**Continuous Monitoring on Docker with ELK Stack**

**(ELECTIVE)**

**Git hub link -** [**https://github.com/SathishSubramanian96/ELKExample.git**](https://github.com/SathishSubramanian96/ELKExample.git)

## **Table of contents**

* General info
* Prerequisites
* Technologies
* Steps to be followed

## **General info**

The aim of the project is to perform continuous monitoring on spring boot application using ELK stack.

**Pre-requisties**

User should have Docker, Jenkins installed on same machine with the set of required plugins in Jenkins (Docker, Docker pipeline, build pipeline, etc.,). Also we need to have docker hub account to proceed. We can create docker hub account using the following url – <https://hub.docker.com>.

**Technologies**

* Github
* Docker
* Docker-Compose
* Jenkins

**Steps to be followed**

**STEP 1** Set up ELK stack on Docker

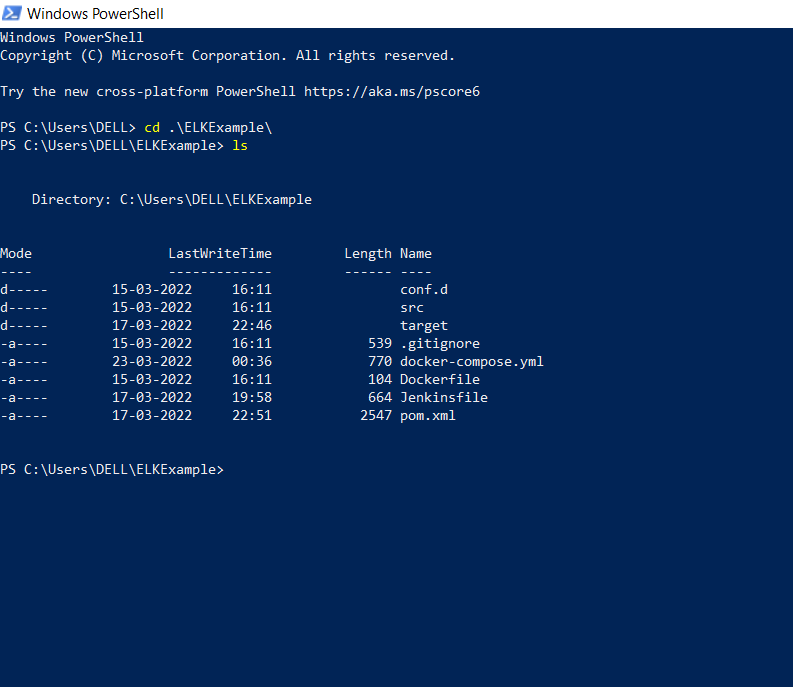
* We will create a Docker Compose YAML file that will run all three components of ELK stack on Docker environment. Once that is done, we can access Kibana Dashboard to see the logs pushed to application.
* Download Docker compose file in one of the git repositories and follow the set of commands given below to initialize the ELK stack.

**sudo su**

**git clone** [**https://github.com/SathishSubramanian96/ELKExample.git**](https://github.com/SathishSubramanian96/ELKExample.git)

**cd ELKExample**

**ls -alrt**

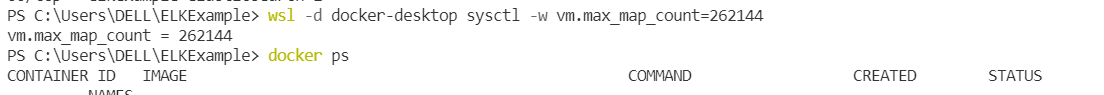


* Start the ELK stack using the docker-compose command. Usually, this binary is not installed on a server. So, follow the set of commands given below to check version and run Docker Compose.

***docker-compose version***

* Before starting the ELK stack, run the command given below so that elastic search is configured properly.

