СБОРКА ПРОЕКТА

После того как все тесты написаны и все зависимости и плагины в РОМ указаны пришло время

перенести нашу программу в Docker, но сначала надо создать jar files из нашего проекта (смотри РОМ)

полученные файлы положить в (смотри ром) ${project.build.directory}/docker-resources

Для этого прописываем :

mvn clean package -DskipTests

ВОЗМОЖНОСТЬ ВОСПРОИЗВОДИТЬ СОБРАННУЮ ПРОГРАММУ С ПОМОЩЬЮ JAVA (без DOCKER-a)

После этого у нас имеется возможность запустить созданный в jar-ках проект с помощью java:

java -cp [where classes are presented] org.testng.TestNG [suite-file-path] <-- выполнит указанный testng\_suit.xml file (то что указал в [suite-file-path])

-cp <-- stands for class path

[where classes are presented] <-- место где расположены скомпилированные .class файлы используемые программой (у нас они в libs)

org.testng.TestNG <-- инструкция уведомляющая maven что хотим использовать suit testNg файлы

[suite-file-path] <-- путь до suit testNg файла который хотим выполнить

Например: (ПОМНИ ПРО ДИРЕКТОРИЮ ГДЕ РАСПОЛОЖЕНЫ НУЖНЫЕ ФАЙЛЫ, ИНАЧЕ НЕ СРАБОТАЕТ)

java -cp .\libs\\* org.testng.TestNG -parallel "none" test\_sutes/flight-reservation.xml

-parallel "none" <--можно устанавливать опции такие же как в test suit-ах

На этом этапе можно переопределять переменные с помощью java -D команд в консоли

We can override the system properties in .properties file by using -D[name\_of\_the\_parameter=new\_value\_for\_parameter]

Например:

java -Dselenium\_grid\_enabled=true -cp "./libs/\*" org.testng.TestNG .\test\_sutes\vendorPortal.xml

-Dselenium\_grid\_enabled=true <-- переписывает значение параметра под названием: selenium\_grid\_enabled

или можно

-Dbrowser=firefox <-- переписывает значение параметра под названием: browser

СОЗДАНИЕ первой DOCKER IMAGE из Dockerfile

Пришло время создать Docker имедж. Но для чего? Мы создадим Docker image с java и закинем в эту

имедж наш проект, и так как в контейнере уже будет джава мы сможем запускать контейнер и внутри его

запускать на этой java наш проект,

для этого в корне проекта надо создать Dockerfile, на этом этапе он будет выглядеть так:

FROM bellsoft/liberica-openjdk-alpine:17

WORKDIR /home/ocker-resources

ADD /target/docker-resources ./

здесь: берём за основу имедж где уже стоит java, в контейнере мы устанавливаем директорию с которой

мы начнём работать (инструкция - WORKDIR), сюда же добовляем паку с проектом которую сгенерил maven.

Теперь в терминале, в директории где лежит наш Dockerfile создаём имедж

docker build -t=delete\_me\_i\_am\_experement\_for\_framework .

Теперь можно запустить созданную имедж в итерактивном режиме и посмотреть что мы имеем там наши файлы.

docker run -it delete\_me\_i\_am\_experement\_for\_framework

и наш проект там будет, и java, но запустить мы ничего не сможем так как там нет браузеров!

Мы можем установить на наш имедж ещо и браузеры, но это не рационально, требования по тестированию

могут изменятся не только по типу браузера но и по версии. Можно воспользоватся selenium grid, добавит браузеров в него и

зделать из него тоже имедж.

