Achieving Continuous Delivery with *Docker*

for Java Enterprise applications

Petyo Dimitrov | Musala Soft JSC

18-19 November, Sofia

ista 2015 conference

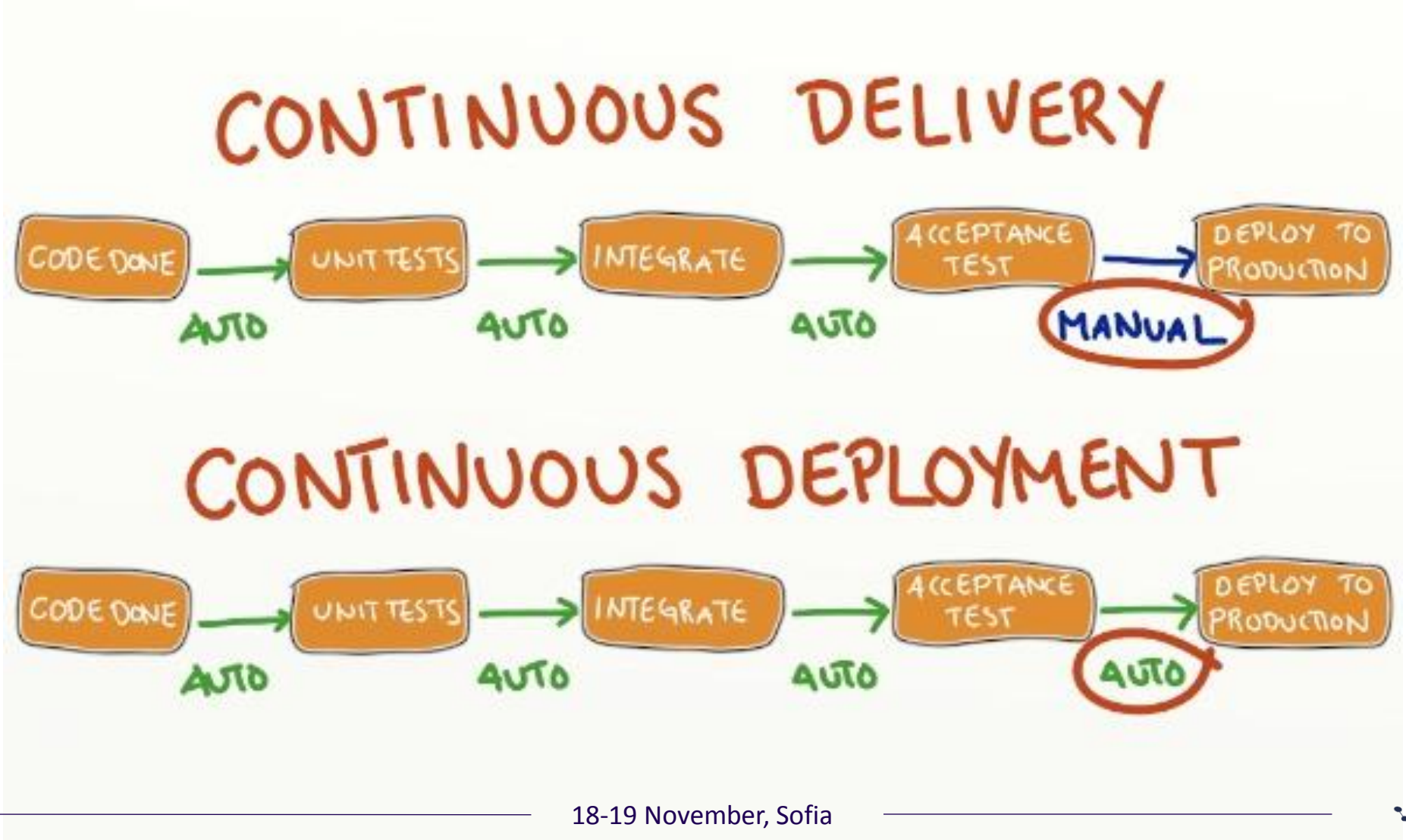
Agenda

- A typical Enterprise Java landscape
- Continuous Delivery pipeline
- Identified problems
- What are Docker containers?
- Docker "ecosystem"
- Demo
- Tips & Tricks

