**Graduate Independent Study:**

**Survey Paper**

**DevOps and Docker**



**By:**

**Bony Roy**

**Student ID: 898161054**

**CPSC 599**

**Spring, 2020**

**Professor: Dr. Doina Bein**

**Department of Computer Science**

**California State University, Fullerton**

**April 27, 2020**

**Table of Contents**

Abstract………………………………………………………………………………………………...3

1.0 Introduction…………………………...............................................................................................3

2.0 Literature Review..............................................................................................................................4

3.0 Discussion……………….................................................................................................................8

4.0 Conclusion………………................................................................................................................16

References..............................................................................................................................................17

**Table of Figures**

Figure 1: Continuous deployment pipeline……………………………………………………………..5

Figure 2: Architecture of VMs vs Containers…………………………………………………………..8

Figure 3: Docker architecture…………………………………………………………………………...9

# **Abstract**

Lightweight cloud technologies like Docker containers are coming into the picture and gaining popularity because they allow business organizations and users to deploy applications more efficiently, faster and efficiently in any environment than traditional VM or virtual machines. As more companies are now adapting to use docker, it is very important to know how the basic DevOps works. Docker is only a small part of the whole DevOps world. This paper aims to look into an in-depth analysis of DevOps and terminologies associated with it. Also, host one project into DockerHub to understand how it works in the real world.

Keywords: Cloud computing, Container, DevOps, Docker, Virtual Machines.

# **1.0 Introduction**

**DevOps:**

DevOps as per the Wikipedia definition is a set of practices that combines software development and information-technology operations which aims to shorten the systems development life cycle and provide continuous delivery with high software quality. It also integrates processes, people, and technology for an automated software delivery that is scalable, agile, and cost-effective. DevOps combines two terms development and operations. It is basically the start-to-end responsibility of deploying an infrastructure through product design, requirement gathering, build, testing, application deployment, and at the end monitoring. Basic idea is to deliver a repeatable and consistent environment for faster software development.

There are several phases in DevOps Continuous development, testing, integration, deployment, monitoring, and virtualization. Continuous software development is broken into phases and we can use GIT for that, testing phase users can do using selenium, for integration users can use Jenkins, for the deployment of application users can use Docker, for monitoring users can use tools like a stack, Splunk, etc. DevOps has several benefits like minimization of system failures, increased cooperation between teams, consistency, increased deployment.

**Cloud Computing:**

On the other hand cloud computing is basically on-demand storage, system, and computing power delivered over the internet without active management by the user. Nowadays cloud computing is a very important concept and most of the organizations use the cloud. Although the concept of cloud computing is fairly new the similar concepts of cloud computing existed from long before. “Cloud computing can be defined as a computing environment where computing needs by one party can be outsourced to another party and when need be arise to use the computing power or resources like database or emails, they can access them via internet”[8]. We basically move our data from one PC(personal computer) or desktop to large data centers maintained by some large corporations. The main benefit of cloud computing is that consumers or customers do not have to pay upfront for the infrastructure, or software installation, or manpower to manage the infrastructure. “The main goal of cloud computing is to make a better use of distributed resources, combine them to achieve higher throughput and be able to solve large scale computation problems''[8]. Some of the key players in the cloud computing industry are Microsoft(Azure), Amazon(AWS), and Google(GCP). Consumers can select cloud services based on their needs.

The basic idea of this paper is to understand both DevOps and cloud computing. Also, to host one of the personal projects into a docker hub and explain the steps performed.

# **2.0 Literature Review**

In today’s software development industry building and developing software is not only about writing code, but we also need to follow many steps to deliver working software. Now, most software companies support hundreds and tens of millions of users with small technical teams. “ This efficiency constraint makes scalability even more challenging, as it means that you need to design and build systems that become more efficient as they grow”[7]. Before deep diving, we need to discuss a few terms important with respect to DevOps.

**CI/CD:** It generally refers to combining continuous integration(CI) and continuous delivery(CD). They are set of operating principles and practices that help the software development teams to deliver clean codes more frequently and efficiently.

**Microservices and container:** Microservice is a software architecture pattern, where complex software applications consist of independent and small processes communicating with each other using language APIs. The microservices are highly decoupled and only aim to build small tasks of any bigger software. On the other hand, a container is a part of the software that packages or bundles up the code and of the dependencies. So, the concerned application could run reliably and quickly.

Below are the few steps to build scalable and easily manageable software to automate any software development process possible.

**Build:** In this step, we mainly combine the source code and its related dependencies to build a runnable/executable instance of a product that we end users can use. Programming languages like C++, Java needs to be compiled but Python, Ruby doesn’t require compilation.

**Testing:** Whenever we are building any application the first thing we should automate is testing.

“The main reason why automated testing is a sound investment is that the overall cost of manual testing grows much faster over time than the overall cost of automated testing”[7]. Manual testing doesn’t need upfront investment, all we need are testers. Although the starting cost is low, we need to invest much more over time. On the other hand “Automated testing” needs upfront investment, as we need to set testing software and use continuous integration servers for deploy. But once everything is set up the cost is low. “Two tools that are worth recommending for end-to-end tests are Jmeter and Selenium. Jmeter is great at testing low-level Hypertext Transfer Protocol (HTTP) web services, HTTP redirects, headers, and cookies and it is also a great tool for performance and load testing. Selenium, on the other hand, allows you to remotely control a browser from within your tests”[7].

**Build and Deployment:** Another step is to automate the build and deployment pipeline. Manual deployments take time. Mainly because in a scalable architecture if we increase the number of services and servers, we need to involve many people and that adds up to the cost. Software release gets complicated, and the integration/testing cycle becomes longer. 3 terms very important for build deployment are continuous integration, continuous delivery, and continuous deployment.

**Continuous integration:** Continuous integration is basically a set of operations that enable the application team to implement small software changes and Checking-in code to version control repositories like GIT. The developers commit their code to a common branch and have automated test cases executed on a shared codebase anytime that code changes are made. This step of automated tests and committing codes to a shared branch allows software developers to detect any issue in integration earlier and with much less effort.

**Continuous delivery:** It starts where continuous integration ends. It automates the application delivery to some infrastructure environments. Basically “The continuous delivery pipeline deploys your software to a set of test environments (usually called dev, testing, or staging)”[7]. One of the important steps of that process is the software is built, packages, and deployed in a uniform way without human interaction.

**Continuous Deployment:** It is the final stage in the pipeline. Here “code is tested, built, deployed, and pushed to production without any human interaction”[7]. So, any commit to a shared code repository triggers the deployment to production.