ЗАПУСК ДОКЕР ИМЕДЖ С SELENIUM GRID

Создадим имедж с гридом + браузеры, запускаем его с помощью docker-compose up (если мы в его директории)

version: "3"

services:

hub:

image: selenium/hub:4.10.0

ports:

- 4444:4444

chrome:

image: selenium/node-chrome:4.10

shm\_size: '2g'

depends\_on:

- hub

deploy:

replicas: 1

environment:

- SE\_EVENT\_BUS\_HOST=hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true

- SE\_NODE\_MAX\_SESSIONS=4

- SE\_VNC\_NO\_PASSWORD=1

firefox:

image: selenium/node-firefox:4.10

depends\_on:

- hub

deploy:

replicas: 1

environment:

- SE\_EVENT\_BUS\_HOST=hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true

- SE\_NODE\_MAX\_SESSIONS=4

- SE\_VNC\_NO\_PASSWORD=1

Далее стартуем нашу имедж с проектом, делаем volume mapping чтобы видеть результаты выполнения

у себя на локальной машине.

docker run -it -v /C/for\_experements/docker\_experements/DockerFile/vins\_guru\_framwork/test\_output:/home/docker-resources/test-output delete\_me\_i\_am\_experement\_for\_framework

Теперь надо выполнить java команды и переопределить некоторые properties чтобы docker выполнял

тесты на гриде а не искал браузеры у себя в имедже.

java -Dbrowser=chrome -Dselenium\_grid\_enabled=true -Dselenium\_grid\_hubHost='172.23.96.1' -cp 'libs/\*' org.testng.TestNG test\_sutes/vendorPortal.xml

Блять столько потратил время из-за этих ковычек, просто пизда. Этот код работает в интелледж терминале с директории

где лежит построенный мавеном проект, но этот же код не работает в имедж, хотя в имедж всё тоже

самое. Имедж не может достучатся до грида, она просто не видит грид. Всё, разобрался. Всё дело в ip адресе, когда выполняеш

ipconfig в терминале то получаеш 3 этих адресов (отметил стрелками):

Connection-specific DNS Suffix . :

Link-local IPv6 Address . . . . . : fe80::5fb2:2f75:3ec8:a16d%31

IPv4 Address. . . . . . . . . . . : 172.23.96.1 <--------------

Subnet Mask . . . . . . . . . . . : 255.255.240.0

Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet 2:

Connection-specific DNS Suffix . :

IPv4 Address. . . . . . . . . . . : 192.168.56.1 <--------------

Subnet Mask . . . . . . . . . . . : 255.255.255.0

Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . :

IPv4 Address. . . . . . . . . . . : 192.168.100.2 <--------------

Subnet Mask . . . . . . . . . . . : 255.255.255.0

Default Gateway . . . . . . . . . : 192.168.100.1

Дак вот до грида можно достучатся используя IP адрес 172.23.96.1 или 192.168.100.2 . как и почему надо ещо учить/читать.

ADDING ENTRYPOINT TO DOCKERFILE

Чтобы не писать длинные команды с java в консоли контейнера (можно легко ошибится) можно зделать точку входа, когда

создастся контейнер, docker сразу выполнит то, что указано в ENTRYPOINT.

Dockerfile:

FROM bellsoft/liberica-openjdk-alpine:17

WORKDIR /home/docker-resources

ADD /target/docker-resources ./

ENTRYPOINT java -Dbrowser=${BROWSER} -Dselenium\_grid\_enabled=true -Dselenium.grid.hubHost=${HUB\_HOST} -cp 'libs/\*' org.testng.TestNG test\_sutes/${TEST\_SUITE}

Удаляем нашу image и соберём новую с таким же именем, но с обновлённым Dockerfile-ом.

Пересоберём проект:

mvn clean package -DskipTests

Построим имедж:

docker build -t=delete\_me\_i\_am\_experement\_for\_framework .

Так как в ENTRYPOINT есть параметры надо обязательно их указывать при запуске, указываем и запускаем. Не знаеш какие параметры

указывать при старте? Смотри в Dockerfile в ENTRYPOINT. Делаем это c volume mapping и видим на сколько длинная команда получилась...

