v
```

Add hello.txt to your ed-lab1 repo

Copy the hello.txt file from the previous exercise into your ed-lab1 repository.

```
cp ~/wd/hello_world/hello.txt .
git status
git add hello.txt
git status
git commit -m "adding hello.txt to repository"
```

Sharing updates to remote repository

At this point we have added the file to our local repository but not to our remote repository.

```
git push origin master
```

Verify update to github, by reloading your repository

Outcomes

- git configured on your node
- ed-lab1 repo created on github, with local repo cloned

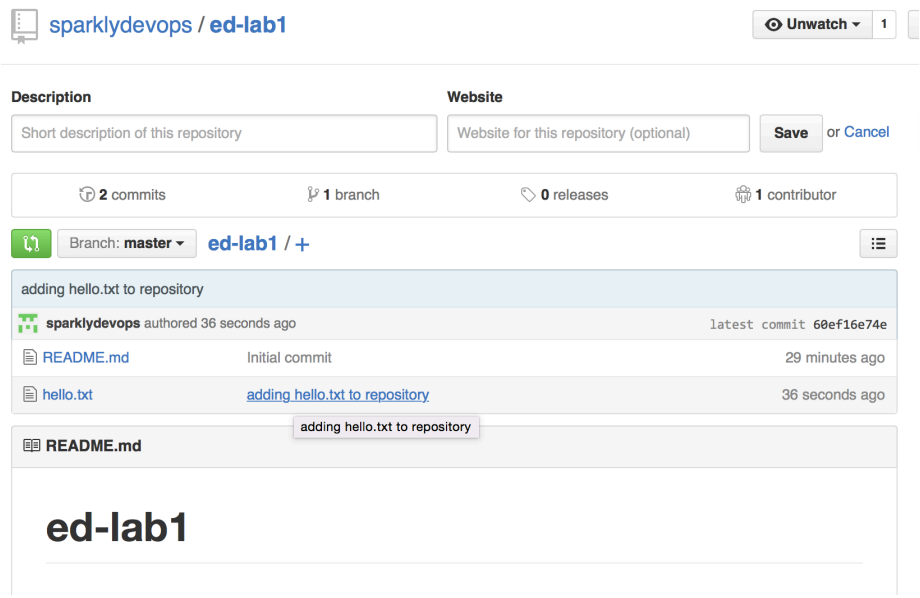


Figure 4: Verify updates to github

Verify Local Environment

Connect to Node

SSH is a generic term used for the SSH protocol. ssh is the client command for connecting to remote nodes. sshd is the server program that runs on the remote node that must be up and running to allow for connections.

ssh, Secure Shell is a way to connect to remote nodes. There are clients available on multiple operating systems. It is not a true shell like bash and tcsh. You need a username and password, or passphrase if using ssh keys.

We have pre-established a privileged account on the remote node, and providing you with a username and password. When you use the ssh client, whether ssh on Mac or a GUI like putty on Windows, you are connecting to that node.

Basic Unix Commands

This is not an exhaustive set of commands. If you are not familiar with the unix command line, this will help you understand what the commands we are asking to use do throughout the day.

Some of these commands have optional or required arguments. You can find out more about any command using the `man COMMAND`.

- man - Access the manual page about a command.
- pwd - present working directory. It gives you information about the current location on the file system.
- cd - change your current directory.
- ls - lists the files and directories in the current working directory
- mkdir - create a directory
- touch -
- mv - can move a file into a directory, or rename a file/directory to a new name
- rm - remove files and directories
- cp - copies files and directories.
- cat - concatenates and prints file to standard output

There is a tutorial available at Linuxcommand.org.

Introduction to Version Control

- Background for Version Control

Introduction to Git

- git basics
- Windows or Mac git GUI client

Git configuration

- More information about git configuration

Set your preferred git editor

If you don't have a preferred editor, use nano. nano is an easy editor to get started with. The basic commands:

Open a file for editing