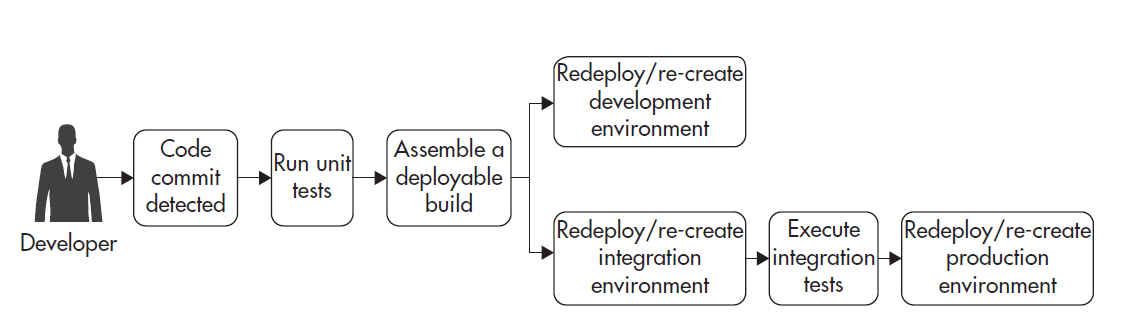


Figure 1: Continuous deployment pipeline [7]

As per the above architecture, the software should move through the software deployment pipeline automatically, as an example if we do one commit on the integration branch that will automatically trigger integration tests. After the build is assembled the software code will be deployed into the development and testing environment. After all the successful automatic end-to-end tests in the testing environment, the software will be deployed into the production environment.

Various tools and technologies are available nowadays to implement continuous deployment. But building this continuous deployment pipeline is really challenging for the DevOps. DevOps needs the know programming and tools as a developer and at the same time, they need to know how to deploy software and setting up the environment, servers as Ops professionals.

**Tools related to continues integration and continuous deployment:**

To successfully perform continuous integration and continuous deployment we need to know many tools nowadays. GIT, which is a distributed version control system, we can use for tracking changes in source code. Maven is a build tool. The deployment pipeline could be controlled using Jenkins or Atlassian Bamboo (Both are used for continuous integration and continuous delivery). In addition to Jenkins, we need to decide how to configure servers. If we like to increase the number of servers over time we must have tools to manage their configuration. So, tools like Chef or Puppet are popular for all of this server configuration management. To deploy the application to production we might integrate with the cloud service providers to create server images. AWS has an amazon machine image(AMI) for this.

**Version control tools:**

“Version control systems” are a type of software tools that help software developers to manage changes in “source code” over time. This type of version control software keeps track of every modification in the codebase in a special kind of DB(database) system. So, if in any case developers make mistakes, they can turn back to the previous version of code available. Also, in some cases compare the new version of code with the previous version of code to detect issues and fix the mistakes with very minimum effort.

Tools like GIT, Bitbucket, Github are used.

**Software Build tools:**

Software build tools help to automate the creation applications from source code. They automate a variety of tasks, s]like downloading dependencies, packaging, compiling, tests, and deployment, etc. We can trigger them through the command line, IDE or by continuous integration tools after checking the code from a repository and to “build machine”. The different programming languages have different build systems. Tools like Gulp(JavaScript task runner), NAnt (.NET environment), Grunt(JavaScript task runner), Maven(Primarily used for Java), Docker (software container platform)- preferred replacement for Virtual Machines (VMs).

**Configuration management tools:**

The configuration of your computer or server is one of the important parts of any application. For maintaining any enterprise-level software every configuration changes we must track as it generally has hundreds or thousands of computers. The configuration management tools mainly control this type of change, help to monitor, or ensure that organizational infrastructure is correctly configured.

Basically used for maintaining inventory(servers etc.). Before choosing the configuration management(CM) tool we need to know whether our application needs servers, cloud networking or routing, VPN(Virtual private network), Load Balancers, Access policy, Autoscaling, etc. Tools can be used are Terraform, SaltStack, Ansible, Chef, Puppet, etc.

**Testing tools:**

We all know testing is an important part of software development. Although the previous testing is to be done with manual testers. But now we heavily depend on Automatic testing. Testing can also have two categories- white box testing and black-box testing. White box or “see-through box” testing involves a test of design, internal structure, and code of any software. Tests security issues, the flow of specific inputs, expected outputs. On the other hand, in the case of black-box testing software tester doesn’t know the internal design of an application or software, so it is like a “Black box”. Testing tools heavily used nowadays are Ranorex, SoapUI, Checkmarx, Specflow, Pally, Browsersync, Pytest, Mocha, Jmeter, Selenium, etc.

**BI/Monitoring/logging tools:**

BI (Business intelligence) is a process that converts raw data to take informed action. It helps to visualize or analyze data that gives managers, stakeholders, or leaders to make a better decision. They can analyze by combining internal and external data. Google Analytics is an important reporting tool. “The main motivation to automate monitoring and alerting of your systems is to increase your availability by reducing mean time to recovery (MTTR)”[7]. MTTR can be defined as below:

MTTR = Time to discover + Time to respond + Time to investigate + Time to fix[7]

Through the monitoring tools, IT(Information Technology) teams monitor entire system health, infrastructure, servers, cloud deployments, and networks. LogicMonitor, Datadog are tools for monitoring. Logging tools help to analyze logs, so that the system's high availability and low latency can be ensured. Logstash is a tool for help processing logs. Other well-known tools are App dynamics, Moogsoft, Splunk, Sentry, Datadog, Elasticsearch, etc.

**Continuous integration tools:** CI(continuous integration) and CD(Continuous development) are foundations of DevOps. Software developers need to integrate their codes multiple times into a shared repository, and once they push their changes, the CI server runs automatic unit tests, then informs the developer about success or any failure. Tools used for CI are Semaphore, Drone.io, Wercker, Codeship, Travis CI, Jenkins, etc.

**Cloud Tools:**

Cloud tools deliver their services over the internet and business organizations can use those resources on the go without maintaining their own infrastructure. They can choose to pay as you go, model, which basically allows users to pay for the services they have used. “Google cloud platform” provided by Google, “Amazon Web Services” by AWS, “Azure” by Microsoft, Openstack, Heroku, Rackspace are few important cloud providers.

3 main categories in cloud computing are Infrastructure as a Service(IAAS), Platform as a Service(PAAS), Software as a Service(SAAS).

Platform as a service(PaaS):

Platform as a service provides a solid platform that allows clients or customers to run, develop, and manage their customized applications (e.g. Google AppEngine1) without the complication of maintaining and building the hardware and software tools required for launching or developing the application.

Infrastructure as a service(IaaS):

Infrastructure as a service is a form of [cloud computing](https://searchcloudcomputing.techtarget.com/definition/cloud-computing) that provides virtualized computing resources over the internet. Here a cloud provider hosts infrastructure components such as a server, networking hardware, storage, and virtualization layer.

Software as a service(SaaS):

Software licensing and delivery model in which a third-party software provider centrally hosts software, and the software is licensed on a subscription basis (e.g. Salesforce1).

