SE-441 Continuous Delivery and DevOps
Winter 2018-2019
Homework 7
Due On: March 7, 2019
100 points

Please submit your answers to D2L by March 7, 2019.

Assignment

This week you will automate the setup of the Spring PetClinic locally under Vagrant. You will also setup a deployment to Heroku, a simple cloud service provider. Finally, you will walk through a very simple Chef tutorial.

Vagrant

You will first use Vagrant to define a new virtual machine with the Spring PetClinic already installed.

Install Vagrant

If you don't already have Vagrant installed, download it an install it. These instructions are based off of the current version of Vagrant. At the time of writing that version was 1.9.1.

1. Visit `https://www.vagrantup.com/downloads.html`.
2. Download the current version of Vagrant for your operating system.
3. Run the setup file to install the software. You should be able to accept the defaults.

Note: *For Vagrant, or any other kind of virtualization to work, you may need to manually enable virtualization from your BIOS settings typically available via some kind of function key combination when your machine starts up. Check your machine's documentation for specific instructions.*

Create a New Box

In this section you will prepare a box for use in automation.

1. Create a new box using the following command:

   ```
   vagrant box add centos/7
   ```

   You will see output similar to the following:

   ```
   ==> box: Loading metadata for box 'centos/7'
   box: URL: https://atlas.hashicorp.com/centos/7
   This box can work with multiple providers! The providers that it can work
   ↪ with are listed below. Please review the list and choose the provider
   ↪ you will be working with.
   ```

```
1) libvirt
2) virtualbox

Enter your choice: 2
```

Choose the embedded VirtualBox provider as shown. You will then see output similar to the following as the OS image is downloaded:

```
==> box: Adding box 'centos/7' (v1611.01) for provider: virtualbox
box: Downloading: https://atlas.hashicorp.com/centos/boxes/7/versions/
↪  1611.01/providers/virtualbox.box
box: Progress: 100% (Rate: 509k/s, Estimated time remaining: --:--:--)
==> box: Successfully added box 'centos/7' (v1603.01) for 'virtualbox'!
```

The time required to download the base image will depend on your network connection and any other traffic that's consuming your network resources.

Create a Vagrantfile

In this section you will prepare a "Vagrantfile" and use it to automate the creation of a simple VM for your project.

1. Open a command prompt.
2. Switch to your Spring PetClinic project directory and create a new "Vagrantfile" using the following command:

   ```
   vagrant init
   ```

   You should see output similar to the following:

   ```
   A `Vagrantfile` has been placed in this directory. You are now
   ready to `vagrant up` your first virtual environment! Please read
   the comments in the Vagrantfile as well as documentation on
   `vagrantup.com` for more information on using Vagrant.
   ```

3. Create a file called **bootstrap.sh**:

   ```
   #!/usr/bin/env bash

   # update all existing packages
   sudo yum -y update

   # install wget and dos2unix
   sudo yum -y install wget
   sudo yum -y install dos2unix

   # download java
   ```

```
sudo wget -c --header "Cookie: oraclelicense=accept-securebackup-cookie"
↪  http://download.oracle.com/otn-pub/java/jdk/8u131-
↪  b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm -O
↪  /tmp/jdk-8-linux-x64.rpm

# install java
sudo yum -y install /tmp/jdk-8-linux-x64.rpm
sudo rm /tmp/jdk-8-linux-x64.rpm
```

4. Edit **Vagrantfile** so that it appears similar to the following:

```
...
Vagrant.configure(2) do |config|
...
config.vm.box = "centos/7"
...
config.vm.network "forwarded_port", guest: 8080, host: 8484
...
config.vm.synced_folder ".", "/home/vagrant/sync", disabled: true
config.vm.provision :shell, path: "bootstrap.sh"
config.vm.provision :file,  source: "target/spring-petclinic-1.4.2.jar",
↪  destination: "/tmp/spring-petclinic-1.4.2.jar", run: "always"
config.vm.provision :shell, inline: "java -jar
↪  /tmp/spring-petclinic-1.4.2.jar &", run: "always"
...
end
```

Note: *Part of the configuration requires that the spring-petclinic.jar be available in the target directory. Make sure you have built the JAR using Maven before you try the next step or it will fail.*

