

LAB EXERCISE: RedHat OpenShift with Contrail 5.0

Author

Savithru Lokanath <slokanath@juniper.net>, v1.4

Prerequisites

- An AWS account with permissions to CRUD Compute, Storage & Network resources
- CentOS 7 subscription
- GitHub account & please fork <https://github.com/savithruml/django-ex>

Installation

Follow [this guide](#) to bring up the cluster on AWS

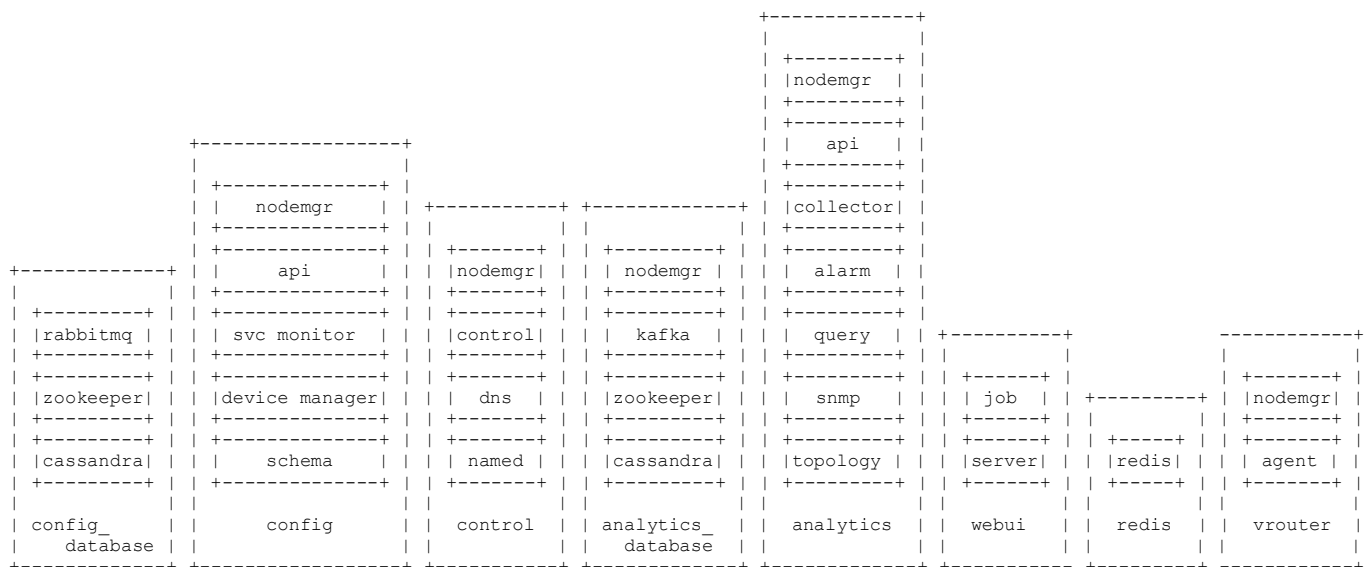
NOTE: This installer provisions 1 Master (Control) & 1 Minion (Compute)

Architecture

Starting with release 5.0, all Contrail services are containerized & are orchestrated by OpenShift Container Platform.

OpenShift Container Platform leverages the Kubernetes concept of a *pod*, which is one or more [containers](#) deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed.

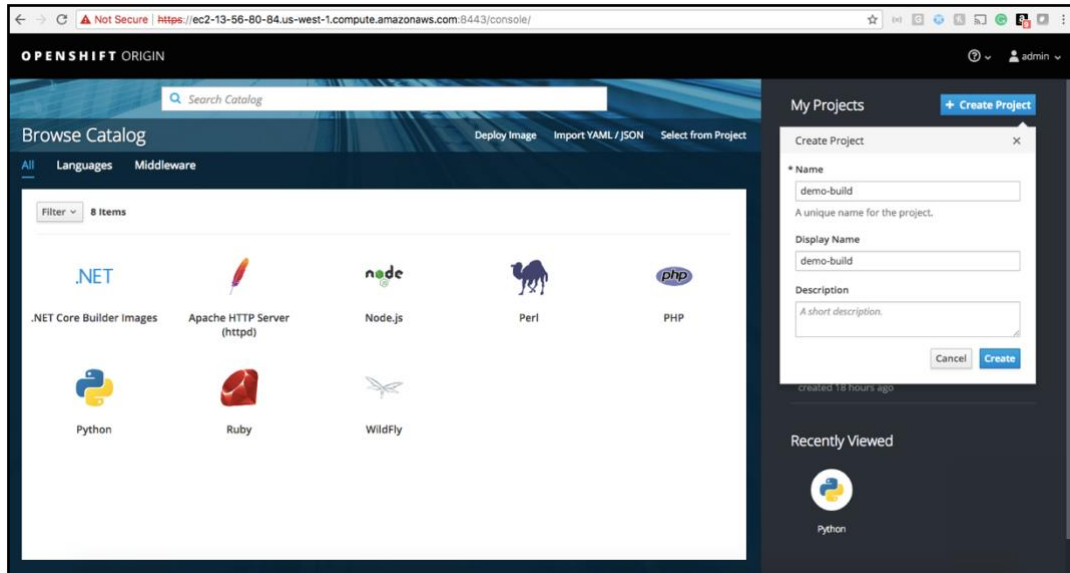
The inner structure of Contrail *pods* is as below



