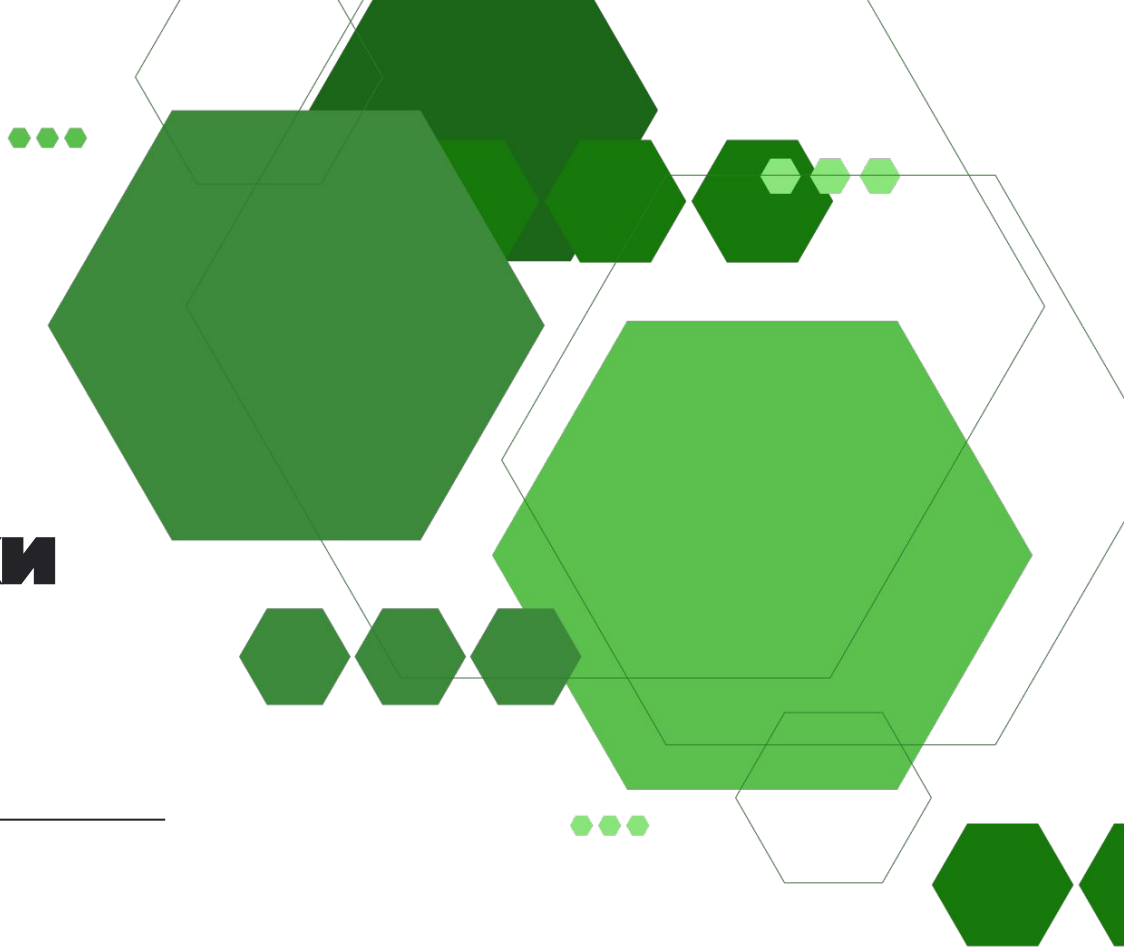


SKILLFACTORY

Вебинар: Система сборки Docker

Николай Юрьевич Мищенко

Ментор курса «DevOps-инженер»



Начинаем!

Меня хорошо видно и слышно?

Если **ДА**, поставьте в чат ++

Если **НЕТ**, напишите об этом в чат

Если проблемы наблюдаются только у Вас,
попробуйте перезайти в конференцию
по той же ссылке (выбрав “Войти с использованием звука компьютера”).



SKILLFACTORY



Николай Юрьевич Мищенко

DevOps Engineer at TechPlanet.pro

Ментор курса «DevOps-инженер»