5. Start the VM:

```
vagrant up
```

The first time you start the VM you may see output like the following:

```
==>  Provider 'virtualbox' not found. We'll automatically install it
↪  now...
The installation process will start below. Human interaction may be
required at some points. If you're uncomfortable with automatically
installing this provider, you can safely Ctrl-C this process and install
it manually.
==>  Downloading VirtualBox 5.0.10...
This may not be the latest version of VirtualBox, but it is a version
that is known to work well. Over time, we'll update the version that
is installed.
Progress: 100% (Rate: 2160k/s, Estimated time remaining: --:--:--)
==>  Installing VirtualBox. This will take a few minutes...
A couple pop-ups will occur during this installation process to
```

```
ask for admin privileges as well as to install Oracle drivers.
Please say yes to both. If you're uncomfortable with this, please
install VirtualBox manually.
==>  VirtualBox has successfully been installed!
```

If you don't already have Oracle VirtualBox installed, you will be prompted to do so. Accept the defaults and it will install VirtualBox for you. If VirtualBox is already installed it will likely be able to use that installation.

Note: *If you are on a windows environment, you must disable the hyperv feature within Windows for this to work. Otherwise you will likely get a Blue Screen Of Death when Vagrant attempts to start the VM.*

After VirtualBox has been installed, an on subsequent starts of the VM, you should see lots of output that starts with the following:

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'centos/7'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'centos/7' is up to date...
==> default: Setting the name of the VM:
 ↪  vagrant_default_1463163881927_54788
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
default: Adapter 1: nat
==> default: Forwarding ports...
default: 8080 (guest) => 8484 (host) (adapter 1)
default:   22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
default: SSH address: 127.0.0.1:2222
default: SSH username: vagrant
default: SSH auth method: private key
default: Warning: Remote connection disconnect. Retrying...
default:
default: Vagrant insecure key detected. Vagrant will automatically replace
default: this with a newly generated keypair for better security.
default:
default: Inserting generated public key within guest...
default: Removing insecure key from the guest if it's present...
default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
default: No guest additions were detected on the base box for this VM!
 ↪  Guest
default: additions are required for forwarded ports, shared folders, host
 ↪  only
```

```
default: networking, and more. If SSH fails on this machine, please
↪  install
default: the guest additions and repackage the box to continue.
default:
default: This is not an error message; everything may continue to work
↪  properly,
default: in which case you may ignore this message.
==> default: Running provisioner: file...
==> default: Running provisioner: shell...
default: Running:
↪  C:/Users/cjones1/AppData/Local/Temp/vagrant-shell20160513-2884-47qc18.sh
...
...
...
==> default: 2017-02-20 15:36:40.300  INFO 14266 --- [nio-8080-exec-1]
↪  o.s.web.servlet.DispatcherServlet        : FrameworkServlet
↪  'dispatcherServlet': initialization started
==> default: 2017-02-20 15:36:40.354  INFO 14266 --- [nio-8080-exec-1]
↪  o.s.web.servlet.DispatcherServlet        : FrameworkServlet
↪  'dispatcherServlet': initialization completed in 54 ms
```

Much more output will display in the middle as the steps in **`bootstrap.sh`** are executed. The amount of time it will take will depend on your network connection and overall computer speed.

6. Verify that the Spring PetClinic is running:

   `http://localhost:8484`

   Note: *Notice that we configured the VM to expose its own port 8080 on the host's port 8484. That means that you can't have anything running on that port.*

Heroku

The Spring PetClinic application build needs to be tweaked so that it works on Heroku.

Reconfigure some Plugins

In order to work properly with how Heroku builds Java applications, we need to add a new plugin to our `pom.xml` and remove an existing one.

1. Edit the **pom.xml** and add the following code after the last plugin in the build section of the file:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-dependency-plugin</artifactId>
  <version>2.4</version>
  <executions>
    <execution>
      <id>copy-dependencies</id>
      <phase>package</phase>
      <goals><goal>copy-dependencies</goal></goals>
    </execution>
  </executions>
</plugin>
```

2. Comment out the "git-commit-id-plugin" build plugin. Leaving it in causes problems during the build because of the locations that Heroku uses for the files. It can likely be made to work, but I did not choose to spend the time to track down the solution.

Commit and Push the Changes to GitHub

Heroku will be linked to your GitHub repository. You need to commit and push your changes to GitHub for them to be visible to Heroku.

