qaware.de

QA|WARE
SOFTWARE ENGINEERING

# Chapter 3: Virtualization

Lukas Buchner
lukas.buchner@qaware.de

# Requirements Cloud Computing

- **Organization of Resources** in a data center
- More efficient usage of given resources to **minimize costs**
- **Isolation of resources**. Customers should not see and influence others. Avoid side effects. Security
- **Decoupling from the hardware** for more operational flexibility and robustness in the event of failures
- Resources should be allocated flexibly. Control using **software defined resources**

- The solution to these requirements is **virtualization**, which makes **cloud computing** possible in the first place..

# Virtualization

**Virtualization**: the creation of virtual realities and their mapping onto physical reality.

Purpose:

- **Multiplicity:** Creation of multiple virtual realities within a single physical reality
- **Decoupling:** Dissolve the bond and dependency on reality
- **Isolation:** Avoiding physical side effects between virtual realities

# Virtualization types

**Virtualization** is representative of several fundamentally different concepts and technologies.

## Virtualization of hardware infrastructure

1. Emulation
2. Full virtualization (Type 2 virtualization)
3. Para virtualization (Type 1 virtualization)

## Virtualization of software infrastructure

4. Operating system virtualization (*Containerization*)
5. Application virtualization (*Runtime*)

# Virtualization and Cloud Computing

- **Decoupling from the hardware** for more operational flexibility and robustness in the event of failures.

- **Standardization of resource capacities** on heterogeneous and changing hardware (“S instance”, “XL instance”).

- **Centralized control and provision of computing resources** about the software-defined resources provided by virtualization.

# Virtualization types:
Hardware virtualization

QA|WARE

# What is being virtualized?

- **Processor**
  - Virtual cores
  - Dispatching processor instructions to real cores
- **Memory**
  - Virtual main memory partition
  - Management of the real representation (in RAM, on hard disk, ballooning)
- **Network**
  - Virtual network interfaces and virtual network infrastructures (VLAN)
  - Bridges between virtual and real networks
- **Storage**
  - Virtual hard disk drives. Mapping to files in the real file system. Volumes either pre-allocated or dynamically growing.
  - Virtual SANs (Storage Area Networks) via distribution of a virtual disk drive data to many storage units.

# What is being virtualized?

QA|WARE

- **GPU**
  - Frame buffer (2D-Array with pixel data)
  - 3D-Functionality (DirectX, OpenGL), see https://en.wikipedia.org/wiki/GPU_virtualization
  - Computing (AI, Simulations) (Increasingly important for the Cloud)
- **Peripherals such as USB, mouse, keyboard**
- **Timer, Interrupt Controller**

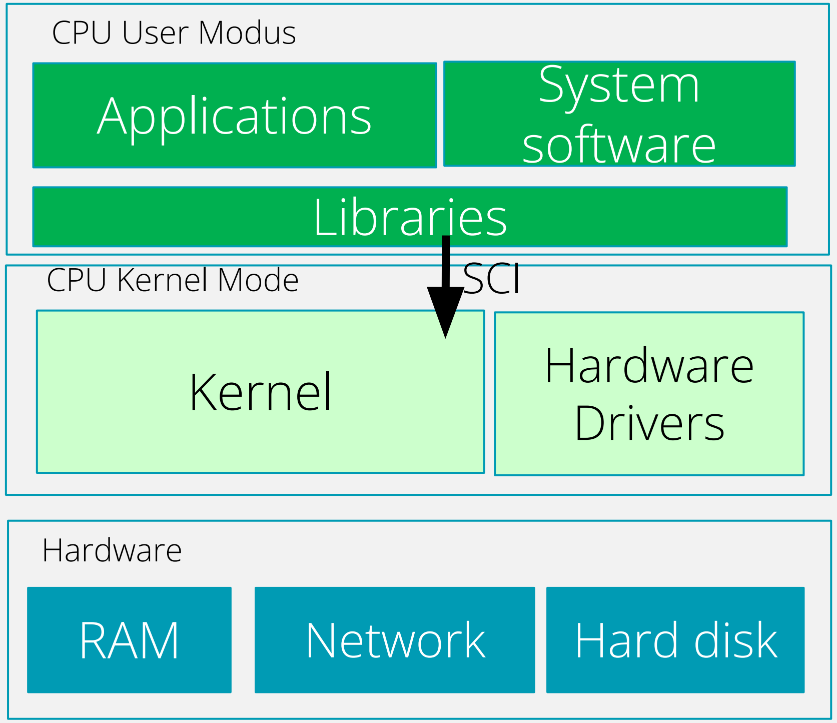
# Classic structure of an operating system with computer architecture support

### CPU User Mode

- Lowest authorization level
- No direct hardware access
- Memory protection via the memory management unit

### CPU Kernel Mode

- User Mode calls the kernel via the System Call Interface (SCI).
- Currently, the SCI in Linux consists of about 380 system calls.
- Highest level of authorization
- Privileged CPU instructions
- Access to hardware via drivers
- Takes over, for example, file system management and application scheduling



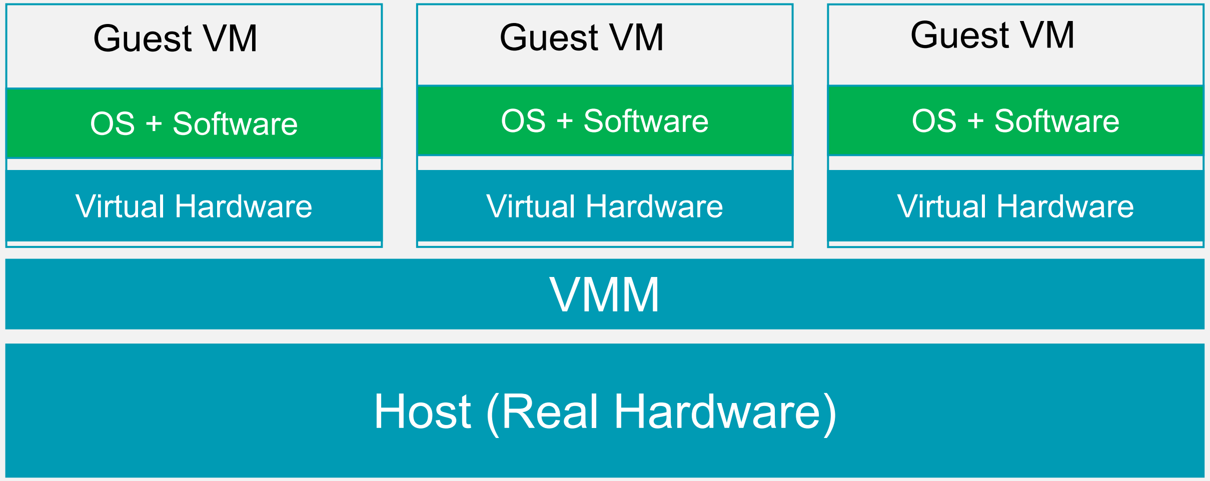
# Hardware support

- **Software based virtualization**
  - In the classic operating model, only the user mode has the necessary isolation properties. The kernel mode can be emulated here, e.g. using “trap-and-emulate”, with at least a 10% performance loss.
- **Hardware assisted virtualization**
  - CPU extensions such as Intel-VT and AMD-V were therefore often developed. These extensions add a new processor mode (e.g. virtual execution mode) in which the guest operating system perceives itself as working with full privileges, but the host operating system remains protected.
- Virtual main memory partition in real physical memory. (The zero is shifting). Management of the real representation using the management memory unit (MMU).
- For the passthrough of the interfaces of real hardware devices, the displacement of the zero must be compensated by an IOMMU (I/O Memory Management Unit).

# Hardware virtualization

- **Hardware virtualization** divides the resources of a computer system so that they can be used by multiple independent operating system instances.

- The requirements of the operating system instances are intercepted by the virtualization software (**virtual machine monitor, VMM**) and implemented on the actual hardware.



**Host**

- The computer that runs one or more virtual machines and provides the necessary hardware resources.

**Guest**

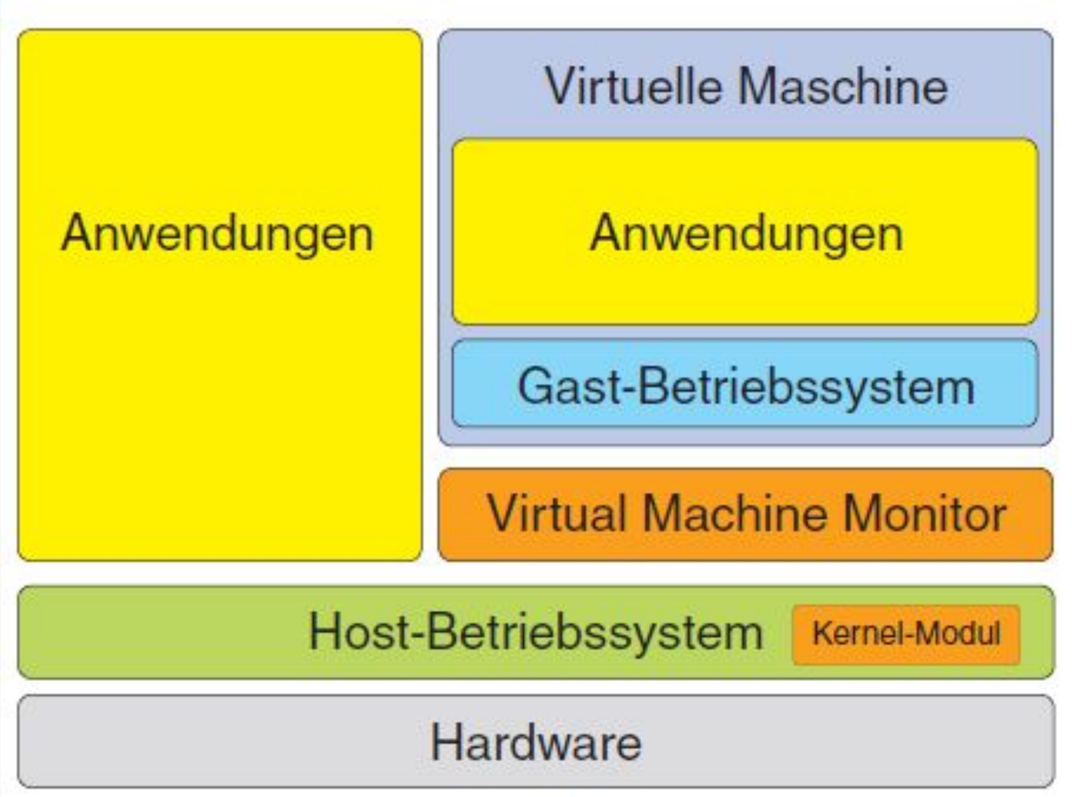
- A runnable / running virtual machine

**VMM** (Virtual Machine Monitor)

- The control software for managing guests and host resources

# Hardware virtualization: Full virtualization

- Each guest operating system has its own virtual computer with virtual resources such as CPU, main memory, disk drives, network cards, etc.
- The VMM runs as an application hosted by the host operating system (type 2 hypervisor)
- The VMM distributes the computer's hardware resources among the VMs
- The VMM partially emulates hardware that is not designed for simultaneous access by multiple operating systems (e.g. network cards, graphics cards)
- Performance loss: 5-10%.