```
$ nano FILENAME
```

Save a file after editing

```
CTRL+x, "y", ENTER
```

Exit a file

`CTRL+x`

Sharing updates to remote repository

Basic git commands: push, pull, fetch

- Git Push
- Git Pull
- Git Fetch

Team Activity 5

In pairs, discuss your current work environment.

- Who are the members of your team (at work)?
- What version control system do you use?
- Who are the people who have commit access?
- What's the flow of code from design to deploy?

While one person shares their environment, the other person should draw a diagram to represent the information shared. Use circles to represent people, triangles to represent code repos, and rectangles to represent infrastructure. Use arrows to represent relationships and flow.

After this activity you should:

- From looking at your version control setup, understand the flow of access control.
- See where bottlenecks and improvements may be helpful.

Adding Collaboration

Have one person in your team create a repo called ed-lab2.

- Elect one person from your group to create a repo called ed-lab2. Follow the process from Lab 1.

Grant privileges to the ed-lab2 repo to your team.

- Share your github identity with the team
- Click on the gear to access the Settings tab
- Click on Collaborators
- Add Collaborators based on github name

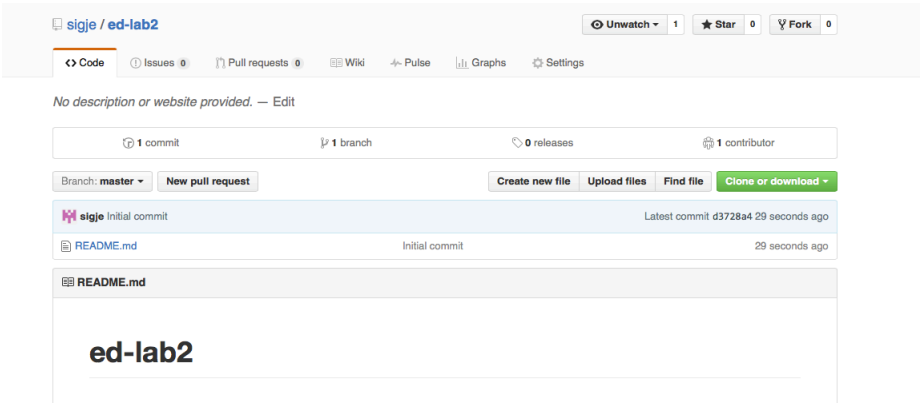


Figure 5: Click on the gear to access the Settings tab

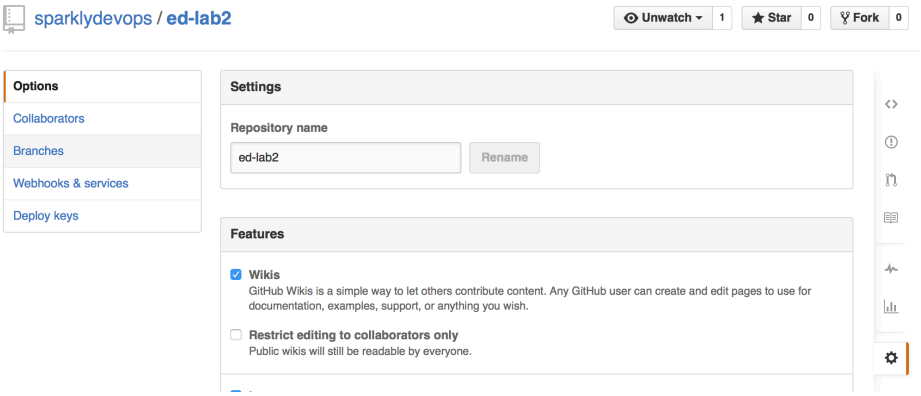


Figure 6: Click on Collaborators

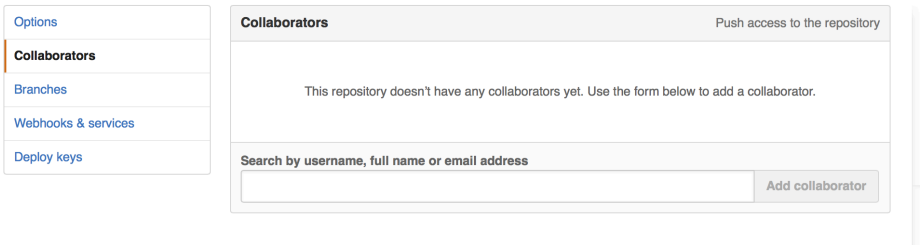


Figure 7: Add Collaborators based on github name

- If you have successfully added, they will show up as a Collaborator.

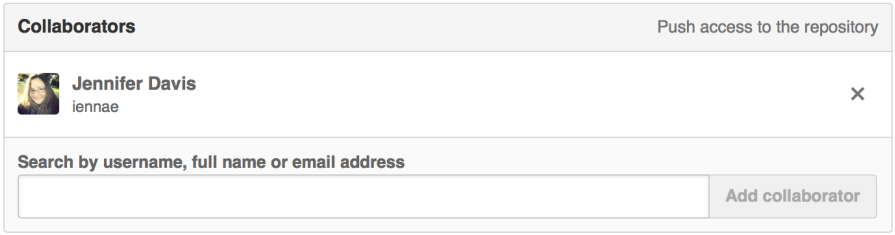


Figure 8: If you have successfully added, they will show up as a Collaborator

Clone the ed-lab2 repo to your node

Clone your teams repo to your node. Add a file to the repo with your firstname as the file name. Replace FIRSTNAME in the following with your first name.

