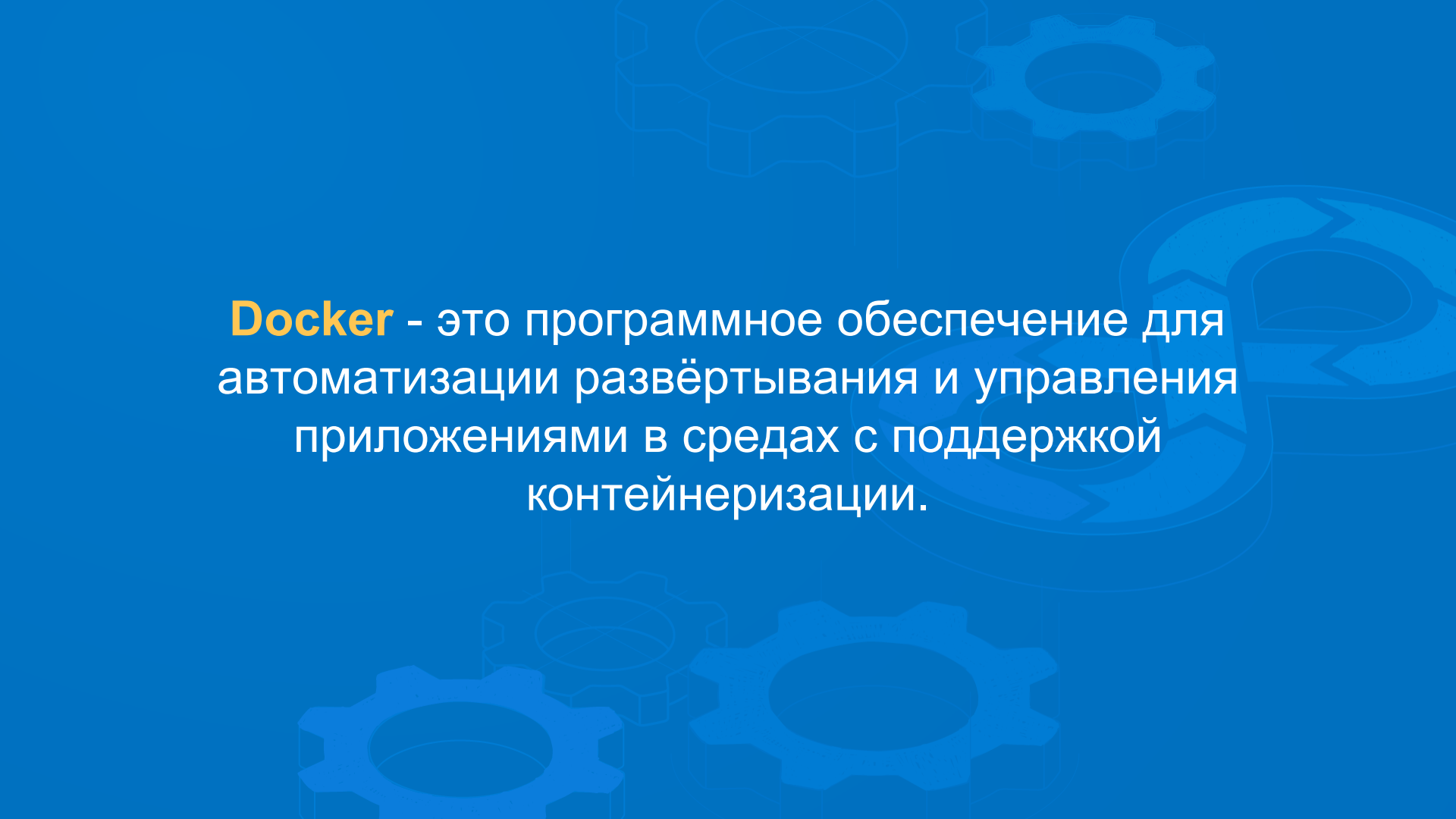


# Docker. Основы. Образ. Cli. Dockerfile.



The background is a solid blue color. It features several faint, stylized gear icons of different sizes and a large circular arrow icon, suggesting a technical or engineering theme.

