Vagrant, Ansible, Docker for developers and architects

Richard Attermeyer

- father, software architect and developer

Focus Areas

- Software Architecture
- Java Technologies
- Continuous Delivery und DevOps

Richard.Attermeyer@opitz-consulting.com

@rattermeyer

github.com/rattermeyer

xing.to/rat

http://de.slideshare.net/opitzconsulting

Agenda

High-Level Overview

VMs, Config Mgmt, Container Development + Continuous Delivery

Continuous Delivery

Collection of techniques, processes and **tools**, which are used to improve the process of software delivery

Where to start?

Challenges

Setting up the work environment

Distributing work environment changes

Versioning of work environments

„Works on my machine“

Virtual machines

Solution Approach

Golden Image

But:

Golden Image: Problems

- Big
- Distribution takes long
- Simple customizing is difficult
- Every small change leads to a big amount of data and to a complete reinstallation
- No collaboration
- Versioning is difficult

And now?

VAGRANT

Development environments made easy

Developer-Workflow

```
> git clone https://gh.com/rattermeyer/jenkins-in-a-box.git*
> cd jenkins-in-a-box
> vagrant up
```

* git clone https://github.com/rattermeyer/jenkins-in-a-box.git

Vagrant: Vagrantfile

```
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "phusion/ubuntu-14.04-amd64"
  config.vm.provider "virtualbox" do |vb|
    vb.customize ["modifyvm", :id, "--memory", "1024"]
       vb.customize ["modifyvm", :id, "--cpus", "1"]
  end
  config.vm.provision "puppet" do |puppet|
    puppet.manifests_path = "puppet/manifests"
    puppet.manifest_file  = "site.pp"
    puppet.module_path    = "puppet/modules"
    puppet.options        = "--verbose --debug"
  end
 config.vm.network "private_network", ip: "192.168.33.10"
end
```

Vagrant: Vagrantfile

```
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |co
  config.vm.box = "phusion/ubuntu-14.04-amd64"
  config.vm.provider "virtualbox" do |vb|
    vb.customize ["modifyvm", :id, "--memory", "1024"]
       vb.customize ["modifyvm", :id, "--cpus", "1"]
  end
  config.vm.provision "puppet" do |puppet|
    puppet.manifests_path = "puppet/manifests"
    puppet.manifest_file  = "site.pp"
    puppet.module_path    = "puppet/modules"
    puppet.options        = "--verbose --debug"
  end
 config.vm.network "private_network", ip: "192.168.33.10"
end
```

Starting Point:
Base Box

Vagrant: Vagrantfile

```
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "phusion/ubuntu-14.04-amd64"
  config.vm.provider "virtualbox" do |vb|
    vb.customize ["modifyvm", :id, "--memory", "1024"]
      vb.customize ["modifyvm", :id, "--cpus", "1"]
  end
  config.vm.provision "puppet" do |puppet|
    puppet.manifests_path = "puppet/manifests"
    puppet.manifest_file  = "site.pp"
    puppet.module_path    = "puppet/modules"
    puppet.options        = "--verbose --debug"
  end
 config.vm.network "private_network", ip: "192.168.33.10"
end
```

VM Customization

Vagrant: Vagrantfile

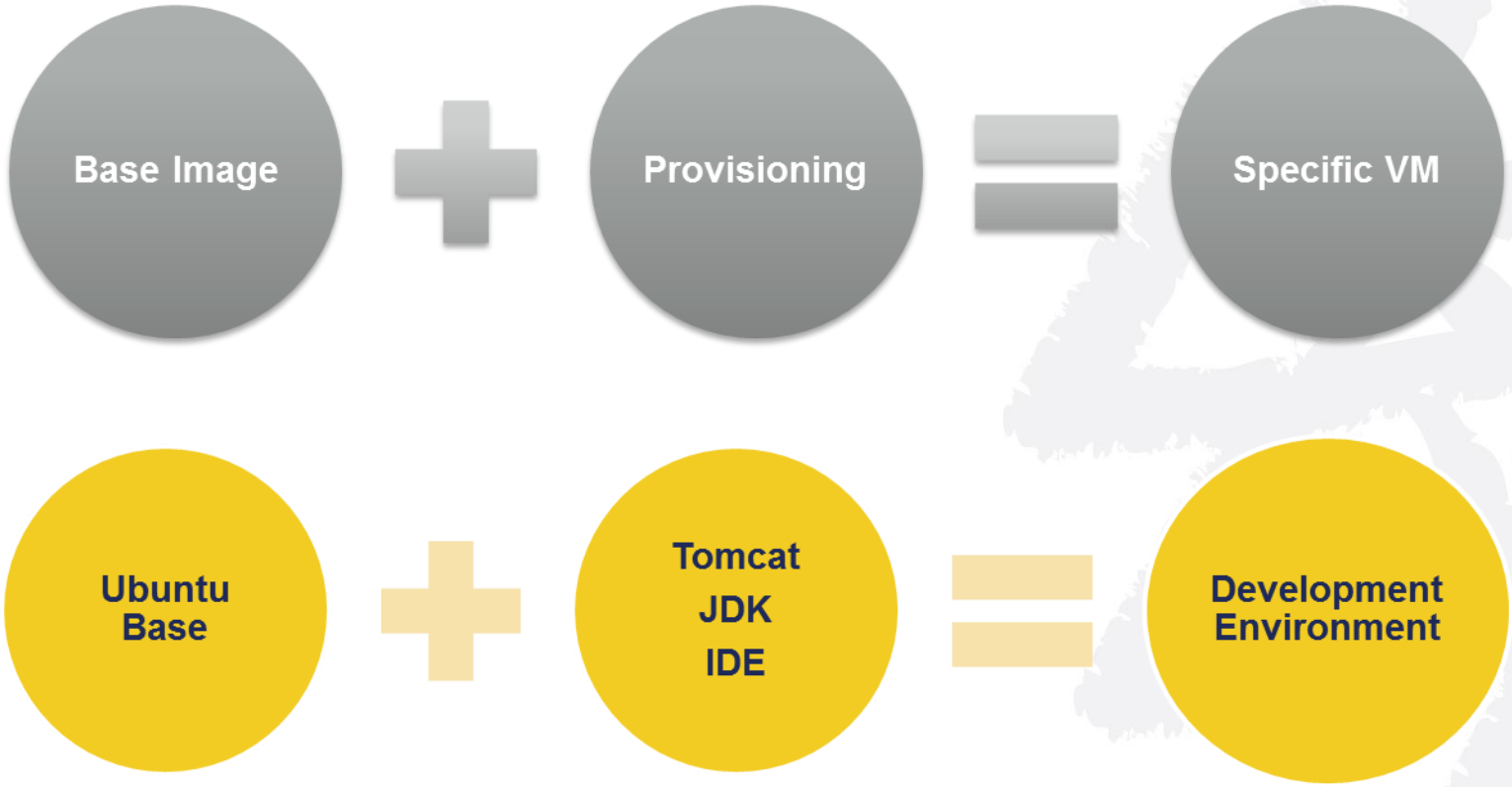
```
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "phusion/ubuntu-14.04-amd64"
  config.vm.provider "virtualbox" do |vb|
    vb.customize ["modifyvm", :id, "--memory", "1024"]
       vb.customize ["modifyvm", :id, "--cpus", "1"]
  end
  config.vm.provision "puppet" do |puppet|
    puppet.manifests_path = "puppet/manifests"
    puppet.manifest_file  = "site.pp"
    puppet.module_path    = "puppet/modules"
    puppet.options        = "--verbose --debug"
  end
 config.vm.network "private_network", ip: "192.168.33.10"
end
```

Provisioning

Base Box

- Pre-assembled Vagrant VM image, ready-to-run
- Custom build possible
- Base Box is base for further provisioning
- Use „Packer“ to create base box

Provisioning



Provisioning

Small Base Box

High flexibility

Long duration of provisioning

Bigger Base Box

Low flexibility

Shorter duration of provisioning

ANSIBLE

Why Ansible

Simplicity

Chef, Puppet, Salt are great tools as well, may be more complex to start with, steeper learning curve, etc.

