qaware.de

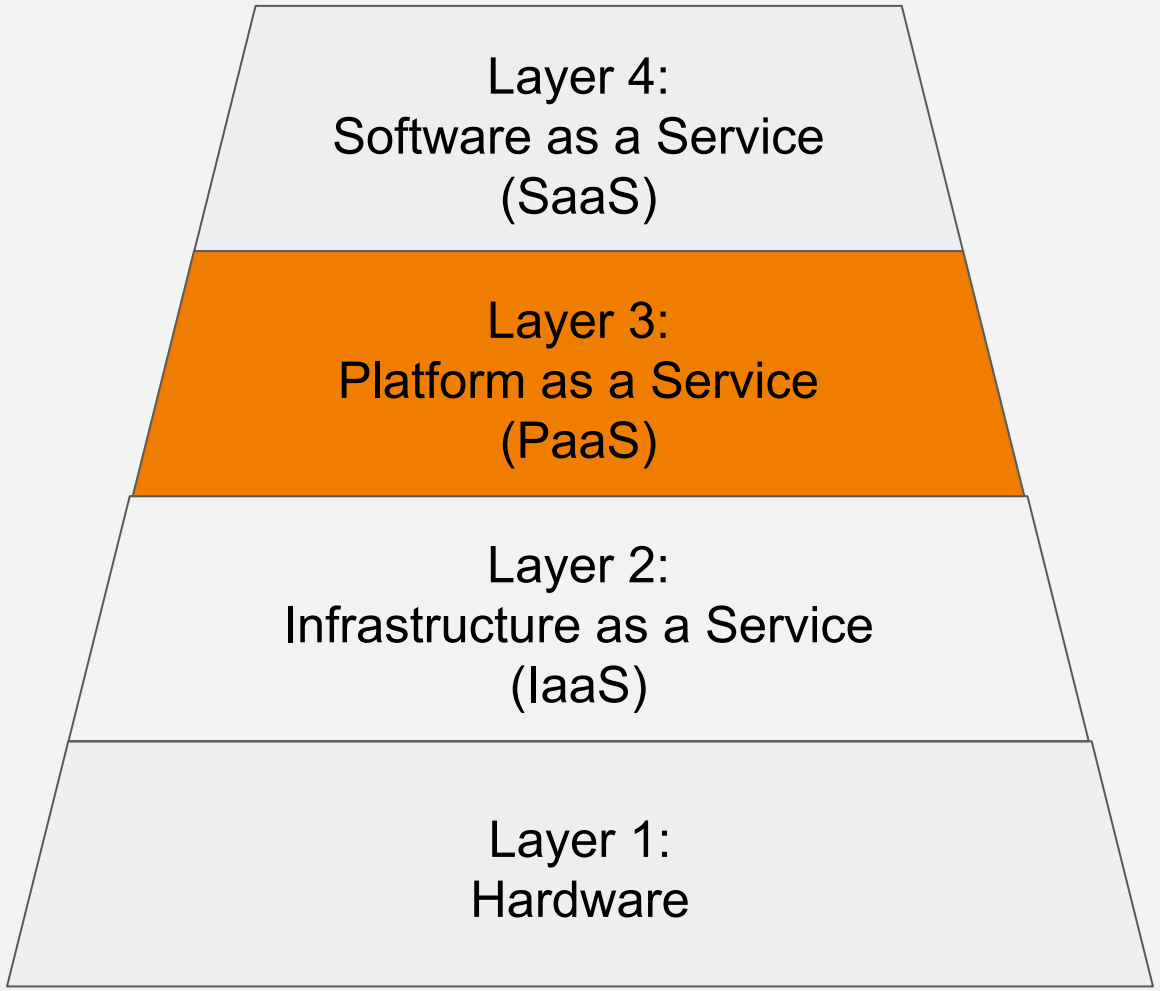
QA|WARE
SOFTWARE ENGINEERING

Platform-as-a-Service (PaaS) & Internal Developer Platforms

Lukas Buchner
lukas.buchner@qaware.de

PaaS-Cloud Basics

The layered model of cloud computing: From metal to application.



