# Introduction

Generally while configuring the Selenium grid we need to host multiple virtual machines as nodes and we need to connect every single node with the hub. Also, when we setup a normal grid we want to download selenium server jar file and run that jar file on each computer in which we are going to setup the selenium grid.

This is costly and sometimes a time-consuming task for the testers. However, Docker helps us to solve cost concerning and time-consuming problems.

Docker was invented by Solomon Hykey and was launched into the software industry in 2013 March as an open-source tool. Now, developers and system engineers are using Docker for several reasons.

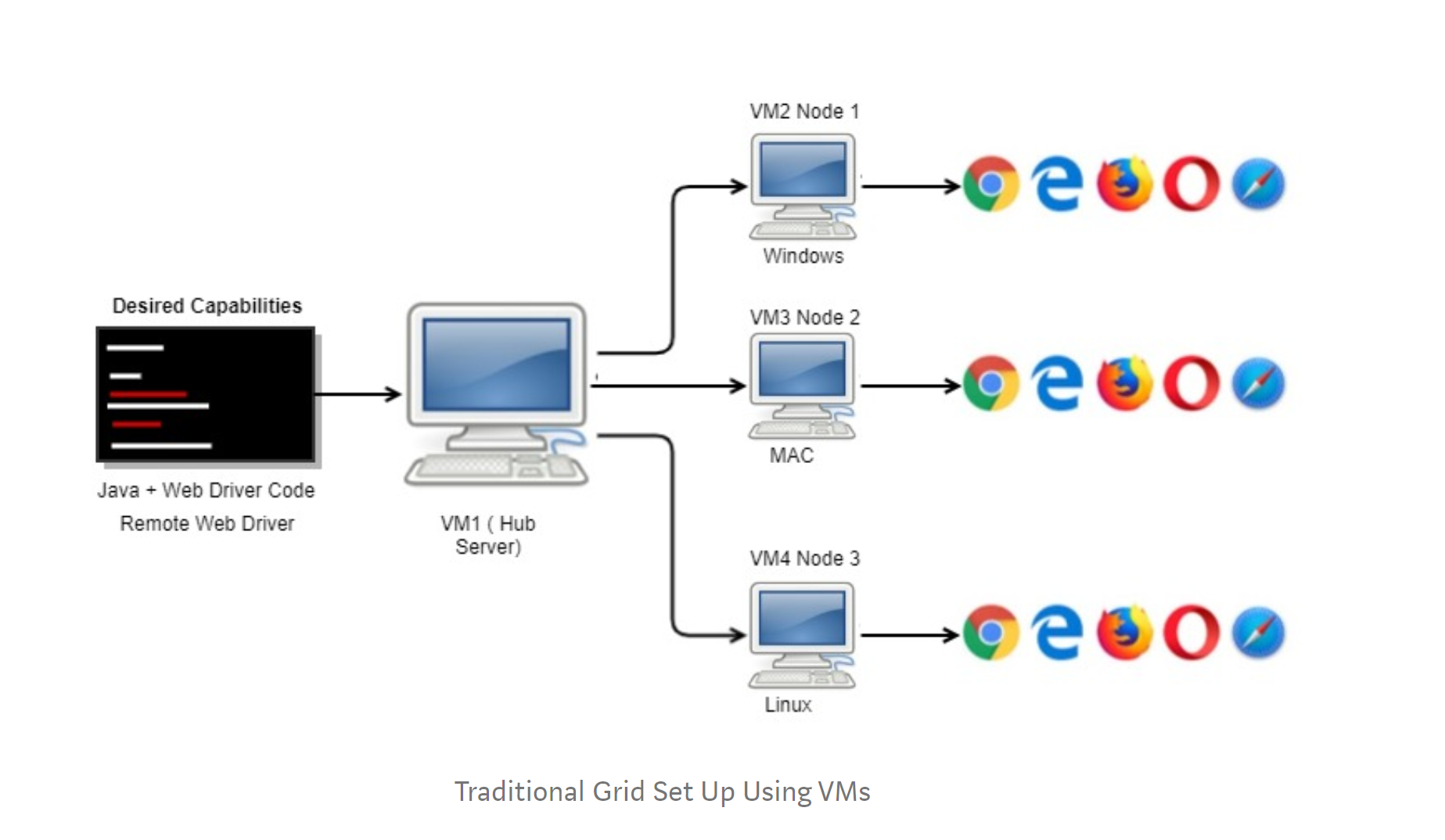
In this document, we are going to see how Docker can be leveraged for parallel testing.