For larger roll-outs: Know your requirements and quality-attributes and evaluate different products

Other Business Drivers?

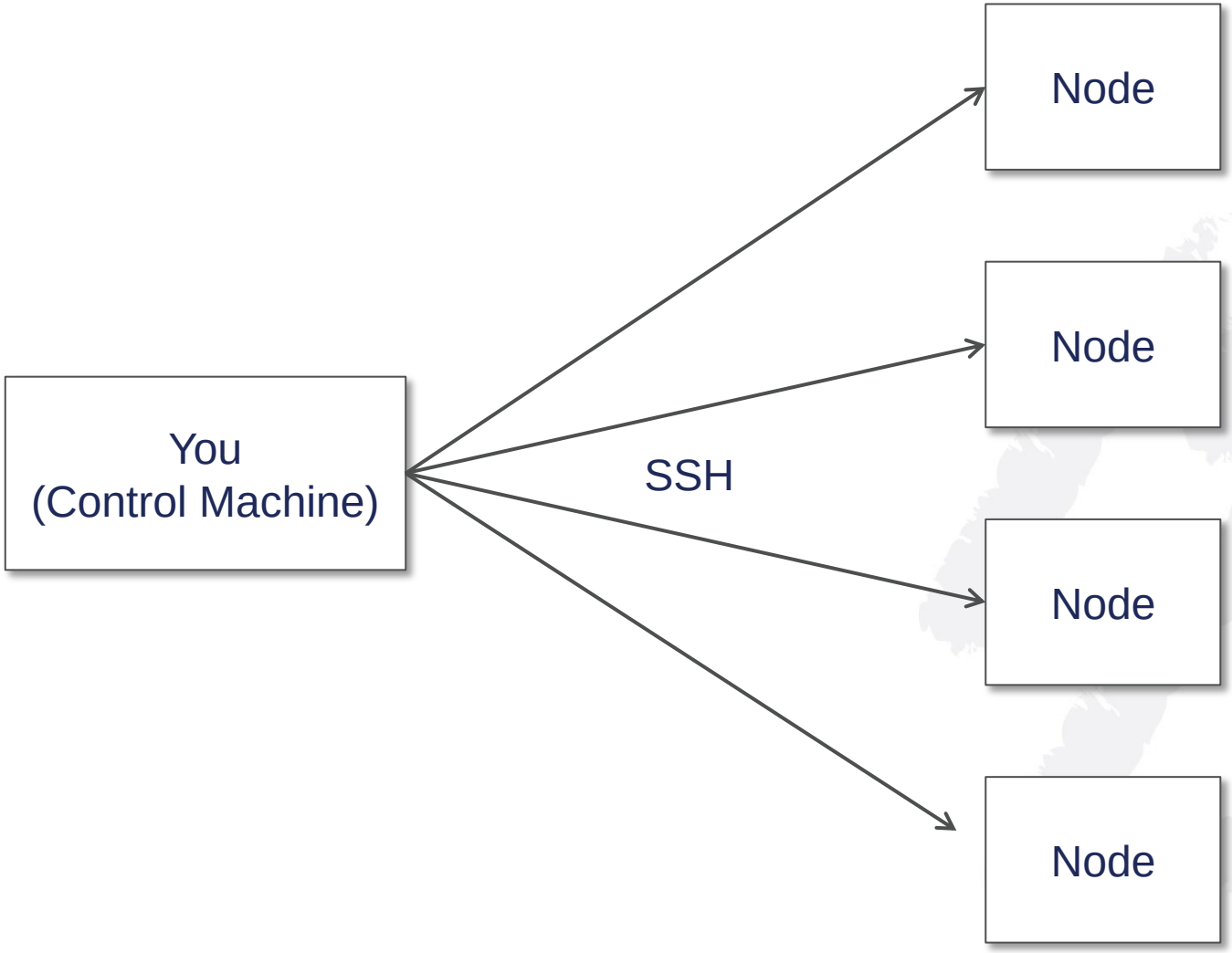
- **Transparency**
  - System definition at central place
  - System definition is clearly structured and comprehensible
  - Reporting of changes
- **Automatization**
  - System build on demand
  - Not only initially, but also over the whole lifecycle
- **Reproducibility**
  - System build is reliably reproducible via the definition file
  - Changes are versionable

Pre-conditions to use Ansible

Ansible installed on control machine

Python required on all managed nodes/servers

ssh public-key setup to connect to hosts



Ansible works via ssh. No agent on node required.

Playbook

- Written in YAML
- Declare configurations / desired state
- Orchestrate steps of any manual ordered process
- Can launch tasks (async and sync)
- Kept in source control

Playbook Sample

```
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service: name=httpd state=started
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

Playbook Sample

Restrictions

```
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service: name=httpd state=started
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

Playbook Sample

```
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service: name=httpd state=started
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

Tasks

Playbook Sample: Tasks

```
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: roo
  tasks:
  - name: ensure apache is at the latest version
    yum: pkg=httpd state=latest
  - name: write the apache config file
                rc=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service: name=httpd state=started
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