# Hardware virtualization: para-virtualization

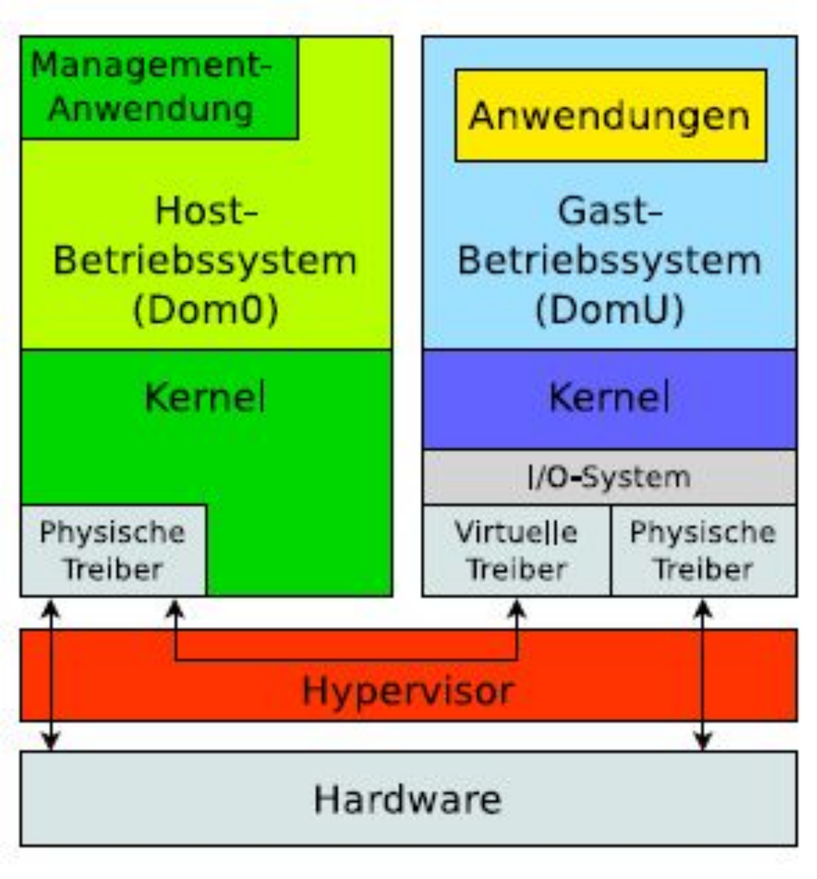
The hypervisor runs directly on the available hardware. It is therefore an operating system that is exclusively designed for virtualization.

Virtual drivers must be added to the guest operating system in order for it to interact with the hypervisor.

No directly low-level virtualized hardware resources (CPU, RAM, etc.) are available to the guest operating system, but an API is available for use by the virtual drivers.

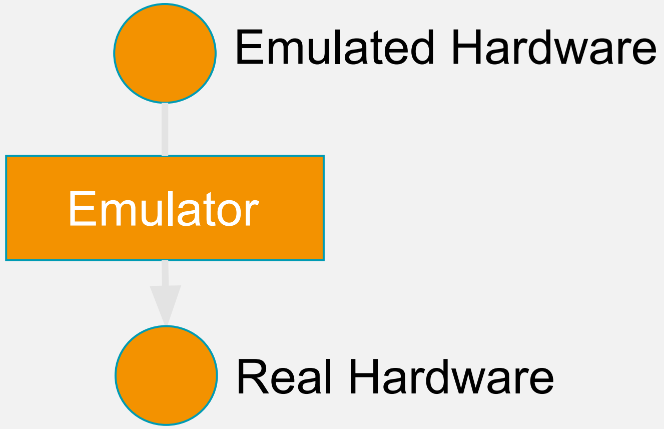
- Supported operating systems and hardware variants from the guest's point of view are limited per hypervisor implementation.
- The hypervisor uses the drivers of a host operating system to access the real hardware. This eliminates the need for the hypervisor to implement its own drivers.

Performance loss: 2-3%



# For completeness:
# What is emulation and application virtualization?

- Emulation: Recreates the hardware of a non-existent or incompatible computer system or parts of a corresponding computer system.
- Purpose, among other things: to preserve old software.
- (Example: Gameboy in the browser)



Emulated Hardware

Emulator

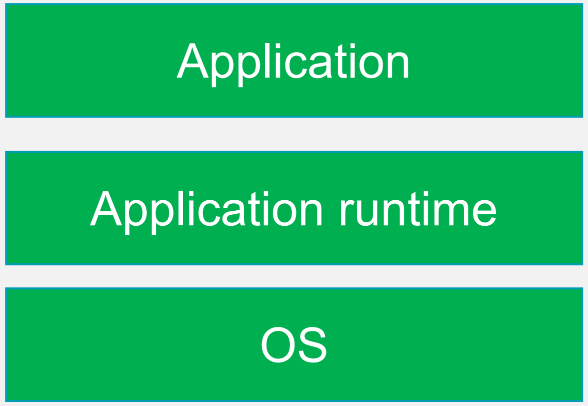
Real Hardware

# Virtualization, but high-performance

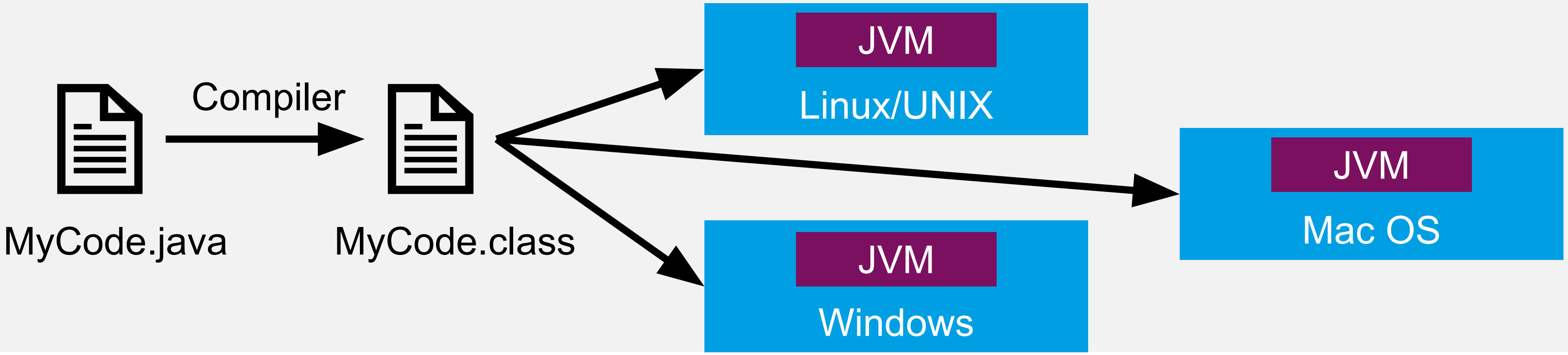
- Emulations fulfill many of the requirements for cloud computing, such as isolation and decoupling. However, they are very slow when replicating an entire computer architecture and are therefore unsuitable for mass productive use.
  - The main culprit is the **CPU**.
- Can the same goals be achieved with minimal additional resource expenditure?
  - Answer: Yes, but only if the guest architecture of the virtualized system is the same as the host architecture.
  - x86 host → x86 guest

# For completeness:
# What is emulation and application virtualization?

- Application virtualization: provides applications with a programming interface and a runtime environment that is completely decoupled from the underlying operating system.
- Purpose, among other things: portable applications.



- Beispiel: JVM



# Virtualization in the enterprise environment

In addition to the advantages mentioned so far, today's VM solutions offer many more features:

- **Resource management during operation**
  - Memory ballooning: dynamically increasing main memory
  - Changing the number of virtual cores
  - Changing disk size with virtual SANs (Storage Area Networks)
- **Live Migration**: Moving the running physical machine to a different hardware within milliseconds
  - CPU State
  - Memory
  - Storage
  - Network
- Attention: Creating (genuine) randomness is even more difficult in a VM than with real hardware (e.g. using mouse and keyboard input).
- This is where hypervisors provide interfaces to obtain additional randomness.

# Hardware virtualization with Vagrant and VirtualBox

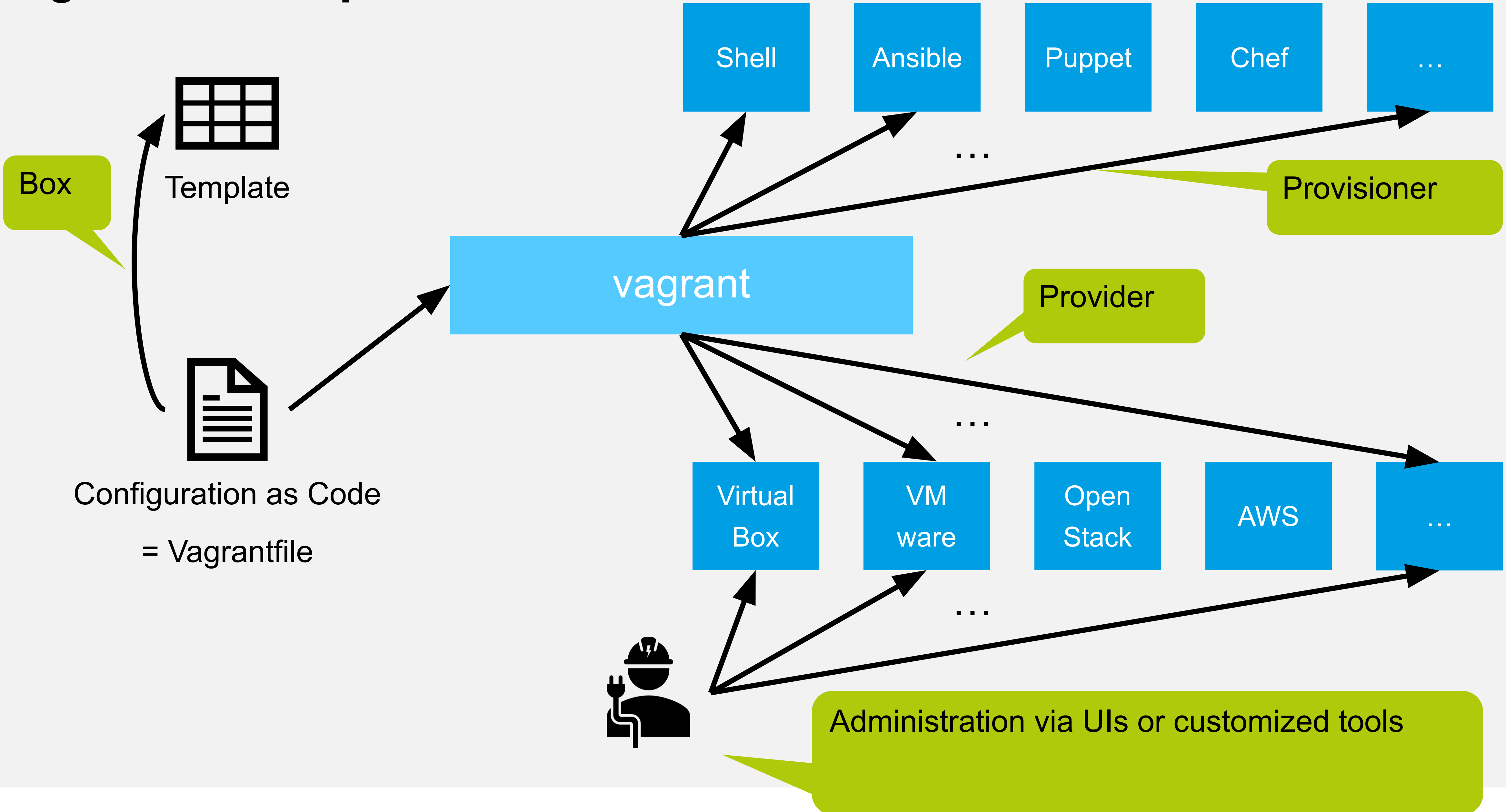
# Hardware virtualization with Vagrant and VirtualBox

