# AWS Method 1 (2 points) Description:

The 1st Method we are going to discuss is a simpler method, wherein we create an instance of our Linux server in AWS. Once we have the server instance then we can move the python source code, the requisite titanic data set files into a directory.

Once the files are available, then we can install all the requisite packages using pip that are needed for successful execution of our python program. After successful execution, we can make sure we have the requisite output files.

|  |  |
| --- | --- |
| Strengths | Weakness |
| Easy maintainability in case of small applications. | In this methodology scalability of our application is difficult. This is ideal for small applications. |
| Minimal infrastructure cost involved, since we are just using the Linux server in AWS and nothing more. | Manual installation process required, a configuration management specialist must make sure the environment is ready before we can execute the process. |
| Very less learning curve. Person with basic Linux background can get the job done. | Ideal only for small applications. |

# AWS Method 2 (2 points) Description:

The 2nd method is the we are creating a Docker image/container for the application and we are migrating the Docker image/container to the AWS server. For file storage, we can leverage the Amazon S3 to store our titanic dataset files and we can retrieve the files for our analysis. In my homework3 I have built the Docker image/container in the Linux instance and run the program. Instead we can leverage Amazon Container Service and achieve the same result.

|  |  |
| --- | --- |
| Strengths | Weakness |
| All the required environment is set for execution by going for Docker. The programmer can just concentrate on development rather than wasting time on environment setup. | In this approach, we are bound to the environment the container is built on, if there is some upgradation required, then the whole environment needs to be built. I see the approach has challenges when we want to upgrade our application. |
| The Docker container can be run of multiple server instances, there by application can be scaled in no matter of time. | Navigating the AWS screen’s might be little cumbersome. Needs expertise and good knowledge. |
| This approach is more suitable for large applications. | Some learning curve is required in terms of understanding how to use Docker effectively. |

# GCP Method 1 (2 points) Description:

GCP we can set up a virtual machine. We can install Docker and successfully run my python Titanic code. Data will be stored locally on the instance created, but could have been kept separately (and in larger volumes) in various formats (Cloud Storage, Cloud SQL, NoSQL, etc.)

Google has its own container system called Kubernetes. We could try to leverage that rather than using Docker. In GCP we can create clusters of containerized virtual machines to allow for collaborative work and it is called Container Engine.

|  |  |
| --- | --- |
| Strengths | Weakness |
| Easy maintainability in case of small applications. | I think GCP is costlier compared to AWS. |
| Minimal infrastructure cost involved, since we are just using the Linux server in GCP and nothing more. | Not best for huge volume of data, because the cost my escalate pretty quickly. |
| Access of linux vm from web page rather than through terminal. This has more flexibility. | Ideal only for small applications. |

# GCP Method 2 (2 points) Description:

Going for Googles specialized machine learning service designed to handle big data. With huge volume of data from different sources, we can use Googles existing services like search et.al. capabilities to consume the data, wrangle, predict and score the threats in real time basis. Machine learning service is integrated with Cloud Dataflow service that integrates with Apache Spark and Kafka. Even we can leverage Tensor flow.

|  |  |
| --- | --- |
| Strengths | Weakness |
| Extremely powerful computing systems with proven results. | By going to the Googles, Machine learning platform we will not be in a position of try different ML platform, we might be stuck with what Google must offer. This is like getting trapped in one eco system. |
| Google is leader as far as Machine Learning, so this will be a go to platform for industrialized datasets and computation logics. | Establishing such systems might be expensive. |

# Steps Followed for HW3:

For this homework, we must sign into AWS Console.

