# docker

Andreas Tiefenthaler @pxlpnk

# What is this about?

- Docker in Theory
- Technology
- Tiny live demo
- potential/common use cases
- Some code
- Bigger live demo
- Cons

**Docker** is an **open-source** engine that **automates** the deployment of any application as a **lightweight**, **portable**, self-sufficient container that will **run** virtually **anywhere**.

# Main Features

Can **encapsulate** any **payload** and its **dependencies**

**Lightweight**, virtually no performance or start-up penalty, **quick** to **move** and **manipulate**

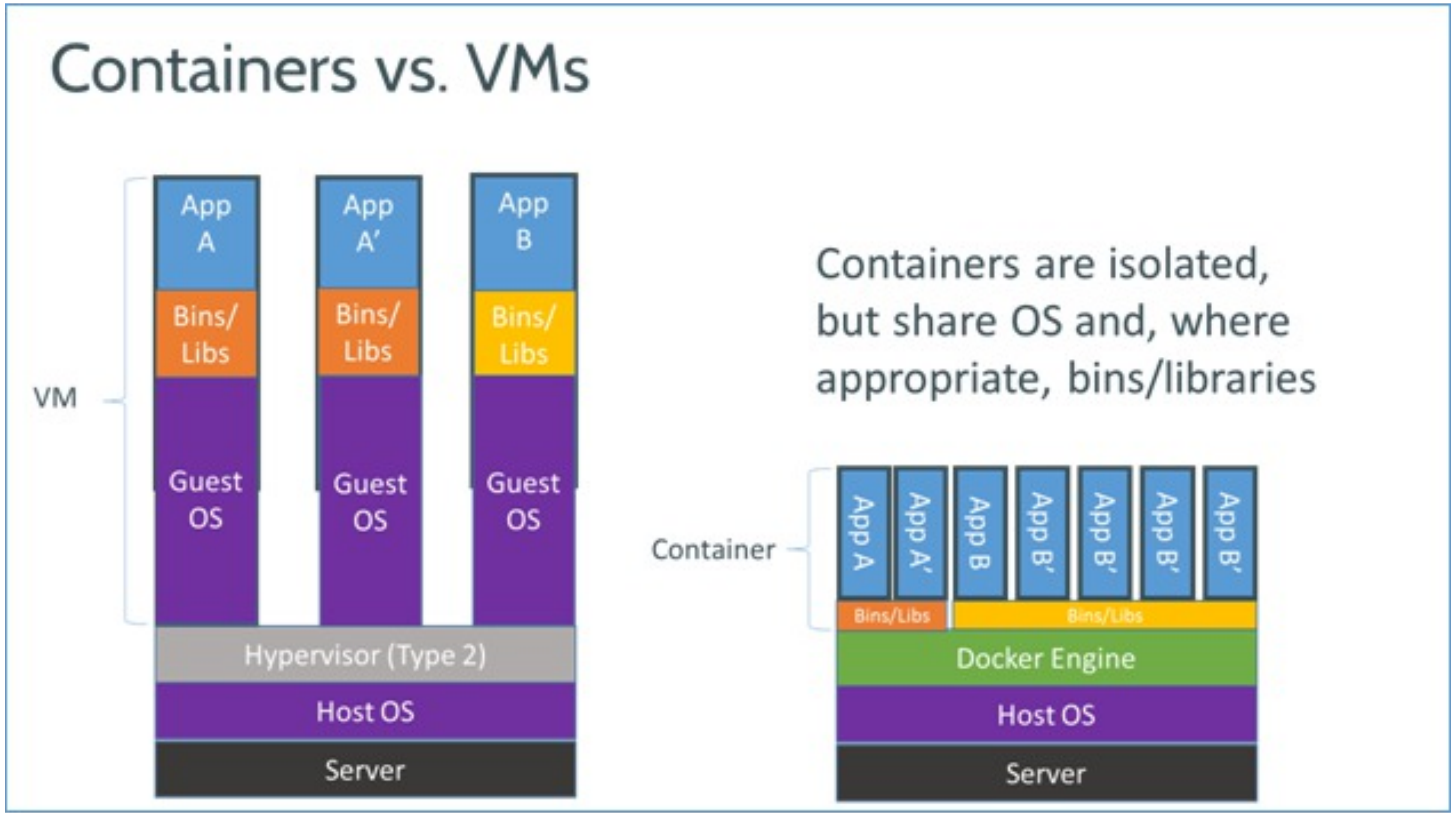
**Separation** of **duties**

code vs infrastructure

Resource, network, and content **isolation**.