Dev landscape for Java EE applications

maven

Continuous Delivery pipeline



potential issues

inadequate packaging

of application artifacts

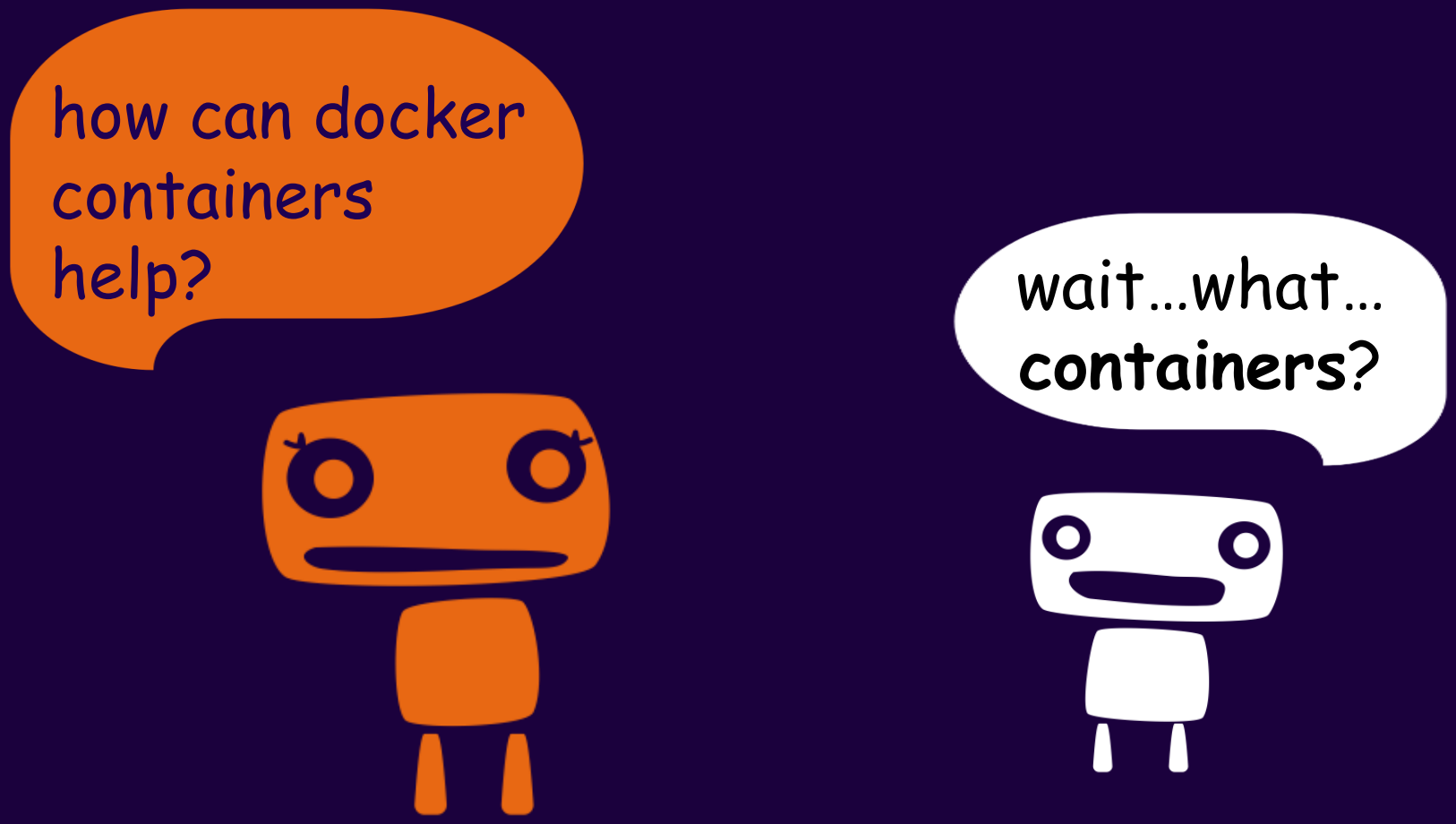
inconsistencies across *environments*

high cost

supporting multiple static environments

lack of freedom

experimenting with new languages, technologies and frameworks


how can docker containers help?
wait…what… **containers**?


getting sea sick here!!!

sea-free analogy

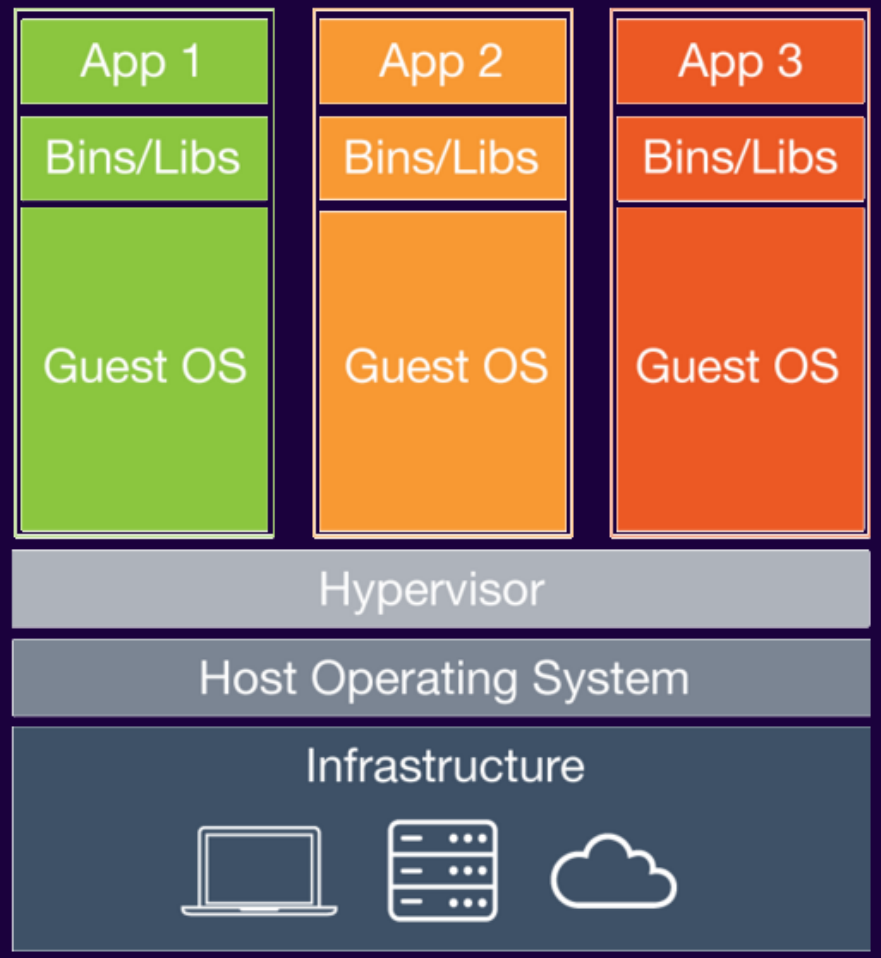
High level view of a container

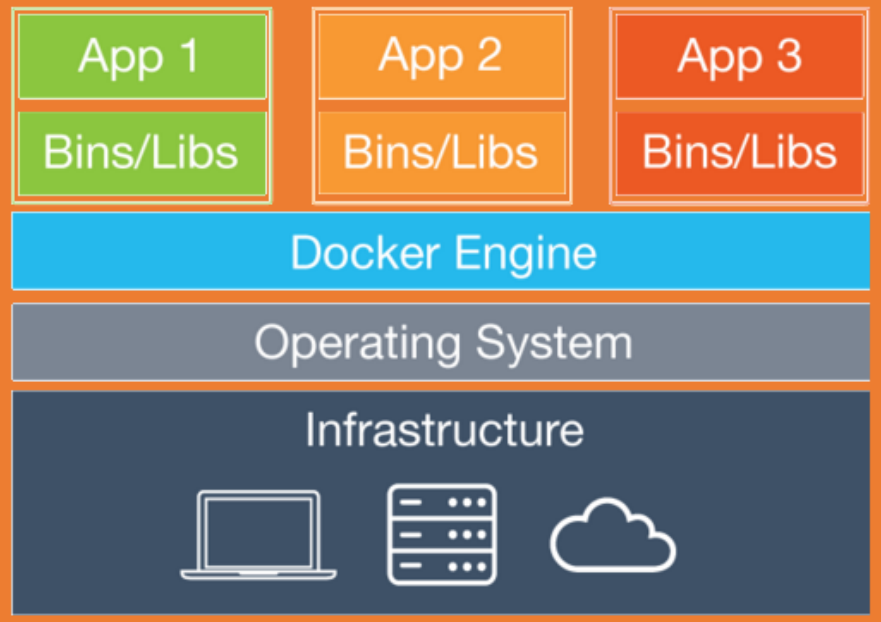
- **it is like a lightweight Virtual Machine**
- it provides:
  - own process space
  - own network interface
  - running stuff as root

