OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема вебинара

Docker - основы контейнеризации Part 2

Шиков Станислав

Начальник отдела автоматизации ООО “Кодер”

Эл. почта stenlav@mail.ru

Правила вебинара

Активно
участвуем

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Маршрут вебинара

- Volumes and Networks
- Закрепление навыков работы с Docker CLI
- Установка и настройка собственного Docker-registry
- ENTRYPOINT vs CMD
- Рефлексия

Цели вебинара

После занятия вы сможете

1. Научиться управлять жизненным циклом Docker
2. Понять основные принципы работы Docker-registry
3. Применить на практике навыки управления Docker

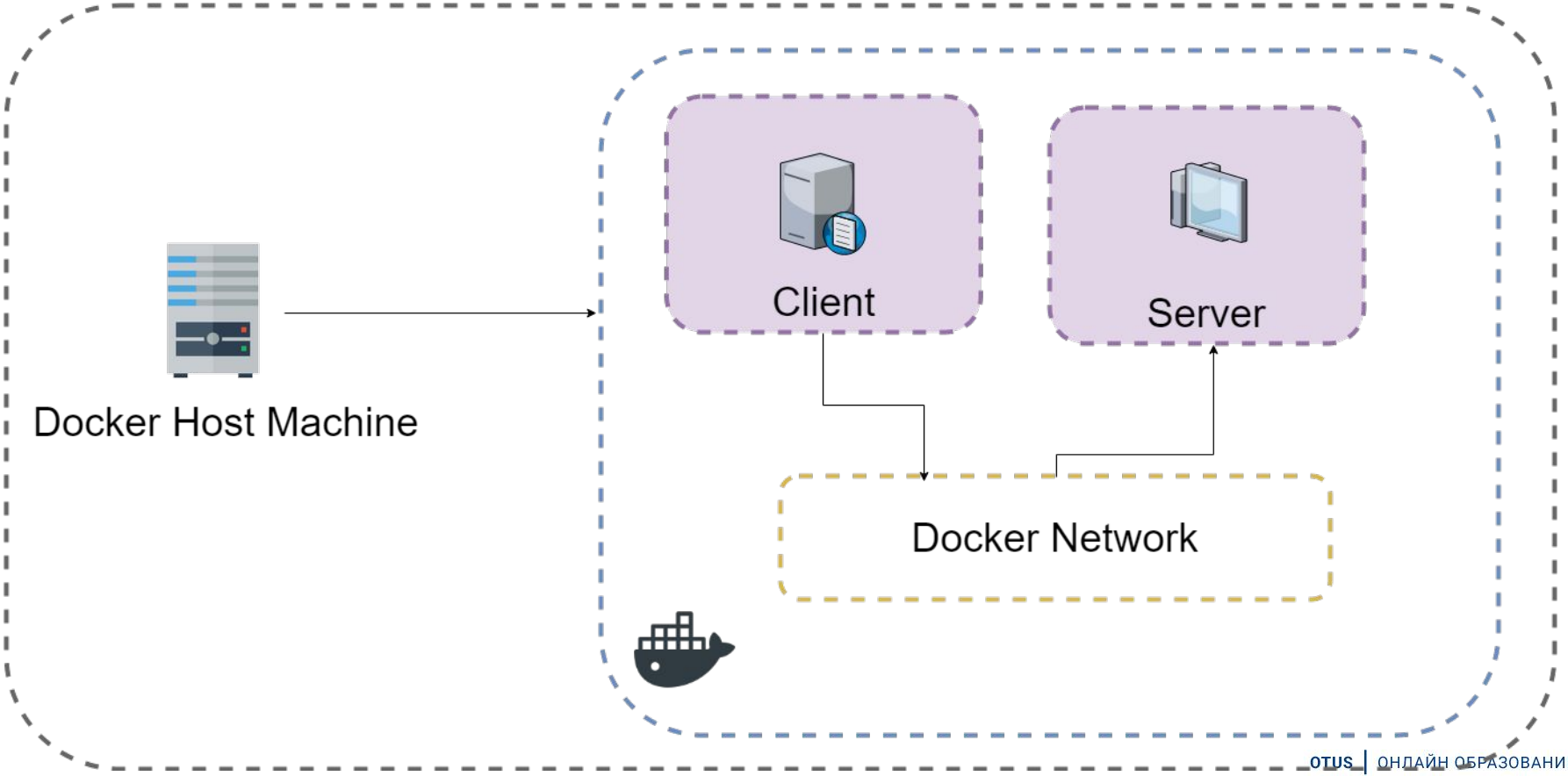
Смысл

Зачем вам это уметь

1. В 2019 г. эксперты Gartner говорили, что 30% организаций по всему миру используют контейнерные приложения, а в 2022 г. эта цифра достигает 70%.
2. Размер мирового рынка коммерческого контейнерного программного обеспечения будет расти в 2018-2023 гг. в среднем на 30% ежегодно, превысив к концу периода $1,6 млрд — такой прогноз дается в отчете Technology Multi-Tenant Server Software Market Tracker аналитиков IHS Markit.
3. Знание в области контейнеризации поможет в работе, как программиста так и системного администратора. Также этот навык востребован в большинстве современных IT вакансий

