# Setting up an on-demand Selenium Grid with containers

[](https://medium.com/@moruganti_33642?source=post_page-----c43aceccd5d9--------------------------------)

[Mahesh Oruganti](https://medium.com/@moruganti_33642?source=post_page-----c43aceccd5d9--------------------------------)

[Follow](https://medium.com/m/signin?actionUrl=%2F_%2Fsubscribe%2Fuser%2F4af4db048f23&operation=register&redirect=https%3A%2F%2Ftechblog.dotdash.com%2Fsetting-up-a-selenium-grid-with-docker-containers-for-running-automation-tests-c43aceccd5d9&source=post_page-4af4db048f23----c43aceccd5d9---------------------follow_byline-----------)

[Apr 16, 2019](https://techblog.dotdash.com/setting-up-a-selenium-grid-with-docker-containers-for-running-automation-tests-c43aceccd5d9?source=post_page-----c43aceccd5d9--------------------------------) · 8 min read

*The objective of this post is to explain, how to set up or spin up chrome containers(selenium grid with chrome nodes containers) using docker containerization way to run Automation Tests.*

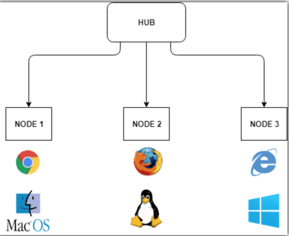
Image for post



**Note:** Note: if you are new to docker world, please go through a previous post ([docker-basics](https://medium.com/@moruganti_33642/understanding-docker-basics-be35dc939949)) to understand the basics of docker.

# ****What is Selenium Grid****

Image for post



**Selenium Grid**

***Selenium Grid:****Selenium Grid is a project created for executing tests on different browsers and operating systems. It has a simple architecture: A hub and one or more nodes. You run your test across the hub and hub distributes the test across different browsers.*

## ****Advantages of Selenium Grid:****

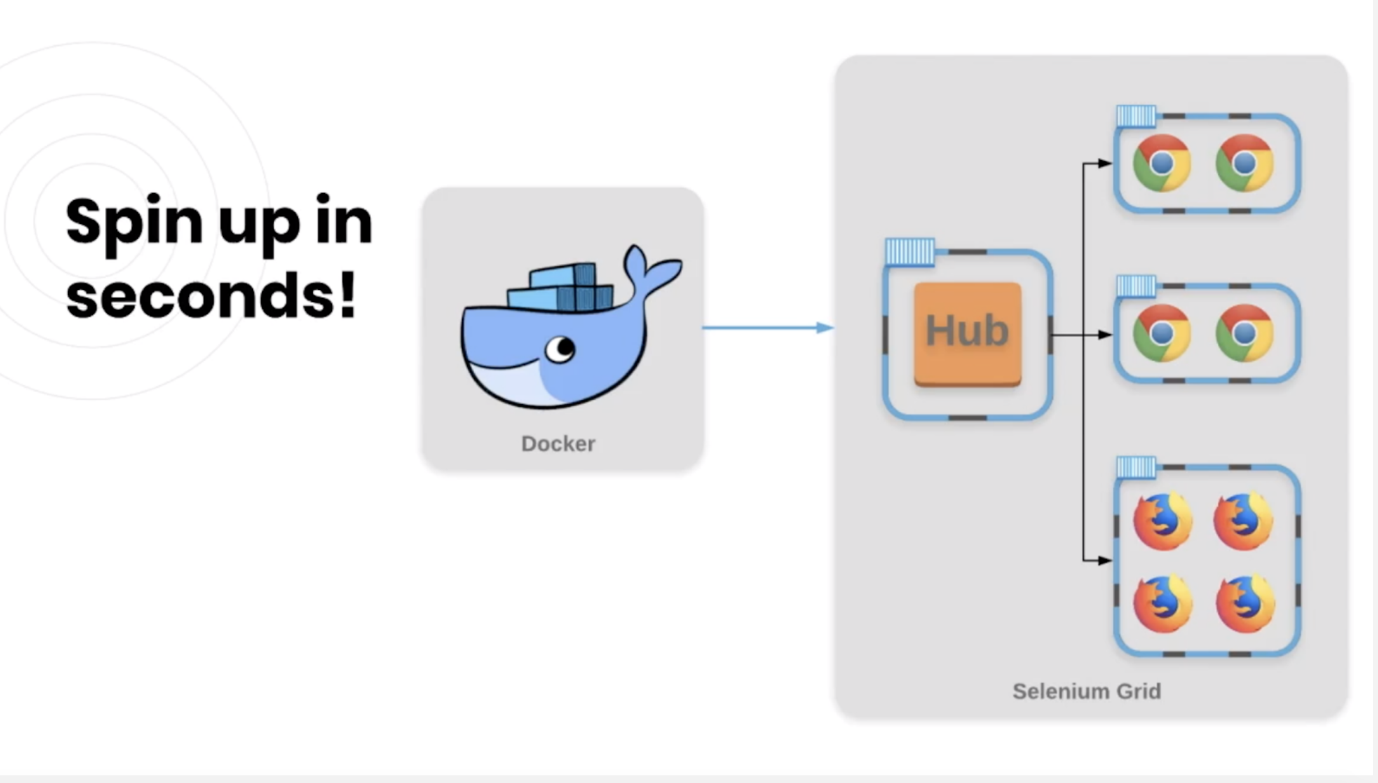
* Reduces execution time with distributed testing.
* Allows cross-browser and platform testing