Based on the security and cloud policy cloud can be divided as a Public Cloud, Private Cloud, and Hybrid cloud.

# **3.0 Discussion**

**Virtual Machine vs Containers:**

**Virtualization** is basically developing a virtual model of everything(not actual), including computer networks, storage systems, virtual computer hardware, etc.. Although, it is a very old concept, it is extensively used in cloud computing environments. “**Virtual machines** (VMs) have been the backbone at the infrastructure layer providing virtualized operating systems (OSs)”[3]. On the other hand, containers are basically based on a very lightweight virtualization concept. Containers consume less time and fewer resources. “Although VMs and containers are both virtualization techniques, they solve different problems”[3]. Containers mainly used for platform-as-a-service (PaaS) software- it focuses on interoperability and portability, while still adhering to OS virtualization principles. “VMs, on the other hand, are about hardware allocation and management (machines that can be turned on/off and be provisioned)—that is, there’s an infrastructure-as-a-service (IaaS) focus on hardware virtualization”.

Figure 3 below explains the structural difference between VMs and Containers:

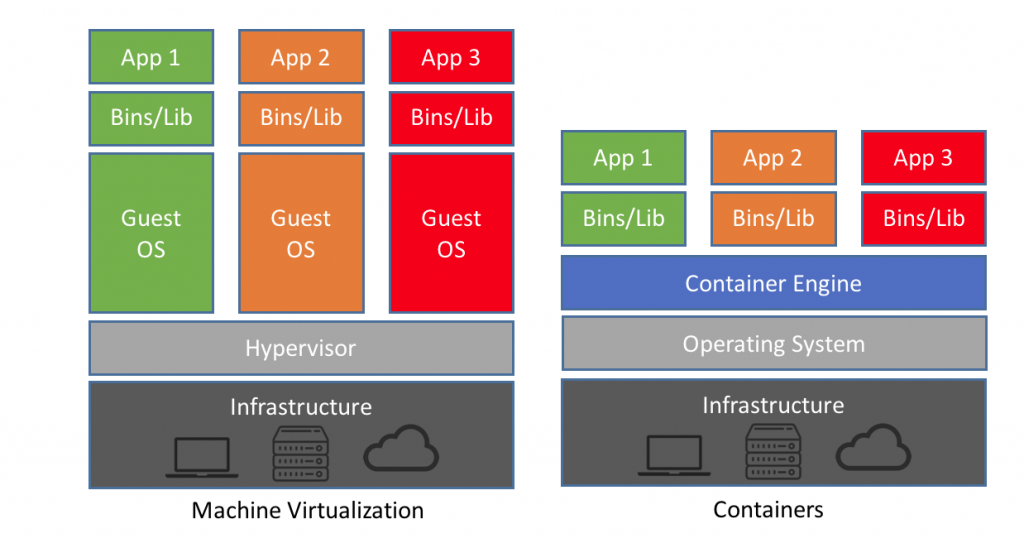


Figure 2: Architecture of VMs vs Containers[12]

Virtual machine uses an extra layer between the host operating system and guest operating system. This layer is known as a Hypervisor. “Successful tools like Docker are frameworks built around container engines that allow container engines to act as a portable mechanism to package and run applications as containers”[3]. “As docker does not use any guest operating system that makes a big difference in performance between a docker container and a virtual machine technology”[12].

We will describe below a simple implementation of the docker and hosting application into Docker Hub.

**Docker:**

“Docker bundles programming into institutionalized units considered holders that have everything the product needs to run including code, libraries, runtime, framework, and devices”[6]. “Docker container enables the sharing of operation system and supporting libraries, which is more lightweight, prompt, and scalable than Hypervisor based virtualization”[2]. That’s why it makes docker ideal for application deployment in a microservice architecture. An old concept like virtualization is heavily used in cloud computing. But the operating principle of docker is different. For future work, it will be good to explore cloud application deployment using Docker containers and microservices.

**The internal structure of Docker:**

There are 4 main components of Docker: Docker Images, Docker Registries, Docker Client and Server, and Docker Containers. We will look into every component briefly.

**Docker Images:** Two methods are used to build images. the 1st way is to create an image by reading the only template. “The images of the operating systems create a container with an ability of complete running OS”[12].

Most of the images of the operating systems are base images. We can do changes in the Base image for adding new apps but we need to build a new image. “The second method is to create a docker file. The docker file contains a list of instructions when the “Docker build” command is run from the bash terminal it follows all the instructions given in the docker file and builds an image”.

**Docker Client and Server:** It is a client server-based architecture. The docker server gets the request from the docker client and then processes it accordingly. “It generally comes with a command-line client binary and completes RESTful API. Docker daemon/server and docker client can be run on the same machine or a local docker client can be connected with a remote server or daemon, which is running on another machine”[12].

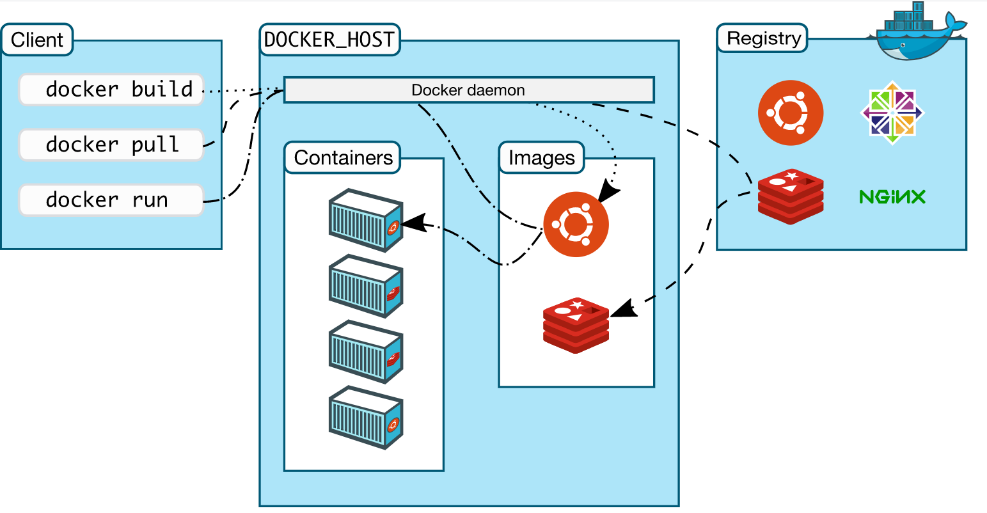


Figure 3: Docker architecture [<https://docs.docker.com/>]

**Docker Registries:** Docker images are placed into docker registries. “It works correspondingly to source code repositories where images can be pushed or pulled from a single source”[12]. Registries can be divided into 2 types Public and Private. “Docker Hub is called a public registry where everyone can pull available images and push their own images without creating an image from the scratch”[12]. But users can distribute the image to an area (Private or Public).