# What is Docker?

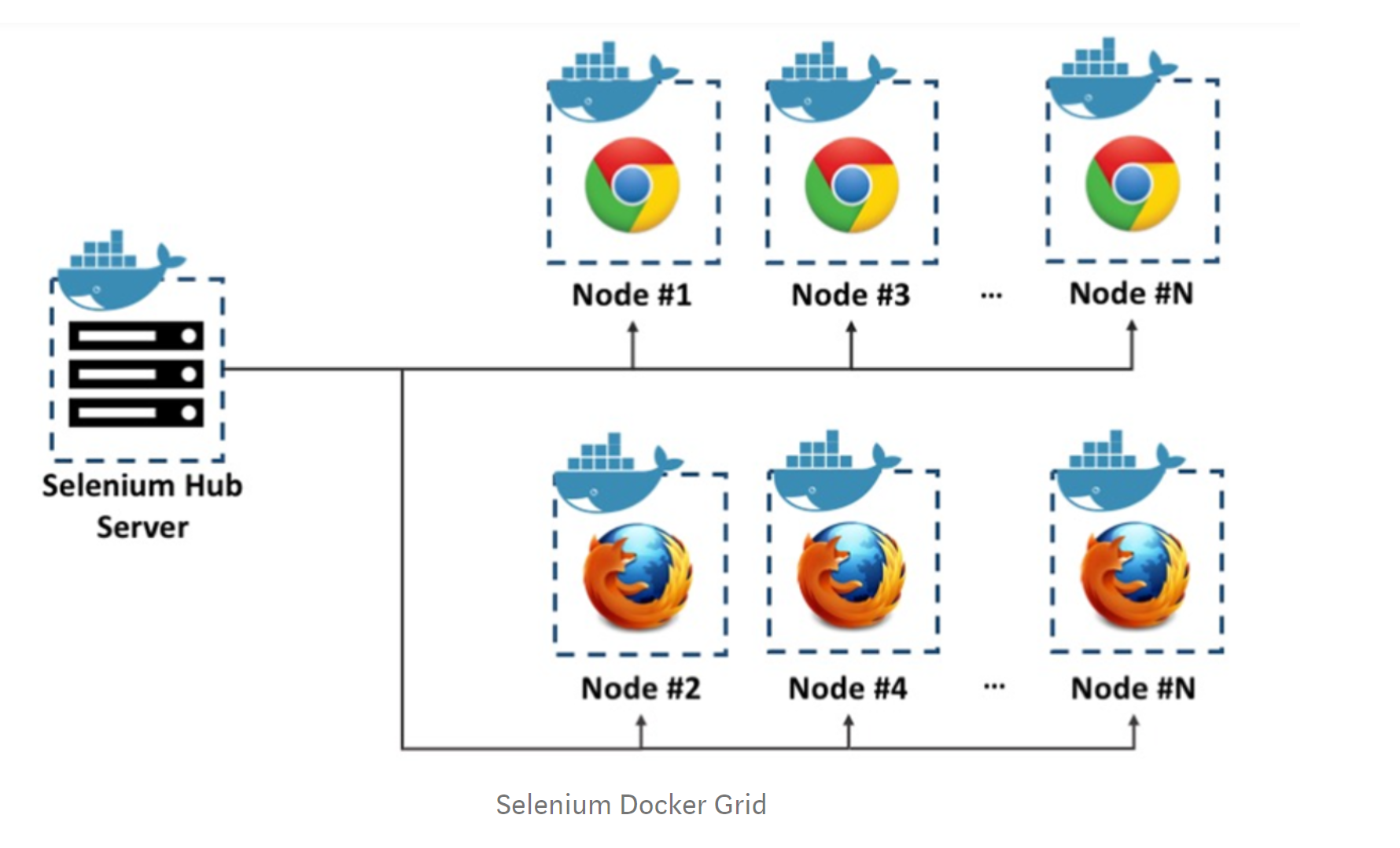
Docker is a tool intended to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code

# Docker Grid Setup

A traditional grid set up with physical machines looks like the below picture. It has a lot of overhead in terms of maintenance — the PC needs to be on all the time. It had to be time-shared amongst different teams. The tests performed flakier and maintaining the setup has always caused trouble and time consuming.