## Challenges of Selenium Grid:

* Hard to configure (Installing Java, downloading selenium standalone server, installing related browsers and drivers, etc.).
* Hard to manage (testing different versions of different browsers).
* Resource usage

# ****Docker Selenium****

Image for post



* It uses the same architecture but each component, hub, and nodes, is a separate container.
* is highly used by test teams to run their tests in parallel on Selenium Grid

## What are the advantages of docker Selenium?

* We got rid of all the dependencies are mentioned above.
* Does not use system resources like VM’s so it’s lightweight.
* We can easily manage all the version complexity with docker registry.

The break down of steps required to set up selenium grid using docker is as following.

## STEP 1 : Pull docker images for Selenium.

As discussed in my previous blog, we need to pull Selenium docker images specific to chrome-debug from docker hub. Login to docker hub over here, <https://hub.docker.com/> and search for the below images and pull them into your computer/system. right now we have the below images available for chrome.

***selenium/hub:****Image for running a Grid Hub*

***selenium/node-chrome-debug:****Grid Node with Chrome installed and runs a VNC server, needs to be connected to a Grid Hub*

**Note: node-{Browser Name}** and **node-{Browser Name}-debug** cannot be used alone; **they must be connected to a hub**. **Debug versions include a VNC server** to visually debug the browser during the test.

## Check Images in local file system:

verify images exist pulled from docker hub by running command docker images. (make sure docker engine running in your computer).

