DEPLOYING FUNCTIONAL TESTING TOOLS ON CONTINUOUS INTEGRATION

Presented by Zineb Regragui & Eric Merisier
AGENDA
Agenda

• How to accelerate testing tools deployment?
• Dockers concepts and benefits
• Tutorial Docker Desktop
• Question?
HOW TO ACCELERATE TESTING TOOLS DEPLOYMENT?
Business Expectations towards the projects

• High Responsiveness
  • Responding quickly to the customers needs to differentiate and gain market shares

• High Productivity
  • Reducing the Time to Market of new functionalities to take the advantage over the competitors

• High Quality
  • Optimizing End User Experience to retain and win customers
The pillars of success

Whatever the methodology, its success relies on:

• Measurements
• Sharing
• Automation
Measurements

**Containers** provide the ability to run a **process** in an isolated environment.

This isolation allows you to run **many services** simultaneously **on a given host**.

Unlike virtual machines, physical resources are not reserved but **allocated dynamically**.

*Docker lets you increase your test volume*
Sharing

**Containers** allow testers to package up a testing tool

**Images** are pushed in a repository and made available to the rest of the team

Several images, public or private, can be assembled to compose a complete testing solution, a **service**

*Docker encourages sharing of components*
Automation

It takes only a few minutes to run containers from an image into a testing environment.

Containers can be created and removed on demand

Container executions can be managed by CI

Docker leverages automation to increase testing frequency
DOCKERS CONCEPTS AND BENEFITS
Docker benefits for testing

Configuration work done only once

Quick deployment on multiple environments

Parallelization of various configurations on the same server

High frequency execution via an automation server

*Increasing speed and reproducibility, Docker improves the productivity of the test team*
TUTORIAL DOCKER DESKTOP
Question?

In your company is the use of containers a common practice?

The use of containers concerns:

1. Applications?
2. Tools?
3. Both?
4. None?
Docker benefits for testing

It’s possible to integrate Docker within a VM
What you will learn

• Docker Toolbox or Docker Desktop?
• How to setup a Selenium-Grid instance
• How to build your own docker image
• How to manage a service with docker compose
• How to integrate your execution in a CI/CD pipeline
Windows Docker installation

• **Docker Toolbox**
  - For old configuration (Hyper-V not supported by system and hardware)
  - Needs to install Virtualbox to install a VM based on Linux

• **Docker Desktop**
  - For new configuration with Hyper-V
  - A server docker engine Linux/windows is integrated
  - It’s easy !!
Docker installation

• After installation and reboot, start the docker Desktop

• A whale icon appears in the task bar
Docker Installation

- Use from a Powershell command (administrator)

```bash
#> docker version
```

https://hub.docker.com/search?q=&type=image
2,693,531 available images, Only 19,482 available images tagged for engine Windows

Don’t switch to windows container agent
Running out our first container

PS D:\temp\GRID> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:b8ba256769a0ac28dd126d584e0a2011cd2877f3f76e093a7ae560f2a5301c00
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
What happened?

Docker Hub → « Hello World » Image

Pull → Docker agent → Start

Docker CLI → Run

Hello World

Execute → Docker Container
Docker repositories for images

- Docker images can be found locally, or pulled from a repository
- Repository can be hosted in the cloud or on premise on a dedicated server.
DOCKER FOR WINDOWS

Selenium-Grid setup
Lab N°1 : Selenium-Grid

• Selenium-Grid allows distributed test execution.
• It consists of a hub and one or more nodes

• Your tests can be run against
  • multiple browsers
  • multiple versions of browser
  • browsers running on different operating systems.
Lab N°1 : Selenium-Grid

• The problem is that:
  • Setup is hard and long-time consuming
  • Configuration management needs efforts
  • Execution requires large resources

• How Docker can help us manage this?
  → Example : Create a HUB with a NODE based on existing image.
Lab N°1 : Selenium-Grid

1. Create a network

docker network create grid

2. Create a Hub

docker run --detach --publish 4444:4444 --net grid --name selenium-hub selenium/hub:3.141.59-titanium