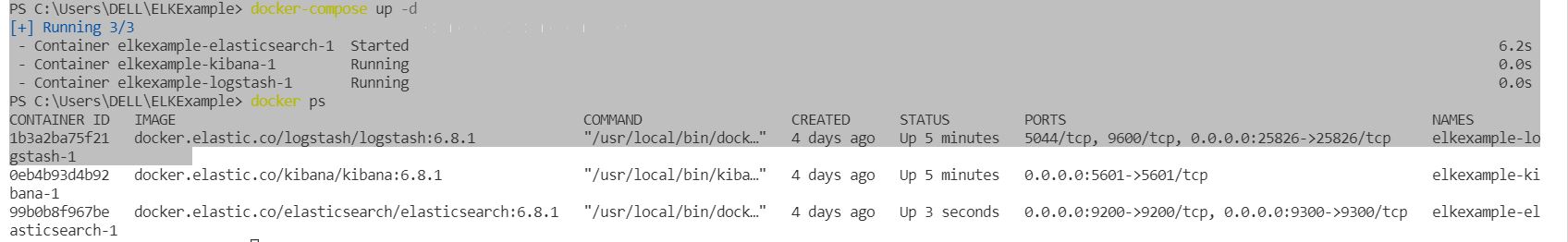
***sysctl -w vm.max\_map\_count=262144***