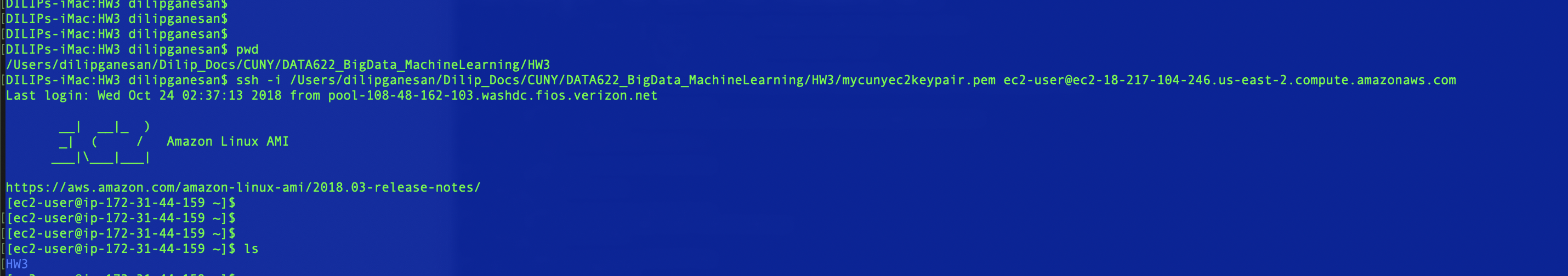
1. Sign into the [AWS Management Console](https://www.google.com/aclk?sa=l&ai=DChcSEwjO1tWC1PzdAhWCq5YKHXSCD4gYABAAGgJ0bA&ei=clm-W8bYAcGq8QWKz5OABQ&sig=AOD64_2AAoUk3WmpvplLVH65kdrCrDqi_A&q=&sqi=2&ved=2ahUKEwjG2dCC1PzdAhVBVbwKHYrnBFAQ0Qx6BAgEEAI&adurl=)
2. Click on the link "EC2" under Services to Launch an Instance.
3. We will be creating a Linux instance, we can also use an existing template which might have all the required software’s and packages already setup, so we can start executing the program. Since we do not have any template for instance, we are going to create a new one.
4. In the instance page, we have all the requisite information required to login to the server. We must make sure we save the pem file for successful login into the server.
5. If we are using windows, we have to use putty to login to vm box, since I use mac i use my terminal to connect to Linux box using ssh.  
   ssh -i /path/to/key/file.pem ec2-user@[dns address]
6. Once we have successfully logged in to the Linux box we must do some basic installation.
7. Install [Git](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git) and [Docker](https://docs.docker.com/install/linux/docker-ce/ubuntu/) and start the Docker service.
8. git clone your repository into the Linux box
9. Build docker image  
   docker build -t [TAG\_NAME].
10. Run the docker image  
    docker run -it [DOCKER IMAGE NAME]
11. On successful execution, we can see the output of our model.