***Note:****I have other images too in my system .Ignore other images showing up on below screenshot.*

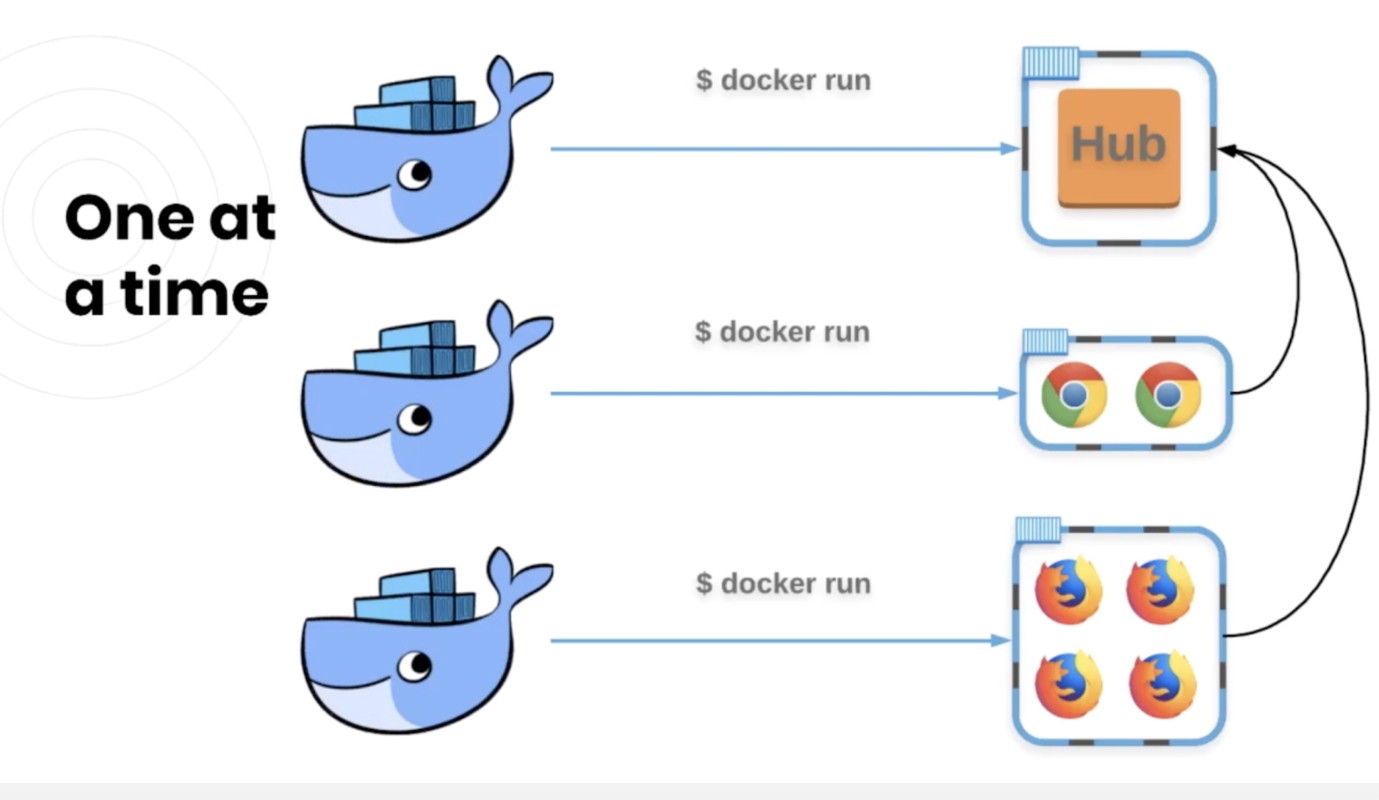
Image for post



## STEP 2 : Configure Images and Run containers (hub, chrome).

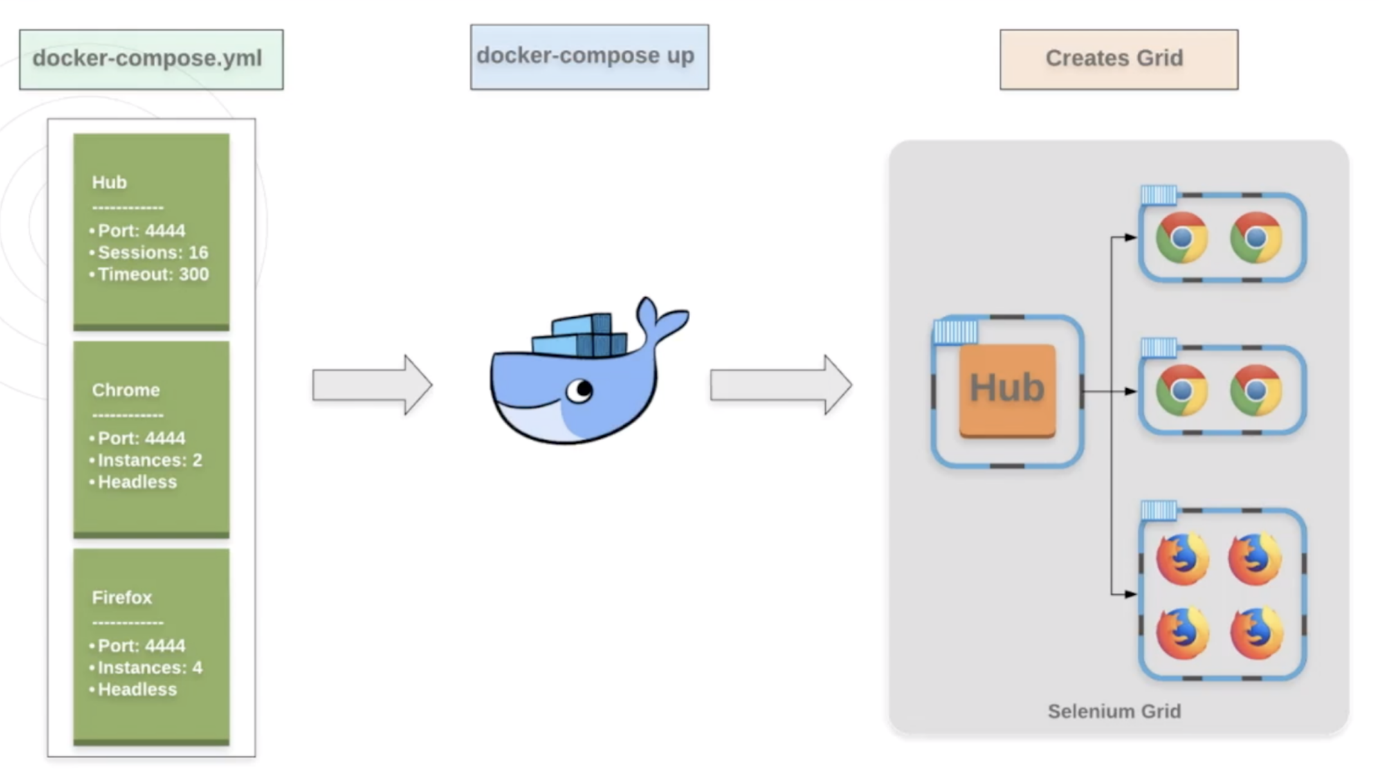
*We can start creating containers now that we have the images. You are able to run and start containers one at a time through the****docker run****command.*

Image for post



But we don’t want to create one container at a time!!. We’d rather create the containers and network them together in ONE command using docker compose and **docker-compose.yml** file**.**