* Start the docker compose services and check them using the following command

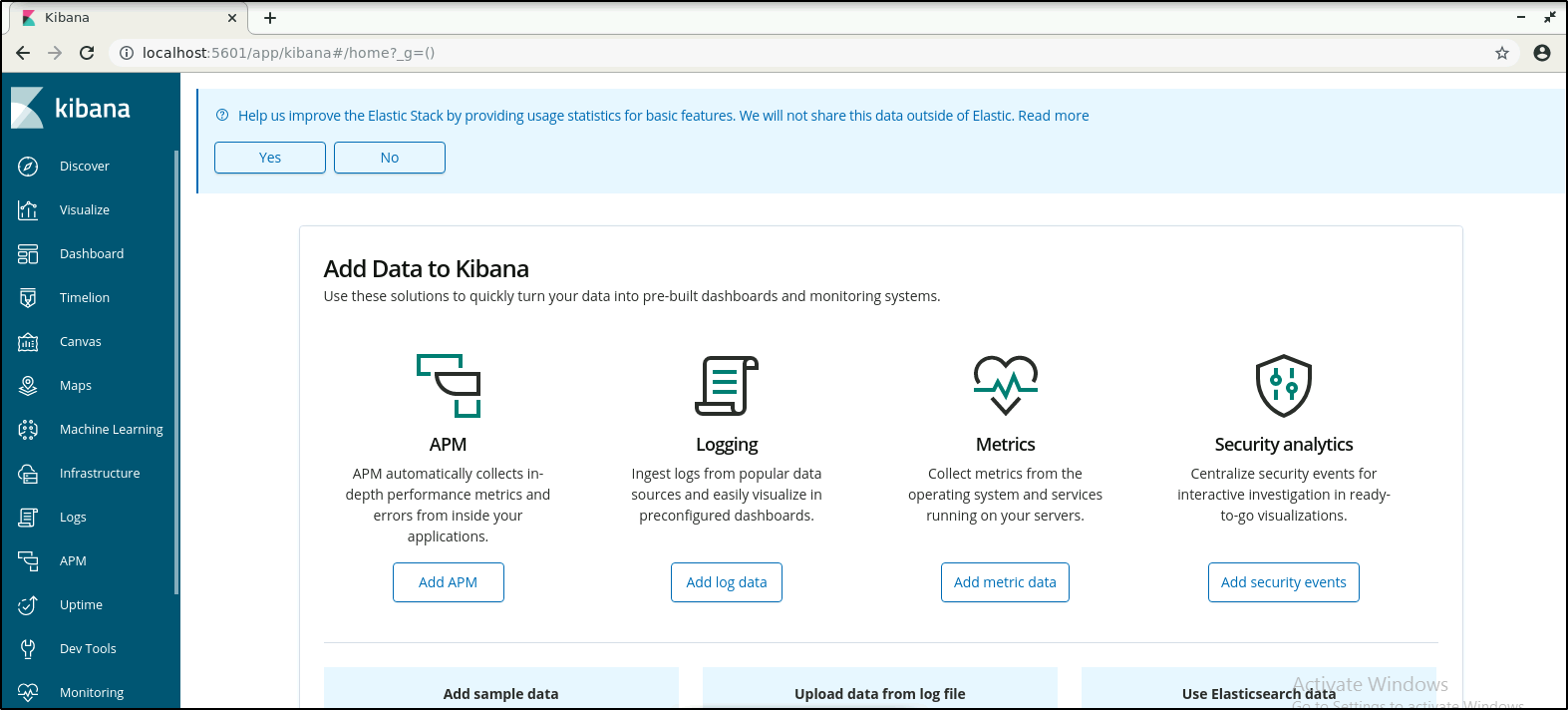
***docker-compose up –d***

***docker ps***



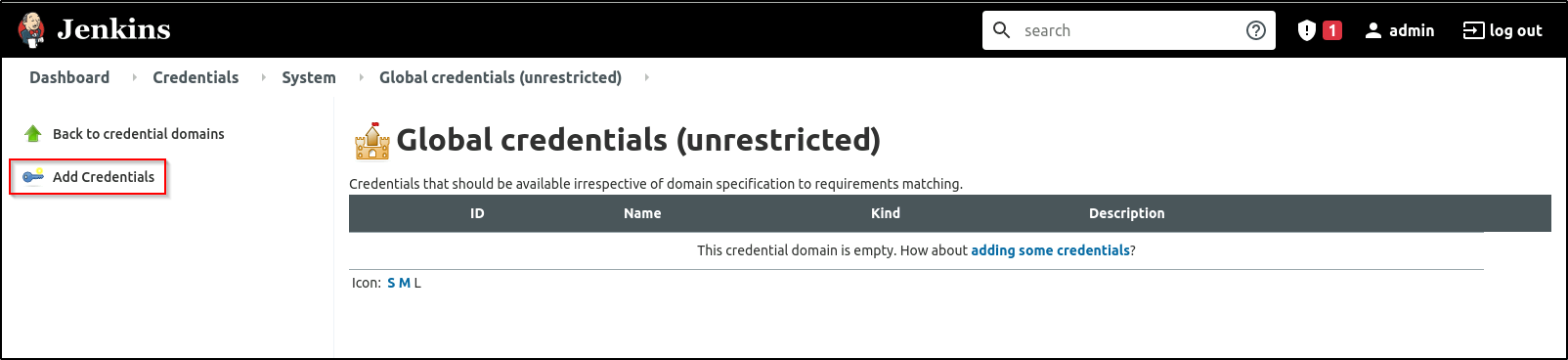
* Open the Kibana URL using the public IP of the host and 5601 port to access the Kibana dashboard.

***http://localhost:5601/app/kibana***



**Step 2** Configure Jenkins pipeline for Docker build and deployment

* From the browser, navigate to <http://localhost:8080> and login to Jenkins.
* Configure your Docker hub credentials in Jenkins. Go to *Manage Jenkins* -> *Manage Credentials* -> click on *Jenkins* link -> click on *Global credentials (unrestricted)* -> click on *Add Credentials* from the left pane.



