Kubernetes 201 Assignment Document

By Srinivasan Rajendran (M1043833)

Table of Contents

[1. Introduction 3](#_Toc37593316)

[2. Environment 3](#_Toc37593317)

[3. Assignment Detailed Explanation 4](#_Toc37593318)

[3.1 Assignment 1: Dockerize, Containerize and Push the Application 4](#_Toc37593319)

[3.1.1. Application Source 4](#_Toc37593320)

[3.1.2. Docker Build 5](#_Toc37593321)

[3.1.3. Docker Run 5](#_Toc37593322)

[3.1.4. Push images into Docker Hub 6](#_Toc37593323)

# Introduction

Kubernetes 201 assignment contains four sections, each of them explained, and I provided the screen shots/yaml/Dockerfile for appropriate sections. This document is the assignment ultimate output of this assignment. Below are the four sub-sections of the assignment.

* 1. Dockerize, Containerize and Push the Application
  2. Kubernetes Service & Cluster
  3. Scalability & Replication
  4. Scaling, Data Persistence

# Environment

Below is the assignment environment details of assignment.

|  |  |
| --- | --- |
| Component | Version |
| Operating System – Ubuntu x86\_64 | 18.04 |
| Minikube[1] | 1.8.1 |
| Docker | 19.03 |

***Instead using datacenter environment like E2C, GKE, in the current assignment using Minikube***.

Minikube[1] is a tool that makes it easy to run Kubernetes locally. Minikube runs a single-node Kubernetes cluster inside a Virtual Machine (VM) on your laptop for users looking to try out Kubernetes or develop with it day-to-day.

# Assignment Detailed Explanation

Below embedded the assignment details. Below sections explaining each of the assignment details with screenshots, description etc.,



## Assignment 1: Dockerize, Containerize and Push the Application

Created Django web application with Postgres database as backend. It is a simple web application and it has the index page it will displays the ‘hello world’ text. Application will listern in port 9000 in host as well as container and database (postgres) listerning in the port 5432 in both container and host. Please note application and database running in different containers. Below I provided the Django web application source code git url for the reference.

### Application Source

Django Source code url: <https://github.com/srinics/django-app.git>. In the source code there is directory called ***docker.*** Which contains below files.

* dockerfile-app – Contains Django application dockerfile

|  |
| --- |
| FROM centos:latest  RUN yum install epel-release gcc python3-pip python3-devel wget nc postgresql-devel -y  RUN mkdir -p /root/srini && cd /root/srini  COPY requirement.txt /root/srini  RUN pip3 install -r /root/srini/requirement.txt  COPY . /root/srini  RUN pip3 freeze Django  CMD while ! echo exit | nc dproject-cnt-db 5432; do sleep 10; done && /usr/bin/python3 /root/srini/manage.py migrate && /usr/bin/python3 /root/srini/manage.py runserver 0.0.0.0:9000  #ENTRYPOINT [ "/usr/bin/python3", "/root/srini/manage.py", "runserver", "0.0.0.0:9000"] |

* dockerfile-db – Contains postgres db dockerfile

|  |
| --- |
| FROM postgres:latest |

Below I created docker images using below command and attached the log file building.

### Docker Build

|  |
| --- |
| docker build -t dproject-img-db -f ./docker/dockerfile-db ./docker/temp/  docker build -t dproject-img-app -f ./docker/dockerfile-app . |

Here

–t - specifies the name of the image - example dproject-img-db

-f – specifies the dockerfile file path - example docker/dockerfile-db

<path> - Third argument species the path.

Also attached the docker build command logs for reference.



Below is the screenshot of docker images output.

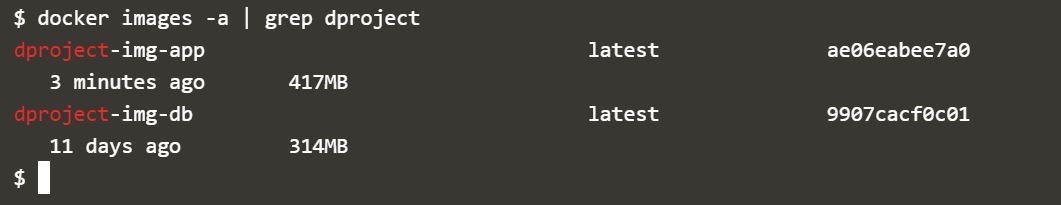


Figure 1: docker Images

### Docker Run

|  |
| --- |
| docker run -itd --name dproject-cnt-db -p 5432:5432 --env-file ./docker/db.env dproject-img-db  docker run -itd --name dproject-cnt-app -p 9000:9000 --link dproject-cnt-db:dproject-cnt-db dproject-img-app |