+

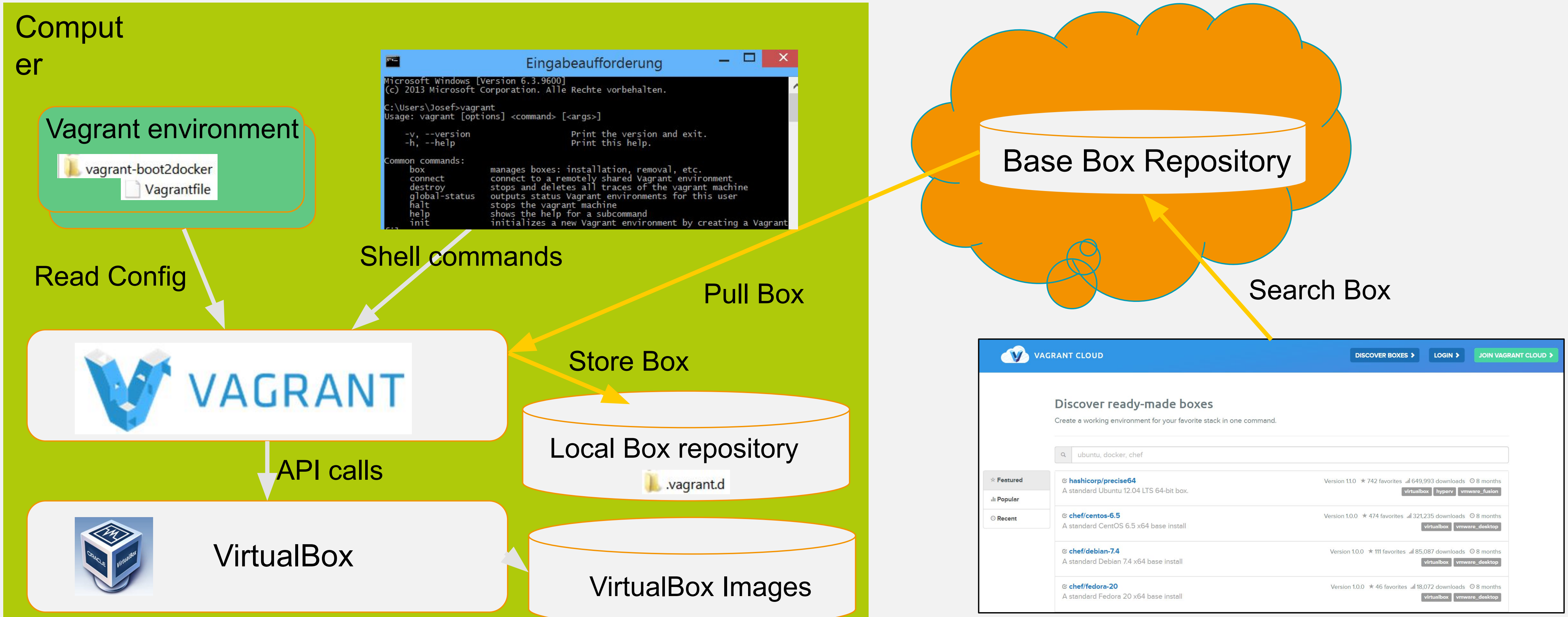
Open Source Type 2 full virtualization software for Windows, Linux, MacOS and Solaris.

Automation software for virtual environments on a computer. Create and control virtual machines via command line.

# Vagrant: Concepts



# Vagrant: A schematic overview.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Josef>vagrant
Usage: vagrant [options] <command> [<args>]

    -v, --version                    Print the version and exit.
    -h, --help                       Print this help.

Common commands:
     box             manages boxes: installation, removal, etc.
     connect         connect to a remotely shared Vagrant environment
     destroy         stops and deletes all traces of the vagrant machine
     global-status   outputs status Vagrant environments for this user
     halt            stops the vagrant machine
     help            shows the help for a subcommand
     init            initializes a new Vagrant environment by creating a Vagrant
```

Discover ready-made boxes

Create a working environment for your favorite stack in one command.

hashicorp/precise64 — Version 1.1.0 ★ 742 favorites 649,993 downloads 8 months
A standard Ubuntu 12.04 LTS 64-bit box. — virtualbox hyperv vmware_fusion

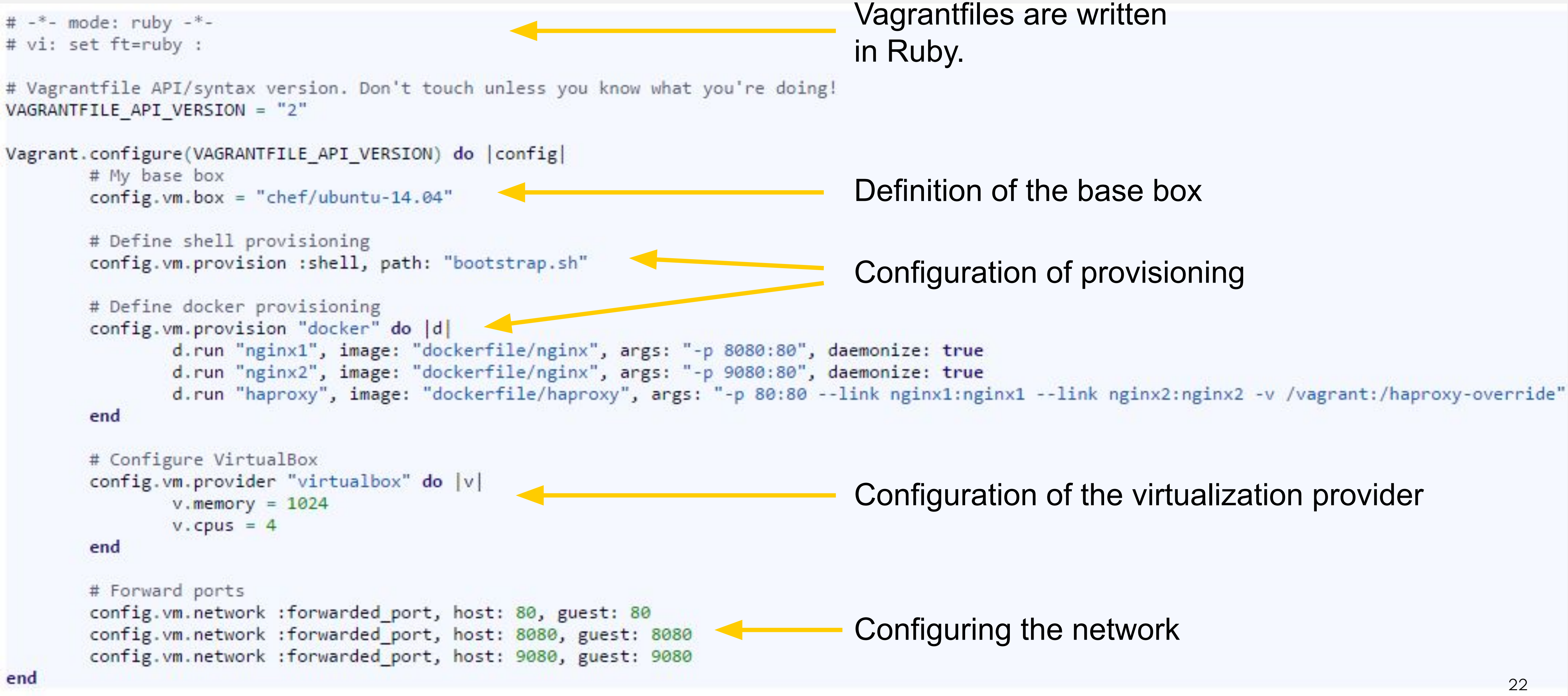
chef/centos-6.5 — Version 1.0.0 ★ 474 favorites 321,235 downloads 8 months
A standard CentOS 6.5 x64 base install — virtualbox vmware_desktop

chef/debian-7.4 — Version 1.0.0 ★ 111 favorites 85,087 downloads 8 months
A standard Debian 7.4 x64 base install — virtualbox vmware_desktop

chef/fedora-20 — Version 1.0.0 ★ 46 favorites 18,072 downloads 8 months
A standard Fedora 20 x64 base install — virtualbox vmware_desktop

https://vagrantcloud.com

# The Vagrantfile describes the virtual machine to be created.



```ruby
# -*- mode: ruby -*-
# vi: set ft=ruby :

# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
        # My base box
        config.vm.box = "chef/ubuntu-14.04"

        # Define shell provisioning
        config.vm.provision :shell, path: "bootstrap.sh"

        # Define docker provisioning
        config.vm.provision "docker" do |d|
                d.run "nginx1", image: "dockerfile/nginx", args: "-p 8080:80", daemonize: true
                d.run "nginx2", image: "dockerfile/nginx", args: "-p 9080:80", daemonize: true
                d.run "haproxy", image: "dockerfile/haproxy", args: "-p 80:80 --link nginx1:nginx1 --link nginx2:nginx2 -v /vagrant:/haproxy-override"
        end

        # Configure VirtualBox
        config.vm.provider "virtualbox" do |v|
                v.memory = 1024
                v.cpus = 4
        end