```
cd ~/wd
git clone git@github.com:USERNAME/ed-lab2.git
cd ~/wd/ed-lab2
touch FIRSTNAME.txt
git add FIRSTNAME.txt
git commit -m "adding my first name"
```

Sync the remote repo

Push your changes to the remote repository.

```
git push origin master
```

As individuals add their name to the repo, you will run into conflicts. This is due to your local repo not being up to date with the remote repo as your team makes changes.

Example Merge Conflict output:

```
[chef@ip-172-31-11-246 ed-lab2]$ touch jennifer.txt
[chef@ip-172-31-11-246 ed-lab2]$ git add jennifer.txt
[chef@ip-172-31-11-246 ed-lab2]$ git commit -m "adding my first name"
[master 00a017a] adding my first name
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 jennifer.txt
[chef@ip-172-31-11-246 ed-lab2]$ git push origin master
To git@github.com:sparklydevops/ed-lab2.git
 ! [rejected]        master -> master (fetch first)
```

```
error: failed to push some refs to 'git@github.com:sparklydevops/ed-lab2.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Work with your team to plan your updates into the repo so that you don't have to repeat this process multiple times. Do a `git pull` to sync your local repo to the to the remote repo.

```
git pull origin master
git push origin master
```

Example output:

```
Merge branch 'master' of github.com:sparklydevops/ed-lab2

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.

[chef@ip-172-31-11-246 ed-lab2]$ git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:sparklydevops/ed-lab2
 * branch            master     -> FETCH_HEAD
   8451292..0d5f3db  master     -> origin/master
Merge made by the 'recursive' strategy.
 jay.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 jay.txt

 $ git push origin master
Warning: Permanently added the RSA host key for IP address '192.30.252.129' to the list of known
 Counting objects: 4, done.
 Compressing objects: 100% (4/4), done.
 Writing objects: 100% (4/4), 591 bytes | 0 bytes/s, done.
 Total 4 (delta 0), reused 0 (delta 0)
 To git@github.com:sparklydevops/ed-lab2.git
    0d5f3db..fdde3d1  master -> master
```

Outcomes

- Creation of ed-lab2 repo
- First name .txt files created for each of your group within the ed-lab2 repo.