Правила вебинара



**Отложите все
дела на
ближайший
час**



**Фиксируйте
важную
информацию
на вебинаре**



**Задавайте
вопросы в чат**







**Делитесь
мнением в
чате**



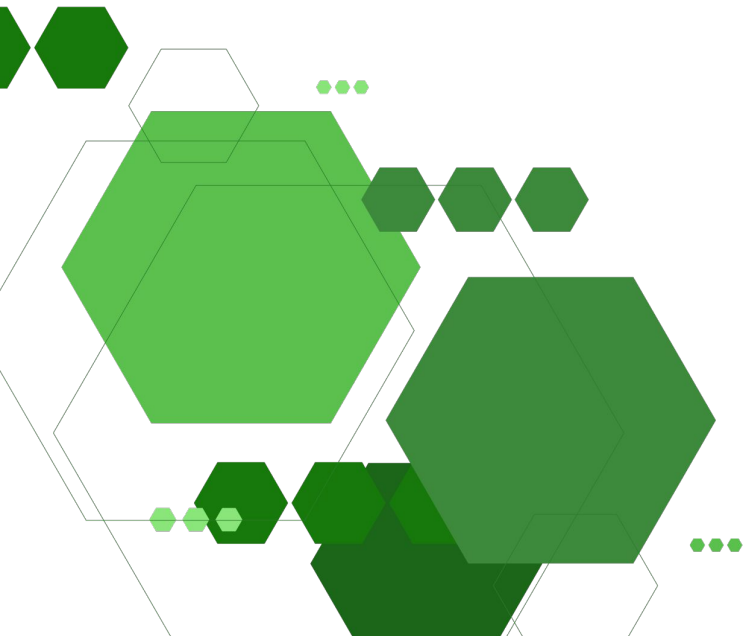
**Ура! Вы
превосходны!**

Запись вебинара будет выложена на платформе lms.skillfactory.ru

План вебинара

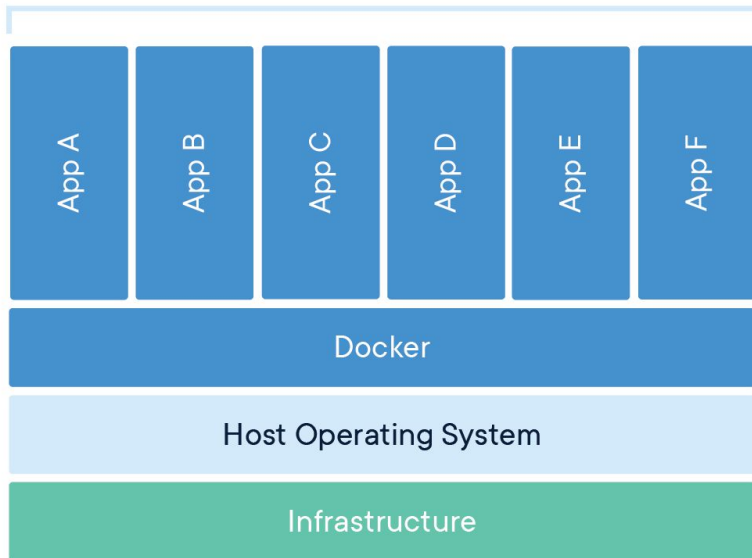
-  Основные понятия Докера.
-  Соберем простое приложение с помощью Docker в GitLab CI.
-  Поговорим о кэшировании.
-  Поговорим об оптимизации места.