Low level view of a container

- **container = "process in a box"**
- shares kernel with host → boots faster
- processes run directly on the host
- there is no device emulation
- none or little CPU, memory, network and I/O overhead

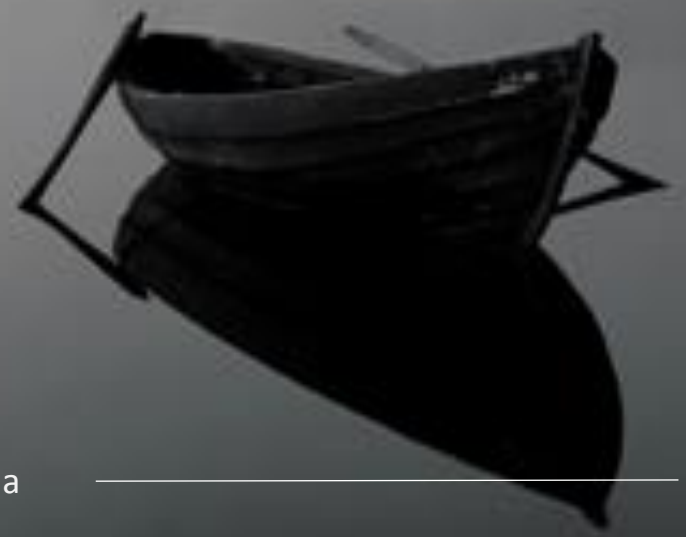
Virtual machines vs Containers





How does it work? – Isolation

- **isolation via namespaces**
  - pid, mnt, net, uts, user, ipc
- **isolation via control groups**
  - memory, cpu, blkio, devices



How does it work? – Storage

- union file system (aufs, overlayfs)
- allows reusing common layers
- reduces traffic and storage
- allows tracking changes
- copy-on-write pattern used for speed

Developers

- care about apps
- put stuff in containers
  - code & data
  - libraries
  - applications

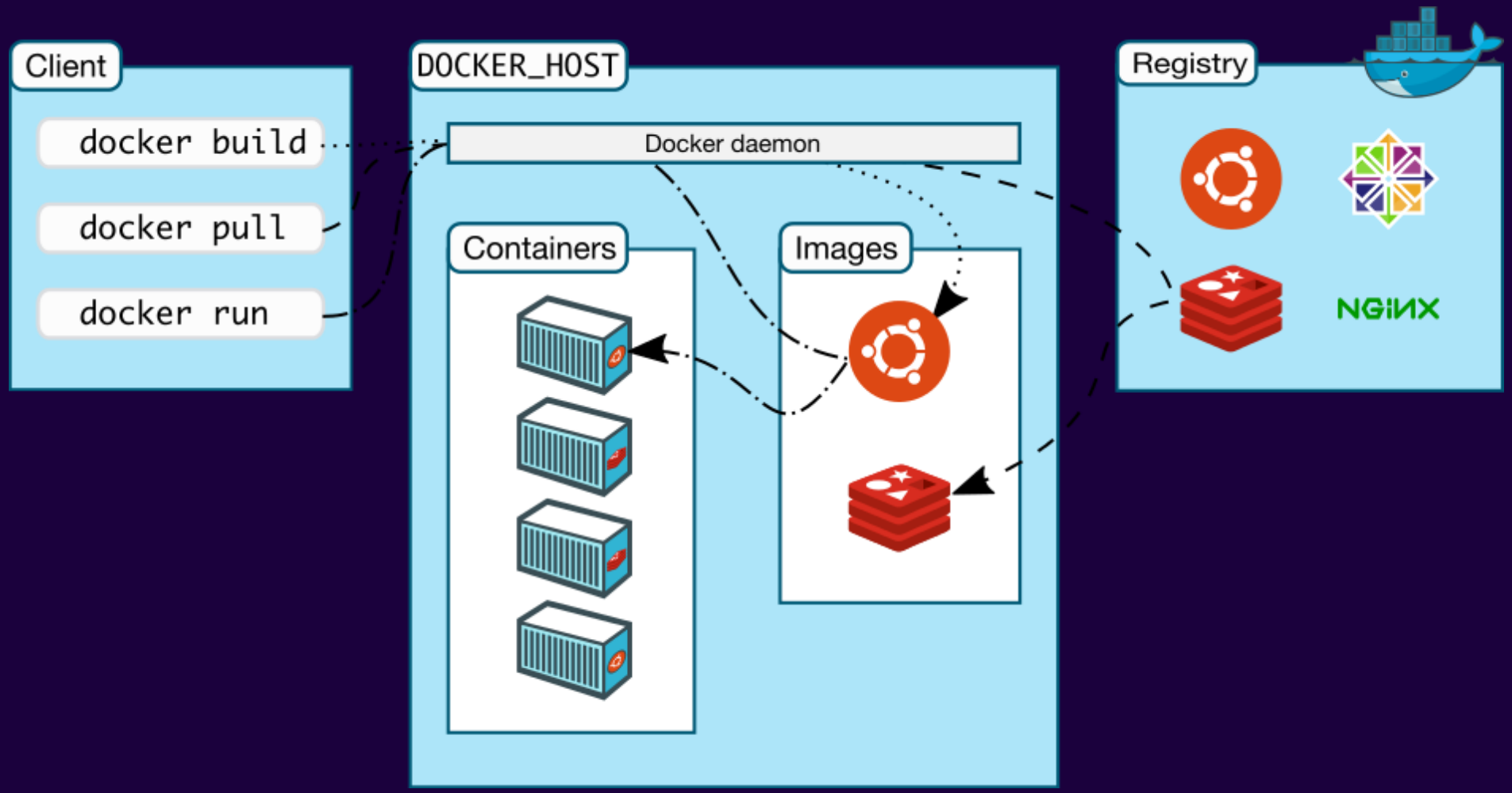
Operations

- care about containers
- work with containers
  - logging & monitoring
  - networking
  - scaling