A Selenium Grid set up in Docker would like this.



So how different are these two pictures?

The Virtual Machines are replaced by Docker containers.

* In the common set up we had to set up every node with a selenium .jar file, and the browser drivers like chrome, Firefox, internet explorer, and also the hub, with the selenium server jar file. Then we had to register each node with the hub and put a Post-It Note on the PC which read “Do NOT switch off”. This is all replaced by a simple link command in Docker.
* We could spin up tons of nodes and get it linked up with the hub container
* We could tear them all down and fire them all up again as opposed to the traditional setup

Docker images and containers are very simple to understand; tons of resources are available on the internet these days.

So, in order to run a grid in a machine, we needed the images for the hub and the nodes, we then had to run them and link them. The following commands were in the order below

***$ docker pull selenium/hub  
$ docker pull selenium/node-chrome  
$ docker run -d -p 4444:4444 -name selenium-hub selenium/hub  
$ docker run -d -link selenium-hub:hub selenium/node-chrome***

We didn’t have to pull the image every time. we just had to use the last two commands for firing up selenium grid.

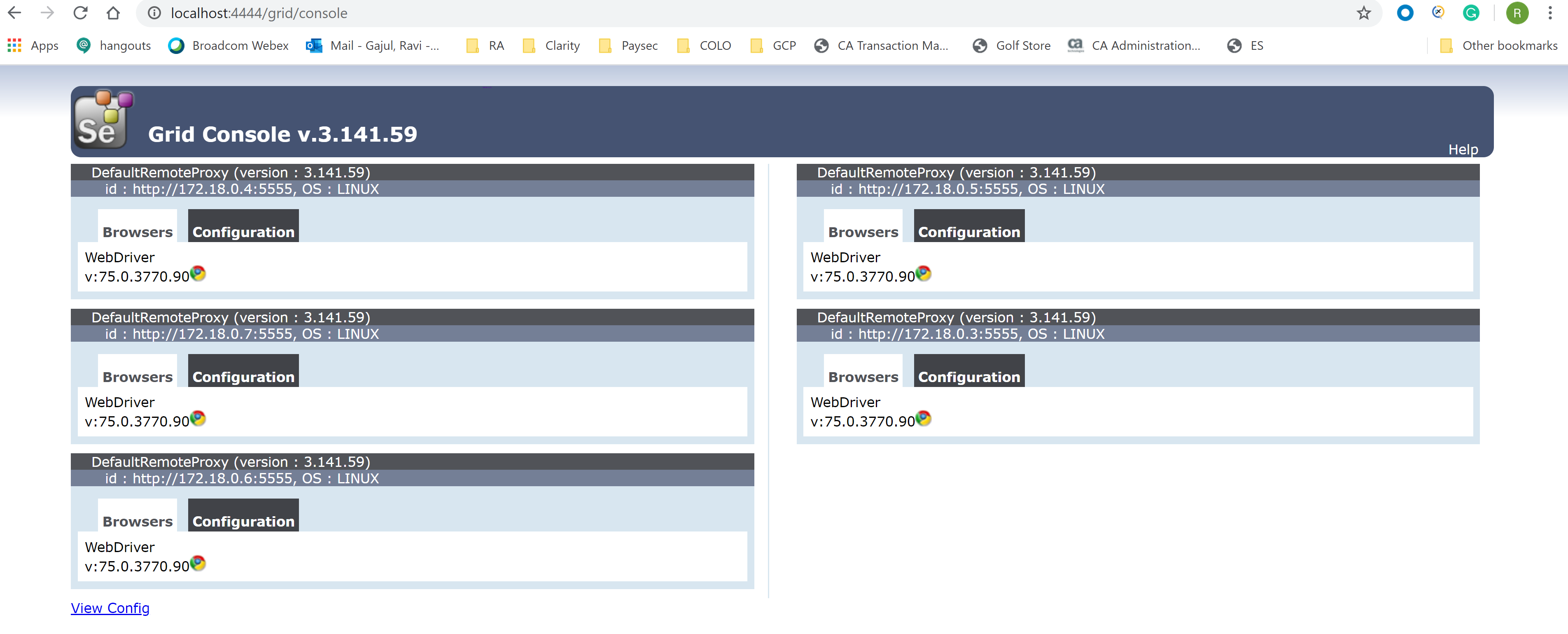
When you then try the **$ docker -ps** command, you should see your container running. Also, if you try and connect to http://localhost:4444/grid/console, you will notice that the hub is running accompanied by the node.