# Screen Shots:

1. Logging inside the AWS Console. We make sure our instance is running.



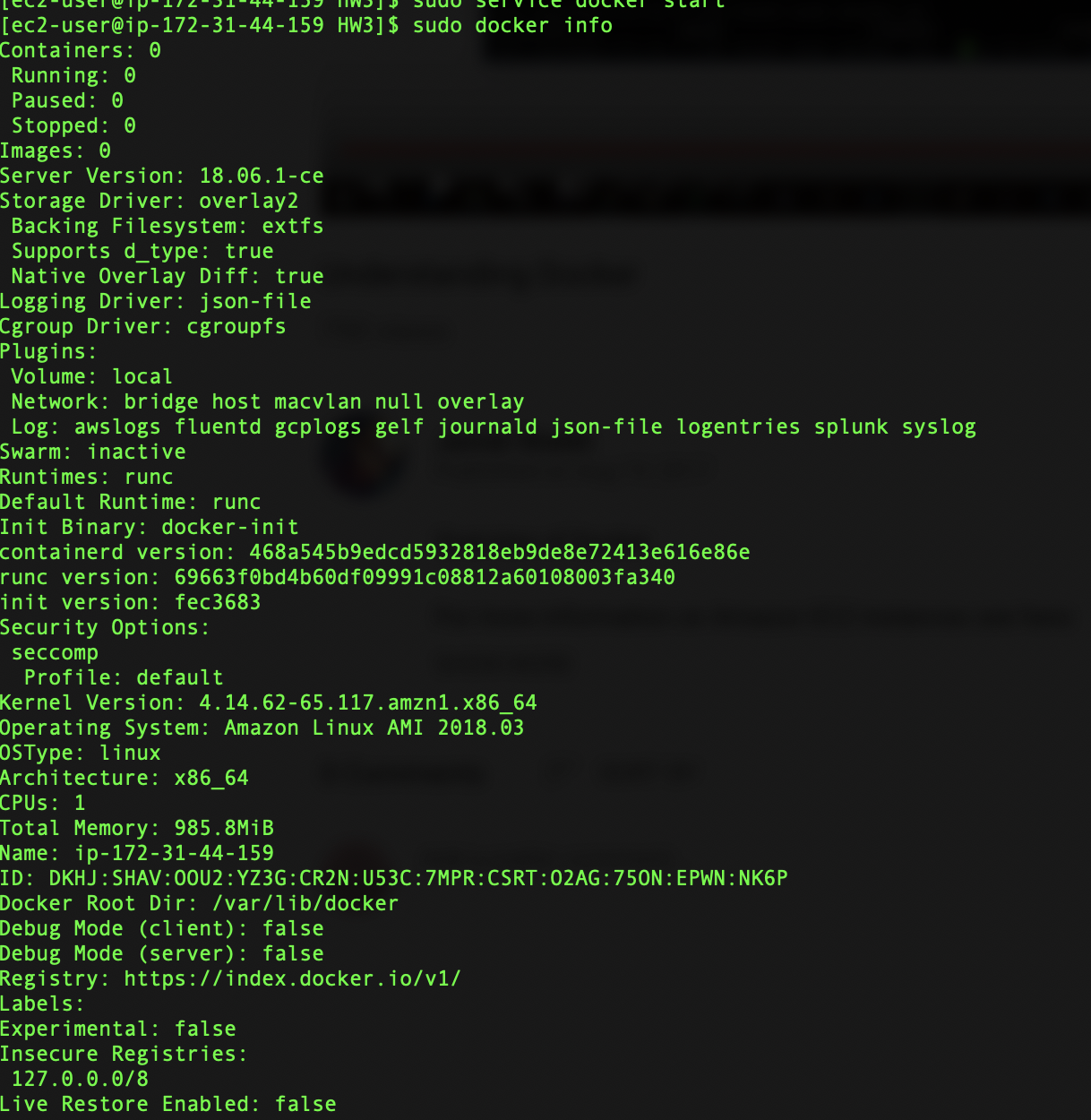
1. Using ssh we will login inside the Linux box.



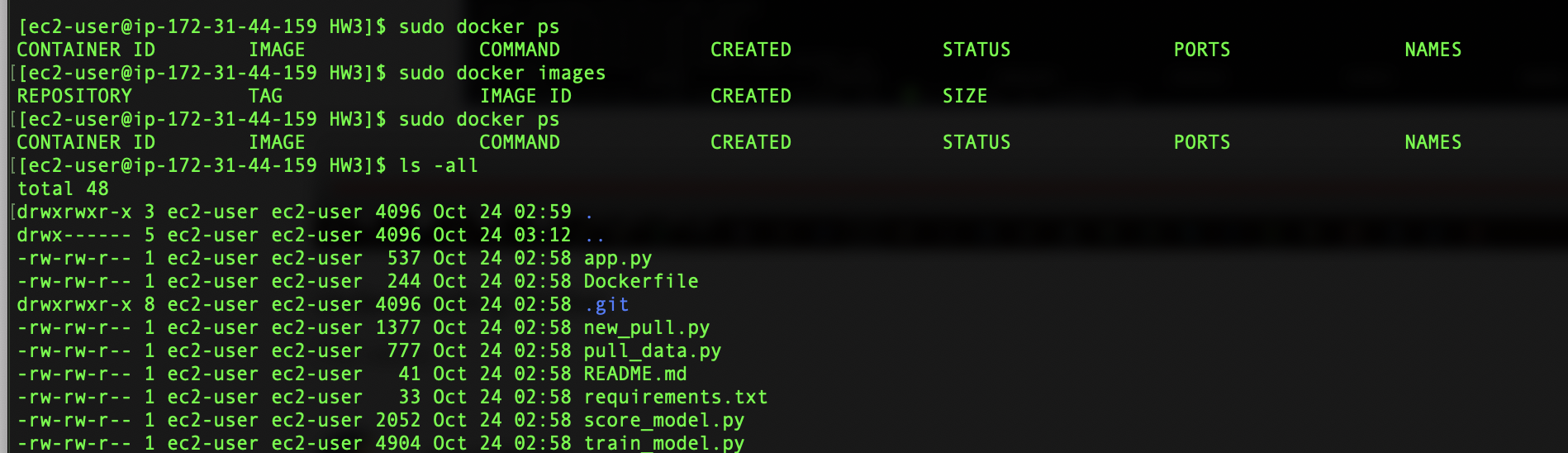
1. Git clone the HW3 inside the EC2 Image.