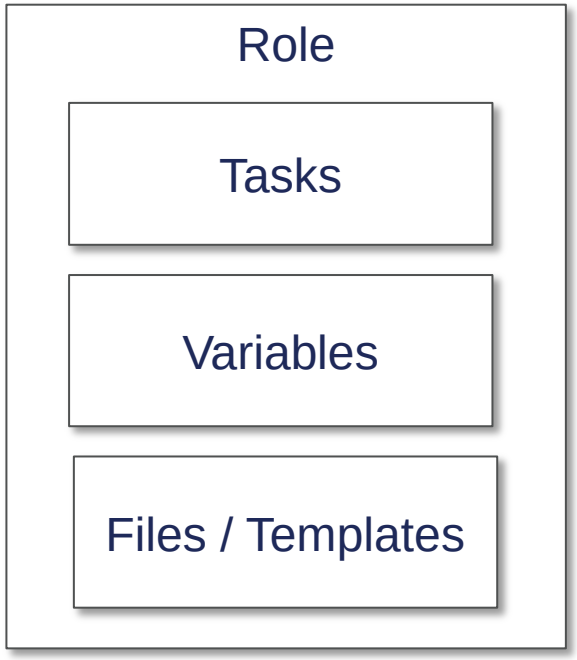
Tasks

Documentation / Reference

Module

Arguments

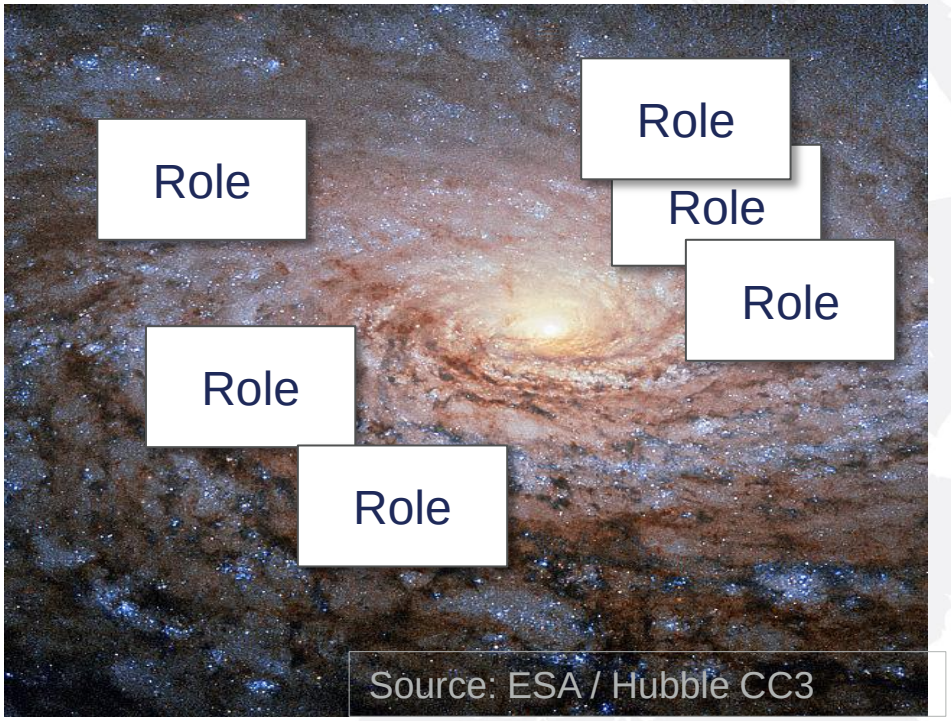
Playbook Sample



- Unit of reuse for system components (e.g. nginx role)
- Contains all tasks, handlers, variables, files and templates for a component configuration
- Follows a directory layout convention

ANSIBLE

GALAXY



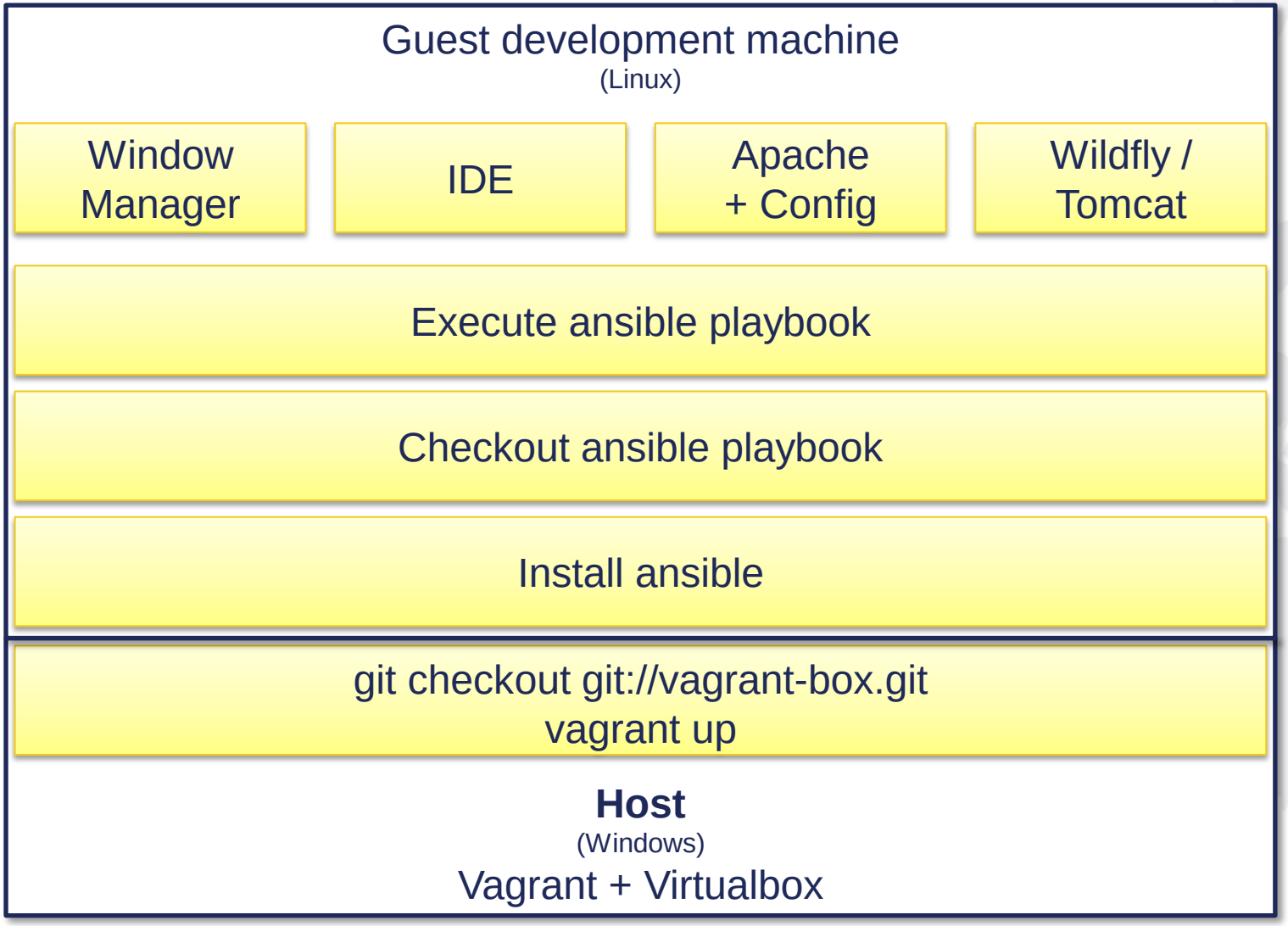
Roles as unit of reuse are published in the „ansible galaxy“

Ansible Galaxy is your hub for finding, reusing and sharing the best Ansible content.