Create a Heroku Account

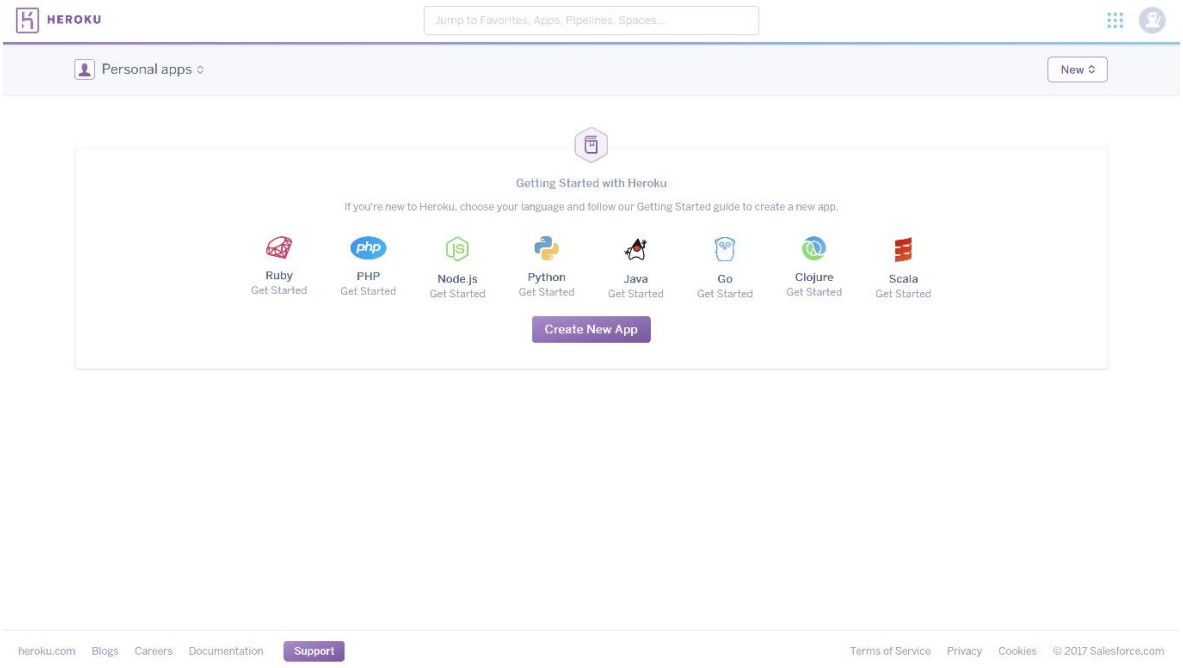
If you don't already have one, create a Heroku account for yourself. Make sure you go with the free tier unless you want to spend more money on this course.

1. Visit `https://www.heroku.com/`.
2. Click the "Sign Up for Free" button and follow the instructions.

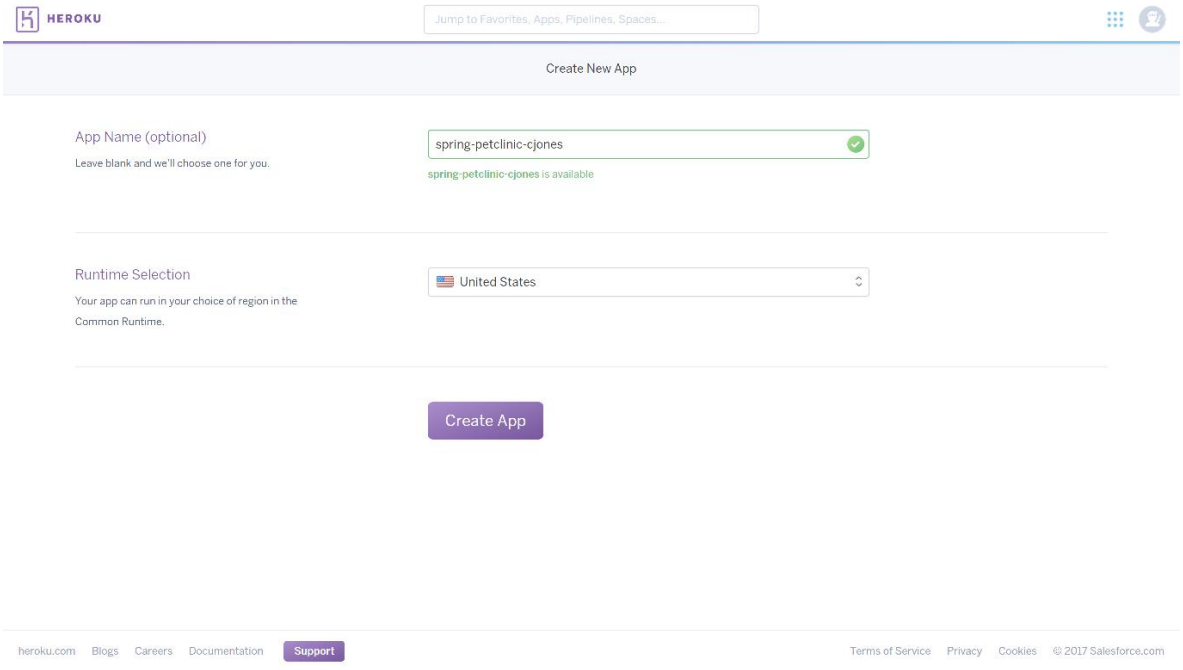
Create a New Application

You need to define a new application to Heroku.