Avoids dependency hell

# Containers vs VMs



Containers vs. VMs

Containers are isolated, but share OS and, where appropriate, bins/libraries

# Container?

- Uses Linux Containers - LXC
- cgroups
- Linux Kernel itself

# LXC

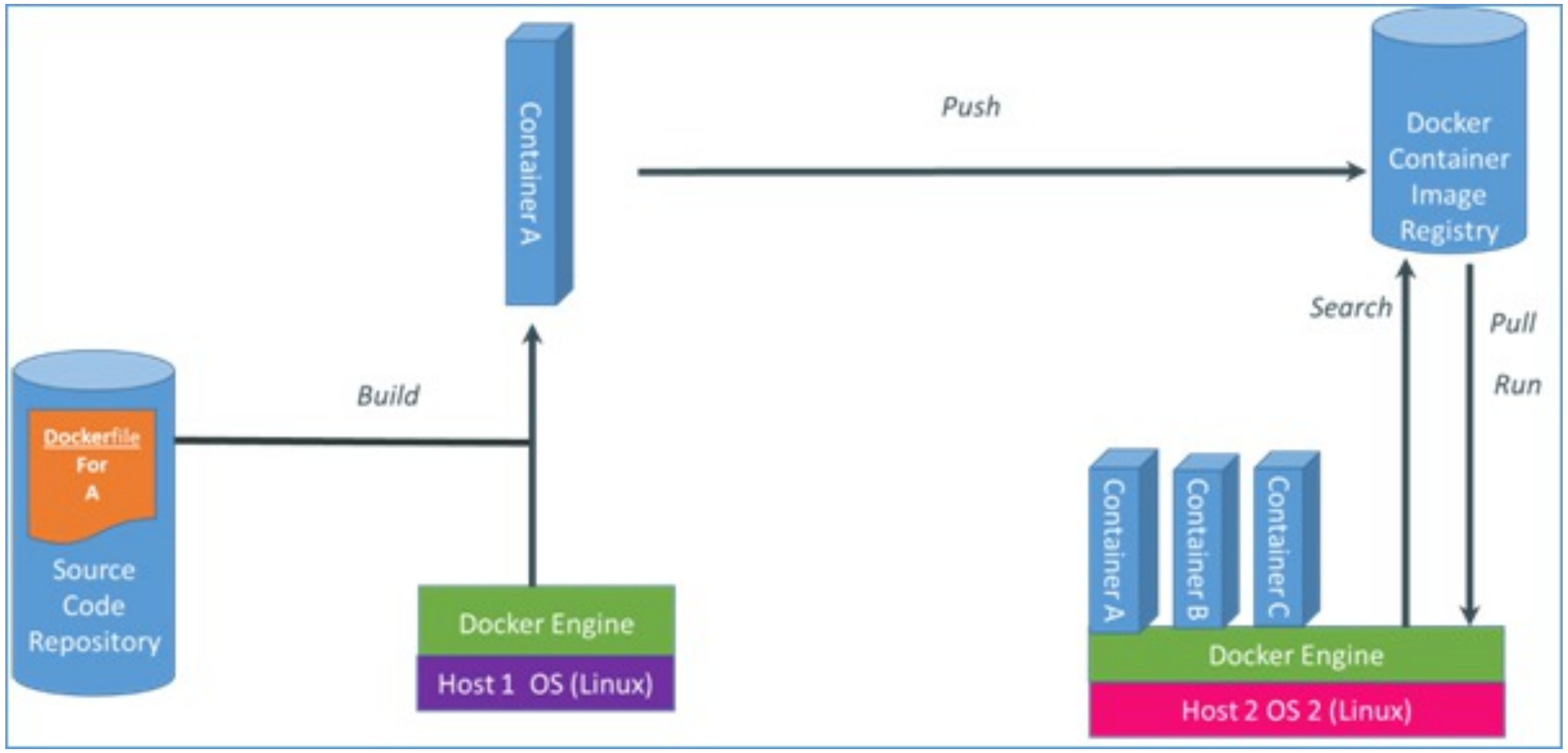
provides **operating system-level virtualization** via a virtual environment that has its **own process** and **network space**.

# c(ontrol) groups

is a Linux **kernel** feature to **limit, account and isolate resource usage** (CPU, memory, disk I/O, etc.) of process groups.

# Basic Functions

- build
- modify
- publish
- search
- run containers



# First demo:

```
docker run ubuntu /bin/echo hello world
```

# Use cases

## Automating the packaging and deployment of applications

Creation of lightweight, private PAAS environments

# Automated testing and continuous integration/ deployment

Deploying and scaling web apps, databases and backend services

Other ideas?

# Who uses docker?

BLITLINE

IMAGE PROCESSING IN THE CLOUD

ebay

https://flynn.io/

Yandex

wercker

CLOUDFLARE

Travis

rackspace

# Some Code

# Dockerfiles

```
FROM        ubuntu:12.10
RUN         apt-get update
RUN         apt-get -y install htop
ENTRYPOINT  ["/bin/bash"]
```

```
$docker build -t container .
```

# Demo

The APP

# The APP

- Sinatra Webservice Container
- Sidekiq Worker Container
- Redis Container

# Sinatra

```ruby
require_relative 'config.rb'
require_relative 'worker'

require 'sinatra'

set :environment, :production unless
ENV['development']

get '/:task/:cost' do |task, cost|
  Worker.perform_async(task, cost)
end
```

# Workers

```
require_relative 'config.rb'
require 'sidekiq'

class Worker
  include Sidekiq::Worker

  def perform(task = 'HardTask', cost = 2)
    puts "Working on #{task} for: #{cost}"
    sleep cost.to_i
  end
end
```

# Config

```
DB_URL = ENV['DB_PORT_6379_TCP_ADDR']
DB_PORT = 6379
DB_NO = 0
DB_NAMESPACE='catcher'

Sidekiq.configure_client do |config|
  config.redis = { namespace:  DB_NAMESPACE, size: 1,
        url: "redis://#{DB_URL}:#{DB_PORT}/#{DB_NO}" }
end

Sidekiq.configure_server do |config|
  config.redis = { namespace: DB_NAMESPACE,
        url: "redis://#{DB_URL}:#{DB_PORT}/#{DB_NO}" }
end
```

# DEMO

# Cons

- **Not** production ready
- **Rapidly** changing
- Should not run Database within it
- Does not give you Security Superpowers (http://blog.docker.io/2013/08/containers-docker-how-secure-are-they/)

**Danke, thank you, dankjewel**

Questions?

Docker: https://www.docker.io/

Source: https://github.com/pxlpnk/docker

Diagrams taken from docker.io