        # Forward ports
        config.vm.network :forwarded_port, host: 80, guest: 80
        config.vm.network :forwarded_port, host: 8080, guest: 8080
        config.vm.network :forwarded_port, host: 9080, guest: 9080
end
```

Vagrantfiles are written in Ruby.

Definition of the base box

Configuration of provisioning

Configuration of the virtualization provider

Configuring the network

# A typical workflow with Vagrant.

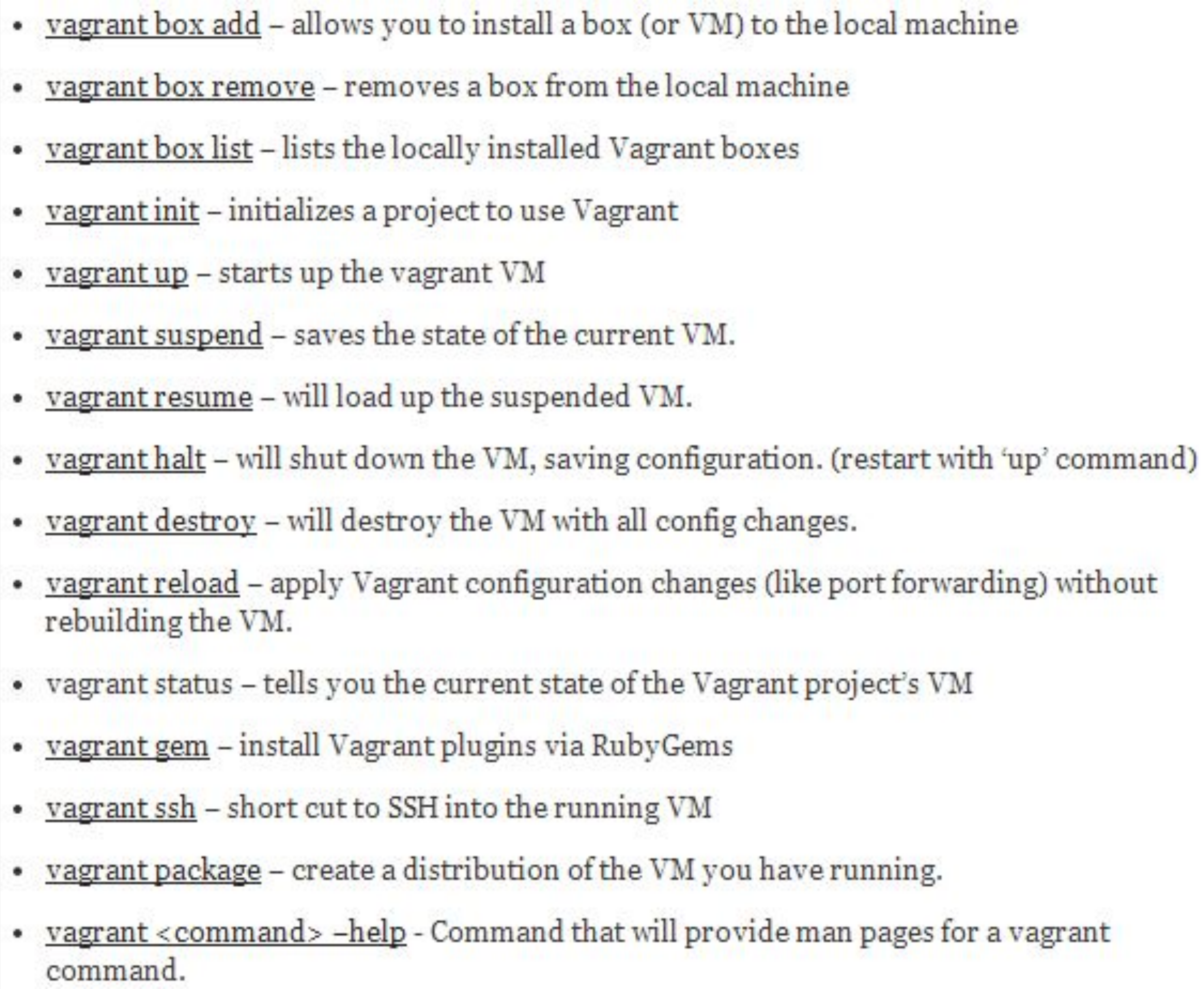
| # | Shell commands | Description |
|---|---|---|
| 1 | `mkdir <box-dir>`<br>`cd <box-dir>` | Create a directory for the Vagrant environment and switch to it |
| 2 | `vagrant init`<br>`[<box-name>]`<br>`[<box-url>]` | Initialize a Vagrant environment. This first creates a Vagrantfile and initializes it with the name and URL of the box (if specified). |
| 3 | | Customize the Vagrantfile as needed (e.g. assign an IP, port mapping between host and guest, directory share between host and guest, etc.) |
| 4 | `vagrant up` | Start the virtual machine (Box □ virtual machine) and configure it according to the Vagrantfile |
| 5 | `vagrant ssh` | Connect to the virtual machine via SSH |
| 6 | `exit` | Exit the SSH command line in the virtual machine |
| 7 | `vagrant halt` | Stop the virtual machine. |

Other useful commands:

- reload: Restarts a VM and updates the configuration according to the Vagrantfile
- package: Recreates a box from a virtual machine

Further commands: http://docs.vagrantup.com/v2/cli/index.html

# Vagrant commands on the command line



- vagrant box add – allows you to install a box (or VM) to the local machine
- vagrant box remove – removes a box from the local machine
- vagrant box list – lists the locally installed Vagrant boxes
- vagrant init – initializes a project to use Vagrant
- vagrant up – starts up the vagrant VM
- vagrant suspend – saves the state of the current VM.
- vagrant resume – will load up the suspended VM.
- vagrant halt – will shut down the VM, saving configuration. (restart with 'up' command)
- vagrant destroy – will destroy the VM with all config changes.
- vagrant reload – apply Vagrant configuration changes (like port forwarding) without rebuilding the VM.
- vagrant status – tells you the current state of the Vagrant project's VM
- vagrant gem – install Vagrant plugins via RubyGems
- vagrant ssh – short cut to SSH into the running VM
- vagrant package – create a distribution of the VM you have running.
- vagrant <command> –help - Command that will provide man pages for a vagrant command.

https://developer.hashicorp.com/vagrant/docs/cli