Let’s Go

Docker Networks



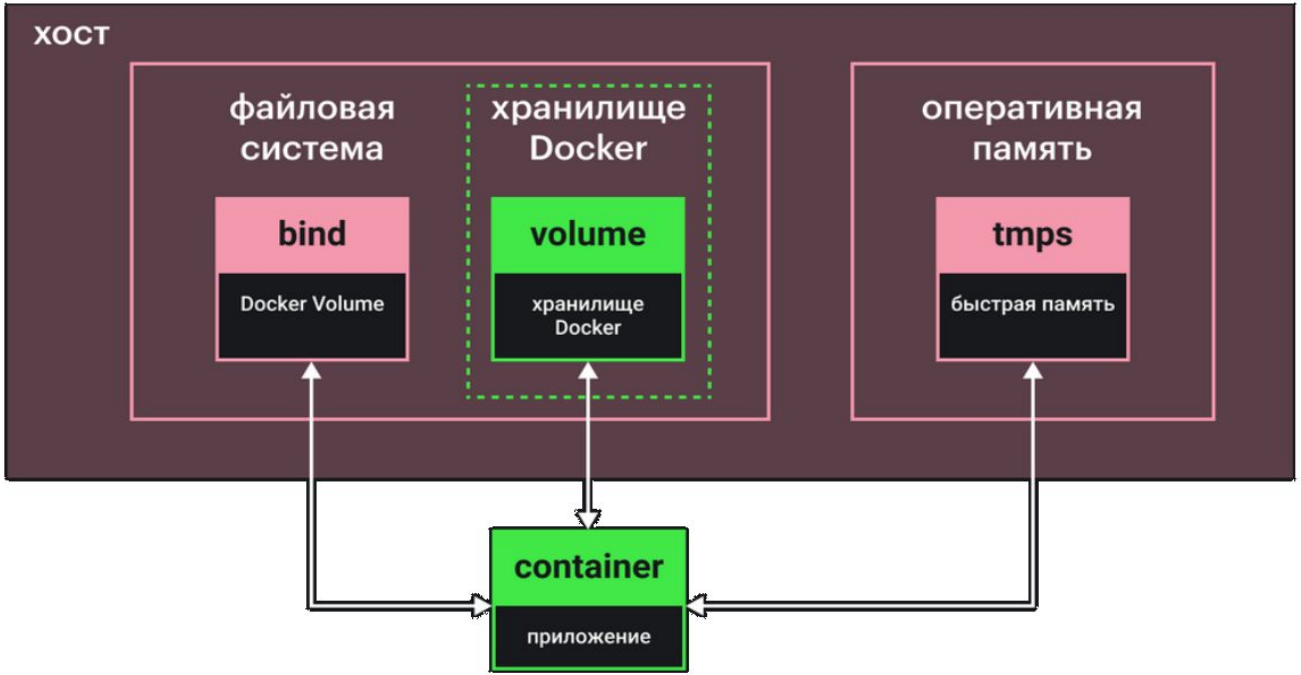
Типы сетевых драйверов Docker

- **Bridge network** при запуске Docker автоматически создается сеть типа мост по умолчанию. Недавно запущенные контейнеры будут автоматически подключаться к нему. Вы также можете создавать пользовательские настраиваемые мостовые сети. Пользовательские мостовые сети превосходят сетевые мосты по умолчанию.
- **Host network :** удаляет сетевую изоляцию между контейнером и хостом Docker и напрямую использует сеть хоста. Если вы запускаете контейнер, который привязывается к порту 80, и вы используете хост-сеть, приложение контейнера доступно через порт 80 по IP-адресу хоста. Означает, что вы не сможете запускать несколько веб-контейнеров на одном хосте, на одном и том же порту, так как порт теперь является общим для всех контейнеров в сети хоста.
- **None network :** в сети такого типа контейнеры не подключены ни к одной сети и не имеют доступа к внешней сети или другим контейнерам. Итак, эта сеть используется, когда вы хотите полностью отключить сетевой стек в контейнере.
- **Overlay network :** Создает внутреннюю частную сеть, которая охватывает все узлы, участвующие в кластере swarm. Таким образом, оверлейные сети облегчают обмен данными между сервисом Docker Swarm и автономным контейнером или между двумя автономными контейнерами на разных демонах Docker.
- **Macvlan network :** Некоторые приложения, особенно устаревшие приложения, отслеживающие сетевой трафик, ожидают прямого подключения к физической сети. В такой ситуации вы можете использовать сетевой драйвер Macvlan для назначения MAC-адреса виртуальному сетевому интерфейсу каждого контейнера, что делает его физическим сетевым интерфейсом, напрямую подключенным к физической сети.

Практическая часть работы с сетью Docker

- *docker run -it -d --name c1 alpine ash*
- *docker run -it -d --name c2 alpine ash*
- *docker exec -it c1 sh –c “ip a”*
- *docker exec -it c2 sh –c “ip a”*
- *docker attach c1*
- *ping –c 2 x.x.x.x*
- *docker network inspect bridge*
- *docker network ls*
- *docker network create my_net или docker network create --driver bridge my_net2*
- *docker run -it -d --name A1 --network my_net2 alpine ash*
- *docker run -it -d --name A2 --network my_net2 alpine ash*
- *docker run -it -d --name A3 --network my_net2 alpine ash*
- *ping -c 2 A1*
- *docker run -it -d --network host --name nginx nginx*