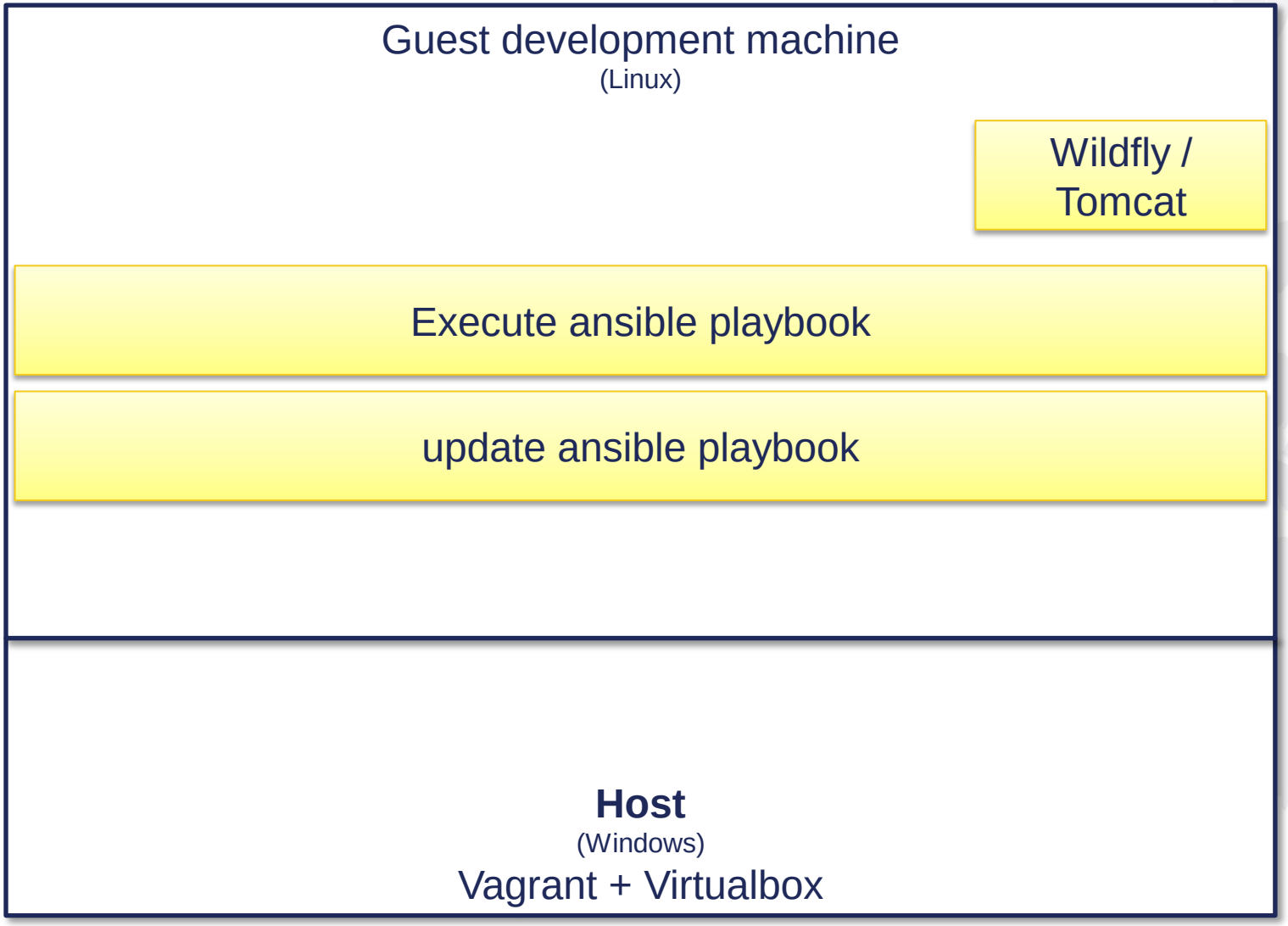
Back to Vagrant

Development environment

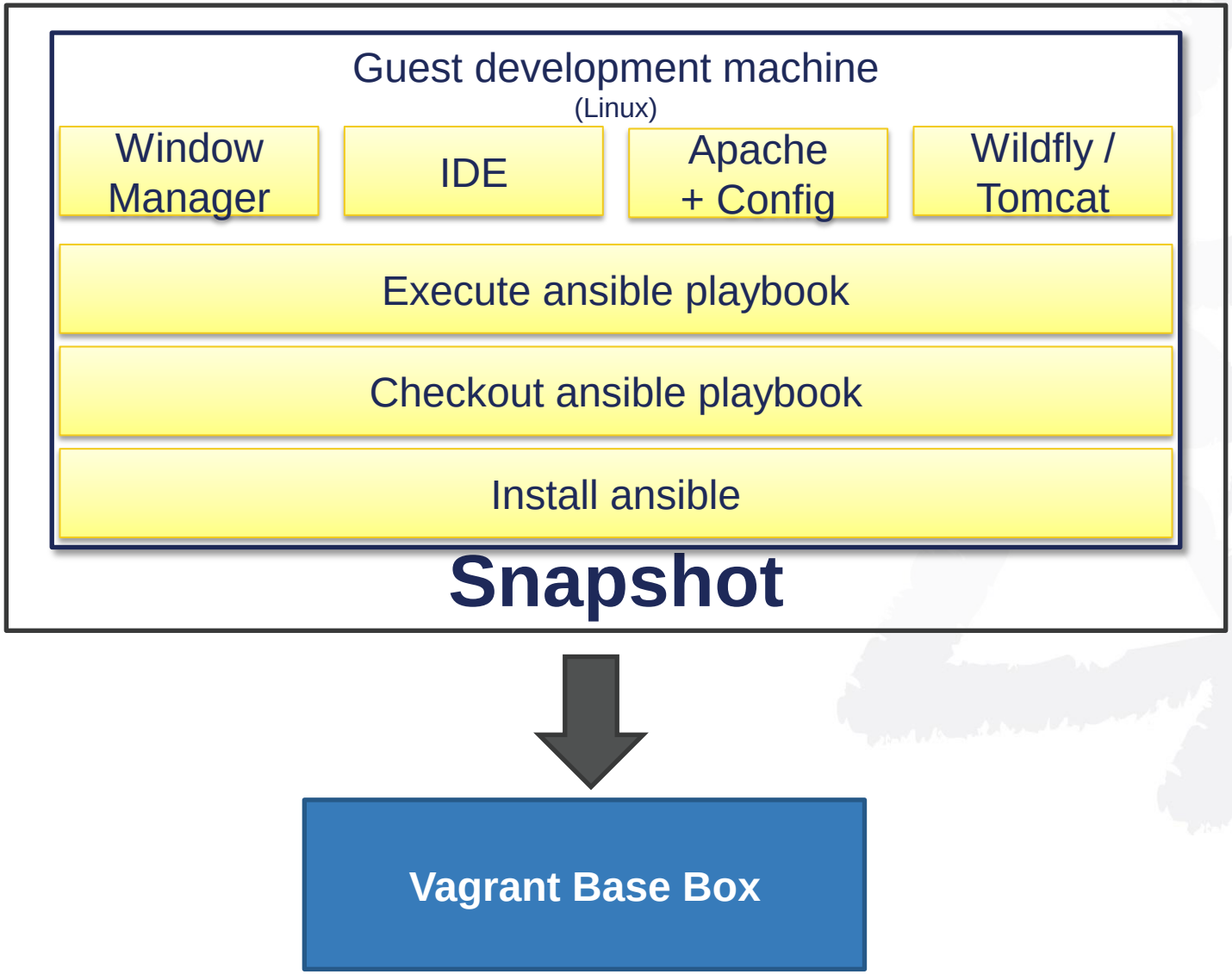
Initial Installation



Use Case: Update Wildfly



Long Provisioning Times



Long Provisioning Times

- **Create a Snapshot of current VM state as your new base box**
- **Start provisioning changes based on this new base box**

Development environment

< 5 machines

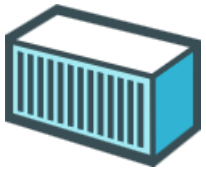
Development environment

< 5 machines

„resource-hungry“

More independent VMs?
Build Once Run Anywhere?

