**Docker on Cloud**

**Things to take from this Blog:**

Docker Image with Automated Built

Container Configuration on Cloud with Azure, Google and IBM Bluemix

Docker on Cloud Containers

Jupyter running on Cloud Containers using Docker

1. **Introduction**

**What is a Containerized Application?**

Containers provide a streamlined, easy to deploy and secure method of implementing specific infrastructure requirements.

A containerized application is an operating system level virtualization method for deploying and running distributed applications without launching an entire virtual machine (VM) for each app. Instead, multiple isolated systems are running on a single control host and access a single kernel.

**What is Docker?**

Docker is an [open-source](https://en.wikipedia.org/wiki/Open-source) project that automates the deployment of [applications](https://en.wikipedia.org/wiki/Application_software) inside [software containers](https://en.wikipedia.org/wiki/Software_container). Its containers wrap up a piece of software in a complete file system that contains everything it needs to run: code, runtime, system tools, system libraries, etc. Docker provides an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux. Docker enables you to quickly, reliably, and consistently deploy applications regardless of environment.

**Why do we need Docker on Cloud?**

*“Alone Docker is not always the answer”*

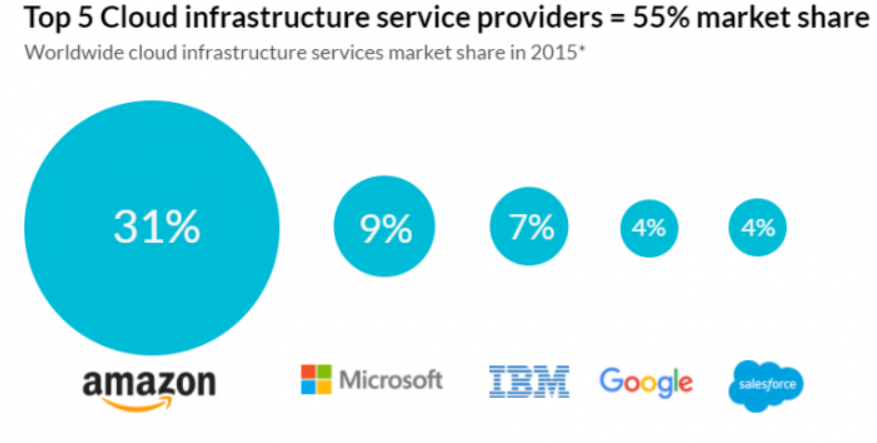
Containers are the answer in some situations while VMs are the answer in others — and, in some cases, both should be deployed at the same time.

Moving containers apps around isn’t very efficient without a need of a management platform that will automatically spin containers up, suspend them or shut down when needed. Also, control how they access resources like the network and data storage.

That’s where orchestration platforms come in. They provide this piece of the container puzzle. Few of orchestration platform are: Kubernetes, Swarm, Mesos and Kontena.

Cloud services provide faster orchestration i.e. automated arrangement, coordination, and management of complex computer systems, and services for production environments. Dockerized applications can be seamlessly moved from local development machines to production deployments using Cloud Services.

Running Docker on various cloud services like Azure, Bluemix, Google Cloud provides a highly reliable, low-cost way to quickly build, run, test, and deploy distributed applications at any scale.



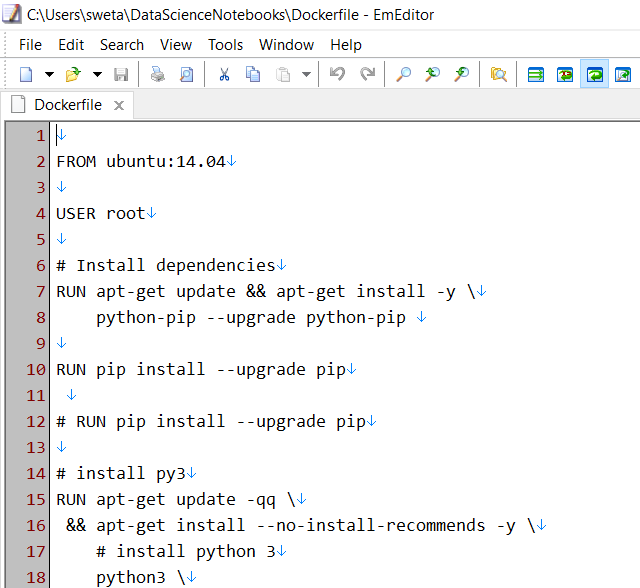
1. **Steps to get started with Docker and Cloud**
2. Build a docker image for using Jupyter notebooks with the help of Dockerfile
3. Run the container and check if the image is working
4. Create a Docker Hub account and publish the image on docker hub
5. Link the image between Docker and GitHub for automated builds on changes in the Dockerfile
6. Create Account on a Cloud Platform
7. Pull your image on Cloud
8. Run Jupyter notebook from Cloud

Step 1 to Step 3 are optional, If you already have an image please go to the Step 4 to get a detailed view of Configuration on Cloud

**Step 1:** **Building Docker Image**

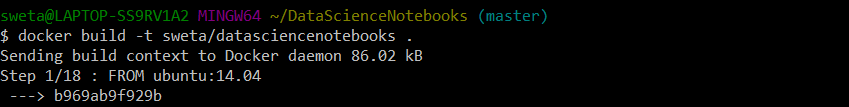
* 1. To build a Docker image we first need to [install Docker](https://docs.docker.com/engine/getstarted/step_one/) on your machine.

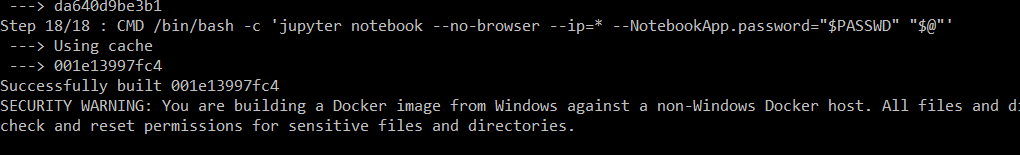
Create a [Dockerfile](https://github.com/bajajsweta/DataScienceNotebooks/blob/master/Dockerfile) with your required applications to be built in an Image



* 1. Build the Image on Docker and it should show the success message.

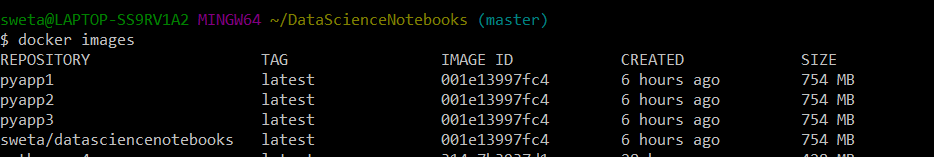
docker build -t sweta/datasciencenotebooks .





1.3. Check whether the image is present in Docker

docker images

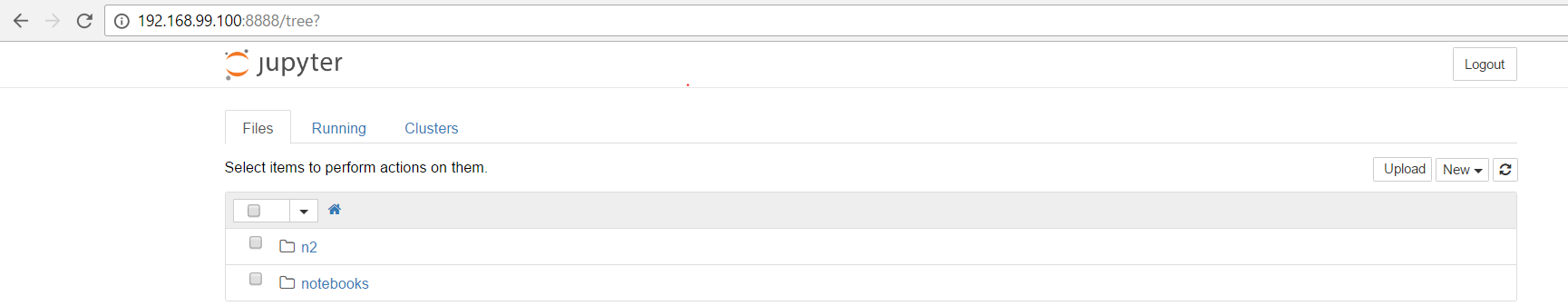


**Step 2:** **Run the container and check if the image is working**

2.1. Run Docker Container to test this image

docker run -d -p=8888:8888 -v=/srv/notebooks:/srv/n2 sweta/datasciencenotebooks

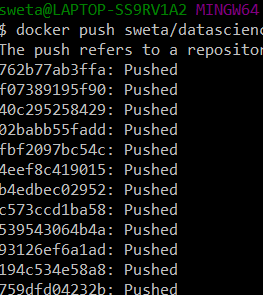
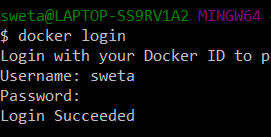




Local Machine with Jupyter Notebook

**Step 3: Run the container and check if the image is working**

* 1. Login to Docker and push the image to be accessed publicly



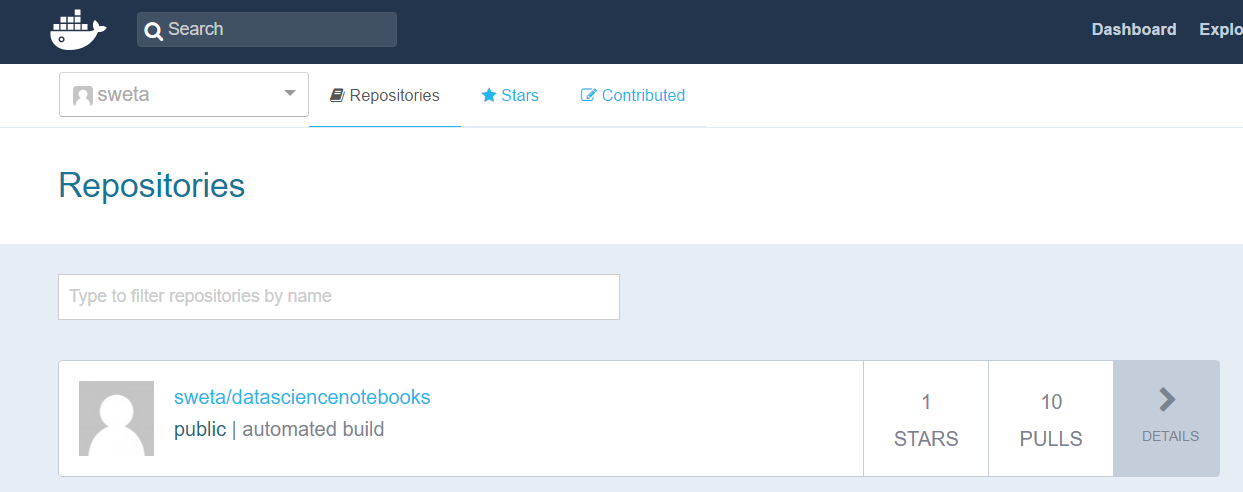
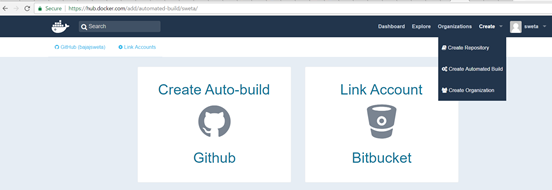


Figure: Shows the published image on Docker Hub

**Step 4: Link the image between Docker and GitHub for automated builds on changes in the Dockerfile**

****

* 1. Link Docker Hub Image and GitHub for automated builds

<https://github.com/bajajsweta/DataScienceNotebooks>

<https://hub.docker.com/r/sweta/datasciencenotebooks/>

**Step 5: Create Account on a Cloud Platform**

* 1. Create account on cloud and deploy the image on cloud

Now, that we know that the image is working with the required Jupyter notebook,

We will go ahead and create the account with and deploy our application

1. Microsoft Azure
2. Google Cloud
3. IBM Bluemix
4. **Detailed Configuration with the three Cloud Platforms**

**Docker on Microsoft Azure**

In this document, we have used Microsoft Azure Container Services to connect to docker

Advantages of using Container Services:

1. Create an optimized container hosting solution

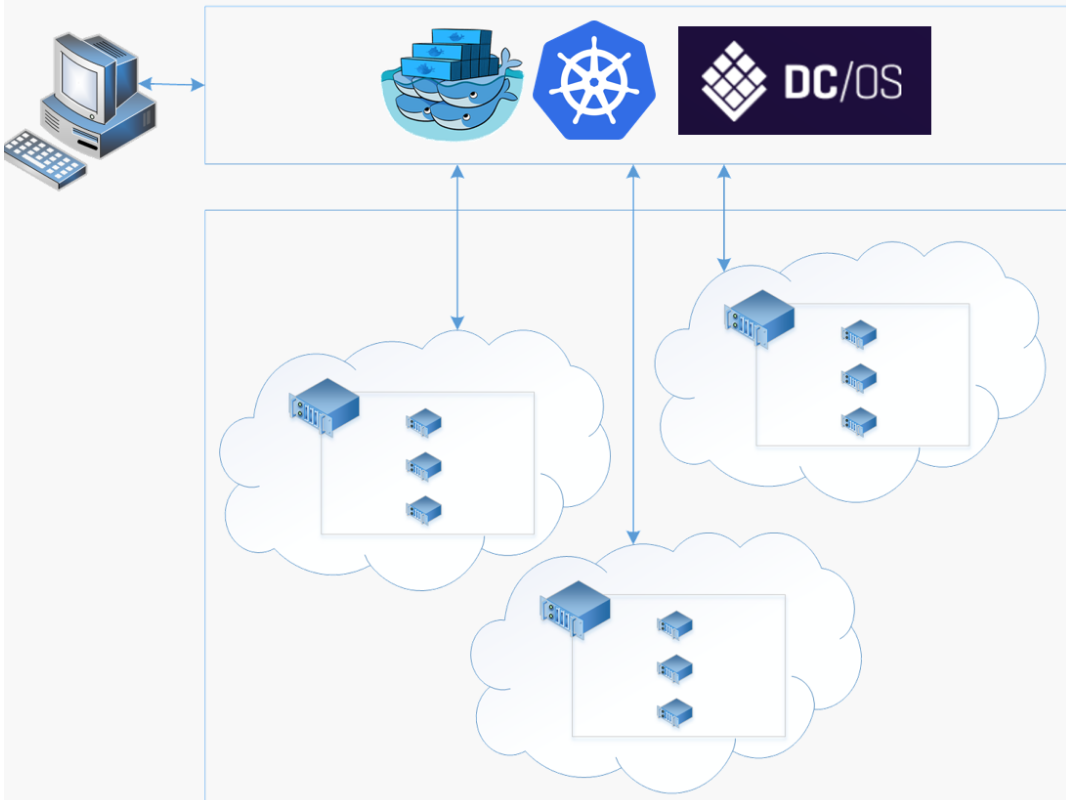
Azure Container Service optimizes the configuration of popular open-source tools and technologies specifically for Azure. You get an open solution that offers portability for both your containers and your application configuration. You select the size, number of hosts, and choice of orchestrator tools—Container Service handles everything else.

1. Manage container applications using familiar tools
2. Scale Orchestrate using DC/OS, Docker Swarm, or Kubernetes

Choose the tools and solution that best suit your needs for Docker container orchestration and scale operations. Use the Mesos-based DC/OS, Kubernetes, or use Docker Swarm and Compose for a pure Docker experience.

1. Migrate container workloads to and from Azure

Azure Container Service only uses open-source components in the orchestration layers to give you portability of full applications—not only individual containers—to migrate seamlessly to and from Azure at will.



