**Настройка первого контейнера**

1. Скачиваем образ Maven с docker hub. Именно этот образ будем использовать для запуска тестов

CMD > docker pull maven:3.8.4-ibmjava-8

1. Проверяем что maven установился и работает. Проверяем версию

CMD > docker run --rm maven:3.8.4-ibmjava-8 mvn -v

1. Запускаем тест. Пока без работы с другими контейнерами. Для этого создаем контейнер, мапим папку на хосте, где содержится скрипт теста. После с помощью -w указываем что при создании контейнера нужно сразу войти в замапленную папку (иначе контейнер не сможет найти нужный тест) и запускаем тест. После прогона теста удаляем контейнер

CMD > docker run --rm --mount type=bind,src=C:\Users\artiv\Desktop\LearnQA\_Docker\Autotests,target=/usr/src/mymaven -w /usr/src/mymaven maven:3.8.4-ibmjava-8 mvn -Dtest=SimpleTest test

Выведется ошибка, так как нет второго контейнера, куда отсылать команды



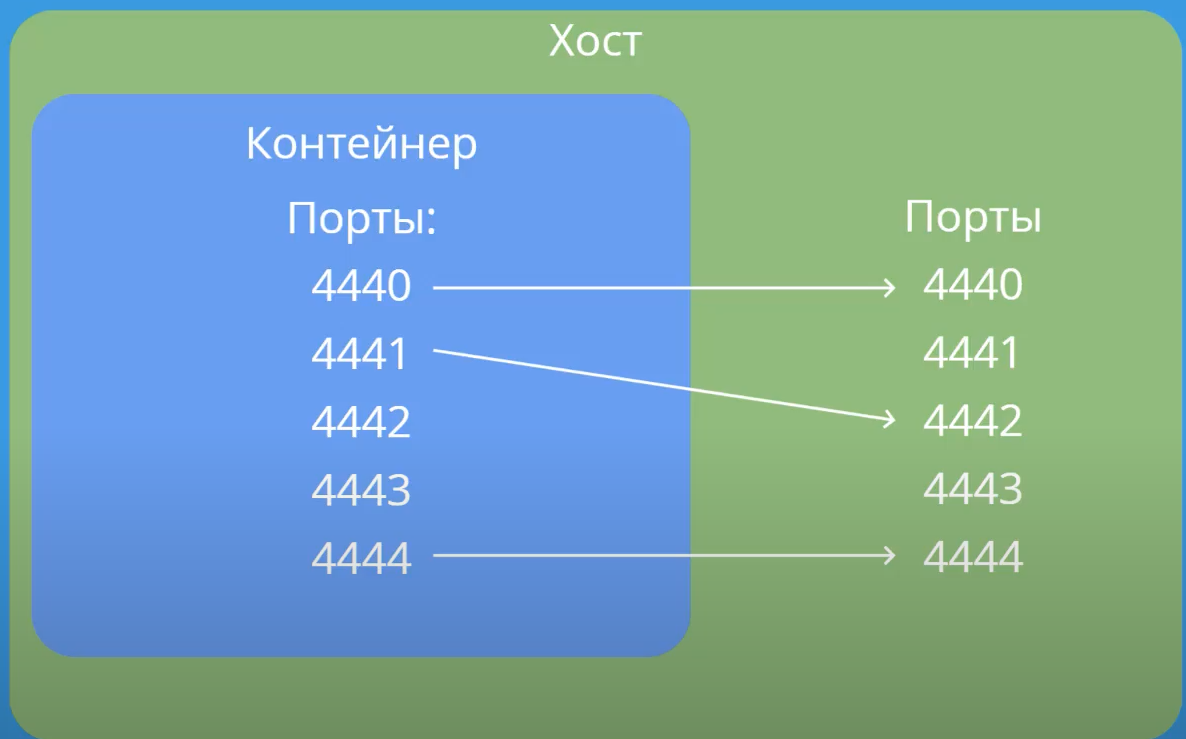
**Настройка второго контейнера**

1. Скачиваем Selenium Server

CMD > docker pull selenium/standalone-chrome:4.1.2

1. Запускаем в фоне контейнер с Selenium Server и привязываем port контейнера на порт хоста, что бы хост машина и контейнер могли работать вместе

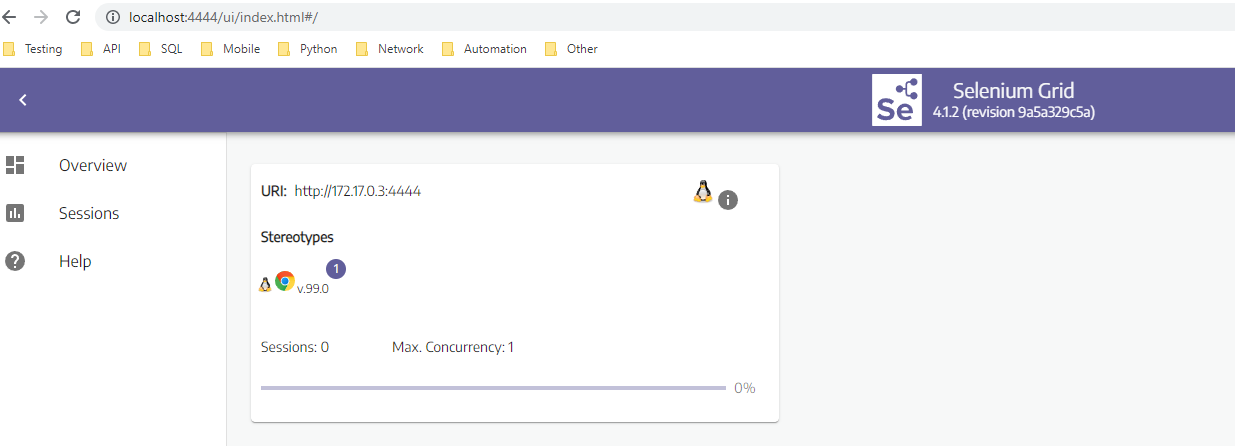
Selenium Server работает в Docker. Так как контейнеры docker изолированы от хост машины, значит их сеть так же изолирована. Для того что бы снаружи (хоста) было видно что происходит на порту внутри (docker контейнера), то нужно привязать внутренний порт к порту хост машины