1. Log into your Heroku account. You should be taken to the main Heroku page, shown below:

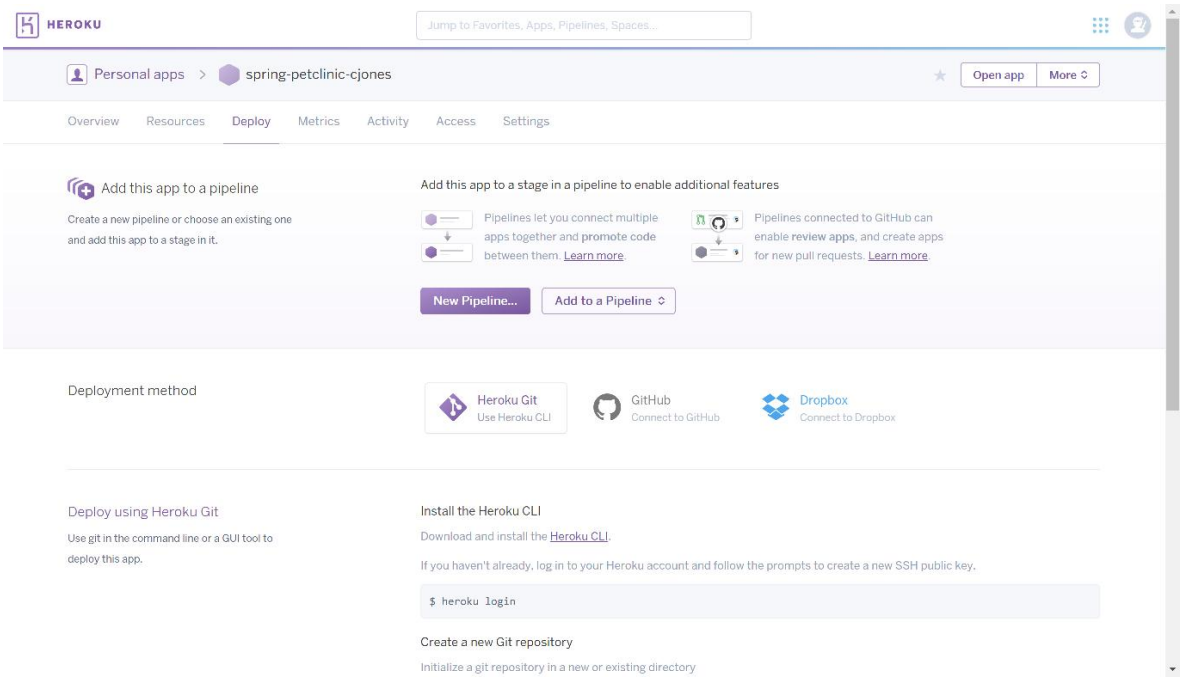


2. Click the "Create New App" button. You will be taken to the "Create New App" page:



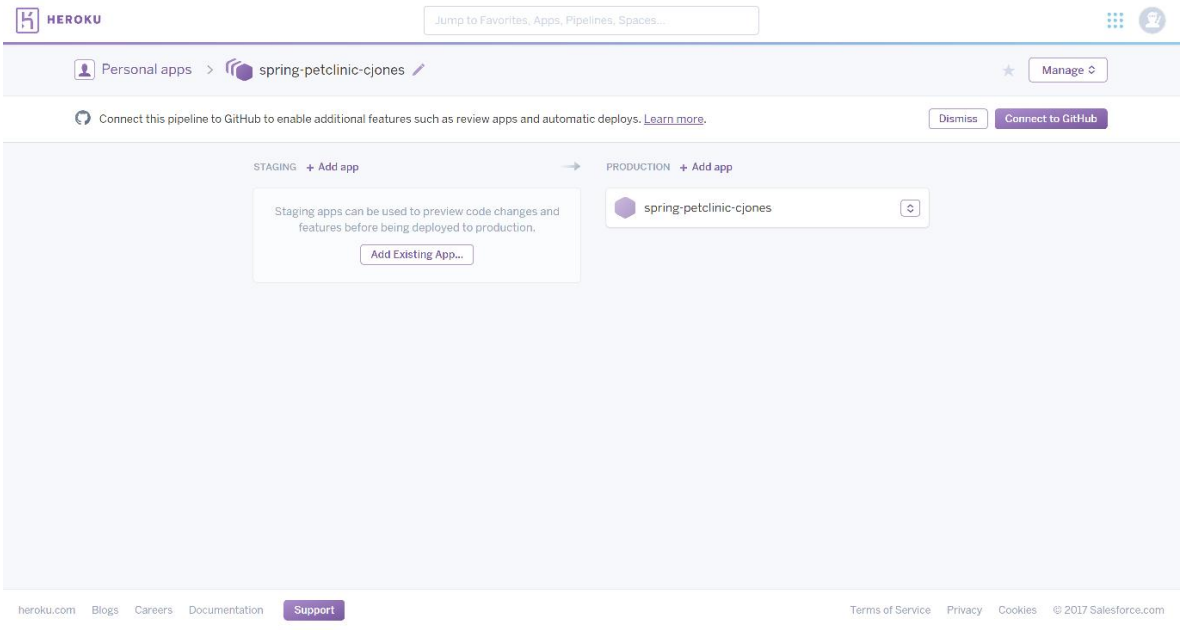
Note: *You can enter a name for your Spring PetClinic app, but it needs to be unique within Heroku. Appending your username should satisfy that requirement.*