Use Case - 1

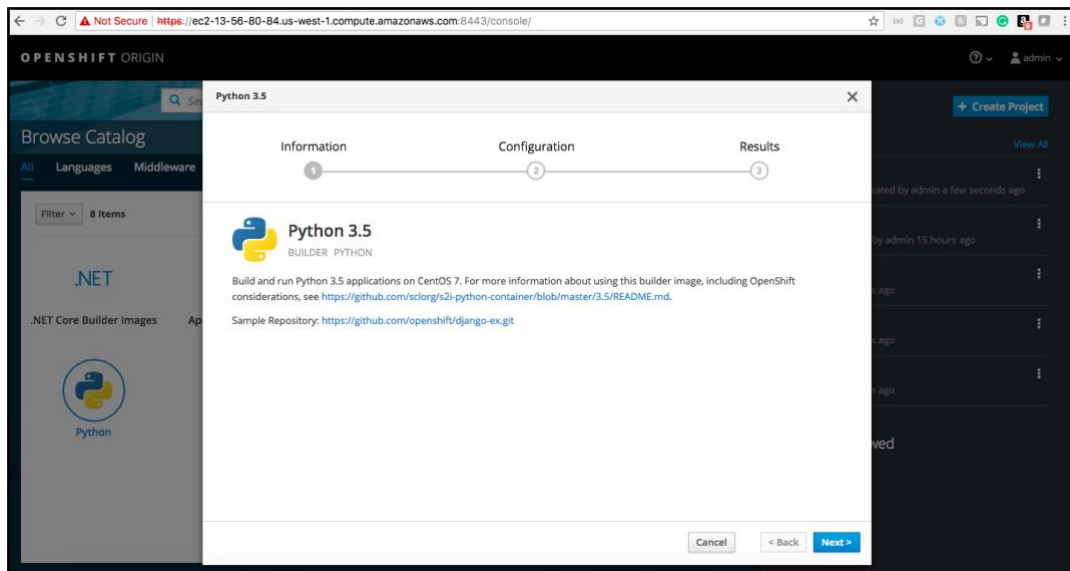
- Build/test/deploy highly scalable applications using OpenShift

1. Login to the OpenShift dashboard & **create** a new project



2. **Launch** a python application in the new project you just created

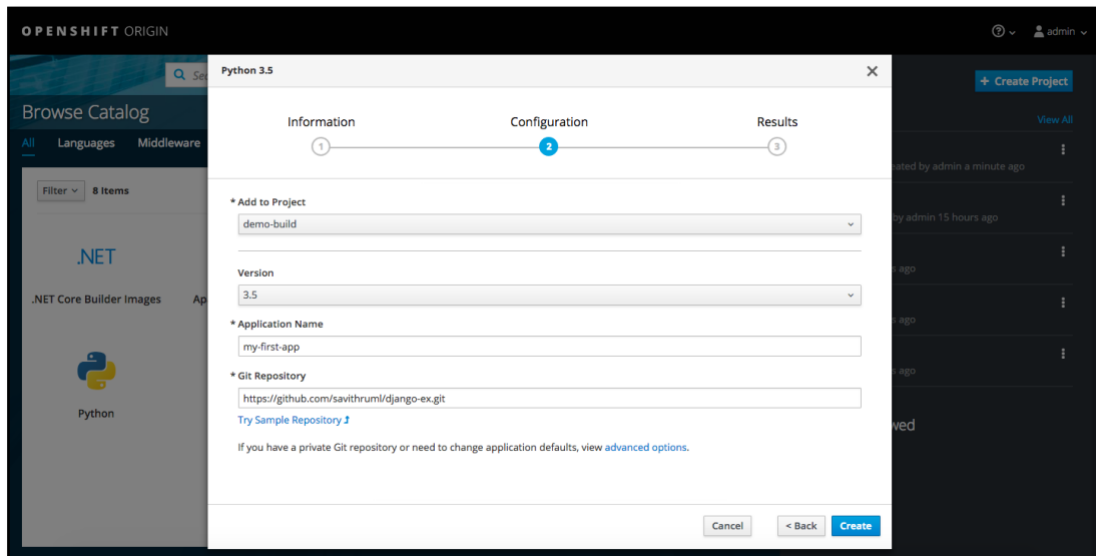
OpenShift Dashboard > Browse Catalog > Python