# Types of virtualization:
# Operating system virtualization

# Hardware virtualizers are heavyweights

- Each **VM** includes a virtual copy of a complete operating system and requires significant **RAM** and **CPU** resources, which are difficult to change dynamically.
- Software development with **VMs** is slower and more complex.
- Due to the size of the **images**, portability is an issue.
- There is no compatibility with other **VM** solutions. Moving between data centers is not easily possible.

# Linux Kernel Namespaces (Isolation through visibility)

A feature of the **Linux kernel** that restricts the view and access to the system:

- process space / process IDs
- network interfaces
- host name
- file system mounts
- IPC (Inter-Process Communication)
- user accounts

The restrictions are transparent to the isolated process.
Namespaces can be nested.

See https://success.docker.com/KBase/Introduction_to_User_Namespaces_in_Docker_Engine

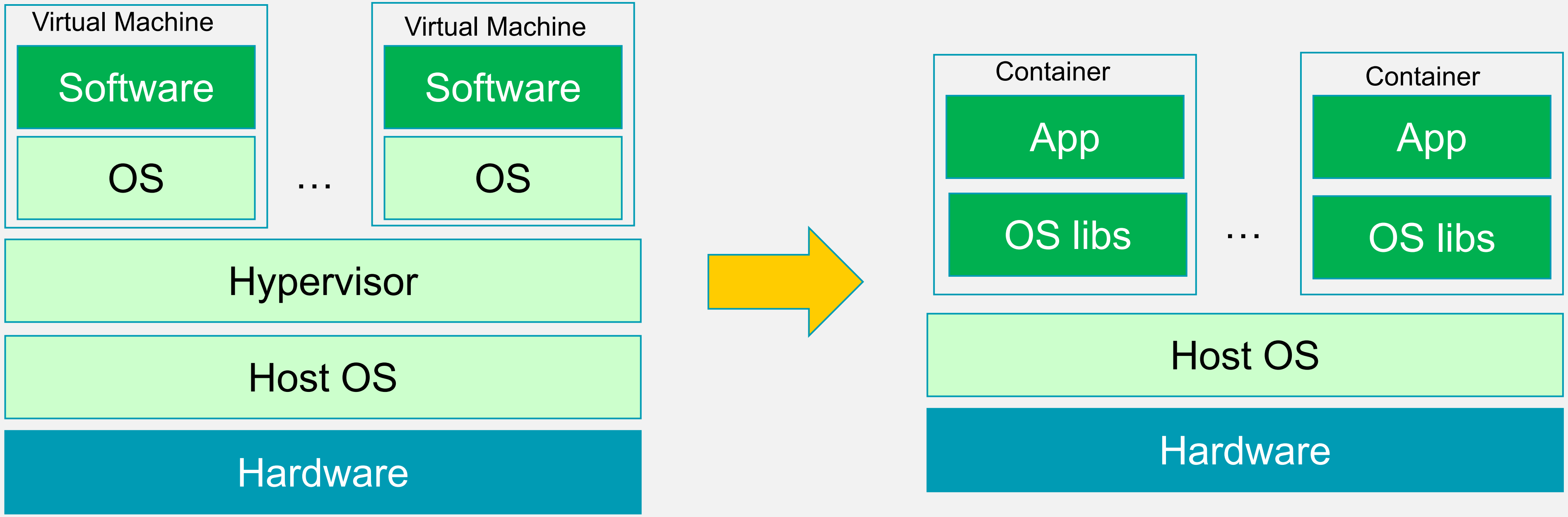
# Linux Cgroups (Isolation through boundaries) /1

A feature of the **Linux kernel** that was largely developed by Google

Grouping **processes** into communities with defined and restricted resource access to:

- Processor
- Main memory
- I/O (esp. network)
- Disk

# Operating system virtualization



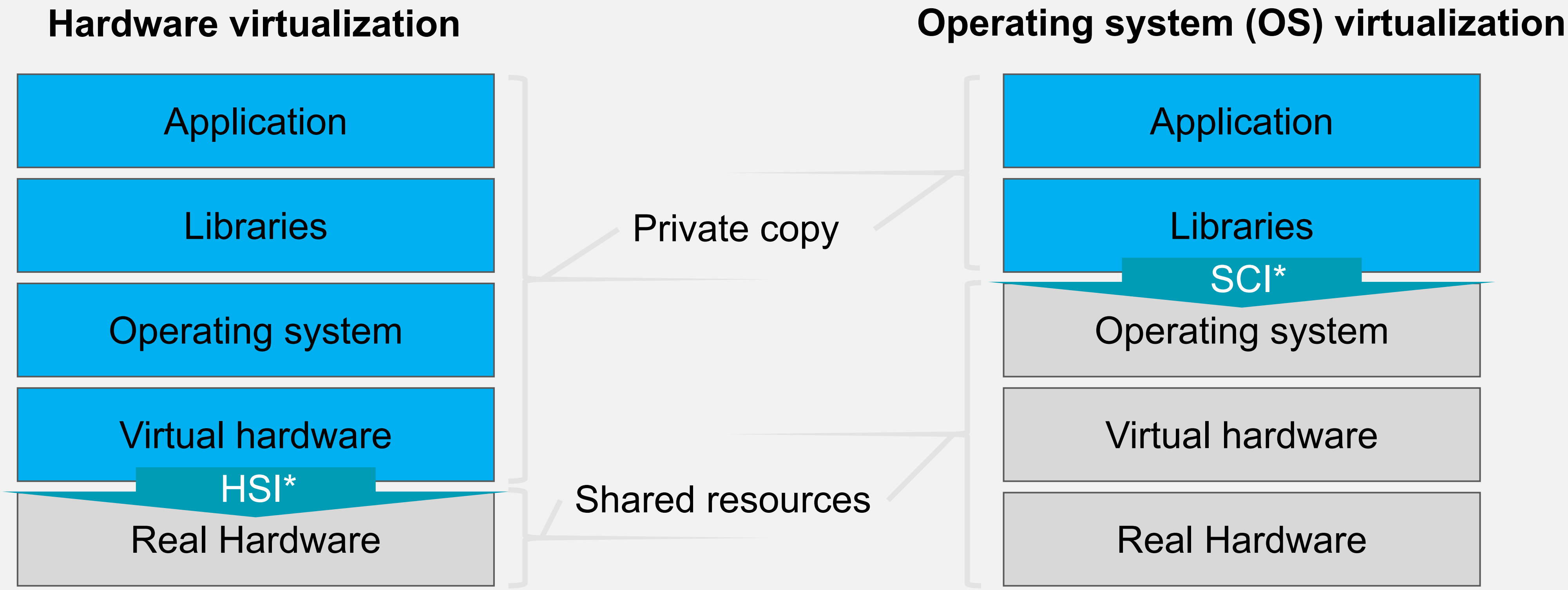
Lightweight virtualization approach: There is no hypervisor. Each app runs directly as a process in the host operating system. However, this is maximally isolated by corresponding OS mechanisms (e.g. Linux LXC).

- Isolation of the process through kernel namespaces (regarding CPU, RAM and disk I/O) and containments
- Isolated file system
- Separate network interface

CPU/RAM overhead generally not measurable (~ 0%)

Startup time = start duration for the first process

# Hardware- vs. Operating system virtualization



- Better insulation
- Higher security

- Smaller private copy volume
- Lower overhead
- Faster startup time

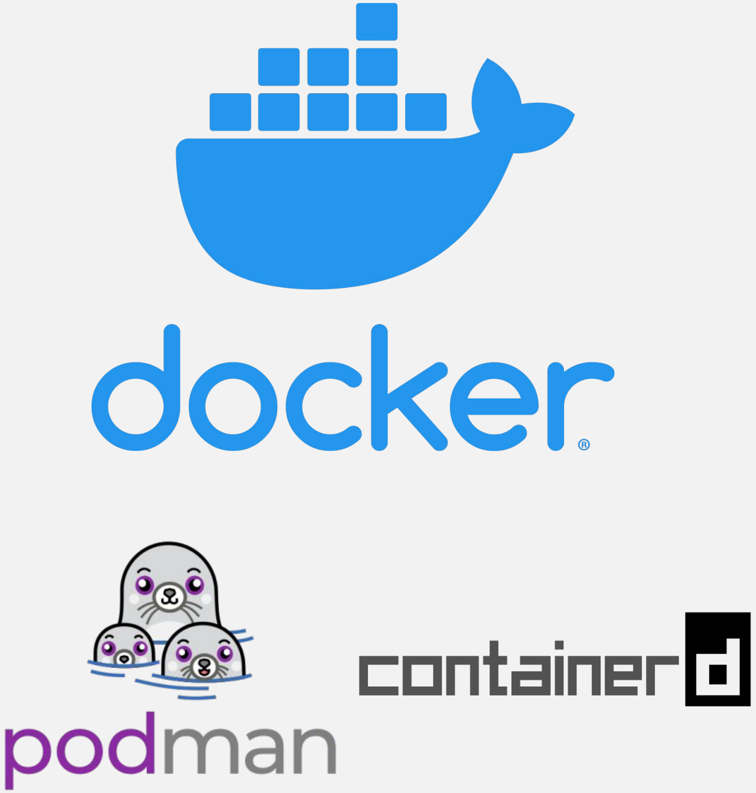
*) HSI = Hardware Software Interface
SCI = System Call Interface

# Example technologies

## Hardware virtualization
## Vagrant und VirtualBox

ORACLE
VirtualBox
VAGRANT

## OS virtualization: Docker



# OS Virtualization with Docker

# Containerization with Docker

http://www.srf.ch/kultur/im-fokus/brasilien/favelas-im-wandel-die-armen-muessen-weichen

FOR RACING USE ONLY
ON
OFF
ENGINE START
START
My App

Standard format for operations: start, stop, configure, wire, debug + software logistics.

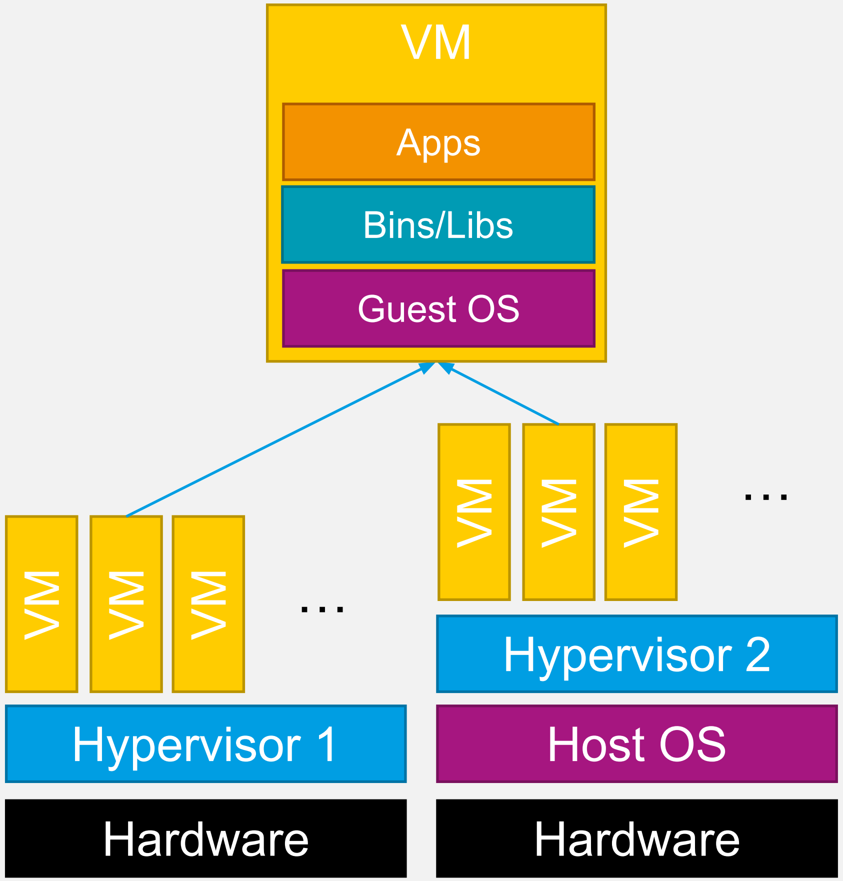
# Docker

Docker is an automation environment for operating system virtualization.

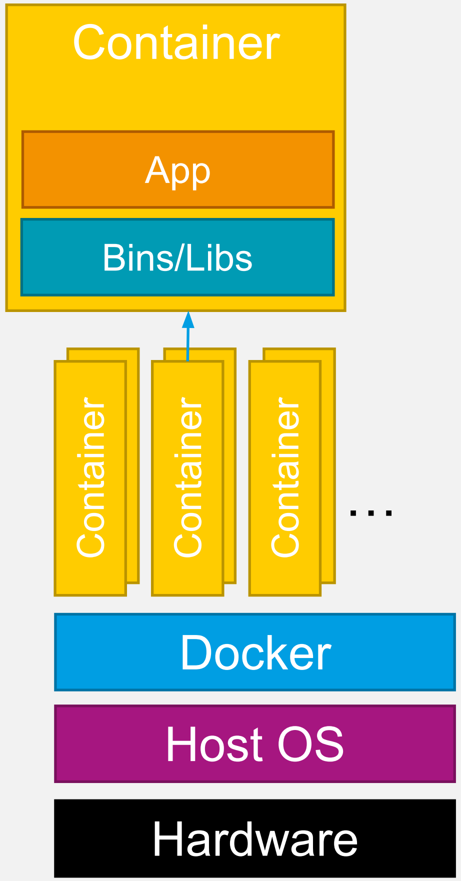
Docker currently supports Linux as a host operating system. Since Windows Server 2016, a Windows variant is now also available.