docker

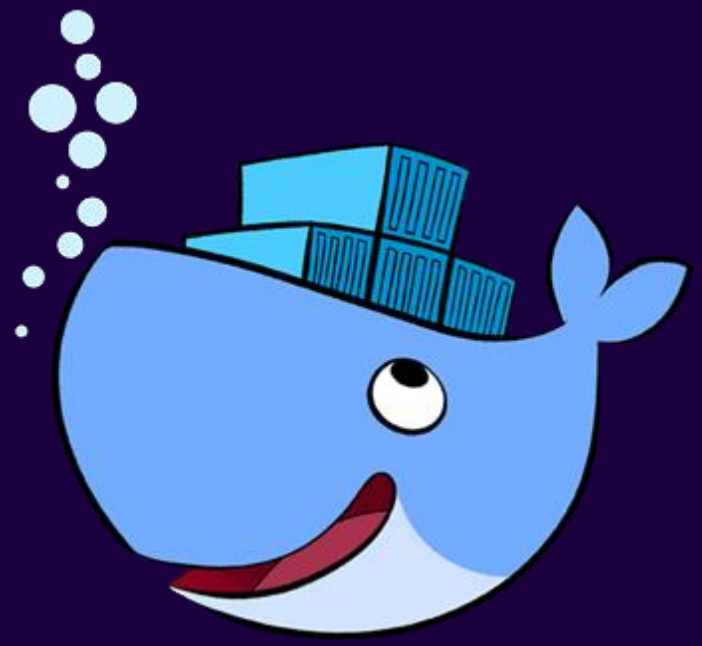
"ecosystem"

Client-Server model



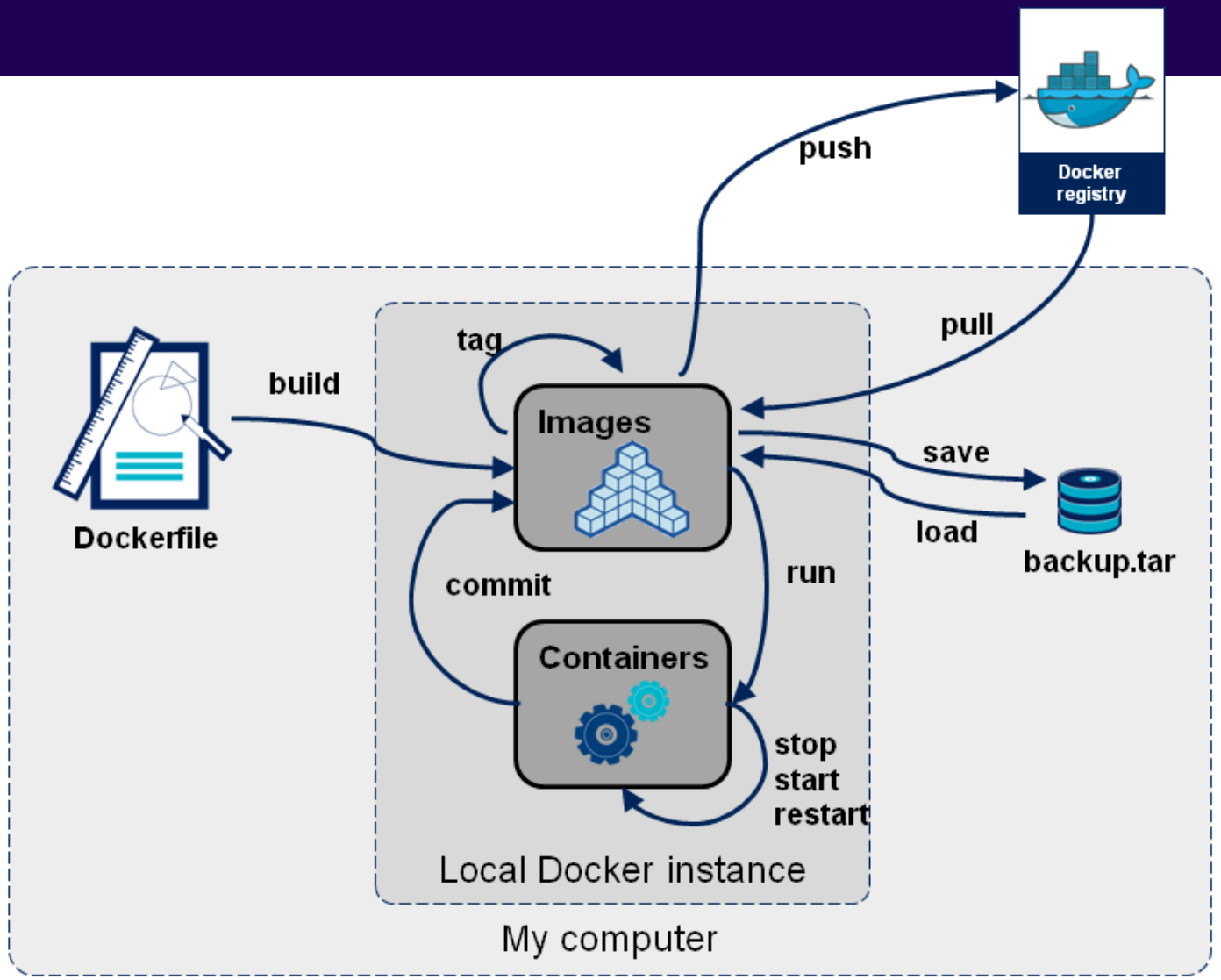
Docker engine

- runs and **commoditizes** Linux containers
- uses copy-on-write for quick provisioning
- runs as daemon and has CLI
- allows building and sharing images
- functionality is exposed via REST API
- defines standard format for containers



