ACS SOLUTIONS

Whitepaper

Scale-up Selenium Test Execution with Docker

A Case Study

Table of Contents

1 Abstract

2 Introduction

3 Software and Hardware Requirements

4 Steps to configure Selenium Grid using Docker

5 Benefits of using Docker with Selenium Grid

6 Case Study

7 Conclusion

8 Contributors

Abstract

In Agile software development, test automation has evolved as a critical component of the continuous delivery of software as better automation test coverage gives stakeholders more accurate information on how stable high-value business features of their application and business are. This will, in turn, help them in making more informed decisions at the business level. So, getting feedback early in the cycle is critical.

In addition, teams must look forwards to scale up test execution quickly with Docker, which would minimize the installation and environment setting effort.

This paper also covers:

1. Benefits of using Docker for test automation
2. Detailed instructions for setting up Docker on local or dedicated automation servers to improve scalability of automation infrastructure.
3. Frequent challenges encountered and how to overcome them
4. A case study where it was implemented, and the results achieved
5. How Docker can be implemented with various cloud providers (AWS, Azure, GCP)?

The major benefits of using Docker include:

1. Parallel execution to share early feedback
2. Cross-browser test coverage
3. Easy maintenance of infrastructure
4. Open-source tech stack for test automation

And a few more that will be discussed in this paper.

Introduction

During Test Automation, test coverage based on the number of automation test cases being executed must be increased while focusing on decreasing the time required to run test automation suites. As the automation suites grow, the time to run them increases. Automation frameworks need to have the ability to run multiple test cases or test suites in parallel. In such a scenario, parallel execution using a selenium grid helps.

Here tests will run independently of one another in different nodes. More infrastructure/servers are needed to handle execution. In the selenium grid, one can have one hub and many nodes. The hub will coordinate with nodes and run tests cases on them. In the traditional grid set up virtual machines are used as hubs and nodes.

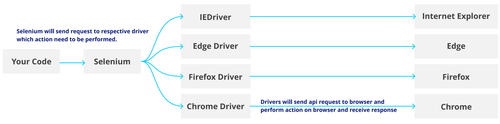
Software and Hardware Requirements

1. Docker - Software for deploying docker containers and managing infrastructure.
2. Servers – One or more servers based on the elasticity that is needed for infrastructure.
3. All the software that is used for selenium automation like IDE for coding and programming language libraries like Java/Python etc.
4. Jenkins for continuous integration.
5. Git for source control
6. Automation Framework, which is developed to support selenium grid executions.

Selenium Grid

Selenium Grid is a hub-nodes architecture proxy server. The server will act as the Hub. It facilitates parallel execution of test scripts where one can configure different OS or browsers. Those are called nodes. Tests contact the hub to obtain access to remote browser instances. The Hub takes the requests from test scripts and then sends them to nodes based on the browser configuration.

The below image shows how selenium works:



Below image shows how the selenium grid works:

For deploying a selenium grid, Docker needs to be pre-installed on a server. There are multiple ways to deploy based on the requirement. For example, if only one server is used and smaller number of test cases need to run in parallel, the entire selenium grid can be deployed in one server.

If the requirement is to run multiple tests in parallel and there is a need for scalable selenium grid infrastructure, one can distribute docker containers across multiple servers. At the moment the docker swarm is being used by us to maintain the grid setup.

Steps to configure Selenium Grid using Docker

Step1: Pull the following docker images:

1. Run the below command to pull the hub image
   $ Docker pull selenium/hub
2. Run the below command to pull the node container that would have a Chrome browser
   $ Docker pull selenium/node-chrome
3. Run the below command to pull the node container that would have a Firefox browser
   $ Docker pull selenium/node-firefox

Note: By default, Docker installs the latest available browser versions.

4. Run the below command to verify the images
   $ docker images

```
REPOSITORY                      TAG                 IMAGE ID       CREATED       SIZE
selenium/standalone-firefox     4.0.0-20211013      fef033580704   2 weeks ago   1.08GB
selenium/standalone-chrome      4.0.0-20211013      201110c834d3   2 weeks ago   1.19GB
selenium/hub                    <none>              ad49ff5fe5cc   2 weeks ago   364MB
```

Step 2: Configure the infrastructure

Introduction to docker swarm:

A Docker Swarm is a group of either physical or virtual machines running the Docker application and configured to join in a cluster. Once a group of machines has been clustered together, the usual Docker commands can be executed, but the machines in the cluster will now carry them out. A swarm manager controls the cluster's activities, and machines that have joined the cluster are referred to as nodes.

One of the key benefits associated with the operation of a docker swarm is the high level of availability offered for applications. In a docker swarm, there are typically several worker nodes and at least one manager node responsible for handling the worker nodes' resources efficiently and ensuring that the cluster operates efficiently.

