Deployment using Templates:

A. Accessing OpenShift Cluster for the workshop
B. Uploading template through terminal by logging into OpenShift client
C. Using a template from web console
D. Editing the same uploaded template
E. Creating a template from the existing objects

Prerequisites to perform this lab.

(1) Create IBM Cloud account

https://cloud.ibm.com

(2) Install OpenShift Client (oc)

https://cloud.ibm.com/docs/openshift?topic=openshift-openshift-cli

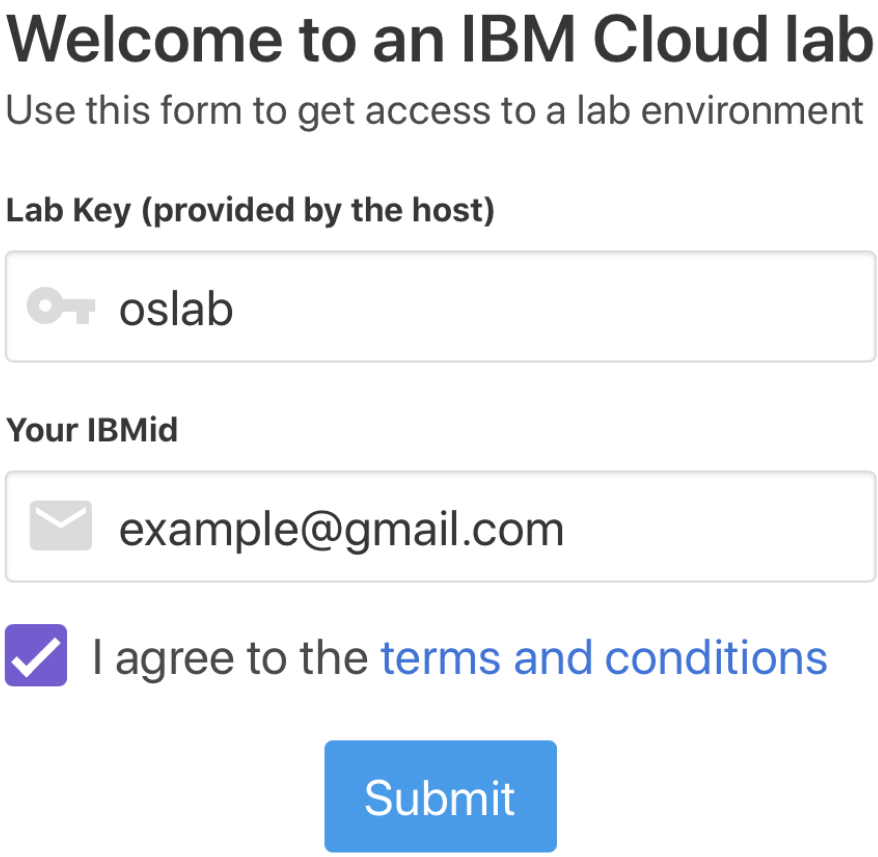
A. Accessing OpenShift Cluster for the workshop

For the purpose of this workshop organizers pre-provisioned for attendees free RedHat OpenShift Clusters. Thanks to that you can try and learn how to use them.

The specification of these Red Hat OpenShift clusters are the following

3 Worker Nodes Each Worker Node has 4 CPU cores with 16GB Ram the abbreviated description: 3 x 4 x 16

Use this URL to claim the clusters: https://osrhwt.mybluemix.net



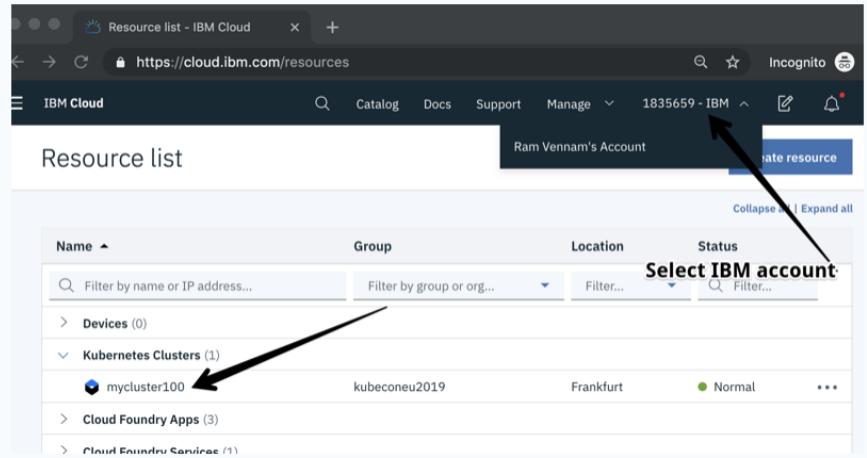
Enter the Lab Key given by your instructor, as well as the email address associated with your IBM Cloud account.

After you hit submit, you will be given a cluster for the duration of the workshop.

You should be given the result page showing the successful assignment.

Congratulations!

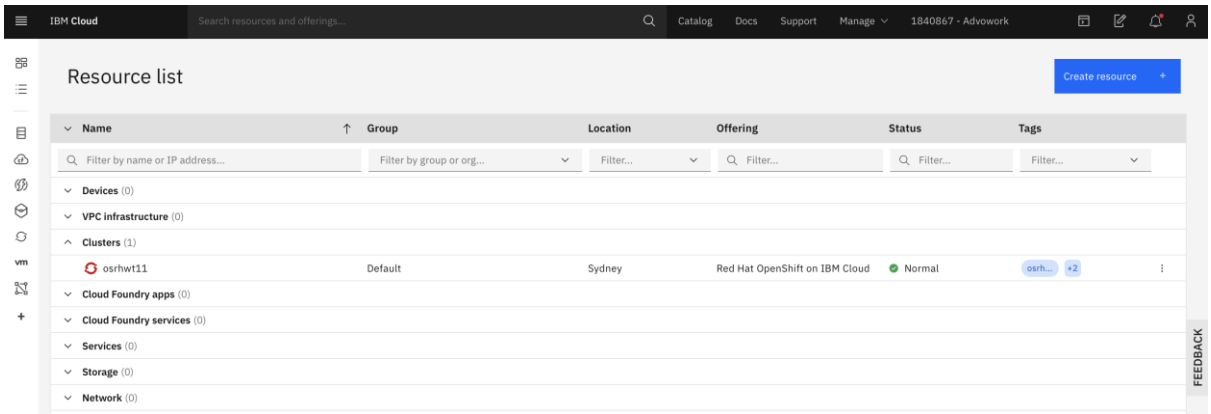
You have been assigned a openshift cluster.



1. You have been invited to the lab account.
2. Log in to this **IBM Cloud account**
3. Navigate to Clusters.
4. Select the cluster named **osrhwt11**

When you sign up / log in to IBM Cloud, from the Dashboard you need to navigate to IBM Org **(Advowork**), you can see the option above 'Create Resources tab' beside 'Manage'

Next, go to the resource list, from the hamburger menu top left to verify your cluster.

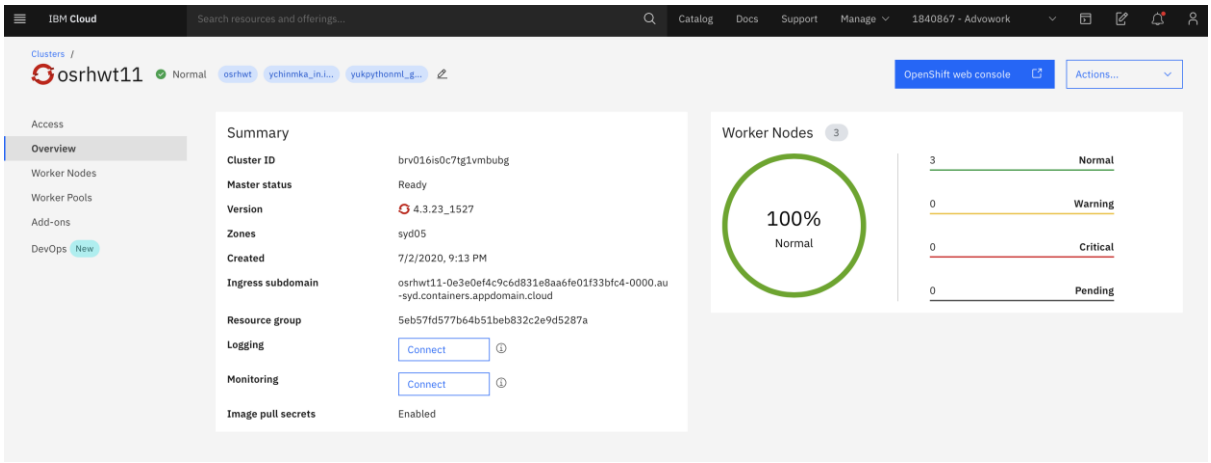


Congratulations! You obtained a Red Hat OpenShift 4.3 cluster. Now you are ready for next lab

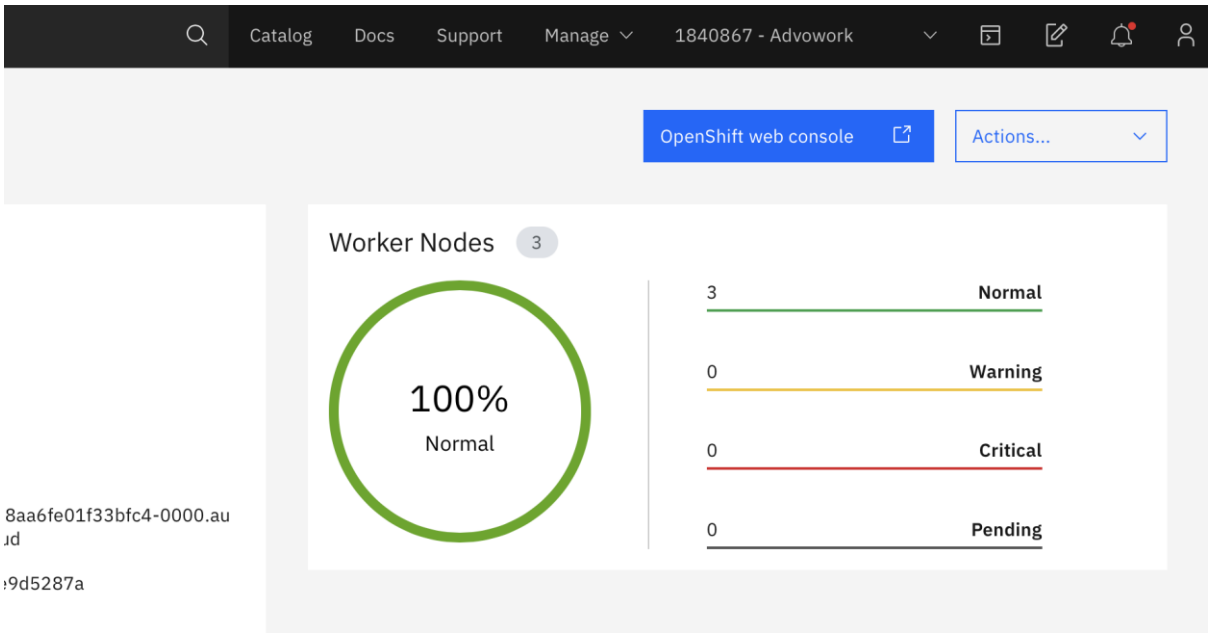
B. Uploading template through terminal by logging into openshift client.