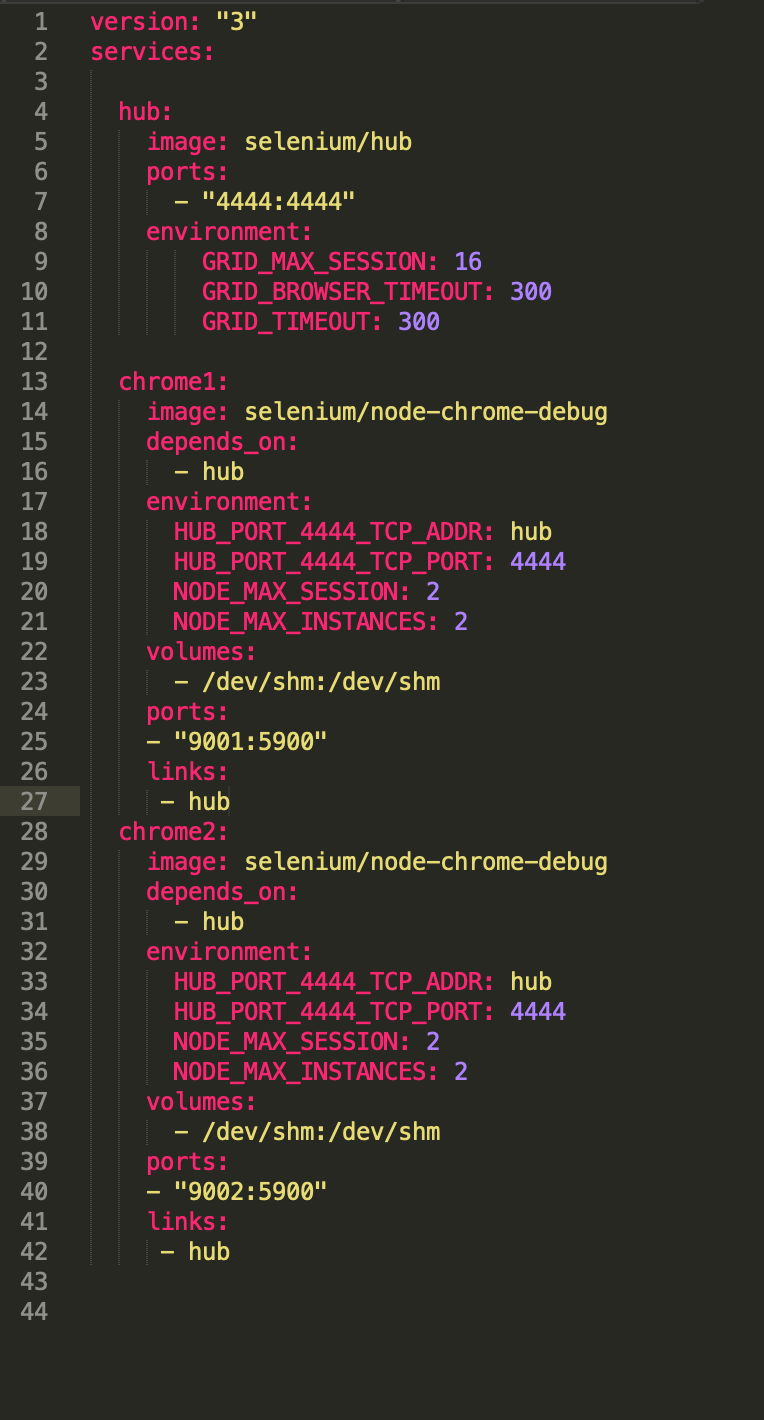
Image for post



**STEP 3 : Docker Compose and YML Overview:**

* **docker-compose** is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application’s services. Then, using a single command, you create and start all the services from your configuration.
* You define a single yml file with the services you want and then use the single command to create all of it at once. here is the sample **docker-compose.yml**file.

Image for post



Ex: **docker-compose.yml file**

* **Version**: “3” — as of this recording, this is the latest version of docker-compose files
* **Services** (or containers): — this is where we list the images we’ll be using and their configuration
* **image**: determine which container will be used to start container.
* **ports:**publish port with <HOST:CONTAINER> format.
* **volumes:**mount host path to service with <HOST:CONTAINER> format.
* **depends\_on**: containers start according to dependency order. In our example chrome and firefox will start after hub starts
* **environment**: specifies environment variables. You can see other environment variables of selenium images by using docker image inspect <IMAGE\_ID>

***Notes:****We’re calling the first container “selenium-hub” and saying that:  
It will use the selenium/hub image  
It will expose a port at 4444 (we’ll use this later to connect to the grid)  
It will allow 16 sessions or tests to run at one time  
And we’ll set some grid timeouts to the recommended 300 seconds*