Build



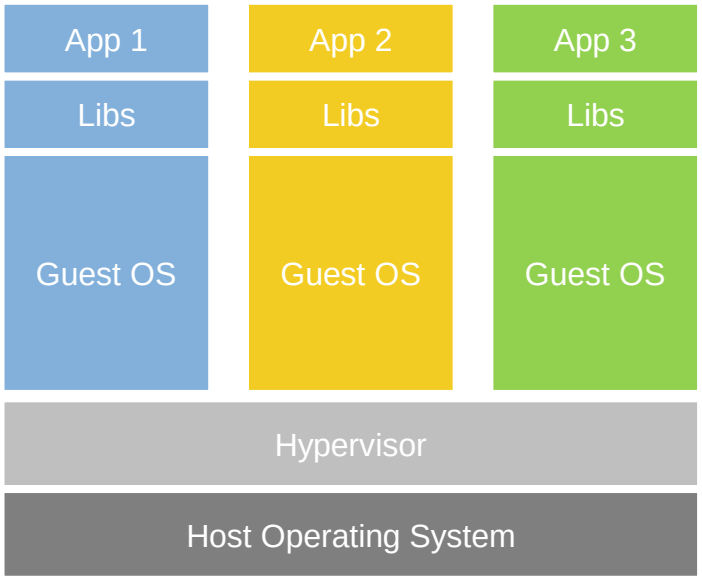
Ship

Run

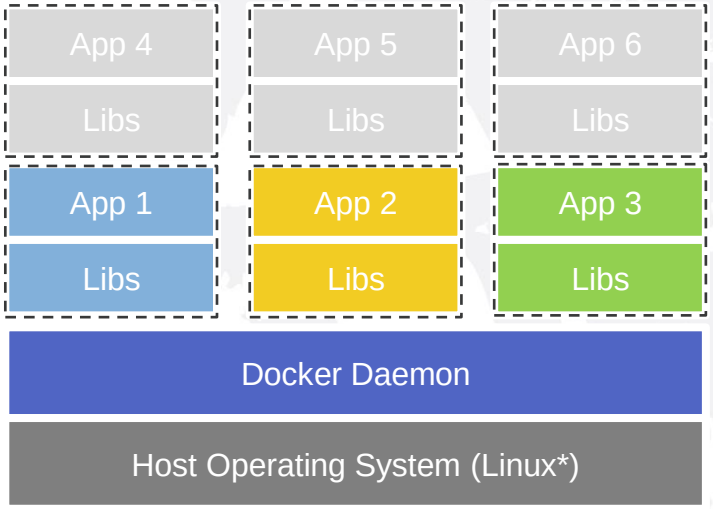
Bildquelle / URL

BSD Jails / Solaris Zones
Linux Containers
Docker Container / Images

Docker Overview



Virtual Machines



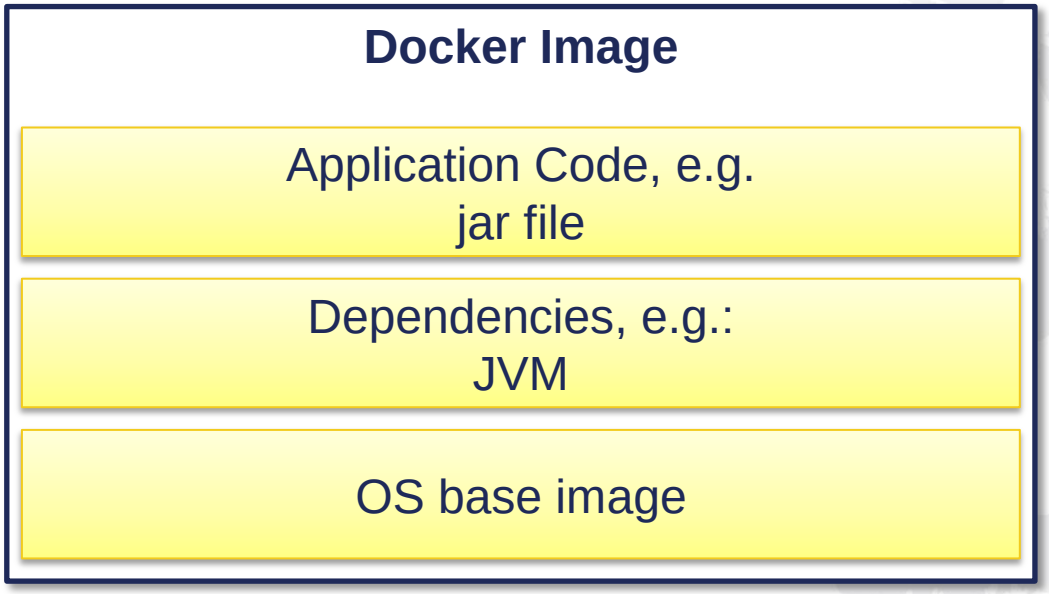
Docker Container

* Windows Server 2016 TP3:
supports docker containers

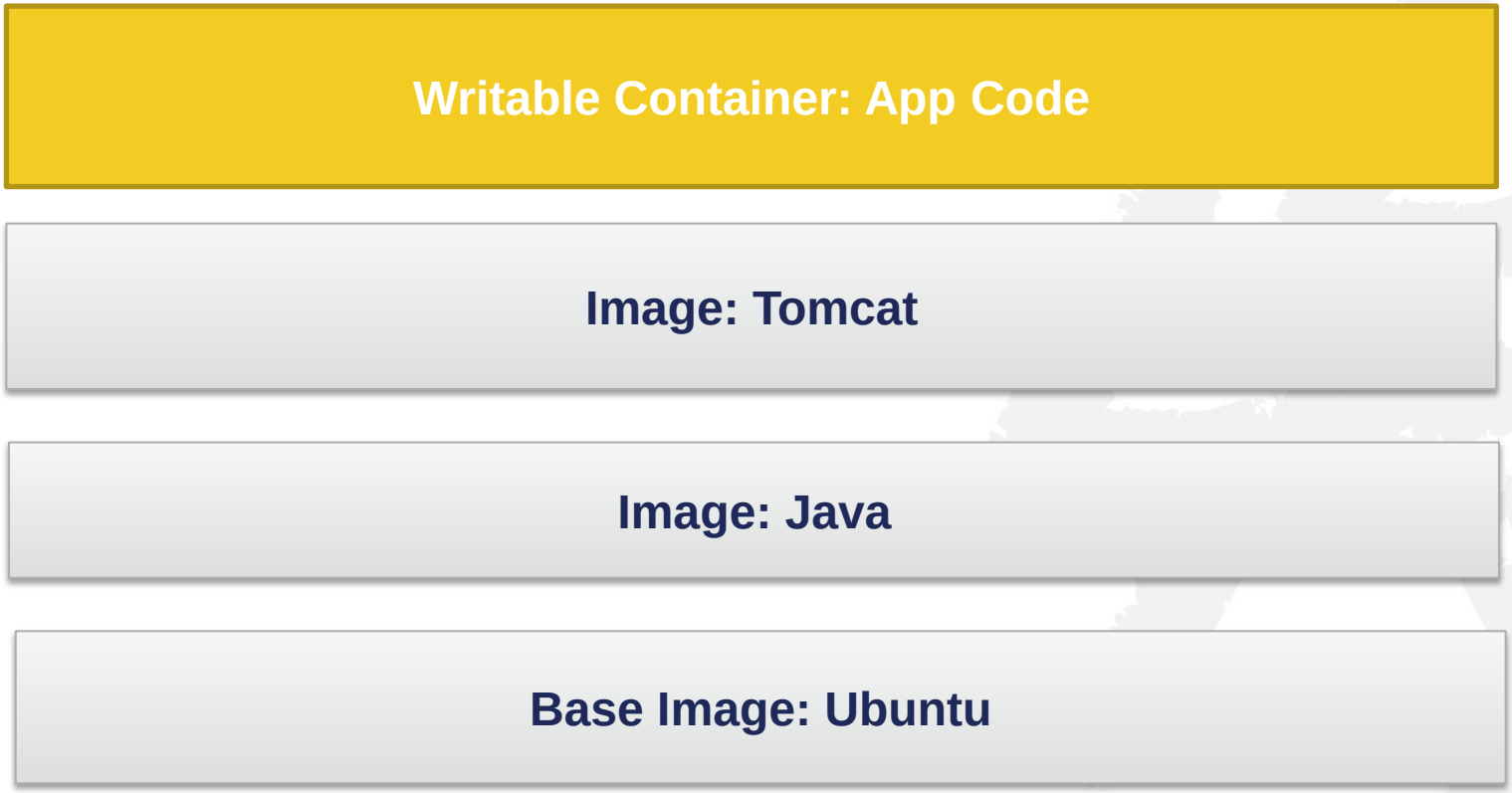
Starting a „bash“

docker run –it ubuntu bash

Inside a docker image



Layered FS


Writable Container: App Code
Image: Tomcat
Image: Java
Base Image: Ubuntu

Dockerfile: Spring Boot Application Container

```
FROM rattermeyer/ubuntu-jdk:1.0
maintainer richard.attermeyer@gmail.com

ENV PROJECT_VERSION 0.0.1-SNAPSHOT
ENV PROJECT_NAME todo-list-backend

RUN mkdir /opt/${PROJECT_NAME}
ADD ${PROJECT_NAME}-${PROJECT_VERSION}.jar /opt/${PROJECT_NAME}/

EXPOSE 8080

ENTRYPOINT java -jar /opt/${PROJECT_NAME}/${PROJECT_NAME}-
${PROJECT_VERSION}.jar
```

Base Image

Dockerfile: Spring Boot Application Container

```
FROM rattermeyer/ubuntu-jdk:1.0
maintainer richard.attermeyer@gmail.com

