

Домашнее задание:

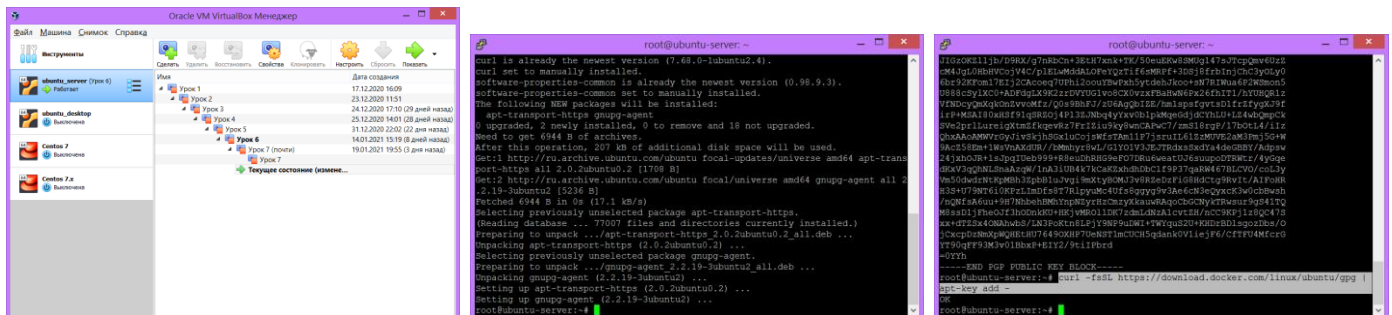
1. Переустановить операционную систему, подключить репозиторий Docker.
2. Запустить контейнер с Ubuntu.
3. * Используя Dockerfile, собрать связку nginx + PHP-FPM в одном контейнере.

1. Переустановить операционную систему, подключить репозиторий Docker.

Поступим проще, восстановим состояние VM на конец 6-го урока (после каждого ДЗ делал снимки). Зайдем под «root» (sudo su -).

Установим пакеты, необходимые для работы apt по протоколу HTTPS (apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common -y).

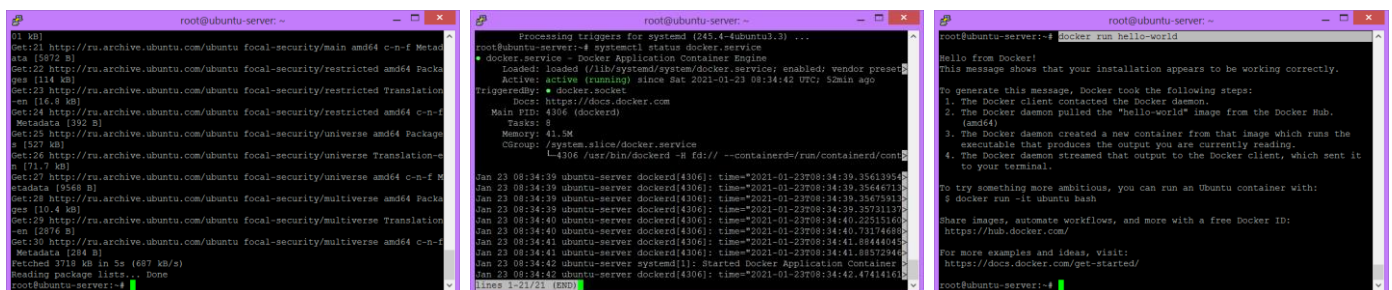
Просмотрим ключ репозитория на экране (curl -fsSL https://download.docker.com/linux/ubuntu/gpg) и добавим его (curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -).



Подключаем репозиторий (add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable").

Обновляем список пакетов (apt update). И устанавливаем пакет (apt install docker-ce -y). Убедимся, что «docker» установлен и стартовал (systemctl status docker.service).