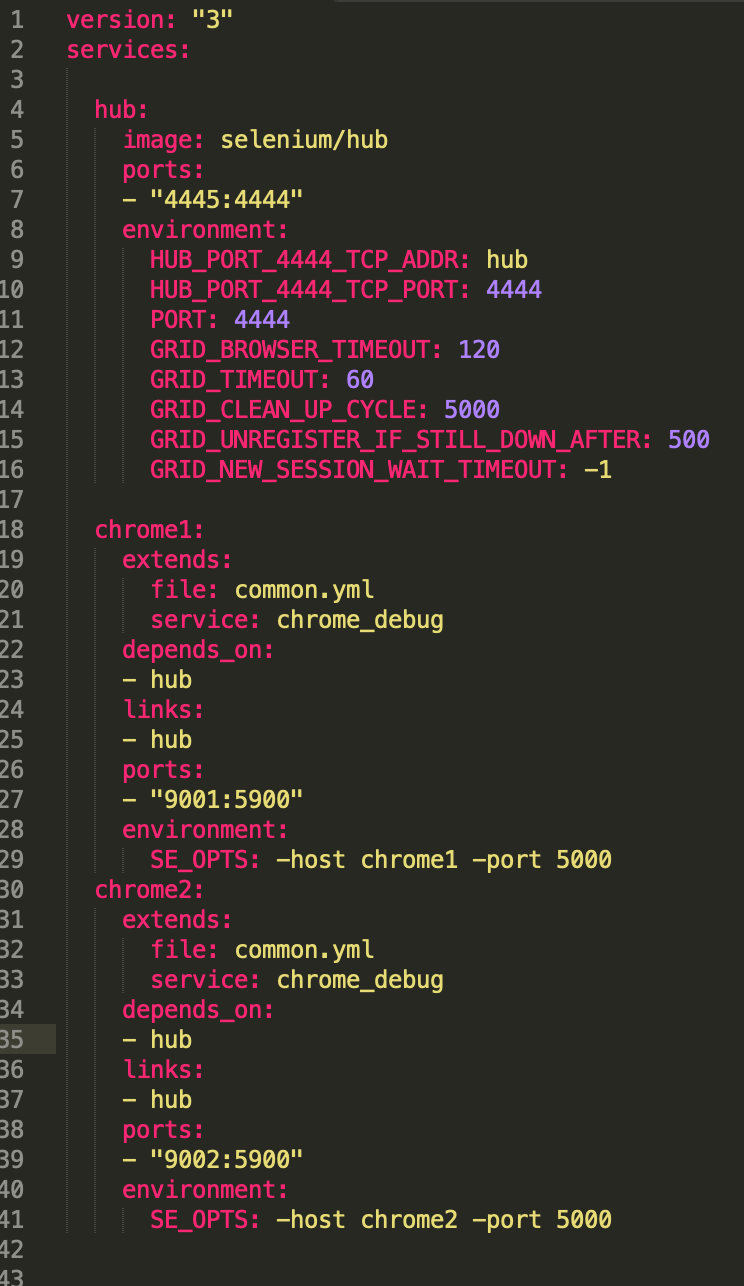
***Notes:****I added one Hub service and two chrome-debug node services. if you notice, the configuration(image, environment, volume, links, etc ..) for chrome-debug nodes is pretty much same except ports. so we can avoid repeated stuff over here by moving common configuration into another yml file and reference that file over here. so here is my final common.yml file and docker-compose.yml file looks like(added some extra useful configs but you can remove them as per your need).*

Image for post



**common.yml**

Image for post



**docker-compose.yml**

***Note:****port numbers should be different for each chrome-debug service. update to version:”2" if you see any issues.*

## ****STEP 4 :****Spin up Grid/Chrome Containers Using Docker Compose:

*There are two different ways****docker Compose****available.*

1. docker Compose comes by default with**docker engine installation.**
2. download or Install it from [**https://github.com/docker/compose/releases/**](https://github.com/docker/compose/releases/)

**Note:** if you are using mac/Linux you can run below command to install it from docker GitHub. update url with uname values (run uname -s and uname -m commands in your shell and copy paste values ). you can also change the version number(1.23.2) to latest if you want.

**sudo curl -L \”**[**https://github.com/docker/compose/releases/download/1.23.2/docker-compose-\$(uname**](https://github.com/docker/compose/releases/download/1.23.2/docker-compose-/$(uname)**-s)-\$(uname -m)\” -o /usr/local/bin/docker-compose**

Once the download is complete, apply executable permissions to the Compose binary.

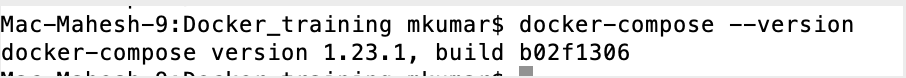
**sudo chmod +x /usr/local/bin/docker-compose**

## To Uninstall docker Compose:

sudo rm /usr/local/bin/docker-compose

**verify docker-compose installed:**

Image for post



**docker-compose**

*Once you install****docker-compose****you can directly run docker-compose.yml from the directory where the file exist , But you should have docker engine running in your computer.*

*you can also specify location of the docker-compose.yml file using -f flag.*

***1) docker-compose up –d (****by default**it looks for docker-compose.yml or docker-compose.yaml file in the current directory and run it. if not find, you will get below error message)*

ERROR:

Can’t find a suitable configuration file in this directory or any

parent. Are you in the right directory?