ОСНОВНЫЕ ПОНЯТИЯ ДОКЕРА

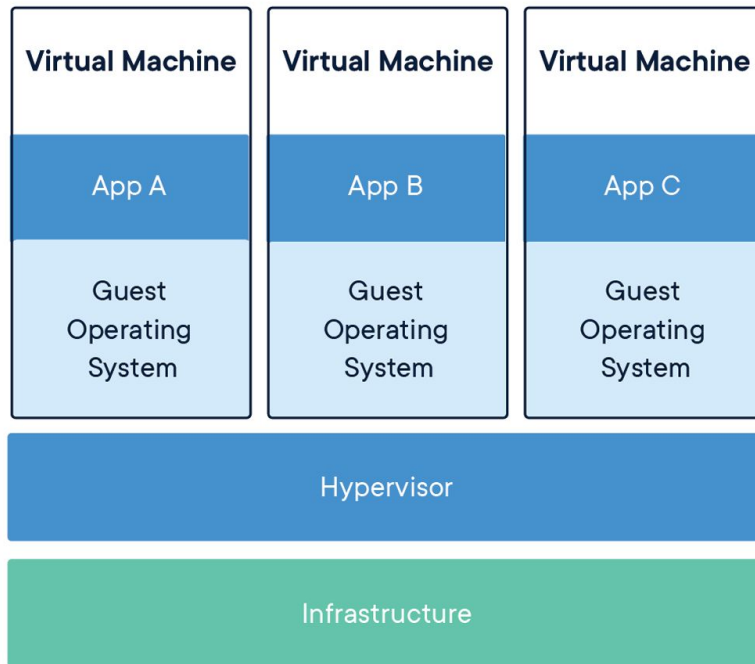


Docker vs Virtual Machines

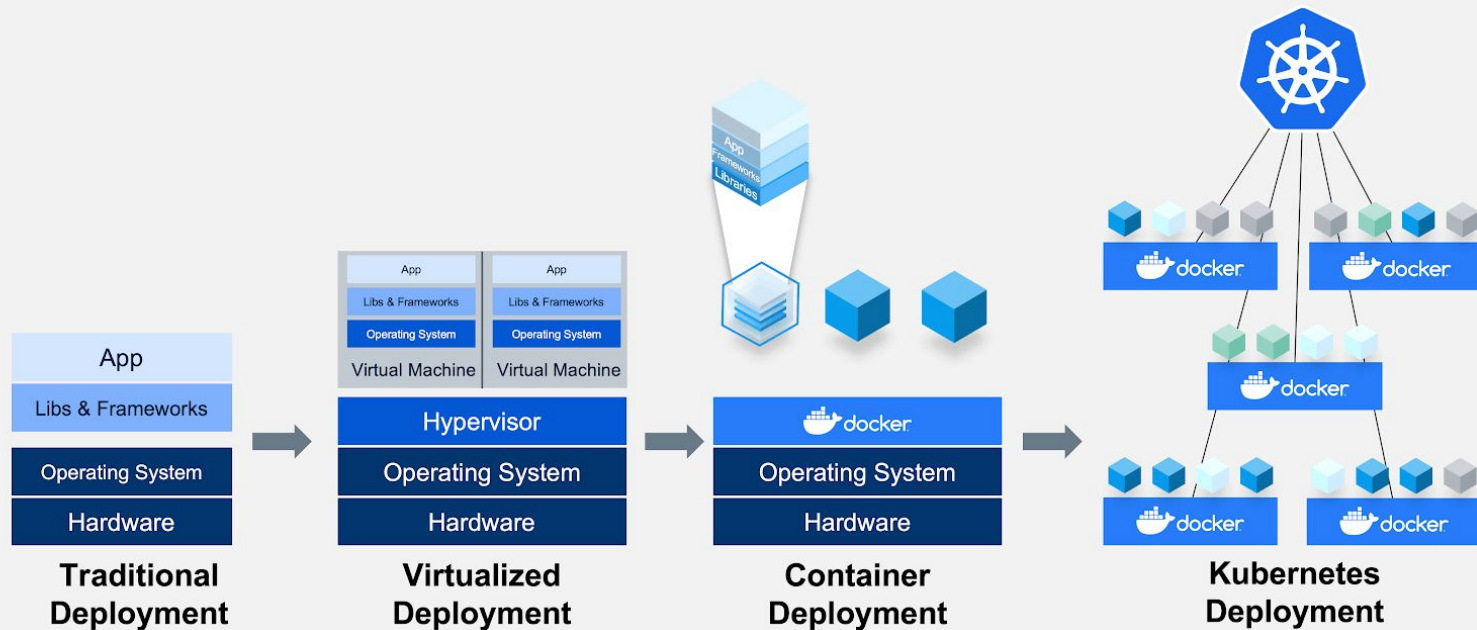
Containerized Applications



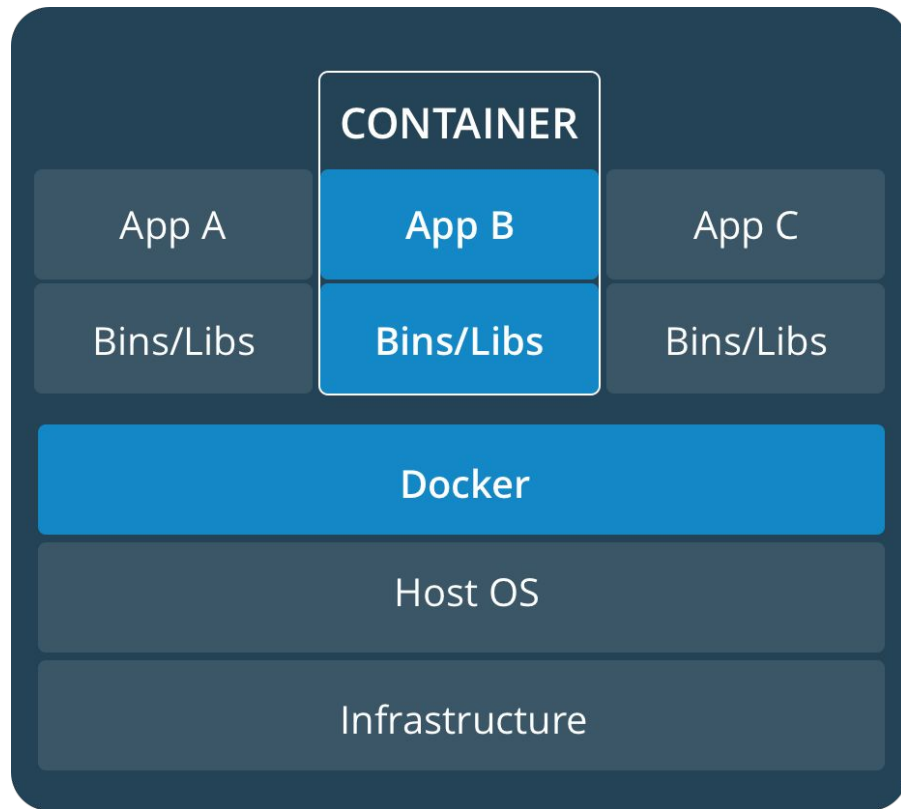
Virtual Machine



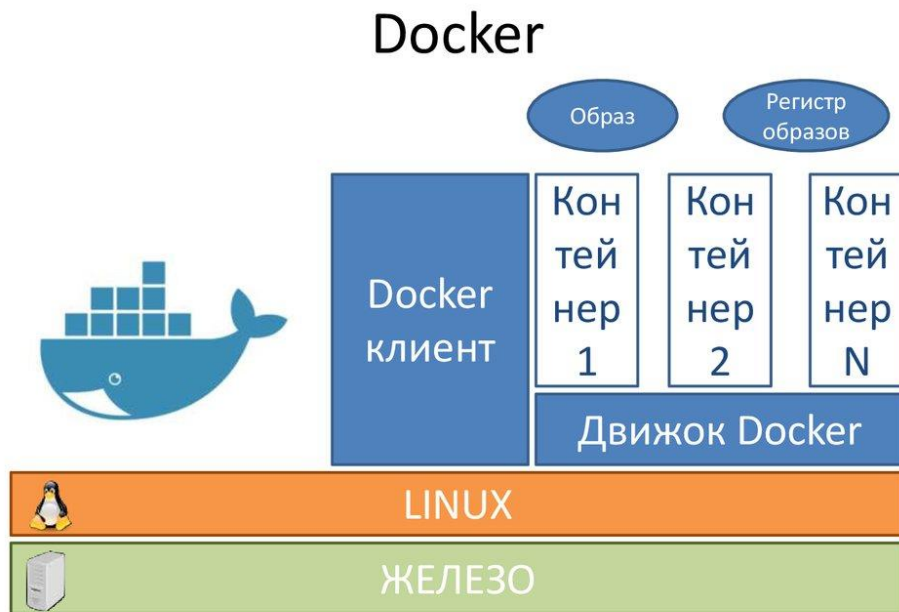
Comparing Containers and Virtual Machines