- Customizable software services
- CaaS: Component as a service (e.g. Google Charts)
- BaaS: Backend as a Service
- Transparent Updates

Target Group: Users

- Platform services
- Application programming interfaces (APIs)
- Abstraction of technical infrastructure

Target group: Developers

- Elasticity
- Virtual resourcepools
- Technical infrastructure: Machines, Servers (DNS, DHCP, LB, NAS, …)

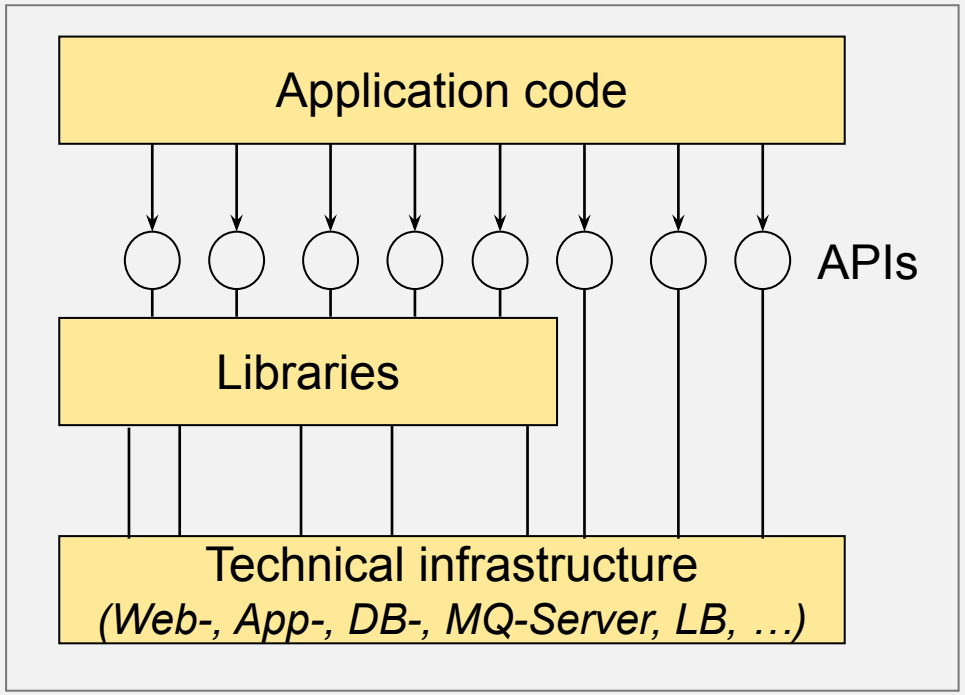
Target group: Operations

- Computers
- Network
- Storage

https://www.youtube.com/watch?v=M988_fsOSWo

The Problem: Stovepipe Architecture.
Integration is time consuming.