3. Create a Node (Chrome + Vnc)

docker run -d -p 5900:5900 --net grid --name selenium-node-chrome -e HUB_HOST=selenium-hub -e NODE_MAX_INSTANCES=5 -e NODE_MAX_SESSION=1 -e VNC_NO_PASSWORD=1 -v /dev/shm:/dev/shm selenium/node-chrome-debug:3.141.59-titanium

Hostname = name of hub container

Name container

Exposure port

Run background

Network

Images
Lab N°1 : Selenium-Grid

Your first grid is already up and running!
Lab N°1 : Selenium-Grid
DOCKER FOR WINDOWS

Docker image building
Docker image

• A Docker image is a pre-built environment for a certain service or tool

• It’s a collection of files, libraries and configuration files that build up an environment

• The main source of docker images is the docker store

But what if you need to create your own image?
Start your container in interactive mode

Setup manually a configuration step And test your container

Commit Container to image

Build your image from the Dockerfile

Is better for sharing

Experiments and stores the good Steps for the Dockerfile

Custom image

Base Image
Dockerfile

- Dockerfile content: command instructions
- Docker CLI to build image from Dockerfile
Scalability, modularity
DOCKER FOR WINDOWS
Robot Framework image building
Lab N°2 : Robot FW container experimenting

• How to create a robot Framework container

• We have to :
  1. choose a base image (alpine : minimalist linux librarie)
  2. Identify all the required steps to meet our objective
**Lab N°2 : Robot FW container experimenting**

- The **alpine image** will be our starting point
  - With an access to RFW tests files
  - With an access to GRID Selenium

- **4 identified steps needed to have an instance available**
  - Install python3
  - Upgrade PIP
  - Install RobotFramework
  - Install SeleniumLibrary
Lab N°2 : Robot FW container experimenting

Run an alpine container in **interactive mode**

```bash
// Get IP of DockerNAT (accessible since localhost and containers) to access on Selenium GRID
$ipGrid=$(Get-NetIPAddress -InterfaceAlias 'vEthernet (DockerNAT)' | %{$_._IPAddress})

// set up container (--interactive –tty)
docker run -it --name devRBF -v d:/temp/RBF:/tmp/RBF --add-host="seleniumgrid:$ ipGrid alpine

// Copy Robot FW tests files in our local filesystem d:/temp/RBF/tests/
```

- testSuiteLog.robot.txt
- testSuiteSelenium.robot.txt
Lab N°2 : Robot FW container experimenting

- We are now connected to the container

```bash
// From the container, verify the access to grid selenium
more /etc/hosts
apk --no-cache add curl
curl http://seleniumgrid:4444/wd/hub/status

// From the container, verify the access to the robot framework tests files
ls /tmp/RBF/tests/
```

```
# ls /tmp/RBF/tests/
testSuiteLog.robot testSuiteSelenium.robot
```
Lab N°2 : Robot FW container experimenting

• The next step is the installation of
  • Python3
  • Robot Framework
  • Selenium library for Robot FW

```bash
apk add --no-cache python3 &&
pip3 install --no-cache --upgrade pip setuptools wheel &&
pip3 install --no-cache --upgrade robotframework &&
pip3 install --no-cache --upgrade robotframework-seleniumlibrary
```
Lab N°2 : Robot FW container experimenting

```bash
# Make add --no-cache python3 &
> pip3 install --no-cache --upgrade pip setuptools wheel &
> pip3 install --no-cache --upgrade robotframework &
> pip3 install --no-cache --upgrade robotframework-seleniumlibrary &
```