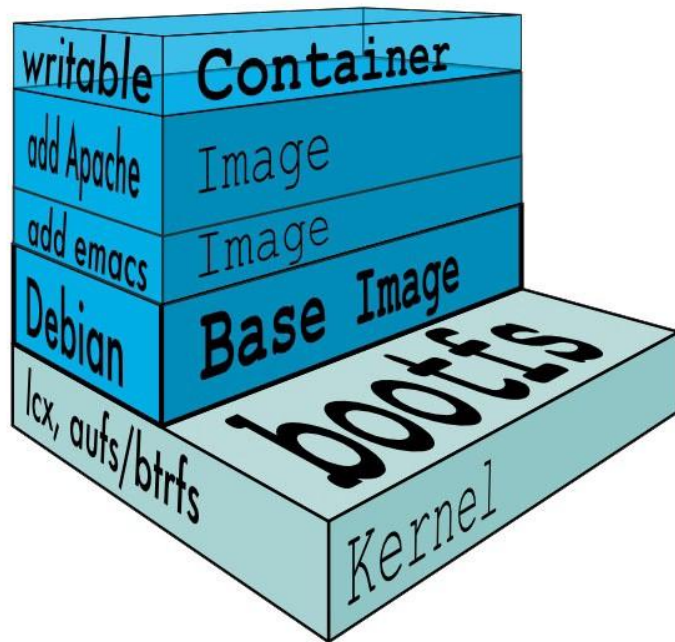
Container



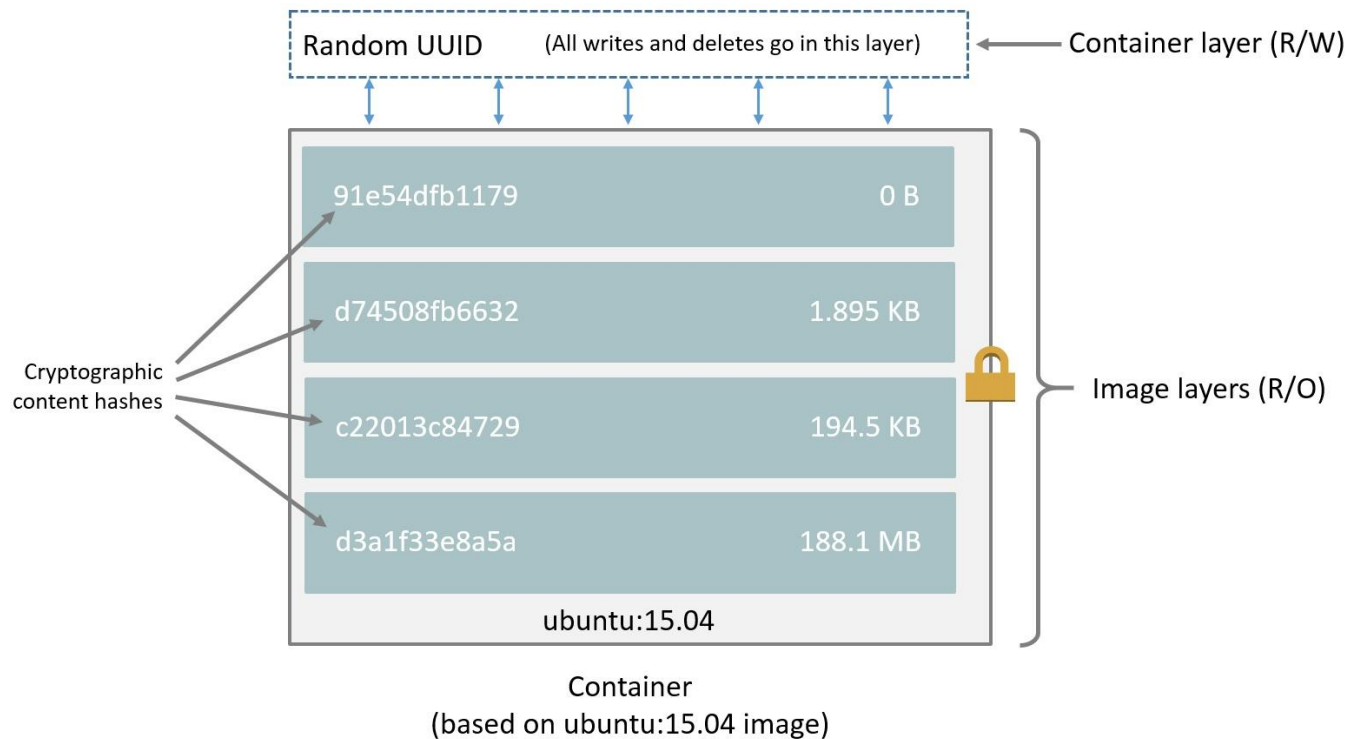
Container



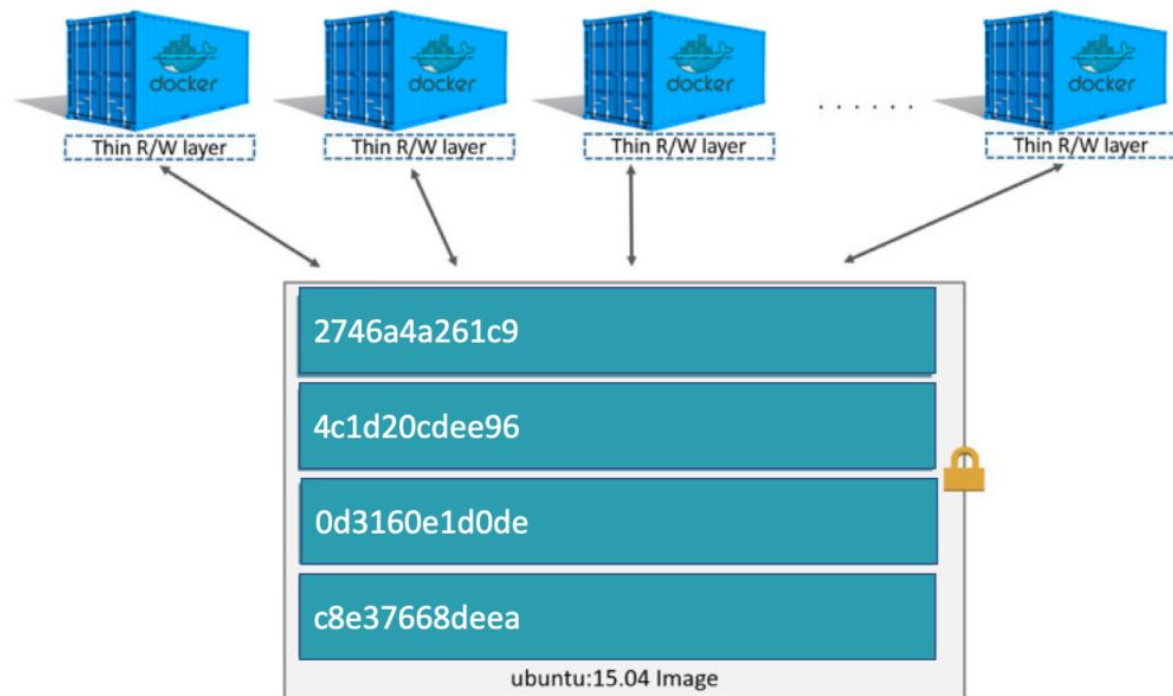
Docker Image Structure



Docker Image Structure

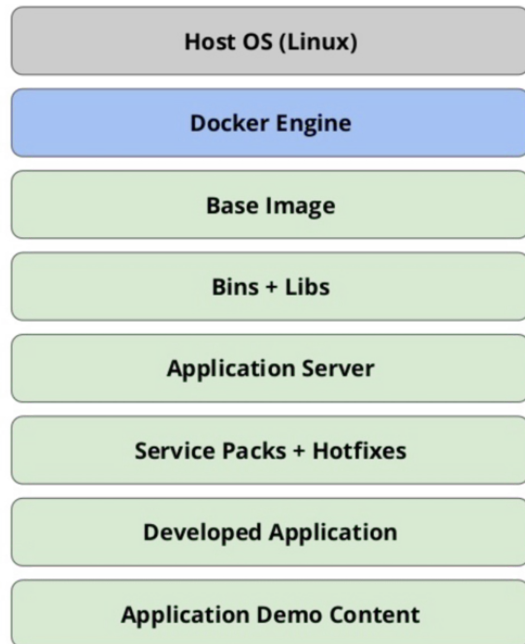


Docker Image Structure



Docker Image Structure

Layers of a Docker Image



Dockerfile (Image Definition)

```
# Base Image
FROM ubuntu:latest

# Install Java
RUN apt-get update && \
    apt-get install -y java7

# Download and unpack AEM
ADD http://repo.eggs.de/aem-quickstart.jar /opt
ADD http://repo.eggs.de/license.properties /opt
RUN java -jar /opt/aem-quickstart.jar -unpack

# Install Service Packs and Hotfixes
RUN mkdir /opt/crx-quickstart/install
ADD http://repo.eggs.de/aem-updates.zip
    /opt/crx-quickstart/install

# Install Custom Application
ADD http://repo.eggs.de/my-app.zip
    /opt/crx-quickstart/install

# Install Demo Content
ADD http://repo.eggs.de/my-demo-content.zip
    /opt/crx-quickstart/install
```

Dockerfile

```
FROM ubuntu:16.04
MAINTAINER John Doe <john.doe@example.com>

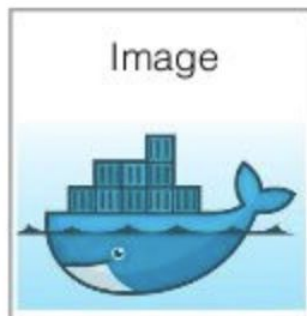
RUN apt-get update \
    && apt-get install -y \
    python3 \
    python3-dev \
    python3-pip \
    && pip install Flask

WORKDIR /app
COPY . /app
RUN python3 -m pip install -r requirements.txt

EXPOSE 5000
CMD ["python3", "app.py"]
```