Adding Collaboration

For more resources on collaboration with git:

- Adding collaborators to a personal repository
- Tutorial from Atlassian on Collaboration
- Different workflows - Atlassian tutorial
- Git Workflows Book

Sharing updates to remote repository

Basic git commands: push, pull, fetch

- Git Push
- Git Pull
- Git Fetch

Setup a Chef Repository

Before starting any work in this lab, make sure that you read through the entire requirements as a team. Define which items are tasks, which are dependent tasks, and what tasks could potentially cause conflicts in your repository.

You can track this on your kanban board.

Have one person in your team create a repo called ed-lab3.

- Elect one person from your group to create a repo called `ed-lab3`. Follow the process from Lab 1 with one exception, don't create a readme.

Grant privileges to the ed-lab3 repo to your team.

Follow the process from Lab 2 to grant access to your full team.

Generate the chef repo.

In this exercise we are creating a monolithic chef-repo. The monolithic chef-repo will hold the global policy items as well as any cookbooks created in this workshop. Generally, when using with a Chef server you choose between one of two strategies.

You can learn more about these supported workflows at this Chef RFC: https://github.com/chef/chef-rfc/blob/master/rfc019-chef-workflows.md

1. Monolithic Workflow - 1 chef-repo containing everything that maps to 1 repo in source control.
2. 1 cookbook per repo + policy only chef-repo.

```
cd ~/wd
chef generate repo ed-lab3
cd ~/wd/ed-lab3
git init .
git add .
git commit -m "Initial creation of chef repo"
```

Save the changes and commit back to the ed-lab3 repo created.

```
git remote add origin git@github.com:USERNAME/ed-lab3.git
git push -u origin master
```

Example Output

```
[chef@ip-172-31-11-246 ed-lab3]$ git add .
[chef@ip-172-31-11-246 ed-lab3]$ git commit -m "initial creation of chef repo"
[master (root-commit) a590574] Initial creation of chef repo
 16 files changed, 351 insertions(+)
 create mode 100644 .chef-repo.txt
 create mode 100644 .gitignore
 create mode 100644 LICENSE
 create mode 100644 README.md
 create mode 100644 chefignore
 create mode 100644 cookbooks/README.md
 create mode 100644 cookbooks/example/README.md
 create mode 100644 cookbooks/example/attributes/default.rb
 create mode 100644 cookbooks/example/metadata.rb
 create mode 100644 cookbooks/example/recipes/default.rb
 create mode 100644 data_bags/README.md
 create mode 100644 data_bags/example/example_item.json
 create mode 100644 environments/README.md
 create mode 100644 environments/example.json
 create mode 100644 roles/README.md
 create mode 100644 roles/example.json
[chef@ip-172-31-11-246 ed-lab3]$ git remote add origin git@github.com:sparklydevops/ed-lab3.gi
```

```
[chef@ip-172-31-11-246 ed-lab3]$ git push -u origin master
Counting objects: 24, done.
Compressing objects: 100% (20/20), done.
Writing objects: 100% (24/24), 5.71 KiB | 0 bytes/s, done.
Total 24 (delta 2), reused 0 (delta 0)
To git@github.com:sparklydevops/ed-lab3.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Clone the ed-lab3 repo to your node

If you didn't have the task to generate the `ed-lab3` chef-repo, clone your teams repo to your node.

Replace *USERNAME* with the github identify of the person who created the repo.

```
cd ~/wd
git clone git@github.com:USERNAME/ed-lab3.git
cd ~/wd/ed-lab3
```

Split your team up into pairs, max of 3 people per "pair"

- Driver - person working with keyboard
- Observer - person looking at the code

Work with your team to plan your updates into the repo so that you can do the `git pull`, `git push` to sync your local repo to the to the remote repo as needed.

Create application cookbook - Pair 1

The *driver* will type out the commands. The *observer* will verify for mistakes.

```
cd ~/wd/ed-lab3/cookbooks
chef generate cookbook app
git add app
git commit -m "creation of app cookbook"
git push origin master
cd ~/wd/ed-lab3/cookbooks/app
```

Install Apache via a Recipe - Pair 1

The *driver* and *observer* should switch roles. The *driver* will type out the commands. The *observer* will verify for mistakes.

Generate the `install_apache` recipe in the `app` cookbook.