**Steps to Get Connected to Windows Azure:**

1. If you are using a windows machine, first install puttygen and putty to generate a private and public key.

Download Link: <http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html?>

1. Use Puttygen to generate the private key in the .ppk format and the public key

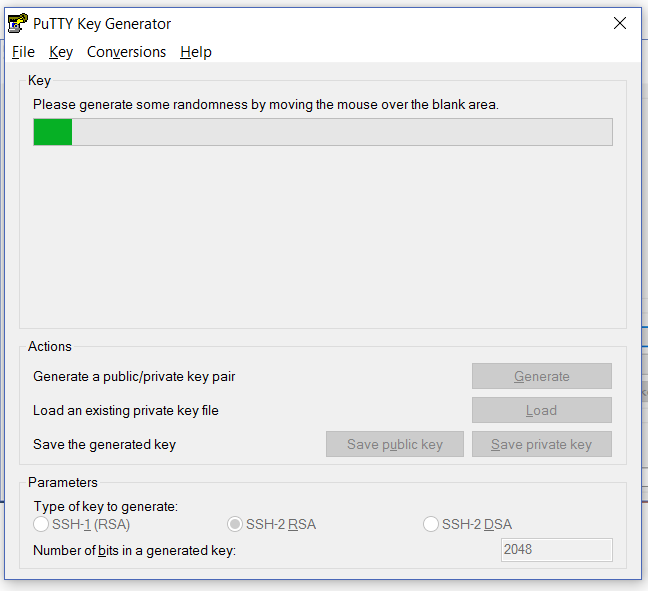


Figure: Shows the Puttygen screen to Generate the keys

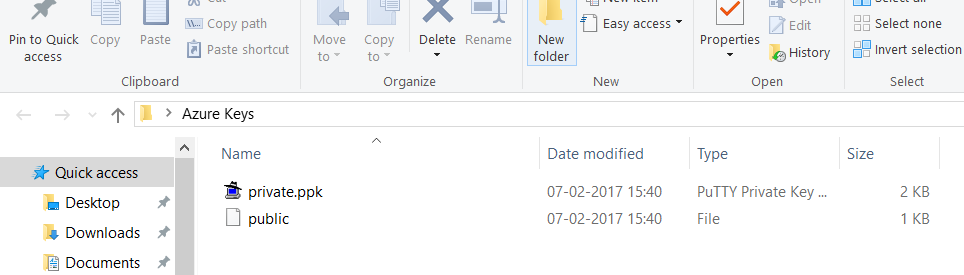
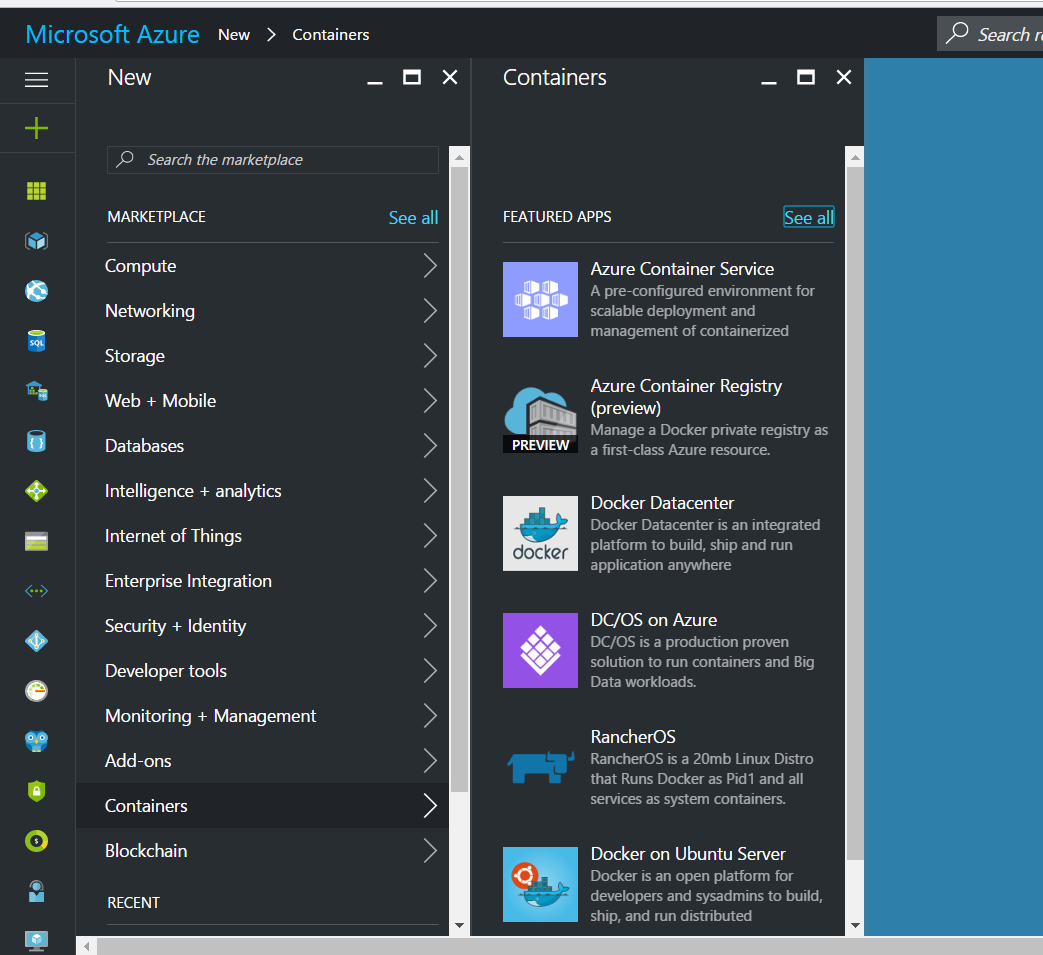
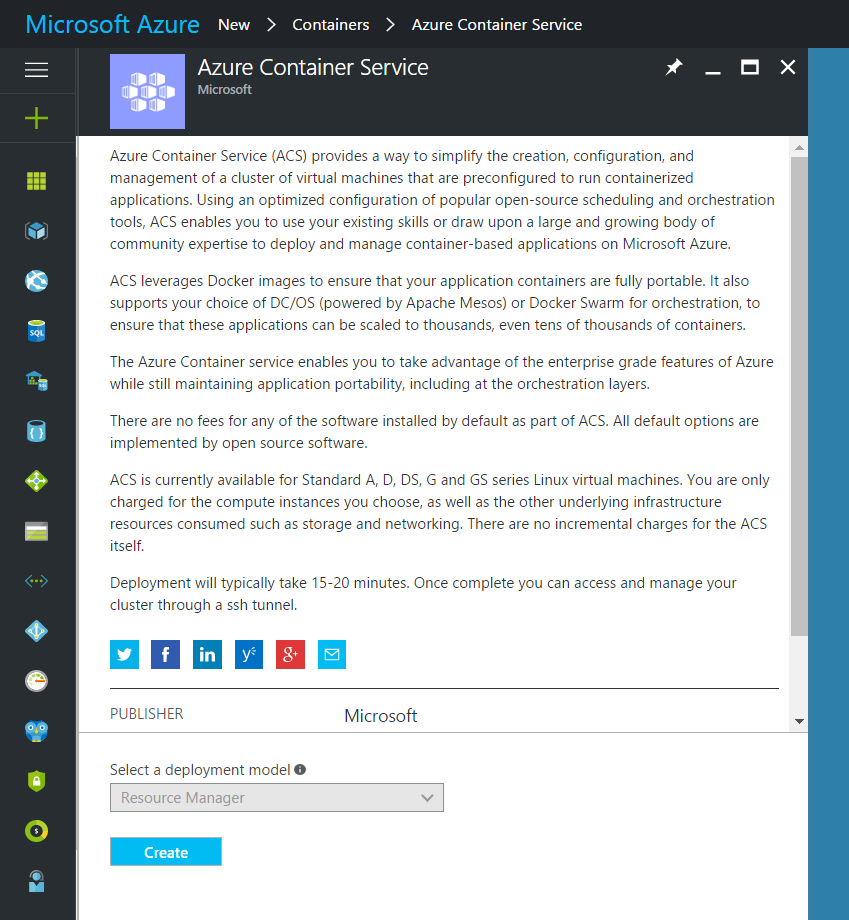
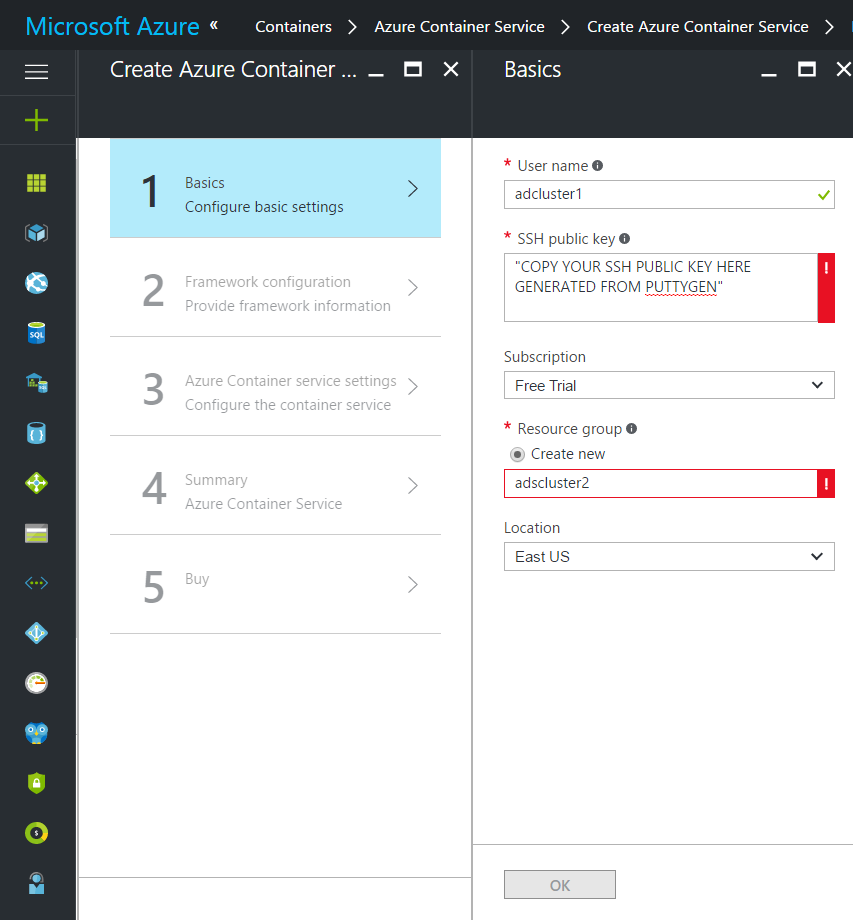


Figure: Shows that the Private and Public Key need to be stored

1.  Login to Windows Azure and you should create an account there. After creating an account, you should be able to navigate to the create new container. Azure Container Service (ACS) provides a way to simplify the creation, configuration, and management of a cluster of virtual machines that are preconfigured to run containerized application.

Agree to create an Azure Container Service

1. Enter your own Usernames, SSH Key and Resource Group Name



1. The Next Step is selecting the ‘Orchestration Configuration’

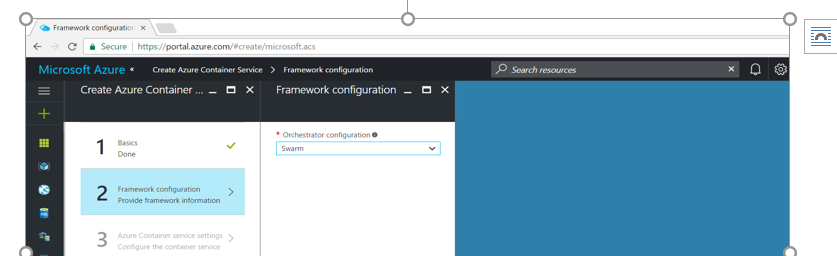
**What is Orchestration?**

Orchestration describes automated arrangement, coordination, and management of complex computer systems, and services. It is often discussed as having an inherent intelligence or even implicitly autonomic control, but those are largely aspirations or analogies rather than technical descriptions

There are various orchestrators like Kubernetes, Swarm and DC/OS but we will choose ***SWARM*** for our configuration.

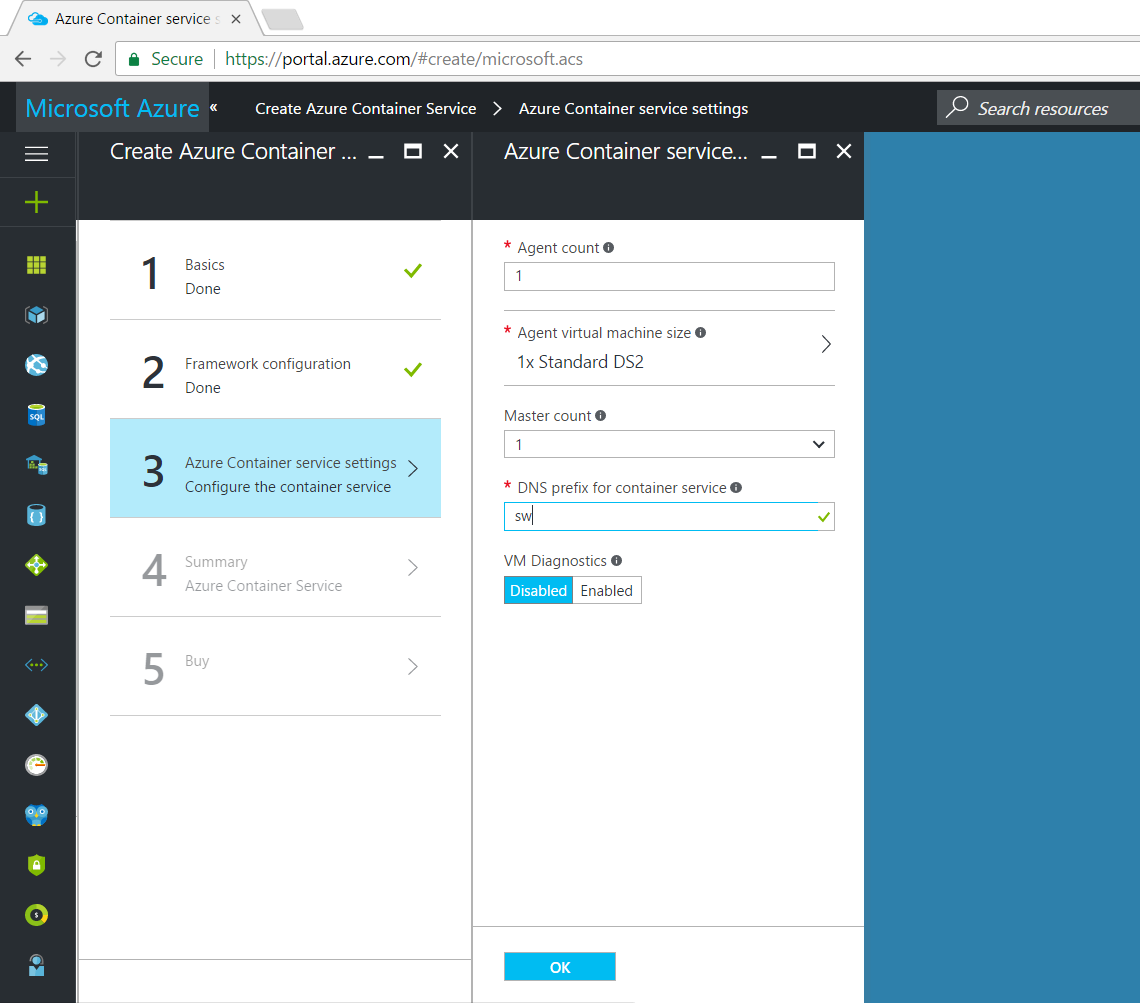
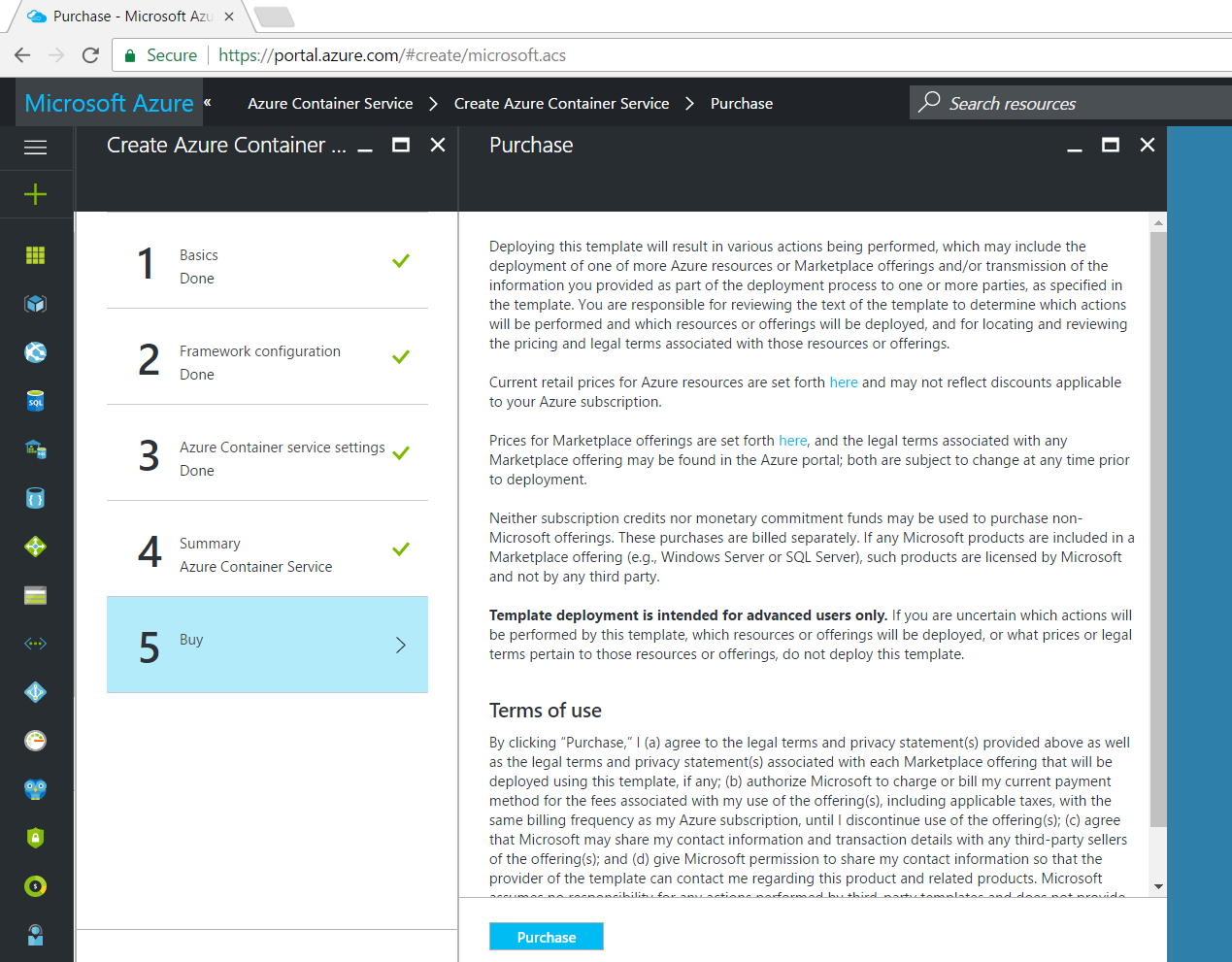
Docker Swarm is a native clustering tool for Docker which turns multiple Docker engines into a cluster and makes that a cluster by making it look like a single Docker engine. So, when we have an application with many virtual machines wanting to work as one engine deploying a docker swarm on azure swarm would be easier.