Привязываем внешний порт 4444 к внутреннему 4444 (Selenium Server по дефолту работает на этом порте, но мы должны явно это указать) После запуска сервер висит на localhost:4444. Мы можем привязать любой внешний порты к внутреннему, главное что бы внутренний был 4444

CMD > docker run -d --rm -p 4444:4444 selenium/standalone-chrome:4.1.2

После мы можем с хоста зайти на Selenium Server. Но если мы создадим сессию в браузере, то браузер не увидим, так как он запускается в docker контейнере, но нам это и не нужно, ведь он поднимается что бы с ним взаимодействовали автотесты



**Настройка сети**

Настраиваем сеть так, что бы не только хост машина могла работать с Selenium Server, но и другие контейнеры

1. Создаем Ubuntu и скачиваем curl

CMD > docker run -it ubuntu bash

Bash > apt-get update

Bash > apt-get install apt-file

Bash > apt-file update

Bash > apt-get install curl

exit

1. Создаем образ из этого контейнера (где установлен curl)

CMD > docker commit [id] ubuntu\_with\_curl

1. Создаем контейнер из созданного образа, но указываем --network host, что бы docker контейнер видел наш хост

CMD > docker run -it --rm --network host ubuntu\_with\_curl bash

По дефолту созданный контейнер имеет опцию bridge. Но когда мы прописываем опцию host, то контейнер имеет доступ к сетевому пространству хост машины

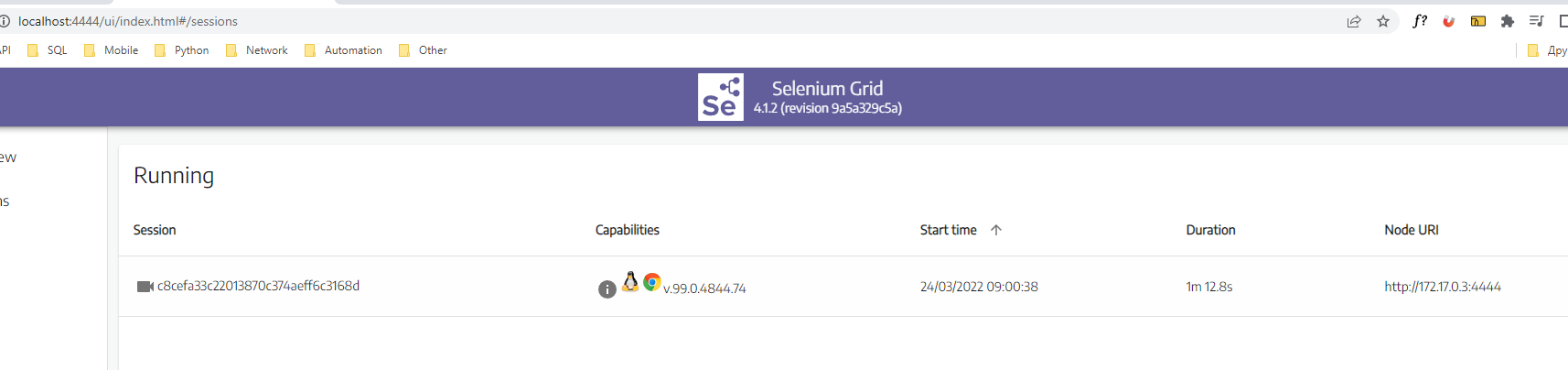
Network:

* bridge
* overlay
* macvlan
* host

1. Создаем сессию на Selenium сервере

Bash > curl 'http://localhost:4444/wd/hub/session' --data-binary '{"desiredCapabilities":{"browserName":"chrome"}}'

Если после этого перейдем на хосте в Selenium Server, то увидим, что запущена сессия.



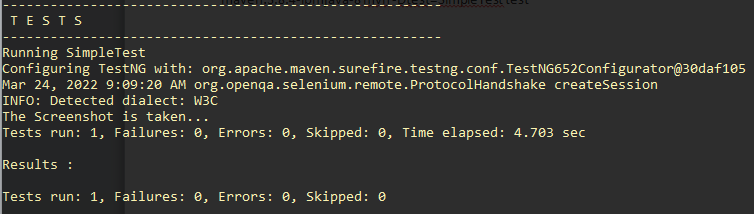
**Запуск тестов**

1. Стартуем контейнер с Selenium Server

docker run -d --rm -p 4444:4444 selenium/standalone-chrome:4.1.2

1. Запустим второй контейнер, который будет гонять тесты

docker run --rm --network host --mount type=bind,src=C:\Users\artiv\Desktop\LearnQA\_Docker\Autotests,target=/usr/src/mymaven -w /usr/src/mymaven maven:3.8.4-ibmjava-8 mvn -Dtest=SimpleTest test



1. Можно посмотреть логи

docker logs [id контейнера]