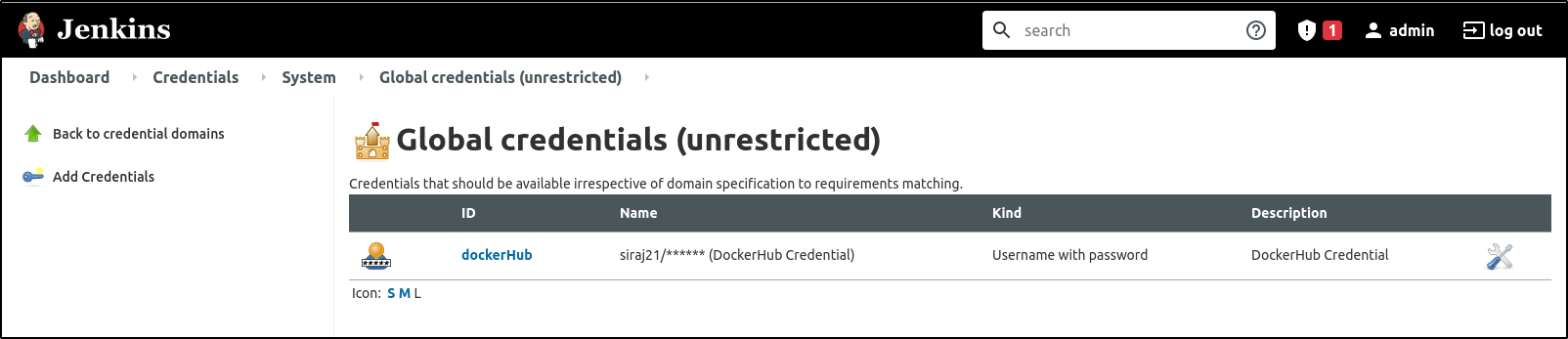
* Add the details as shown below

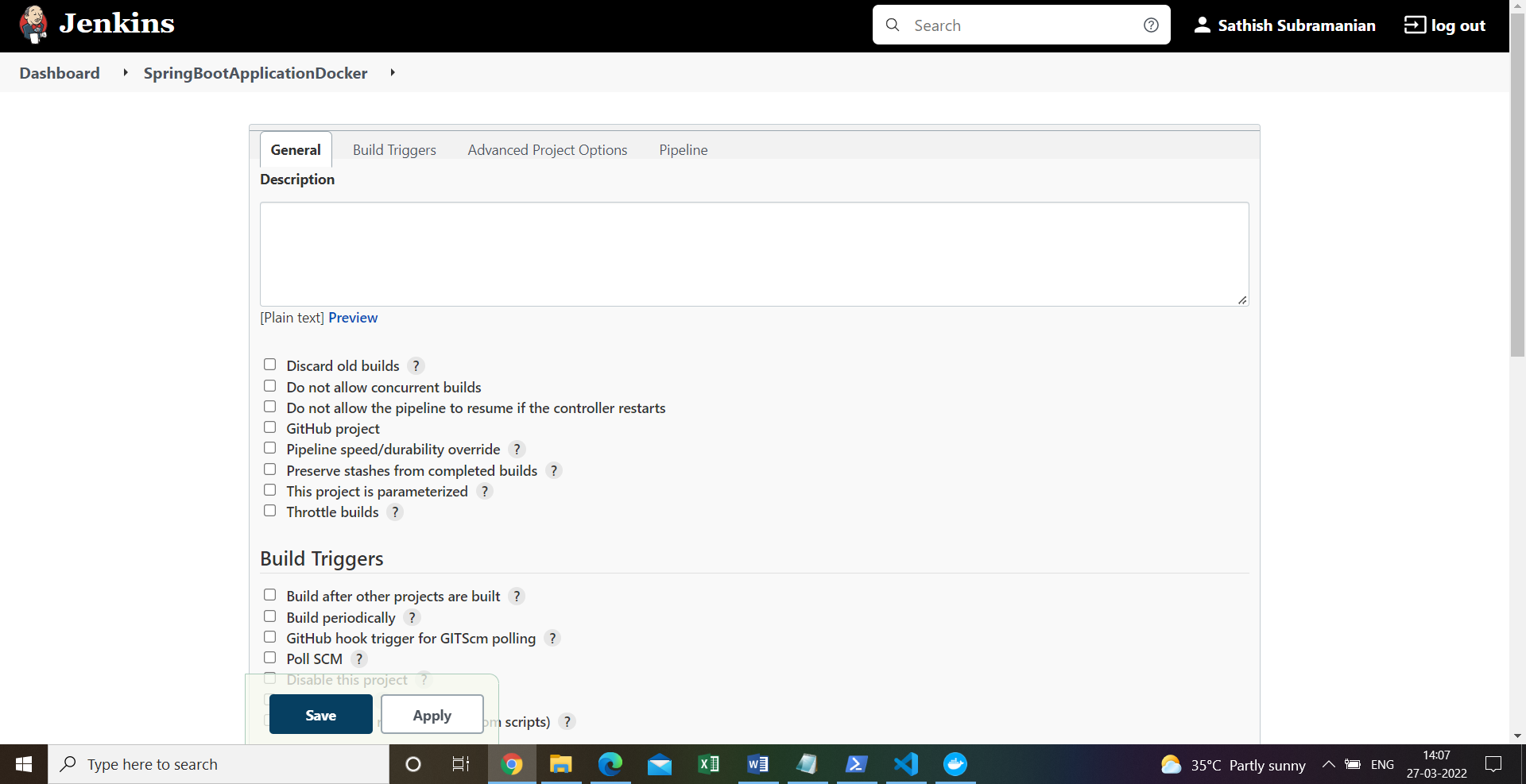
Username: *<Your\_DockerHub\_Username>*