For now, we will just look at the configuration of azure container interactions with docker



1. Here, you can select your configuration, you could also enter the number of virtual machine you want to create and of what configuration.

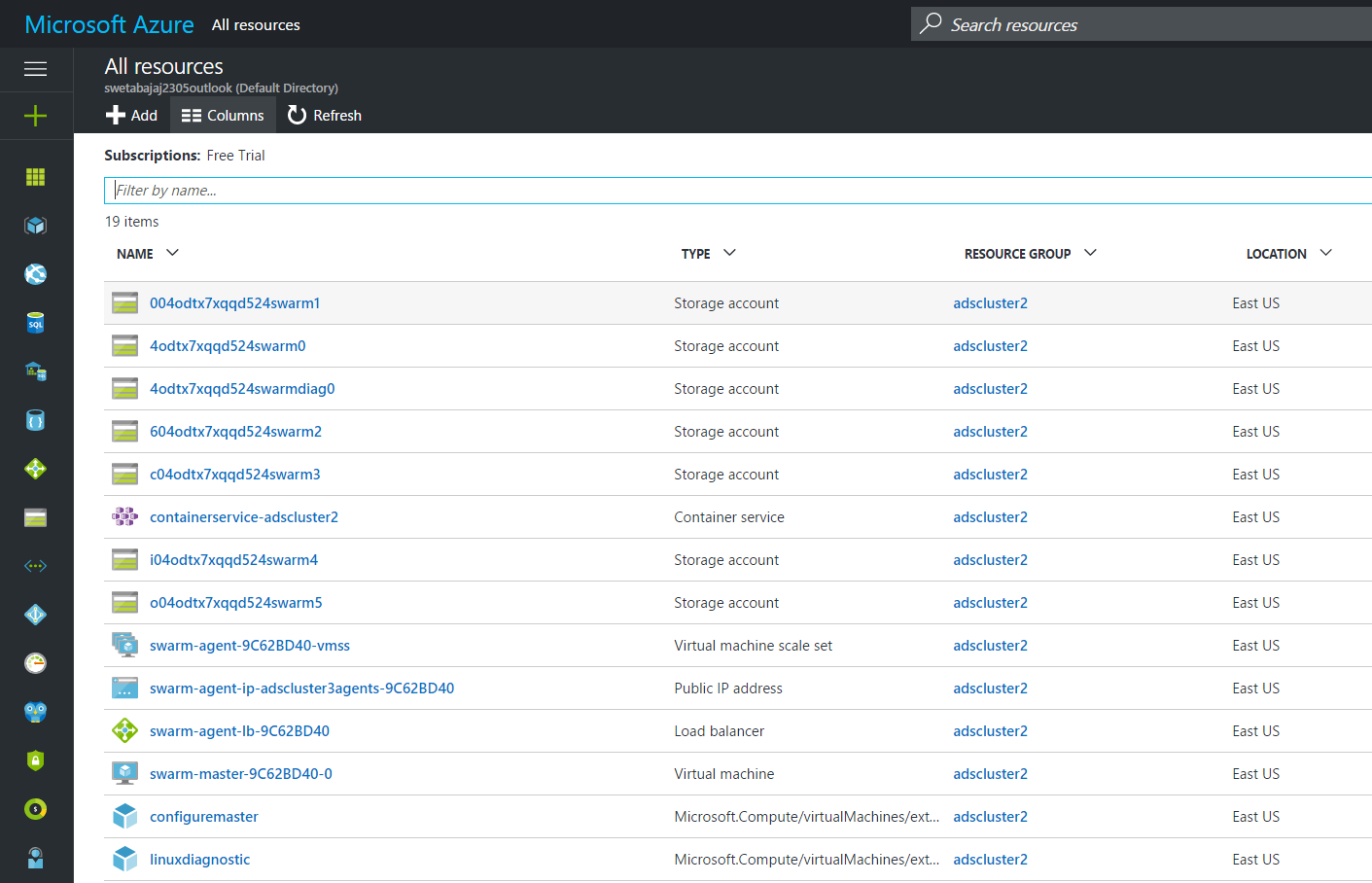
adscluster3

Purchase this to ahead with your configuration

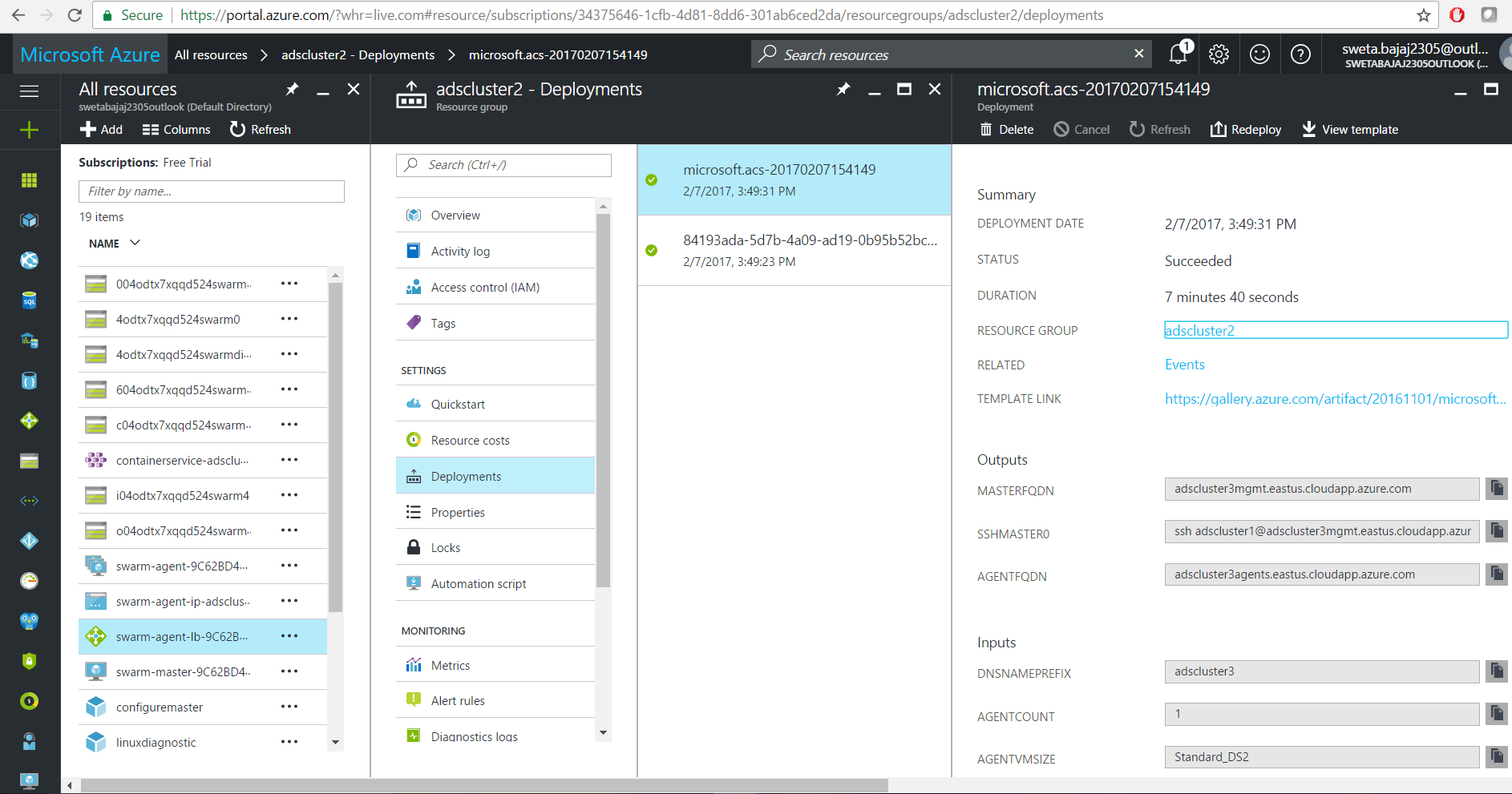
1. The application may take some time to Deploy but you should be able to see your containers ready on the dashboard



1. Now, go to ‘All resources’ and check your resource group



1. Check your deployments and copy the ‘master DNS’ and ‘Agent DNS’.

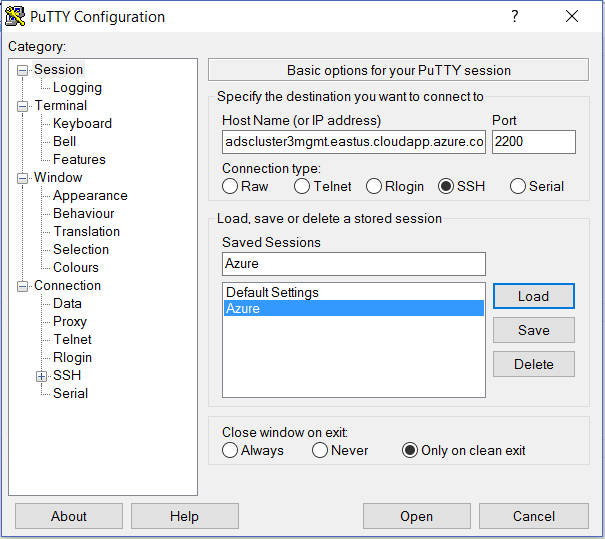


Agent DNS

Master DNS

1. You will need the master DNS to connect to the console through putty and Agent DNS to connect to your client application
2. To Connect to putty, you will need your ‘Master Load Balancer’ i.e the ‘Master DNS’, Port Number: 2200

Save Session Name: Azure



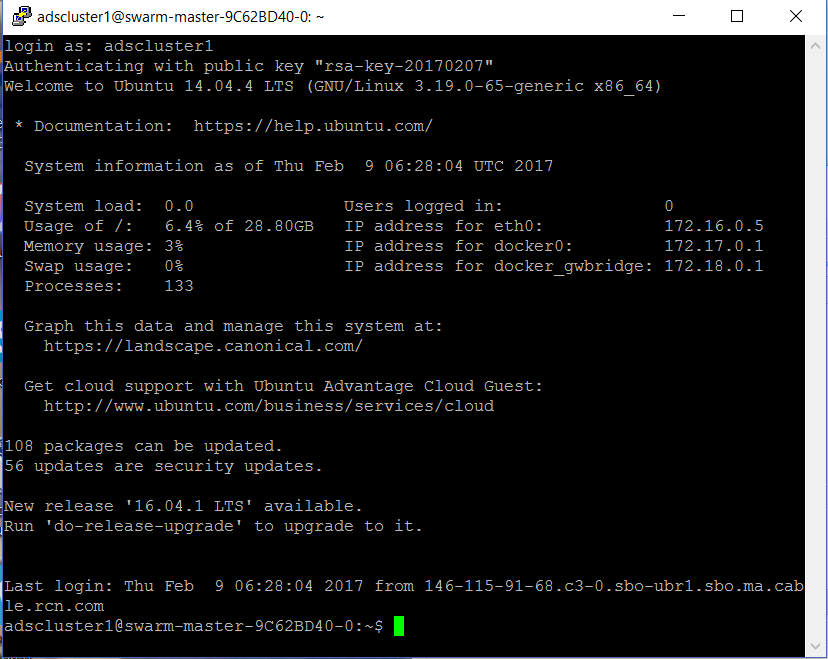
Navigate to SSH Tab and load private key



1. Navigate to Tunnels and Add the Port Number and destination. This basically creates a Tunnel to connect to your cloud machine

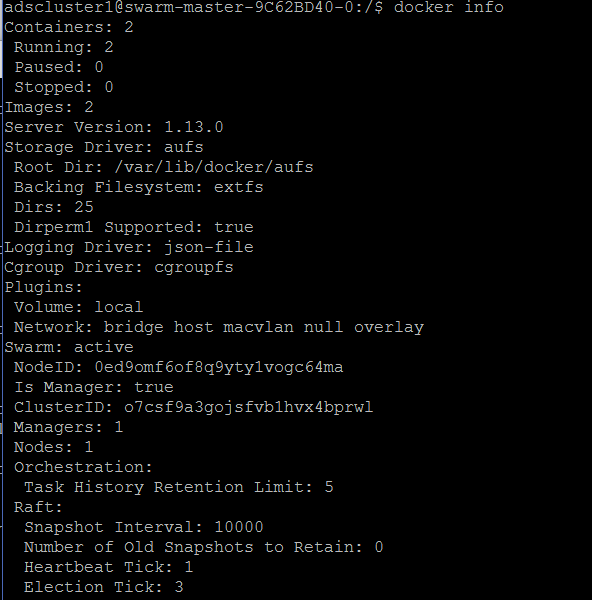


1. After opening this connection that you have just saved, you should be able to connect to the cloud console i.e the ***Azure Container***

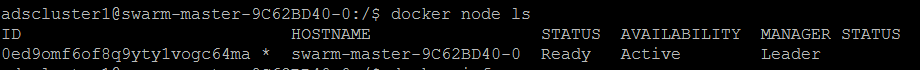


13.1 Once you have logged in to the Azure Container you can run docker commands on swarm and check the activities and Swarm status in the Azure Container

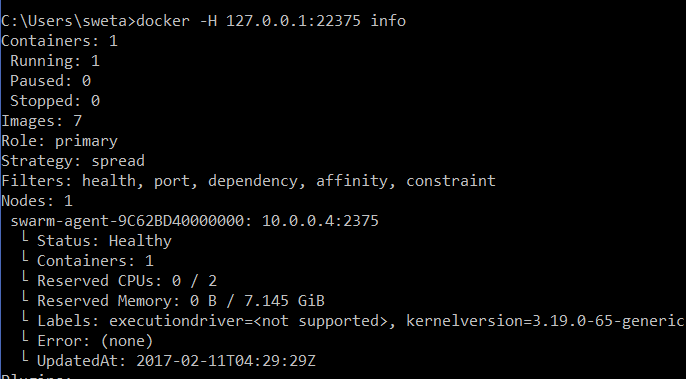
docker info



13.2 Check the active node docker node ls



1. Tunnel the Docker socket over SSH to remotely run commands on the cluster.



Now that your cloud container is up you can connect to your local machine to interact with docker. If you should build an image that is present on your local machine, you can connect to the local console and build an image through it.

docker -H 127.0.0.1:22375 ps -a

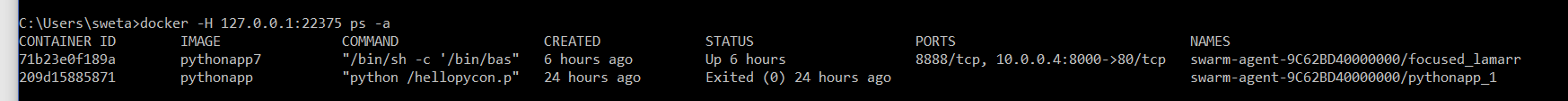
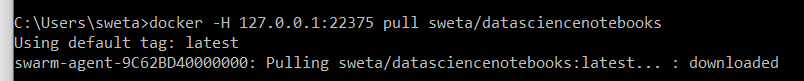