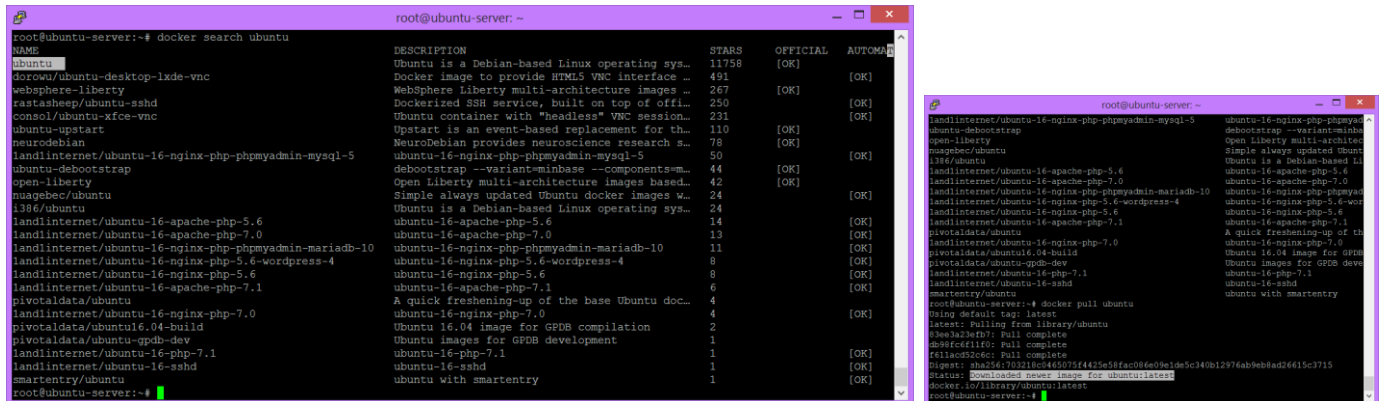
Проверим, что контейнеры запускаются (docker run hello-world), даже если их нет локально, то перед этим, загружаются с репозитория «The Docker daemon pulled the "hello-world" image from the Docker Hub»...



2. Запустить контейнер с Ubuntu.

Найдем образы с «ubuntu» (docker search ubuntu).

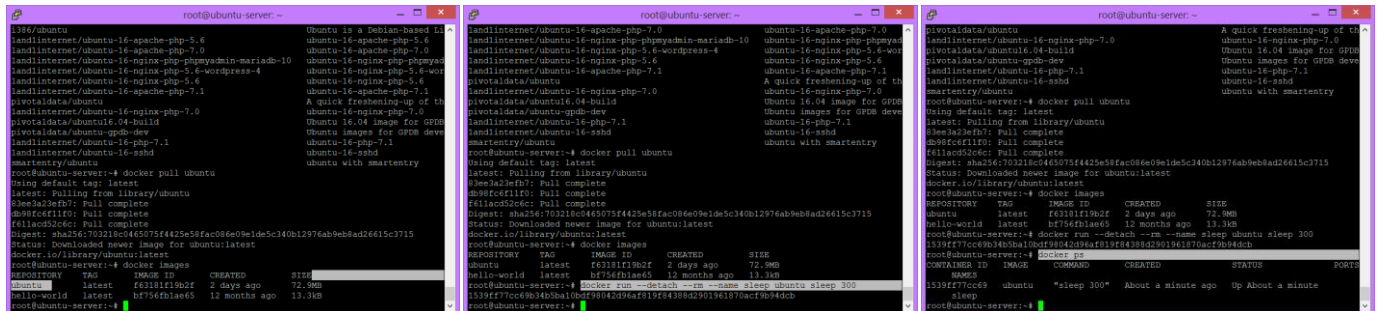
Скачаем имидж «ubuntu» (docker pull ubuntu)



Посмотрим, какие имиджи мы уже скачали (docker images), «ubuntu» среди них.