Configuration:

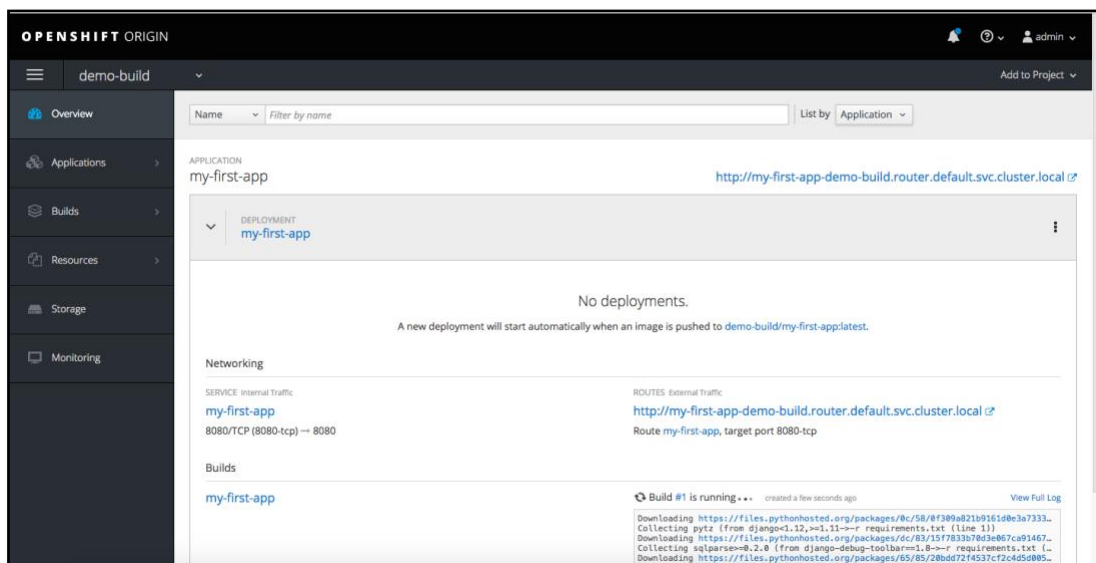
- **Add to Project:** Choose the new project
- **Version:** 3.5
- **Application Name:** <user-defined-name>
- **Git Repository:** <https://github.com/<your-github-username>/django-ex.git>

Hit **Create** once you have configured the parameters



3. Monitor the build process

OpenShift Dashboard > Projects > Overview



The build process will

- **Clone** the source repository (from GitHub)
- **Download** the prerequisites required to run the application
- **Build** the image & **store** it in the local docker registry
- **Deploy** the image to the application pod
- **Host** the application & **expose** the route/service

To check the build details, navigate to *OpenShift Dashboard > Builds > Builds*

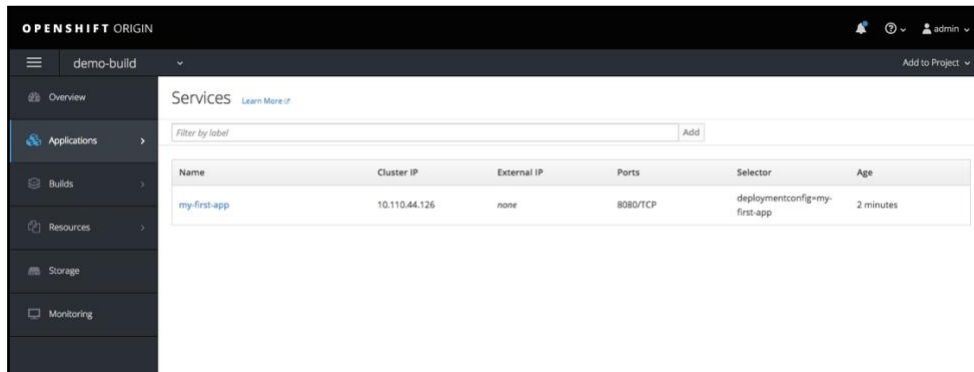
Name	Last Build	Status	Duration	Created
my-first-app	#1	Complete	57 seconds	a minute ago

There are containers which come & go. They are responsible for deploying the app containers & terminate after finishing their task