Figure: Shows the containers that are present on the

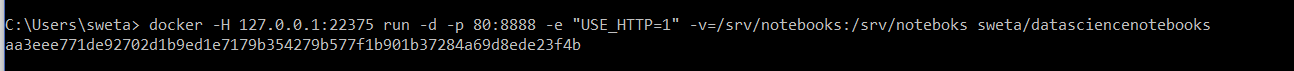
1. Pull the Docker Image on cloud

docker -H 127.0.0.1:22375 pull sweta/datasciencenotebooks



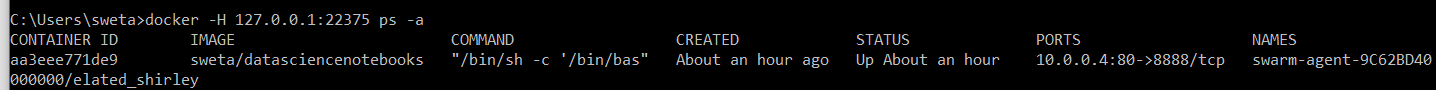
1. Run the container

docker -H 127.0.0.1:22375 run -d -p 80:8888 -e "USE\_HTTP=1" -v=/srv/notebooks:/srv/noteboks sweta/datasciencenotebooks

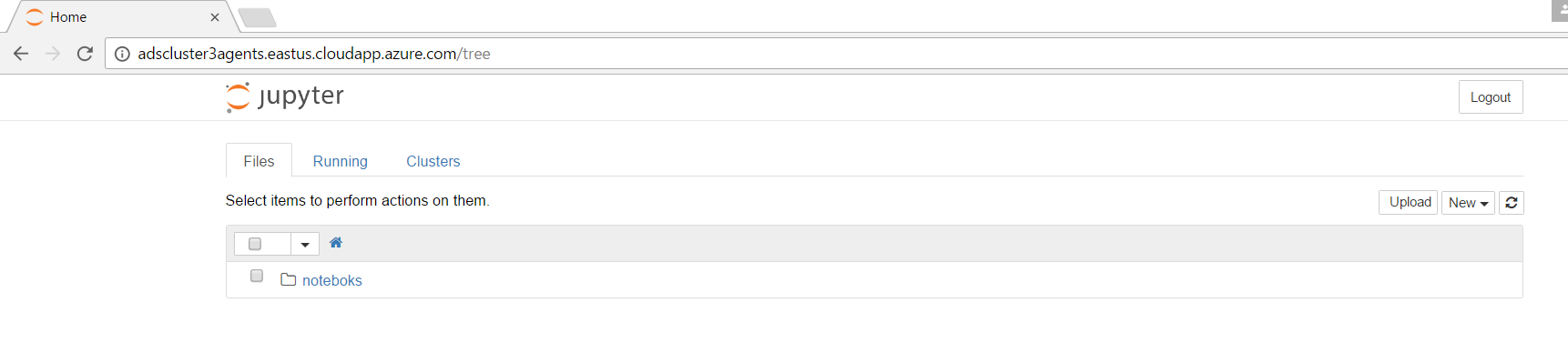


Check if the container is running

docker -H 127.0.0.1:22375 ps -a



1. Now, to see if the Jupyter notebook is working copy the ‘Agent DNS’ mentioned in Step 10 and use the port 80



Azure Cloud with Jupyter Notebook

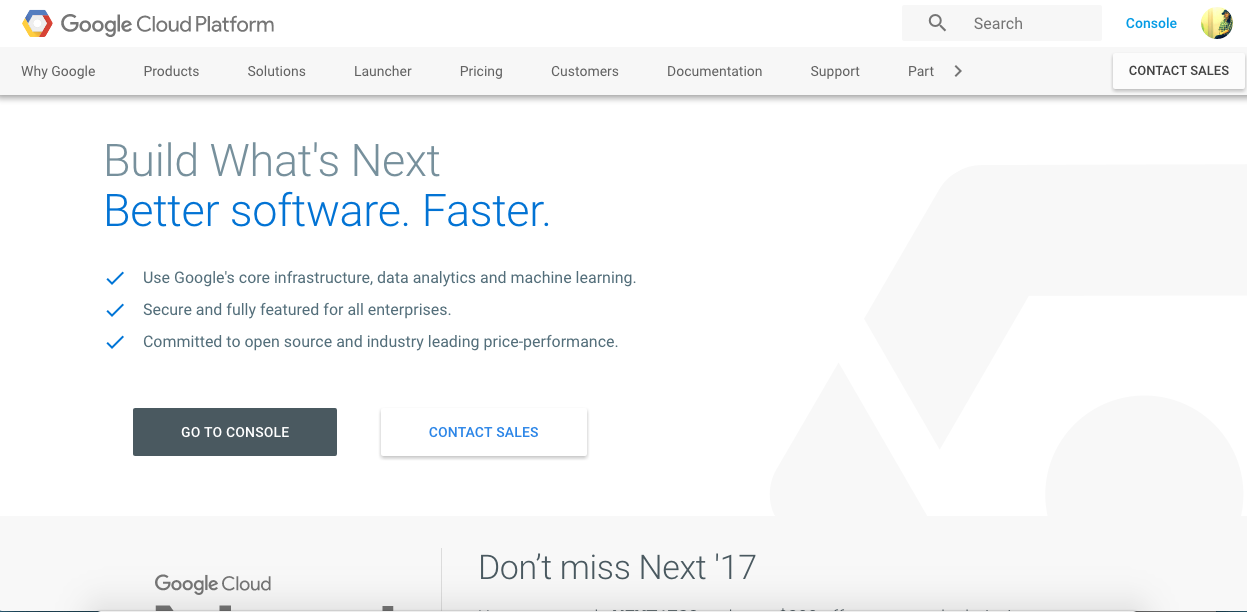
1. **Docker on Google Cloud Platform**

Google Cloud Platform consists of a set of physical assets, such as computers and hard drives, and virtual resources, such as virtual machines(VMs), that are contained in Google’s data centers around the globe. It is Secure, global , high-performance, cost-effective.

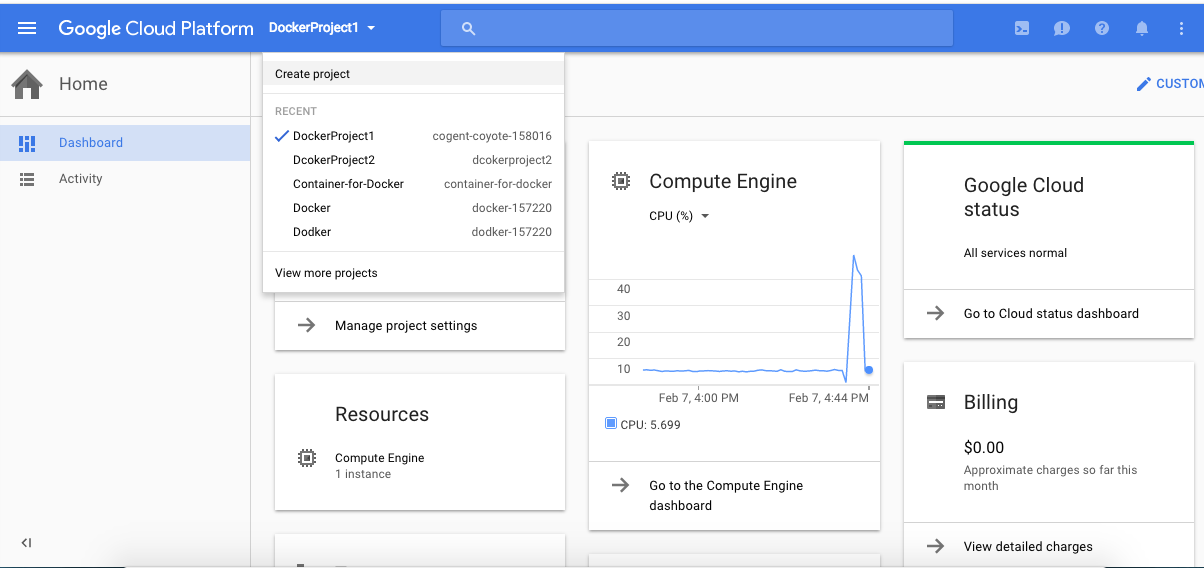
Google Container Engine is a powerful cluster manager and orchestration system for running you Docker Containers. Container Engine schedules your containers into cluster and manages them automatically based on the requirements such as CPU and memory, number of replicas. The cluster can be set up in minutes. Google Compute Engine is based on Virtual Machine and Container Engine is based on Docker and its VM is a container-optimized OS image that includes Docker and Kubernetes Kubelet to manage containers.

**Steps to configure Docker on Google Cloud Platform:**

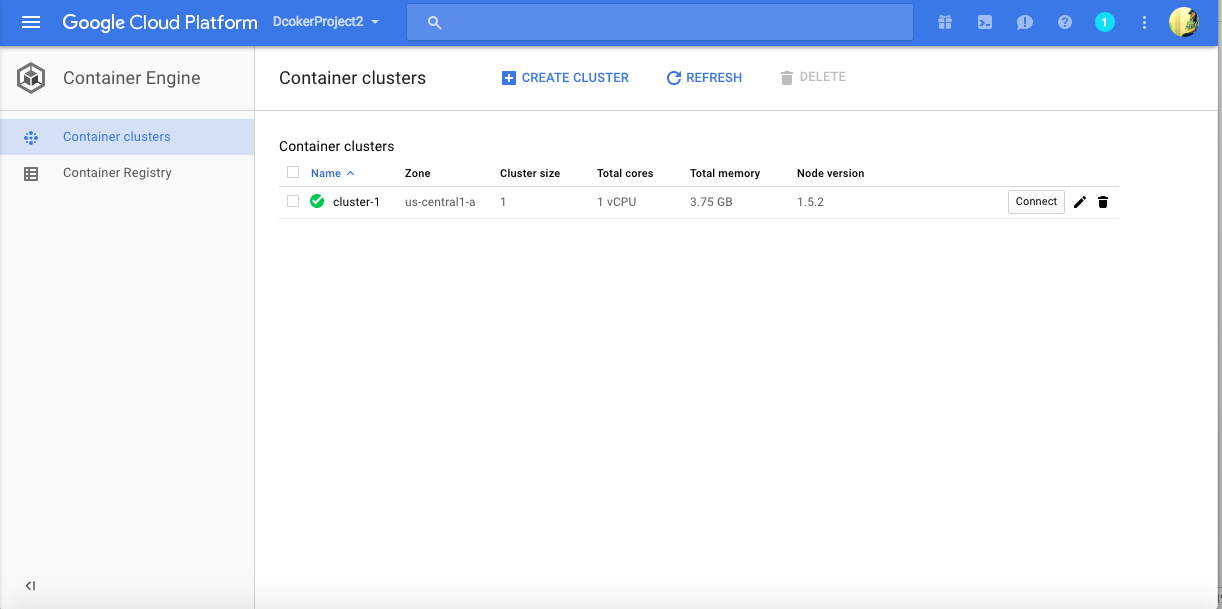
1. Create a free trial account on google cloud platform and click on GO TO CONSOLE.



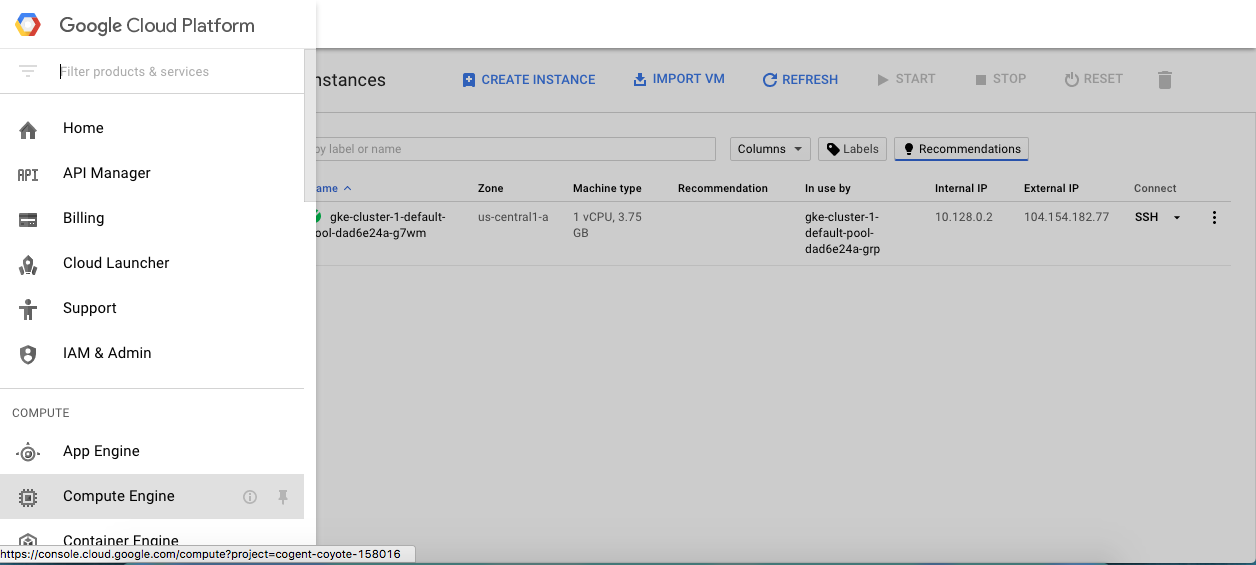
1. Click on create project to create a new project.



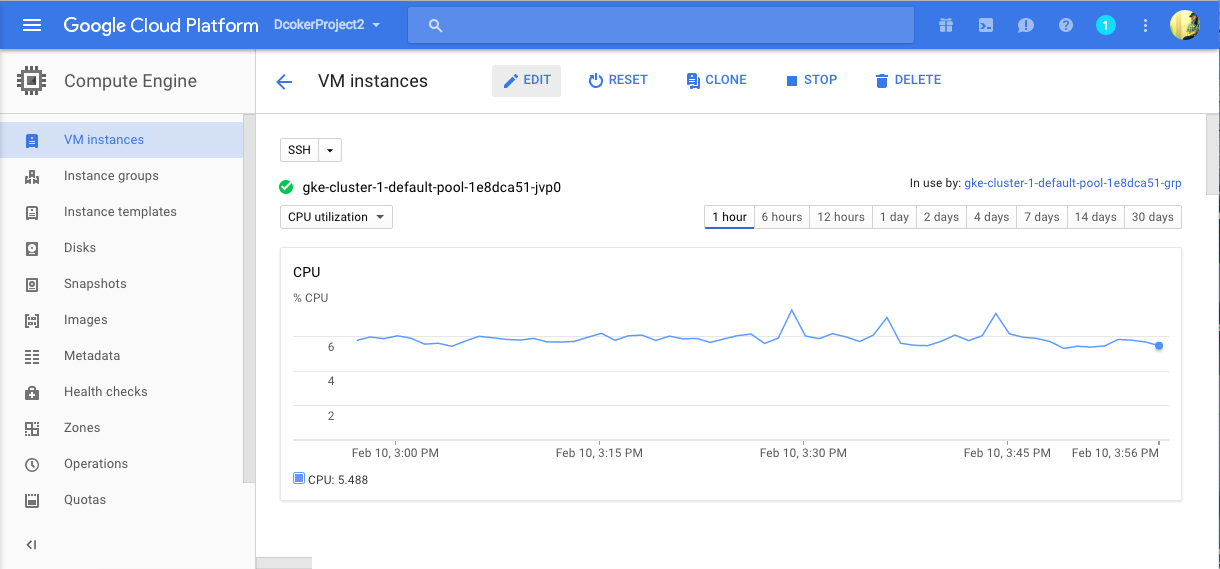
1. Click on product and services and select Container Engine and create a cluster.