fetch http://docker.dns.alpahlinux.org/alpah5a/v1.10/community/386_64/AMC/index.tar.gz
(1/1) Installing libbz2 (1.0.6-7)
(1/1) Installing libxml (2.8.21-2)
(1/1) Installing libffi (3.2.1-r0)
(1/1) Installing zlib (1.2.11-1)
(1/1) Installing ncurses-termcap-base (6.1-pcwsx518-r0)
(1/1) Installing ncurses-termcap (6.1-pcwsx518-r0)
(1/1) Installing ncurses-libs (6.1-pcwsx518-r0)
(1/1) Installing readline (6.0.0-5)
(1/1) Installing sqlite-libs (3.32.1-2)
(1/1) Installing python (3.7.4-4)
executing /bin/sh -v 1.0.3.0-1.2-trigger
Ok: 72 MB in 29 packages
Collecting pip
  Downloading https://files.pythonhosted.org/packages/3b/db/5e3e70b132e2a7e74b0cc14e5e5d5f107f9e7384cobf32f92e40dc2e5514a794e/pip-19.2.3-py2.py3-none-any.whl (1.4MB)
  1.4MB  07:43/6s
Collecting setuptools
  Downloading https://files.pythonhosted.org/packages/3b/3b/0e05b2f7926baed287c83964e1ec6768e81f0eb547145df7979e20b4ae4e2e/setuptools-41.2.0-py2.py3-none-any.whl (79kB)
  79kB  00:00/0s
Collecting wheel
  Downloading https://files.pythonhosted.org/packages/3b/1a/27a6d9fe5d7ee5b0d5b5852d4871a01b83e7e88266c7f/wheel-0.33.6-cp27-none-any.whl (1.8MB)
  1.8MB  00:00/0s
Successfully installed wheel-0.33.6

Installing collected packages: pip, setuptools, wheel
Found existing installation: pip 19.0.3
Uninstalling pip-19.0.3
Successfully uninstalled pip-19.0.3
Found existing installation: setuptools 40.8.0
Uninstalling setuptools-40.8.0

Successfully installed pip-19.2.3 setuptools-41.2.0 wheel-0.33.6

Collecting robotframework
  Downloading https://files.pythonhosted.org/packages/22/8f/39f74c5e45f978b595e6386605b21b735886e821b7d4c6e6c4a0d6bba42f32e9d/robotframework-3.1.1-py2.py3-none-any.whl (882kB)
  882kB  00:00/0s
Successfully installed robotframework-3.1.1

Collecting robotframework-seleniumlibrary
  Downloading https://files.pythonhosted.org/packages/92/7b/6f36e489c887d86527397875f69f4465c6d33545215d2fa548c5f9f/robotframework_seleniumlibrary-4.0.0-py2.py3-none-any.whl (89kB)
  89kB  00:00/0s
Collecting selenium==3.141 (from robotframework-seleniumlibrary)
  Downloading https://files.pythonhosted.org/packages/0b/8e/44342a470bbedce4ebbfaf1f1e31719009698d1102ab4ed508e752e068505/3/selenium-3.141.0-py2.py3-none-any.whl (94kB)
  94kB  00:00/0s
Requirement already satisfied, skipping upgrade: robotframework==3.6.4 (in /usr/lib/python)
  Downloading https://files.pythonhosted.org/packages/1f/0b/4f490f0e0bacedeb4a2f25f1e7110209698d1102ab4ed508e752e068505/selenium-3.141.0-py2.py3-none-any.whl (94kB)
  94kB  00:00/0s
Successfully installed selenium==3.141 (from robotframework-seleniumlibrary)

Successfully installed robotframework-seleniumlibrary-4.0.0 selenium-3.141.0 urllib3-1.25.6
```
Lab N°2 : Robot FW container experimenting

- The container has been successfully upgraded with the RBF components
- Verify the execution of a Robot FW tests on the Selenium GRID

```
robot --outputdir /tmp/RBF/output --xunit TEST-result.xml --debugfile TEST-debug.txt /tmp/RBF/tests/testSuiteSelenium.robot
```

```
# robot --outputdir /tmp/RBF/output --xunit TEST-result.xml --debugfile TEST-debug.txt /tmp/RBF/tests/testSuiteSelenium.robot
testSuiteSelenium
-----------------------------------------------
testCaseUCAAT
seleniumGrid= http://seleniumgrid-4444/wd/hub
capabilities= {'browserName': 'chrome', 'version': '', 'platform': 'ANY', 'goog:chromeOptions': {'extensions': [], 'args': ['test']}
STEP GET TO URL= https://ucaat.etsi.org/
STEP GET TITLE= Home
STEP SCREENSHOT= /tmp/RBF/output/UCAAT.png
testCaseUCAAT
-----------------------------------------------
testSuiteSelenium | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