3. Click the "Create App" button. You will be taken to the "Deploy" tab of the new application.



4. Click the "New Pipeline" button. The default pipeline name is the same as the application. You can keep the default.

5. Click the "Create Pipeline" button. You will be taken to the pipelines page. This page allows you to define more complex promotion processes by adding new stages to the pipeline so that your application could start in one environment and gradually be promoted up into production. We're going to keep it simple and just have the production environment.

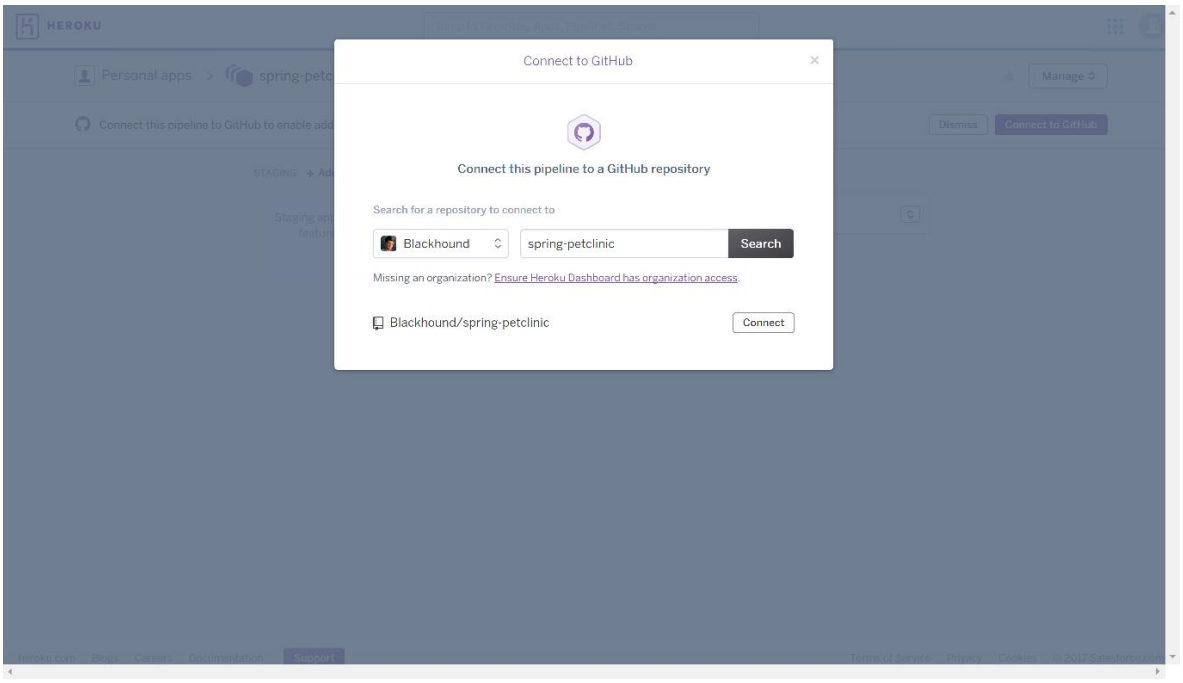


6. Click the "Connect to GitHub" button in the upper-right of the page.