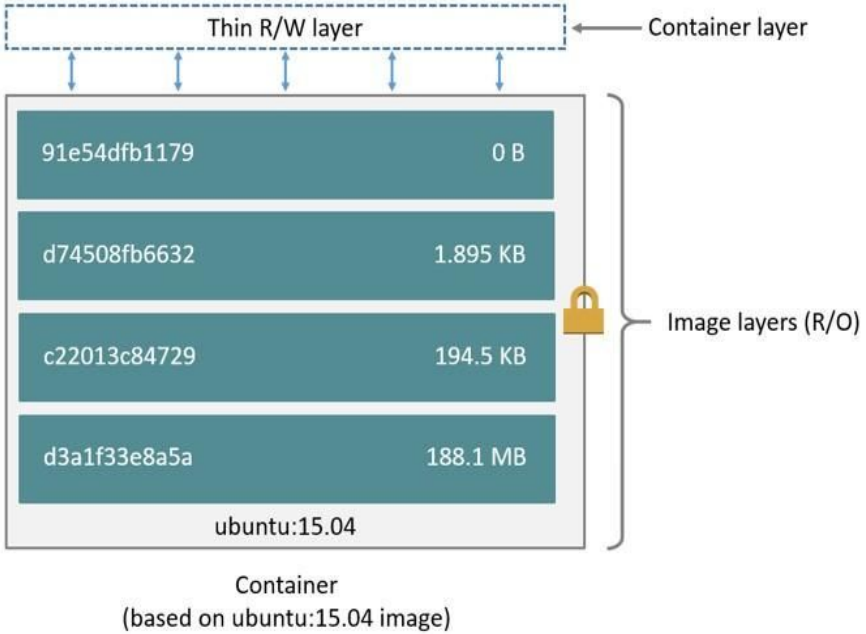
Хранение данных в Docker

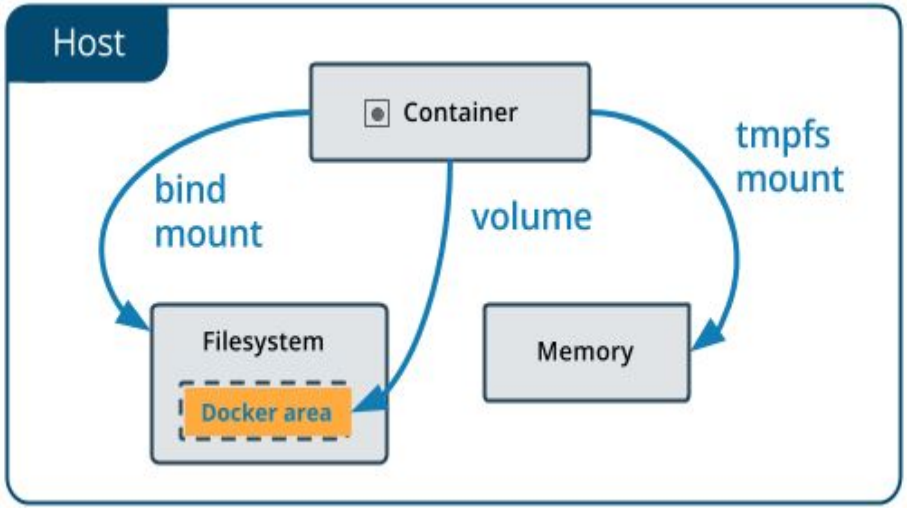


Какие данные стоит хранить постоянно?

- Конфигурационные файлы
- Скачиваемые для использования плагины
- Базы данных
- Настройки пользователей
- Артефакты работы приложения

Постоянное хранение данных





Drivers

overlay2	overlay2 является предпочтительным драйвером хранилища для всех поддерживаемых в настоящее время дистрибутивов Linux и не требует дополнительной настройки
overlay	Устаревший overlay драйвер использовался для ядер, которые не поддерживали функцию «multiple-lowerdir»
Btrfs, ZFS	В btrfs и zfs драйверах имеются дополнительные опции, такие как создание «снапшотов», но требуют большего обслуживания и настройки
aufs	Aufs Драйвер запоминающего устройства был предпочтительным драйвер для хранения Докер 18.06 и старше, при работе на Ubuntu 14.04 на ядре 3.13 , которая не имела поддержки overlay2. Однако в текущих версиях Ubuntu и Debian теперь есть поддержка драйвера overlay2, который является рекомендуемым драйвером
devicemapper	Использовался для производственных сред, но имел очень низкую производительность. Devicemapper был рекомендованным драйвером хранилища для CentOS и RHEL, поскольку их версия ядра не поддерживала overlay2. Однако в текущих версиях CentOS и RHEL есть поддержка overlay2