Docker was developed as a tool for a cloud provider and is now one of the most visible and active open-source ecosystems.

# Containerization with Docker

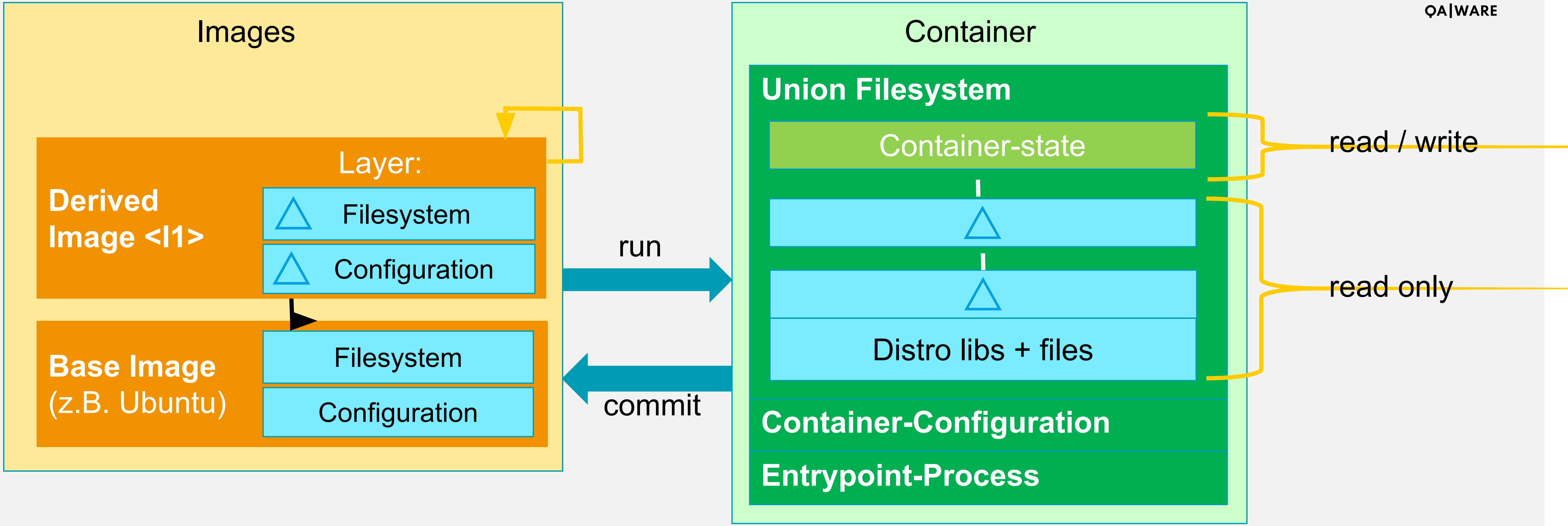


Type 1 / Type 2 Virtualization



Containerization

# At the center of Docker are images and containers.



**stationary and transportable state**

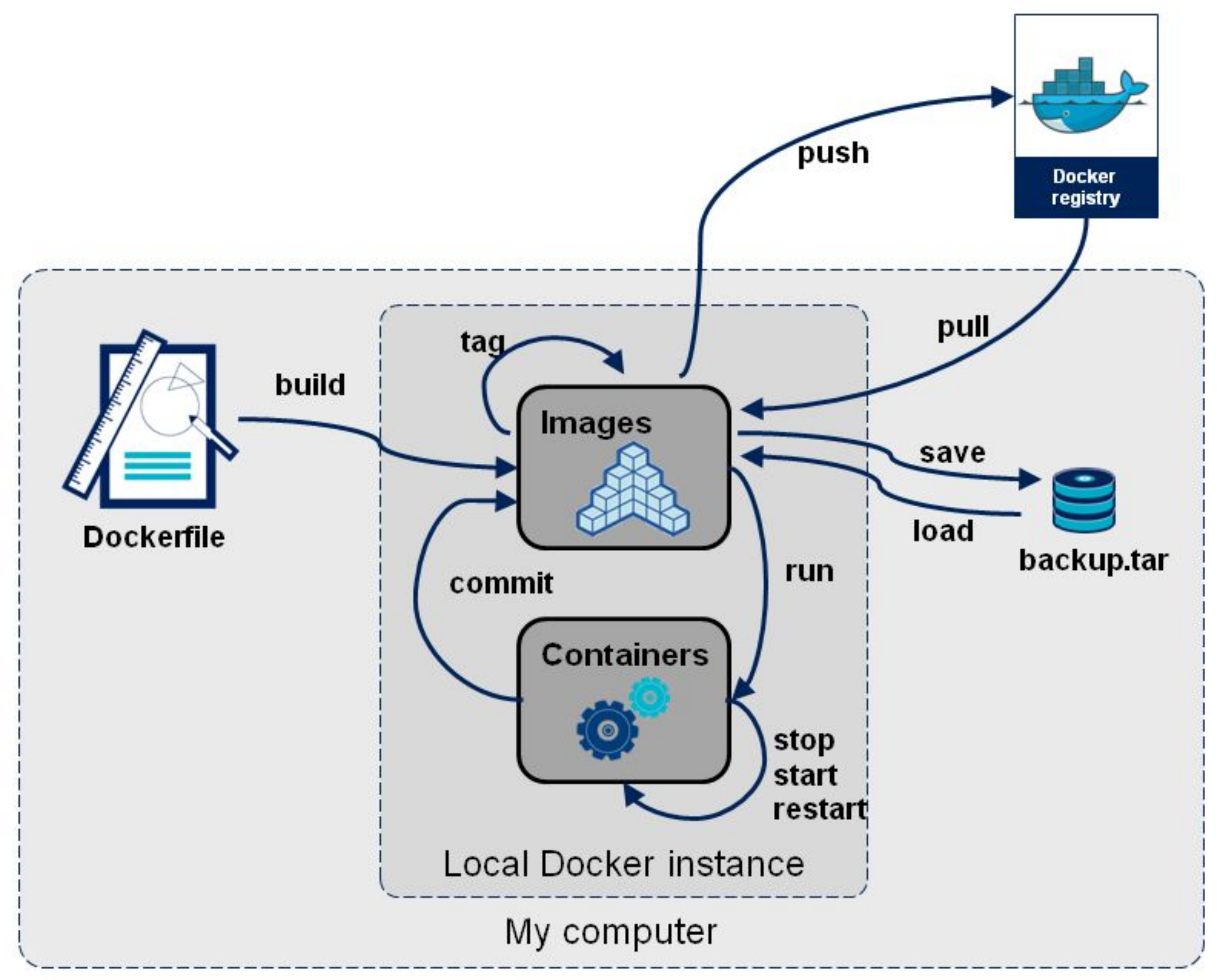
**Running state**

A container runs as long as its entrypoint process is in the foreground. Docker remembers the container state..

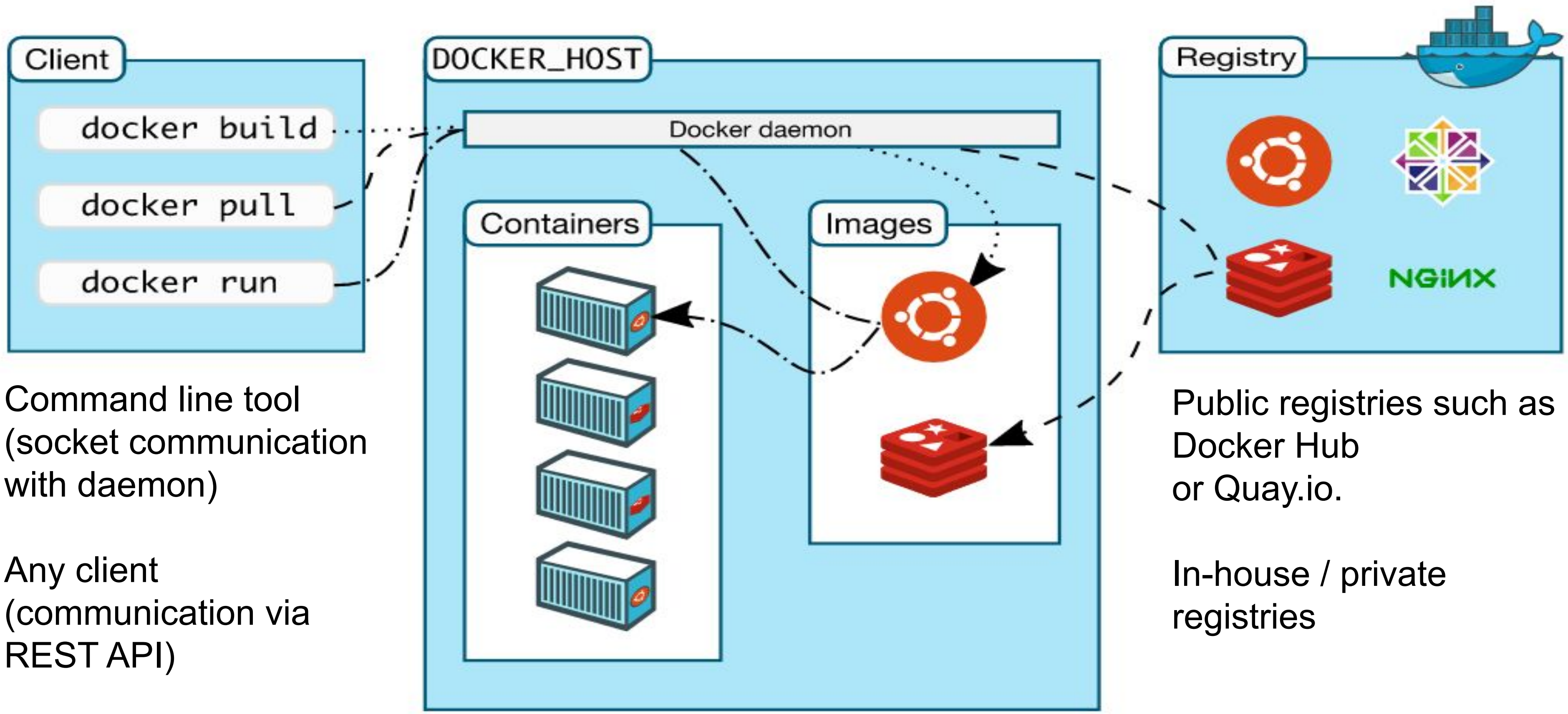
# Demo

QA|WARE

# The Docker Workflow.



# The Docker architecture.



Command line tool (socket communication with daemon)

Any client (communication via REST API)

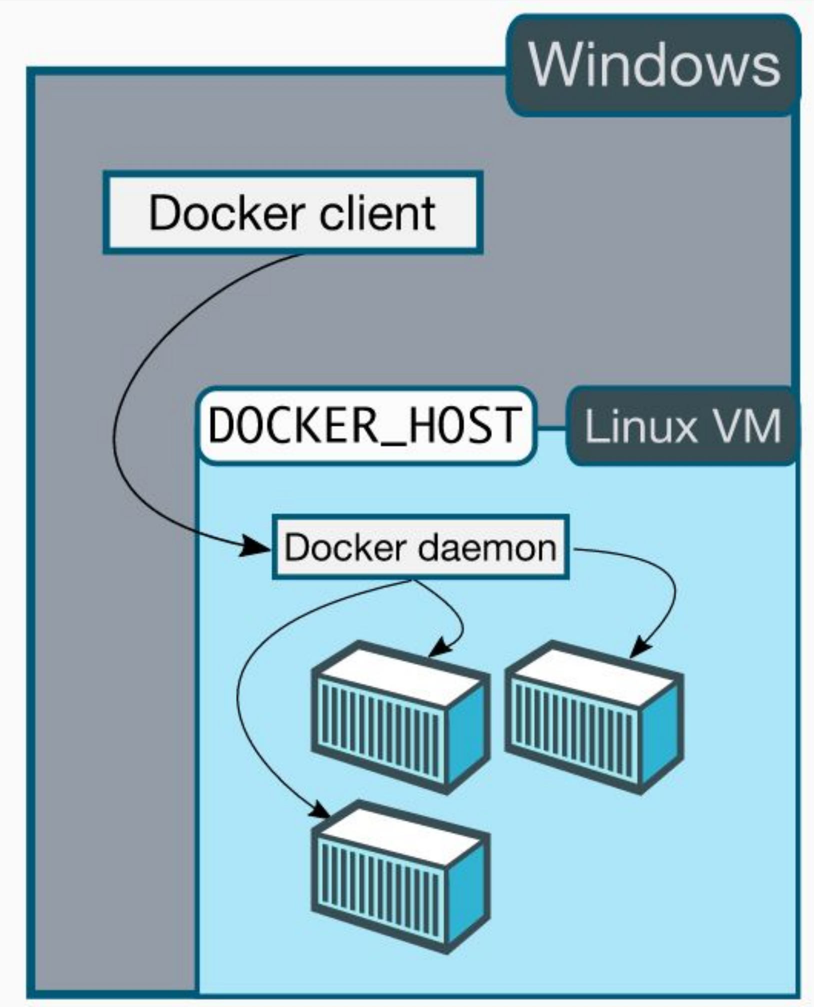
Public registries such as Docker Hub or Quay.io.

In-house / private registries

The docker daemon is the central control unit and runs directly as a process in the host operating system. It manages all local

Bildquelle: https://docs.docker.com/engine/docker-overview/

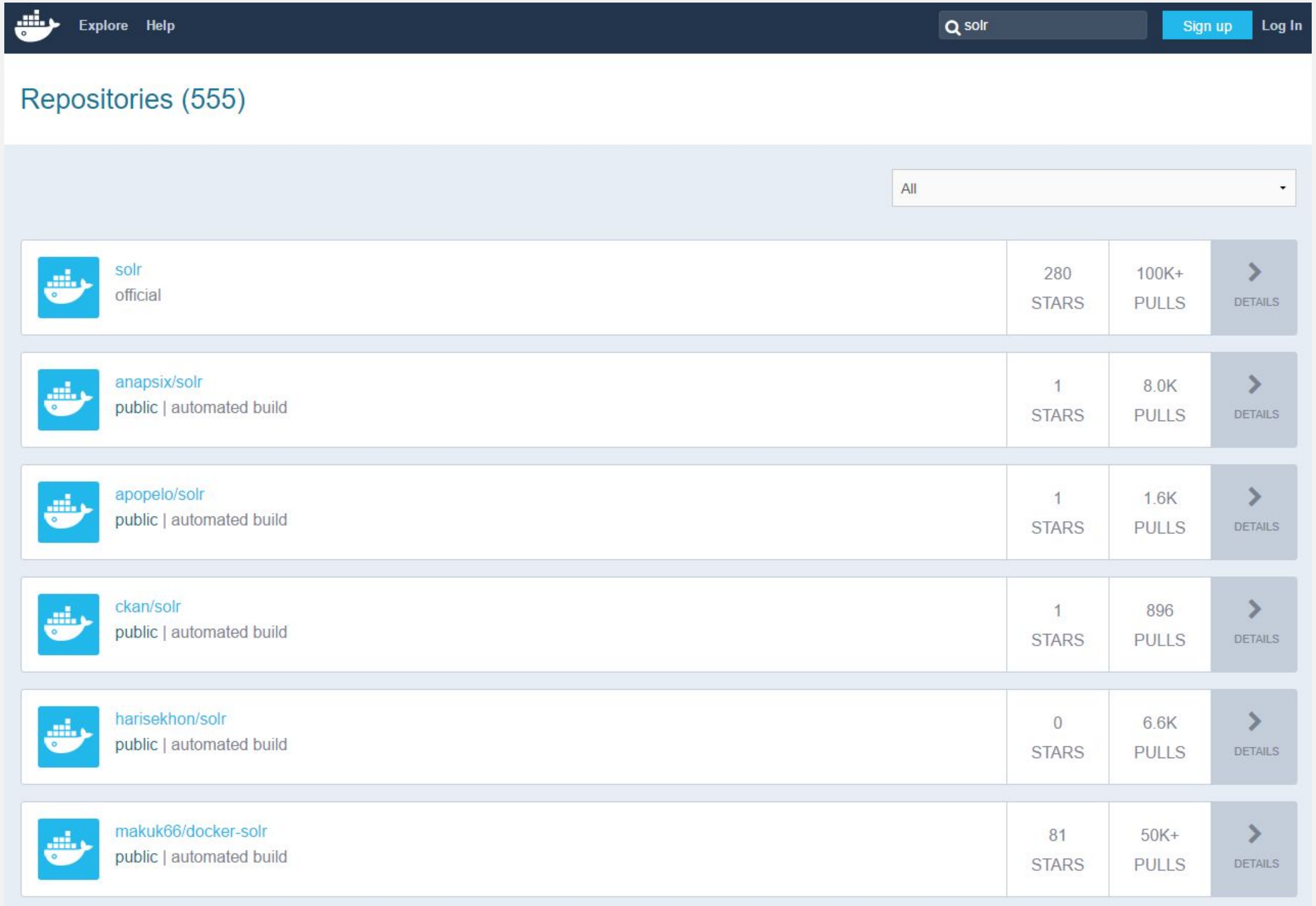
# Interaction between host and Docker machine.



Bildquelle: http://docs.docker.com/engine/installation/windows

# hub.docker.com is the standard public registry for Docker images.

QA|WARE



Repositories (555)

| | | | |
|---|---|---|---|
| solr<br>official | 280<br>STARS | 100K+<br>PULLS | DETAILS |
| anapsix/solr<br>public \| automated build | 1<br>STARS | 8.0K<br>PULLS | DETAILS |
| apopelo/solr<br>public \| automated build | 1<br>STARS | 1.6K<br>PULLS | DETAILS |
| ckan/solr<br>public \| automated build | 1<br>STARS | 896<br>PULLS | DETAILS |
| harisekhon/solr<br>public \| automated build | 0<br>STARS | 6.6K<br>PULLS | DETAILS |
| makuk66/docker-solr<br>public \| automated build | 81<br>STARS | 50K+<br>PULLS | DETAILS |

# Provisioning Images with the Dockerfile

A Dockerfile generates a new image based on another image. This automates the following actions:

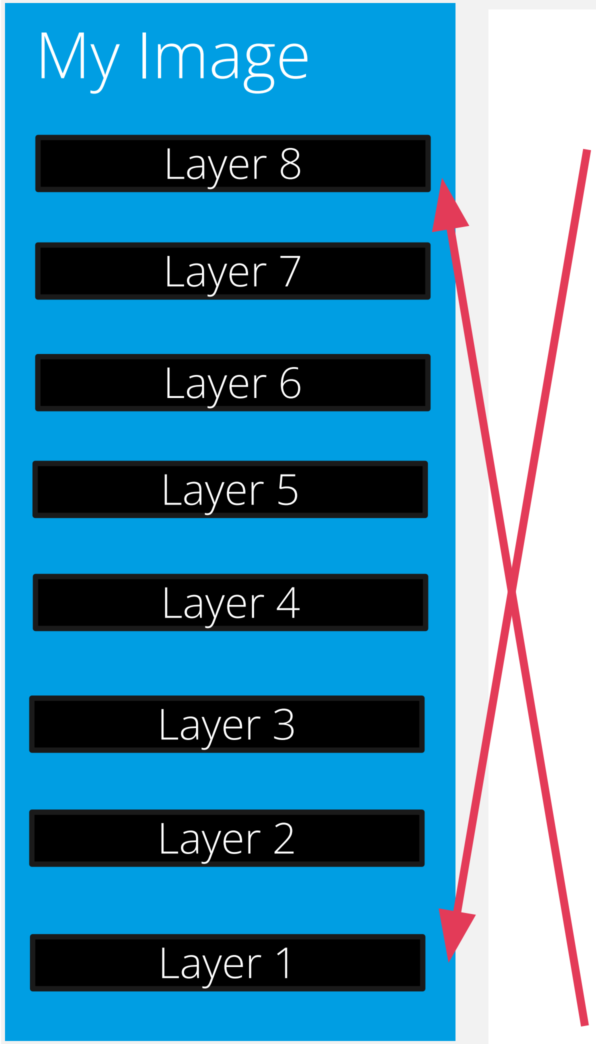
- Configuration of the image and the resulting containers
- Execution of provisioning actions

A Dockerfile is thus an image representation as an alternative to a physical image
(a building share vs. a building component).

- Repeatability in the construction of containers
- Automated creation of images without having to distribute them
- Flexibility in the configuration and in the software versions used
- Simple syntax and therefore easy to use

Command: `docker build -t <target_image_name> <Dockerfile>`

# The Dockerfile defines Structure and content of the image.



Never use latest. Antipattern