**Docker** - это программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации.



# Монолитная эра

- Приложения монолитные
- Куча зависимостей
- Долгая разработка до релиза
- Все инстансы знаем по именам
- Используем виртуализацию
  - Один сервер – несколько VM
  - Resource Management
  - Изоляция окружений





# Системы виртуализации

**vmware®**

 Microsoft  
**Hyper-V**



 **QEMU**



# Системы виртуализации

vmware®



Microsoft  
Hyper-V

Оверхед на гипервизор,  
большие образы, медленно



QEMU

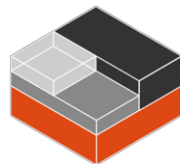


# Системы виртуализации на уровне ядра



OpenVZ

Systemd-nspawn

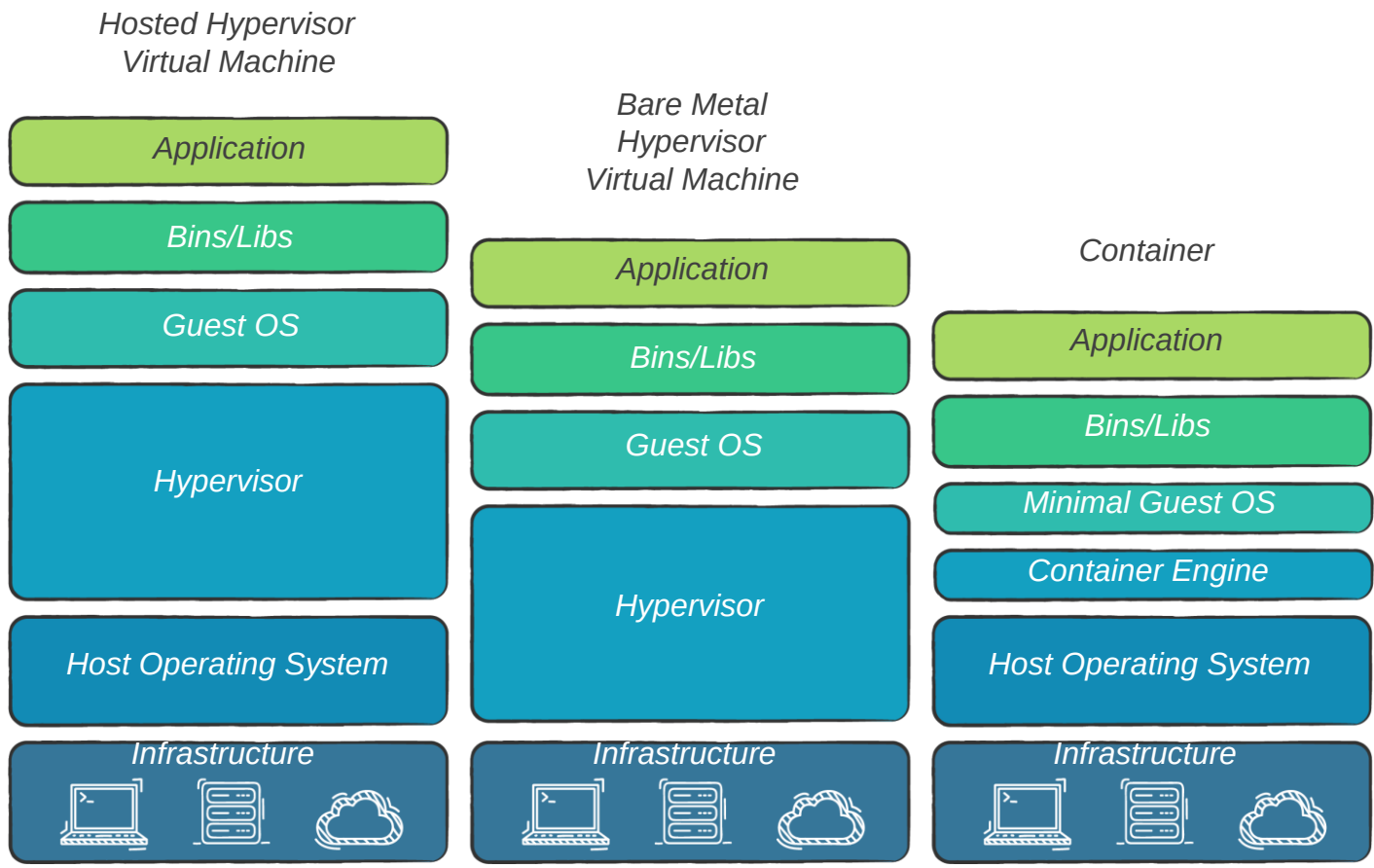


**LXC**



# Разница виртуалки и контейнера

- Виртуальная машина подразумевает виртуализацию железа для запуска гостевой ОС
- Контейнер использует ядро хостовой системы
- В виртуалке может работать любая ОС
- В контейнере только Linux (недавно и Windows)
- Виртуалка хороша для изоляции
- Контейнер для изоляции - плохо