debug: /tmp/RBF/output/TEST-debug.txt
output: /tmp/RBF/output/output.xml
xunit: /tmp/RBF/output/TEST-result.xml
log: /tmp/RBF/output/log.html
report: /tmp/RBF/output/report.html
#`
```
Lab N°2 : Robot FW container experimenting

• Test the container behaviour

  Start / Exit container with interactive mode
  Start / Exec command line / Stop container

• You can conserve or delete the experimental container
Lab N°2 : Robot FW container experimenting
Lab N°3 : Robot FW image creation

• Update the dockerfile with the lines:

   FROM alpine
   RUN apk add --no-cache python3 && \
       pip3 install --no-cache --upgrade pip setuptools wheel && \
       pip3 install --no-cache --upgrade robotframework && \
       pip3 install --no-cache --upgrade robotframework-seleniumlibrary && \
       pip3 freeze
   WORKDIR /tmp/RBF/tests/
   ENTRYPOINT ['robot', '--outputdir', '/tmp/RBF/output', '--xunit', 'TEST-result.xml', '--debugfile', 'TEST-debug.txt']
   CMD ['

• Build new image

   cd D:/temp/RBF/build/ (directory of the dockerfile)
   docker build -t alpine/rbf:latest . (create image from dockerfile)
Lab N°3 : Robot FW image creation

• Run container from new image Robot FW

Commande line = ENTRYPOINT (fixed) + CMD (can be overwrited, by default launch all test in directory)
--rm (remove) : delete container after this running

docker run --rm -v d:/temp/RBF:/tmp/RBF --add-host="seleniumgrid:"$ipGrid alpine/rbf
or

docker run --rm -v d:/temp/RBF:/tmp/RBF --add-host="seleniumgrid:"$ipGrid alpine/rbf
testSuiteSelenium.robot
Lab N°3 : Robot FW image creation

- Result under D:\temp\RBF\output