Password: *<Your\_DockerHub\_Password>*



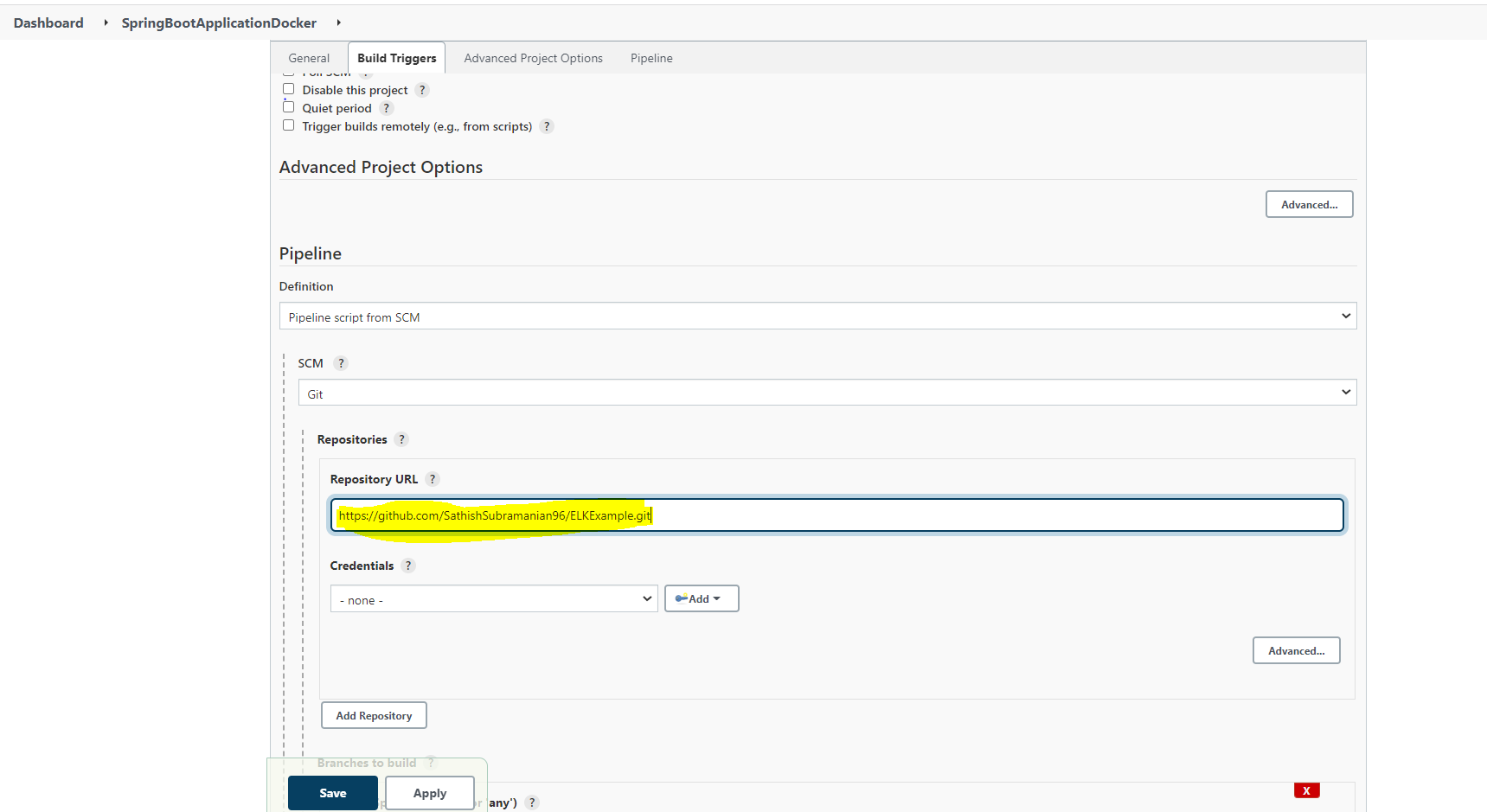
* You should now see the credentials saved as shown below