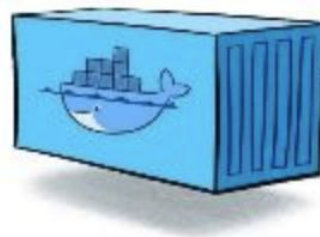
Dockerfile

build



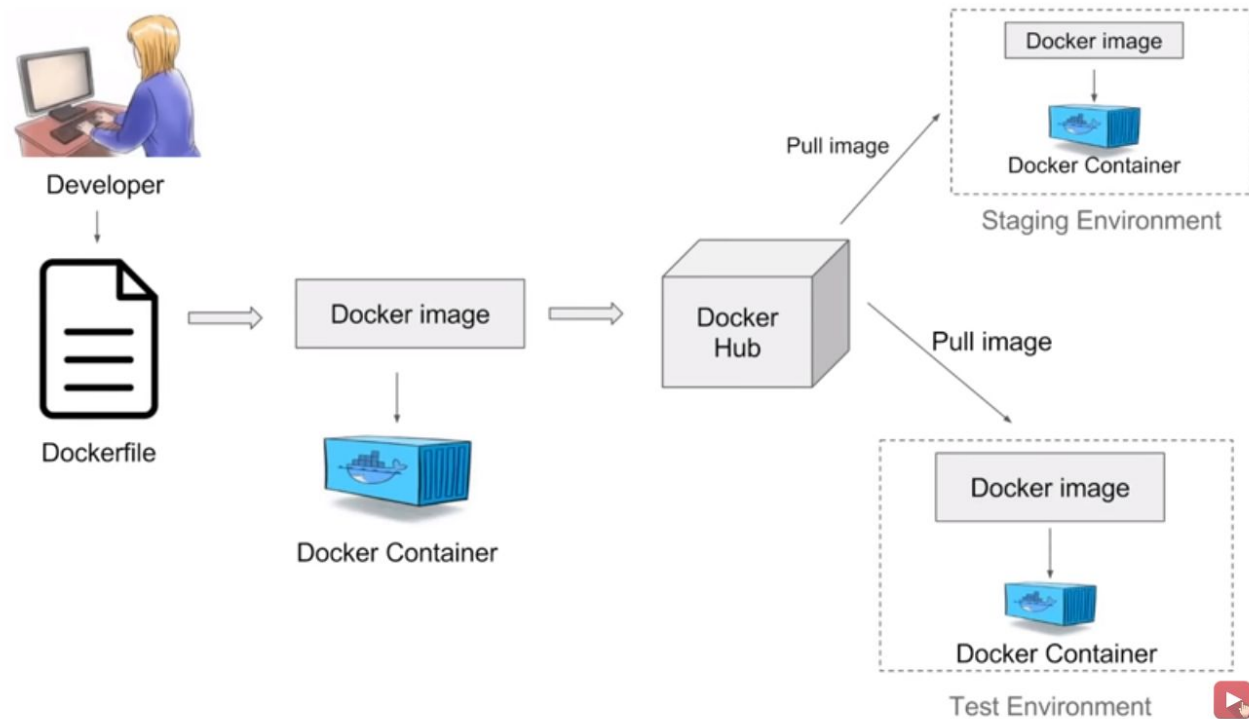
Docker Image

run



Docker Container

Dockerfile



Images Registry

Basic taxonomy in Docker

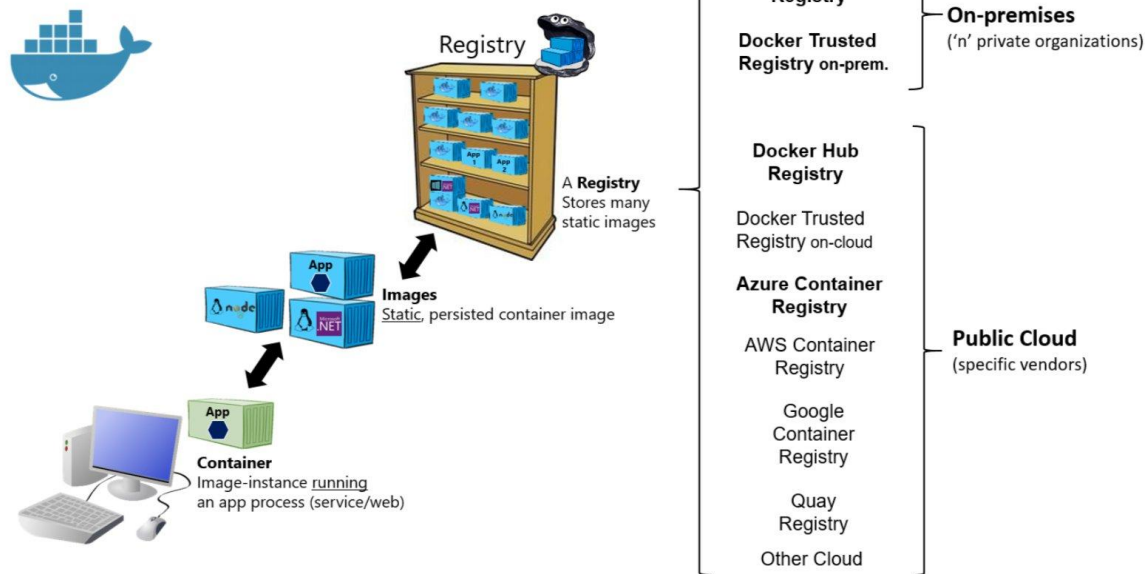
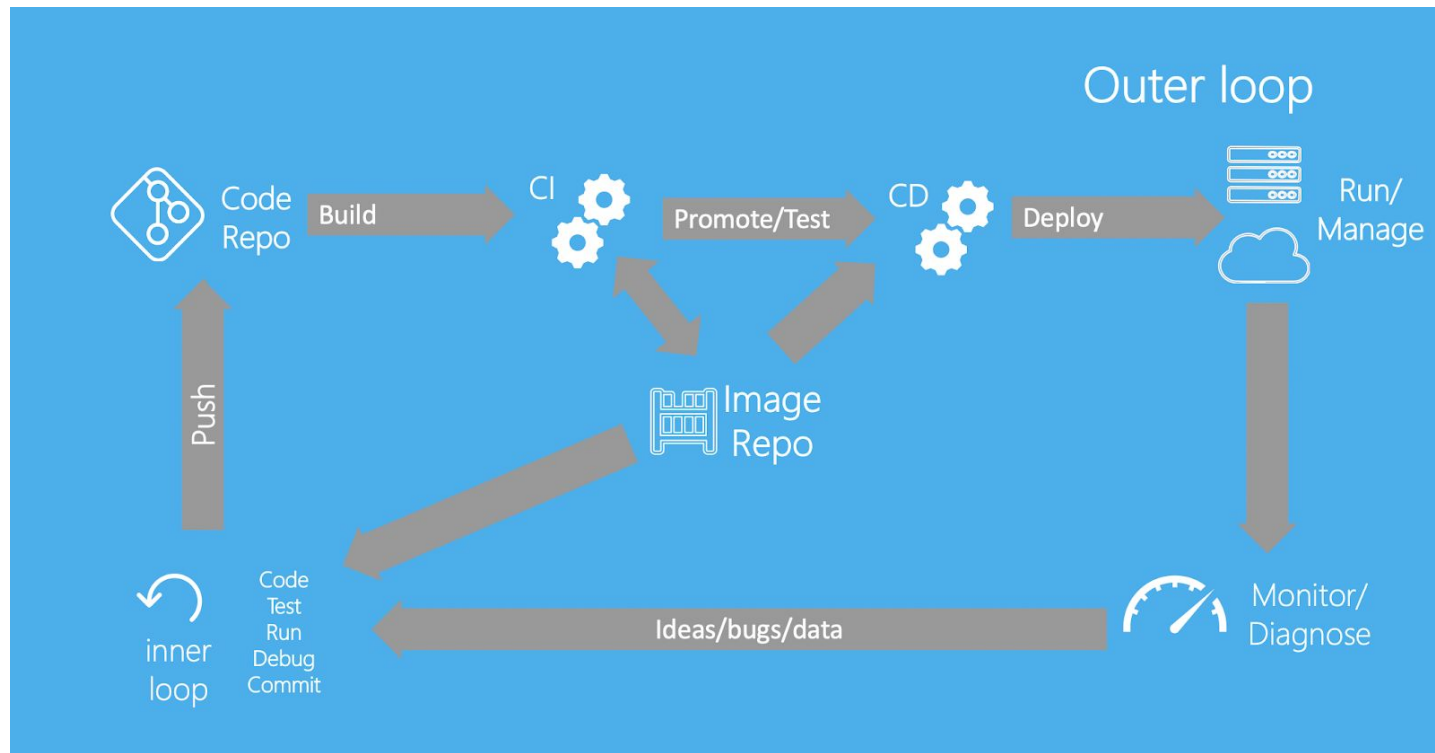


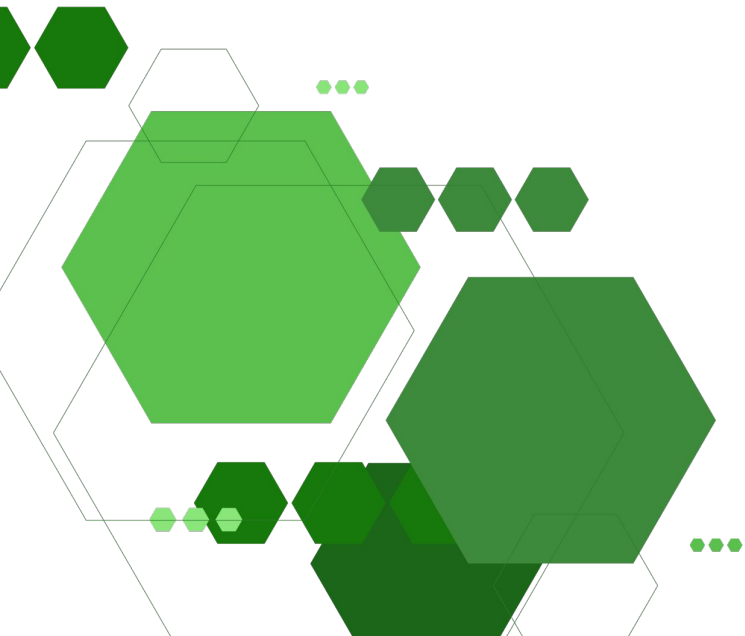
Figure 2-4. Taxonomy of Docker terms and concepts

Docker cycle in CI/CD



Соберем простое приложение

с помощью Docker в GitLab CI



Соберем простое приложение

Соберём локально

- Скачаем Dockerfile

```
git clone https://gitlab.com/nikolai.mishchenkov/docker.git
```

- Соберём образ

```
docker build -t podinfo .
```

- Запустим контейнер

```
docker run --rm -d --name podinfo -p 80:9898 podinfo
```