Запустим команду «sleep 300» в контейнер «ubuntu» на исполнение. При этом, сразу освободим консоль «--detach», а после завершения произойдет автоудаление «--rm» (docker run --detach --rm --name sleep ubuntu sleep 300)

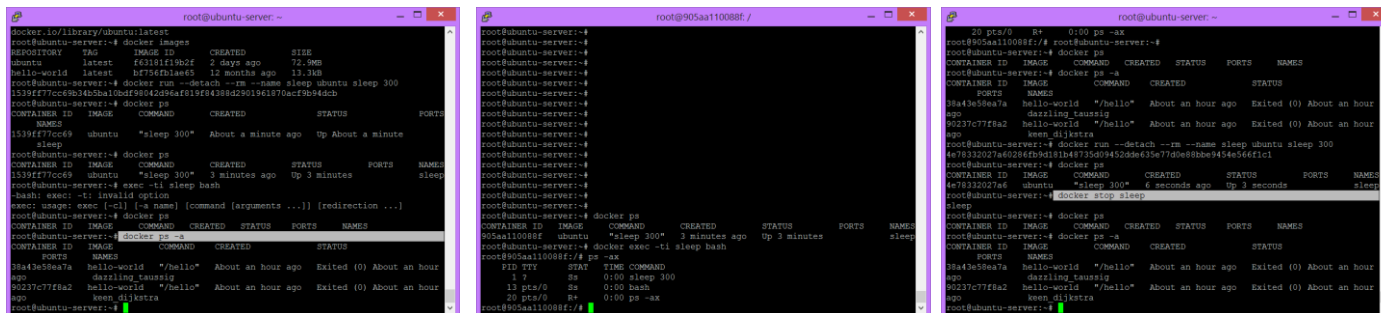
Убедимся, что контейнер работает (docker ps)



По истечению 5 мин, проверим, что из памяти контейнер «sleep» удален (docker ps), и в списке контейнеров, завершивших работу не «засоряет»...(docker ps -a).

Проверим возможность подключения и выполнения в запущенном контейнере своих команд. Снова запустим (docker run --detach --rm --name sleep ubuntu sleep 300), (docker ps) и выполним команду (docker exec -ti sleep bash). Посмотрим перечень процессов (ps -ax).

Для остановки работы экземпляра контейнера, достаточно ввести (docker stop sleep). После остановки, можно удалить экземпляр (docker rm sleep)



Проведем изменения в нашем контейнере:

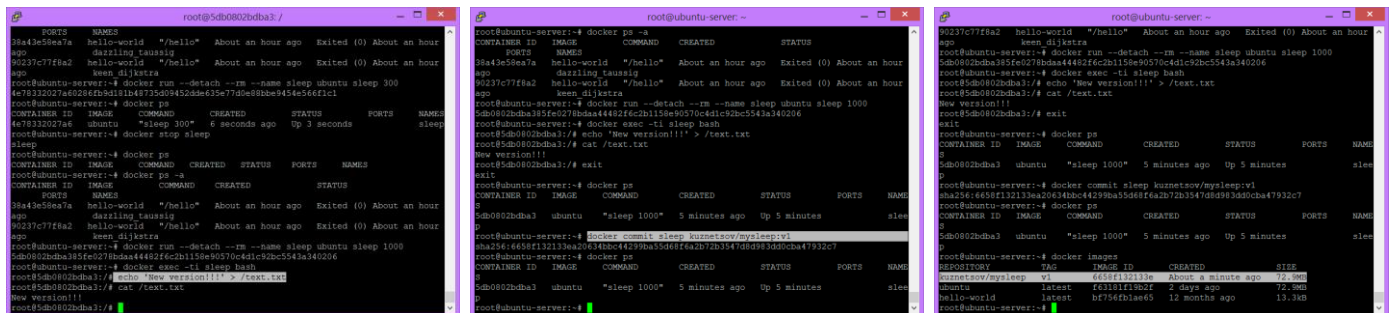
```
docker run --detach --rm --name sleep ubuntu sleep 1000
```

```
docker exec -ti sleep bash
```

```
echo 'New version!!!' > /text.txt
```

Для создания имиджа «mysleep:v1» с текущим состоянием контейнера «sleep», вводим (docker commit sleep kuznetsov1974/mysleep:v1). Имя должно включать имя пользователя на docker HUB «kuznetsov1974».