Note: Prerequisite to do this lab is you need to install Openshift CLI (oc)

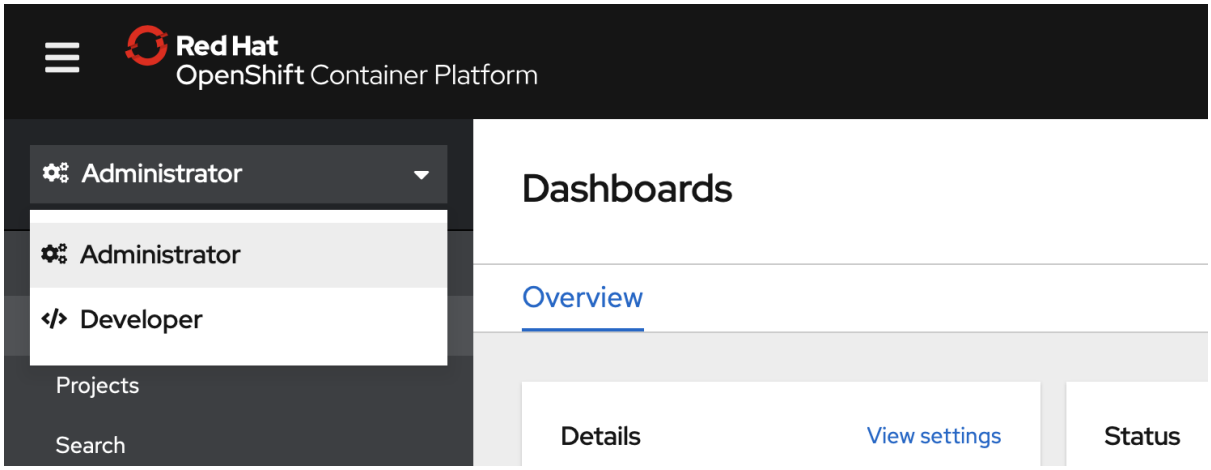
Step 1: Login into IBM Cloud, Open existing OpenShift cluster



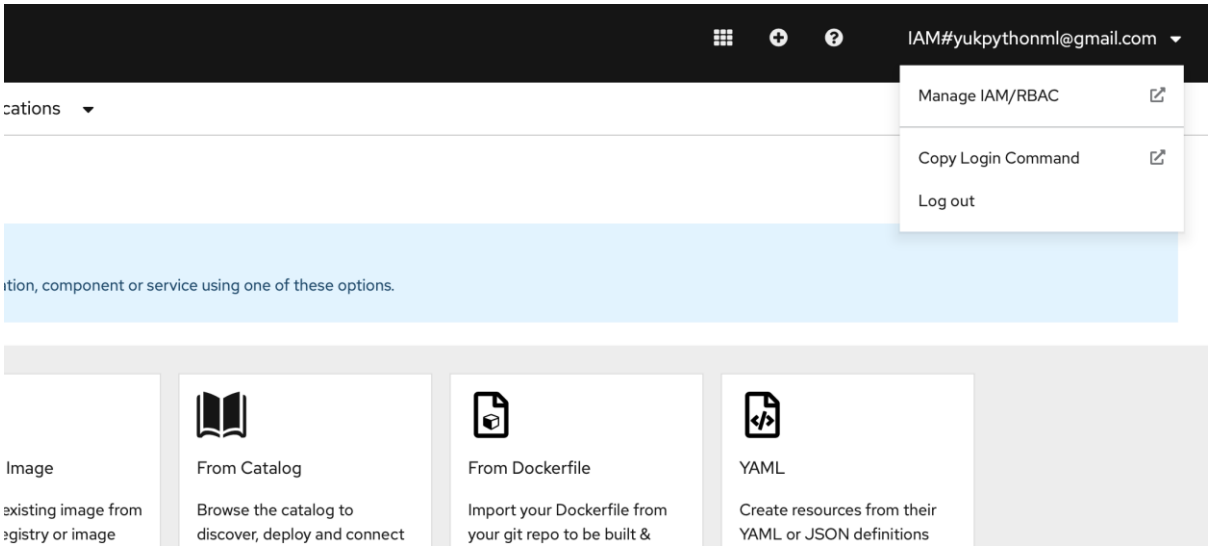
Step 2: Click on ***OpenShift web console*** on right top



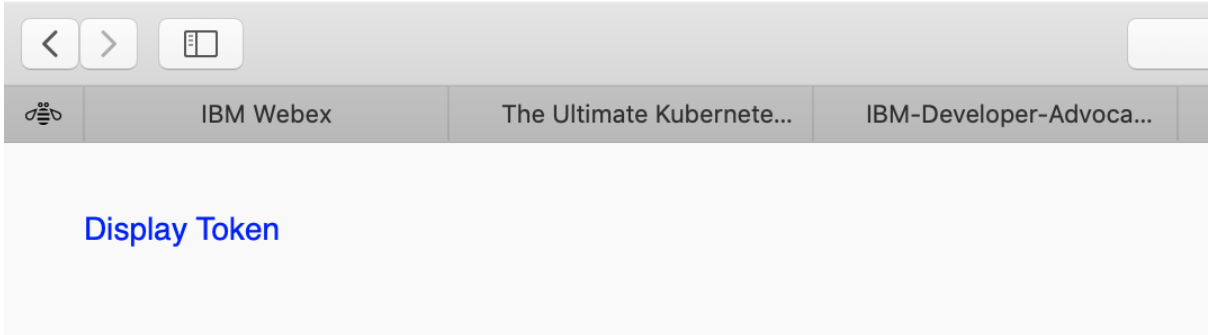
Step 3: RedHat OpenShift Container Platform gets opened. Select the user type to ***Developer***



Step 4: click on ***Copy Login Command*** from right top



Step 5: Click on ***Display Token***



Step 6: Copy the ***oc login*** full command

Step 7: Open new terminal and paste the ***oc login*** full command

```
Yukteshs-MacBook-Pro:~ ychinmka@in.ibm.com$ oc login --token=YbmRBhU8S3unMPBPmxL9v8YGt2nzHATzlI8YuKOXBnc --server=https://c100-e
.au-syd.containers.cloud.ibm.com:30805
Logged into "https://c100-e.au-syd.containers.cloud.ibm.com:30805" as "IAM#yukpythonml@gmail.com" using the token provided.

You have access to 57 projects, the list has been suppressed. You can list all projects with 'oc projects'

Using project "default".
Yukteshs-MacBook-Pro:~ ychinmka@in.ibm.com$
```

Note: We now logged into OpenShift client using terminal, Before this step ensure oc client is installed in your machine.

Step 8: ***oc projects*** displays total existing projects on OpenShift container platform

```
Yukteshs-MacBook-Pro:~ ychinmka@in.ibm.com$ oc projects
You have access to the following projects and can switch between them with 'oc project <projectname>':

    calico-system
  * default
    ibm-cert-store
    ibm-system
    kube-node-lease
```

Let us make our template ready

Note: If you have a JSON or YAML file that defines a template, for example you can upload the template to project using the CLI. This saves the template to the project for repeated use by any user with appropriate access to that project.

Step 9: Clone https://github.com/sclorg/nodejs-ex a sample nodejs application

```
Yukteshs-MacBook-Pro:os301 ychinmka@in.ibm.com$ git clone https://github.com/sclorg/nodejs-ex
Cloning into 'nodejs-ex'...
remote: Enumerating objects: 2, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 653 (delta 0), reused 0 (delta 0), pack-reused 651
Receiving objects: 100% (653/653), 261.18 KiB | 448.00 KiB/s, done.
Resolving deltas: 100% (252/252), done.
Yukteshs-MacBook-Pro:os301 ychinmka@in.ibm.com$ ls
nodejs-ex
Yukteshs-MacBook-Pro:os301 ychinmka@in.ibm.com$
```

Step 10: Move to templates directory to see available templates (.json files)

```
Yukteshs-MacBook-Pro:os301 ychinmka@in.ibm.com$ cd nodejs-ex/
Yukteshs-MacBook-Pro:nodejs-ex ychinmka@in.ibm.com$ ls
README.md       helm            openshift       package.json    server.js       tests
Yukteshs-MacBook-Pro:nodejs-ex ychinmka@in.ibm.com$ cd openshift/
Yukteshs-MacBook-Pro:openshift ychinmka@in.ibm.com$ ls
pipeline        templates
Yukteshs-MacBook-Pro:openshift ychinmka@in.ibm.com$ cd templates/
Yukteshs-MacBook-Pro:templates ychinmka@in.ibm.com$ ls
nodejs-mongodb-persistent.json  nodejs-mongodb.json             nodejs.json
Yukteshs-MacBook-Pro:templates ychinmka@in.ibm.com$
```

Now let us upload above template to a project

Step 11: Create a project by typing ***oc new-project sampleproject***

```
Yukteshs-MacBook-Pro:templates ychinmka@in.ibm.com$ oc new-project sampleproject1
Now using project "sampleproject1" on server "https://c100-e.au-syd.containers.cloud.ibm.com:30805".

You can add applications to this project with the 'new-app' command. For example, try:
```

Note: Count total no. of templates of this project from web console (It will be 95)

Step 12: Uploading a template to your current projects template library, pass the JSON with following command

```
Yukteshs-MacBook-Pro:templates ychinmka@in.ibm.com$ oc create -f nodejs-mongodb.json
template.template.openshift.io/nodejs-mongodb-example created
```

Note: Now after adding above template count the total no. of templates under this project from web console (It will be 96)

Step 13: Create an app in the same ***sampleproject*** by typing oc ***new-app Django-psql-example***