**Docker Containers:** Docker images help to create a docker container. “Containers hold the whole kit required for an application, so the application can be run in an isolated way”[12].

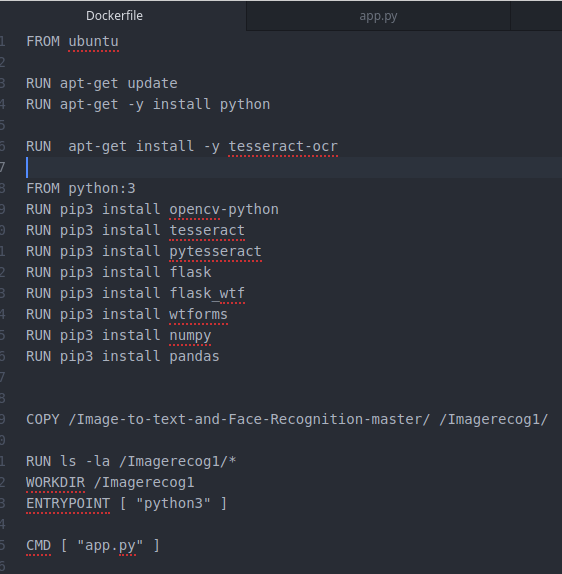
In this section, we will briefly describe how to containerize one sample application created based on a similar idea by [13]. The app is a simple app that converts text from an image to simple text and detects a face from the picture provided. The app is developed in Python Flask and utilizes Tesseract, Numpy, OpenCV, etc. libraries from Python.

**Steps:**

**To install docker:** Visit <https://docs.docker.com/get-docker/> and install as per your system specification.

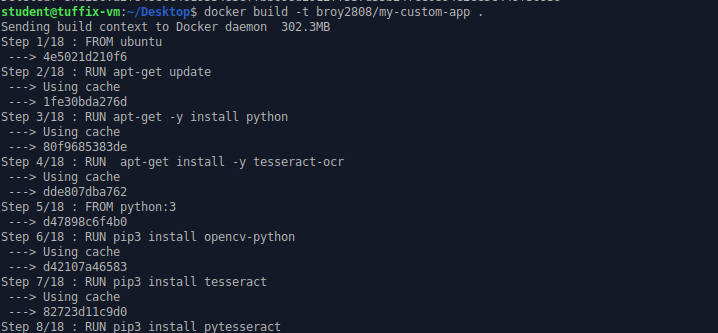
1) As a first step, I have created a docker file with execution commands to install an operating system, python, and its dependencies. This docker file will be used to create a docker image of the application. The image, in turn, helps us up and running our application within minutes on any platform.

Below is the Dockerfile with all steps to execute the project and install all the dependencies.



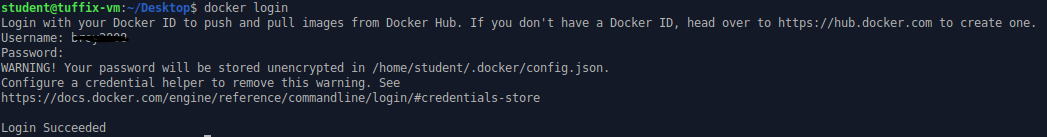
2) 2nd step is to build this Dockerfile with the following command:

* **docker build -t broy2808/my-custom-app .**



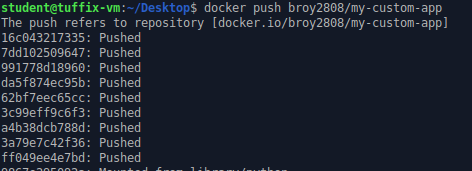
3)To upload the image into Dockerhub. As stated in the docker website “**Dockerhub**”- cloud-based repository, is the easiest way to manage, create, and deliver container applications. It has repositories and users can push and pull images accordingly.

**🡪 Docker login**



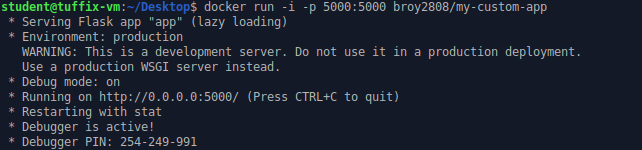
4)Now, you can push the created image into DockerHub with the below command.

**🡪 docker push broy2808/my-custom-app**

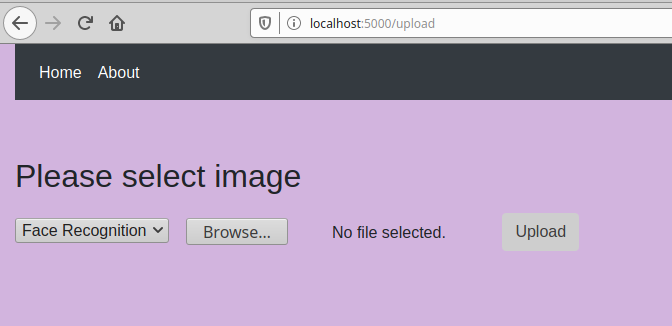


5) We can run the below command to check the image stored locally into docker in your system. Check the app is correctly up and running or not by the below command. Note: The command is specific to the Python flask application. We are using “-p 5000:5000” to mention the port where the application will run.