ENV PROJECT_VERSION 0.0.1-SNAPSHOT
ENV PROJECT_NAME todo-list-backend

RUN mkdir /opt/${PROJECT_NAME}
ADD ${PROJECT_NAME}-${PROJECT_VERSION}.jar /opt/${PROJECT_NAME}/

EXPOSE 8080

ENTRYPOINT java -jar /opt/${PROJECT_NAME}/${PROJECT_NAME}-
${PROJECT_VERSION}.jar
```

Provisioning

Dockerfile: Spring Boot Application Container

```
FROM rattermeyer/ubuntu-jdk:1.0
maintainer richard.attermeyer@gmail.com

ENV PROJECT_VERSION 0.0.1-SNAPSHOT
ENV PROJECT_NAME todo-list-backend

RUN mkdir /opt/${PROJECT_NAME}
ADD ${PROJECT_NAME}-${PROJECT_VERSION}.jar /opt/${PROJECT_NAME}/

EXPOSE 8080

ENTRYPOINT java -jar /opt/${PROJECT_NAME}/${PROJECT_NAME}-
${PROJECT_VERSION}.jar
```

Adding files

Dockerfile: Spring Boot Application Container

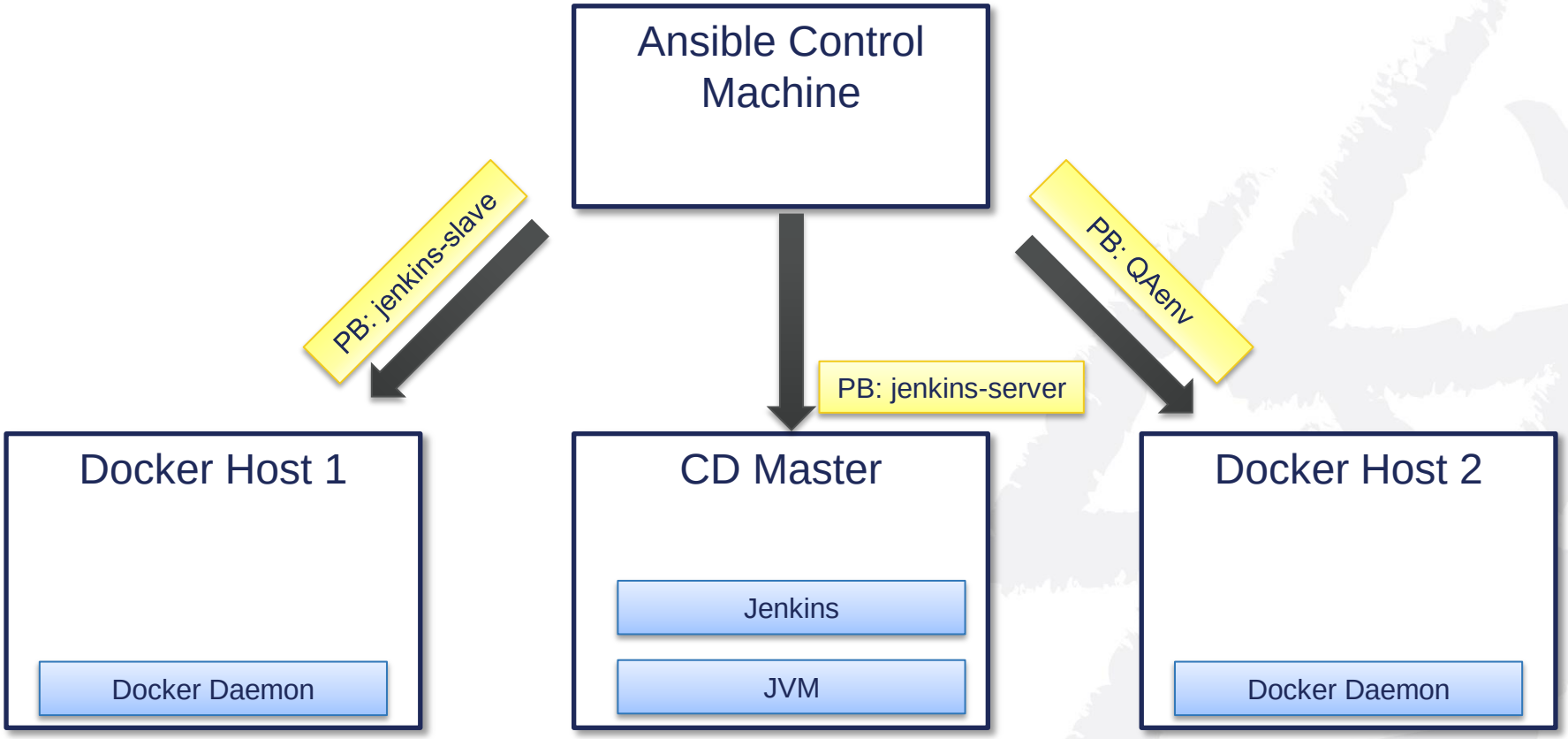
```
FROM rattermeyer/ubuntu-jdk:1.0
maintainer richard.attermeyer@gmail.com

ENV PROJECT_VERSION 0.0.1-SNAPSHOT
ENV PROJECT_NAME todo-list-backend

RUN mkdir /opt/${PROJECT_NAME}
ADD ${PROJECT_NAME}-${PROJECT_VERSION}.jar /opt/$

EXPOSE 8080

ENTRYPOINT java -jar /opt/${PROJECT_NAME}/${PROJECT_NAME}-
${PROJECT_VERSION}.jar
```