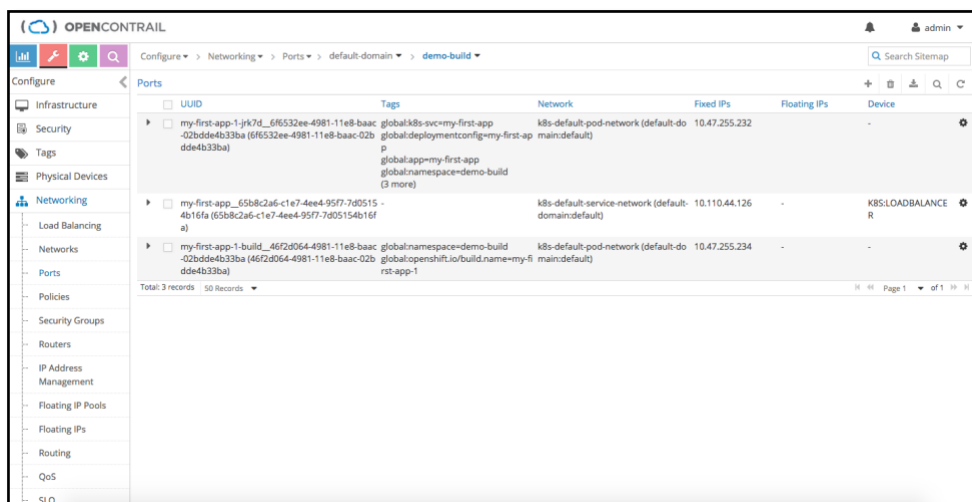
To view running pods, navigate to *OpenShift Dashboard > Applications > Pods*

Name	Status	Containers Ready	Container Restarts	Age
my-first-app-1-jrk7d	Running	1/1	0	a few seconds
my-first-app-1-build	Completed	0/1	0	a minute

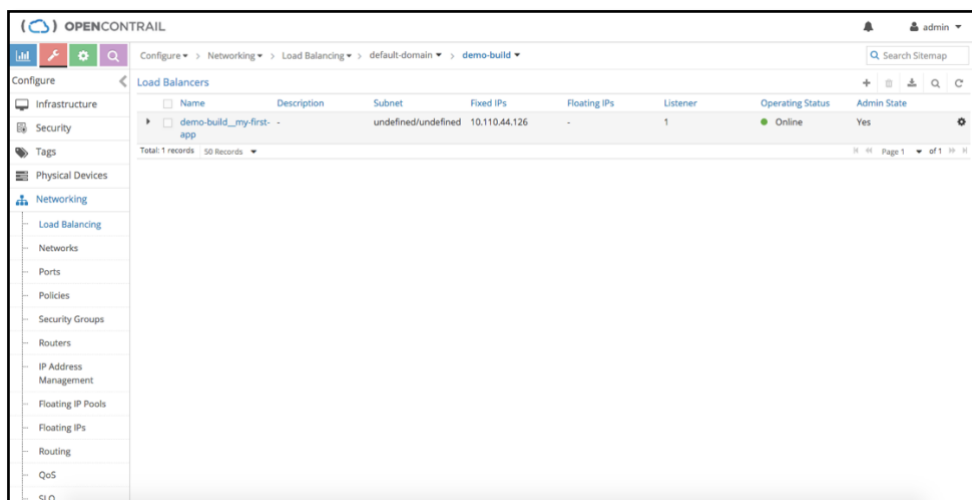
To view the service (logical load balancer) routing traffic to backend pods, navigate to *OpenShift Dashboard > Applications > Services*



Navigate to *Contrail Dashboard > Configure > Networking > Ports* to view all the IP address's assigned to the pods & services



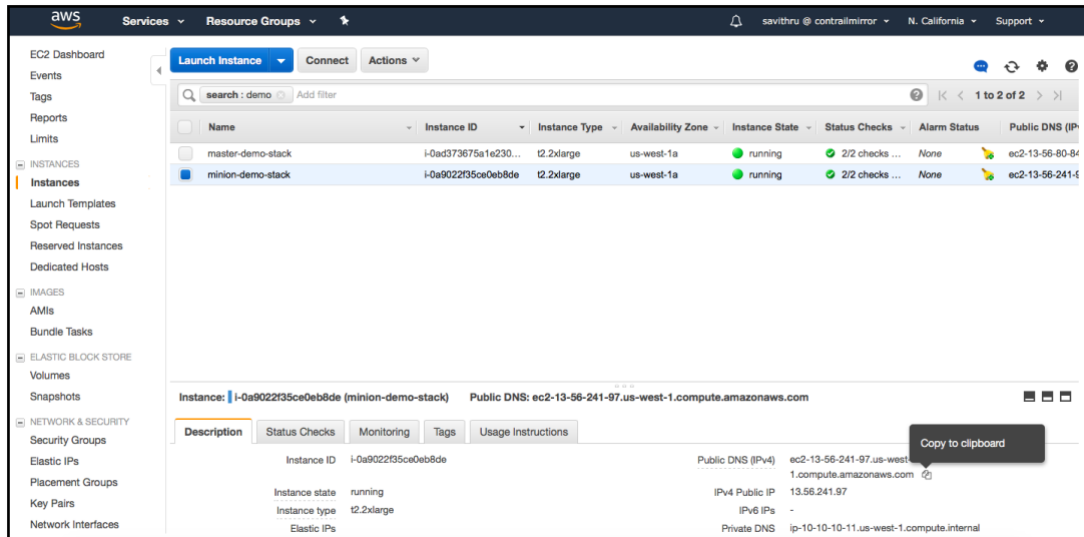
To view services (load balancers), *Contrail Dashboard > Configure > Networking > Load Balancing*