Практическая часть работы с томами Docker

Связанные папки (bind mounts)

```
docker run -d \
  -it \
  --name devtest \
  -v "$(pwd)"/target:/app \
  node:lts
```

```
docker run -d \
  -it \
  --name devtest \
  --mount
type=bind,source="$(pwd)"/target,target=/app \
  node:lts
```

Тома (volumes)

С опцией -v

```
docker run -d \
  --name devtest \
  -v my-vol:/app \
  node:lts
```

С опцией --mount

```
docker run -d \
  --name devtest \
  --mount source=my-vol,target=/app \
  node:lts
```

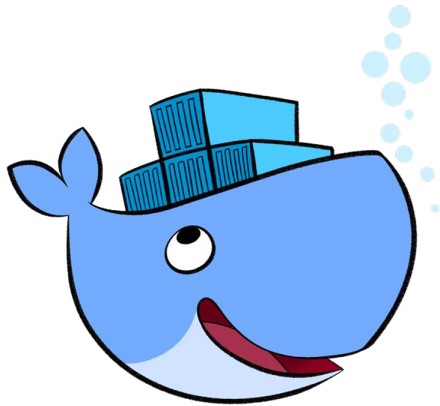
From Dockerfile

```
FROM ubuntu:latest
RUN mkdir /data
WORKDIR /data
RUN echo "Hello from Volume" > test
VOLUME /data
```

Хранение в оперативной памяти

```
docker run -d \
  -it \
  --name tmptest \
  --mount type=tmpfs,destination=/app \
  node:lts
```

```
docker run -d \
  -it \
  --name tmptest \
  --tmpfs /app \
  node:lts
```



ENTRYPOINT vs CMD

Инструкция CMD предоставляет Docker команду, которую нужно выполнить при запуске контейнера. Результаты выполнения этой команды не добавляются в образ во время его сборки. Например, это может быть bash скрипт, или запуск исполняемого файла.