In automation, one can use the same machine as manager and node or have as many nodes and managers as per need. So for deploying selenium grid a docker-compose file needs to be built where all the configurations for docker images have to be specified. Below is the format of the sample docker file and an explanation of the parameters.

```
version: '3.7'

services:
  selenium-hub:
    image: selenium/hub:4.0.0-20211013
    ports:
      - "4442:4442"
      - "4443:4443"
      - "4444:4444"
    deploy:
      mode: replicated
      replicas: 1
      placement:
        constraints:
          - node.role == manager
  chrome:
    image: selenium/node-chrome:4.0.0-20211013
    shm_size: 2gb
    volumes:
      - /dev/shm:/dev/shm
    environment:
      - SE_EVENT_BUS_HOST=selenium-hub
      - SE_EVENT_BUS_PUBLISH_PORT=4442
      - SE_EVENT_BUS_SUBSCRIBE_PORT=4443
      - SE_NODE_SESSION_TIMEOUT=600
      - SCREEN_WIDTH=1920
      - SCREEN_HEIGHT=1080
    deploy:
      replicas: 6
    entrypoint: bash -c 'SE_OPTS="--host $$HOSTNAME" /opt/bin/entry_point.sh'
```

This is the format of the docker-compose file. Here there are multiple services for selenium hub and selenium nodes. In the above example, there is one selenium hub service and one selenium chrome node service.

Image: Teams must specify the name of the Image the service needs. Here selenium/hub is the Image for deploying the selenium hub container.

Ports: They can map ports from one container to another and containers to host as well.

Volumes: They can map host volumes to container volumes to access files from the host inside the container or to access files from the container to the host.

Environment: They can specify various environment parameters required for deploying a grid.

Replicas: Under chrome service, they can mention how many replicas they need to deploy. They can create as many replicas as they want based on machine configuration.

Placement: They can specify the placement of a particular service. For example, in the above docker-compose file, they are specifying placement for selenium hub in manager. If they do not specify it, the swarm will deploy hub in any swarm node or manager.

1.1 On Local Machine

The entire selenium grid infrastructure can be deployed in a single machine. This can be done by using the docker swarm and the above docker-compose file. Here a single machine will act as both manager and node. Below are the steps for the setup.

1. First, one needs to initialize the docker swarm by using the below command. This command will initialize the docker swarm, and a selenium gid can be deployed into the swarm.
   docker swarm init

2. Now the stack needs to be deployed by using the below command. In the below command, one needs to specify the path of compose file and the name they want to give to the stack.
   docker stack deploy --compose-file=docker-compose.yml selenium

3. Now they can verify if the stack is deployed or not by using the below command. This will display all deployed services. In this case, selenium hub and selenium chrome nodes. It will display the count of replicas of a particular service.
   docker service ls

1.2 On Multiple Servers

The entire selenium grid infrastructure can be deployed in a single machine. This can be done by using the docker swarm and the above docker-compose file. Here a single machine will act as both manager and node. Below are the steps for the setup.

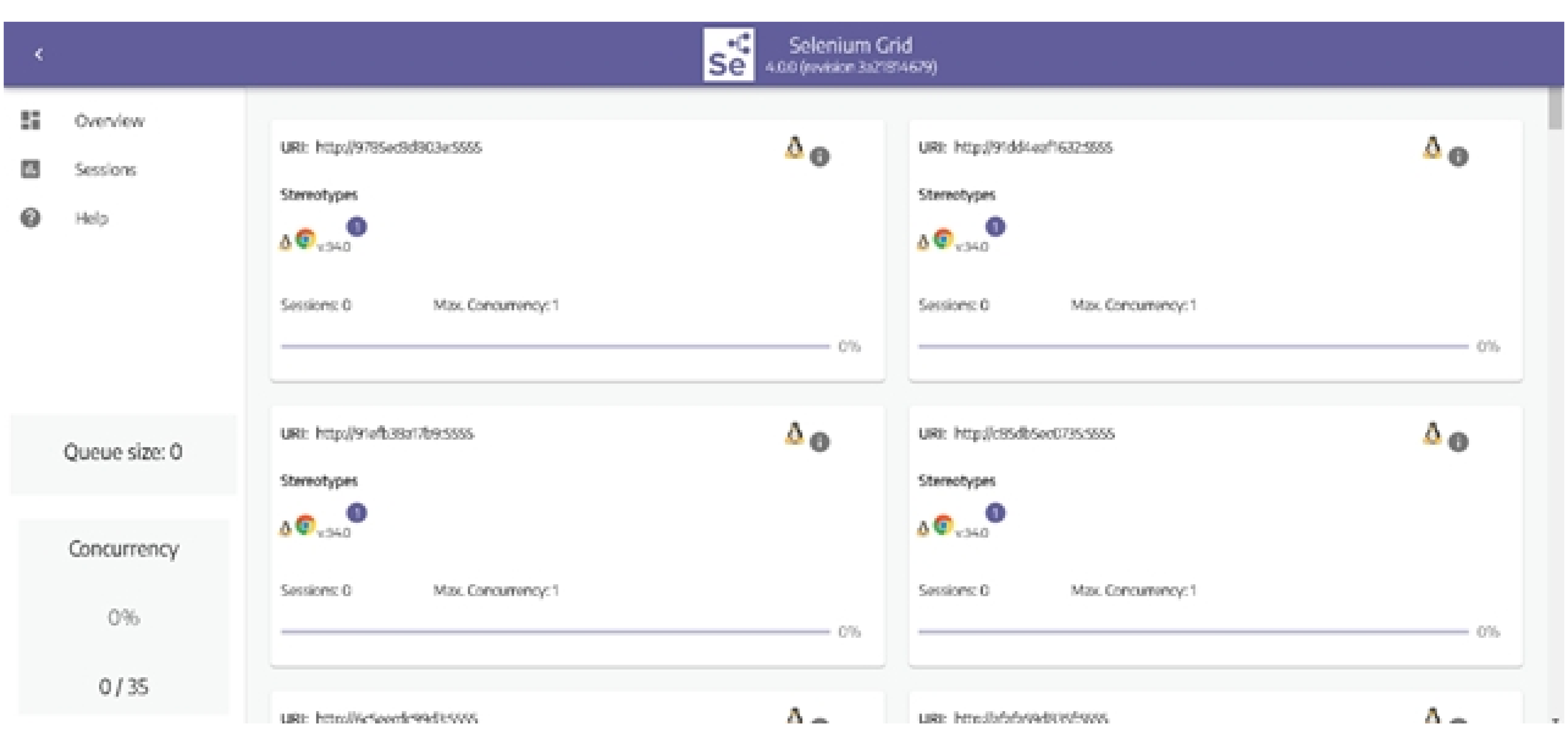
1. First, one needs to initialize the docker swarm by using the below command. This command will initialize the docker swarm, and a selenium gid can be deployed into the swarm.
   docker swarm init