Here

-itd – specifies interactive(i), demonize(d), Allocate a pseudo-TTY(t)

--name – specifies name of the container

-p – species port <HOST\_PORT>:<CONTAINER\_PORT>

--env-file – species the env file which will used during docker run

--link – specifies the network below container

Please note dproject-cnt-app container depends on dproject-cnt-db ( postgres db). In case user wants to run docker manually in that case first run dproject-cnt-db command and wait for still 5432 port up because it might take few seconds to up then run dproject-cnt-app.

Also attached the docker run command logs for reference.



Below is the screenshot of docker ps once executed after docker run commands.



Figure 2: docker ps

### Push images into Docker Hub

I created account (srinics) in docker hub (<https://hub.docker.com>) and followed below steps to push the docker images. As a prerequisites created two repositories

* srinics/django-project – repository for Django app images
* srinics/postgres-db – repository for postgres db images

#### Docker Login

Login into docker using docker cli commands as like below screenshot. Please note provide the password of docker hub.

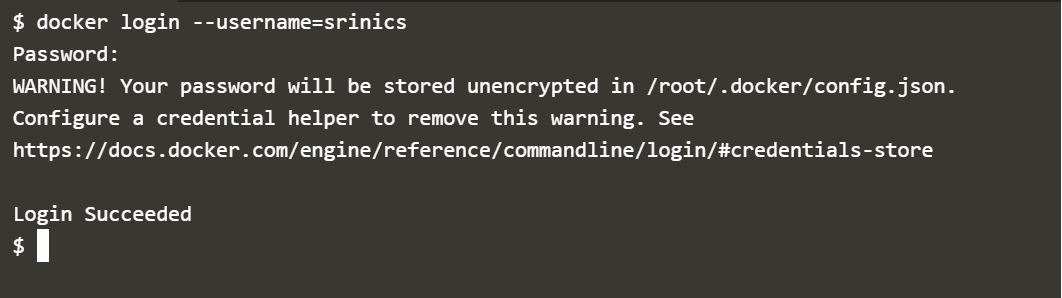


Figure 3: docker login

#### Docker image tagging

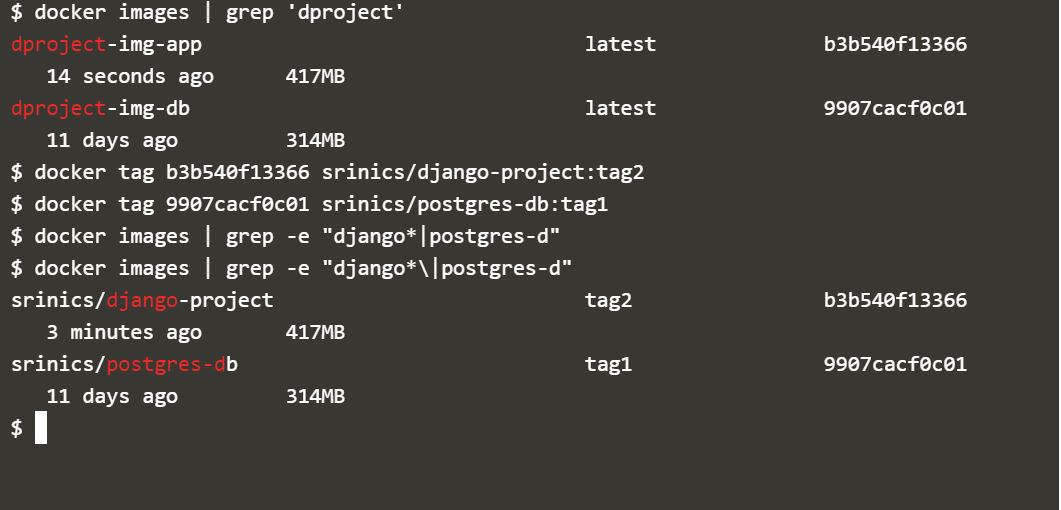
Tag the created images using docker cli commands as like below screenshot.

Figure 4: docker images

#### Docker push

Push the docker images into repositories using docker cli commands as like below screenshot.