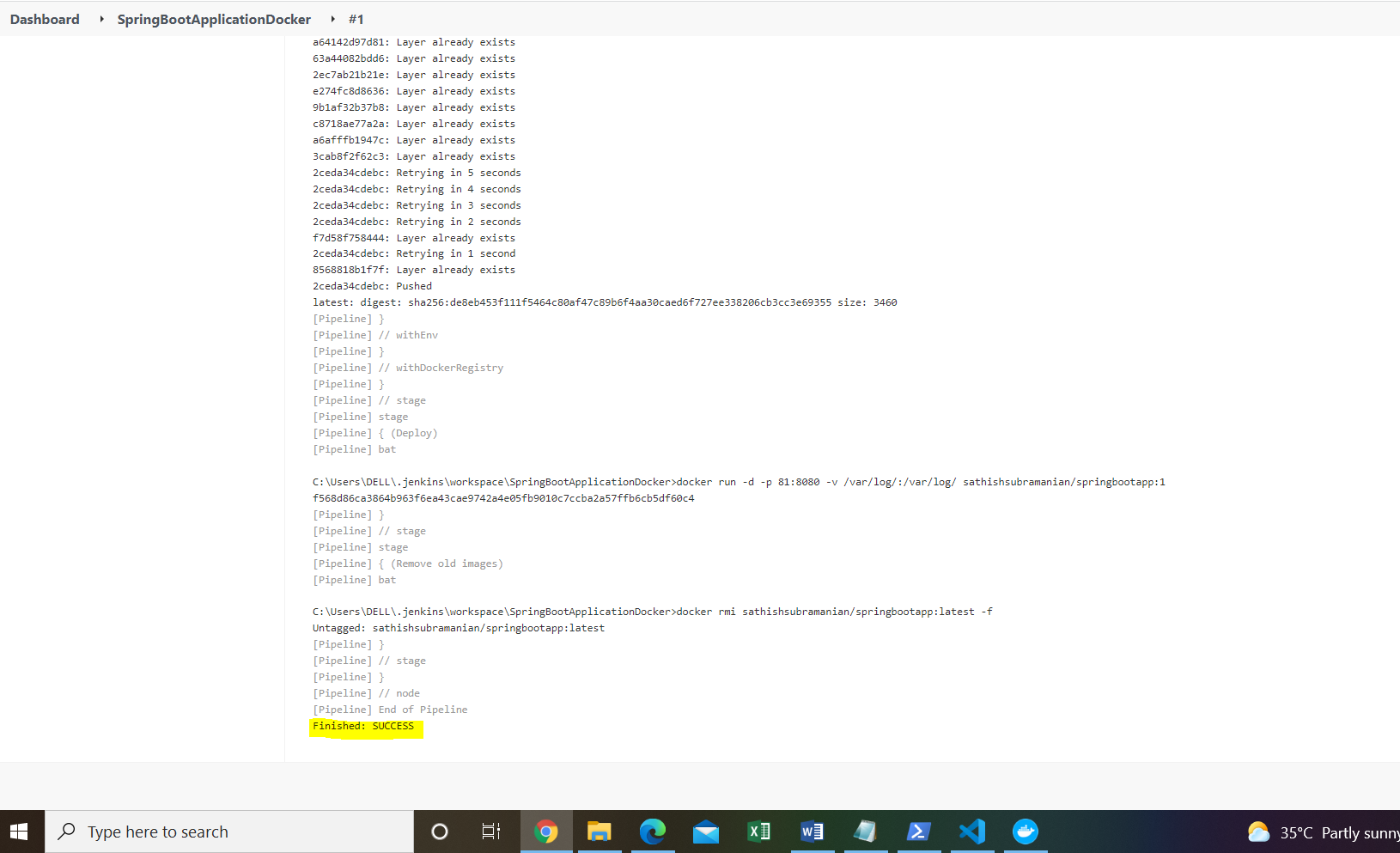
* Create a Jenkins pipeline job to fetch Jenkinsfile from the URL mentioned below. 
* You can either use the below git repository or Fork it in your Github account and use it

[***https://github.com/SathishSubramanian96/ELKExample.git***](https://github.com/SathishSubramanian96/ELKExample.git)

* Configure the job as shown in the screenshot below and then run the build.