Docker commands

- ~40 commands
- for working with:
  - images
  - containers
  - registry

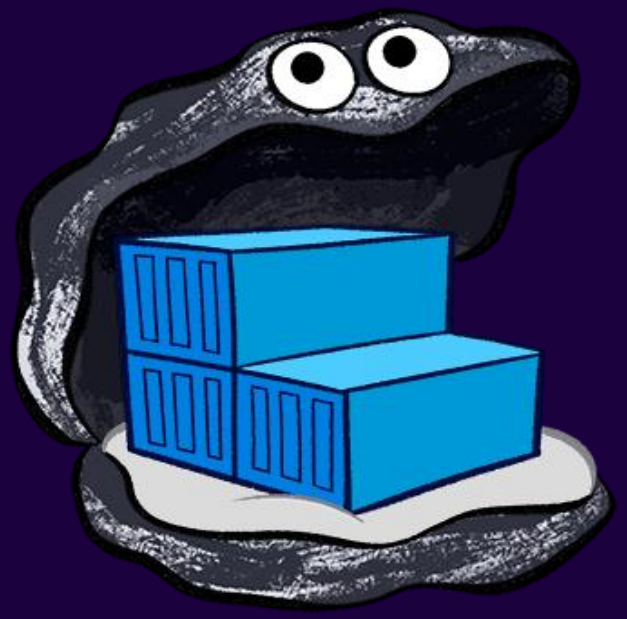


Dockerfile-s

```
FROM jenkins:1.625.1
MAINTAINER Petyo Dimitrov
ENV REFRESHED_AT 2015-10-24

RUN curl -L https://github.com/... > /usr/local/bin/docker-compose
    && chmod +x /usr/local/bin/docker-compose
RUN mkdir /var/cache/jenkins
RUN chown -R jenkins:jenkins /var/cache/jenkins

# Set Defaults
ENV JAVA_OPTS="-Xmx6144m"
ENV JENKINS_OPTS="--handlerCountMax=300 --webroot=/var/cache/jenkins/war"

COPY plugins.txt /plugins.txt
RUN /usr/local/bin/plugins.sh /plugins.txt
```

```
docker build -t myjenkins .
```

Docker Hub/Registry



- collection of ready-made images
- options:
  - public/private registry
  - local registry
- REST API for access

”

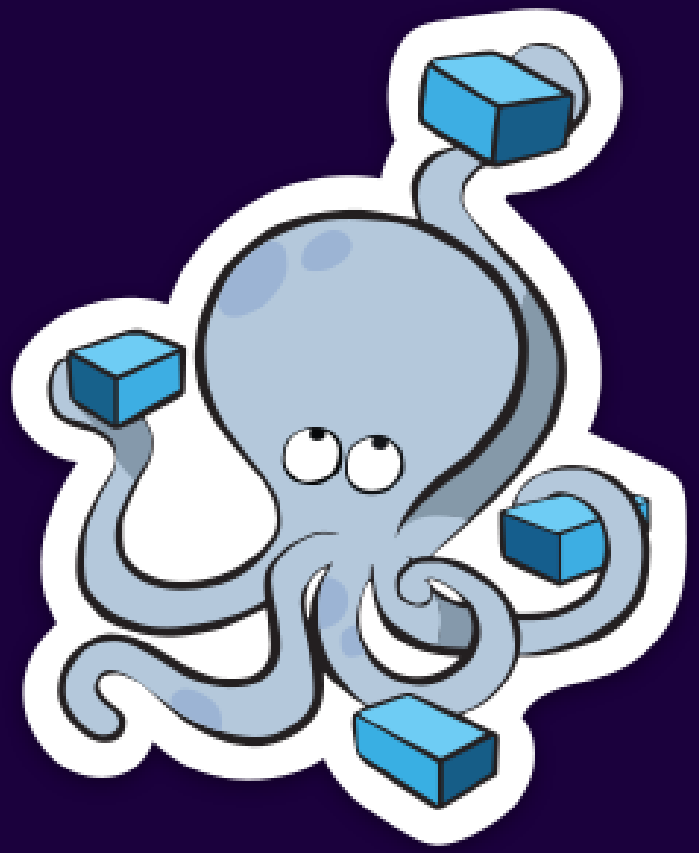
But how to build a stack of containers?

“

Docker Compose

- manages a collection of containers
- fast, isolated development environments using Docker
- define environment via YAML file
- quick and easy to start

```
docker-compose up -d
```



YAML example

```
proxy:
  build: nginx/
  ports:
   - "80:80"
  links:
   - appinstance
  hostname: "proxy"

nosqldb:
  build: mongo/
  hostname: "nosqldb"
  volumes:
   - "/opt/mongodb:/data/db"
```

```
appinstance:
  build: tomcat/
  expose:
   - "8080"
  ports:
   - "8180:8080"
  links:
   - nosqldb
  hostname: "appinstance"
```

”

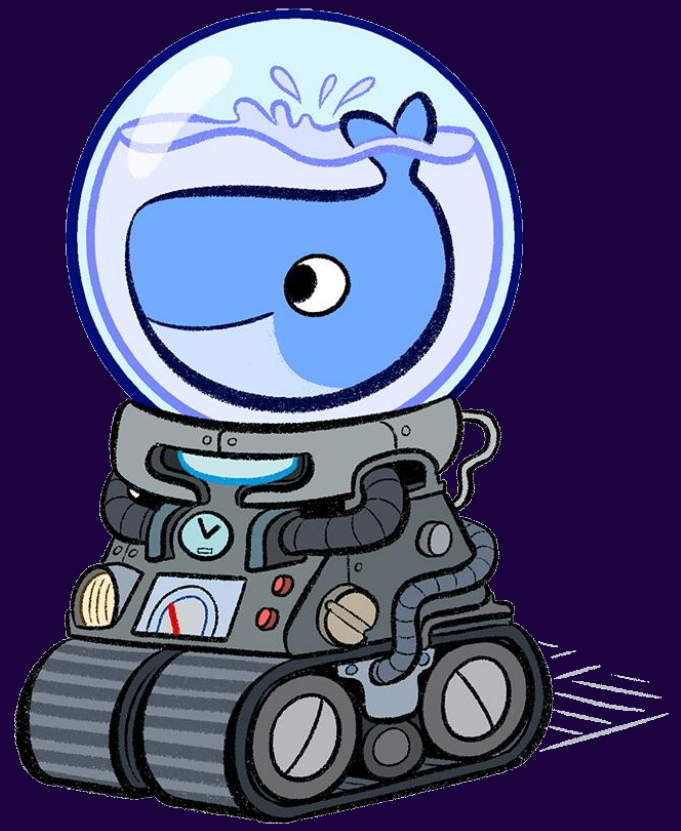
How can I quickly provision a Docker host?