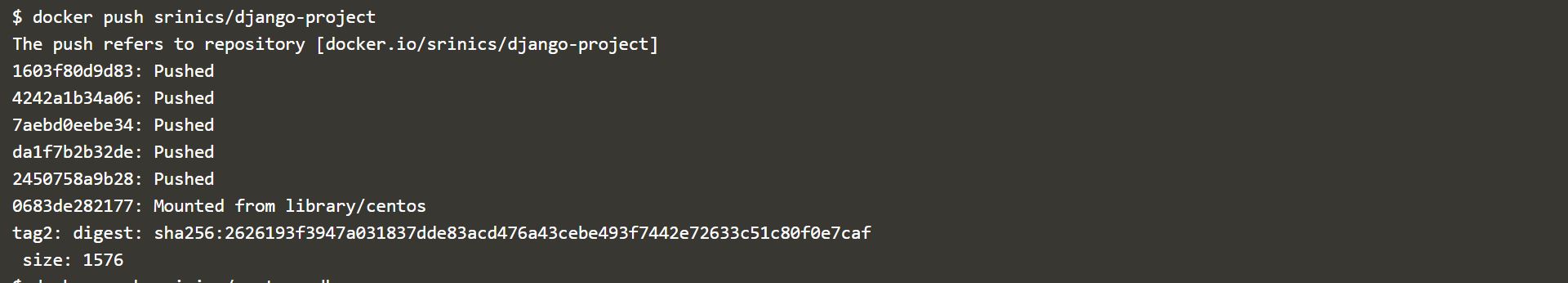
Below is the screenshot of docker push for django app image; which push srinics/django-project:tag2 image into the repo(srinics/django-project).

Figure 5: docker push - django app

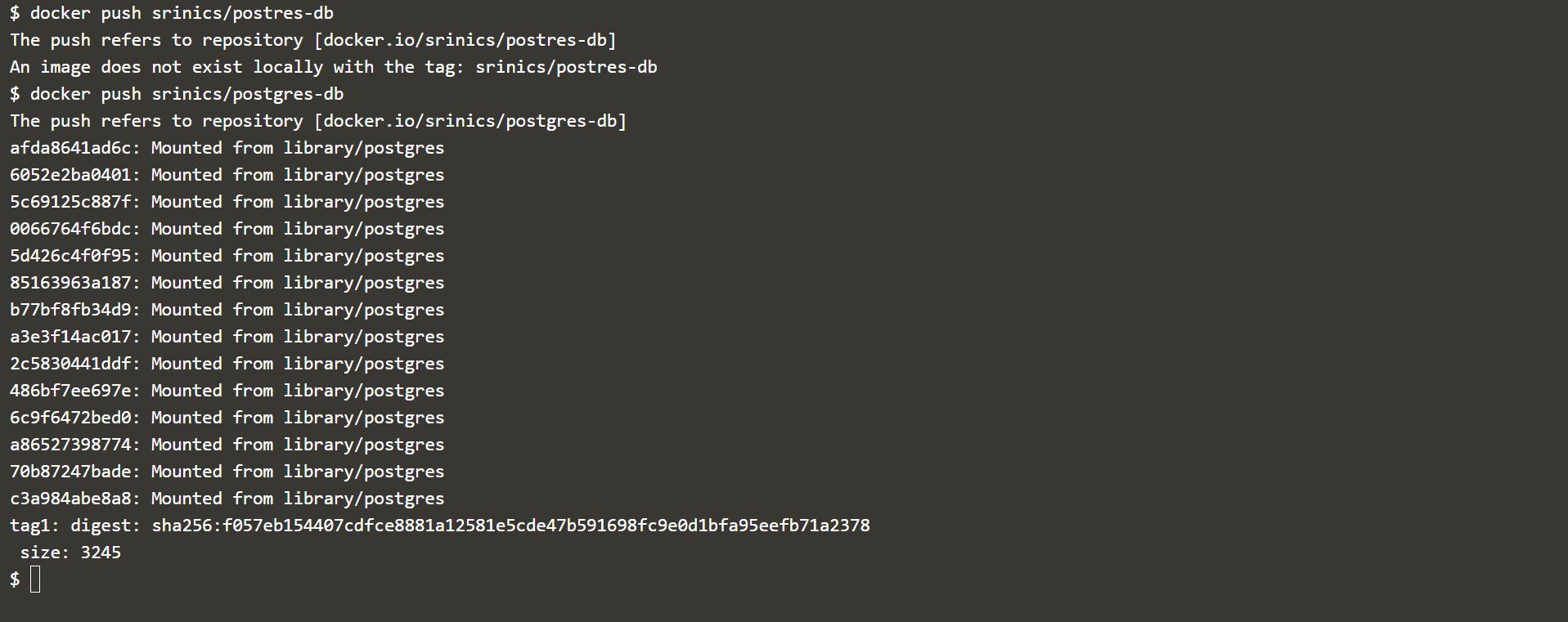
Below is the screenshot of docker push for postgres db image; which push srinics/postgres-db:tag1 image into the repo(srinics/postgres-db).