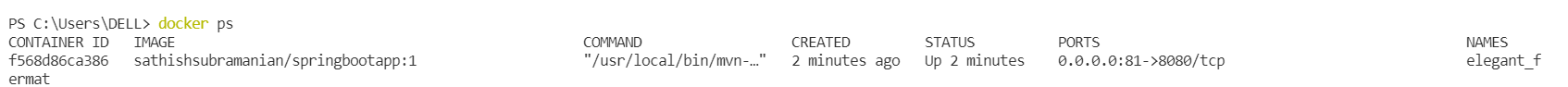


* Build the Jenkins job to deploy the Docker container on the Docker host.
* Jenkins pipeline will complete the build and the deployment process for the Spring Boot application



### We can see the Docker container deployed on the Docker host using the command:

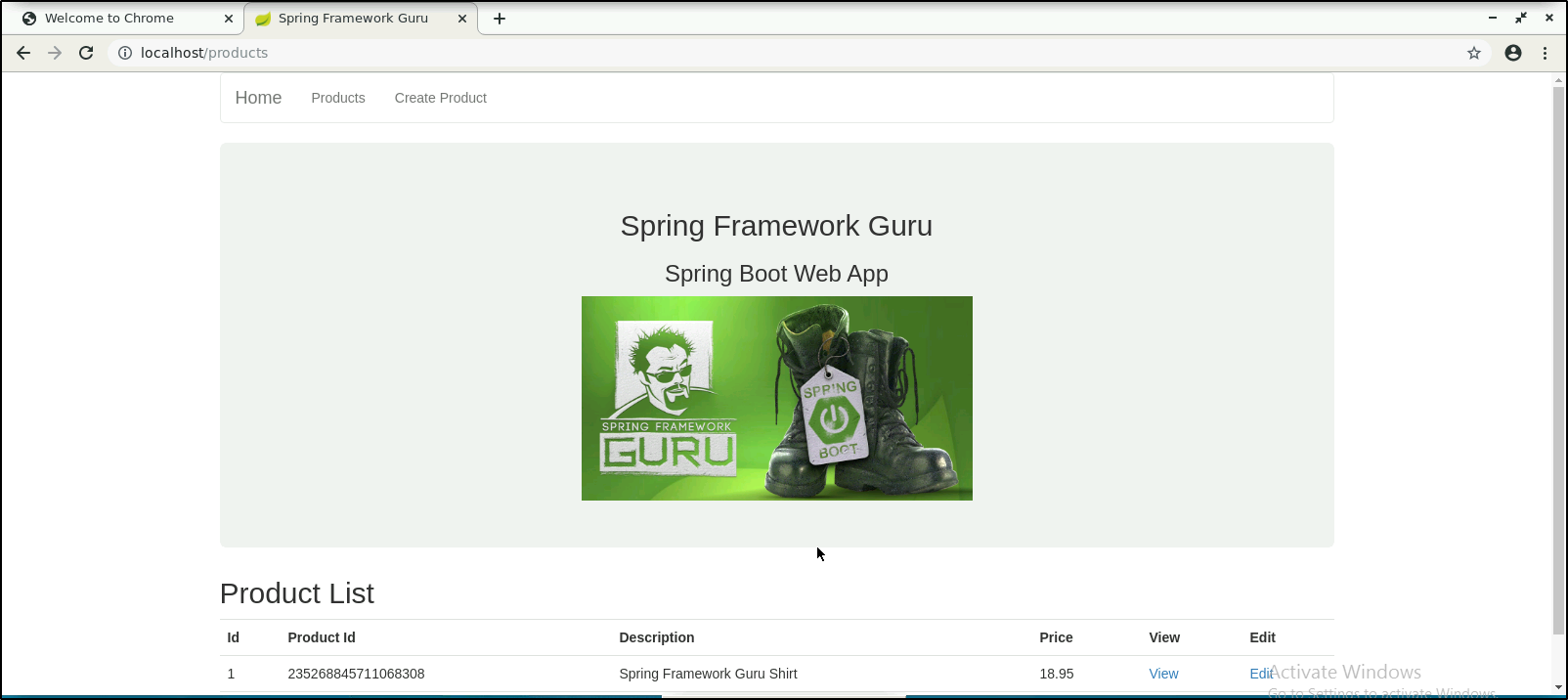
***docker ps | grep springbootapp***



**Step 3**: Run the Spring Boot application in jenkins and check the logs in Kibana

* Access the Spring Boot web application and perform some random activity so that the logs will be pushed to ELK stack.

***http://localhost:81***



* Check the logs pushed to ELK stack in Kibana.
* Navigate to the Kibana dashboard. Select *Index* *Management* from the navigation bar on the left. You can see the logs created.