The System: connected with great manual efforts

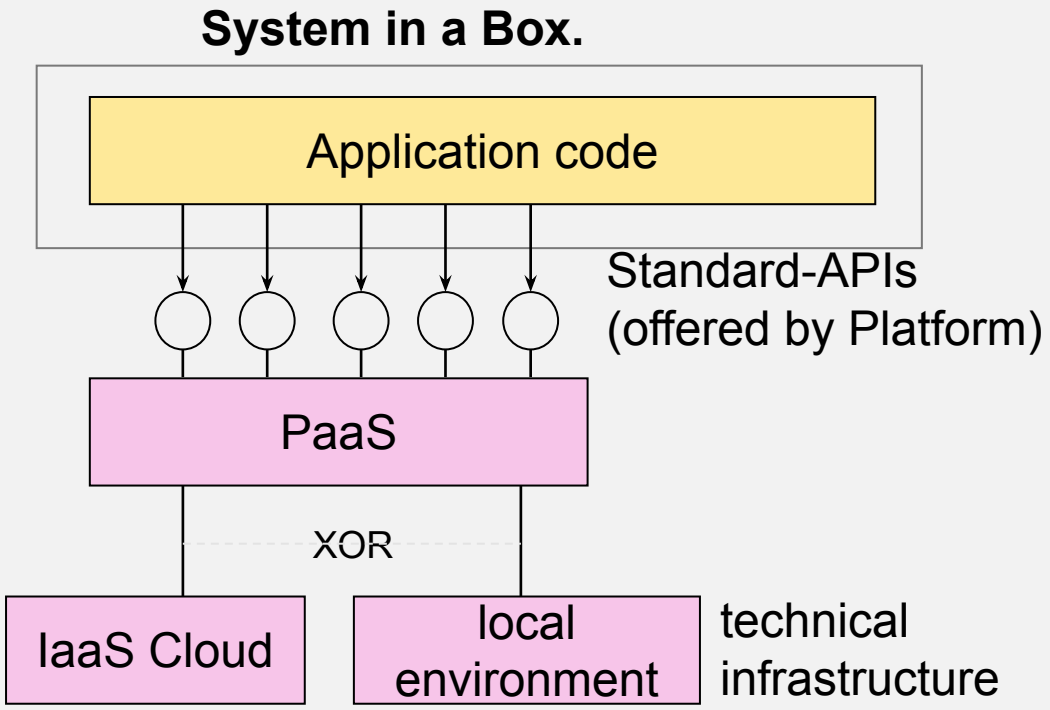


Idea: Platform-as-a-Service offers an ad-hoc Development- and Operations Platform.

The application is deployed either as an application package or as source code. No image with technical infrastructure is required.

The application only interacts with programming or access interfaces of its runtime environment.

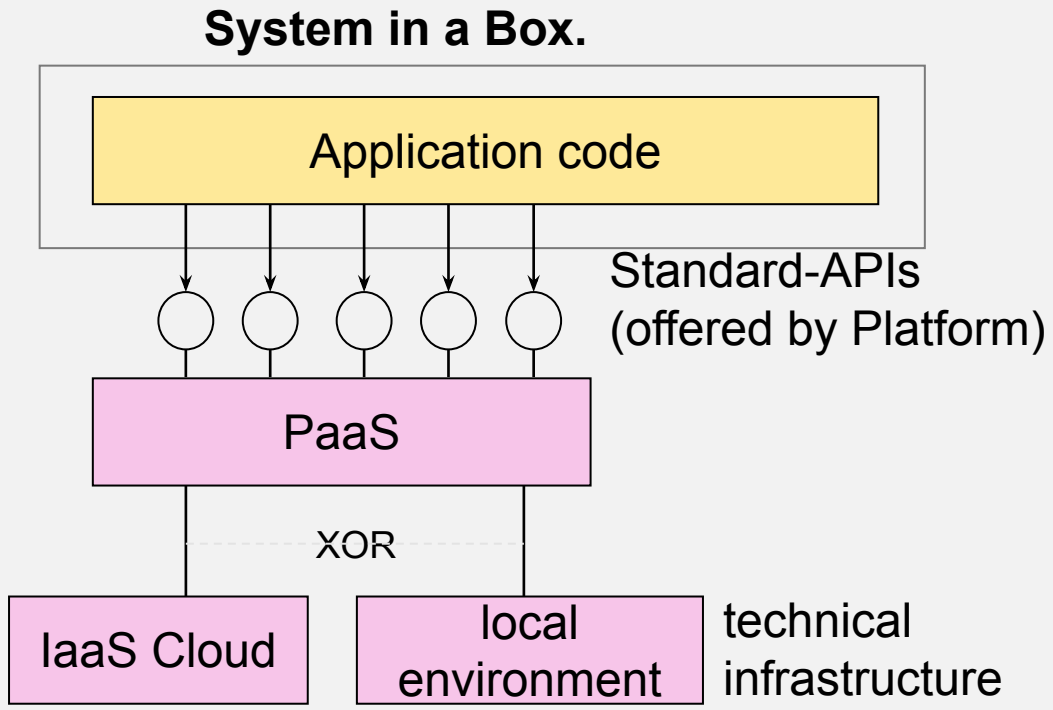
“Engine and operating system should not matter…”
The application is automatically scaled.



Idea: Platform-as-a-Service offers an ad-hoc Development- and Operations Platform.

Development tools (especially plugins for IDEs and build systems, as well as a local testing environment) are available:
“deploy to cloud.”

The platform provides an interface for administration and monitoring of applications.