Starting Point

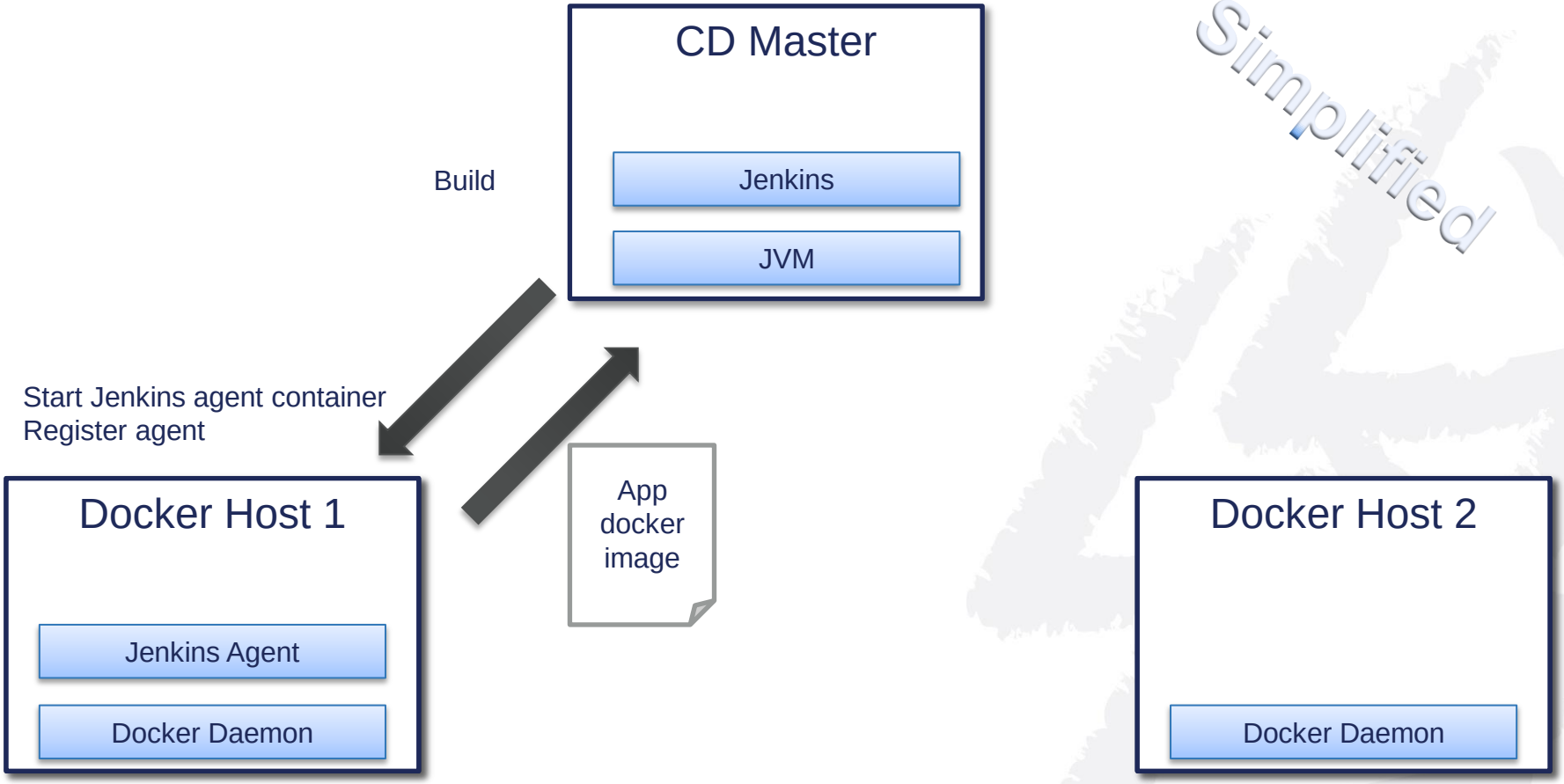
Summary

- **Lightweight**
  Docker Images are much more lightweight than full VMs. The start takes seconds. The images for distribution are normally smaller (only Delta, new FS Layer)
- **Image under version control**
  Hence easier handling of builds. And thus better suited for a Continuous Delivery Pipeline
- **Lots of base images** (again)
- **Don’t rely on environment: Create your own environment**