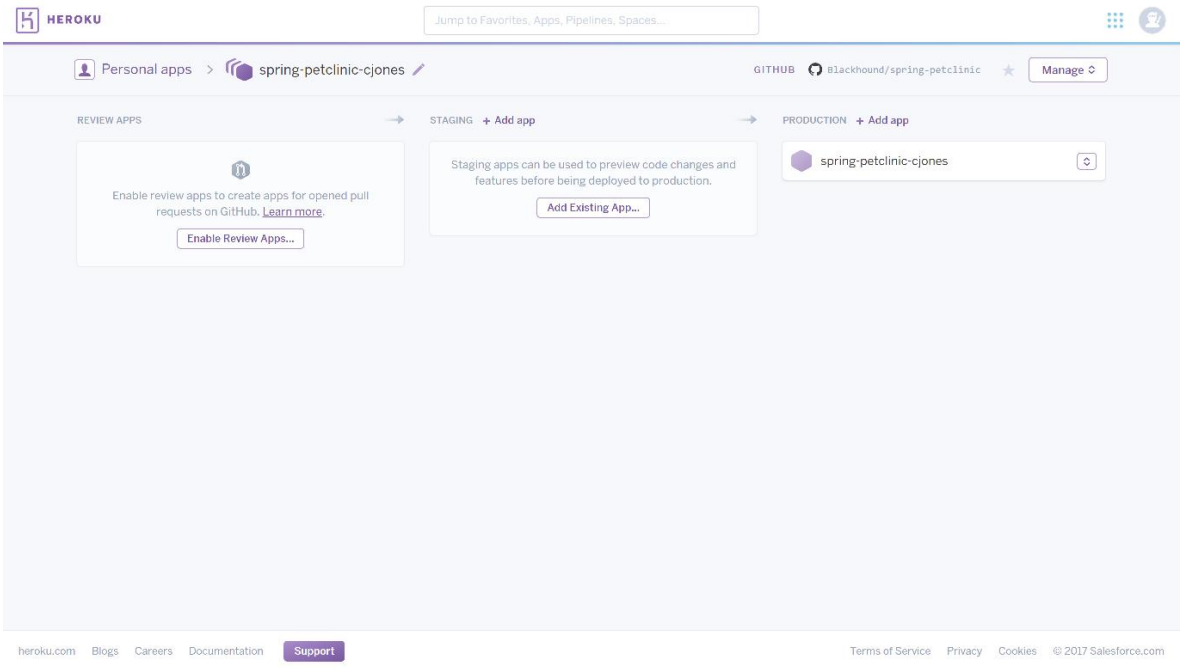
7. Read the informative box describing the benefits of what you planned to do anyway and then click the "Connect to GitHub" button.

8. GitHub will ask for your permission to allow Heroku to access your repositories. Click the "Authorize Application" button at the bottom of the popup. Confirm your GitHub password if prompted.

9. You will now select the repository to which this Heroku application will be attached. Choose your Spring PetClinic repository and click the "Connect" button.



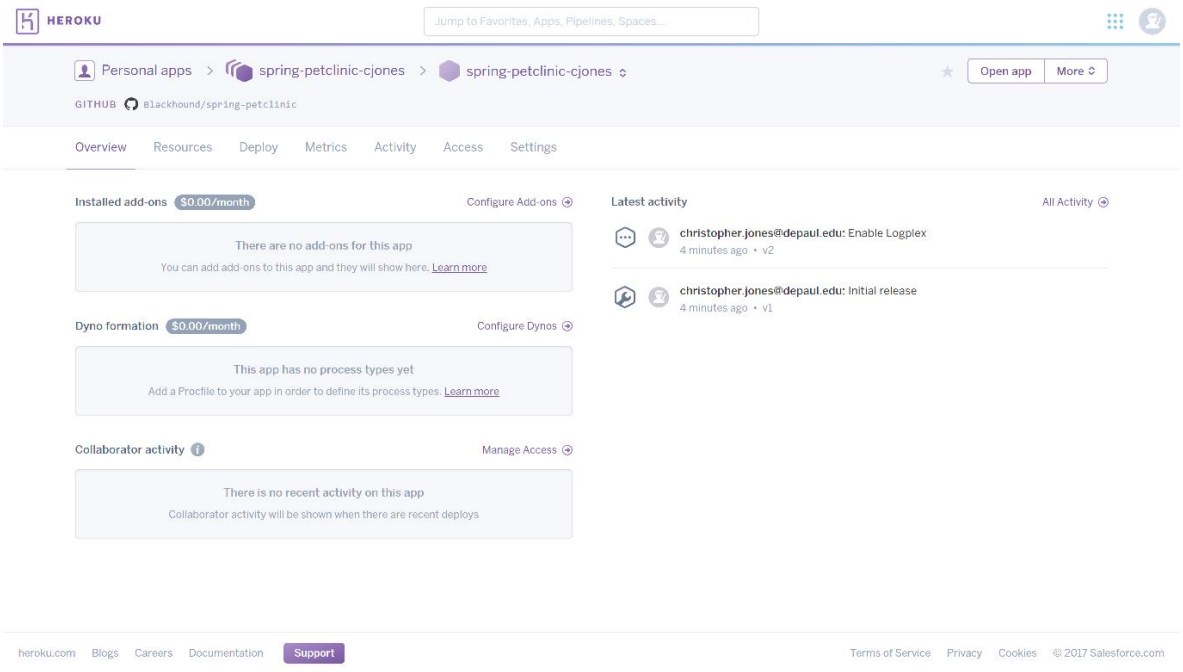
Once the connection is complete you will be returned to the pipeline page, which should look similar to the following:



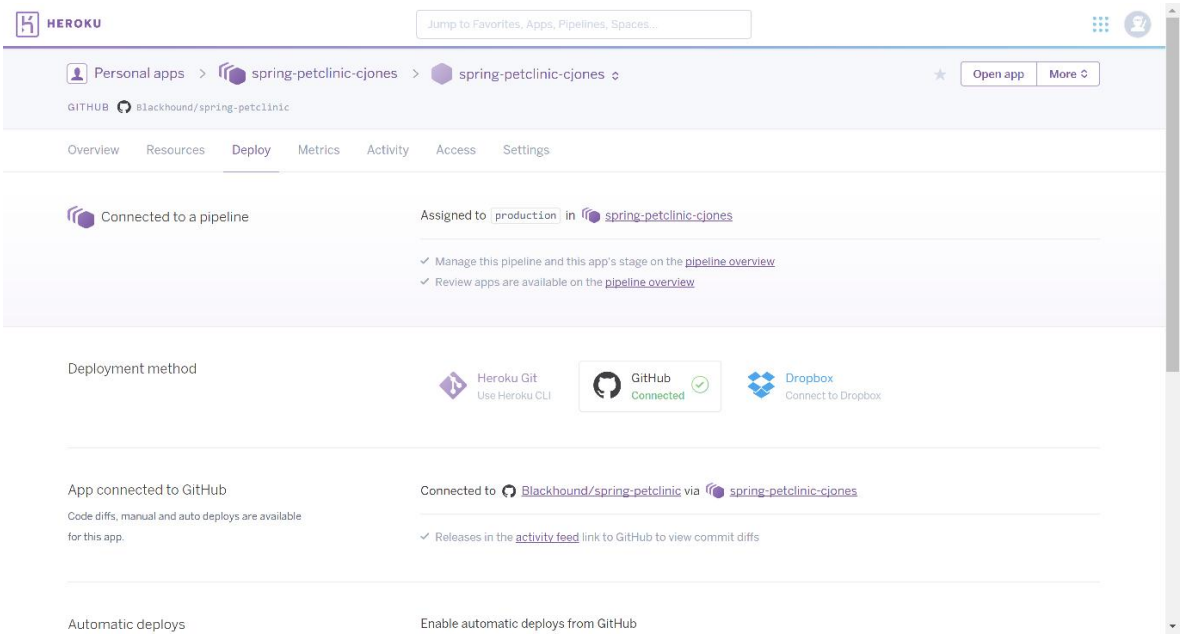
Deploy Your Application to Heroku

You have now linked your Heroku application to your GitHub repository. You now need to request that Heroku perform a build and deploy your application.