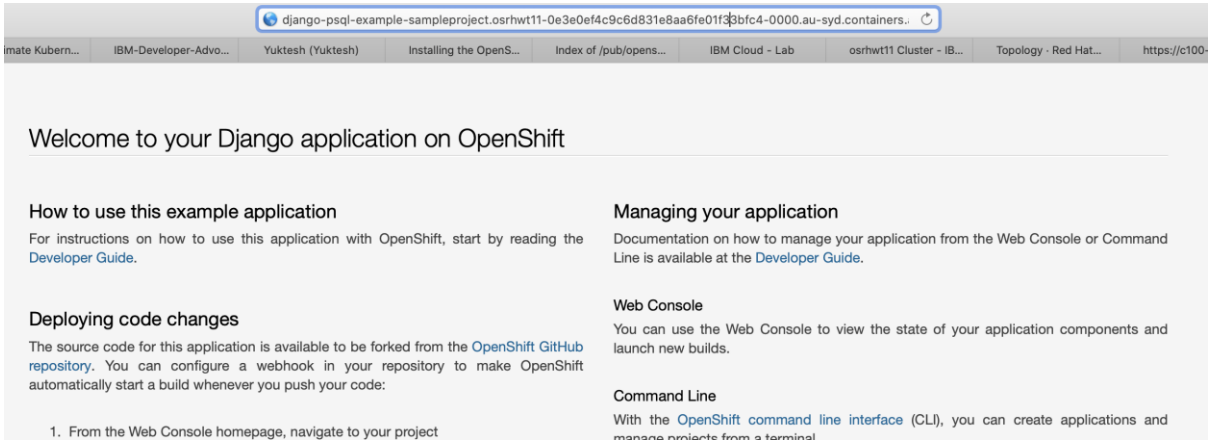
```
Yukteshs-MacBook-Pro:~ ychinmka@in.ibm.com$ oc new-app django-psql-example
--> Deploying template "openshift/django-psql-example" to project sampleproject

     Django + PostgreSQL (Ephemeral)
     ---------
     An example Django application with a PostgreSQL database. For more information about using this t
ft considerations, see https://github.com/sclorg/django-ex/blob/master/README.md.

     WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing.

     The following service(s) have been created in your project: django-psql-example, postgresql.
```

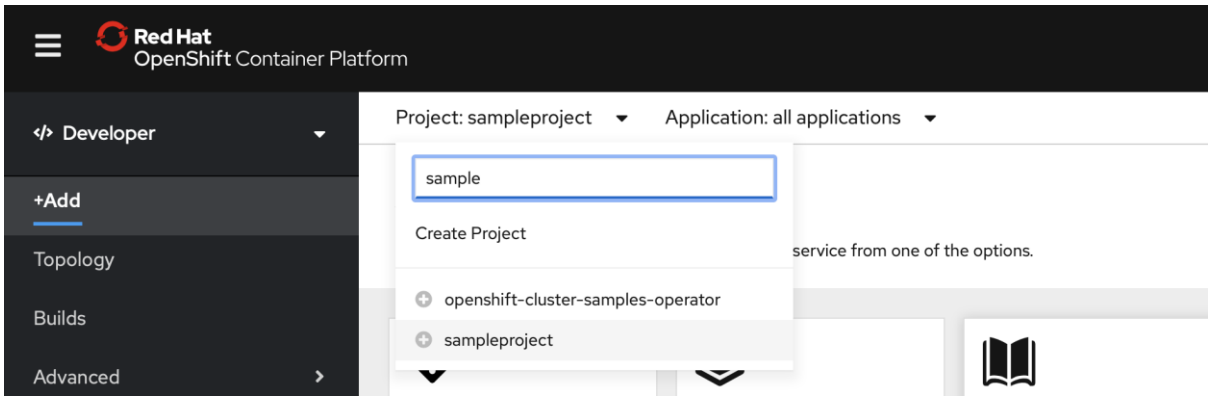
Step 14: You can access application via route ***http://django-psql-example-sampleproject.osrhwt11-0e3e0ef4c9c6d831e8aa6fe01f33bfc4-0000.au-syd.containers.appdomain.cloud***



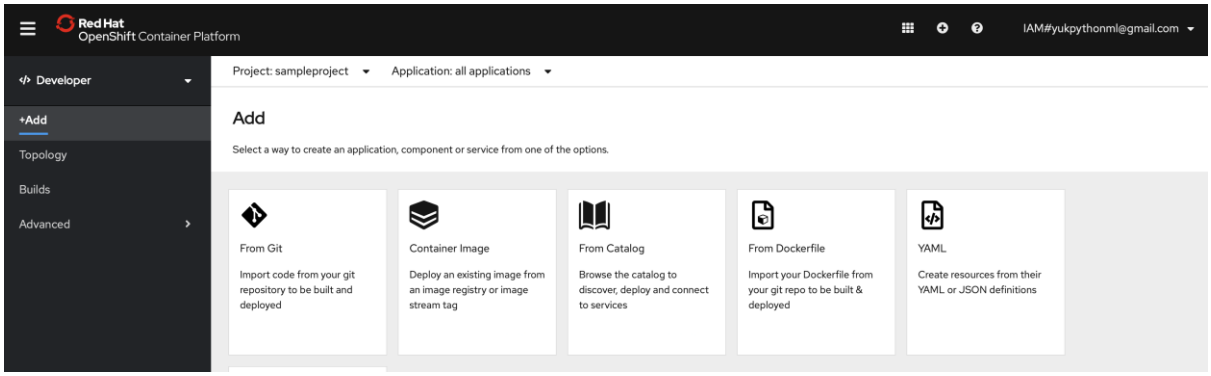
Note: Sometimes it may take 5 to 10min time to start.

C. Using a template from web console

Step 1. Select your project (sampleproject) from OpenShift Container Platform

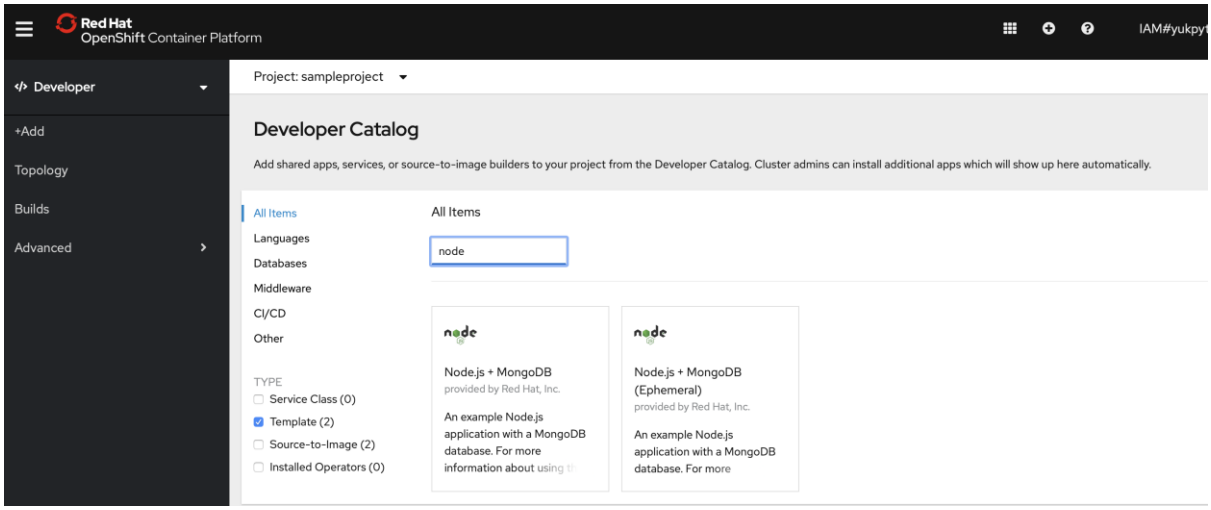


Step 2: select ***From catalog***

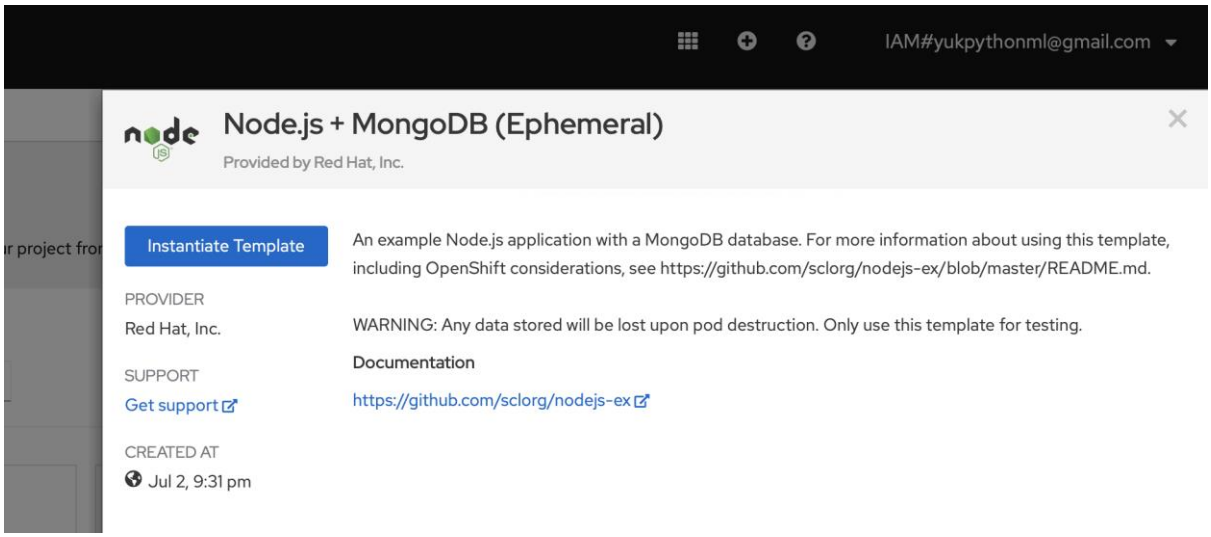


Step 3: select ***template*** and search for ***node*** and click on ***Node.js + MongoDB (Ephemeral)***