Alternatively we can simple run Docker-Compose command for the Grid set up and Hub and nodes registration.

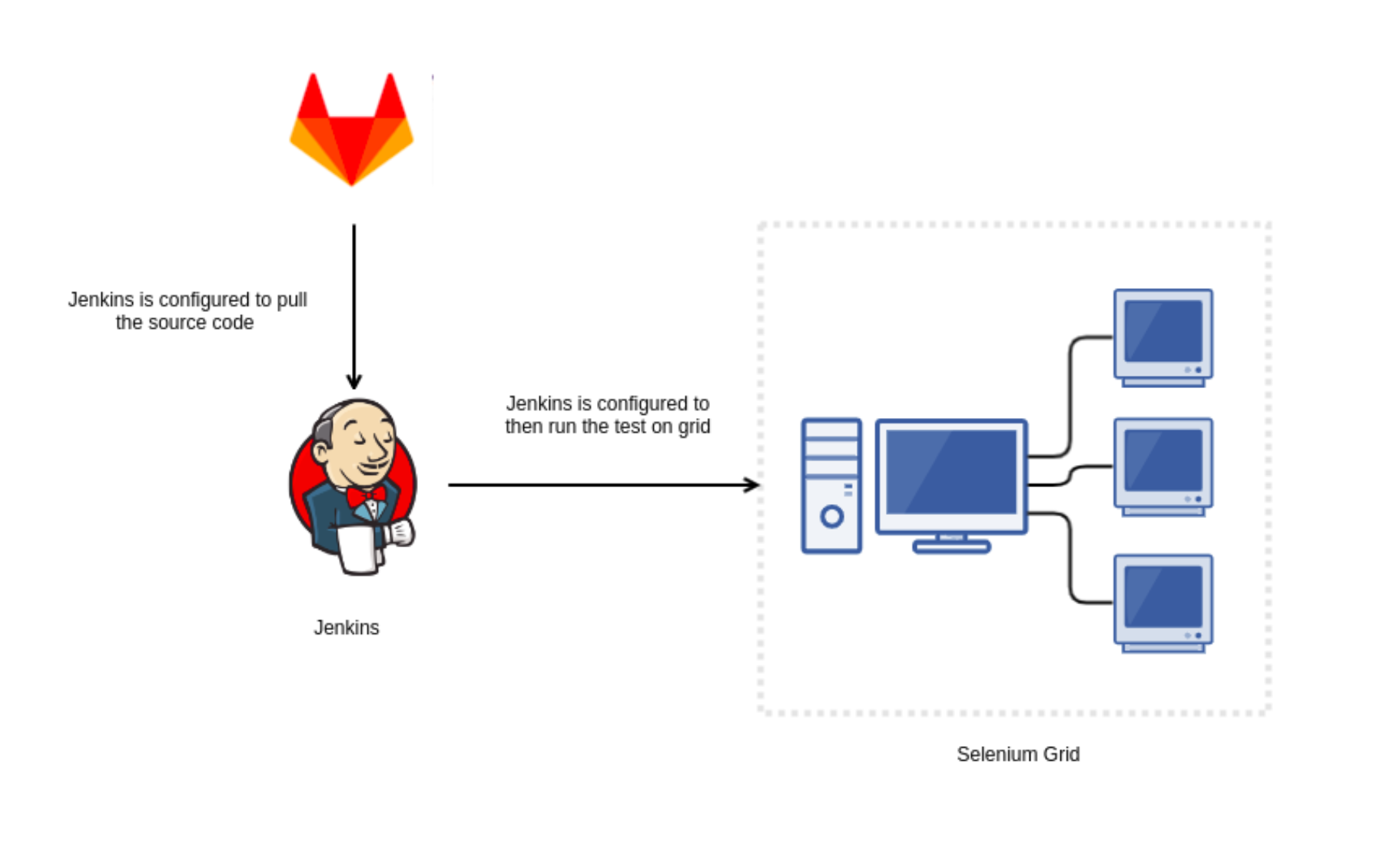
The below picture shows the grid console with 5 nodes fired up with simple docker command. The compose file is inserted for reference.