Supported filenames: docker-compose.yml, docker-compose.yaml

***2) docker-compose -f docker-compose.yml up -d***

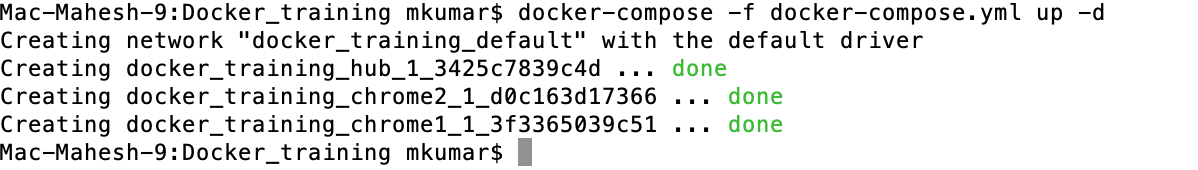
***3) docker-compose -f /root/home/test/docker-compose.yml up -d***

***Note:****If you want to run your services in the background, you can pass the -d flag for “detached” mode. -f is for file.*

You would be able to see services up and running once you spin up services(hub, chrome nodes). you can run **docker ps**command for verifying.

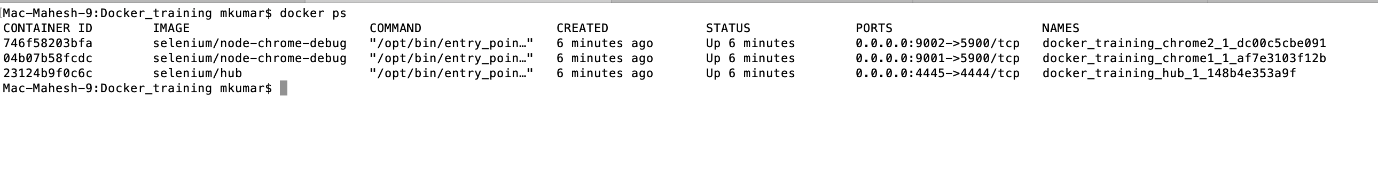
***Note:****containers names starts with current working directory name by default( in this case it is****docker\_training****). also we get create network bridge which is used for communicate from one containers/services to others.*

Image for post



**Image 1**

Image for post

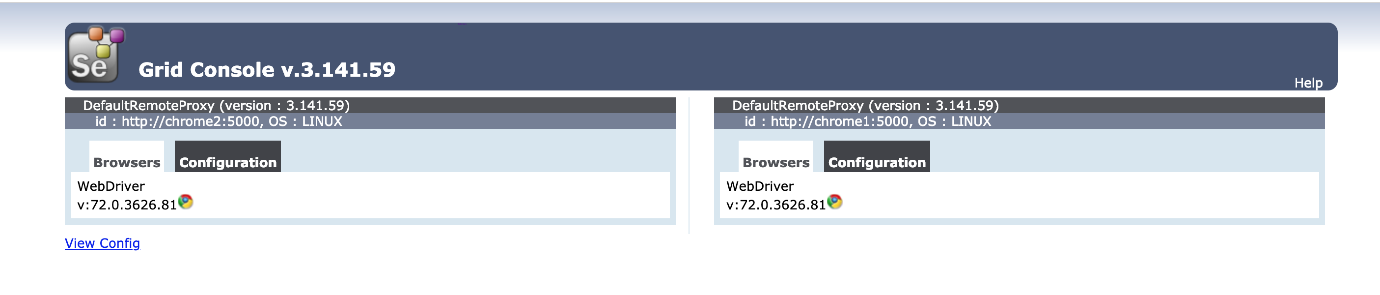


**Image 2**

**Verify Hub and Nodes Are Up:**

Just go to [http://localhost:4445/grid/console](http://localhost:4444/grid/console) and you would be able to see grid is up and running with two chrome debug nodes.

Image for post



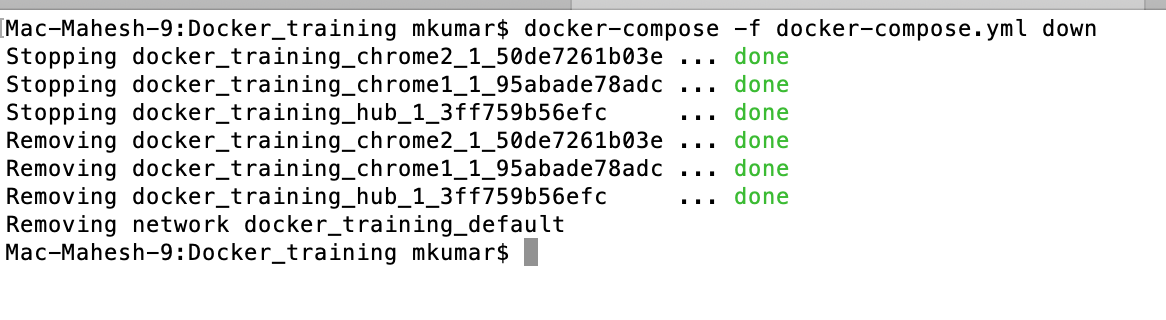
Now you can run your automation tests using target remote grid(localhost:4445). We are not going to discuss about test creation and execution in this post.