# Что используется для контейнеризации?

## Namespaces

- PID
- Networking
- Mount
- User

## Control Groups

- Memory
- CPU
- Block I/O
- Network



# Эра контейнеров

- Другая философия
- Один процесс – один контейнер
- Все зависимости в контейнере
- Чем меньше образ – тем лучше
- Инстансы становятся эфемерными
- Расцвет Docker





# Docker

- Меняет философию
- Стандартизирует упаковку приложения
- Решает вопрос зависимостей
- Гарантирует воспроизводимость
- Минимум (или совсем нет?) оверхеда



# Docker

*Docker  
Daemon*

*Docker CLI*

*Dockerfile*

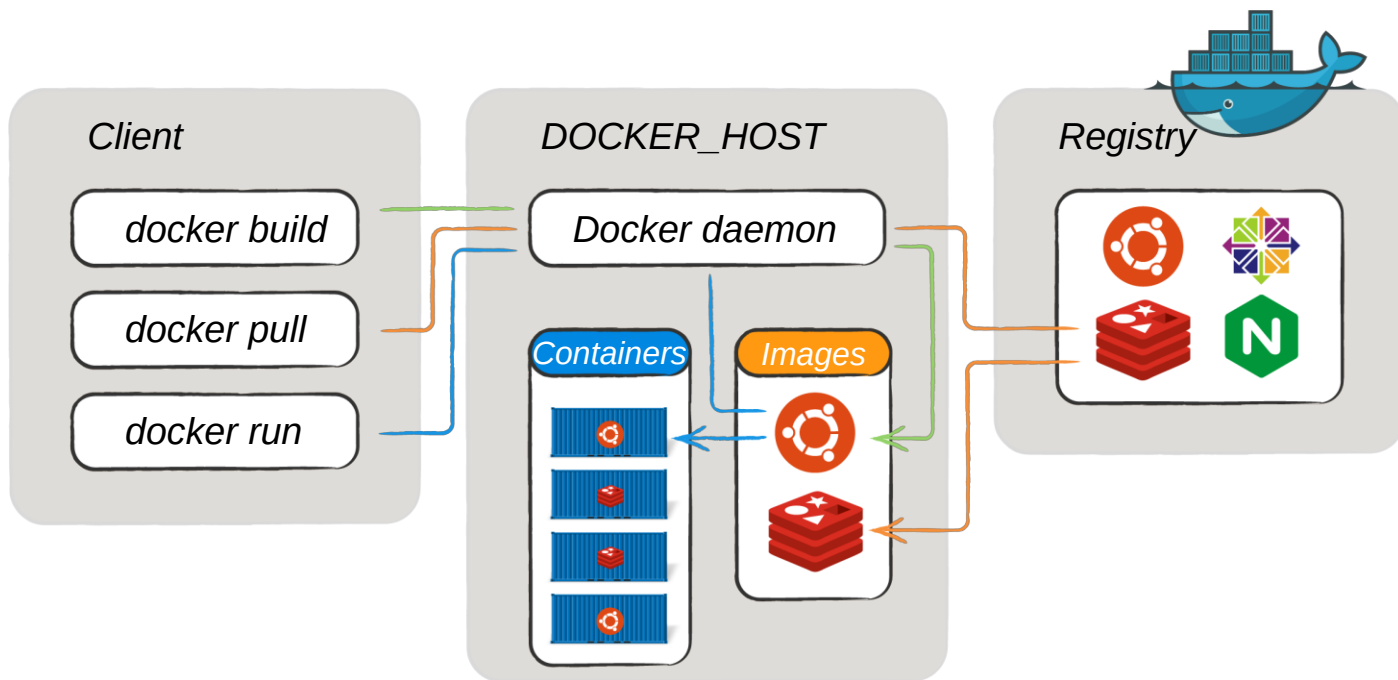
*Image*

*Container*

*Docker  
registry*



# Docker components





# Docker Daemon

- Серверная часть
- Работает на хост-машине
- Скачивает образы и запускает из них контейнеры
- Создает сеть между контейнерами
- Собирает логи контейнеров
- Создает новый образ