* **docker run -i -p 5000:5000 broy2808/my-custom-app**

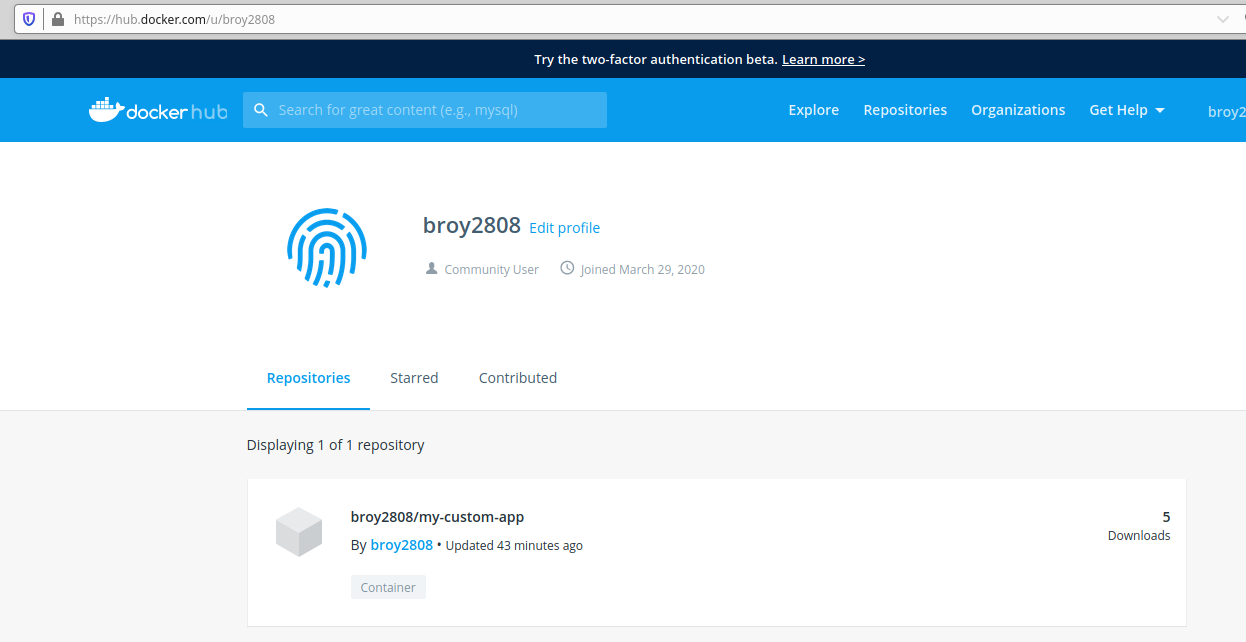


6)Up and running applications within minutes.

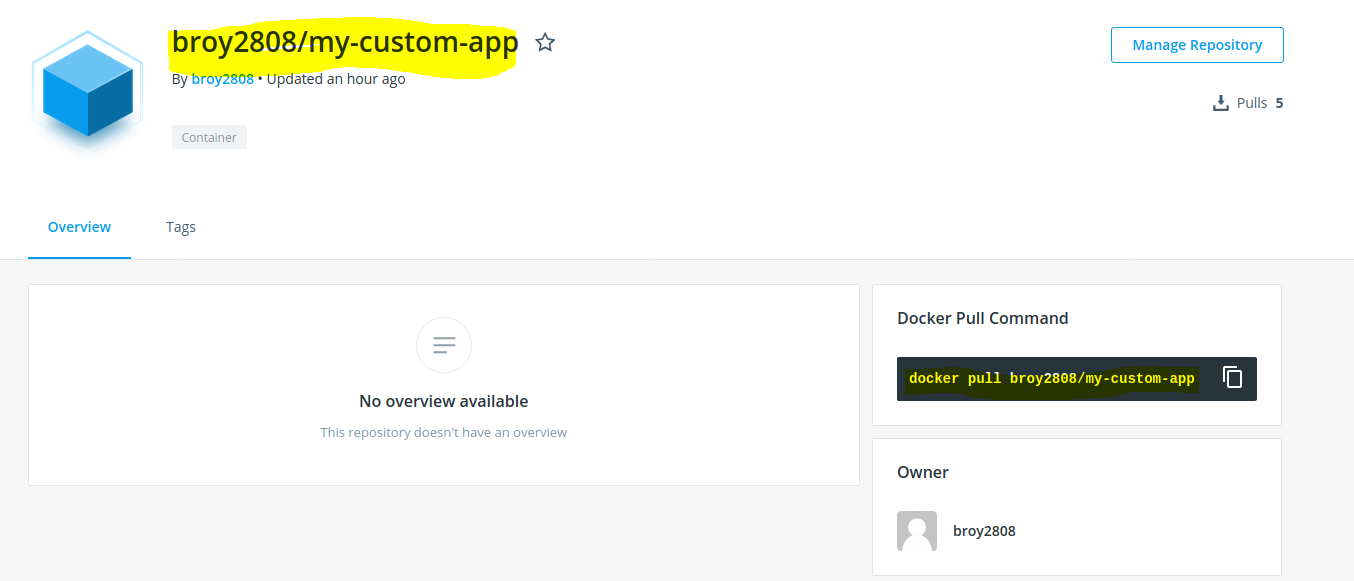


We can upload this image to Dockerhub. So, that it can be accessible to everyone.

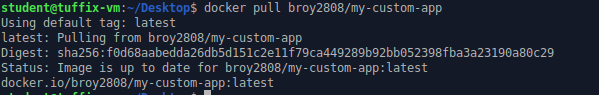
Below is a snapshot of Dockerhub.



The uploaded image can be found under repositories. The image is named “**broy2808/my-custom-app**”. Users can pull the image from the docker hub through the command “**docker pull broy2808/my-custom-app**”.



Running application by pulling image from Dockerhub:-



Run command “**docker images**” to check all the images in the system and you can validate the image you downloaded.



You can then simply use “**docker run -i -p 5000:5000 broy2808/my-custom-app**” to run the application. Look into step 5 for the details.

**Few other important commands->** Use additional commands as necessary with each of them.

1) To check all the running containers:-> **docker ps -a**

2) To delete a container🡪 **docker rm [container id]**

3) To delete an image🡪 **docker rmi [image name]**

Note: Before deleting an image you need to delete all the attached containers up and running based on that image. Otherwise use “**docker rmi [image name] - -force”** to delete the image forcefully.