Проверим (docker images), имидж появился в списке...



```
root@5db0802bda3:/# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
5db0802bda3    hello-world    "hello"   About an hour ago   Exited (0)   About an hour ago   hello-world

root@5db0802bda3:/# docker run --detach --rm --name sleep ubuntu sleep 1000
5db0802bda3

root@5db0802bda3:/# docker exec -ti sleep bash
root@5db0802bda3:/# echo 'New version!!!' > /text.txt
New version!!!
root@5db0802bda3:/# cat /text.txt
New version!!!

root@5db0802bda3:/# docker commit sleep kuznetsov/mysleep:v1
sha256:6458f132133a208348b04429ba55d68f6a2b72b3547d8d93d0c8a47932c7
root@5db0802bda3:/# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
5db0802bda3    ubuntu    "sleep 1000"   5 minutes ago   Up 5 minutes   sleep

root@5db0802bda3:/# docker images
REPOSITORY    TAG       IMAGE ID       CREATED      SIZE
kuznetsov/mysleep   latest    6458f132133a   About 4 minutes ago   72.5MB
hello-world     latest    bf756fab65     12 months ago       13.4KB
```

3. Используя Dockerfile, собрать связку nginx + PHP-FPM в одном контейнере.

Почистим все (docker system prune -a) - глубокая чистка (остановленные контейнеры и все имиджи). Если контейнер запущен, то имидж не удаляется. Скачаем пакет с «nginx» (docker run --detach --name mynginx ubuntu sleep 1000).

Сделаем некоторые настройки. Создаем папку (mkdir nginx). Перейдем в папку (cd nginx).

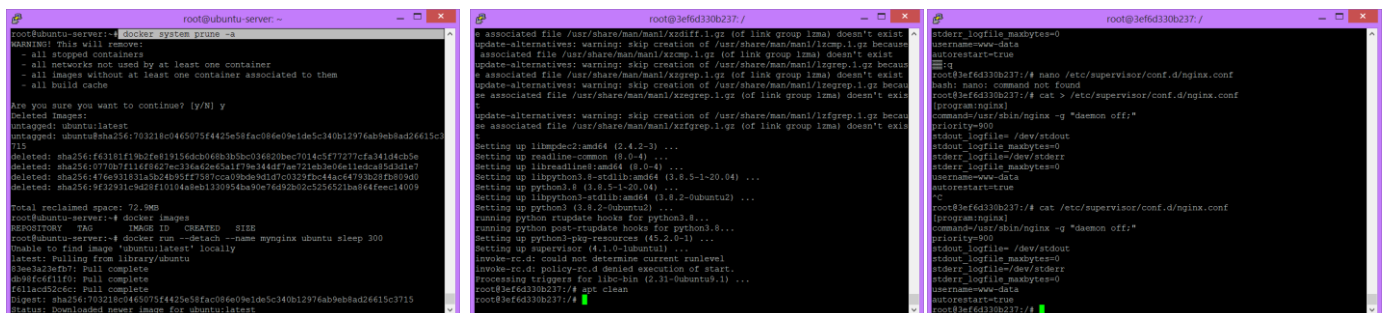
Скачиваем конфигурационный файл для «supervisor»:

wget <https://github.com/okteto/supervisor-nginx/raw/master/nginx.conf>

Подключимся к контейнеру «mynginx» (docker exec -ti mynginx bash), обновим в контейнере (apt update) и проинсталлируем «nginx» (apt install nginx). Поставим службу «supervisor» (apt install supervisor) - менеджер пользовательских процессов. И почистим систему (apt clean) - удаляет apt кеш(пакеты), независимо от возраста или необходимости.

Создадим файл с конфигурацией для «supervisor» (скачали ранее).

```
[program:nginx]
command=/usr/sbin/nginx -g "daemon off;"
priority=900
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
username=www-data
autorestart=true
```