“

Docker Machine

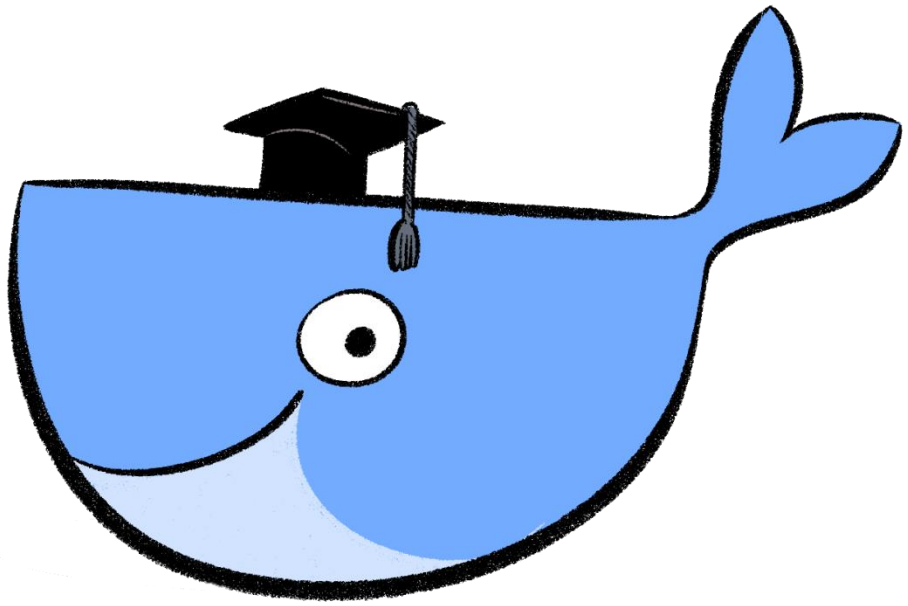
- allows creating Docker hosts on local computer or in cloud providers
- automatically creates *host*, installs Docker and configures the *client*
- offers commands to start, stop, restart and inspect a host

```
docker-machine create
 --driver virtualbox default
```



Docker Summary

- put my software in containers
- run those containers anywhere
- write recipes to automatically build containers