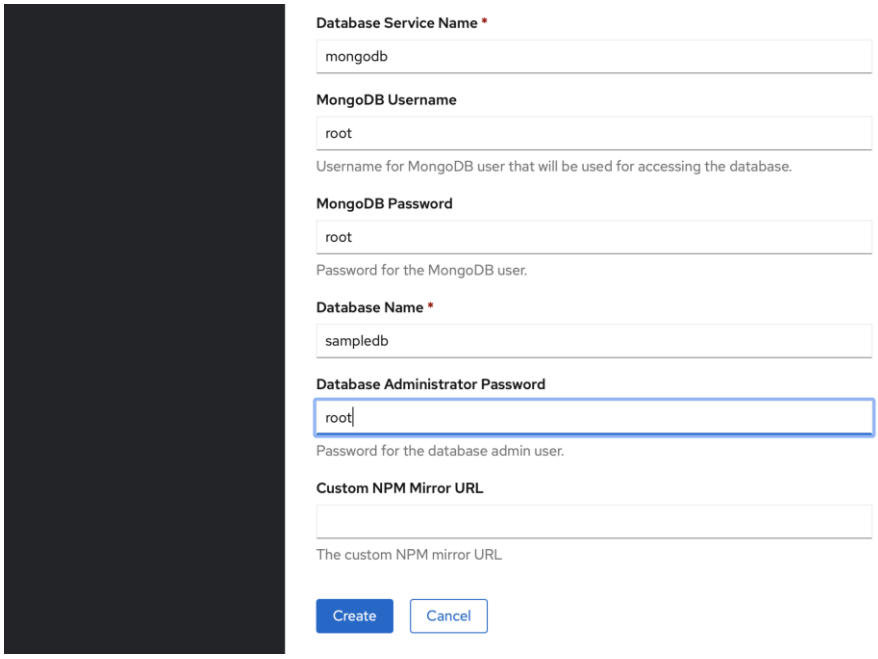
Note: Any data stored will be lost upon pod destruction. Only use this template for testing



Step 4: Click on ***Instantiate Template***

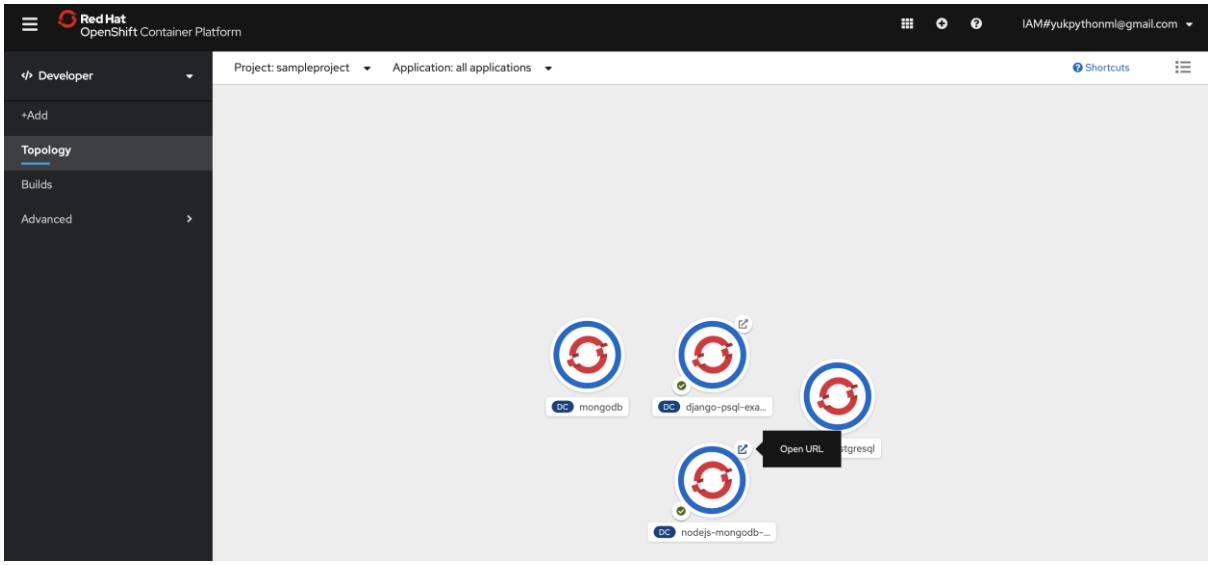


Step 5: Give MongoDB Username, Password and DBA Password as '***root***' and click on ***create***

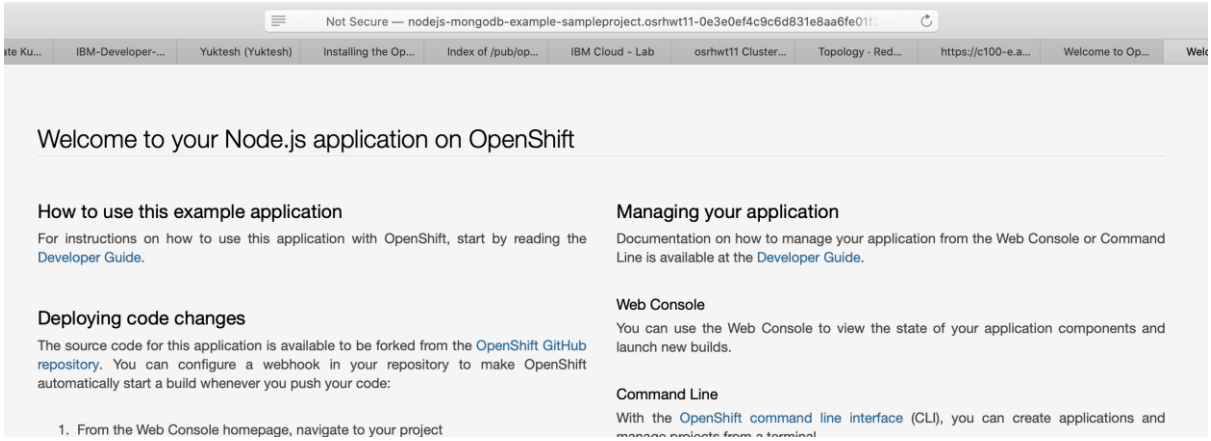


We should be able to see 4 pods running under ***sampleproject*** in Topology

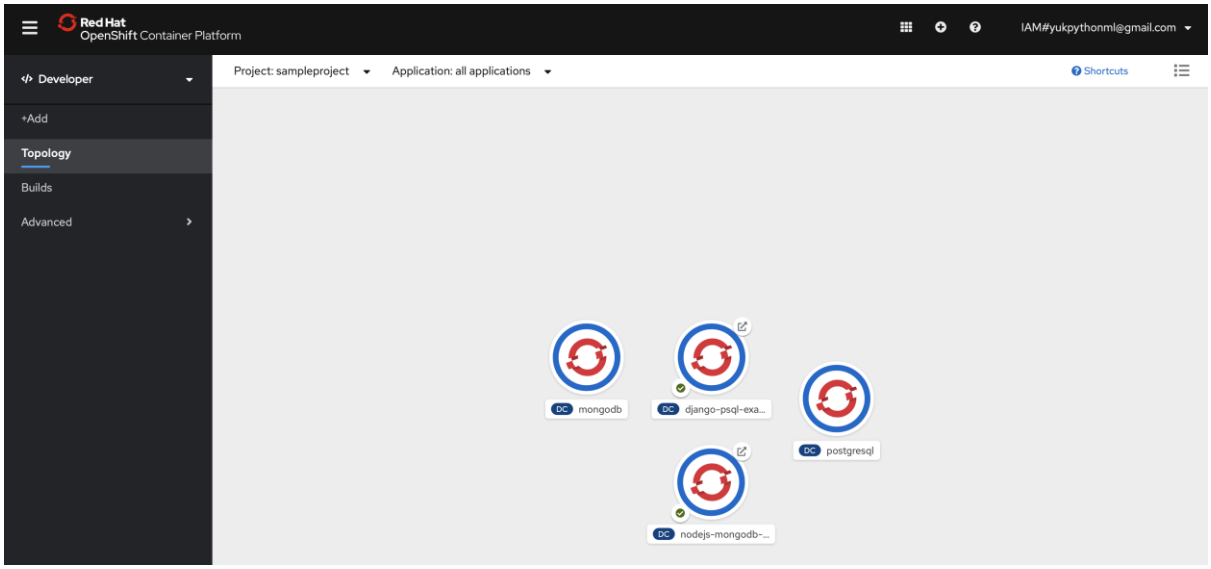
Step 6: From Topology select **nodejs-mongodb** icon (must turn into blue it indicates it is ready to use)



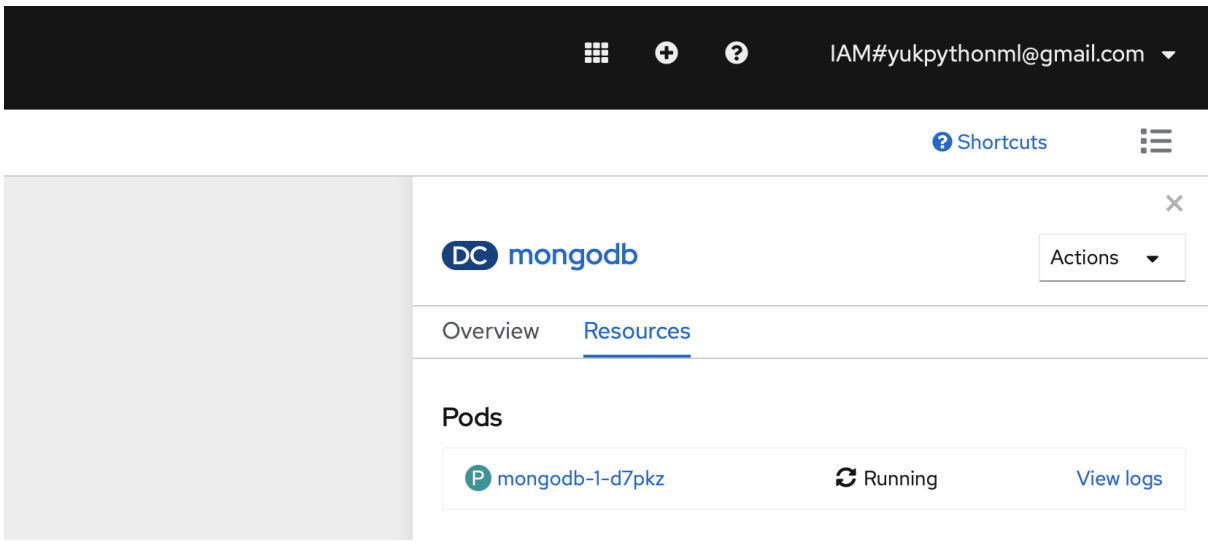
Step 7: Click on Open URL icon of **nodejs-mongodb** pod to access the application