```
PS D:\temp\RBF\output> Get-ChildItem -Name
log.html
output.xml
report.html
TEST-debug.txt
TEST-result.xml
UCAAT.png
PS D:\temp\RBF\output> |
```
Lab N°3 : Robot FW image creation
A service needs several containers
Example for the GRID Selenium
Lab N°4: Docker Compose and Jenkins

Orchestrate the running of 7 containers with `docker compose`

- **Docker Robot FW**
- **Docker HUB**
- **Docker NODE CHROME**
- **Docker NODE FIREFOX**
- **Docker NOVNC**
- **Docker NODE CHROME x11vnc**
- **Docker NODE FIREFOX x11vnc**

Run container from a JOB Jenkins

The tests and results are stored on workspace jenkins job

View execution

Sources of the Tests

http

The tests and results are stored on workspace jenkins job

http

http

http

http

http
Lab N°4 : Docker Compose

• Docker compose helps us, through a YAML file, to manage different containers as a service

• It can specifies:
  • Dependencies between containers
  • Volumes, ports

version: "3"
services:
selenium-hub:
  image: selenium/hub:3.141.59-titanium
  container_name: selenium-hub
  ports:
    - "4444:4444"
  chrome:
    image: selenium/node-chrome:3.141.59-titanium
    container_name: selenium-node-chrome
    volumes:
      - /dev/shm:/dev/shm
    depends_on:
      - selenium-hub
    environment:
      - HUB_HOST=selenium-hub
      - HUB_PORT=4444
      - NODE_MAX_INSTANCES=4
      - NODE_MAX_SESSION=3
  firefox:
    image: selenium/node-firefox:3.141.59-titanium
    container_name: selenium-node-firefox
    volumes:
      - /dev/shm:/dev/shm
    depends_on:
      - selenium-hub
    environment:
      - HUB_HOST=selenium-hub
      - HUB_PORT=4444
      - NODE_MAX_INSTANCES=4
      - NODE_MAX_SESSION=3

deploy:
  replicas: 2
Lab N°4 : Docker Compose

- Run Service as demon
  `docker-compose up -d`

- Stop Service
  `docker-compose down`

→ It’s easy !!!

- Force execution on the NODE Firefox VNC
  `docker run --rm -v d:/temp/RBF:/tmp/RBF --add-host="seleniumgrid":$ipGrid alpine/rbf --variable FORCE_VNC:TRUE --variable BROWSER:FIREFOX testSuiteSelenium.robot`

The test script sets the capabilities so that the Selenium HUB directs the test to the expected Selenium NODE.
Lab N°4 : Jenkins

- Run Service as demon (see the installation of Jenkins in annex)

\[
\text{docker run -u root -d -p 8080:8080 -p 50000:50000 -v D:/Docker/Jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -v D:/temp/RBF:/tmp/RBF --name jenkins jenkinsci/blueocean:1.18.1-bcc31d32159f}
\]

- Or else restart the service

\[
\text{docker start jenkins}
\]
Lab N°4 : Use Solution

- Finally, the tester use only http connection

<table>
<thead>
<tr>
<th>Services</th>
<th>Urls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consult the state of Selenium GRID</td>
<td><a href="http://localhost:4444/grid/console">http://localhost:4444/grid/console</a></td>
</tr>
<tr>
<td>Jenkins to launch and schedule the tests and consult the results</td>
<td><a href="http://localhost:8080">http://localhost:8080</a></td>
</tr>
<tr>
<td>View in real time test execution on chrome</td>
<td><a href="http://localhost:8081/vnc.html">http://localhost:8081/vnc.html</a></td>
</tr>
<tr>
<td>View in real time test execution on firefox</td>
<td><a href="http://localhost:8082/vnc.html">http://localhost:8082/vnc.html</a></td>
</tr>
</tbody>
</table>

The final solution is made on servers, the urls will not be on localhost, there will certainly be a reverse proxy (nginx) to provide the various services without the ports being visible.
To improve, it's need to mount a volume between the Grid Nodes and Jenkins if the web application has a transfer file feature. (upload file or download file)
Lab N°4 : Docker Compose and Jenkins
Go further for management and remote deployment

- Kubernetes/Docker-Swarm: are open-source container-orchestration system for automating application deployment, scaling, and management. 
  

- Ansible is an open-source software provisioning, configuration management, and application-deployment tool. (SSH Agent-Less)
Congratulations
We managed to use tools without installing them, but only through docker
ANNEX
Docker Desktop installation

**System requirements**
- **Windows 10 64-bit**: Pro, Enterprise, or Education (Build 15063 or later).
- Hyper-V and Containers Windows features must be enabled.

**Hardware prerequisites**
- 64 bit processor with **Second Level Address Translation (SLAT)**
- 4GB system RAM
- BIOS-level hardware virtualization support must be enabled in the BIOS settings.

**Documentation** [https://docs.docker.com/docker-for-windows/install/](https://docs.docker.com/docker-for-windows/install/)
Docker best practices and security

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
https://www.docker.com/blog/intro-guide-to-dockerfile-best-practices/
https://snyk.io/blog/10-docker-image-security-best-practices/
https://cloud.google.com/blog/products/gcp/7-best-practices-for-building-containers

Example for docker file:
Use official/certified image
Package a single application per container
Minimize the number of layers
Don’t use default USER (root)
Use no-cache
Attached : Installation of Jenkins from image

- **Create un local workdir** : `D:/Docker/Jenkins (persistence of installation and data)`
- **Launch this command**
  
  ```
docker run -u root --rm -d -p 8080:8080 -p 50000:50000 -v D:/Docker/Jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -v D:/temp/RBF:/tmp/RBF --name jenkins jenkinsci/blueocean:1.18.1-bcc31d32159f
  ```

- **Copy le password that is in D:\Docker\Jenkins\secrets\initialAdminPassword**
- **Continue the installation since**  
  - Install the suggested plugins
  - Create an admin user

- [https://github.com/auriuki/jenkins-docker](https://github.com/auriuki/jenkins-docker)
- [https://hub.docker.com/r/jenkinsci/blueocean](https://hub.docker.com/r/jenkinsci/blueocean)