Вот ещё кое-что, что нужно знать об инструкции CMD:

1. В одном файле Dockerfile может присутствовать лишь одна инструкция CMD. Если в файле есть несколько таких инструкций, система проигнорирует все кроме последней.
2. Инструкция CMD может иметь exec-форму. Если в эту инструкцию не входит упоминание исполняемого файла, тогда в файле должна присутствовать инструкция ENTRYPOINT. В таком случае обе эти инструкции должны быть представлены в формате JSON.
3. Аргументы командной строки, передаваемые docker run, переопределяют аргументы, предоставленные инструкции CMD в Dockerfile.

ENTRYPOINT vs CMD

Инструкция ENTRYPOINT позволяет задавать команду с аргументами, которая должна выполняться при запуске контейнера. Она похожа на команду CMD, но параметры, задаваемые в ENTRYPOINT, не перезаписываются в том случае, если контейнер запускают с параметрами командной строки.

Вместо этого аргументы командной строки, передаваемые в конструкции вида docker run my_image_name, добавляются к аргументам, задаваемым инструкцией ENTRYPOINT. Например, после выполнения команды вида docker run my_image bash аргумент bash добавится в конец списка аргументов, заданных с помощью ENTRYPOINT.

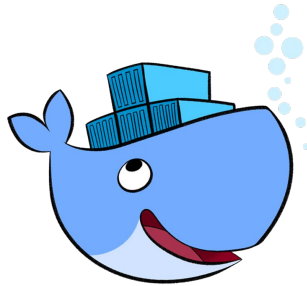
В документации к Docker есть несколько рекомендаций, касающихся того, какую инструкцию, CMD или ENTRYPOINT, стоит выбрать в качестве инструмента для выполнения команд при запуске контейнера:

1. Если при каждом запуске контейнера нужно выполнять одну и ту же команду — используйте ENTRYPOINT.
2. Если контейнер будет использоваться в роли приложения — используйте ENTRYPOINT.
3. Если вы знаете, что при запуске контейнера вам понадобится передавать ему аргументы, которые могут перезаписывать аргументы, указанные в Dockerfile, используйте CMD.

Документация Docker рекомендует использовать exec-форму ENTRYPOINT: ENTRYPOINT ["executable", "param1", "param2"].