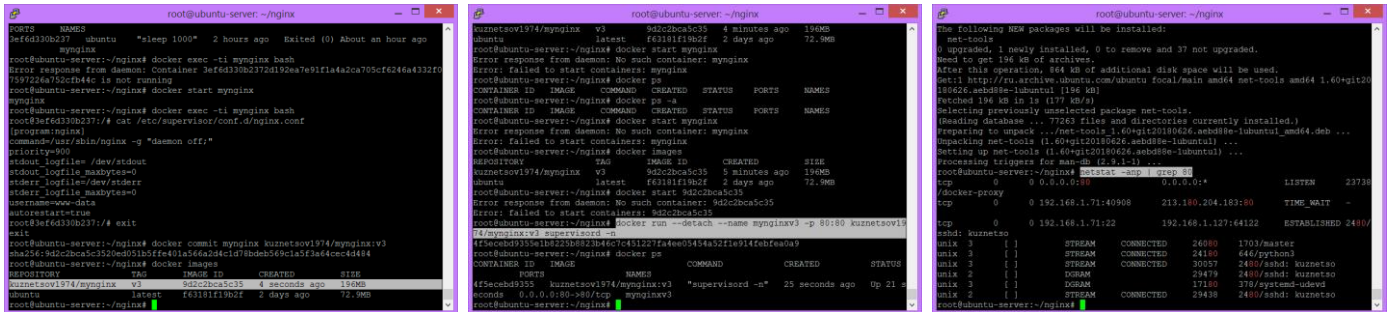


```
root@5db0802bda3:/# docker system prune -a
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one container
- all images without at least one container associated to them
- all build cache
Are you sure you want to continue? [y/N] y
Deleted images:
untagged: ubuntu:latest
untagged: sha256:703218c04650754425e58fac08609e1de5c340b12976ab9ebad26415c3715
Deleted: sha256:63181f19b2f819354dc0468b3b5c036820bec7014c5f7277cfa341d4c65e
Deleted: sha256:0770b7b7f1f8627c336a2e3a1f79e34d7ae721eb3e06e1edca553d1e7
Deleted: sha256:47693181ab44b45f970c9b086d4d7c023f0c44c6d4f93b2b32b39d0
Deleted: sha256:9f32931c9d89f10194a8bb1330954ba90e76d92b02c3256321ba864fnc14009
Total reclaimed space: 72 MB
root@5db0802bda3:/# docker run --detach --name mynginx ubuntu sleep 1000
5db0802bda3
root@5db0802bda3:/# docker exec -ti mynginx bash
root@5db0802bda3:/# apt update
Get:1 http://deb.debian.org/debian bookworm InRelease [156 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [51.6 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.1 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [9329 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [6948 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [28.7 kB]
Fetched 95.1 kB in 1s (10.5 kB/s)
Reading package lists... Done
root@5db0802bda3:/# apt install nginx supervisor
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nginx is already the newest version (1:1.24.0-2+b1).
supervisor is already the newest version (2:2.21.0-3+b1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@5db0802bda3:/# cat /etc/supervisor/conf.d/nginx.conf
[program:nginx]
command=/usr/sbin/nginx -g "daemon off;"
priority=900
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
username=www-data
autorestart=true
```


Сделаем имидж (docker commit mynginx kuznetsov1974/mynginx:v3). Убедимся, что имидж появился (docker images).

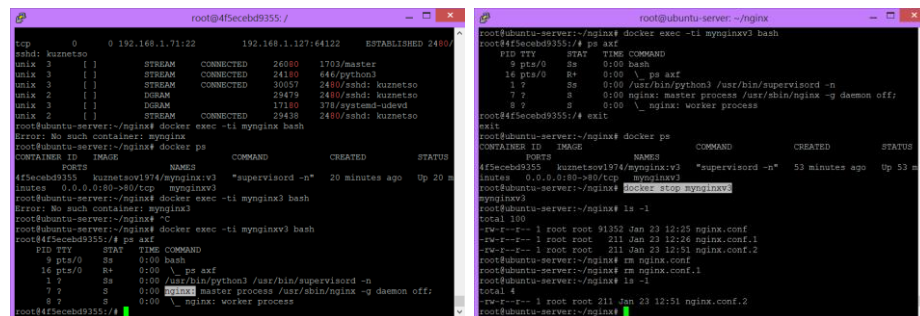
Остановим имеющийся экземпляр, удалим его и запустим наш имидж под именем «mynginxv3» (docker run --detach --name mynginxv3 -p 80:80 kuznetsov1974/mynginx:v3 supervisor -n). При этом, мы после запуска освободили консоль «--detach» и пробросили порты «-p 80:80». Слева находится локальный порт (порт нашей машины), а справа — порт контейнера.