1. Click on your pipeline. You will be brought to the "Overview" tab of the pipeline details page:

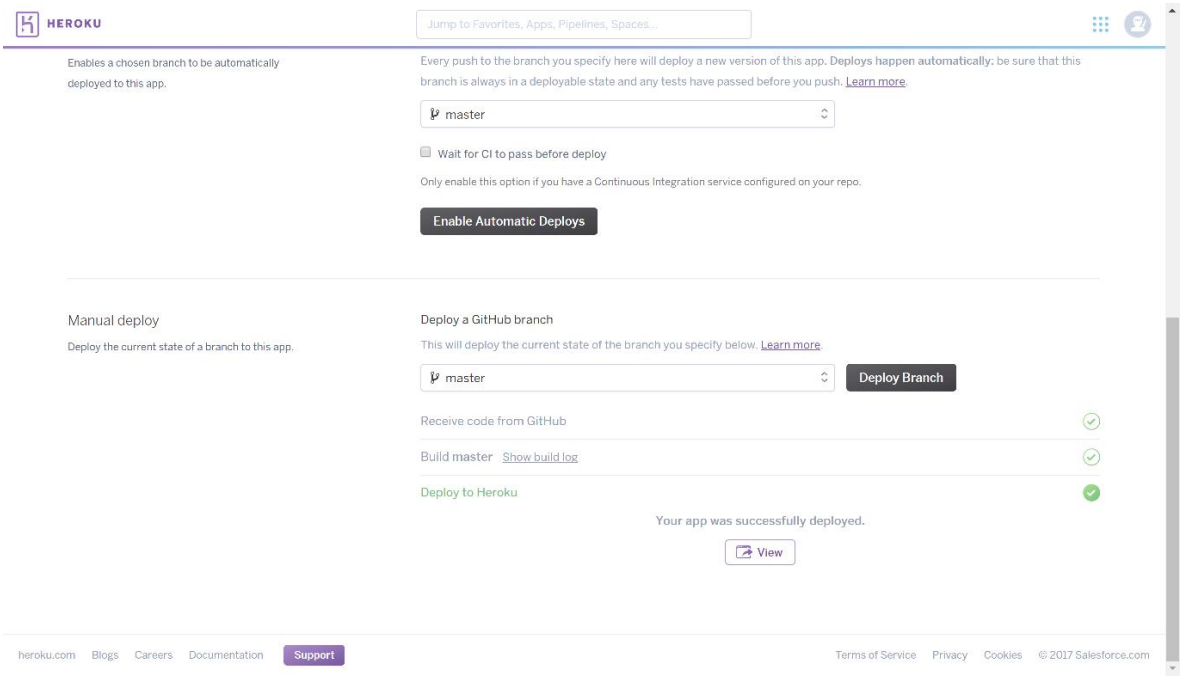


2. Click on the "Deploy" tab. You will be presented with the following:



If you scroll down the page you will find options for enabling the automatic deployment of applications on each commit to the Github repository. We won't enable this yet.

3. Click the "Deploy Branch" button in the "Manual deploy" section at the bottom of the page. This will trigger a manual clone, build, and deployment of the Spring PetClinic.



Note: *Heroku will automatically clone the repository, detect that it is a Maven application, and perform the appropriate goals during the build and deployment process. However, things can always go wrong. If your application fails to deploy, review the logs carefully and make any necessary changes to the* ***`pom.xml`****.*

4. Click the "View" button at the bottom of the page. A new browser window should open and the Spring PetClinic should be displayed.

Chef

In this final section, you will complete some simple Chef tutorials.

Install the Chef Development Kit

Download and install the Chef Development Kit available at `https://docs.chef.io/install\_dk.html`

Complete the Quickstart

Complete the Quickstart Tutorial at `https://docs.chef.io/quick\_start.html`

Deliverables [75 pts]

For this week, please provide screen captures in an MS Word document or PDF that show:

VAGRANT

- 5 pts The output from the `vagrant box list` command after you execute the `vagrant box add` command.
- 5 pts The output from the `vagrant status` command when your VM **is not** running.
- 5 pts Your browser accessing the main page of the website from the Vagrant server while the VM **is not** running at `http://localhost:8484`.
- 5 pts The output from the `vagrant status` command when your VM **is** running.
- 5 pts Your browser accessing the main page of the website from the Vagrant server when your VM **is** running at `http://localhost:8484`.

HEROKU

- 5 pts Your Heroku application list showing the Spring PetClinic application.
- 5 pts Your Heroku Spring PetClinic pipeline.
- 5 pts Your browser accessing the main page of the Spring PetClinic from Heroku.

CHEF

- 5 pts The output of the `generate cookbook` command.
- 5 pts The directory structure for your cookbook.
- 5 pts The content of the `first_cookbook.rb` recipe.
- 5 pts The successful execution of the `first_cookbook.rb` recipe the first time. Show the directory where the `test.txt` file will be created and the contents of that file after running the chef recipe.
- 5 pts The successful execution of the `first_cookbook.rb` recipe after deleting the file. Show the directory that used to contain the `test.txt` file and the contents of that file after running the chef recipe.
- 5 pts The successful execution of the `first_cookbook.rb` recipe after changing the contents of the `test.txt` file. Show the contents of `test.txt` file after your change and the contents of that file after running the chef recipe.
- 5 pts The successful execution of the `first_cookbook.rb` recipe after changing the contents of the recipe itself. Show the contents of `first_cookbook.rb` file after your change and the contents of `test.txt` after running the chef recipe.