## ****Stop the Services/Containers:****

*To stop the services you could run below command. it will stop the services and remove them.*

***docker-compose -f docker-compose.yml down***

Image for post



**docker-compose.yml down**

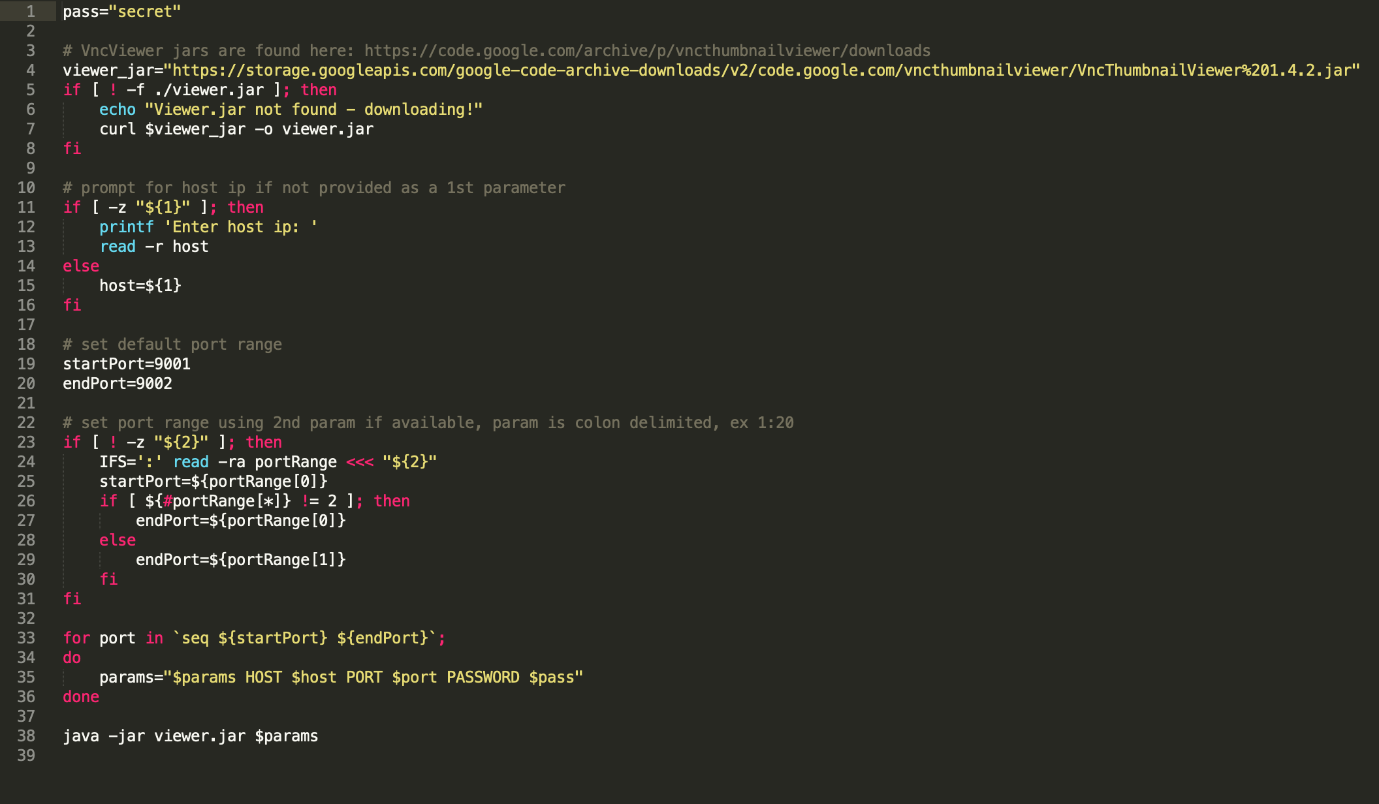
## ****STEP 5 : VNC to Containers.****

Since we ran **chrome-debug**containers and which comes with vnc server, we could connect to vnc server and watch/debug Automation Tests. To login to vnc server,

1. Get IP address of your machine/host (**ifconfig -a**if you are using Mac/Linux machines) where you spun up containers/services.
2. Create a file called view.sh and add the below source code. Run the **view.sh** in your command terminal and pass IP address of your host as input. It will bring up GUI of chrome containers just created. please refer below screen shot **Containers**. you can also use various ways to log into vnc server. (for example you can download vnc client and add server/port configs and login).
3. If you are Mac user and if you have installed “**Connect to Server**”, you can Vnc to server using IP Address and container Port numbers(in this case 9001, 9002)
4. Download View.sh source code from github <https://github.com/MaheshOruganti/docker-selenium/blob/master/view.sh>