```
cd ~/wd/ed-lab3/cookbooks/app
chef generate recipe . install_apache
nano recipes/install_apache.rb
```

Add the follow `resources` to the `install_apache` recipe.

```
package 'httpd'

service 'httpd' do
  action [ :enable, :start ]
end
```

Edit the default recipe:

```
nano recipes/default.rb
```

Update the contents of the `default.rb` recipe with the following contents:

```
include_recipe 'app::install_apache'
```

Update the `.kitchen.yml`.

```
nano .kitchen.yml
```

Update the `.kitchen.yml` configuration.

- Change the driver name to *docker*.
- Delete the ubuntu platform
- Modify centos to centos-6.5. Avoid complexity of systemd and RHEL
- Add the forwarded ports section.

Update the contents of your .kitchen.yml to match:

```
driver:
  name: docker

## The forwarded_port port feature lets you connect to ports on the VM guest via
## localhost on the host.
## see also: https://docs.vagrantup.com/v2/networking/forwarded_ports.html

network:
    - ["forwarded_port", {guest: 80, host: 80}]

provisioner:
```

```
  name: chef_zero

## require_chef_omnibus specifies a specific chef version to install. You can
## also set this to `true` to always use the latest version.
## see also: https://docs.chef.io/config_yml_kitchen.html

#  require_chef_omnibus: 12.5.0

platforms:
  - name: centos-6.5
    driver_config:
      forward:
      - 80:80

suites:
  - name: default
    attributes:
```

```
platforms:
  - name: centos-6.5
    driver_config:
      forward:
      - 80:80
```

`kitchen login` and verify on the docker image directly.

```
chef install
kitchen converge
```

How do you know if your recipe worked? Kitchen converge finishes without errors, and you have a port up and running. You can check by browsing directly to the node because you have set up port forwarding!

Validate your node has apache up and running:

```
curl localhost
```

Once you have verified that you have Apache up and running, commit your changes to your local and remote repositories.

```
cd ~/wd/ed-lab3/cookbooks
git add app
git commit -m "install and configure apache"
git push origin master
```

Include mysql cookbook from supermarket - Pair 2

This step depends on the app cookbook being created by Pair 1. Make sure that you have pulled from the remote repository.

```
cd ~/wd/ed-lab3
git pull origin master
cd ~/wd/ed-lab3/cookbooks/app
```

If you get an error with the *app* directory not existing, coordinate with the first Pair on your team.

Supermarket is the Chef community site. Before using community cookbooks in your environment, always inspect the cookbook. You run the code with root privileges!

Edit the default recipe:

```
nano ~/wd/ed-lab3/cookbooks/app/recipes/default.rb
```

Update the contents of the `default.rb` recipe with the following contents:

```
include_recipe 'app::mysql_service'
```

We will use the mysql cookbook. The mysql cookbook is a library cookbook, and contains no recipes. It only has resources that we can use that extend the available resources to manage. We will use the mysql_service resource. You can also read the examples in the mysql cookbook README to understand how to use the other available mysql resources.

Generate the `mysql_service` recipe in the `app` cookbook.

```
chef generate recipe . mysql_service
nano recipes/mysql_service.rb
```

Add the follow `resource` to the `mysql_service` recipe.

```
mysql_service 'joengo' do
  port '3306'
  version '5.5'
  initial_root_password 'banana'
  action [:create, :start]
end
```

Edit the *metadata.rb* file to add a dependency on the mysql community cookbook. In your production environment, you would validate the contents of mysql prior to using it in a deployment scenario.:

```
nano metadata.rb
```

Update the contents of the `metadata.rb` file.:

```
depends 'mysql', '~> 6.0'
```

Update the `.kitchen.yml`.