Чтобы убедиться в работе, установим (apt install net-tools) и запустим (netstat -anp | grep 80).



Подключимся к экземпляру (docker exec -ti mynginxv3 bash) и выведем активные процессы (ps axf) – цель достигнута...

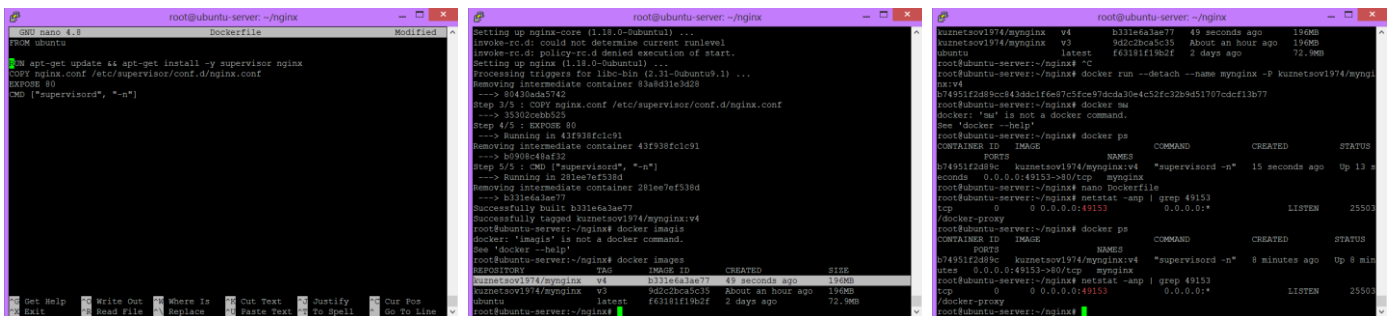
Останавливаем экземпляр (docker stop mynginxv3)...



Теперь, перейдем к самому заданию, т.е. используя «Dockerfile» ... Скачиваем Docker (wget <https://github.com/okteto/supervisor-nginx/raw/master/Dockerfile>). Добавим настройку порта «EXPOSE 80».

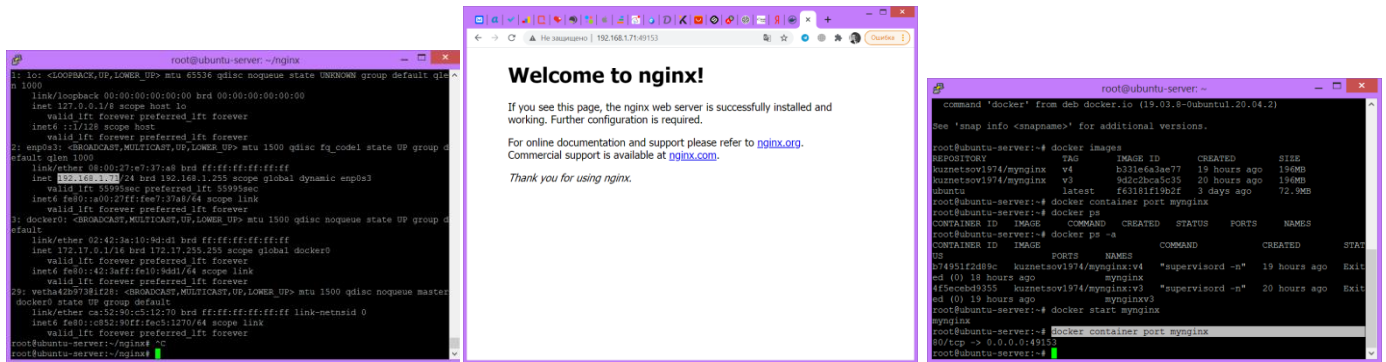
Соберем новый имидж «mynginx:v4» (docker build -t kuznetsov1974/mynginx:v4 .) используя файл «Dockerfile» в текущей директории «.». Убедимся, что Docker сделал в автоматическом режиме все тоже самое, что и мы выше «вручную». Запустим (docker images) – появился новый имидж «kuznetsov1974/mynginx v4».