4) To run an image🡪 **docker run [imagename].**

5) To Starts one or more stopped containers-> **docker start**

6) To Stops one or more running containers-> **docker stop**

7) Builds an image from a Docker file-> **docker build**

8) Pulls an image or a repository from a registry-> **docker pull**

9) Pushes an image or a repository to a registry-> **docker push**

10)  Searches the Docker Hub for images-> **docker search**

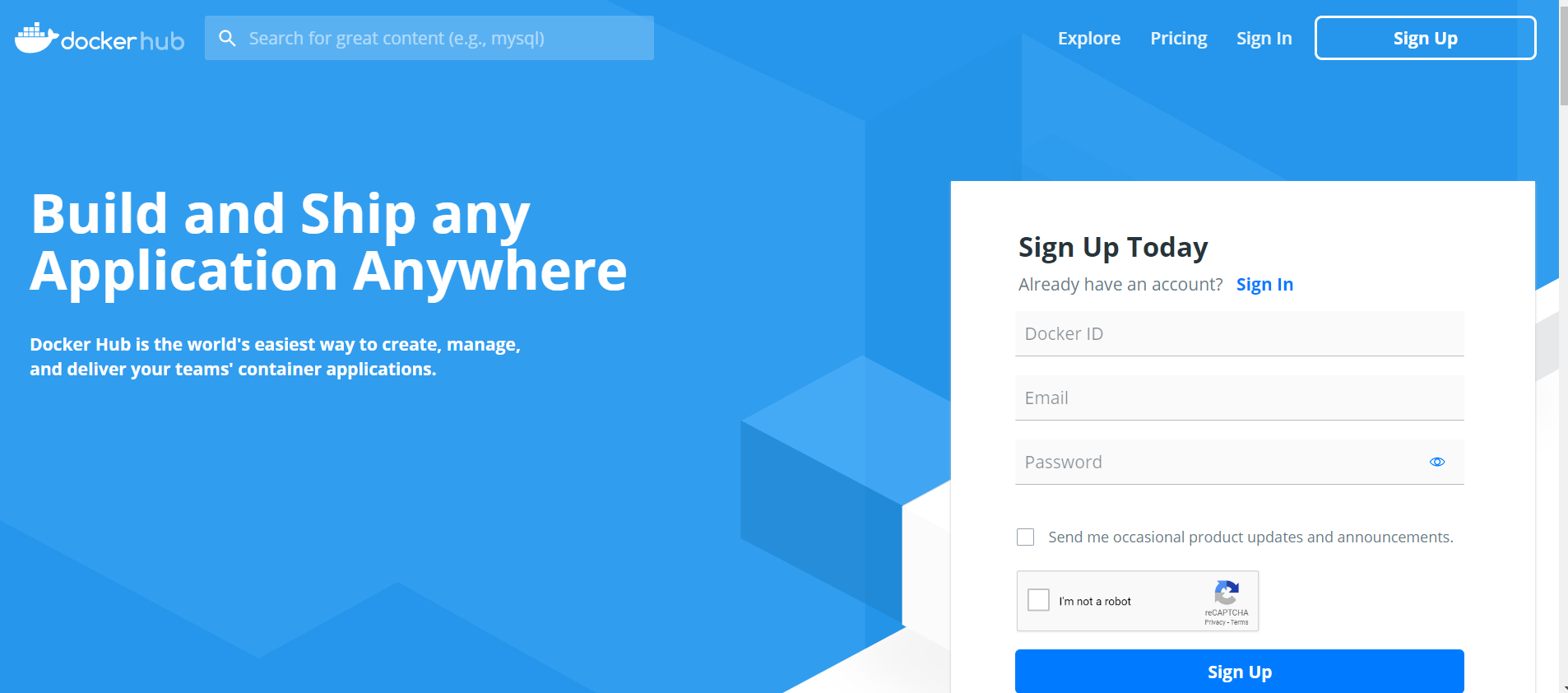
11) Attaches to a running container-> **docker attach**

12) Creates a new image from a container’s changes🡪 **docker commit**

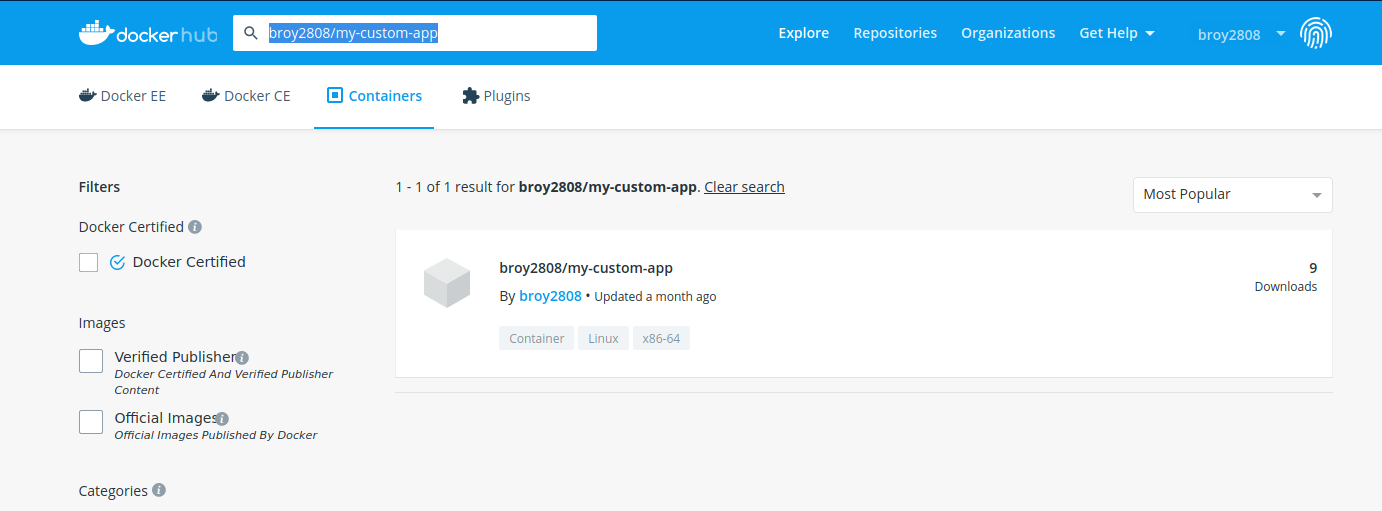
Please visit “<https://docs.docker.com/engine/reference/commandline/docker/>” to view all the commands and their uses.

**How to run my application by pulling from docker hub:**

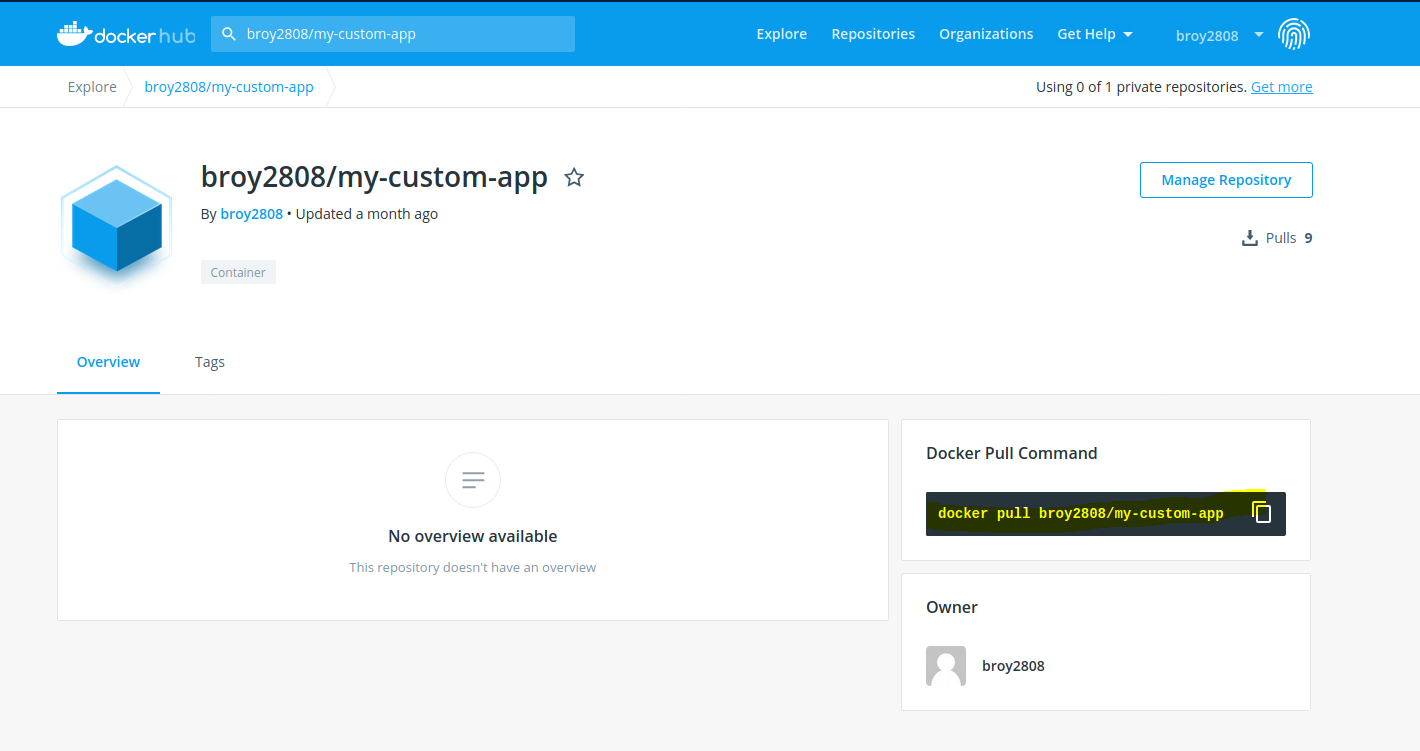
Logging and sign up into docker hub:



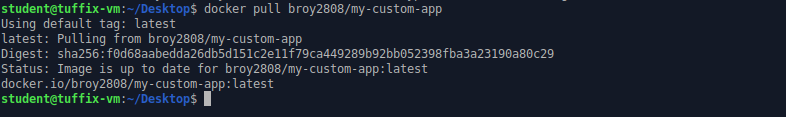
After login search image id **broy2808/my-custom-app.**



Once you open that you will see instructions about how to pull this image. Command is Highlighted in yellow.

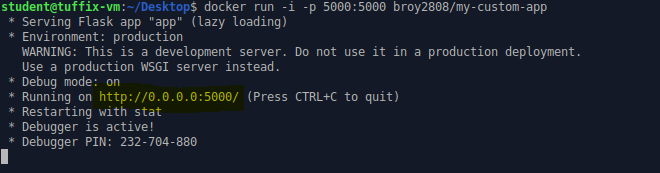


Running this image: copy the command **docker pull broy2808/my-custom-app** and run that in your computer in desired folder.

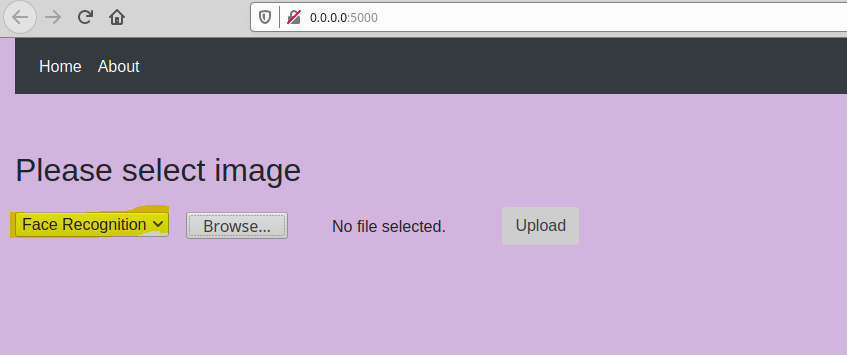


Now run command **docker run -i -p 5000:5000 broy2808/my-custom-app** to up and running the image:

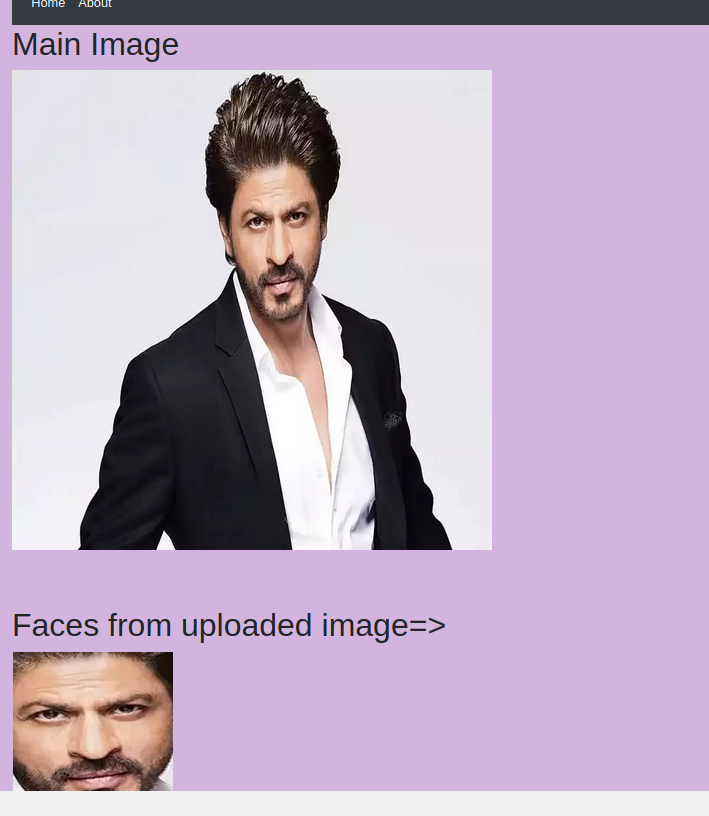
Note: 5000 is the port number here and flask default port number is 5000.