2. Now the stack needs to be deployed by using the below command. In the below command, one needs to specify the path of compose file and the name they want to give to the stack.
   docker stack deploy --compose-file=docker-compose.yml selenium

3. Now they can verify if the stack is deployed or not by using the below command. This will display all deployed services. In this case, selenium hub and selenium chrome nodes. It will display the count of replicas of a particular service.
   docker service ls

Step3: After deploying the selenium grid, the grid can be verified by opening the below URL in the browser.

http://<swarm-manager-ip>:4444



Step 4: Run the scripts.

One can implement parallel execution at the framework level and run multiple tests in parallel by creating a Remote Web Driver pointing to the docker swarm manager.

Step 5: To remove the entire selenium grid stack, one must run the below command.

docker stack rm selenium

Benefits of using Docker with Selenium Grid

If Docker is used, a selenium grid can be implemented using docker containers, and it can be done in a single machine, or the infrastructure and parallel execution count can be increased easily. One of the biggest benefits of Docker is its capacity to scale. It's particularly helpful because running a selenium grid on virtual machines or even separate machines takes up a lot of unnecessary computing overhead.

Docker images share some system resources, so they need fewer resources than a virtual machine. Several nodes can be deployed in a single instance. With Docker, by using docker swarm, networking also can be handled easily. Few advantages:

1. Easily scalable automation infrastructure.
2. Reduced cost for resources like servers, multiple VMs etc.
3. Easy to deploy in multiple servers.
4. The entire infra can be made open source and c Windows/Linux servers can be used.

Case Study

Customer need:
The client needed an independent testing partner who could develop perfect mechanisms for testing their enterprise applications and perform automation to the extent possible while balancing cost, time & ROI.

Problem statement:
"We need to run 1500 selenium tests as part of regression. We have used the selenium grid using multiple virtual machines, and it took almost 8 hours for the entire suite to run. This became a challenge as running those many tests on all environments took too much time, and analyzing the failures of automation tests is a tedious task."

Solution provided:
Our team started analyzing the use of Docker, and as each container runs separately, they could deploy many docker containers at once in a machine. By using the docker swarm, they could deploy containers on multiple servers. They started with two servers and set up a docker swarm in those two servers. Now they could deploy as many containers as they wanted using docker swarm based on our server configurations. They deployed 10 selenium node containers per server and a hub in the manager server. In this way, they managed to run 20 tests in parallel. This worked fine, and execution time decreased drastically. Then they took another great step of horizontal scaling and increased the servers. If they increased one server at a time, they would get 10 more parallel selenium nodes to run tests.

Benefits:
In this way, finally, they used four servers and managed to run 40 tests in parallel. Execution time decreased to below one hour for all the tests, and the client was able to run tests on multiple environments for each deployment to identify the failures faster.

Conclusion

To scale up test execution of Selenium scripts, Docker is very handy, easy to use, and cost-free.

Get in touch with us for feasibility studies, joint workshops, pilots & ideation sessions. Let's connect

Contributors

Sai Pawan Lingutla
Senior Software Engineer, Project Delivery
saipawan.lingutla@acsicorp.com

Geetha Pavani Achutuni
Senior Manager, Project Delivery
geetha.achutuni@acsicorp.com