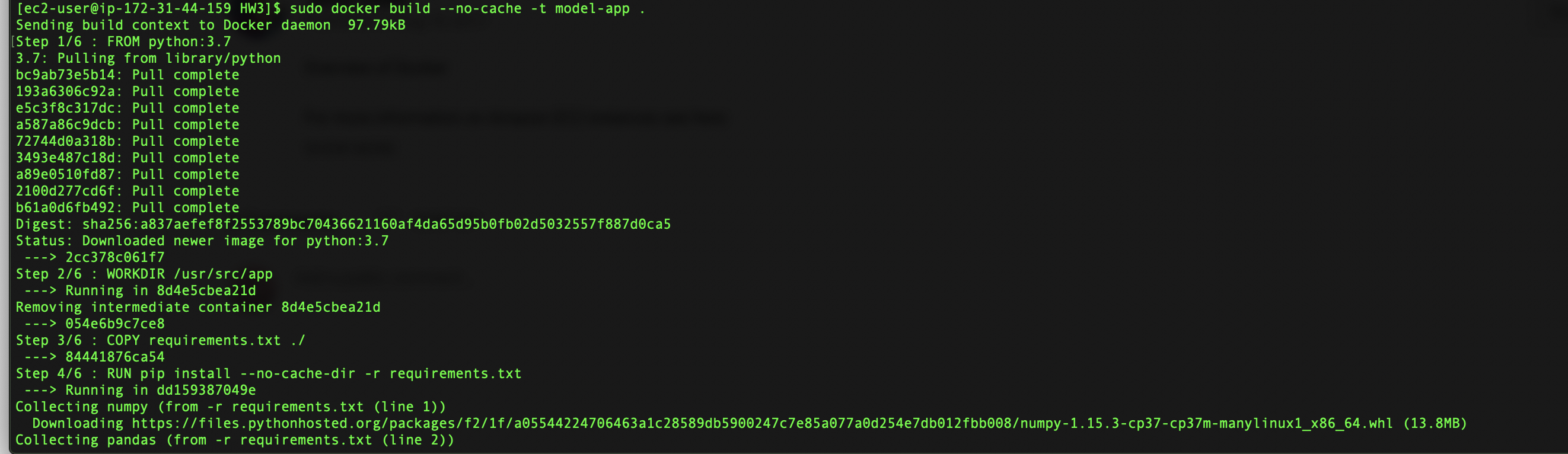
1. Starting the Docker Service.



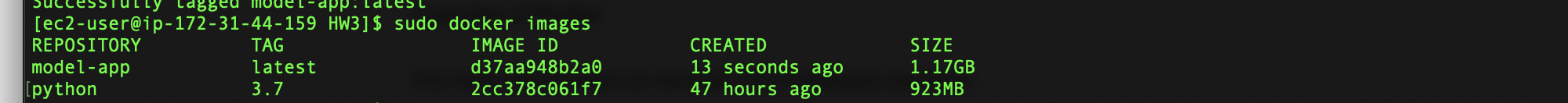
1. Making sure whether are there any existing images/containers.



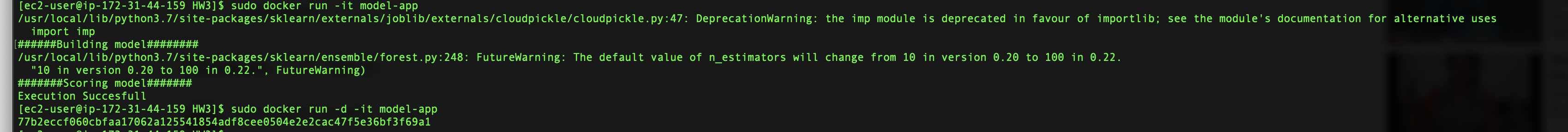
1. Running sudo docker build –no-cache –t model-app .



1. After successfully building the Image, making sure we have the image ready for execution.



1. Execution of docker container.



1. Our Model output with logging enabled.