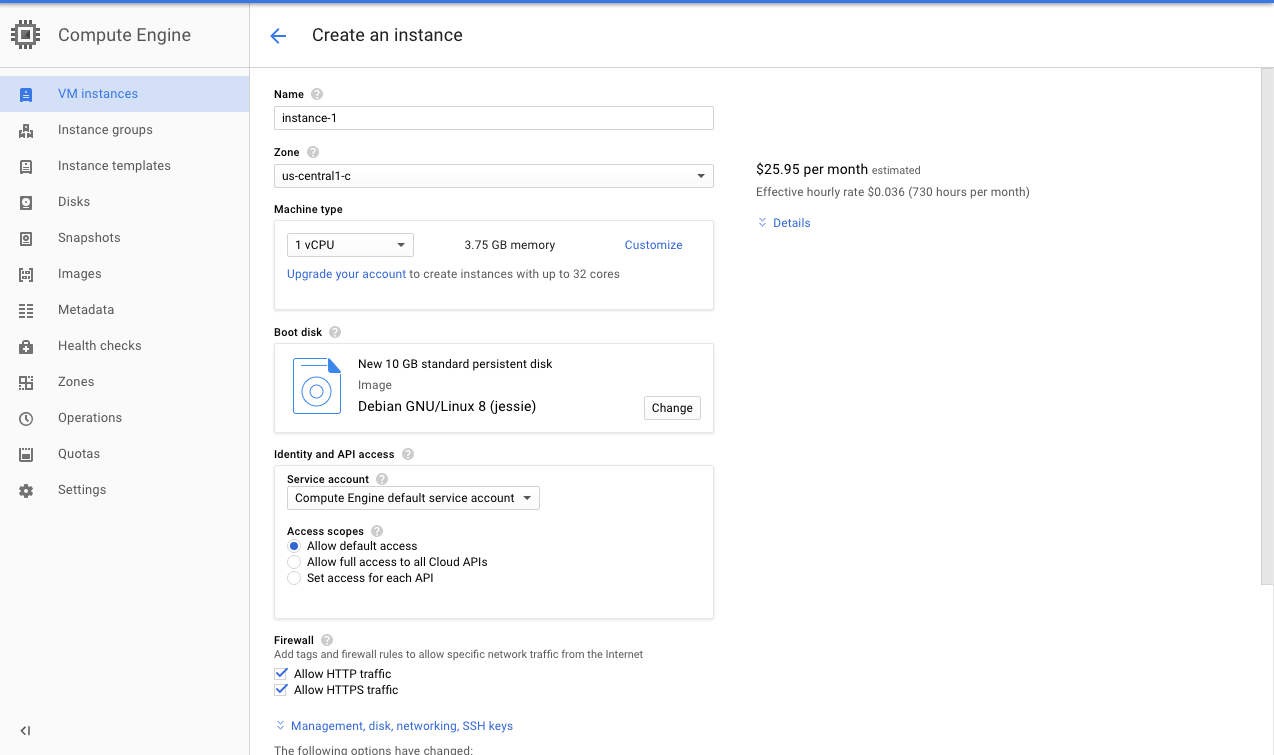
1. Once container is created, navigate to Compute Engine. A VM instance will be created in the cluster.



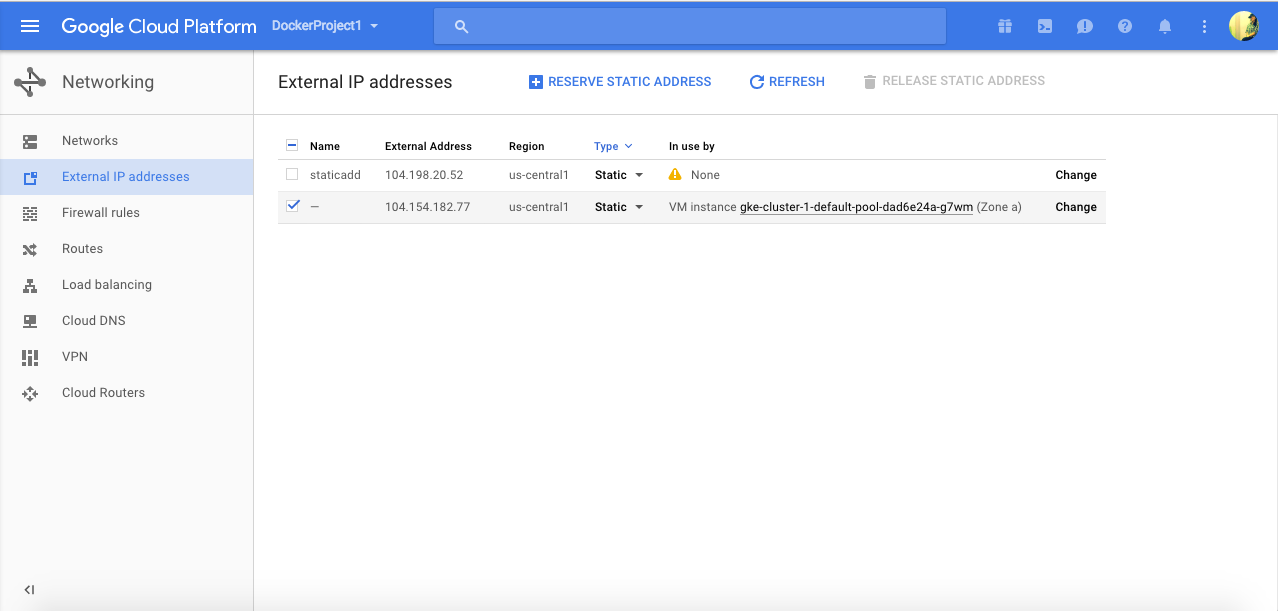
1. Click on the instance name and click on edit.



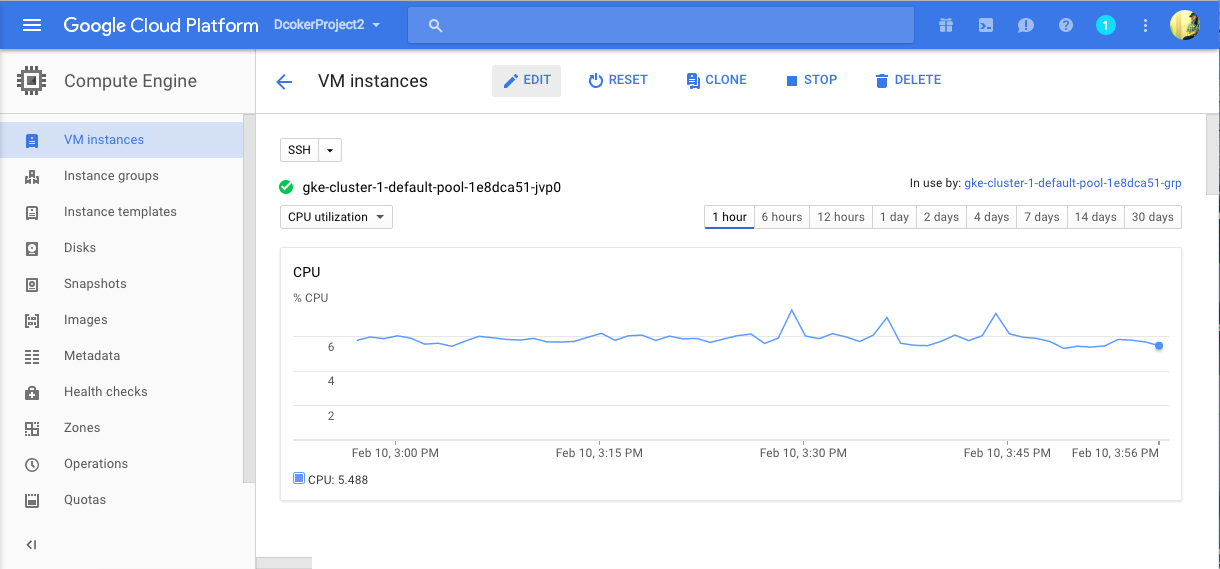
1. Scroll down and check Allow HTTP traffic and HTTPs traffic.



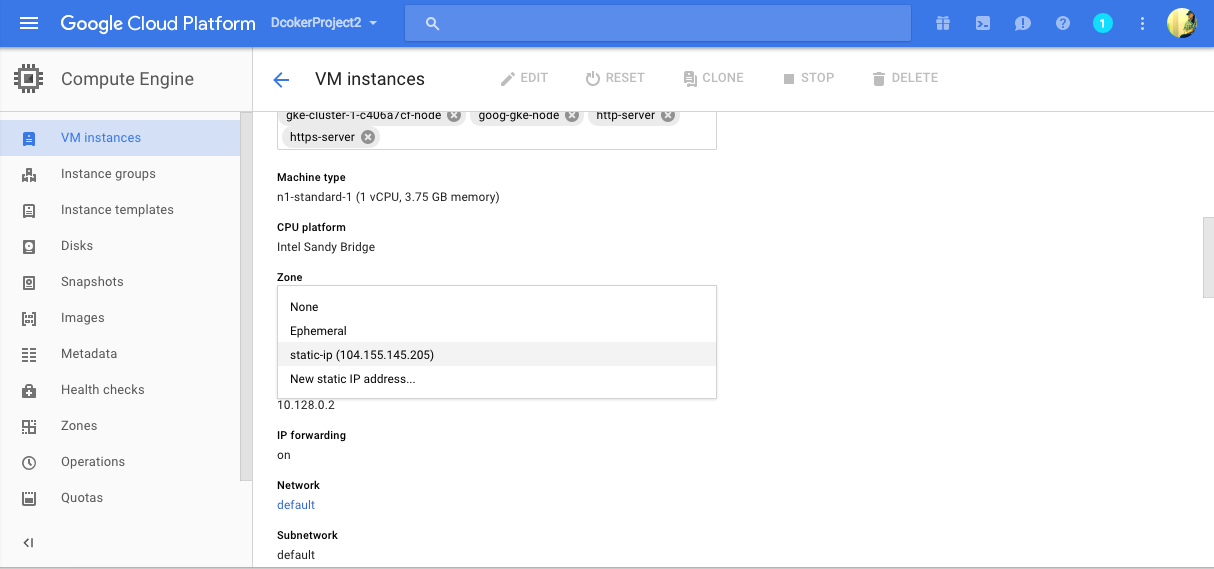
1. If you try to link to basic IP address to access jupyter notebook it will show, “This site can’t be reached” error. The problem is that the default ephemeral external IP address needs to be promoted a static external IP.
2. Change it to Static IP navigate to External IP address page on GCE Console and promote the ephemeral IP to a static IP. Navigate to **Networking -> External IP address.**



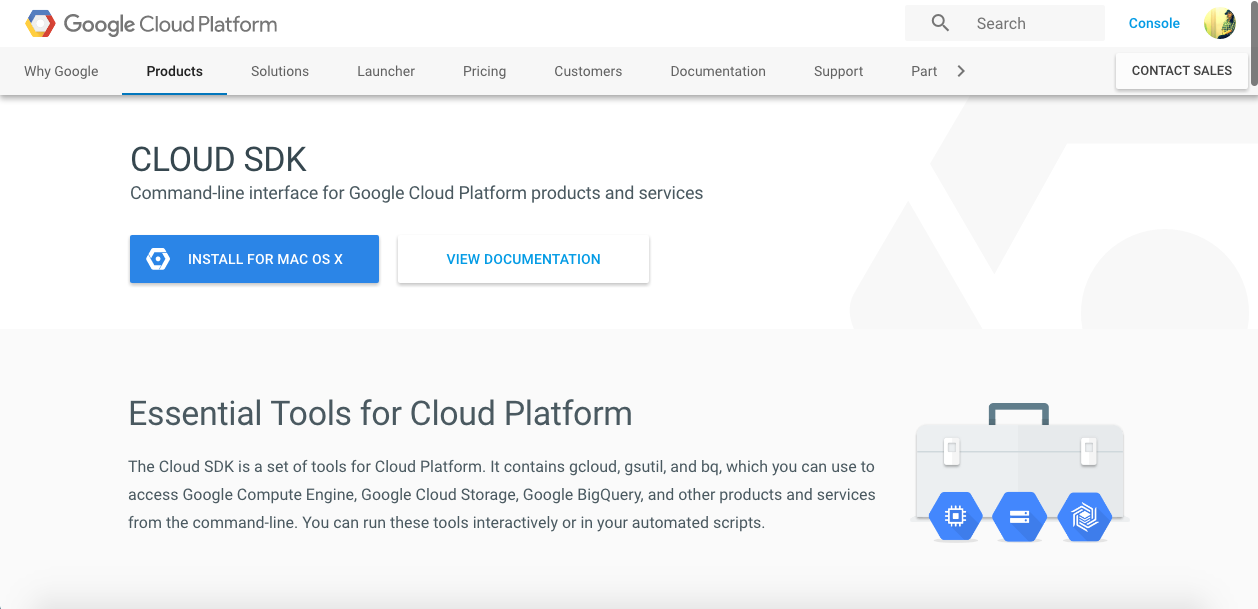
1. Once this is changed navigate back to the Compute Engine and click on the VM instance🡪 Edit



1. Scroll down and change External IP to Static IP from the drop down which you have reserved and save the changes.

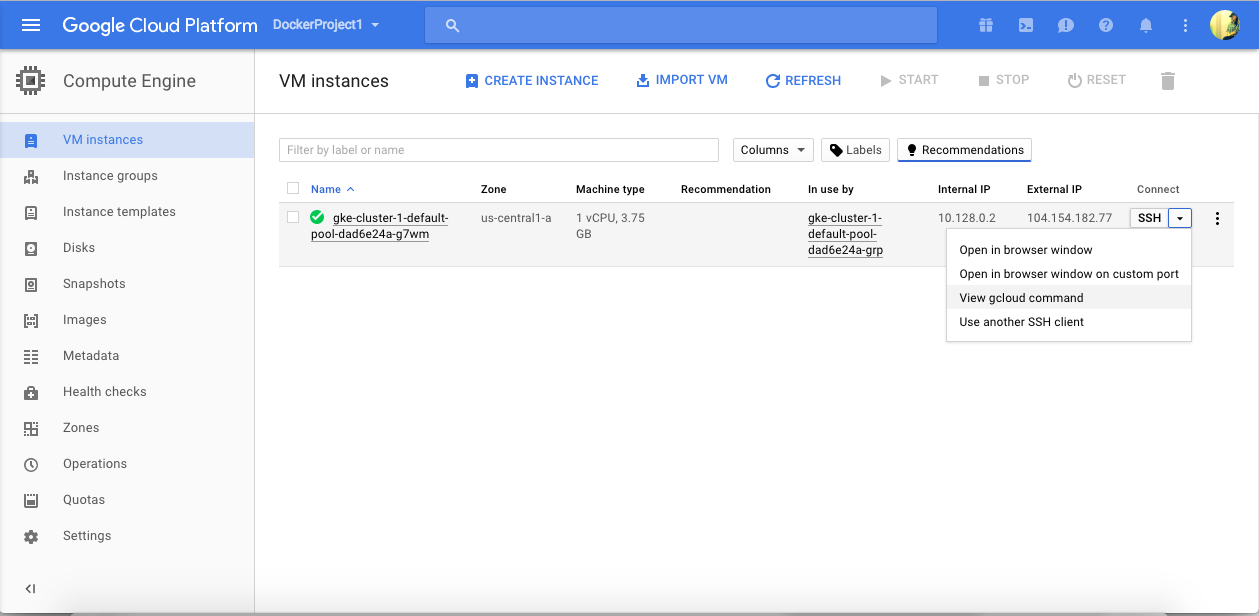
****

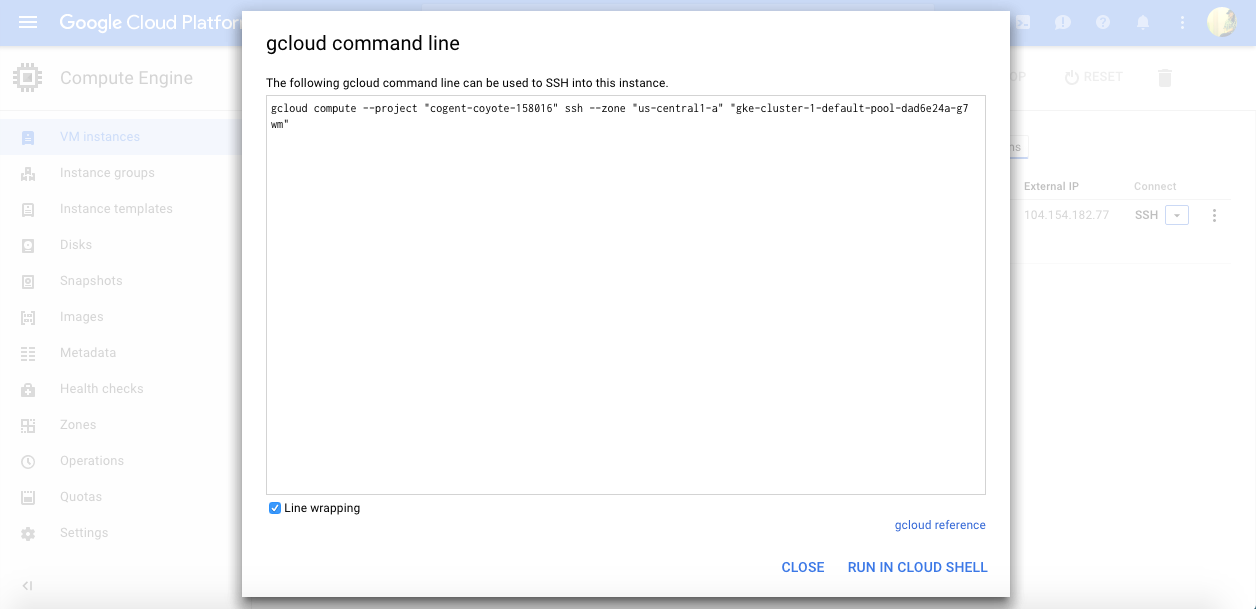
1. With the instance created, gcloud SDK makes it possible to connect via SSH from a local machine. Install google cloud SDK from: [**https://cloud.google.com/sdk/**](https://cloud.google.com/sdk/)**.** Also make sure python version is **2.7.x**



1. To connect to the container, go back to compute engine🡪 VM instances and right click on SSH and Click on View gcloud command to view the command. Copy the command and run it on the terminal. Also, you can either run this in cloud shell provided by the google or on the terminal and you will be connected to the container.