# Docker CLI

Консольная утилита  
для работы с докер-демоном

Может работать не только локально,  
но и по сети



# Docker – основные команды:

- **docker search <<name>>** - поиск образа в регистри
- **docker pull <<name>>** - скачать образ из регистри на машину
- **docker build <</path/to/dir>>** - собрать образ
- **docker run <<name>>** - запустить контейнер
- **docker rm <<name>>** - удалить контейнер
- **docker ps** - список работающих контейнеров
- **docker logs <<name>>** - логи контейнера
- **docker start/stop/restart <<name>>** - работа с контейнером



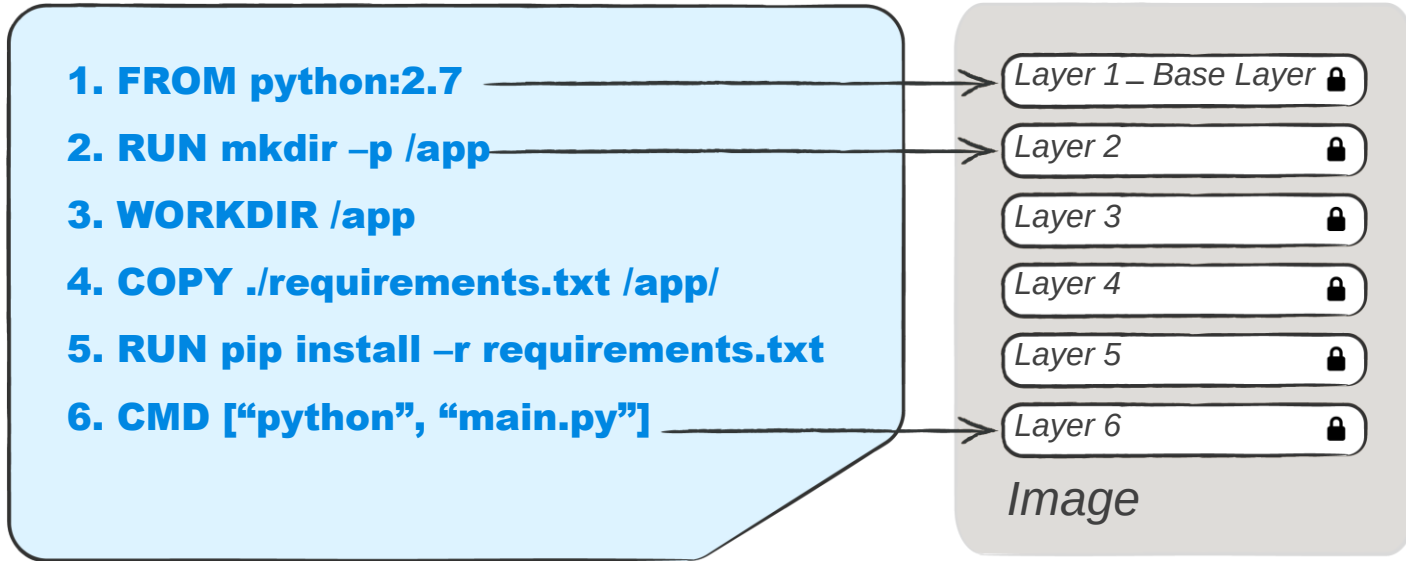


# Dockerfile

Инструкция для создания образа

Почти каждая команда инструкции – новый слой

## Dockerfile



<https://docs.docker.com/engine/reference/builder/>



# Image

- Упаковка нашего контейнера
- Из них запускаются контейнеры
- Хранятся в докер-реестрах (registry)
- Имеют hash, имя и tag
- Имеют «слоёную» структуру
- Создаются (build'ются) по инструкции (Dockerfile)



## Docker registry

- Хранит образы докера
- Общедоступный стандартный реестр – dockerhub
- Но можно сделать свой



# Container

- Запускается из образа
- Изолирован
- Должен содержать в себе всё для работы приложения
- 1 процесс – 1 контейнер



## Полезные ссылки:

- <https://docs.docker.com>

- <https://labs.play-with-docker.com>

- [https://habr.com/ru/company/swordfish\\_security/blog/537280/](https://habr.com/ru/company/swordfish_security/blog/537280/)

- <https://habr.com/ru/company/otus/blog/585636/>

- <https://habr.com/ru/company/selectel/blog/279281/>

- <https://fabiokung.com/2014/03/13/memory-inside-linux-containers/>

- [https://docs.docker.com/config/containers/resource\\_constraints/](https://docs.docker.com/config/containers/resource_constraints/)