# docker run -e BROWSER=chrome -e HUB\_HOST=192.168.100.2 -e TEST\_SUITE=vendorPortal.xml -v /C/for\_experements/docker\_experements/DockerFile/vins\_guru\_framwork/test\_output:/home/docker-resources/test-output delete\_me\_i\_am\_experement\_for\_framework

ОБЬЕДИНЯЕМ ГРИД + ИМЕДЖ В ОДИН .YAML FILE

На этом этапе у нас есть грид(chrome+firefox) и Dockerfile(Entrypoint). Сначала запускаем грид, потом имедж (java+проект), при этом чтобы попасть в грид

надо постоянно указывать порт грида при запуске имедж frameWork-a, так как по факту грид <--это одна имедж, а delete\_me\_i\_am\_experement\_for\_framework <--это

другая имедж, т.е. у нас 2 файла, что неудобно и хранить их и запускать. Надо обьединить всё в один файл.

yaml.file:

version: "3"

services:

hub:

image: selenium/hub:4.10.0

ports:

- 4444:4444

chrome:

image: selenium/node-chrome:4.10

shm\_size: '2g'

depends\_on:

- hub

environment:

- SE\_EVENT\_BUS\_HOST=hub # BY specifiing the ports the chrome container can talk to hub container

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442 #PORT

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true # podklychaem mnogopotochky

- SE\_NODE\_MAX\_SESSIONS=4 # ykazivaem skolko thred-ov v 1 syshnasti

- SE\_VNC\_NO\_PASSWORD=1 # in order to viw the work flow without password

firefox:

image: selenium/node-firefox:4.10

depends\_on:

- hub

environment:

- SE\_EVENT\_BUS\_HOST=hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true

- SE\_NODE\_MAX\_SESSIONS=4

- SE\_VNC\_NO\_PASSWORD=1

vendorPortal:

image: delete\_me\_i\_am\_experement\_for\_framework

depends\_on:

- firefox

environment:

- BROWSER=firefox

- HUB\_HOST=hub

- TEST\_SUITE=vendorPortal.xml

volumes:

- ./output/vendorPortal/:/home/docker-resources/test-output

И здесь обнаружилась приинтереснейшая штука. Ну с грид-ом всё понятно там настройки прежние. А вот при настройке сервиса vendorPortal мы используем имедж

которую мы построили из Dockerfile-a, затем инструкция depends\_on по словам автора она якобы задаёт порядок включения сервисов но не выполняет их в заданном

порядке, т.е. наш сервис в котором указано depends\_on: - firefox стартанёт после firefox но это (по словам автора) негарантирует что Docker когда стартанёт

firefox, дождётся полного его (firefox-a) включения и готовности к работе, и только потом стартанёт наш сервис, нет (по словам автора) это горонтирует лишь

то, что Docker сначала стартанёт firefox, а затем стартанёт наш контейнер и он не будет ждать полную загрузку firefox-a, он просто как бы расставляет

приоритет только для старта сервисов, а кто там загрузится и когда он это типа не контролирует. Незнаю как там правильно но по факту тесты не выполняются

первый раз, потом как бы выполняются хз. Покрутил по разному, бывает грид готов и тесты проходят, а бывает грид не готов и тесты не проходят,

особенно когда 2 сьюта. Чтобы тесты выполнялись корректно надо убедится что грид готов к работе (готов выполнить тесты).

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CHECKING GRID STATUS (в dockerfile добовляем curl + jq + runner)

curl - команда позволяет делать запросы на сервер/сайт и получать ответы, вот ссылка почитать: https://reqbin.com/req/c-kdnocjul/curl-commands.

Запустим только грид без браузеров. Yaml file будет такой:

version: "3"

services:

hub:

image: selenium/hub:4.10.0

ports:

- 4444:4444

С помощью curl will send request: curl http://192.168.56.1:4444/status

and will get response in a JSON format:

{

"value": {

"ready": false,

"message": "Selenium Grid not ready.",

"nodes": [

]

}

}

Как видим хоть грид и работает но так как у него нет браузеров то message: "Selenium Grid not ready.".