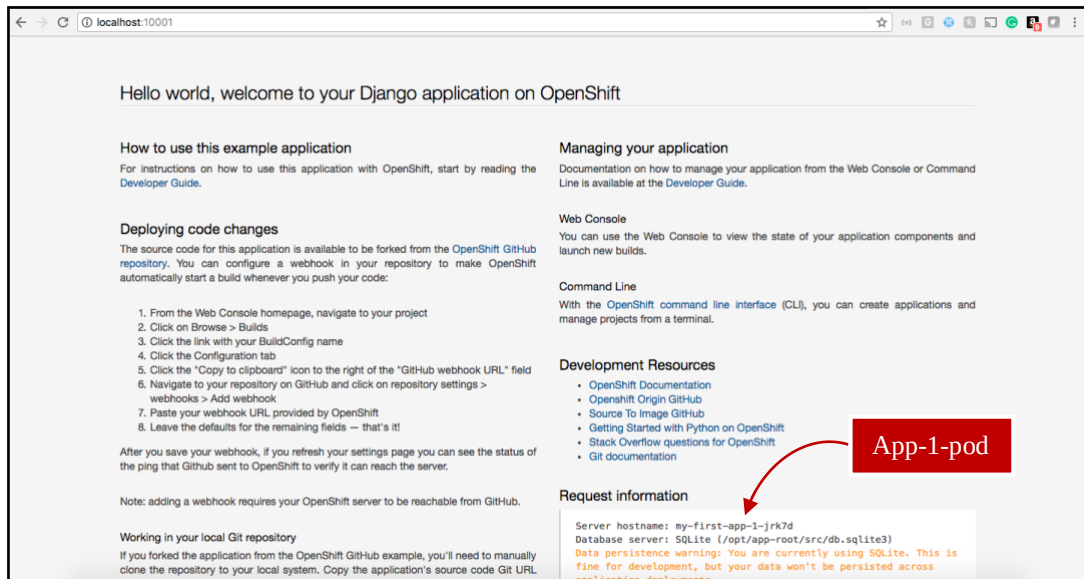
4. Once the application is built & exposed, **verify** that you can access the application

Get the minion's public DNS from AWS dashboard. With this, we will create an SSH tunnel to our localhost

(localhost)# ssh -L localhost:10101:<service-IP-address>:8080 root@<minion-public-DNS>



Once done, the application should be available at **localhost:10101**

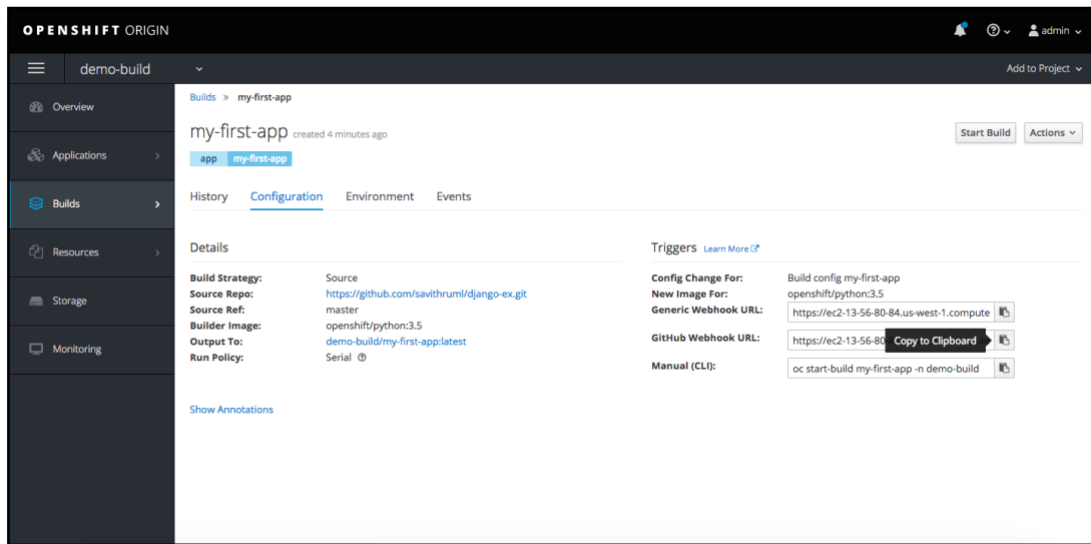


5. Add a webhook trigger to OpenShift

OpenShift supports GitHub webhook triggers to launch create new build

Let us add a webhook trigger to the existing application, so that OpenShift will start a new build & host the application whenever we make a change to the code in our GitHub repository

