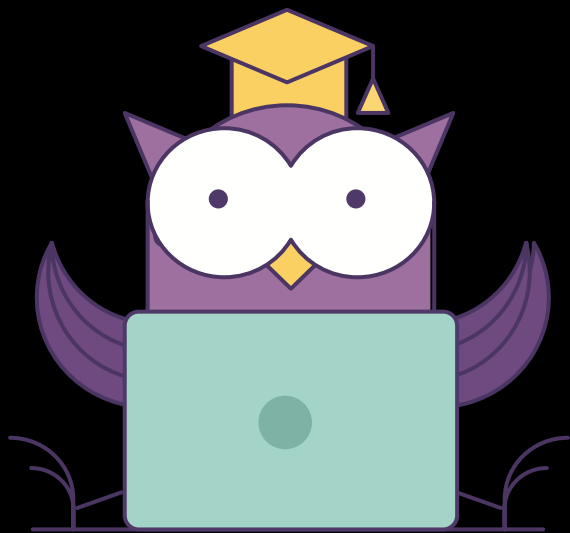





ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте  если все хорошо

Не забыть включить запись!

Docker

Python QA Engineer



Правила вебинара

Паузы между блоками

Вопросы пишем в чат

Обсуждения в Slack

Цели вебинара

1. Что такое Docker?
2. Основы использования
3. Использование с Python

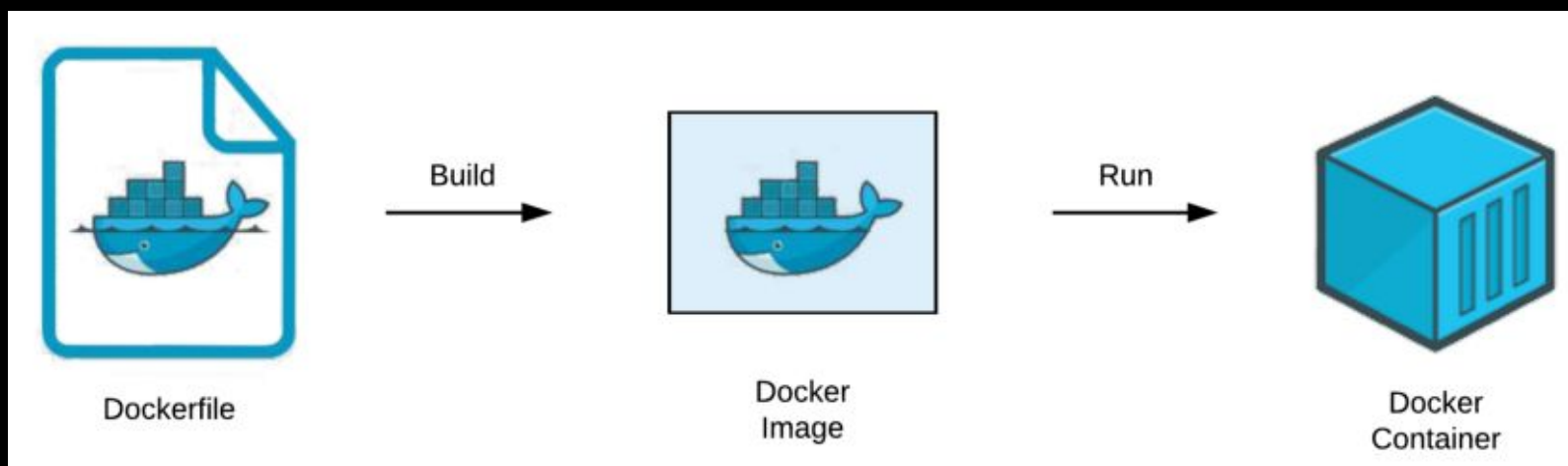
01

Что такое Docker

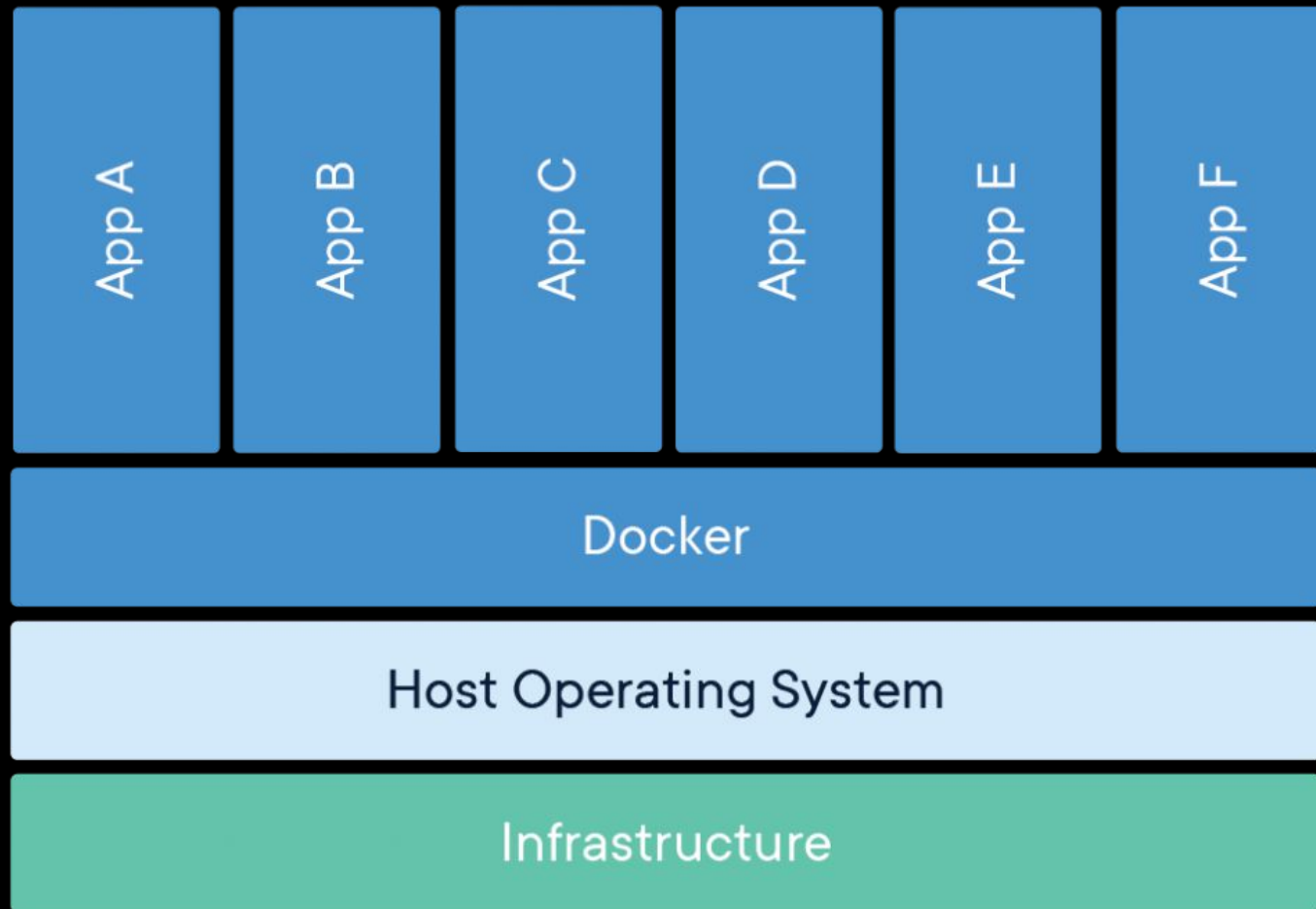
Что такое Docker?

Если коротко то это средство доставки программного обеспечения в формате специальных файлов **dockerfile** которые описывают как и из чего нужно собирать образ **image** который потом можно будет запустить в контейнере.

Из **images** можно создать контейнеры на любом физическом сервере, где установлен docker. Если **image** это совокупность ресурсов, то **container** это уже сама созданная машина, которую можно запускать, останавливать и параметризировать.



Что такое Docker?



Вопросы



02

ОСНОВЫ ИСПОЛЬЗОВАНИЯ

Репозиторий

https://github.com/konflic/python_qa_docker

ОСНОВЫ ИСПОЛЬЗОВАНИЯ

`docker run [--rm] hello-world` - проверка что docker работает
`docker run = docker create + docker start [-a, --attach] container-id`
`docker create` - создать контейнер
`docker start [--attach, -a, -i] container-id` - запустить контейнер
`docker ps [-a, --all]` - посмотреть контейнеры в системе
`docker stop/kill container-id` - остановить или убить контейнер

ОСНОВЫ ИСПОЛЬЗОВАНИЯ

`docker images` - посмотреть images в системе

`docker system prune` - удаляет только контейнеры (-a + images)

`docker build [-t name] dockerfile` - создать image из файла `dockerfile`

`docker logs container-id` - получить логи из контейнера

`docker run -it container-id` - интерактивный режим

`docker cp container-id:/... .` - скопировать что-нибудь из контейнера

`docker run -p local:container container-id` - прокинуть порты наружу

ОСНОВЫ ИСПОЛЬЗОВАНИЯ

`docker images` - посмотреть images в системе