- Проверим работу контейнера

```
curl localhost
```

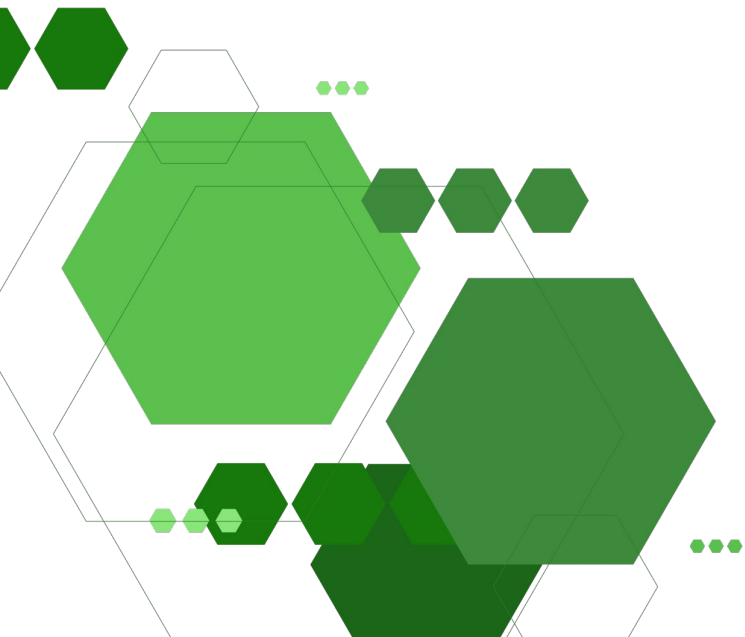
```
http://localhost
```

Соберем простое приложение

Соберём в GitLab CI

- Сделаем себе форк <https://gitlab.com/nikolai.mishchenkov/docker>
- Сделаем Deploy Token
- Добавим Variables GITLAB_CI_PASSWORD, GITLAB_CI_USER

Поговорим о кашировании



Поговорим о кэшировании

Советы по эффективному использованию кэша Docker

- Объединяйте команды apt-get **update** и apt-get **install** в одну цепочку

```
RUN apt-get update && apt-get install -y package-one
```

- Помещайте часто **изменяемый** код как можно ближе к **концу** Dockerfile.
- Разделяйте в разные слои установку и копирование своего кода

```
# Установка зависимостей
```

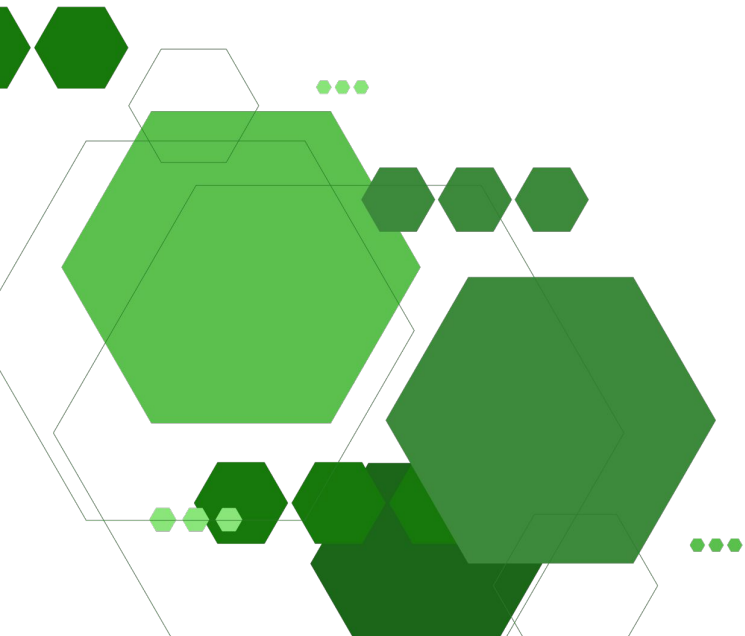
```
COPY package*.json ./
```

```
RUN npm install
```

```
# Копирование файлов проекта и сборка проекта
```

```
COPY ..
```

Поговорим об ОПТИМИЗАЦИИ МЕСТА



Поговорим об оптимизации места

Рекомендации по уменьшению размеров образов

- Компактные образы получаются основанными на **Alpine** Linux

FROM alpine:3.13.1

- Объединяйте в одну инструкцию команды установки пакетов:

```
RUN apt-get update && apt-get install -y \
```

```
package-one \
```

```
package-two \
```

```
package-three \
```

```
&& rm -rf /var/lib/apt/lists/*
```

- Пользуйтесь файлом .dockerignore

Поговорим об оптимизации места

Рекомендации по уменьшению размеров образов

- Используйте многоступенчатую сборку образов

FROM golang:1.7.3 AS **build**

WORKDIR /go/src/github.com/my_name/my_project/

RUN go get -d -v golang.org/x/net/html

COPY app.go .

RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest

RUN apk --no-cache add ca-certificates

WORKDIR /root/

COPY --from=**build** /go/src/github.com/my_name/my_project/app .

CMD ["/app"]

Подведем итоги



Обсудили основные понятия Докера.



Собрали простое приложение с помощью Docker в GitLab CI.



Разобрали вопросы кэширования и варианты оптимизации места

Ваши вопросы

Оставшиеся вопросы можем обсудить в общем канале в Slack

SKILLFACTORY

Успехов в обучении!

Мы всегда рядом

Спасибо за внимание