Добавим к гриду браузер, yaml file будет такой:

version: "3"

services:

hub:

image: selenium/hub:4.10.0

ports:

- 4444:4444

chrome:

image: selenium/node-chrome:4.10

shm\_size: '2g'

depends\_on:

- hub

environment:

- SE\_EVENT\_BUS\_HOST=hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true

- SE\_NODE\_MAX\_SESSIONS=4

- SE\_VNC\_NO\_PASSWORD=1

Выполним команду curl http://192.168.56.1:4444/status

Ответ:

"value": {

"ready": true,

"message": "Selenium Grid ready.",

"nodes": [

{

"id": "7a3f4134-ced2-4740-9f73-1ee1f3cc54c5",

... много ещё чего...

}

}

Весь ответ приводить не буду, но что нас интересеут дак это атрибут "message": "Selenium Grid ready." <-- т.е. атрибут имеет стринговое значение

"Selenium Grid ready." Итак мы нашли способ как получить данную инф. теперь надо воспользоватся ещё одной утилитой JQ <-- this is tool for processing

JSON inputs. Мы можем вызвать: curl http://192.168.56.1:4444/status и передать его результат на вход утилиты JQ например вот так:

curl http://192.168.56.1:4444/status | jq

Результатом будет: тот же респонс но разноцветный/крассивый, нас это не интересует, но jq can parse (анализировать) the JSON and we can extract the

information what ever you like. Например

curl http://192.168.56.1:4444/status | jq .value.ready <-- получим значение атрибута ready.

Идея всего этого заключается - чтобы выполняя эту команду снова и снова,проверять, готов ли грид и если да то только тогда выполнить тесты.

Но мы будем проверять атрибут message, следующей командой:

curl http://192.168.56.1:4444/status | jq .value.message <-- и jq вернёт нам всё что есть в этом атрибуте "Selenium Grid ready." с кавычками,

что нам в принципе не нужно, чтобы получит "чистое" значение используй флаг -r , что обозначает raw.

curl http://192.168.56.1:4444/status | jq -r .value.message = Selenium Grid ready.

Необходимо в проект добавить шел скрипт, смотри runner.sh. этот скрипт должен быть выполнен каждый раз когда запускается имедж, его назначение

запускать проверку о готовности грида и когда грид готов выполнить инструкцию на java, эта инструкция то-же самое что раньше было ENTRYPOINT в Dockerfile-e.

На этом этапе volume mapping-а нет в runner-e!

Dockefile:

FROM bellsoft/liberica-openjdk-alpine:17

RUN apk add curl jq <--- добавили утилиты в shell контейнера (по умолчанию их там нет)

WORKDIR /home/docker-resources <--- директория с которой стартуем в контейнере

ADD /target/docker-resources ./ <--- добовляем наш проект

ADD runner.sh runner.sh <--- добавили runner в имедж

ENTRYPOINT sh runner.sh <--- вызываем shell и выполняем runner

После пересобираем проект в новую имедж.

Yaml.file:

version: "3"

services:

hub:

image: selenium/hub:4.10.0

ports:

- 4444:4444

chrome:

image: selenium/node-chrome:4.10

shm\_size: '2g'

depends\_on:

- hub

environment:

- SE\_EVENT\_BUS\_HOST=hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true

- SE\_NODE\_MAX\_SESSIONS=4

- SE\_VNC\_NO\_PASSWORD=1

firefox:

image: selenium/node-firefox:4.10

depends\_on:

- hub

environment:

- SE\_EVENT\_BUS\_HOST=hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true

- SE\_NODE\_MAX\_SESSIONS=4

- SE\_VNC\_NO\_PASSWORD=1

vendorPortal:

image: frame\_work\_image

depends\_on:

- hub

environment: # так как переменные по умолчанию обьявленны в runner-e то здесь можно ограничить или вообще удалить

- TEST\_SUITE=vendorPortal.xml

volumes:

#- ./output/vendorPortal/:/home/docker-resources/test-output

- C:/for\_experements/docker\_experements/selenium\_grid/for\_framwork\_(vinsGuruTest)\_grid\_and\_tests-suties/scenario\_1/output/vendorPortal/:/home/docker-resources/test-output

flight-reservation:

image: frame\_work\_image # <--укажи действующую имедж!

depends\_on:

- chrome

environment:

- BROWSER=chrome

- HUB\_HOST=hub

- TEST\_SUITE=flight-reservation.xml

volumes:

#- ./output/flight-reservation/:/home/docker-resources/test-output

- C:/for\_experements/docker\_experements/selenium\_grid/for\_framwork\_(vinsGuruTest)\_grid\_and\_tests-suties/scenario\_1/output/flight-reservation/:/home/docker-resources/test-output

Для старта: docker-compose up и всё.) Но получается что имедж будет хранится на DockerHube а вот yaml file нет!

----------------------------------------------------------------------------------------------------------------------------------------------------------------

PASSING NEW TEST DATA FOR TESTS USING VOLUME MAPPING

When we build project with maven we hard coded the test-data for tests (json files)

The program takes these files and use them as data for running tests, so it looks like we have to

go to project and change these files or create new one in order to run the programm with new data.

But we can change this files in a diferent way. We can do this by volume mapping,

and give new modified file and programm will take it instead of the old one.

As experement in vendorPortal application we will change mike.json file and will

replace it with new fakeMike.json file. In that case we will be expecting test failure because

the new fakeMike.json file will have mistake/wrong parameter. But the porpose is to demonstrate

the ability to change data by volume mapping.

The name of new file can difere from the file in VM

for example:

- ./experement\_test\_data/mike.json:/home/selenium-docker/testData\_inJSON/vendorPortal/mike.json <-- the same name

- ./experement\_test\_data/mike1234.json:/home/selenium-docker/testData\_inJSON/vendorPortal/mike.json <-- new name

And by using this approch we can create the whole diferent xml file for example and change the same file in VM and

get new bihavior of our application

See the yaml file:

version: "3"

services:

hub:

image: selenium/hub:4.10.0

ports:

- 4444:4444

chrome:

image: selenium/node-chrome:4.10

shm\_size: '2g'

depends\_on:

- hub

environment:

- SE\_EVENT\_BUS\_HOST=hub # BY specifiing the ports the chrome container can talk to hub container

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442 #PORT

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true # podklychaem mnogopotochky

- SE\_NODE\_MAX\_SESSIONS=4 # ykazivaem skolko thred-ov v 1 syshnasti

- SE\_VNC\_NO\_PASSWORD=1 # in order to viw the work flow without password

firefox:

image: selenium/node-firefox:4.10

depends\_on:

- hub

environment:

- SE\_EVENT\_BUS\_HOST=hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true

- SE\_NODE\_MAX\_SESSIONS=4

- SE\_VNC\_NO\_PASSWORD=1

vendorPortal:

image: frame\_work\_image

depends\_on:

- hub

environment:

- TEST\_SUITE=vendorPortal.xml

volumes:

- ./output/vendorPortal:/home/docker-resources/test-output

- ./experement\_test\_data/fakeMike.json:/home/docker-resources/testData\_inJSON/vendorPortal/mike.json

flight-reservation:

image: frame\_work\_image

depends\_on:

- hub

environment:

- TEST\_SUITE=flight-reservation.xml

volumes:

- ./output/flight-reservation:/home/docker-resources/test-output

Это даёт больше гибкости самому проекту, получается что можно изменить файлы проекта внеся новые(находящиеся в другой директории),

при этом сам проект не поменяется.