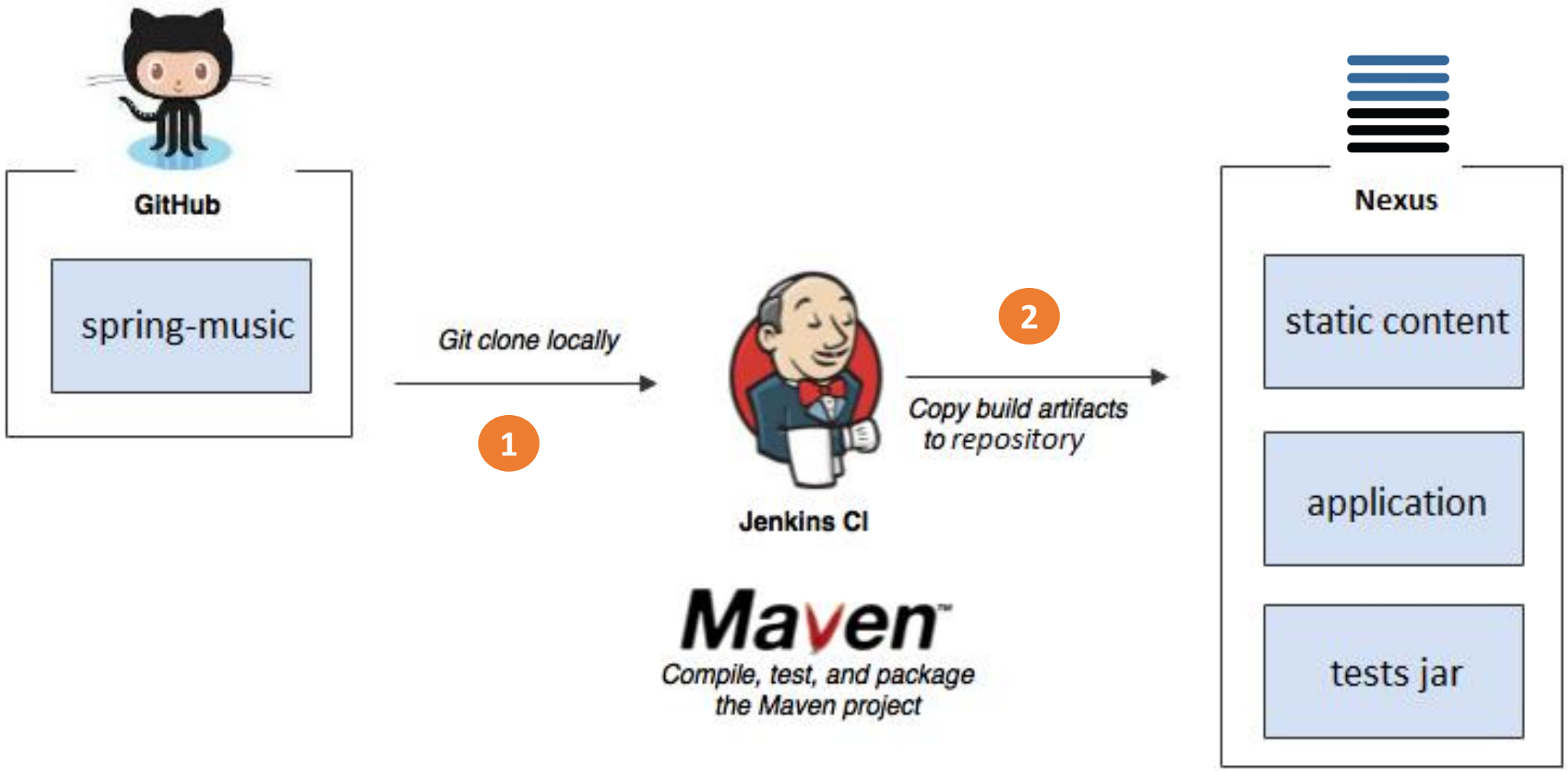
Demonstration

https://github.com/petyodimitrov/spring-music.git

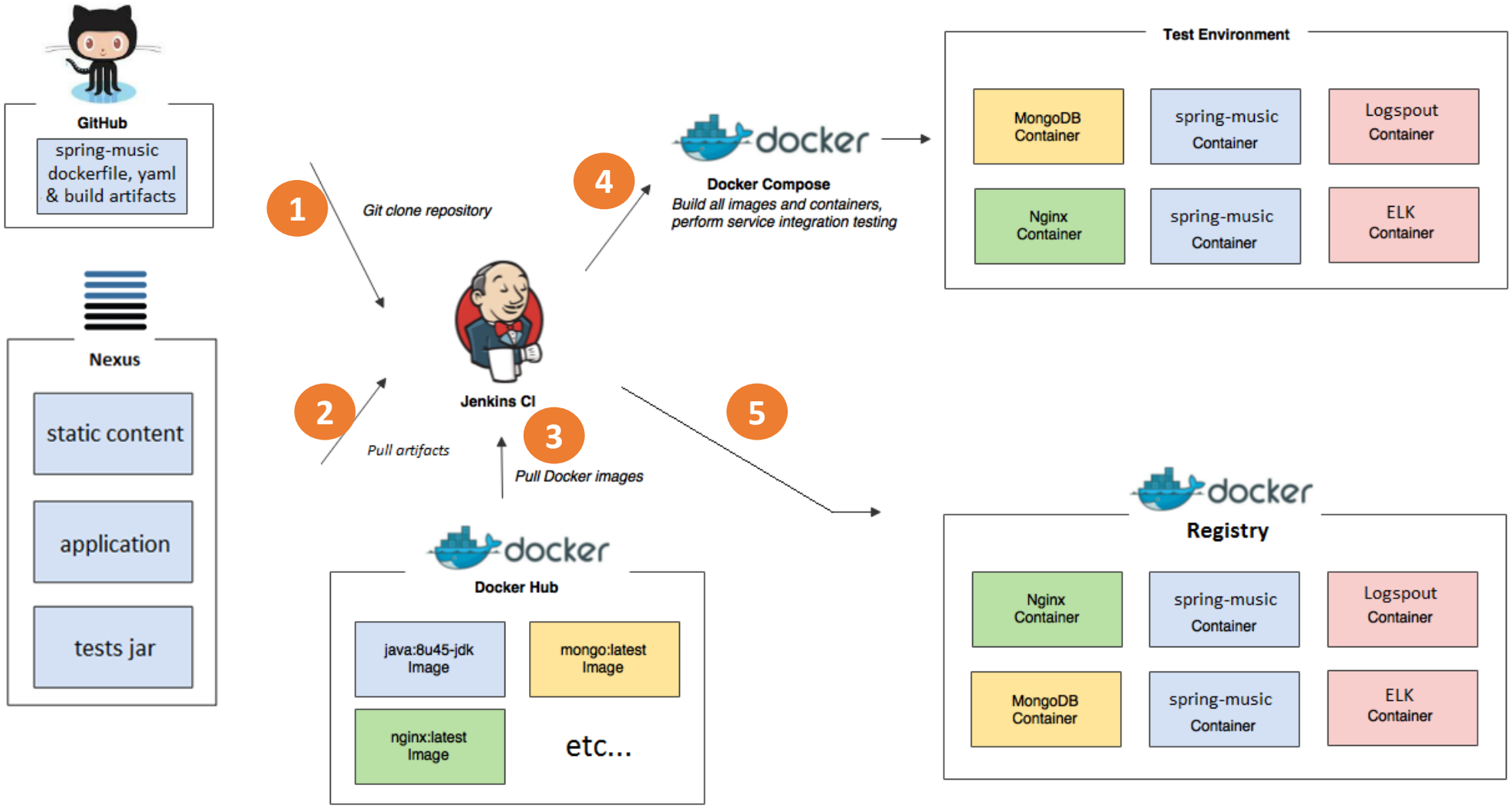
https://github.com/petyodimitrov/app-setup.git