Copy the GitHub webhook URL from *Builds > Builds > my-first-app > Configuration*



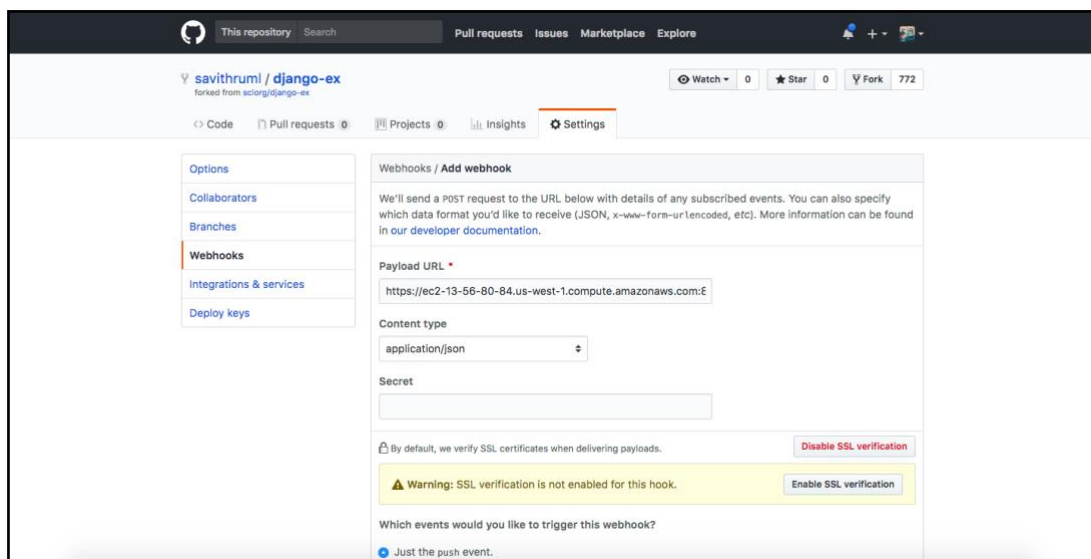
Navigate to <https://www.github.com/<username>/django-ex/settings/hooks> & paste the URL

Configuration:

Payload URL: Paste the URL from previous step

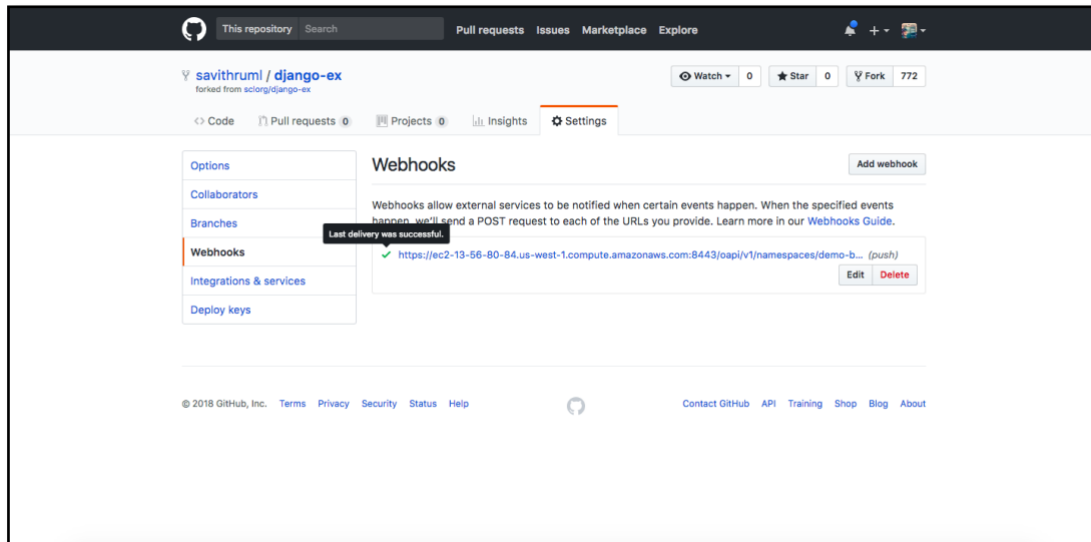
Content type: application/json

Disable SSL Verification & click on **Add Webhook**



6. Verify webhook is successfully added

You should see a green tick against the URL if the webhook was successfully added