**$Docker-compose up –scale chrome =5**

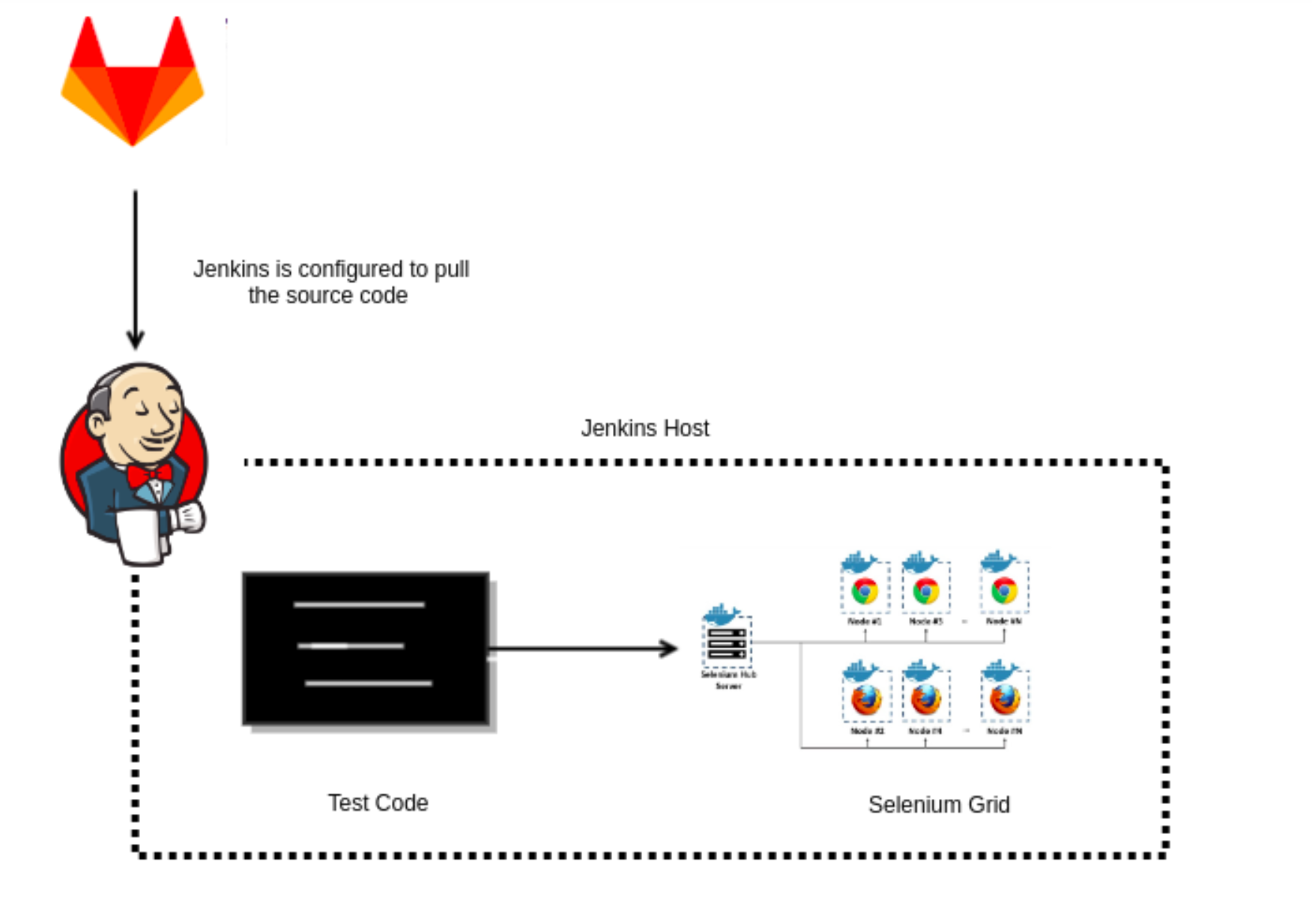
****



# Getting the tests to use the Docker grid

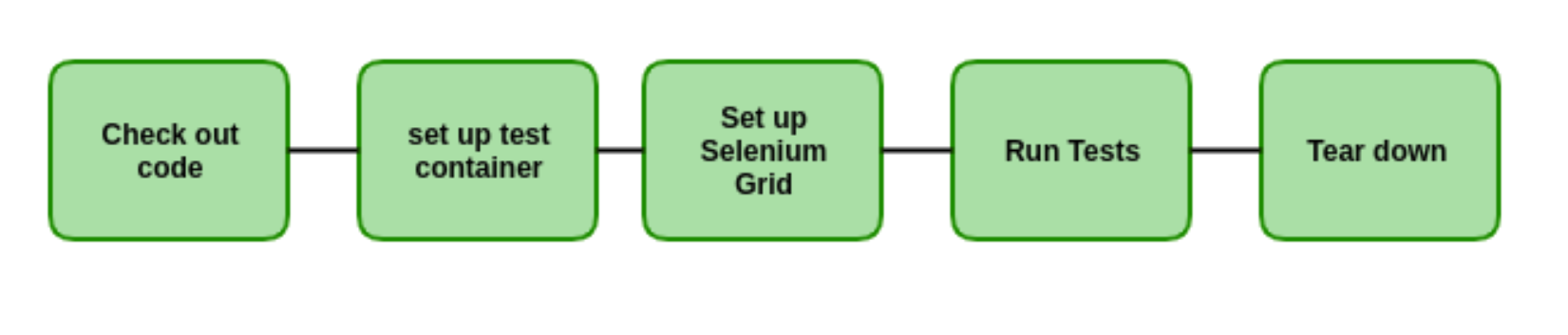
We ran the tests the usual way from the command line and they ran fine. In a traditional setup the tests would be configured like this:

The test code will be pulled into the Jenkins host or on a Linux box and the tests would be run from the Jenkins host, the code is set up to use the grid(this all is done by the pipeline script). When using the Docker grid everything remains the same except the old style grid. So the setup is now simplified to look like this:



As you can see a huge chunk of fat has been removed (from the physical selenium grid VM), and we could tear down the grid after a test run, and fire it up for every subsequent run. We did this with our tests too so that we had a clean host every time we ran the tests.

**The final pipeline looked like this :**



# Conclusion

1. No Virtual Machines are needed for Selenium Hub and Nodes Set up. This saves the cost involved in procuring the VMs. Docker is open source.
2. The number of nodes that are needed for test execution can be quickly scaled to any size depending on the host system space.
3. It takes only a few seconds to set up selenium Grid and register nodes to hub by using docker-compose.yml file.
4. The parallel test execution as you know has its own benefit for reducing the overall execution time of all test cases.

# References

<https://github.com/SeleniumHQ/docker-selenium>

<https://docs.docker.com/>

<https://www.selenium.dev/documentation/en/grid/components_of_a_grid/>

# Author’s biography

**Ravi Gajul** is a senior consultant with Deloitte Consulting, Hyderabad. He has 10.2 years of experience in testing, of which 6.5 years in Deloitte. He worked in public sector, ecommerce, and health care domains. He has played different roles, such as team member, test lead, and onshore coordinator for various projects in Deloitte. He has extensive experience in coordinating with the teams under offshore—on-site model and has worked with teams across competencies.