Figure 6: docker push - postgres db

#### Docker Hub screenshot

Screenshots of docker hub repositories after pushing images.

Figure 7: docker hub - django project tag2 repo image

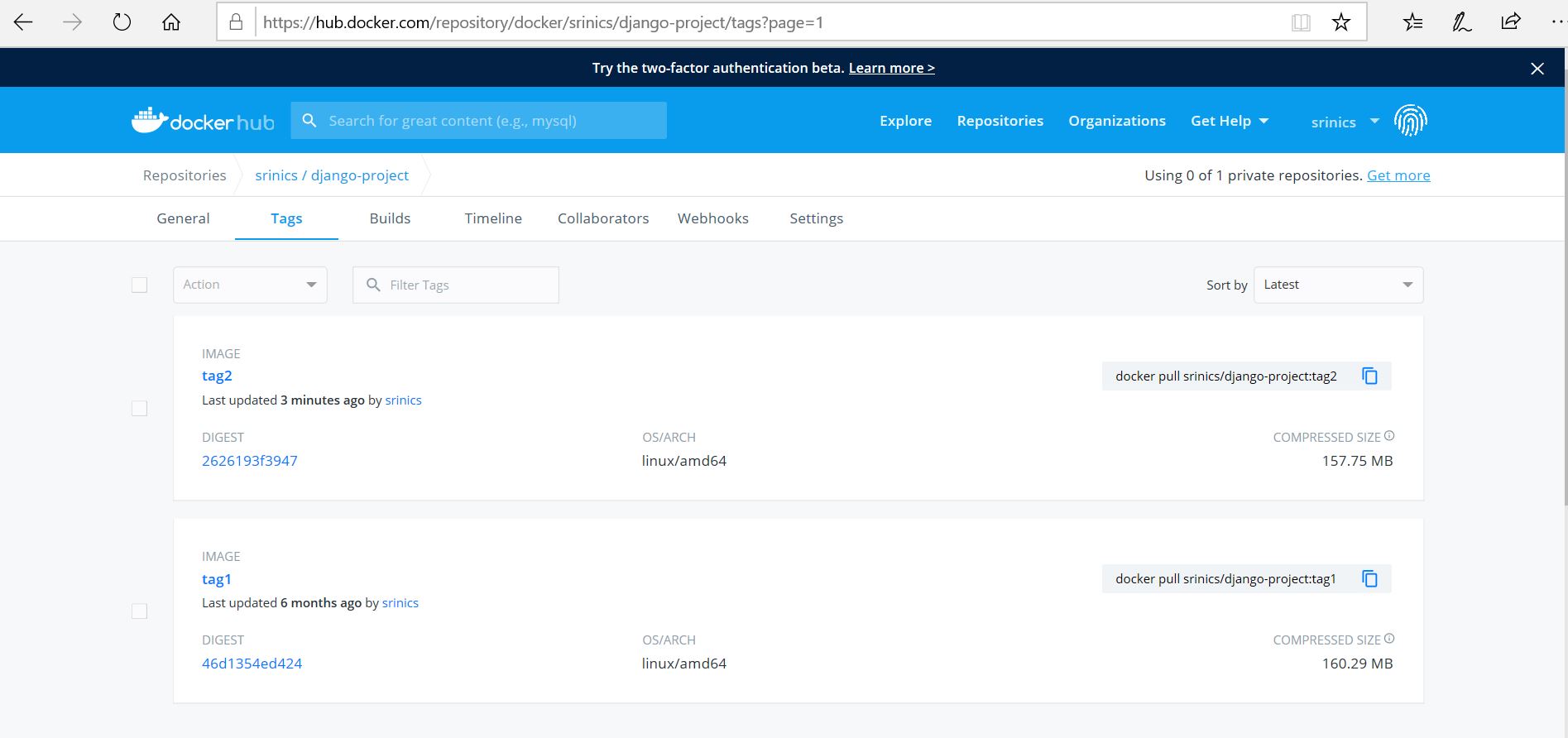


Figure 8: docker hub - postgres-dbt tag1 repo image