7. Make a code change to one of the files of the repository

Change line #214 (banner) in *django-ex/welcome/templates/index.html* to something else

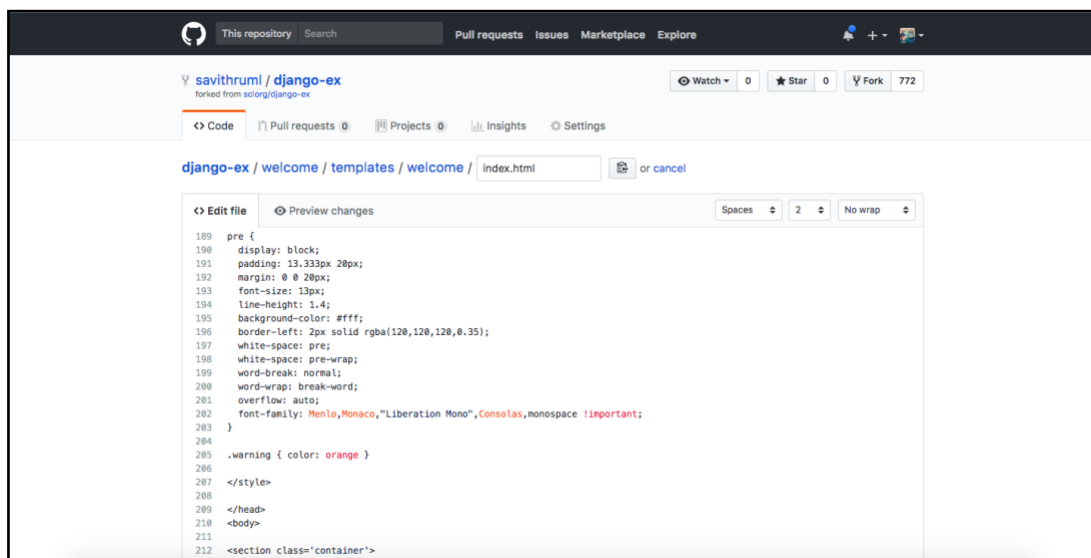
For example, my change looks like this

Previous (line #214):

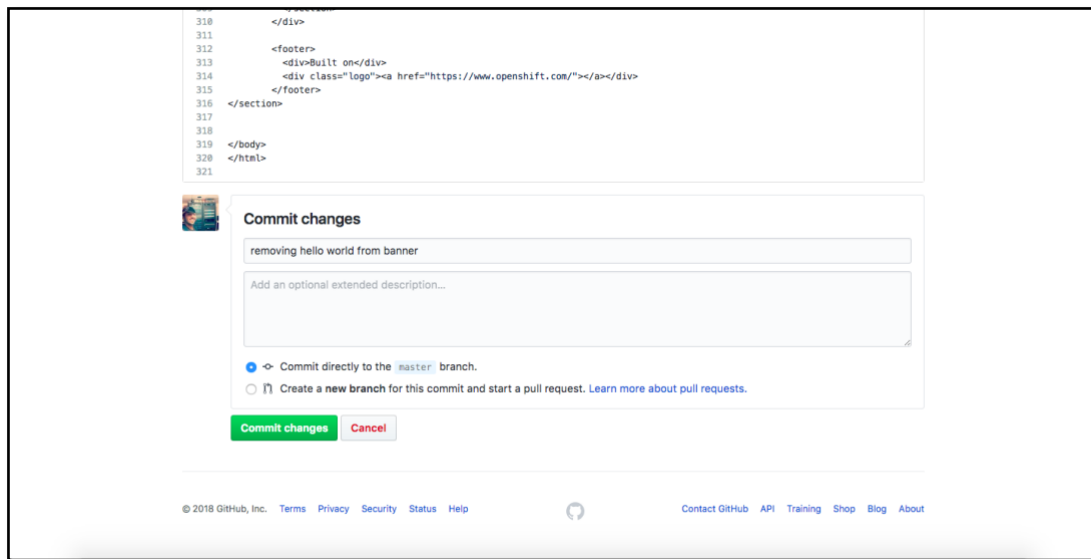
```
<h1>Hello world, welcome to your Django application on OpenShift</h1>
```

Current (line #214):

```
<h1>Welcome to your Django application on OpenShift</h1>
```