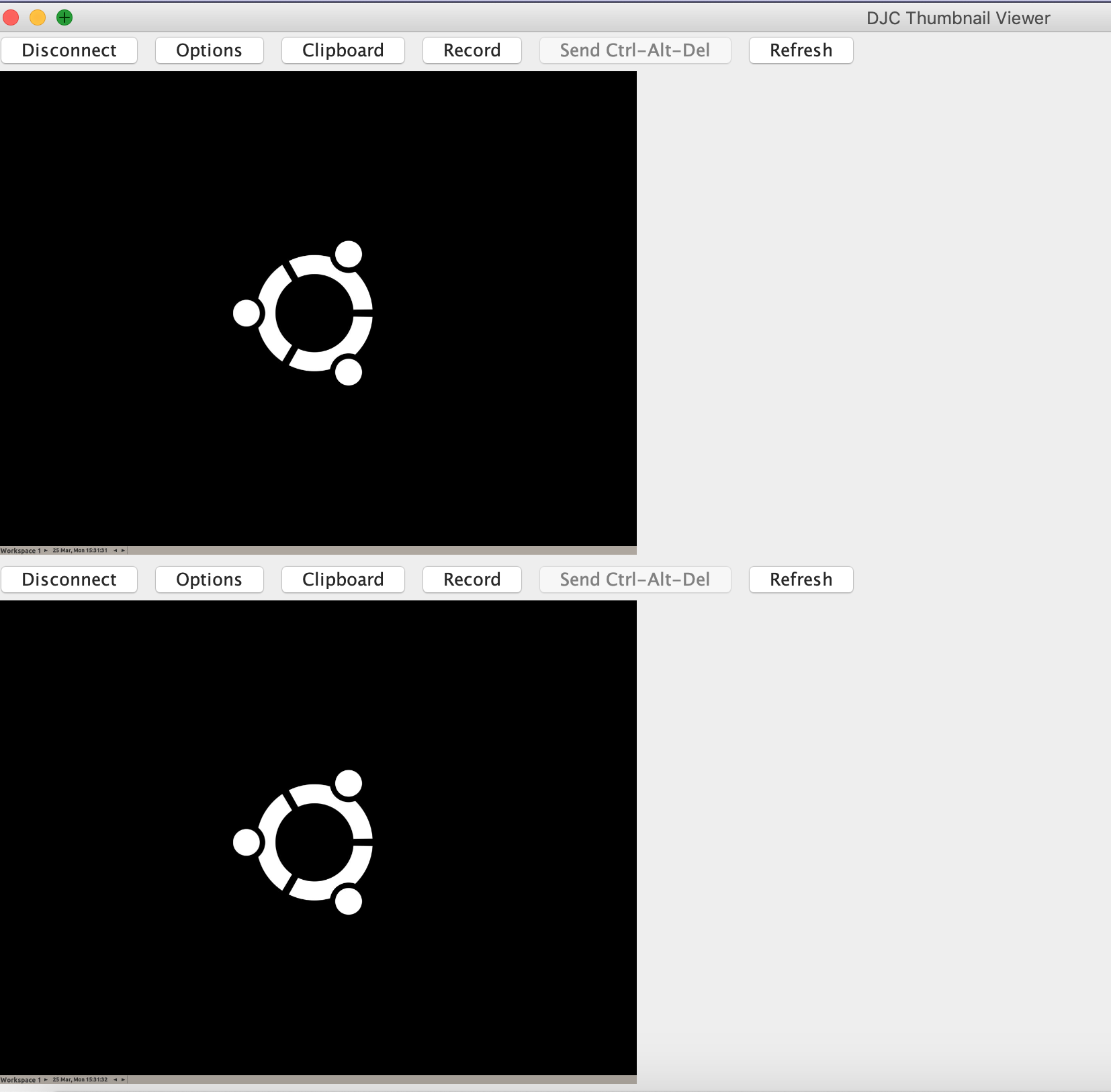
**vnc://<IP Address of your host>:<container Port number>**

Image for post



View.sh

Image for post



Containers

***Notes:****The port numbers I added range from 9001 to 9002 because I have spin up only two chrome-debug nodes with two different port numbers (check docker-compose.yml file). but you can add more chrome-node services(up to 30) as per your requirements and update port numbers range over here.*

## STEP 6 : Login to container.

*Use****docker******ps****to get the name of the existing****container****.*

*Use the command****docker****exec -it <****container****name> /bin/bash to get a bash shell in the****container****.*

Voila we have a working selenium docker instance and chrome debug containers for running automation tests. If you want to spin up firefox/chrome containers, the process should be the same. Also you can spin up both firefox and chrome nodes with grid combination.

Using this approach we could save lot of time, cost effective, accurate test results.

*At Dotdash we run around 8000+ Automation Tests across multiple properties and we run regression tests multiple times everyday against the test infrastructure, where we have selenium grid connected to multiple grid nodes(AWS Ec2 instances). The huge increase in our regression tests mandate we scale our test infrastructure on demand. By using containerized selenium ,we end up running tests on few Jenkins slave machines (we run automated tests using Jenkins job). This allowed us to get rid of the fixed and difficult to scale test infrastructure and find efficiency as well as control the costs of running the AFT’s.*

## References:

[SeleniumHQ/docker-selenium](https://github.com/SeleniumHQ/docker-selenium)

[Overview of docker-compose](https://docs.docker.com/compose/reference/overview/)