Use Cases, costs and benefits

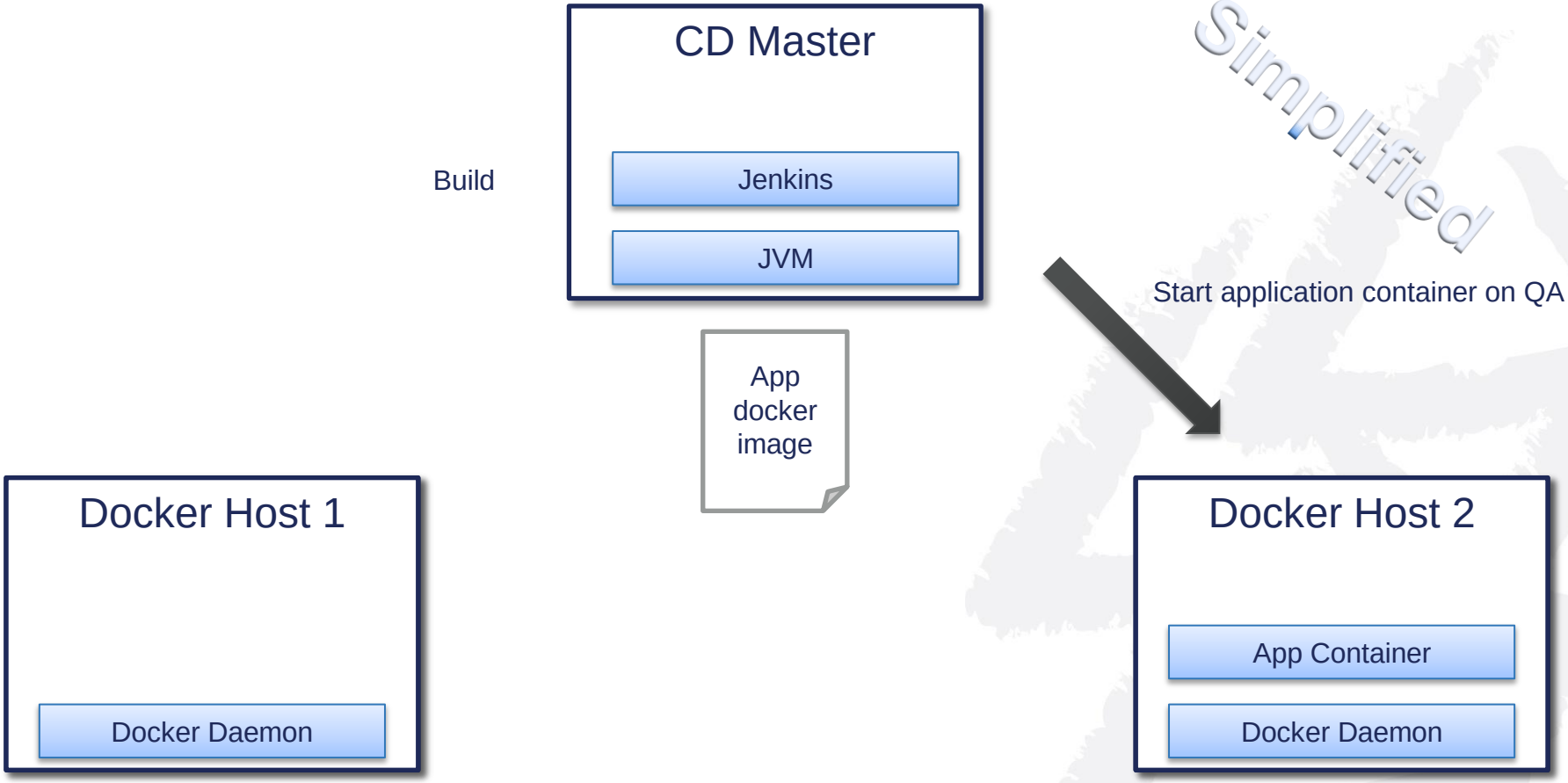


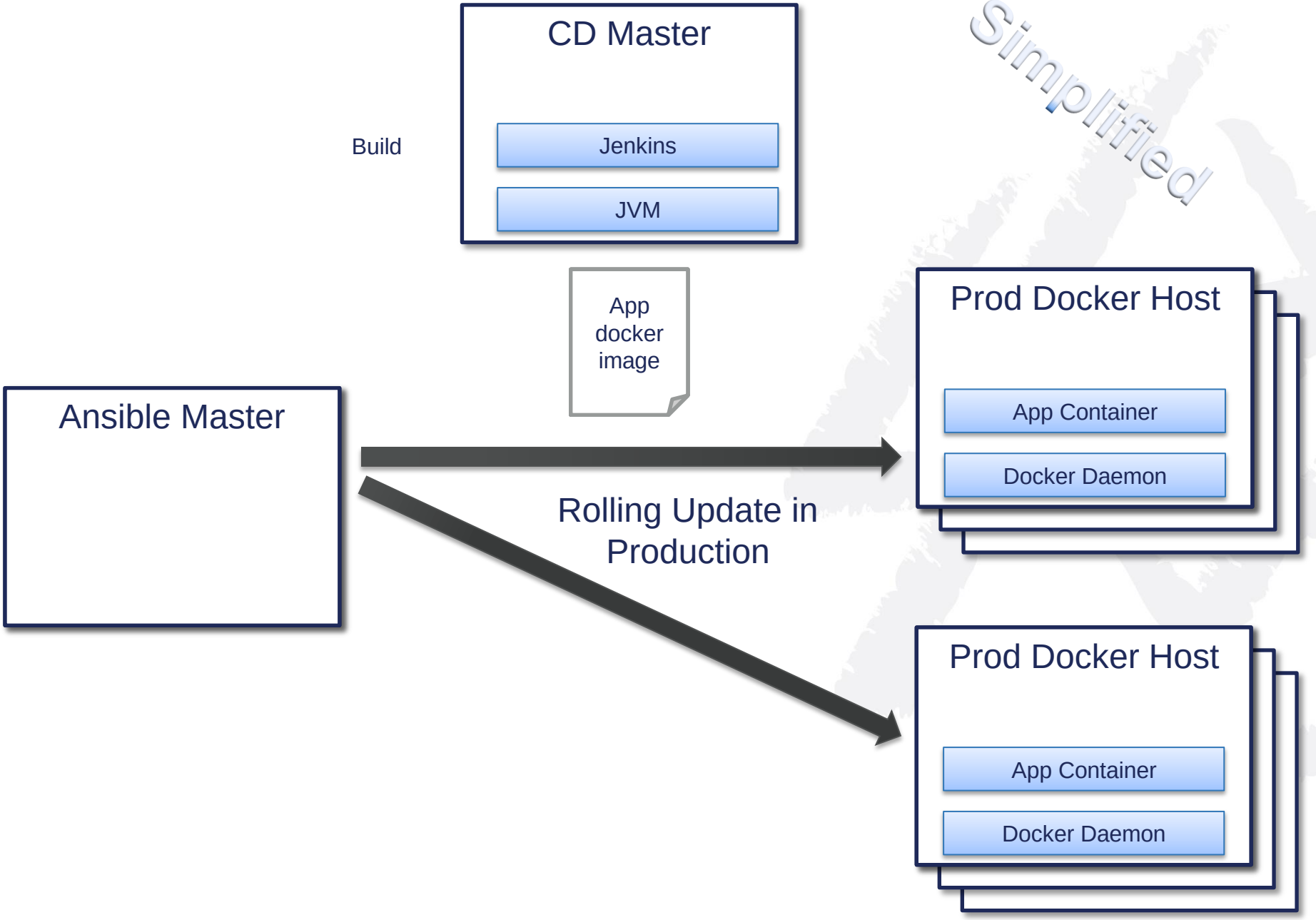
PB= Ansible Playbook



Execute Build
Create docker Image for application
Push image into docker registry
Removes Jenkins agent container

Simplified


Simplified
CD Master
Jenkins
JVM
Build
App docker image
Start application container on QA
Docker Host 1
Docker Daemon
Docker Host 2
App Container
Docker Daemon


Simplified
CD Master
Build
Jenkins
JVM
App docker image
Ansible Master
Rolling Update in Production
Prod Docker Host
App Container
Docker Daemon
Prod Docker Host
App Container
Docker Daemon

Recap

- Vagrant to create development environment (on developer‘s laptop)
- Provision development environment with ansible
- Provision CD, test and production environment with ansible
- Build your project in seperate docker containers
- Start QA environment based on docker containers
- Roll-out to production using ansible

Outlook

Ecosystem

Noel Pennington, Sunflower

Many new projects

Few experiences in enterprise environments

Today hip, tomorrow out

Questions?

Contact details

Richard Attermeyer
Senior Solution Architect

OPITZ CONSULTING Deutschland GmbH
richard.attermeyer@opitz-consulting.de
Telefon +49 2261 60 01-1713
Mobile +49 173 727 9004

youtube.com/opitzconsulting

@OC_WIRE

slideshare.net/opitzconsulting

xing.com/net/opitzconsulting

Image references