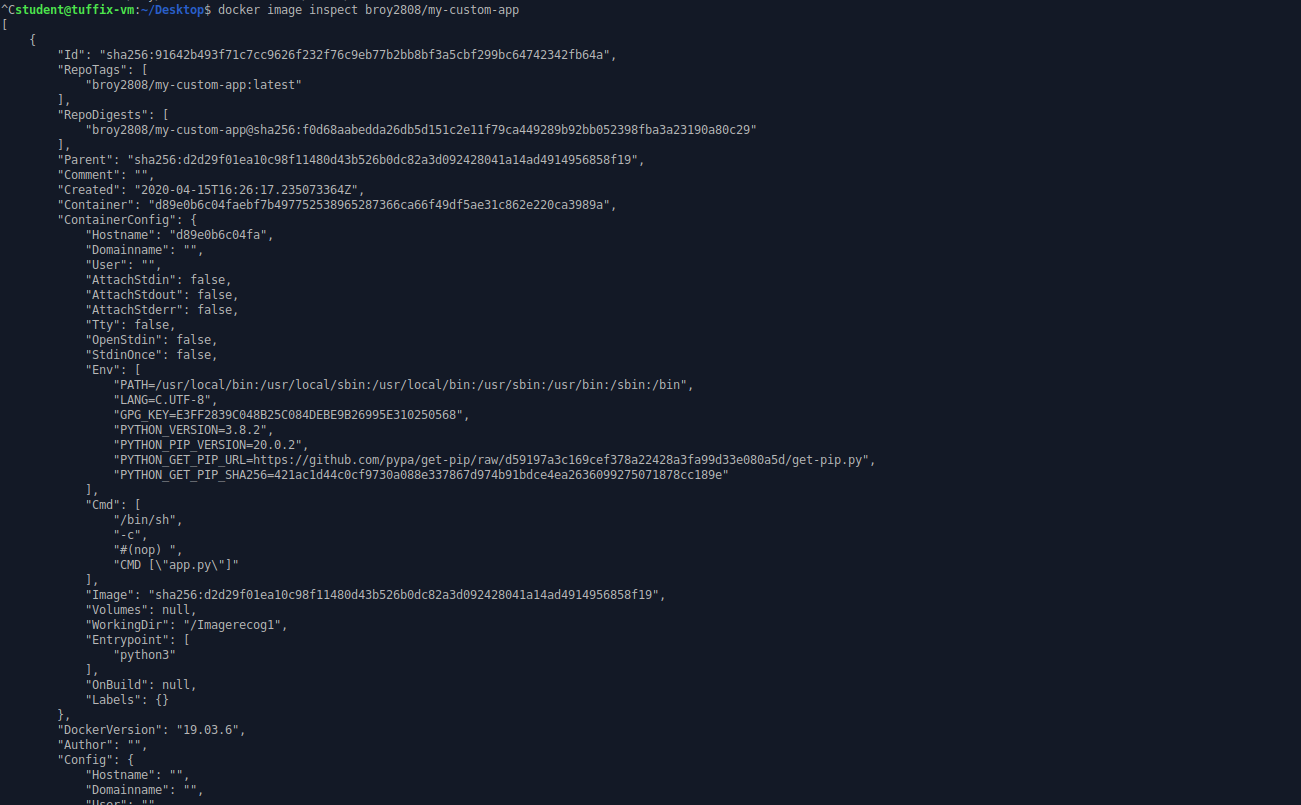
Our application is now up and running. Simply open the link highlighted in yellow and you can see our application will up and run in minutes like below.



Select face recognition and upload image. You can see face detected from uploaded image.



We can also use command **docker image inspect broy2808/my-custom-app** to see details about the specific image like below.



# **4.0 Conclusion**

Companies in today’s fast-moving world face challenges of upgrading their applications continuously, performance stability, and constantly evolving their security requirement. It is very important that they deliver their software to the customer fast enough otherwise their competitors will win the race. So, most of the enterprises focus on the rapid development of software features and operations stability. “Continuous integration and continuous delivery (CI/CD) is a practice that enables rapid software changes while maintaining system stability and security”[5]. Most of the cloud platforms now support DevOps operations. Microsoft Azure has Azure Boards, Azure pipeline, Azure Repos, Azure test plans, Azure artifacts, and Extension marketplace. Similarly, AWS has AWS CodePipeline, AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, AWS CodeStar, Amazon Elastic Container Service, AWS Lambda(serverless computing), etc. AWS CodePipeline is a CI/CD delivery service for reliable and fast infrastructure and application updates. AWS CodeBuild is a build service which runs tests, compiles code, and produces ready to deploy software packages. AWS CodeDeploy automates code deployment to any instance like Amazon EC2. We “can use AWS CodeStar to rapidly orchestrate an end-to-end software release workflow using these services. These are highly available, easily integrated services that can be accessed through the AWS Management Console, AWS APIs, and AWS SDKs like any other AWS service”[5].

“Container technology has a huge potential to substantially advance PaaS technology toward distributed heterogeneous clouds through lightweightness and interoperability”[12]. Several researchers have pointed out the advantages of docker like speed, portability, scalability, density, and rapid delivery. Although researchers have also pointed out concerns like complete virtualization is not provided, network, and data management aspects. But DevOps, in general, is a very promising practice, with all the cloud platforms now supporting it, it will surely grow. The future scope of this research would be utilizing cloud platforms for DevOps and supporting the scientific community.

# **References:**

1) Pérez, A., Moltó, G., Caballer, M., & Calatrava, A. (2018). Serverless computing for container-based architectures. *Future Generation Computer Systems*, *83*, 50–59. doi: 10.1016/j.future.2018.01.022

2) Wan, X., Guan, X., Wang, T., Bai, G., & Choi, B.-Y. (2018). Application deployment using Microservice and Docker containers: Framework and optimization. *Journal of Network and Computer Applications*, *119*, 97–109. doi: 10.1016/j.jnca.2018.07.003

3) Pahl, C. (2015). Containerization and the PaaS Cloud. *IEEE Cloud Computing*, *2*(3), 24–31. doi: 10.1109/mcc.2015.51

4) Willis, J. (2015). Docker and the Three Ways of DevOps. Retrieved from https://goto.docker.com/rs/929-FJL-178/images/20150731-wp\_docker-3-ways-devops.pdf

5) AWS (2017). Practicing Continuous Integration and Continuous Delivery on AWS. Retrieved from https://d1.awsstatic.com/whitepapers/DevOps/practicing-continuous-integration-continuous-delivery-on-AWS.pdf

6) Marathe, N., Gandhi, A., & Shah, J. M. (2019). Docker Swarm and Kubernetes in Cloud Computing Environment. *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. doi: 10.1109/icoei.2019.8862654

7) Ejsmont, A. (2015). *Web scalability for startup engineers: tips & techniques for scaling your Web application*. New York: McGraw-Hill Education.

8) Jadeja, Y., & Modi, K. (2012). Cloud computing - concepts, architecture and challenges. 2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET). doi: 10.1109/icceet.2012.6203873

9) Sakr, S., Liu, A., Batista, D. M., & Alomari, M. (2011). A Survey of Large Scale Data Management Approaches in Cloud Environments. IEEE Communications Surveys & Tutorials, 13(3), 311–336. doi: 10.1109/surv.2011.032211.00087

10) AWS (2020). Overview of Amazon Web Services. Retrieved from https://d1.awsstatic.com/whitepapers/aws-overview.pdf

11) <https://www.plutora.com/ci-cd-tools>

12) Rad, B. B., Ahmadi, M., & Bhatti, H. J. (2017). An Introduction to Docker and Analysis of its Performance. Retrieved from https://www.researchgate.net/publication/318816158\_An\_Introduction\_to\_Docker\_and\_Analysis\_of\_its\_Performance

13)  Alexander Chiu(2017). [HandWriting-OCR-CNN-WebApp](https://github.com/AlChiu/HandWriting-OCR-CNN-WebApp).Retrieved from <https://github.com/AlChiu/HandWriting-OCR-CNN-WebApp>