`docker system prune` - удаляет только контейнеры (-a + images)

`docker build [-t name] dockerfile` - создать image из файла `dockerfile`

`docker logs container-id` - получить логи из контейнера

`docker run -it container-id` - интерактивный режим

`docker cp container-id:/... .` - скопировать что-нибудь из контейнера

`docker run -p local:container container-id` - прокинуть порты наружу

Dockerfile instructions

FROM - базовый образ из которого будет создаваться итоговый

WORKDIR - устанавливает директорию для файловых инструкций

ADD, COPY - команды для копирования и добавления файлов

RUN - выполняет команду в текущем образе и формирует слой

CMD - установить команду по умолчанию для контейнера

ENTRYPOINT - определяют команду при запуске контейнера

ENV - выставляет переменные окружения в контейнере

VOLUME - создать директорию на хосте и установить её в контейнер

EXPOSE - указать контейнеру что этот порт нужно открыть

LABEL - установка лейблов на контейнер

Важно запомнить

1. Контейнер это запущенный инстанс image после работы которого будет сохранено определенное состояние
2. Внутри каждого контейнера после сборки выполняется какая-то команда, которую можно увидеть в разделе `docker ps`
3. Контейнер создается вместе с этой командой по умолчанию, но её можно перезаписывать `docker run [command]`
4. `CMD` - перезаписывается, `ENTRYPOINT` - дополняется, можно их комбинировать
5. `ADD` - может добавлять не только локальные файлы, но и ссылки
6. Перед тем как собирать какой-то образ, нужно сначала погуглить вероятно большая часть необходимого функционала уже реализована.

Вопросы



03

Docker + Python = 

Важно запомнить

1. Для того чтобы не таскать лишние файлы использовать файл `.dockerignore`
2. Не изобретать велосипедов а просто подыскать подходящий базовый образ https://hub.docker.com/_/python
3. <https://github.com/SeleniumHQ/docker-selenium>

вопрос - опрос

<https://otus.ru/polls/12828/>





gifik.net