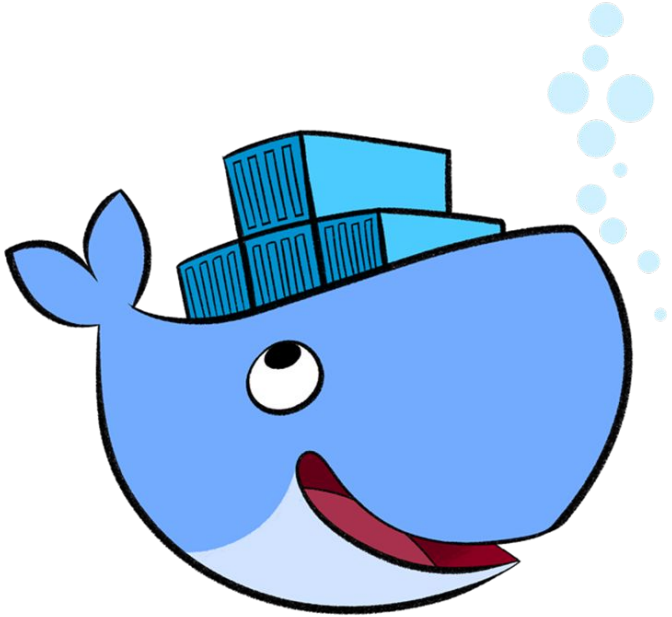
ENTRYPOINT vs CMD

https://github.com/nginxinc/docker-nginx/blob/master/Dockerfile-debian.template



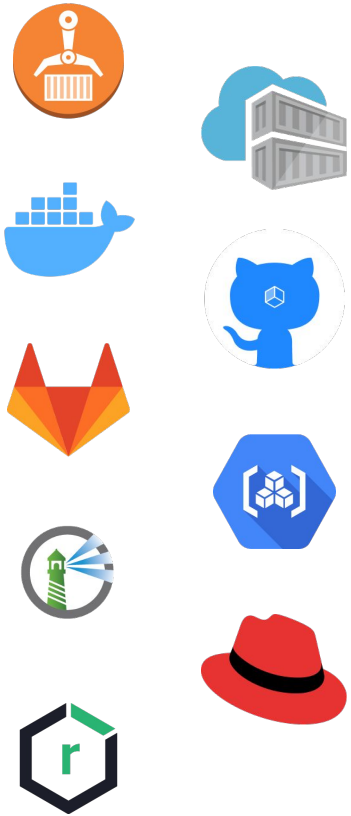
Что такое Docker-registry?

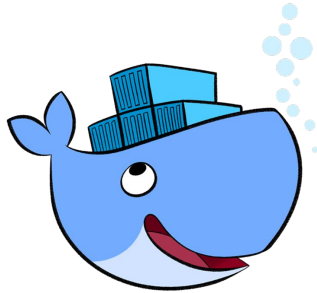
Это репозиторий, в котором хранятся образы. Когда разработчики создают приложения, они размещают свои образы в этих репозиториях, откуда их могут скачать другие люди. Есть публичные репозитории, например Docker Hub. А можно создать свой репозиторий, для использования внутри компании или команды.



Это всё Docker-registry?

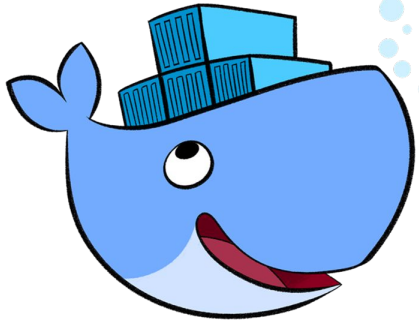
- Amazon Elastic Container Registry (ECR)
- Azure Container Registry (ACR)
- Docker Hub Container Registry
- GitHub Package Registry
- GitLab Container Registry
- Google Artifact Registry (GAR)
- Harbor Container Registry
- Red Hat Quay
- Sonatype Nexus Repository OSS





Установим свой docker-registry

Docker tips and tricks



1 Можно удалять образы не вводя полный хэш
docker rmi 7c9 2f 5cad удалит 3 образа

2 Старайтесь именовать вольюмы чтобы потом их было легче найти
docker run --name psql -d -p 5432:5432 -v psql:/var/lib/postgresql sameersbn/postgresql
docker volume ls

3 docker-compose down позволяет подчистить всё за собой. (networks, containers)
docker-compose down -v --rmi local

4 Удалить только остановленные контейнеры, не удаляя вольюмы
docker rm $(docker ps -aq)

5 Удалить все образы, которые в данный момент не используются в контейнерах
docker rmi $(docker images -q -f "dangling=true")

6 Получить локальный IP контейнера
docker inspect--format="{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}"

7 docker system prune -a -f –volumes - полная очистка неиспользуемых контейнеров, сетей, образов и при указании - вольюмов

Список материалов для изучения

1. **Docker install** - https://docs.docker.com/get-docker/
2. **Portainer** - https://docs.portainer.io/start/install-ce/server/docker
3. **Harbor** - https://goharbor.io/docs/2.0.0/install-config/run-installer-script/
4. Play with Docker

Рефлексия

Цели вебинара

Проверка достижения целей

1. Научиться управлять жизненным циклом Docker
2. Понять основные принципы работы Docker-registry
3. Применить на практике навыки управления Docker

Вопросы для проверки

По пройденному материалу всего вебинара

1. Какие состояния контейнера бывают?
2. Как отправить образ в docker registry? Как получить образ из приватного репозитория?
3. Как создать своё собственное docker-registry?
4. ENTRYPOINT? CMD? VOLUMES? BRIDGE? HOST?

Рефлексия

Насколько тема была для вас сложной?

Как будете применять на практике то,
что узнали на вебинаре?

Спасибо за внимание!

Приходите на следующие вебинары

Шиков Станислав Александрович

Руководитель отдела автоматизации ООО “Кодер”

Эл. почта stenlav@mail.ru