Give a meaningful commit message & **confirm** (commit) the change

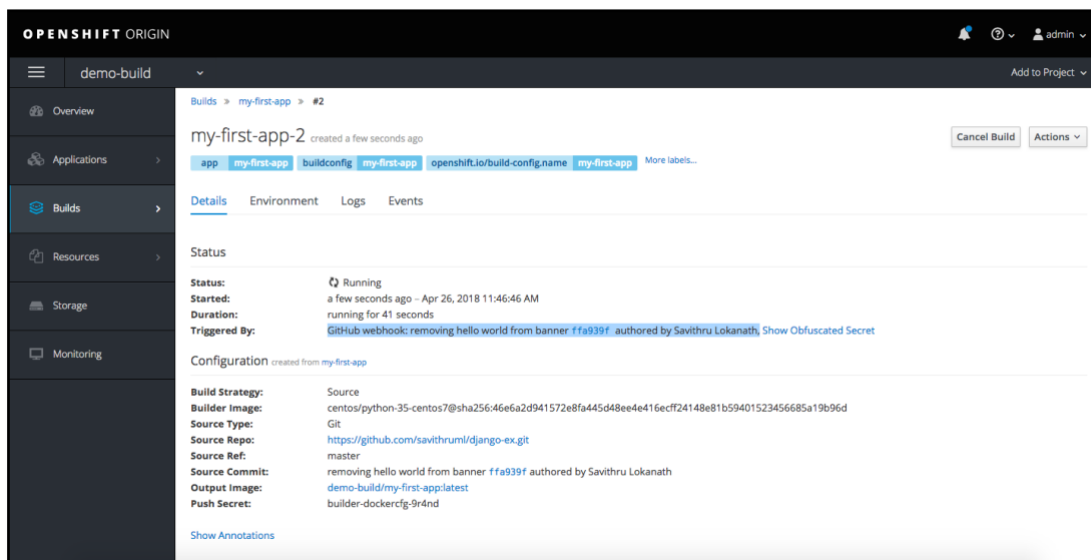


8. Once you have made the change, **go back** to OpenShift dashboard

GitHub sends a payload to OpenShift build URL, notifying the change made to *index.html*

Hence, you should now see a new build (#2) triggered due to the change made in the previous step

NOTE: STEP-3 is repeated against the new source code (change to *index.html*)

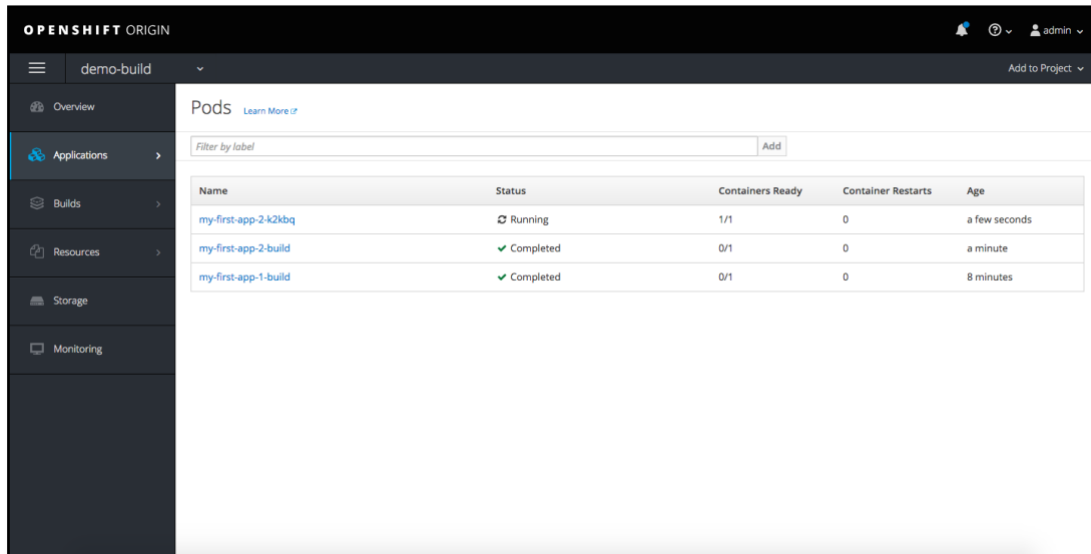


9. Once build #2 has completed, **verify** that the new application is live

Navigate to *Application > Pods* & verify you can see the new application pod

NOTE: The pods holding the old application (#1) is deleted once the new application (#2) is live

NOTE: OpenShift still holds the old application's build pod. If the administrator wants to rollback to an older release, then he can just launch an application pod & need not go through the whole build procedure



10. Now, if you have the web browser open for viewing the old application, hit the **refresh** button. You should see the new application

NOTE: Although the backend pods changed, the service load-balancer did not. Hence, the IP address used to access the application remains the same