```
 nano .kitchen.yml
```

Update the `.kitchen.yml` configuration.

- Change the driver name to *docker*.
- Delete the ubuntu platform
- Modify centos to centos-6.5. Avoid complexity of systemd and RHEL 7.
- Uncomment out the forwarded port section.

Update the contents of your .kitchen.yml to match:

```
driver:
  name: docker

## The forwarded_port port feature lets you connect to ports on the VM guest via
## localhost on the host.
## see also: https://docs.vagrantup.com/v2/networking/forwarded_ports.html

network:
    - ["forwarded_port", {guest: 80, host: 80}]

provisioner:
  name: policyfile_zero

## require_chef_omnibus specifies a specific chef version to install. You can
## also set this to `true` to always use the latest version.
## see also: https://docs.chef.io/config_yml_kitchen.html

#  require_chef_omnibus: 12.5.0

platforms:
  - name: centos-6.5
    driver_config:
      forward:
      - 80:80
suites:
  - name: default
    attributes:
```

Solve dependency constraints, install 3rd party cookbooks. `chef install` will have a `Policyfile.lock.json` as output. `kitchen converge` will set up docker container, install chef (if needed), and converge based on the runlist as descirbed in `Policyfile.rb`.

```
chef install
kitchen converge
```

How do you know if your recipe worked? Kitchen converge finishes without errors, and mysql is up and running. You can check by browsing directly to the node because you have set up port forwarding!

```
cd ~/wd/ed-lab3/cookbooks
git add app
```

```
git commit -m "install and configure mysql"
git push origin master
```

Verify your development environment is consistent

Update your development enviornment.

```
cd ~/wd/ed-lab3
git pull origin master
```

Outcome

You should have an updated *ed-lab3* with

- chef repo with mysql and apache recipes.

Hints in the Lab 3 background

chef generate

- Creating your own Chef Cookbook Generator
- Policyfiles - Older blog post.
- Automating Infrastructure with Chef - intro deck to resources.

Translate a runbook for installing MongoDB into chef

MongoDB is an open-source, document-oriented database designed for ease of development and scaling. The MongoDB documentation site includes a installation guide on how to install MongoDB on Red Hat Enterprise Linux, CentOS Linux, Fedora Linux, or a related system.

- Chef Resource Documentation

Read the installation guide, identify the resources you need, and create a mongodb cookbook populated with an appropriate recipe.

- What users do you need? What groups?
- What packages do you need?
- What configurations do you need?
- Where is data stored?
- What directories do you need to create?
- What services do you need?

Hints:

Use the yum community cookbook

The yum community cookbook provides resource *yum_repository* that allows us to make individual yum repositories available for use.

In the installation guide, we see that we need to configure a yum repo in order to install mongodb packages using yum.

```
[mongodb-org-3.2]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.2/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.2.asc
```

translates to chef dsl as:

```
yum_repository 'mongodb-org-3.2' do
  description 'MongoDB Repository'
baseurl 'https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.2/x86_64/'
  gpgcheck true
  gpgkey 'https://www.mongodb.org/static/pgp/server-3.2.asc'
  enabled true
  action :create
```

Outcome

- mongodb running on the node
- code checked into repository

Creation of MongoDB cookbook

```
cd ~/wd/ed-lab3/cookbooks
git pull origin master
chef generate cookbook mongodb
cd ~/wd/ed-lab3/cookbooks/mongodb
```

Update kitchen.yml to use centos 6.5, docker

Create template to hold the mongodb repo location

```
chef generate template mongodb.repo
```

Commit cookbook to source control

```
git add ~/wd/ed-lab3/cookbooks/mongodb
git commit -m "creation of cookbook mongodb"
git push origin master
```

Creation of install_mongo.rb recipe

Determine the package(s) needed.

Update default.rb to include install_mongo.rb recipe

Update mongodb.repo.erb template

From Mongo Installation guide:

```
For the latest stable release of MongoDB
Use the following repository file:

[mongodb-org-3.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.0/x86_64/
gpgcheck=0
enabled=1
```

If you need further hints, you can check out this working example https://github.com/nathenharvey/install_mongo of minimal requirements to install mongodb.