Запустим наш имидж под именем «mynginxv4» (docker run --detach --name mynginx -P kuznetsov1974/mynginx:v4). При этом, мы после запуска освободили консоль «--detach» и пробросили порт «-P» как указано в настройках «49153->80». Выполним (netstat -anp | grep 49153)



Проверим через браузер «192.168.1.71:49153» - работает (Docker проху перебросил нас внутрь контейнера).

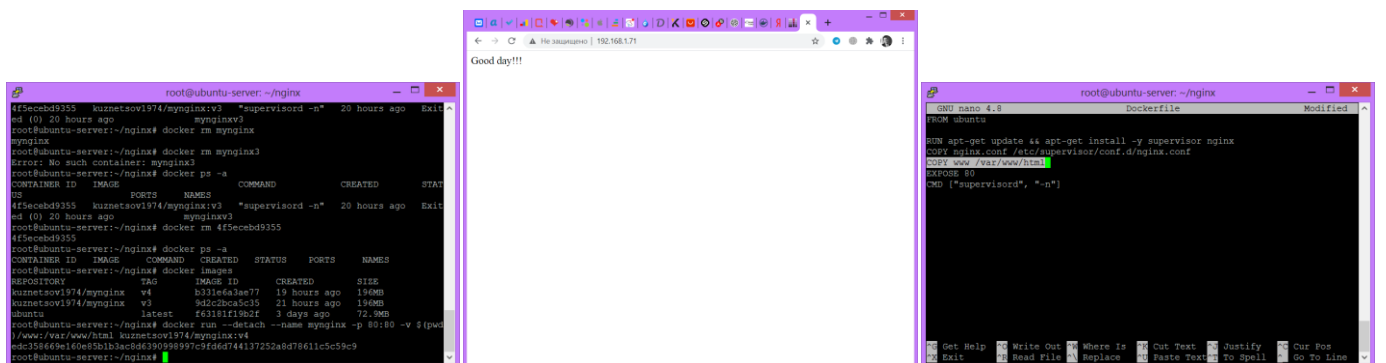
Кстати, порт контейнера можно посмотреть командой (docker container port mynginx)



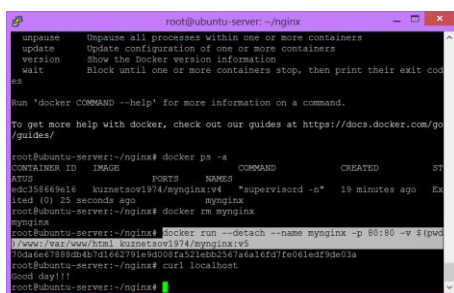
Теперь, подключим локальную папку. Создаем локальный каталог для последующей трансляции его на сайте (mkdir www). Остановим контейнер (docker stop mynginx), удалим (docker rm mynginx) и снова запустим с др. параметрами (docker run --detach --name mynginx -p 80:80 -v \$(pwd)/www:/var/www/html kuznetsov1974/mynginx:v4). При этом, обращаю внимание на то, что мы смонтировали «-v» локальную папку и сможем менять контент сайта «на лету».

В папке создадим файл «index.html», запускающийся при открытии сайта (echo 'Good day!!!' > ./www/index.html). Запустим в браузере «192.168.1.71» и убедимся в успешной трансляции файла.

Не забудем и в dockerfile вставить команду копирования «COPY www /var/www/html» и создать новый имидж (docker build -t kuznetsov1974/mynginx:v5).



Снова остановим, удалим и запустим последнюю версию контейнера (docker run --detach --name mynginx -p 80:80 -v \$(pwd)/www:/var/www/html kuznetsov1974/mynginx:v5). Выполним (curl localhost).



Для того, чтобы заниматься разработкой, но не «резать по живому», достаточно не монтировать локальную папку (docker run --detach --name mynginx -p 80:80 kuznetsov1974/mynginx:v5). При этом, изменения в папке не будут транслироваться вовне.