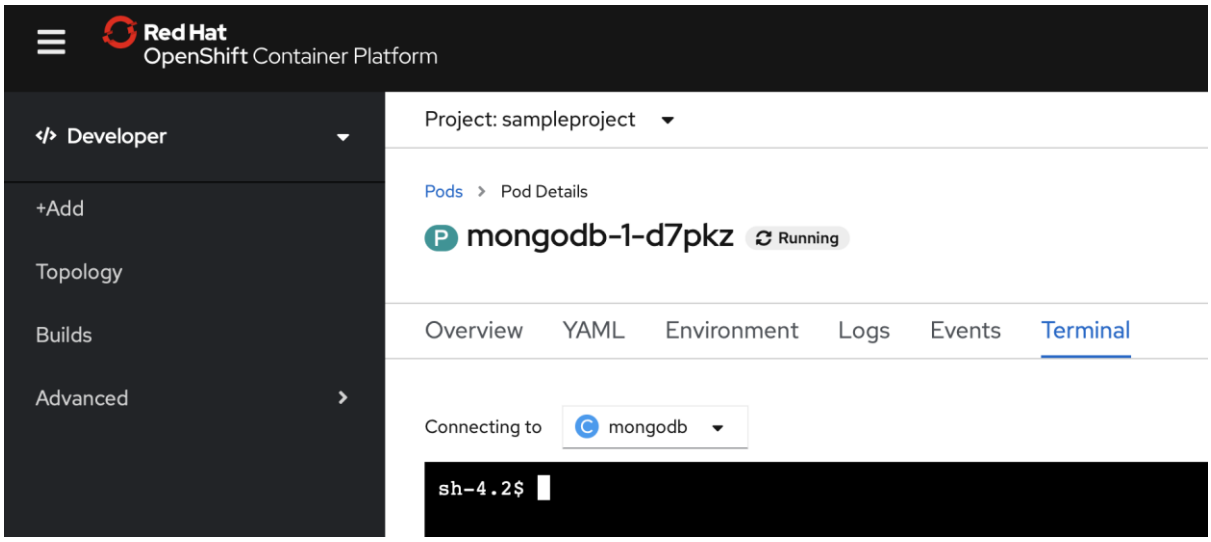
Step 8: Now from Topology select mongodb to access MongoDB database



Step 9: Click on ***mongodb*** Pod

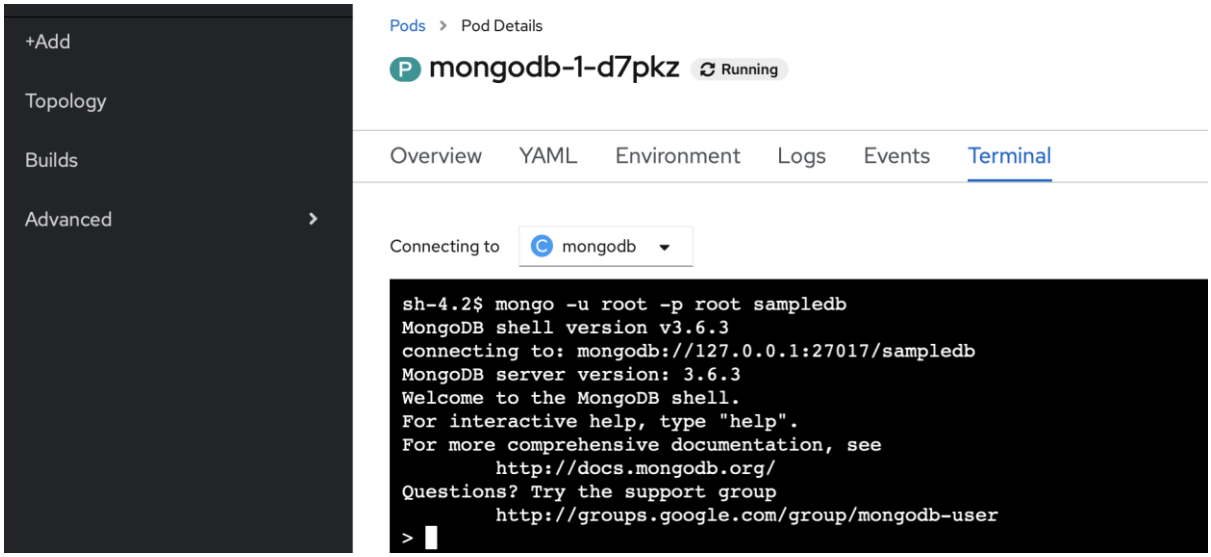


Step 9: Click on terminal



Step 10: Login into mongodb using below command

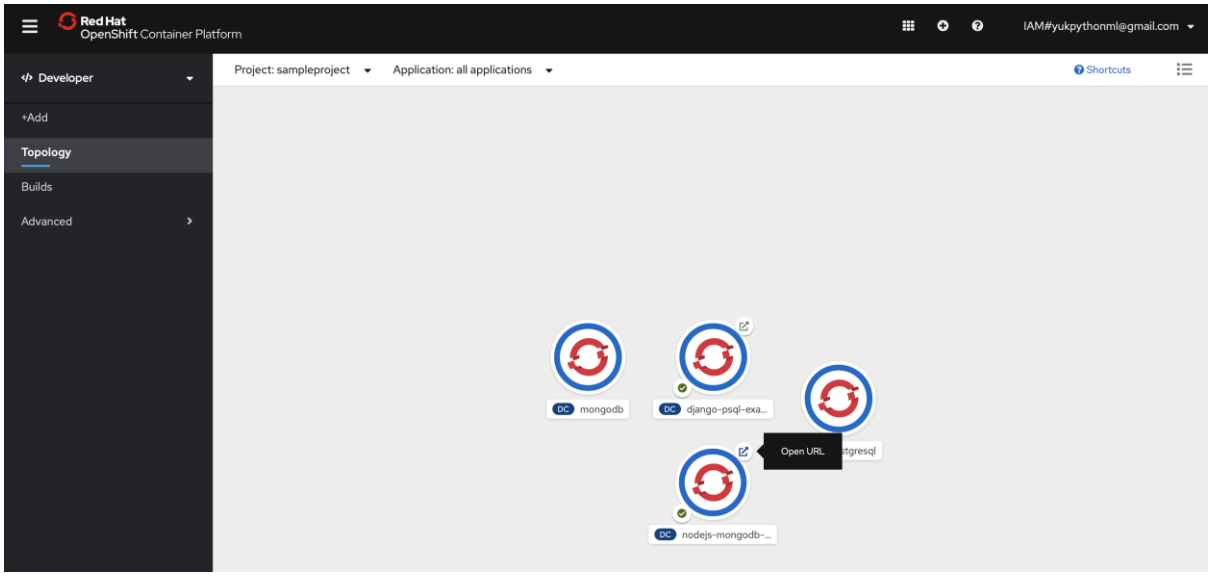
mongo -u root -p root sampledb



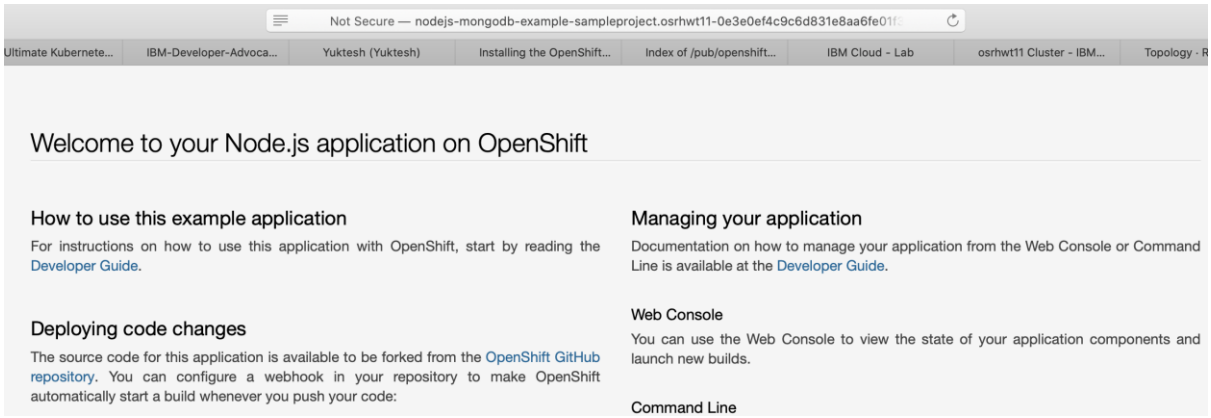
Step 11: Access counts db

D. Editing the recent uploaded template

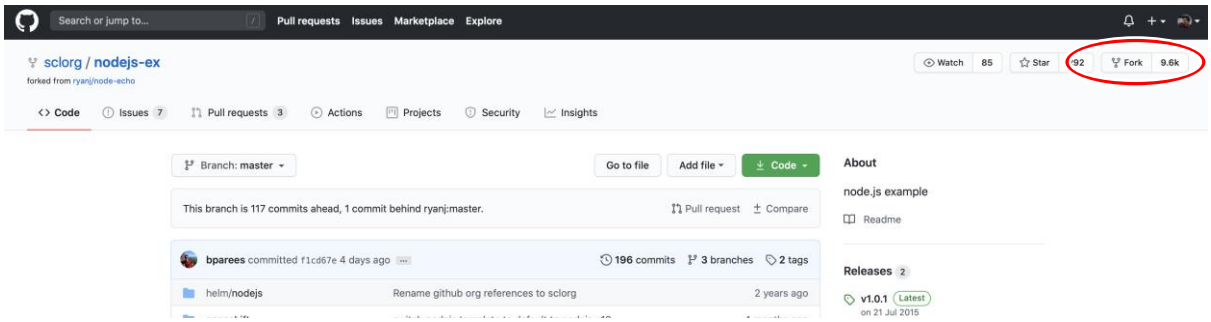
Step 1: From Topology click on Open URL icon of ***nodejs-mongodb*** pod to access the application



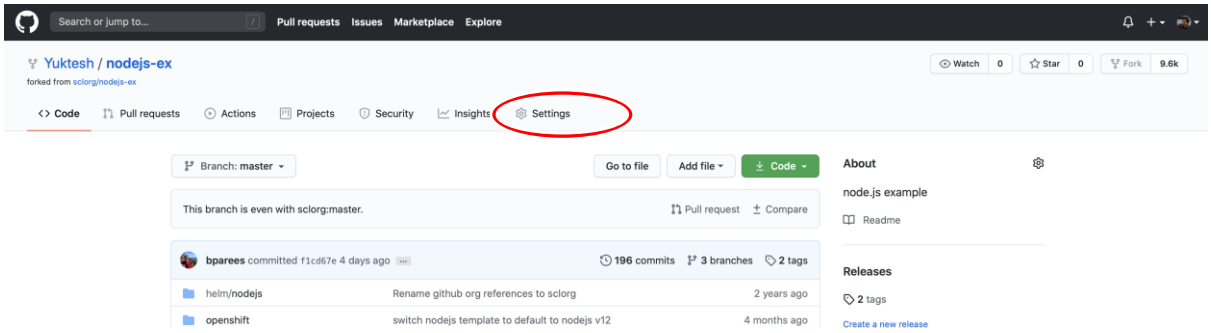
Step 2: Click on ***OpenShift GitHub repository*** under Deploying code changes