The command is **gcloud compute --project "<project-name>" ssh --zone "<your-zone>" "<instance-name>"**

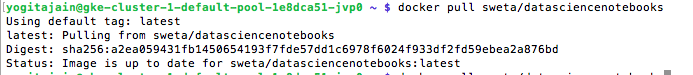






1. Now, you have connected to the container on google cloud. Let’s, pull a docker file image that has jupyter notebook installed on it.

$ docker pull sweta/datasciencenotebooks

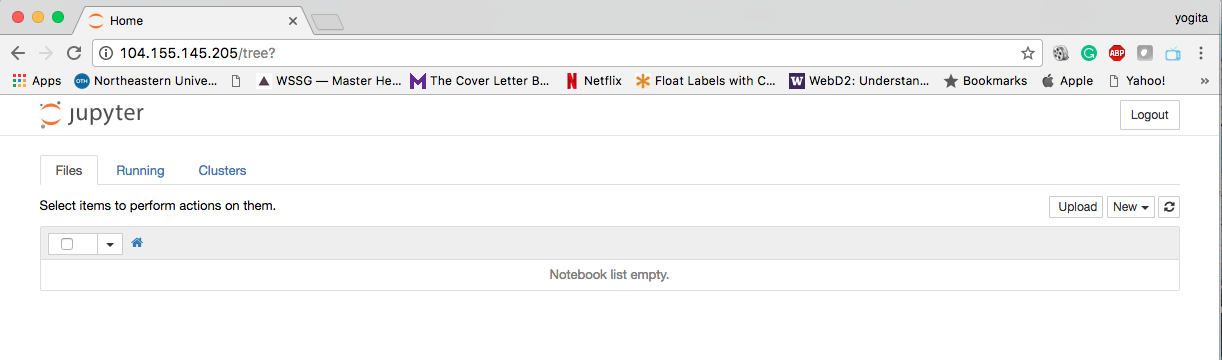


1. Docker run command will run the jupyter node on your static ip with port number 80.

$docker run docker run -d -p 80:8888 -e "USE\_HTTP=1" sweta/datasciencenotebooks



1. Now, type http:<static-ip-address>:80 in your browser and you will be connected to the jupyter notebook.



Google Cloud with Jupyter Notebook

1. **CONFIGURING DOCKER ON IBM BLUEMIX**

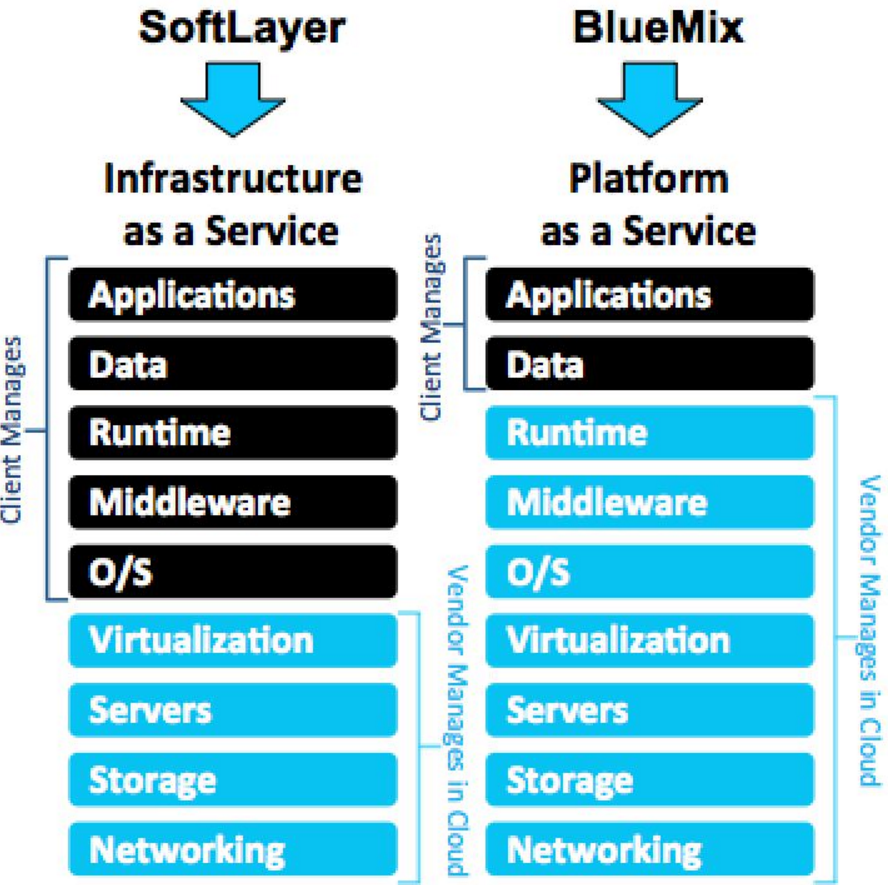
**What is IBM Bluemix?**

IBM Bluemix is a cloud platform as a service (Paas) developed by IBM. It supports several programming languages and services as well as integrated DevOps build, run, deploy and manage applications on the cloud. Bluemix is based on [Cloud Foundry](https://en.wikipedia.org/wiki/Cloud_Foundry) open technology and runs on [Soft Layer](https://en.wikipedia.org/wiki/SoftLayer) infrastructure. It delivers enterprise-level services that can easily integrate with your cloud applications without you needing to know how to install or configure them.

## What is Cloud Foundry?

Cloud Foundry is an open source platform as a service (PaaS) that lets you quickly create and deploy applications on the cloud. Cloud Foundry abstracts the underlying infrastructure needed to run a cloud, letting you focus on the business of building cloud applications.  The beauty of Cloud Foundry is that it provides choice. Developers and organizations can choose:

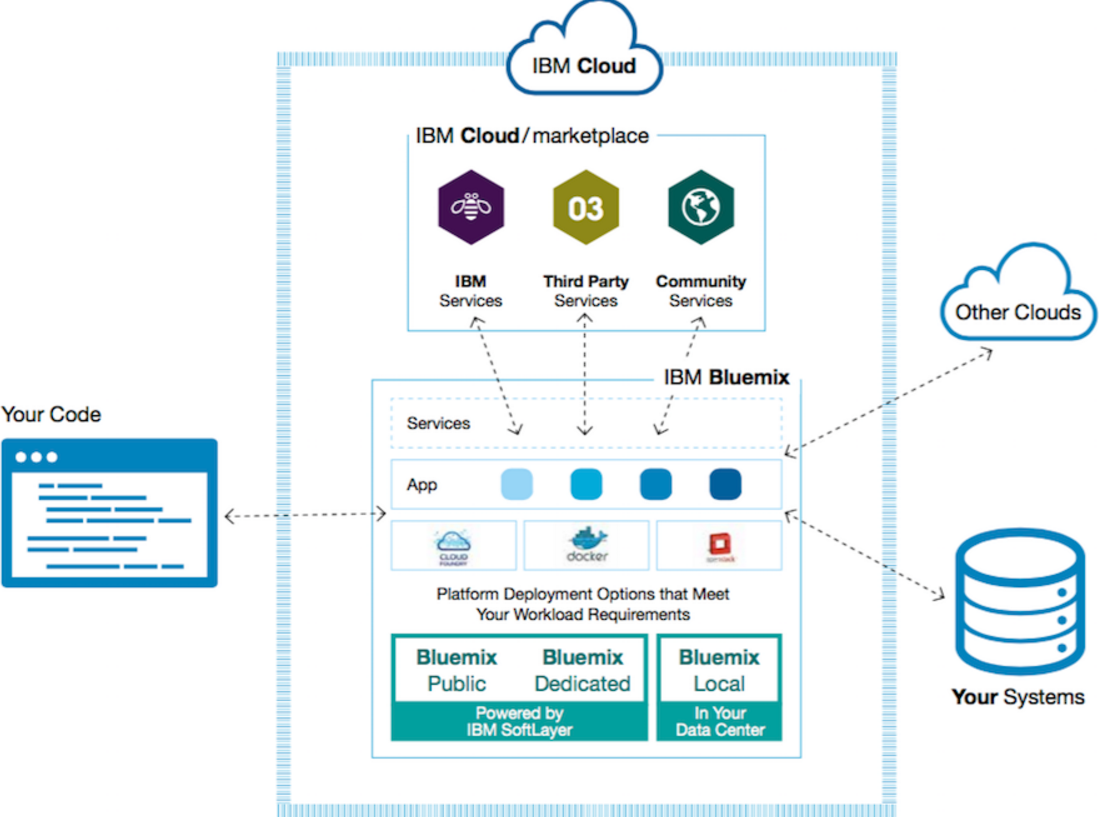
* Development Frameworks:  Cloud Foundry supports Java™ code, Spring, Ruby, Node.js, and custom frameworks.
* Application Services:  Cloud Foundry offers support for MySQL, MongoDB, PostgreSQL, Redis, RabbitMQ, and custom services.
* Clouds:  Developers and organizations can choose to run Cloud Foundry in Public, Private, VMWare and OpenStack-based clouds.



## Why IBM Bluemix?

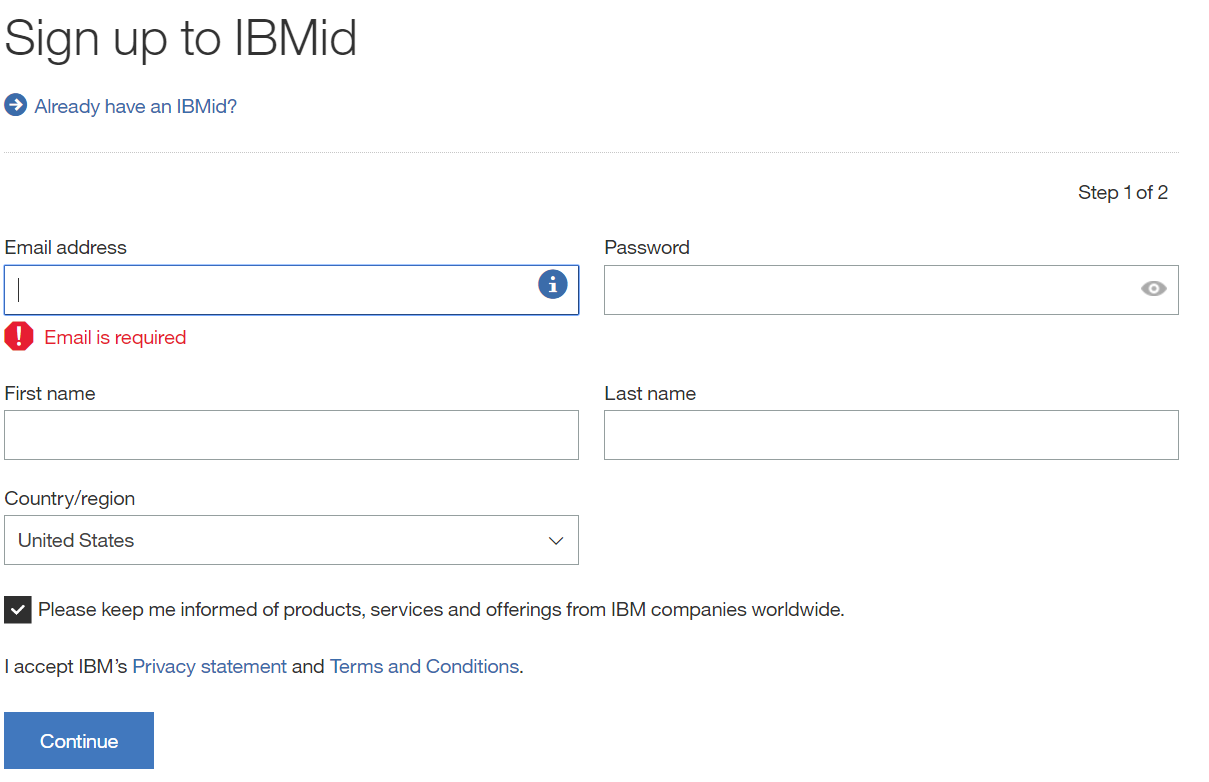
* Simplicity and speed: By focusing on the DevOps model, Bluemix can reduce the downtime of redeploying applications. The integrated environment provided by Bluemix allows developers to automatically deliver code without the hassle of building and debugging installation scripts.
* Command line: The Cloud Foundry (CF) command line provides integration for developers that prefer coding without an integrated development environment (IDE). This is also helpful for developing automation scripts with Bluemix. The CF application programming interfaces (APIs) can be integrated with multiple languages, frameworks and services.
* Eclipse: Since Eclipse is widely used by developers, they can continue to use the tools with which they are comfortable. The Cloud Foundry integration can be installed from the Eclipse Marketplace. This provides integration with Bluemix from the Eclipse client.
* Web IDE: Developers can work with the Web IDE directly in Bluemix. This allows modification of the application without any development environment installed on the developers’ laptops.

**Bluemix Container Service Infrastructure:**



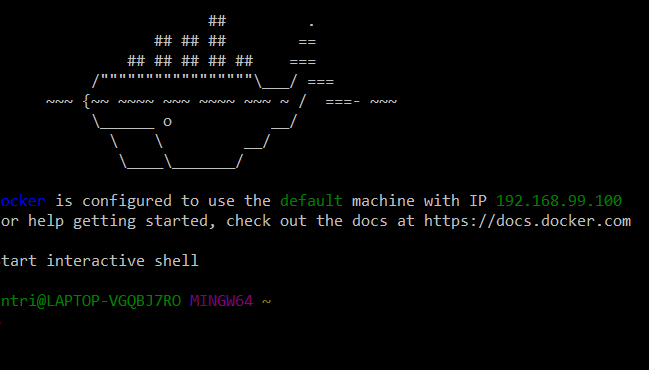
**STEPS TO CONFIGURE DOCKER WITH IBM BLUEMIX:**

1. Setting up a Bluemix account: Create a free trial blue mix account first. Free trial version can be created for 30 days.



1. Open Docker Quickstart Terminal: (All the commands will be entered on this Terminal)

After docker has been installed on the system, Start the Docker Quickstart Terminal



1. Set up the IBM Bluemix Container Service command-line interface

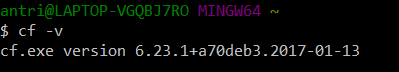
* Blue Mix is built on cloud foundry
* Install Cloud foundry Interface. It is a Github installation.
* Cloud foundry is based on Platform as a service rather than traditional infrastructure as a service.

1. Follow up the below link to install the Command link interface:

* <https://github.com/cloudfoundry/cli/releases>

The below command can be used to check the version of the inerface.

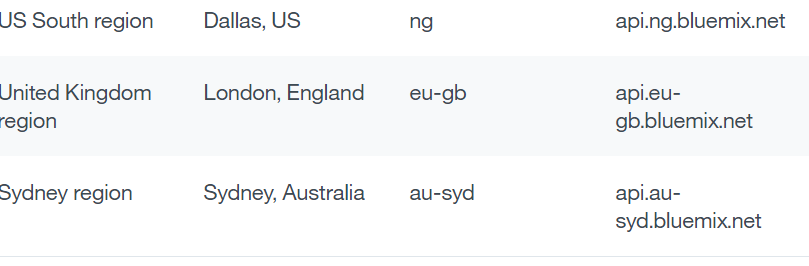
cf -v (cloud foundry version)



1. Login into blue mix using command line:

cf login -a api.ng.bluemix.net (Region US South)

* Every region has a different API to connect it with
* Insures security and faster turnaround time.