```
FROM qaware/alpine-k8s-ibmjava8:8.0-3.10
LABEL maintainer="QAware GmbH <qaware-oss@qaware.de>"

RUN mkdir -p /app

COPY build/libs/zwitscher-service-1.0.1.jar /app/zwitscher-service.jar
COPY src/main/docker/zwitscher-service.conf /app/

ENV JAVA_OPTS -Xmx256m

EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app/zwitscher-service.jar"]
```

# Dockerfile commands

| Element | Meaning |
|---|---|
| FROM <image-name> | Sets to base image (where the new image is derived from) |
| MAINTAINER <author> | Document author |
| RUN <command> | Execute a shell command and commit the result as a new image layer (!) |
| ADD <src> <dest> | Copy a file into the containers. <src> can also be an URL. If <src> refers to a TAR-file, then this file automatically gets un-tared. |
| VOLUME <container-dir> <host-dir> | Mounts a host directory into the container. |
| ENV <key> <value> | Sets an environment variable. This environment variable can be overwritten at container start with the –e command line parameter of `docker run`. |
| ENTRYPOINT <command> | The process to be started at container startup |
| CMD <command> | Parameters to the entrypoint process if no parameters are passed with `docker run` |
| WORKDIR <dir> | Sets the working dir for all following commands |
| EXPOSE <port> | Informs Docker that a container listens on a specific port and this port should be exposed to other containers |
| USER <name> | Sets the user for all container commands |

# Typical commands of a Docker Workflows

| Command | Action |
|---|---|
| `docker build -t <image> .` | Build Docker image with given tag in current directory |
| `docker images` | Prints all local images |
| `docker run`<br>`-d`<br>`-v <volume mounts>`<br>`-p <host-port>:<container-port>`<br>`<image> <entrypoint process>` | Run a Docker image: Creates and runs a container.<br>▪ in background<br>▪ with host directory mounted into the container<br>▪ with port forwarding from host to container<br>▪ image name (and optional entrypoint process) |
| `docker run`<br>`-ti`<br>`<image> /bin/sh` | Run a Docker image and open a shell within the container<br>▪ … with forwarding of local terminal<br>▪ Image name and shell (or „`/bin/bash`“) |
| `docker ps -a` | Prints all containers (without –a = only running containers) |
| `docker commit <container> qaware/foo` | Store container as local image |
| `docker kill <container>`<br>`docker rm <container>` | Terminate container (send SIGKILL to entrypoint process)<br>Remove container |
| `docker rmi -f <image>` | Remove local image |

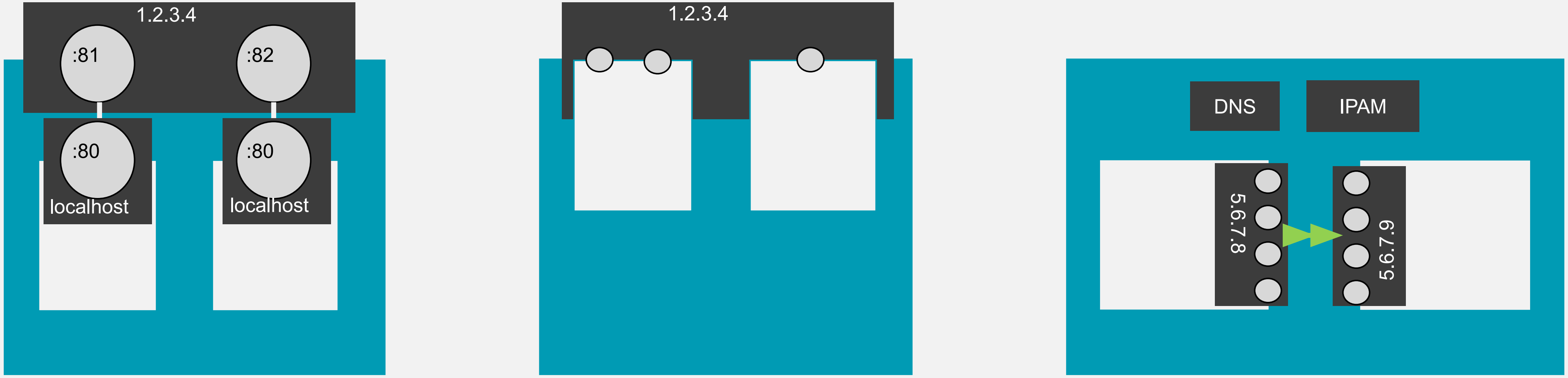
More commands: https://coderwall.com/p/2es5jw/docker-cheat-sheet-with-examples, https://docs.docker.com/reference

# Helpful Commands for Container Troubleshooting

| Command | Action |
|---|---|
| `docker inspect <container>` | Shows container metadata (e.g. IP) |
| `docker logs <container>` | Prints container syslog |
| `docker top <container>` | Prints all running processes within a container (like `ps -a` within the container) |
| `docker exec -ti <container> /bin/sh` | Connect terminal to running container |
| `docker stats <container>` | Shows container runtime statistics (e.g. CPU usage, IO intensity, …) |

Weitere Kommandos: https://coderwall.com/p/2es5jw/docker-cheat-sheet-with-examples, https://docs.docker.com/reference

# Die Docker Networking Modes.



Bridge

Host

Overlay Network