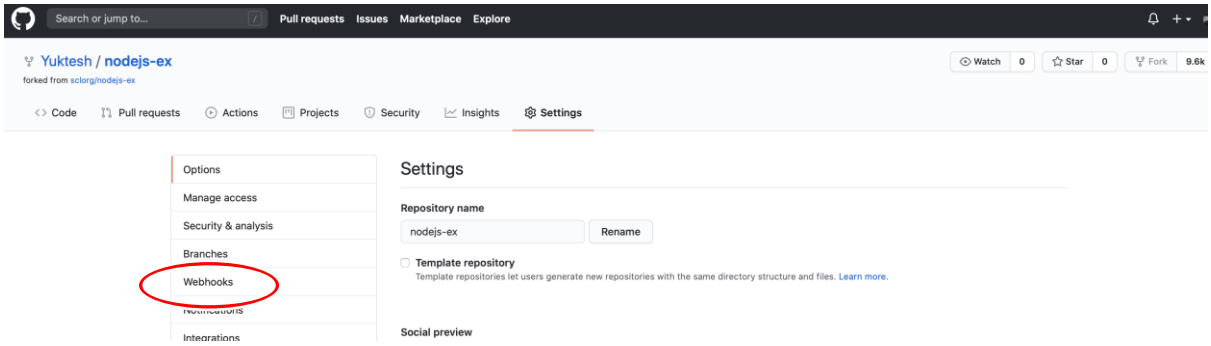
Step 3: Fork the application into your github repository by clicking ***Fork*** at right corner



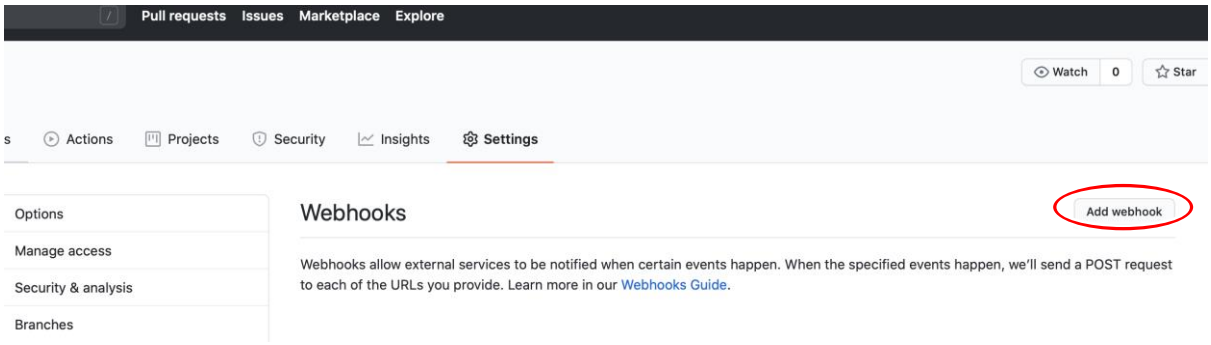
Step 4: To create a webhook click on ***settings***



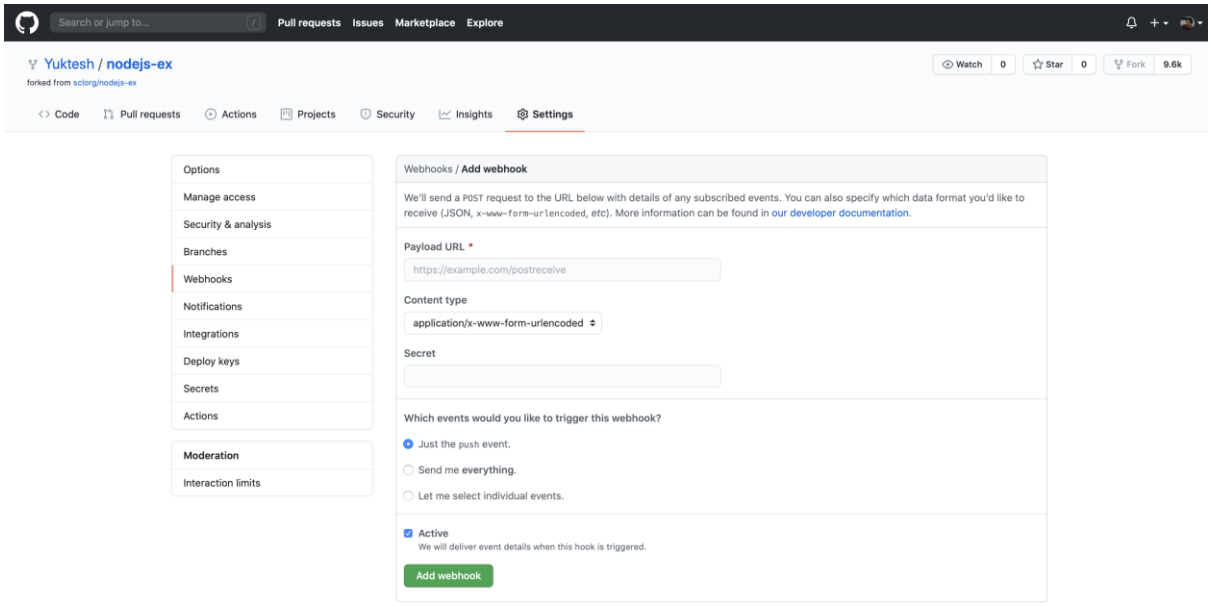
Step 5: Click on ***Webhooks*** at left side



Step 6: Click on ***Add webhook***

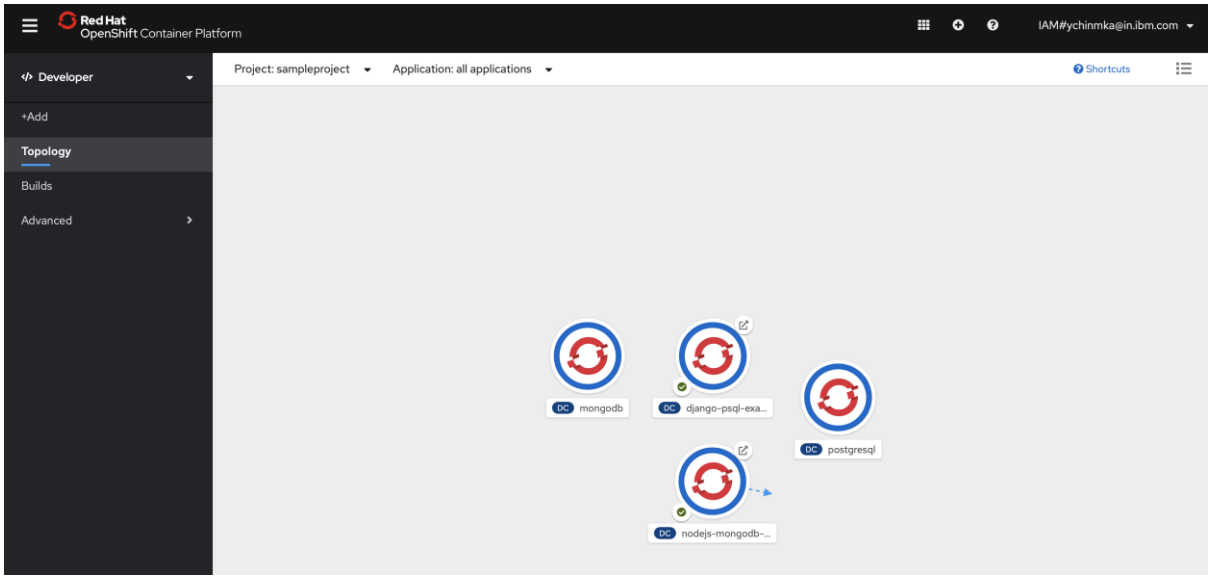


Step 7: Get Authenticate by giving Github password so that you will see below page

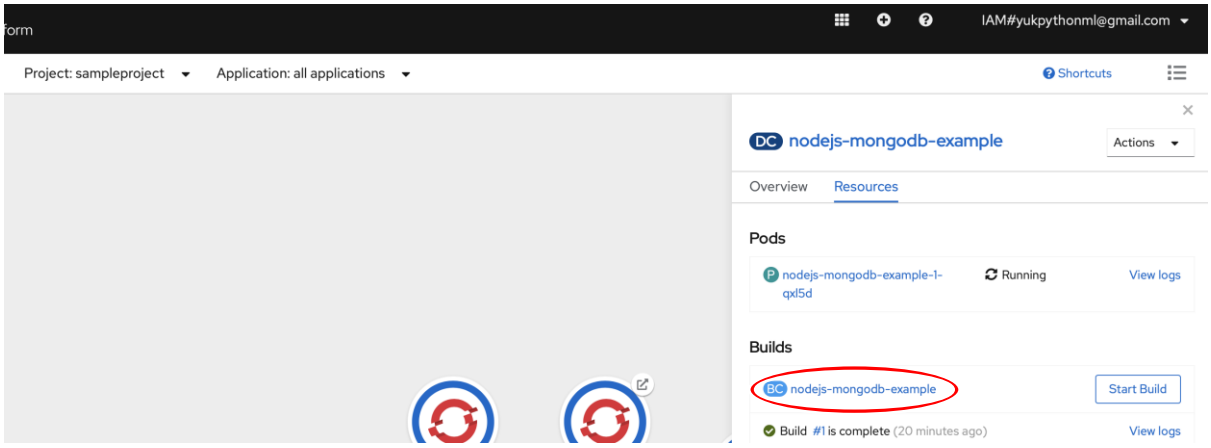


Now you need to bring payload URL from the OpenShift Container Platform

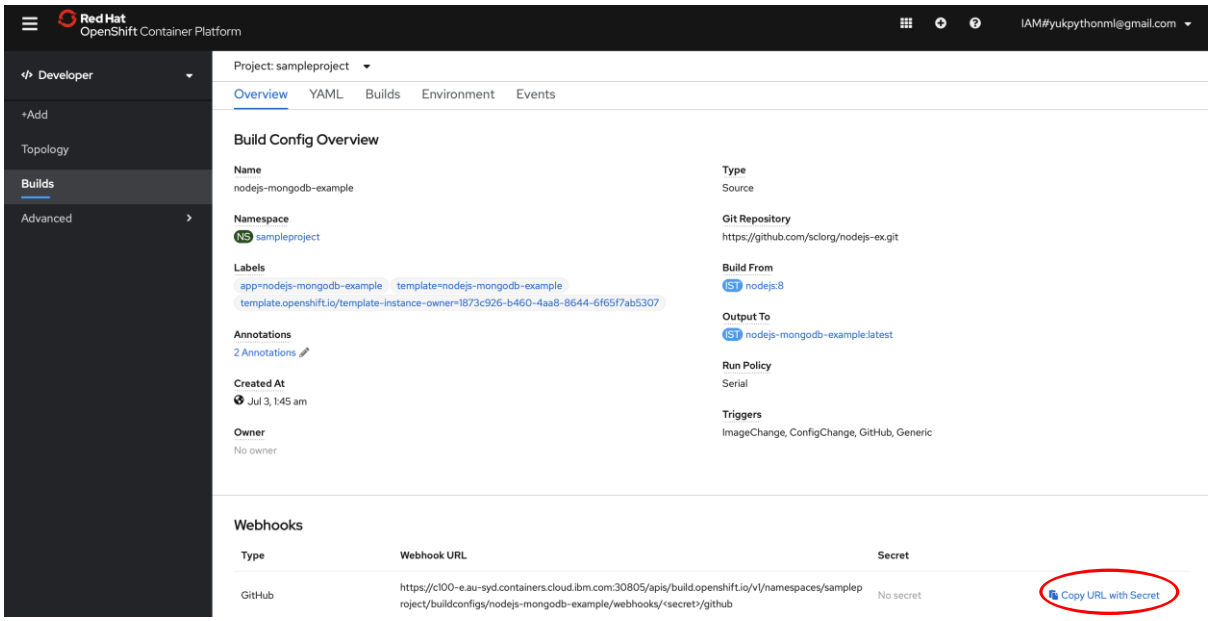
Step 8: Goto OpenShift Container Platform, Topology and click on ***nodejs-mongodb*** pod