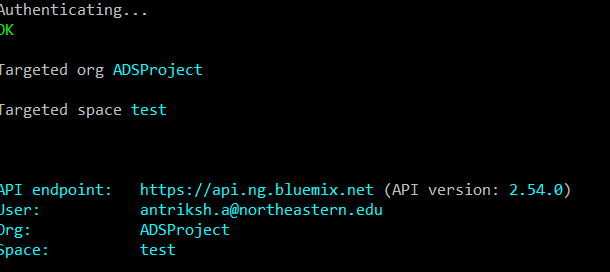


1. Enter your email and password:

After login for to the API the screen will prompt for email and password for your bluemix account (created at step 1)



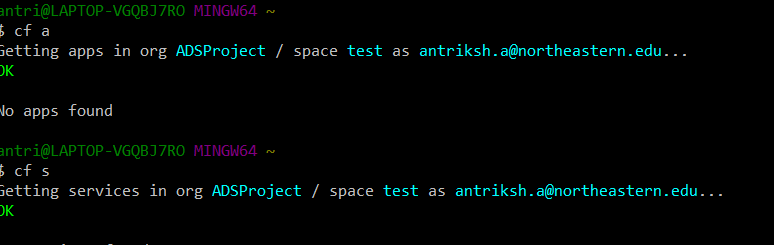
1. Check if the connection is established



(TO CHECK APPLICATIONS AND SERVICES) running on your Blue Mix:

cf a –> application

cf s –> services



1. Install the IBM Bluemix Container Service Cloud Foundry plug-in for your operating system:

* Docker is configured on Bluemix using IBM Container Service.
* Utilizing all the features of the service requires to install the cloud foundry plug-in.
* Different OS have different configurations to install the Plug-in

Below are the commands to set up Container service plug in for windows and Mac.

**FOR WINDOWS:**

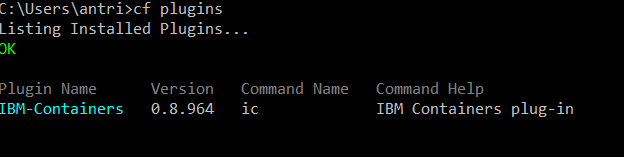
cf install-plugin https://static-ice.ng.bluemix.net/ibm-containers-windows\_x64.exe

**FOR MAC:**

cf install-plugin <https://static-ice.ng.bluemix.net/ibm-containers-mac>

1. Check if plugin has been installed using below command

Cf plugins – to check the plugins installed



1. Create a new Registry Name (Namespace) (Public such as a docker hub)

Docker hub is a public registry

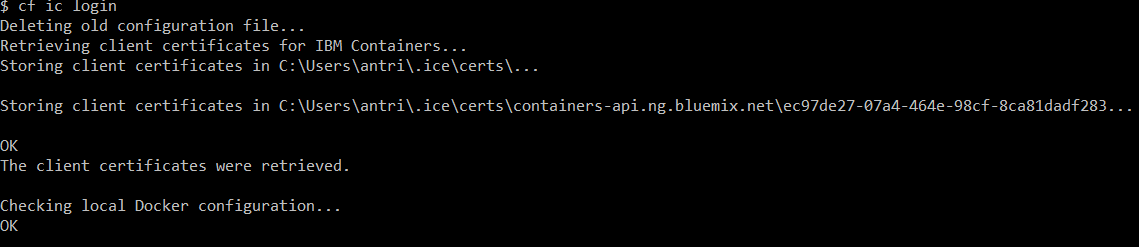
Bluemix lets you create a private registry where you can upload your images.

**- cf ic namespace set <your\_registry\_name\_here>**



1. Login into the IBM Container Service:

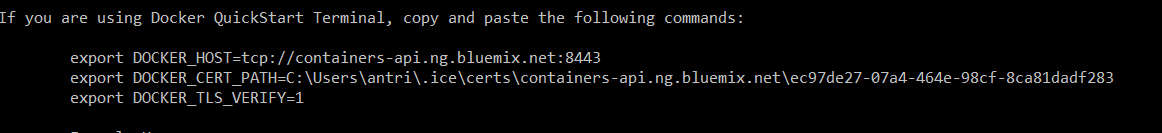
Cf ic login (cf for cloud foundry, ic for ibm container)



1. Accessing Bluemix using Docker or Cloud Foundry commands:

Commands such as - cf ic images can be used to access the bluemix images. However, bluemix provides you an opportunity to access bluemix using docker commands

The below commands allow docker to configure with Bluemix. Copy paste the below commands on docker terminal to configure the same.



1. Use the Docker Images command now to access the Bluemix images: Enter simple docker commands like -docker images , docker ps -a (for container)

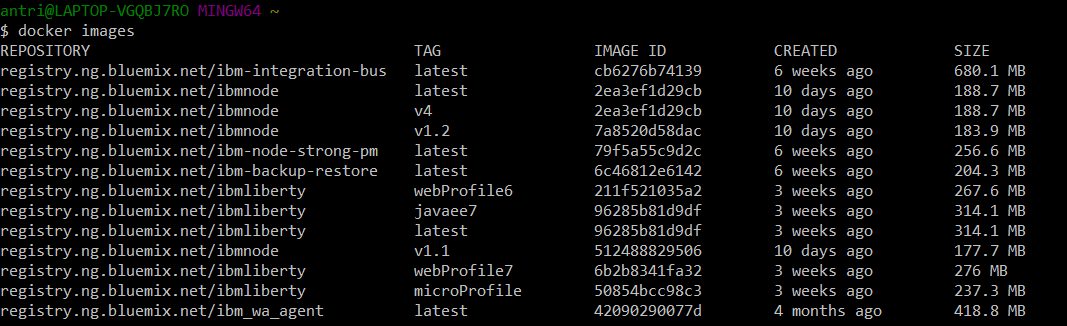


Figure: Shows Images provided by Bluemix

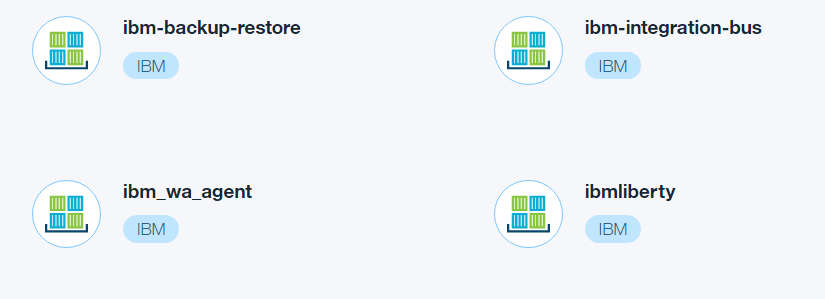
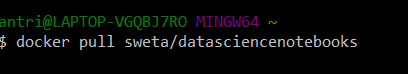


Figure: Shows Images shown on Bluemix Portal

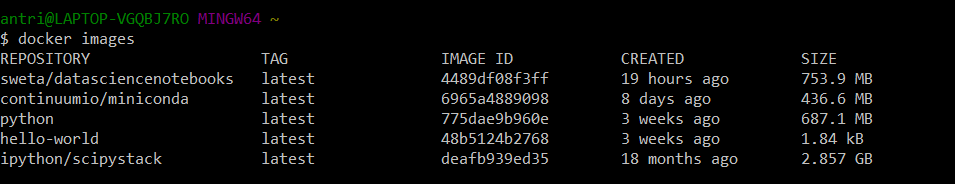
1. Adding Images to private Blue Mix images registry:

* First pull an image from Docker Hub:



(NOTE : BLUEMIX requires images to be pulled to your local docker registry before it can be pushed to bluemix. Docker push and pull from blue mix to docker hub doesn’t work with bluemix)

1. Check if the image has been pulled in your local docker registry:



1. Push the image from Local docker to Bluemix :

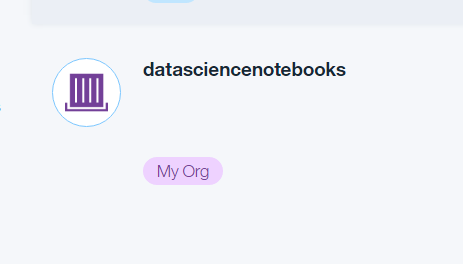
Use the below command on command line to push the image :

cf ic cpi <name\_of\_image> registry.<domainname>/<your\_namespace>/<new\_name>:tag>

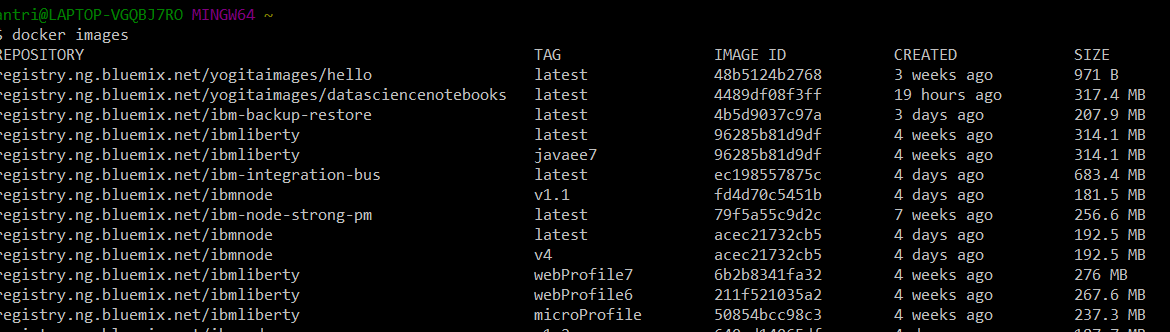
Example:



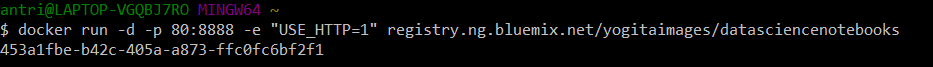
Image will now be seen on BlueMix:



1. Running the image on Bluemix: Check and note the name of the image pushed:

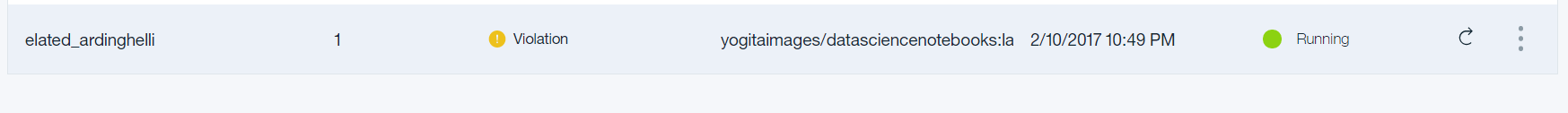


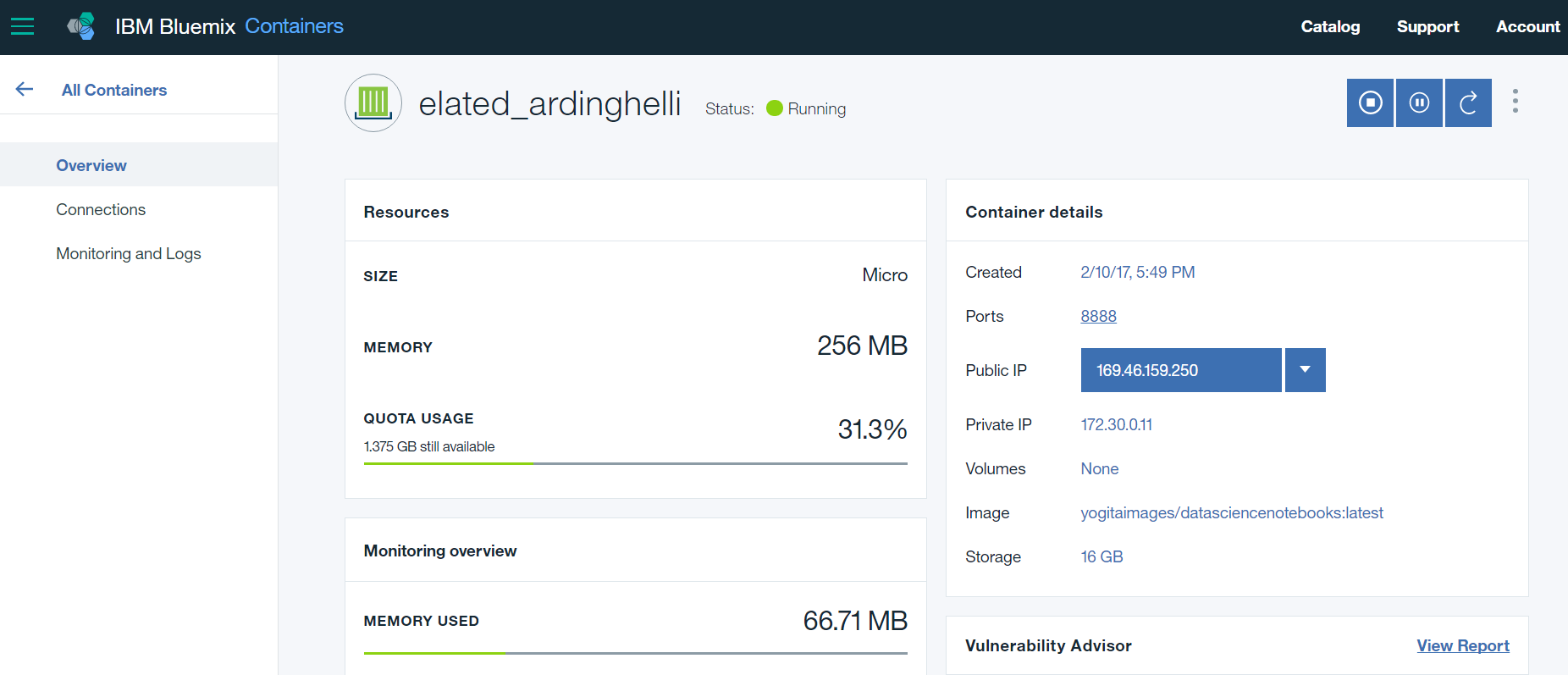
Running the image:



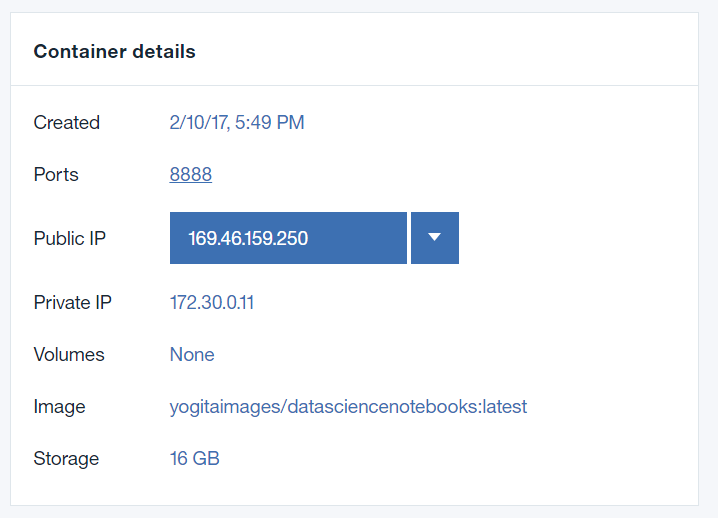
1. Specifies the IP to be used (-p :: Port number)

Name of the image to be run. Container will be running on Bluemix:



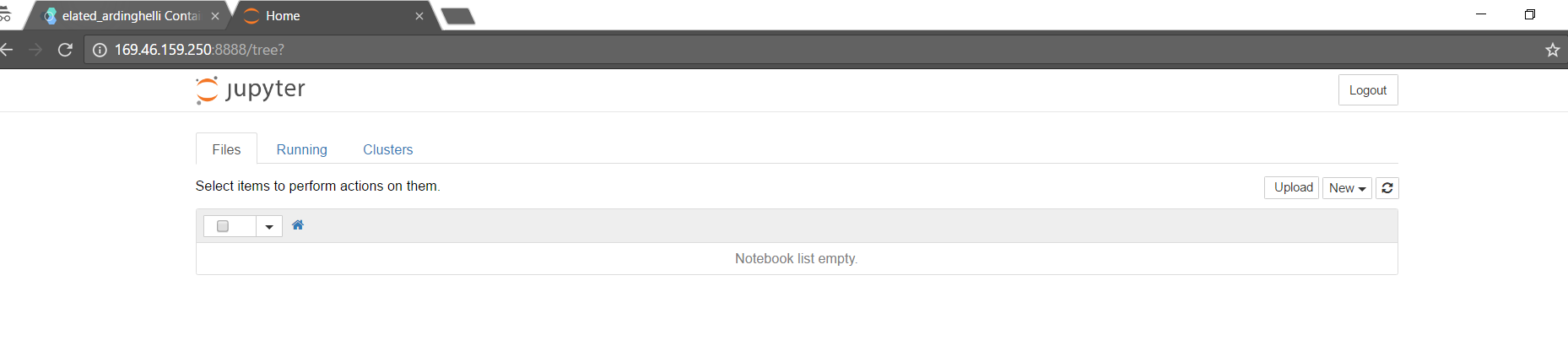


1. Note the Public IP and port generated for the container:



1. Open Browser and enter the URL to check of the container is running :

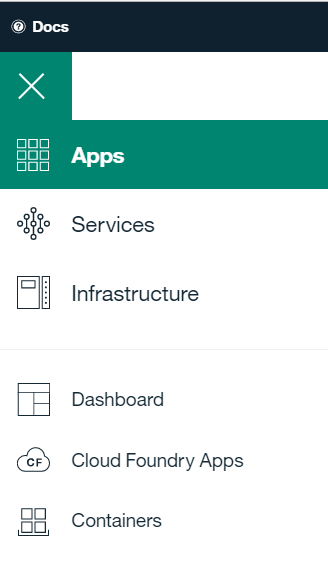
Use the IP address and port number from the above set :