```
docker network ls
docker network inspect bridge
docker network create --driver overlay multi-host-network
docker network connect multi-host-network container1
```

○ Bound port

■ Network interface

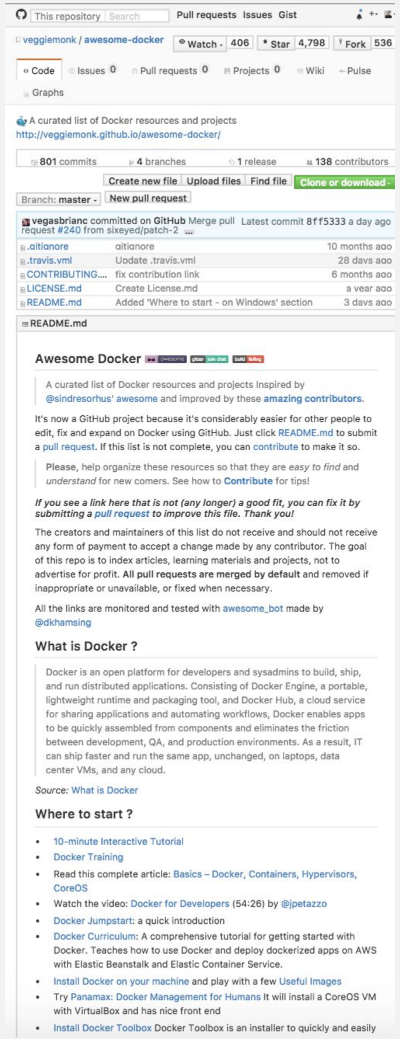
□ Guest

🟦 Host

QA|WARE

# Anhang

# https://github.com/veggiemonk/awesome-docker



A curated list of Docker resources and projects
http://veggiemonk.github.io/awesome-docker/

vegasbrianc committed on GitHub Merge pull request #240 from sixeyed/patch-2 ... Latest commit 8ff5333 a day ago

| | | |
|---|---|---|
| .aitianore | aitianore | 10 months aao |
| .travis.vml | Update .travis.vml | 28 davs aao |
| CONTRIBUTING.... | fix contribution link | 6 months aao |
| LICENSE.md | Create License.md | a vear aao |
| README.md | Added 'Where to start - on Windows' section | 3 davs aao |

README.md

## Awesome Docker

> A curated list of Docker resources and projects Inspired by @sindresorhus' awesome and improved by these **amazing contributors**.

It's now a GitHub project because it's considerably easier for other people to edit, fix and expand on Docker using GitHub. Just click README.md to submit a pull request. If this list is not complete, you can contribute to make it so.

> **Please**, help organize these resources so that they are *easy to find* and *understand* for new comers. See how to **Contribute** for tips!

***If you see a link here that is not (any longer) a good fit, you can fix it by submitting a pull request to improve this file. Thank you!***

The creators and maintainers of this list do not receive and should not receive any form of payment to accept a change made by any contributor. The goal of this repo is to index articles, learning materials and projects, not to advertise for profit. **All pull requests are merged by default** and removed if inappropriate or unavailable, or fixed when necessary.

All the links are monitored and tested with awesome_bot made by @dkhamsing

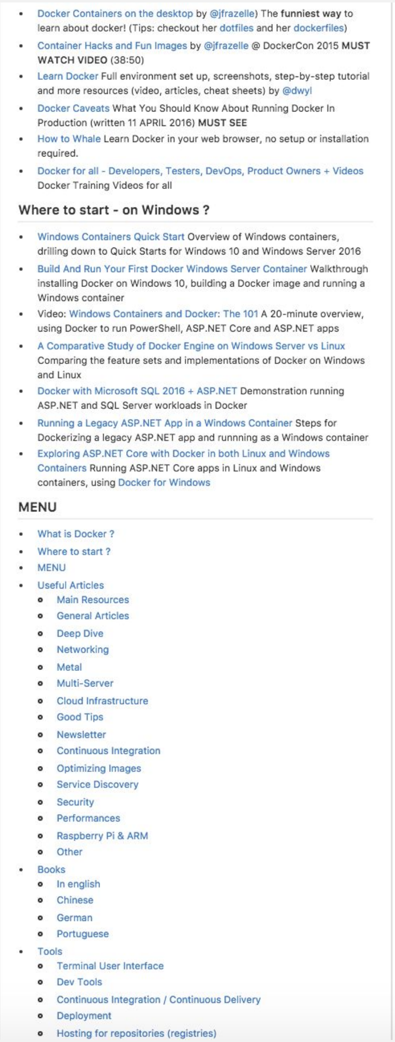
### What is Docker ?

> Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud.

*Source:* What is Docker

### Where to start ?

- 10-minute Interactive Tutorial
- Docker Training
- Read this complete article: Basics – Docker, Containers, Hypervisors, CoreOS
- Watch the video: Docker for Developers (54:26) by @jpetazzo
- Docker Jumpstart: a quick introduction
- Docker Curriculum: A comprehensive tutorial for getting started with Docker. Teaches how to use Docker and deploy dockerized apps on AWS with Elastic Beanstalk and Elastic Container Service.
- Install Docker on your machine and play with a few Useful Images
- Try Panamax: Docker Management for Humans It will install a CoreOS VM with VirtualBox and has nice front end
- Install Docker Toolbox Docker Toolbox is an installer to quickly and easily



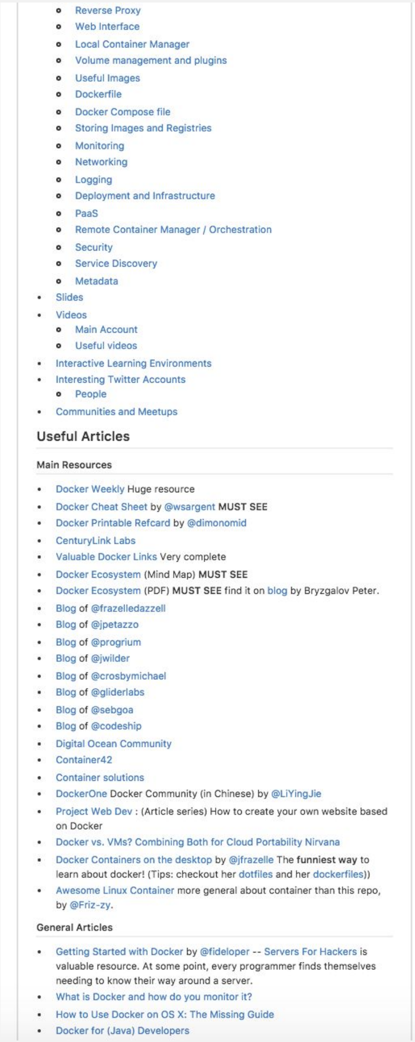
- Docker Containers on the desktop by @jfrazelle) The **funniest way** to learn about docker! (Tips: checkout her dotfiles and her dockerfiles)
- Container Hacks and Fun Images by @jfrazelle @ DockerCon 2015 **MUST WATCH VIDEO** (38:50)
- Learn Docker Full environment set up, screenshots, step-by-step tutorial and more resources (video, articles, cheat sheets) by @dwyl
- Docker Caveats What You Should Know About Running Docker In Production (written 11 APRIL 2016) **MUST SEE**
- How to Whale Learn Docker in your web browser, no setup or installation required.
- Docker for all - Developers, Testers, DevOps, Product Owners + Videos Docker Training Videos for all

### Where to start - on Windows ?

- Windows Containers Quick Start Overview of Windows containers, drilling down to Quick Starts for Windows 10 and Windows Server 2016
- Build And Run Your First Docker Windows Server Container Walkthrough installing Docker on Windows 10, building a Docker image and running a Windows container
- Video: Windows Containers and Docker: The 101 A 20-minute overview, using Docker to run PowerShell, ASP.NET Core and ASP.NET apps
- A Comparative Study of Docker Engine on Windows Server vs Linux Comparing the feature sets and implementations of Docker on Windows and Linux
- Docker with Microsoft SQL 2016 + ASP.NET Demonstration running ASP.NET and SQL Server workloads in Docker
- Running a Legacy ASP.NET App in a Windows Container Steps for Dockerizing a legacy ASP.NET app and runnning as a Windows container
- Exploring ASP.NET Core with Docker in both Linux and Windows Containers Running ASP.NET Core apps in Linux and Windows containers, using Docker for Windows

### MENU

- What is Docker ?
- Where to start ?
- MENU
- Useful Articles
  - Main Resources
  - General Articles
  - Deep Dive
  - Networking
  - Metal
  - Multi-Server
  - Cloud Infrastructure
  - Good Tips
  - Newsletter
  - Continuous Integration
  - Optimizing Images
  - Service Discovery
  - Security
  - Performances
  - Raspberry Pi & ARM
  - Other
- Books
  - In english
  - Chinese
  - German
  - Portuguese
- Tools
  - Terminal User Interface
  - Dev Tools
  - Continuous Integration / Continuous Delivery
  - Deployment
  - Hosting for repositories (registries)



  - Reverse Proxy
  - Web Interface
  - Local Container Manager
  - Volume management and plugins
  - Useful Images
  - Dockerfile
  - Docker Compose file
  - Storing Images and Registries
  - Monitoring
  - Networking
  - Logging
  - Deployment and Infrastructure
  - PaaS
  - Remote Container Manager / Orchestration
  - Security
  - Service Discovery
  - Metadata
- Slides
- Videos
  - Main Account
  - Useful videos
- Interactive Learning Environments
- Interesting Twitter Accounts
  - People
- Communities and Meetups

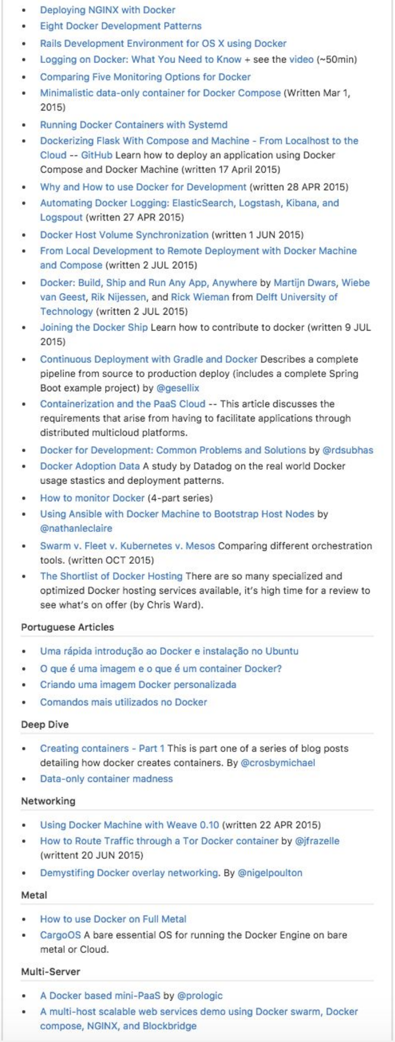
### Useful Articles

#### Main Resources

- Docker Weekly Huge resource
- Docker Cheat Sheet by @wsargent **MUST SEE**
- Docker Printable Refcard by @dimonomid
- CenturyLink Labs
- Valuable Docker Links Very complete
- Docker Ecosystem (Mind Map) **MUST SEE**
- Docker Ecosystem (PDF) **MUST SEE** find it on blog by Bryzgalov Peter.
- Blog of @frazelledazzell
- Blog of @jpetazzo
- Blog of @progrium
- Blog of @jwilder
- Blog of @crosbymichael
- Blog of @gliderlabs
- Blog of @sebgoa
- Blog of @codeship
- Digital Ocean Community
- Container42
- Container solutions
- DockerOne Docker Community (in Chinese) by @LiYingJie
- Project Web Dev : (Article series) How to create your own website based on Docker
- Docker vs. VMs? Combining Both for Cloud Portability Nirvana
- Docker Containers on the desktop by @jfrazelle The **funniest way** to learn about docker! (Tips: checkout her dotfiles and her dockerfiles))
- Awesome Linux Container more general about container than this repo, by @Friz-zy.

#### General Articles

- Getting Started with Docker by @fideloper -- Servers For Hackers is valuable resource. At some point, every programmer finds themselves needing to know their way around a server.
- What is Docker and how do you monitor it?
- How to Use Docker on OS X: The Missing Guide
- Docker for (Java) Developers



- Deploying NGINX with Docker
- Eight Docker Development Patterns
- Rails Development Environment for OS X using Docker
- Logging on Docker: What You Need to Know + see the video (~50min)
- Comparing Five Monitoring Options for Docker
- Minimalistic data-only container for Docker Compose (Written Mar 1, 2015)
- Running Docker Containers with Systemd
- Dockerizing Flask With Compose and Machine - From Localhost to the Cloud -- GitHub Learn how to deploy an application using Docker Compose and Docker Machine (written 17 April 2015)
- Why and How to use Docker for Development (written 28 APR 2015)
- Automating Docker Logging: ElasticSearch, Logstash, Kibana, and Logspout (written 27 APR 2015)
- Docker Host Volume Synchronization (written 1 JUN 2015)
- From Local Development to Remote Deployment with Docker Machine and Compose (written 2 JUL 2015)
- Docker: Build, Ship and Run Any App, Anywhere by Martijn Dwars, Wiebe van Geest, Rik Nijessen, and Rick Wieman from Delft University of Technology (written 2 JUL 2015)
- Joining the Docker Ship Learn how to contribute to docker (written 9 JUL 2015)
- Continuous Deployment with Gradle and Docker Describes a complete pipeline from source to production deploy (includes a complete Spring Boot example project) by @gesellix
- Containerization and the PaaS Cloud -- This article discusses the requirements that arise from having to facilitate applications through distributed multicloud platforms.
- Docker for Development: Common Problems and Solutions by @rdsubhas
- Docker Adoption Data A study by Datadog on the real world Docker usage stastics and deployment patterns.
- How to monitor Docker (4-part series)
- Using Ansible with Docker Machine to Bootstrap Host Nodes by @nathanleclaire
- Swarm v. Fleet v. Kubernetes v. Mesos Comparing different orchestration tools. (written OCT 2015)
- The Shortlist of Docker Hosting There are so many specialized and optimized Docker hosting services available, it's high time for a review to see what's on offer (by Chris Ward).

#### Portuguese Articles

- Uma rápida introdução ao Docker e instalação no Ubuntu
- O que é uma imagem e o que é um container Docker?
- Criando uma imagem Docker personalizada
- Comandos mais utilizados no Docker

#### Deep Dive

- Creating containers - Part 1 This is part one of a series of blog posts detailing how docker creates containers. By @crosbymichael
- Data-only container madness

#### Networking

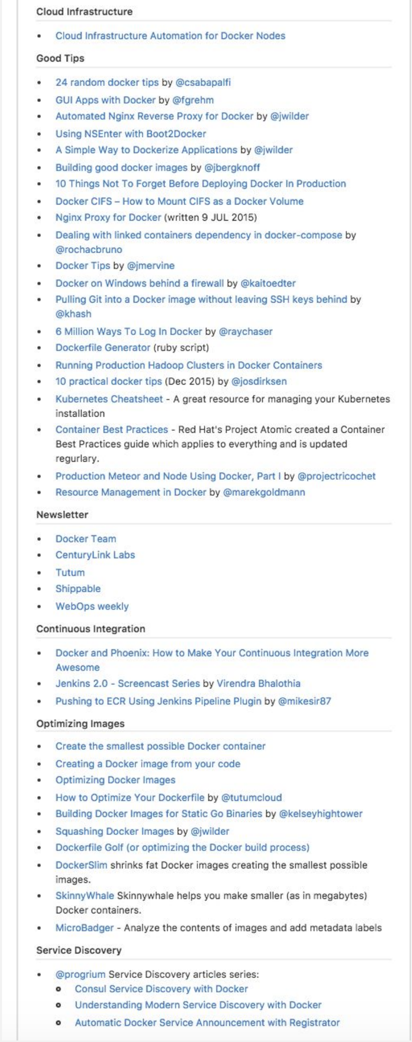
- Using Docker Machine with Weave 0.10 (written 22 APR 2015)
- How to Route Traffic through a Tor Docker container by @jfrazelle (writtent 20 JUN 2015)
- Demystifing Docker overlay networking. By @nigelpoulton

#### Metal

- How to use Docker on Full Metal
- CargoOS A bare essential OS for running the Docker Engine on bare metal or Cloud.

#### Multi-Server

- A Docker based mini-PaaS by @prologic
- A multi-host scalable web services demo using Docker swarm, Docker compose, NGINX, and Blockbridge



#### Cloud Infrastructure

- Cloud Infrastructure Automation for Docker Nodes

#### Good Tips

- 24 random docker tips by @csabapalfi
- GUI Apps with Docker by @fgrehm
- Automated Nginx Reverse Proxy for Docker by @jwilder
- Using NSEnter with Boot2Docker
- A Simple Way to Dockerize Applications by @jwilder
- Building good docker images by @jbergknoff
- 10 Things Not To Forget Before Deploying Docker In Production
- Docker CIFS – How to Mount CIFS as a Docker Volume
- Nginx Proxy for Docker (written 9 JUL 2015)
- Dealing with linked containers dependency in docker-compose by @rochacbruno
- Docker Tips by @jmervine
- Docker on Windows behind a firewall by @kaitoedter
- Pulling Git into a Docker image without leaving SSH keys behind by @khash
- 6 Million Ways To Log In Docker by @raychaser
- Dockerfile Generator (ruby script)
- Running Production Hadoop Clusters in Docker Containers
- 10 practical docker tips (Dec 2015) by @josdirksen
- Kubernetes Cheatsheet - A great resource for managing your Kubernetes installation
- Container Best Practices - Red Hat's Project Atomic created a Container Best Practices guide which applies to everything and is updated regurlary.
- Production Meteor and Node Using Docker, Part I by @projectricochet
- Resource Management in Docker by @marekgoldmann

#### Newsletter

- Docker Team
- CenturyLink Labs
- Tutum
- Shippable
- WebOps weekly

#### Continuous Integration

- Docker and Phoenix: How to Make Your Continuous Integration More Awesome
- Jenkins 2.0 - Screencast Series by Virendra Bhalothia
- Pushing to ECR Using Jenkins Pipeline Plugin by @mikesir87

#### Optimizing Images

- Create the smallest possible Docker container
- Creating a Docker image from your code
- Optimizing Docker Images
- How to Optimize Your Dockerfile by @tutumcloud
- Building Docker Images for Static Go Binaries by @kelseyhightower
- Squashing Docker Images by @jwilder
- Dockerfile Golf (or optimizing the Docker build process)
- DockerSlim shrinks fat Docker images creating the smallest possible images.
- SkinnyWhale Skinnywhale helps you make smaller (as in megabytes) Docker containers.
- MicroBadger - Analyze the contents of images and add metadata labels

#### Service Discovery

- @progrium Service Discovery articles series:
  - Consul Service Discovery with Docker
  - Understanding Modern Service Discovery with Docker
  - Automatic Docker Service Announcement with Registrator

...