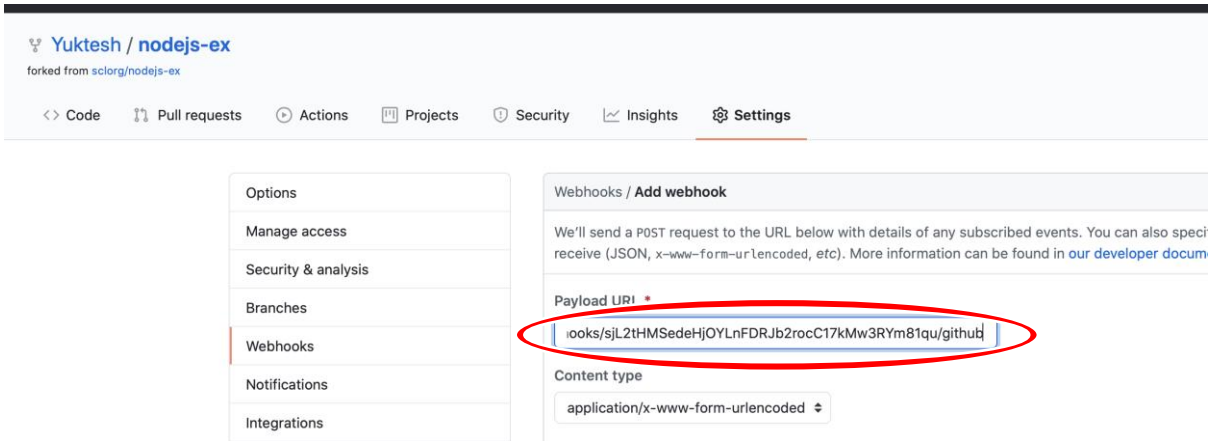
Step 9: Click on URL under ***Builds***



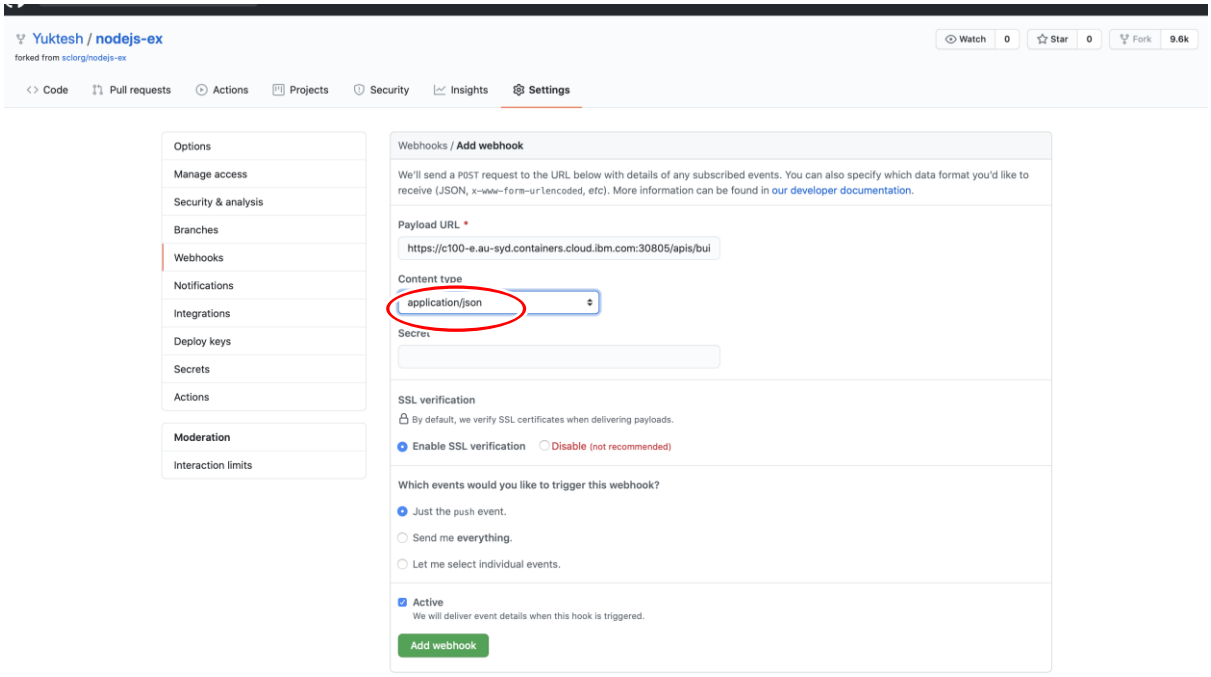
Step 10: Click on ***Copy URL with Secret***



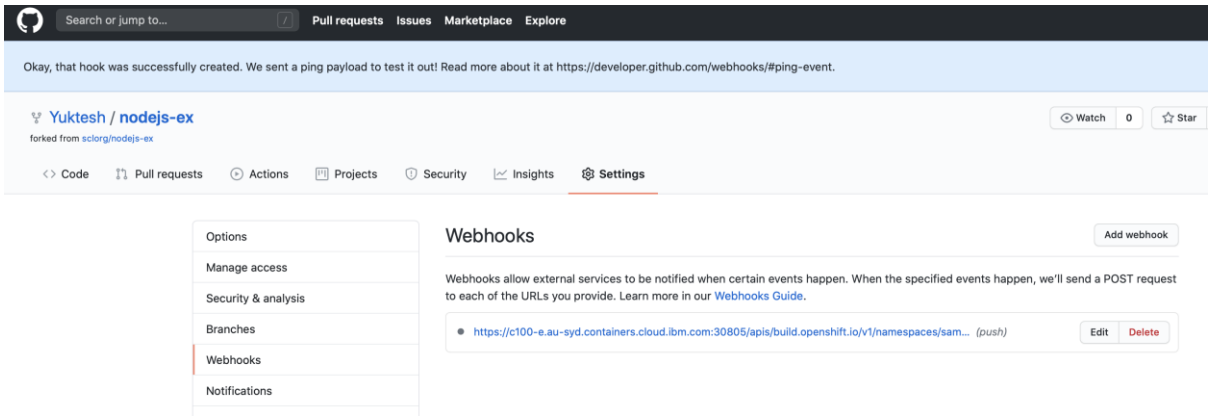
Step 11: Paste (ctrl+v) payload URL textbox of Github page



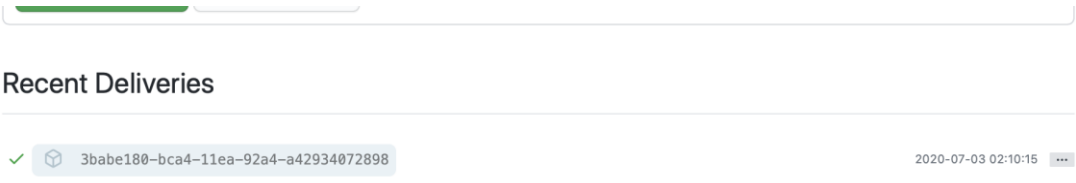
Step 12: Change ***content type*** to ***application/json*** and click on ***Add webhook***



Step 13: Can see now webhook got created, click on the link to check recent deliveries