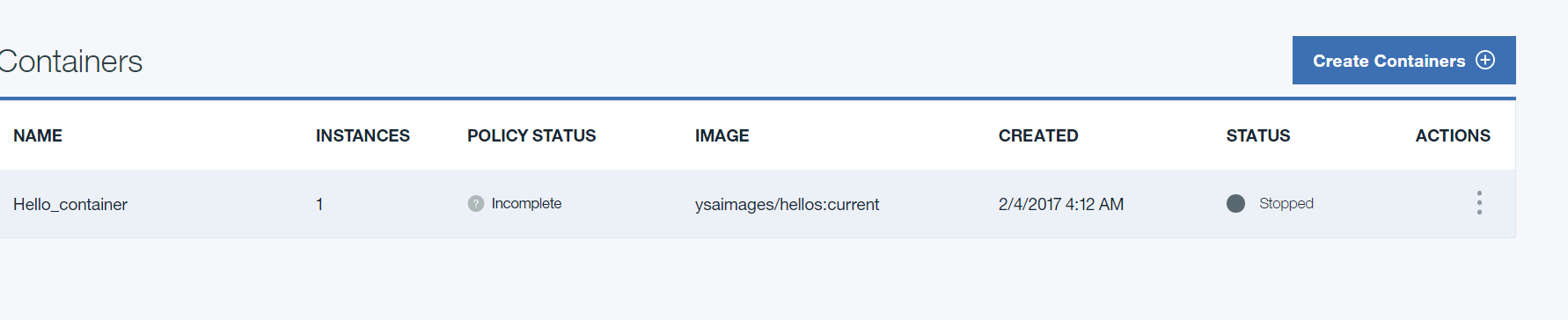
IBM BLUEMIX Cloud with Jupyter Notebook

**CREATING CONTAINER THROUGH BLUE MIX:**

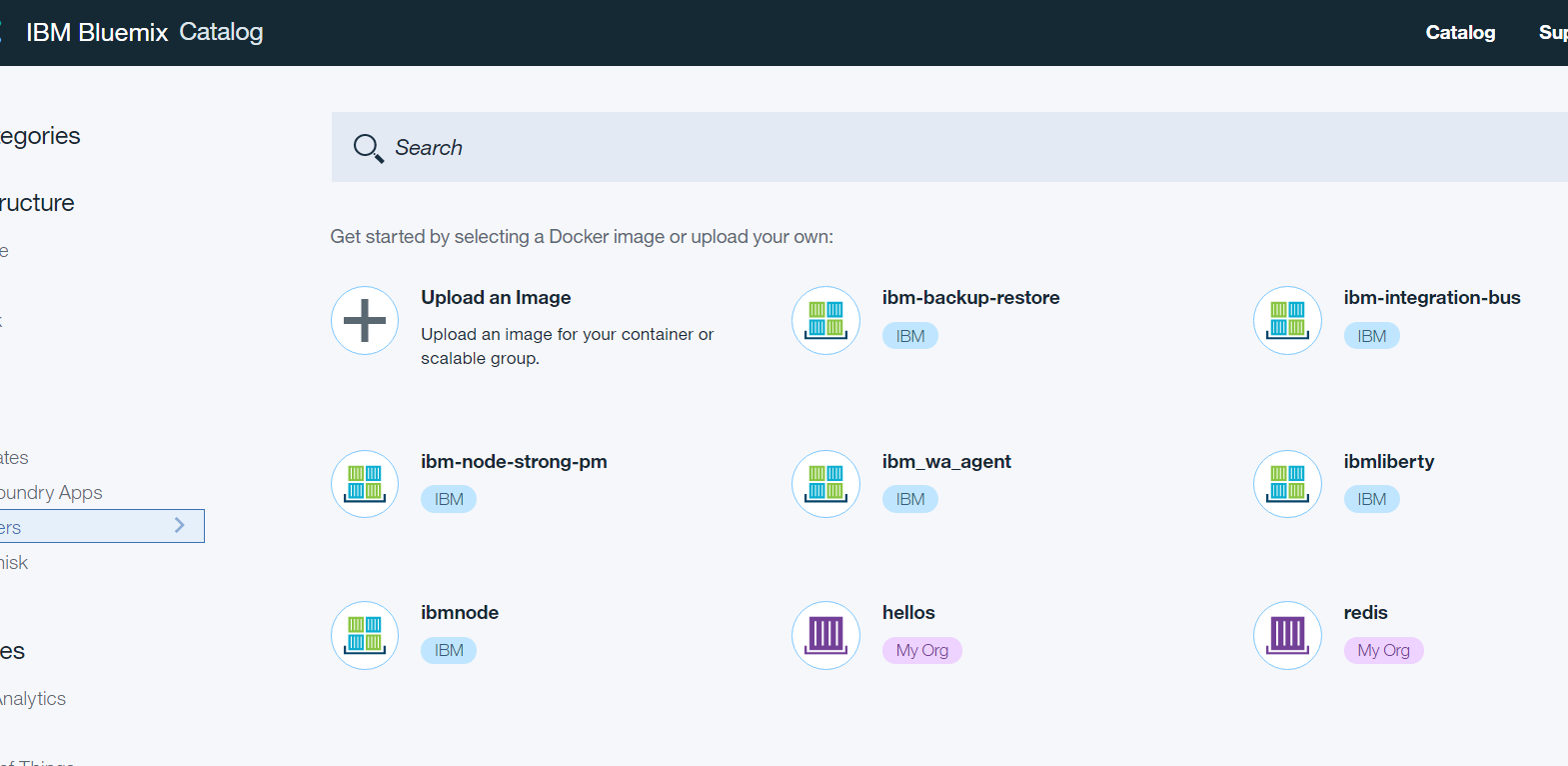
1. Select Container from the drop down on IBM BlueMix:



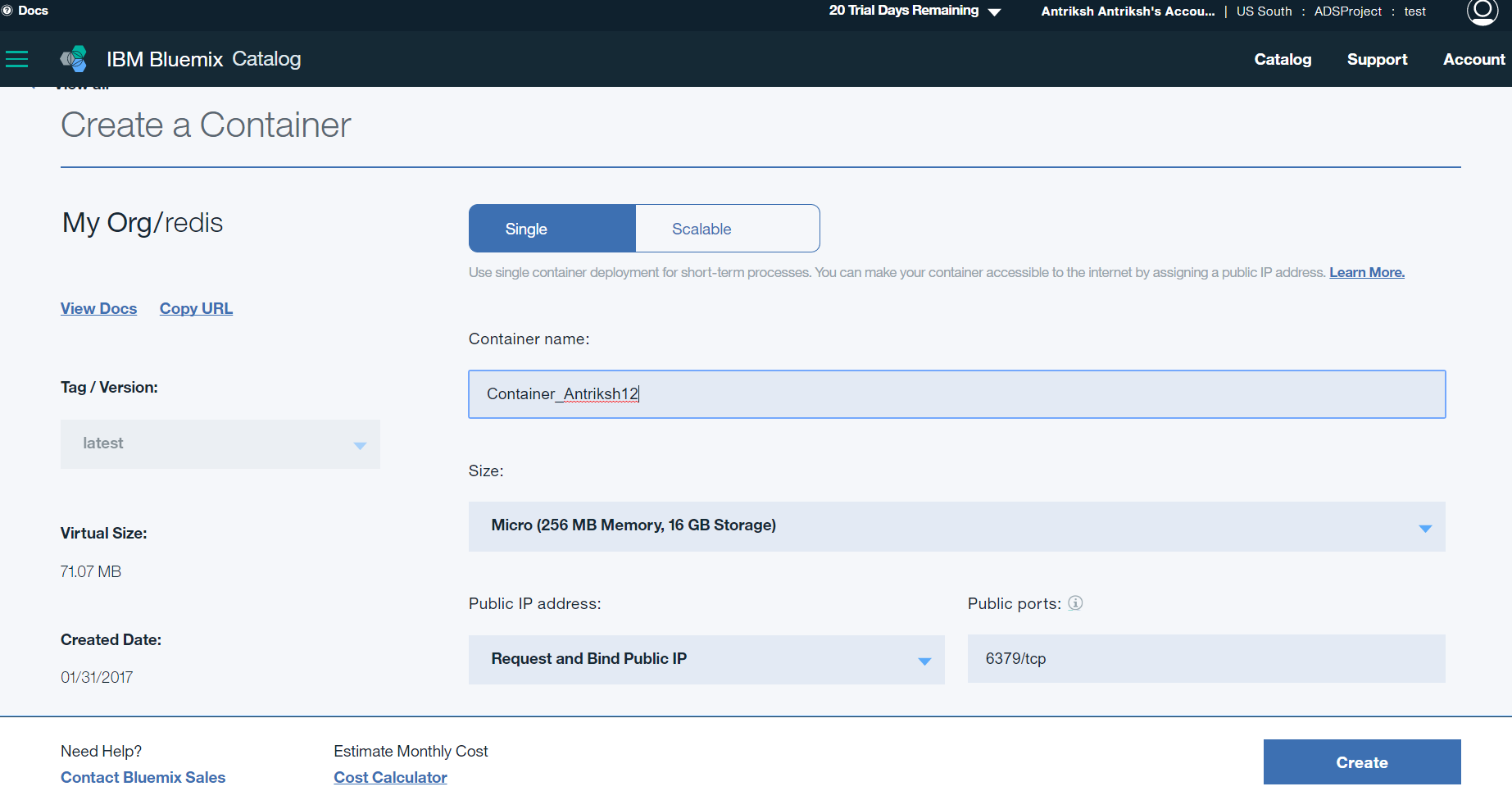
1. Select Create Container option on the containers option:



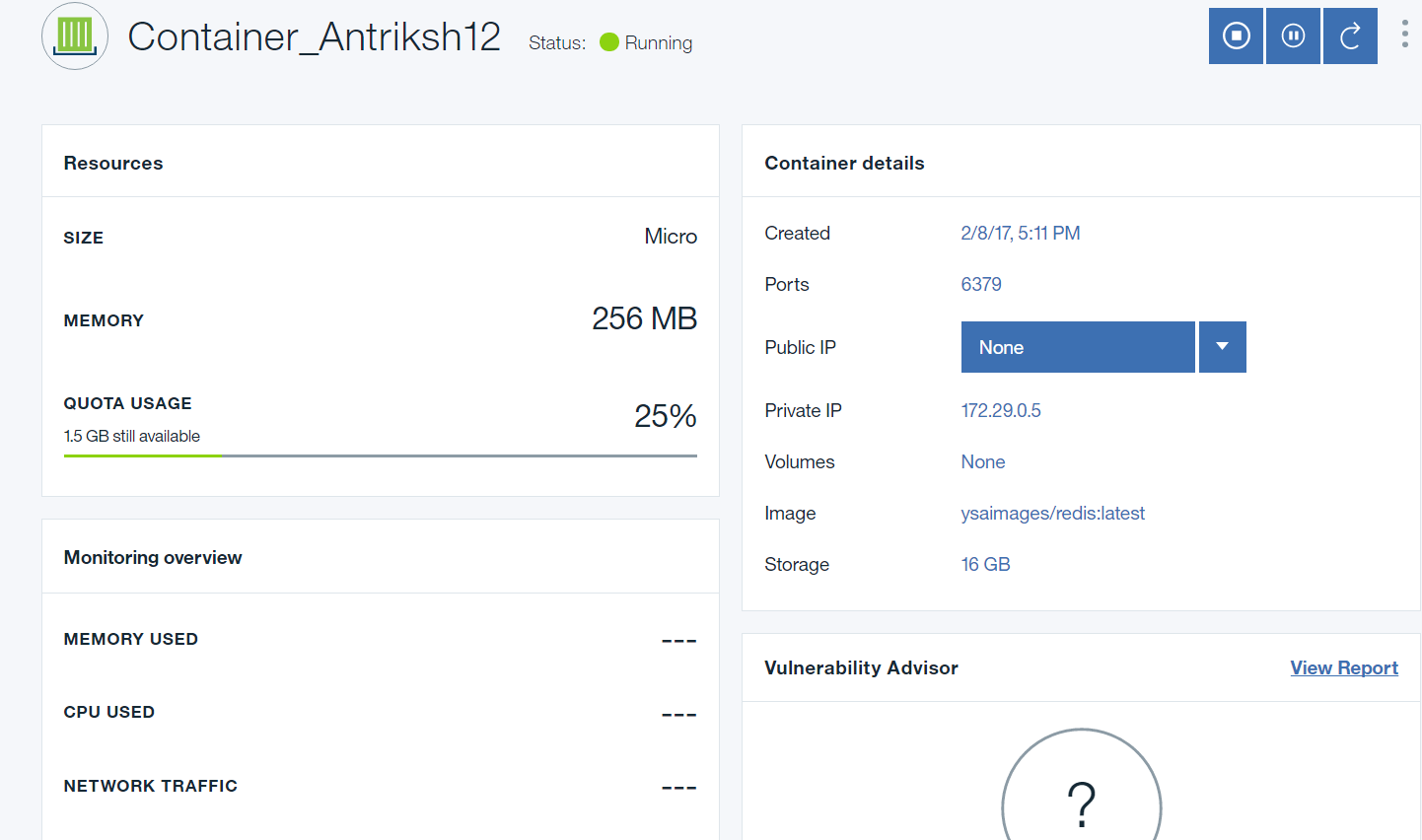
1. Select the image loaded from docker to the bluemix (images in purple)

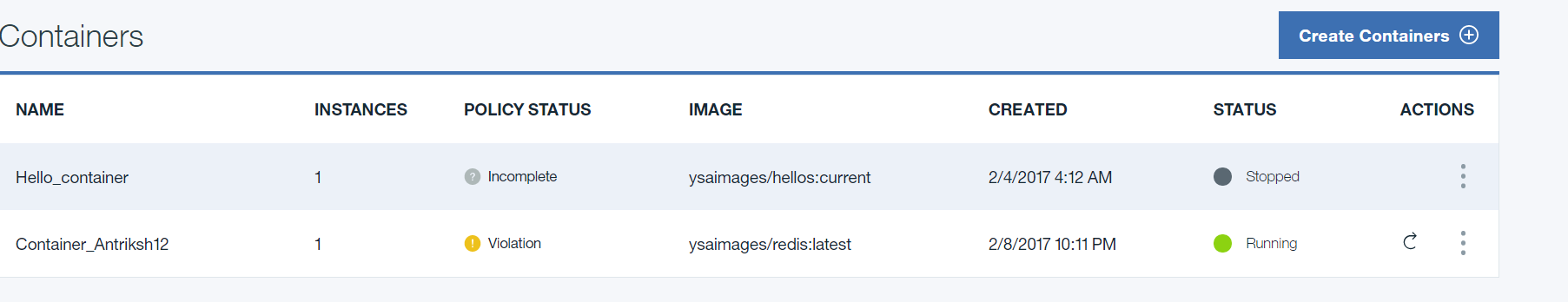


1. Configure the details for the container as shown below:

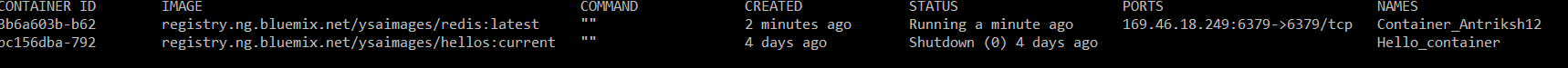


1. Container will be created:





1. CONTAINER IS NOW VISIBLE:



References:

1. <https://docs.docker.com/>
2. <https://github.com/docker/docker>
3. <https://cloud.google.com/compute/docs/containers/>
4. [https://rominirani.com/docker-machine-to-control-docker-hosts-on-google-cloud-3a48b46809dc#.kuvmabgcc](https://rominirani.com/docker-machine-to-control-docker-hosts-on-google-cloud-3a48b46809dc)
5. <https://cloud.google.com/compute/docs/instances/connecting-to-instance>
6. <https://blogs.msdn.microsoft.com/uk_faculty_connection/2016/09/23/getting-started-with-docker-and-container-services/>
7. <https://console.ng.bluemix.net/docs/>
8. <https://cloud.google.com/dataproc/docs/concepts/cluster-web-interfaces#interfaces>
9. <https://jeffdelaney.me/blog/running-jupyter-notebook-google-cloud-platform/>
10. <https://azure.microsoft.com/en-us/services/container-service/>
11. <https://www.youtube.com/watch?v=-wjxpng6jYs>