В этом сценарии меняется только yaml. file

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

МАШТАБИРУЕМ БРАЗЕРЫ В RUN TIM-Е (ДЕЛИМ YAML FILE НА ГРИД И ФРЭЙМВОРК ИМЕДЖ)

Все файлы тут: **C:\for\_experements\docker\_experements\selenium\_grid\for\_framwork\_(vinsGuruTest)\_grid\_and\_tests-suties\scenario\_3**

We will try to scale(маштабировать) the browsers in a run time. The main idea is to create infrastructure (in this case it is browser nodes) whenever you need and scale it as much as you need, and after you finish discart it. До этого времени нам приходилось самим останавливать контейнеры используя Ctrl+c

For that:

1) create yaml file only with hub.

We will keep the framwork image separatly from grid in order to control browser scalling in a run time. Because of that in every service/browser

of the grid teg deploy: replica:0. Т.е. мы собираемся переписать значение этого атрибута когда будем запускать yaml.

version: "3"

services:

hub:

image: selenium/hub:4.10.0

chrome:

image: selenium/node-chrome:4.10

shm\_size: '2g'

depends\_on:

- hub

deploy:

replicas: 0

environment:

- SE\_EVENT\_BUS\_HOST=hub # BY specifiing the ports the chrome container can talk to hub container

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442 #PORT

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true # podklychaem mnogopotochky

- SE\_NODE\_MAX\_SESSIONS=4 # ykazivaem skolko thred-ov v 1 syshnasti

- SE\_VNC\_NO\_PASSWORD=1 # in order to viw the work flow without password

firefox:

image: selenium/node-firefox:4.10

depends\_on:

- hub

deploy:

replicas: 0

environment:

- SE\_EVENT\_BUS\_HOST=hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

- SE\_NODE\_OVERRIDE\_MAX\_SESSIONS=true

- SE\_NODE\_MAX\_SESSIONS=4

- SE\_VNC\_NO\_PASSWORD=1

2).

Our tests we will keep in separate file, and since we alredy have hub(or grid) in a separate file, we dont need hub here.

In order to be able to pass amaunt of browsers we keep it as environment variables

version: "3"

services:

vendorPortal:

image: frame\_work\_image

environment:

- BROWSER=${BROWSER}

- TEST\_SUITE=vendorPortal.xml

volumes:

- ./output/vendorPortal:/home/docker-resources/test-output

flight-reservation:

image: frame\_work\_image

environment:

- BROWSER=${BROWSER}

- HUB\_HOST=192.168.56.1

- TEST\_SUITE=flight-reservation.xml

volumes:

- ./output/flight-reservation:/home/docker-resources/test-output

So the first insructure:

docker-compose -f grid.yaml up --scale chrome=2 -d

Explanation:

-f grid.yaml <-- since we have few dockerCompose files we tell the docker what specifik file should be сompleted.

--scale chrome=2 <-- amaunt of nodes to be present (since we have 2 application we need 2 browsers).

-d <--run it in background.(in order) (in a docker file that responsible for grid the ports are removed so we are not able to

see the UI in a localhost. If you want to be able to see the UI of the grid use teg ports.)

the second insructure:

docker-compose up

Explanation:

Since the second .yaml file contains framwork (image with runner) runner see that grid is presented and execute the java

comand wich is presented in it.

You should be in the same directory where the desired dockerfile is presented, atherwise you need to provide name of the file

with flag -f <name\_of\_the\_file>

If we enter the comand:

docker-compose -f grid.yaml up --scale chrome=2 -d

the docker created 2 nodes of chrome, but if we want to change it for example to firefox tipe:

docker-compose -f grid.yaml up --scale firefox=2 -d

and docker will automaticaly destroy the chrome nodes and instead of it will create firefox, we dont need to delete it manyaly.

Pay attention if in .yaml file the browser set as chrome and you have fitrefox in grid running then you need to change the

OS variable (set BROWSER = ...) and than call for docker-compose up

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++