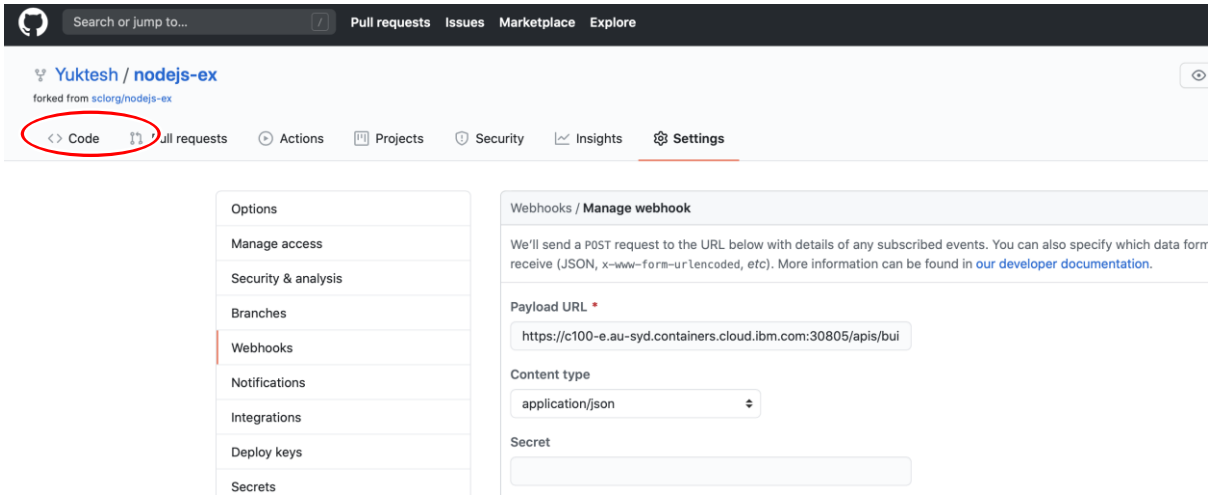


Step 14: Click on the Webhook link you can see Recent Deliveries

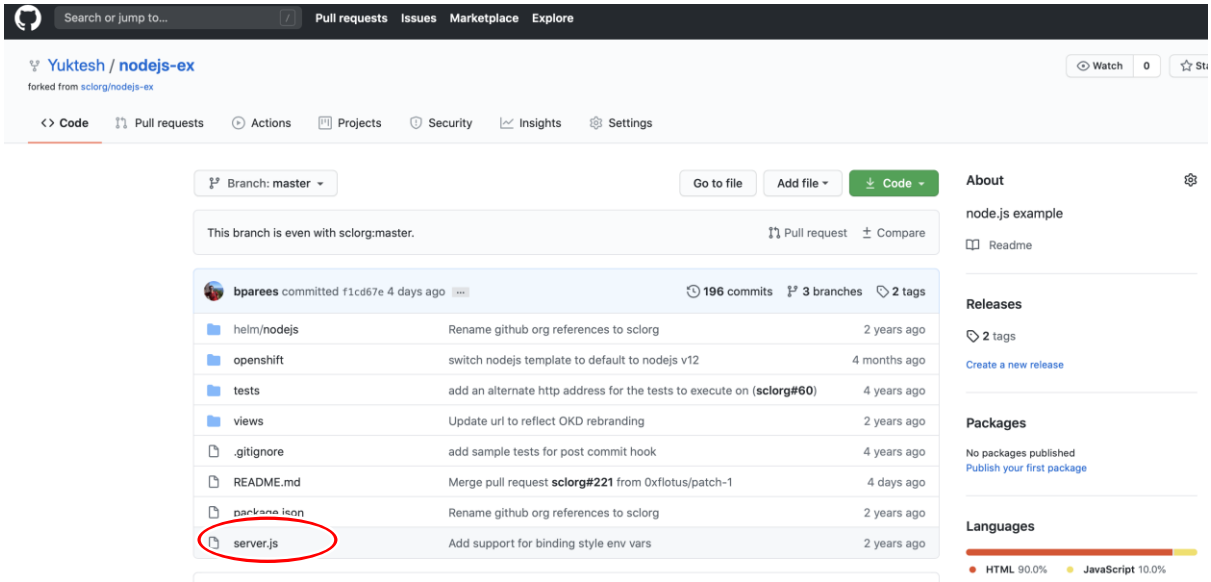


Now, let us do some modification in the code and observe the changes in recent deliveries

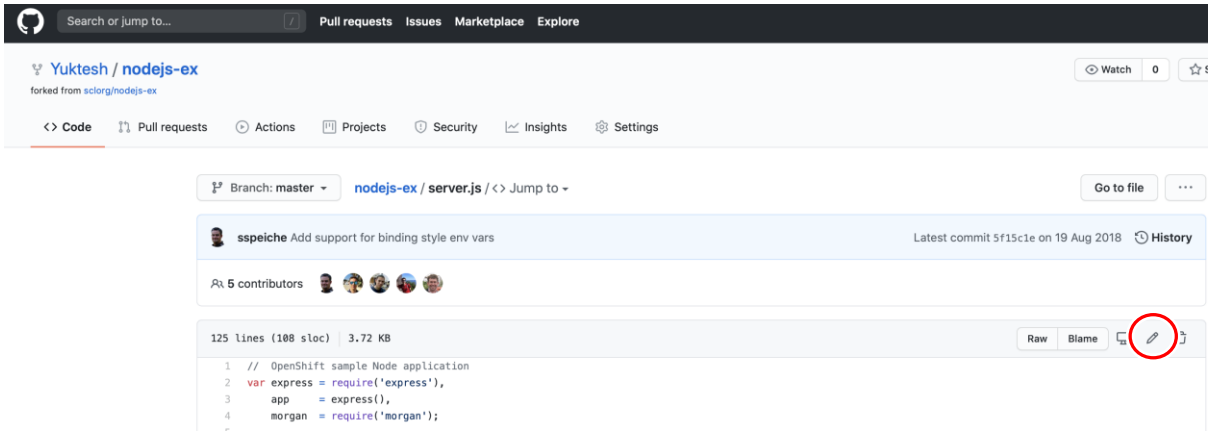
Step 15: Click on ***Code*** on left top



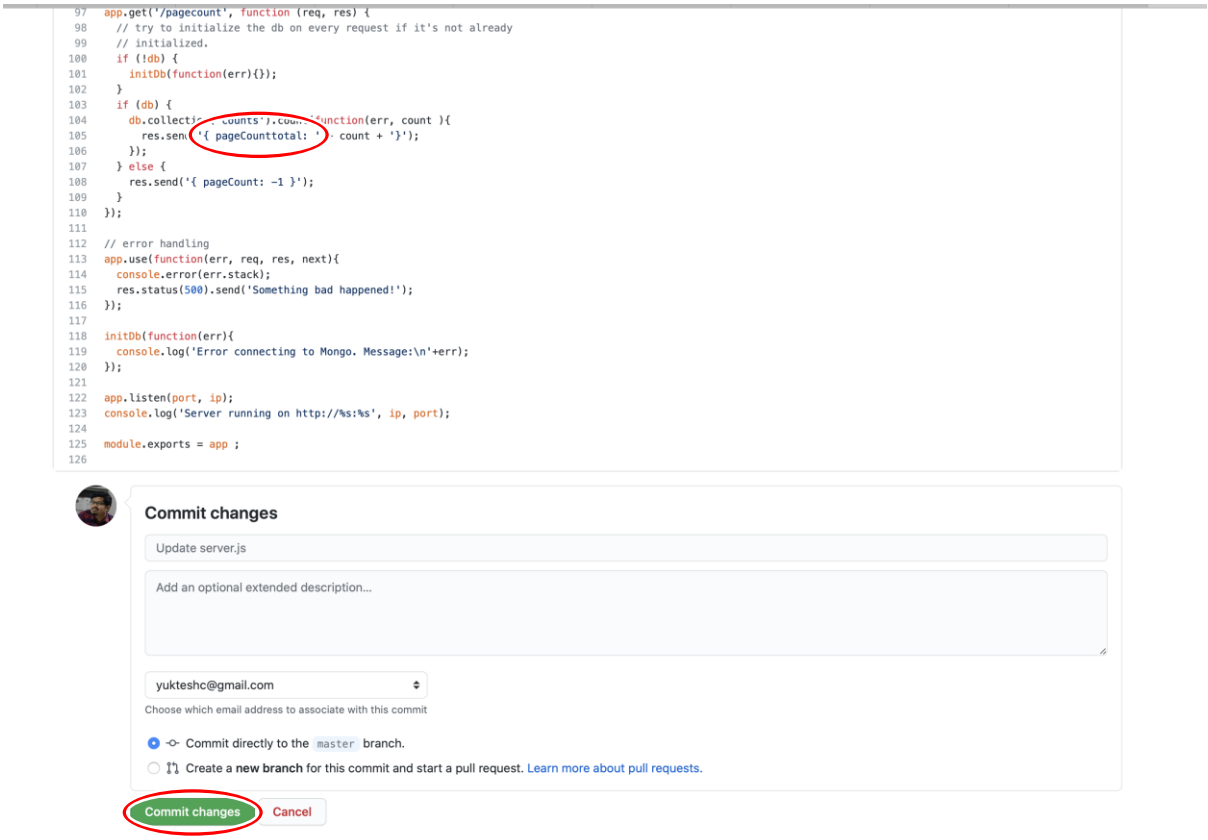
Step 16: Select server.js



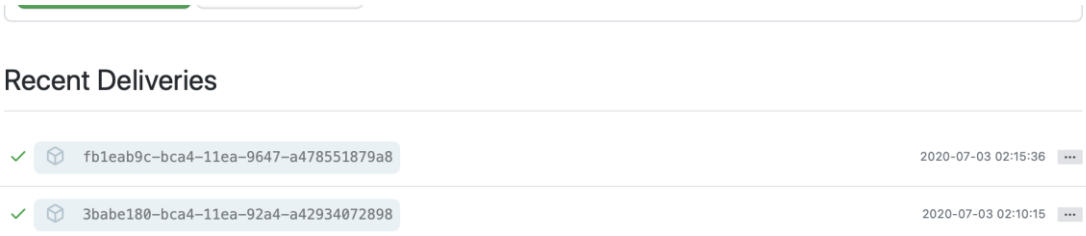
Step 17: Click on pen icon to edit the source code



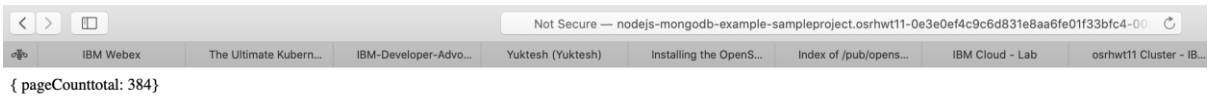
Step 18: Change line 105 ***pageCount*** to ***pageCounttotal*** and click on ***commit changes***



Step 19: Click on Settings -> Webhooks -> Webhook URL and you should be able to see updated Recent Deliveries



Step 20: Go to the URL of the ***nodejs-mongo*** pod and add ***/pageCount*** and for every refresh count will increase by 1



{ pageCounttotal: 384}

Refresh page

{ pageCounttotal: 386}

E. Creating a template from existing objects

Rather than writing an entire template from scratch, you can export existing objects from your project in YAML form, and then modify the YAML from there by adding parameters and other customizations as template form.

Step 1: Export objects in a project in YAML form

```
Yukteshs-MacBook-Pro:~ ychinmka@in.ibm.com$ oc get -o yaml --export all > mytemplate.yaml
Flag --export has been deprecated, This flag is deprecated and will be removed in future.
Yukteshs-MacBook-Pro:~ ychinmka@in.ibm.com$
```

Step 2: Type ***ls*** and check for ***mytemplate.yaml*** file which got created

```
Yukteshs-MacBook-Pro:~ ychinmka@in.ibm.com$ ls
Applications                            Untitled.ipynb
Desktop                                 Untitled1.ipynb
DockerComposeFile                       demo.ipynb
Documents                               dockerfile
Downloads                               emp.xlsx
Functions                               employee.csv
LR_Boston.ipynb                         file1
Library                                 generate-insights-from-multiple-data-sources
Linear Regression                       mytemplate.yaml
```

Step 3: Open and check the content in ***mytemplate.yaml*** by using ***cat*** command

```
Yukteshs-MacBook-Pro:~ ychinmka@in.ibm.com$ cat mytemplate.yaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Pod
  metadata:
    annotations:
      cni.projectcalico.org/podIP: 172.30.28.92/32
      cni.projectcalico.org/podIPs: 172.30.28.92/32
```