https://github.com/petyodimitrov/ci-setup.git

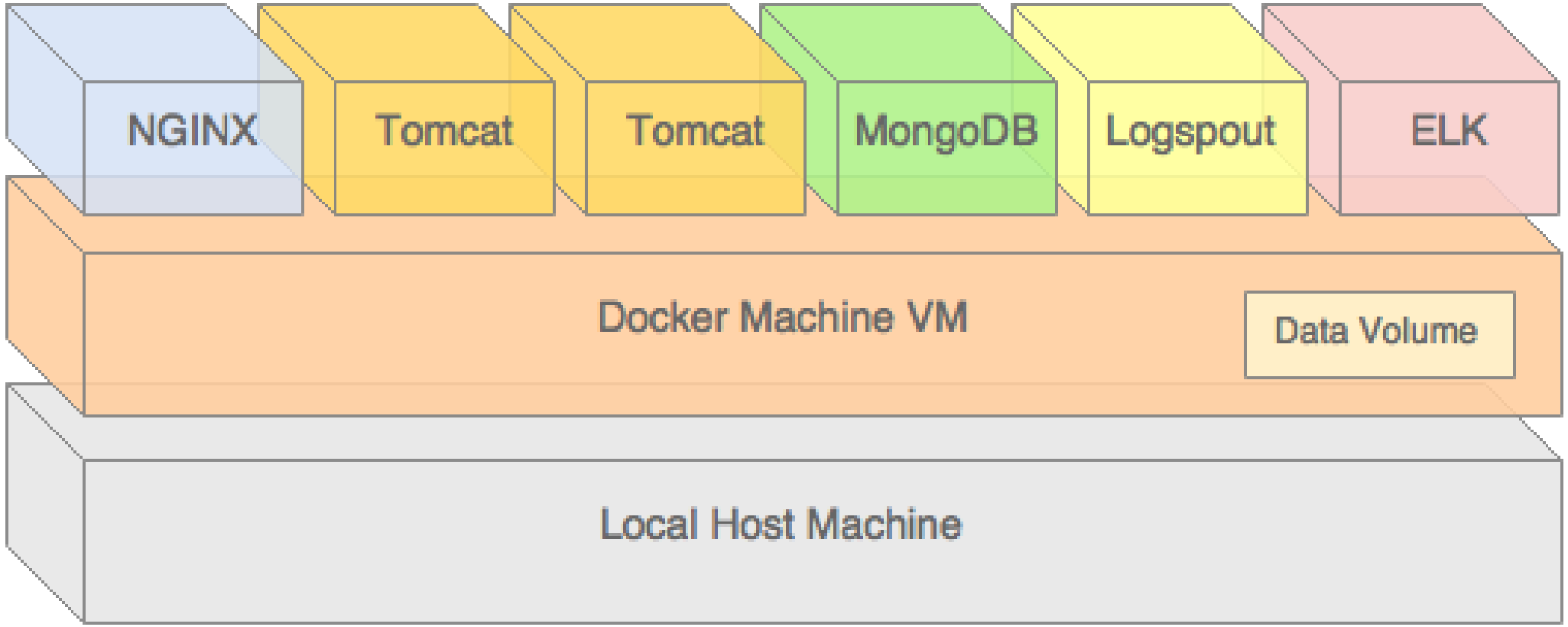
Standard Java application build

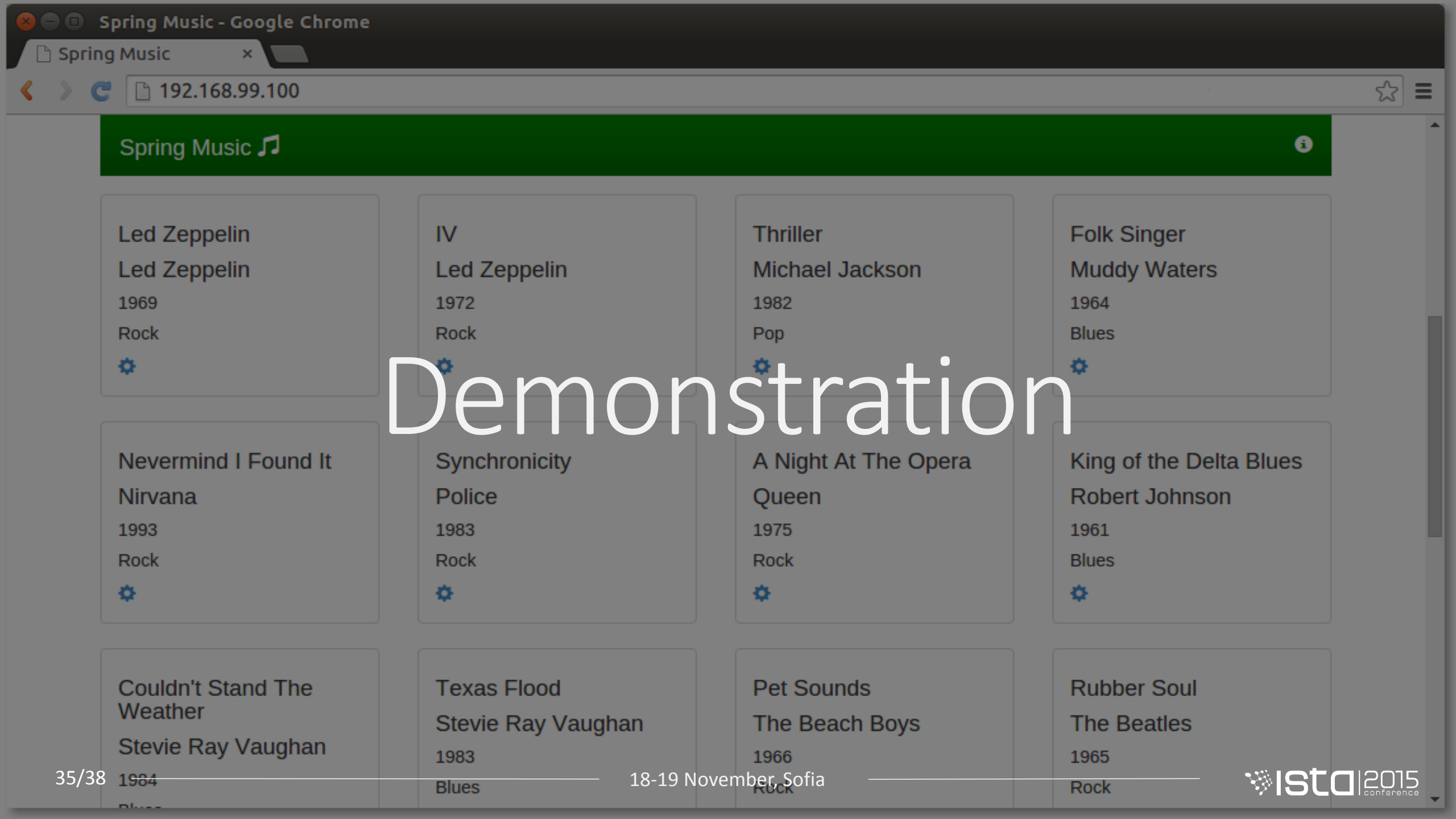


Adding docker images build



Runtime view of containers





Demonstration

Tips & Tricks

- single process per container
- security considerations
- use data containers for portability
- consider reducing image sizes
- etc…

Thank you!

petyo.dimitrov@gmail.com