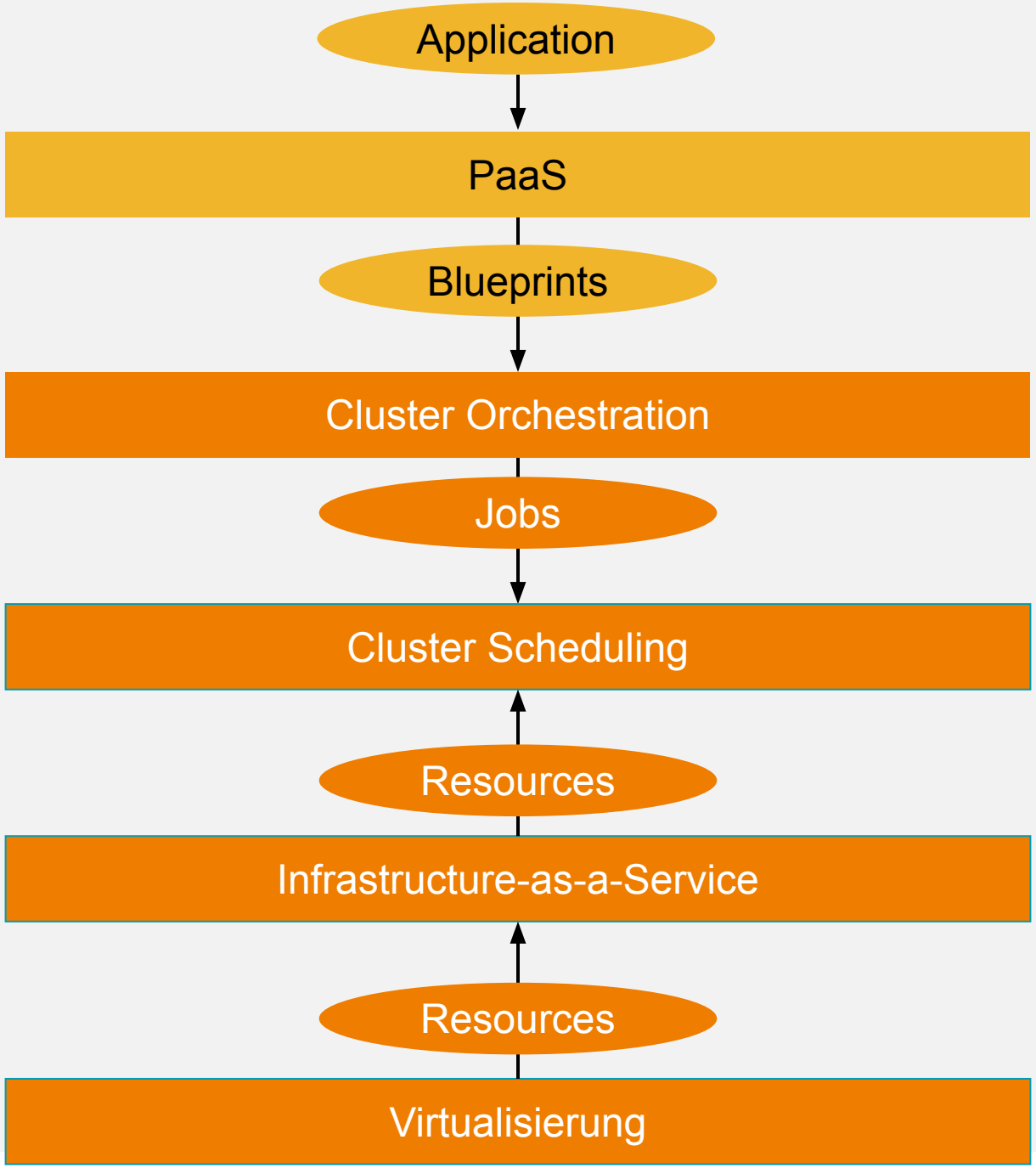


PaaS: Definitions

NIST: The capability provided to the consumer is to **deploy onto the cloud infrastructure** consumer-created or acquired applications created **using programming languages, libraries, services, and tools supported by the provider**. The **consumer does not manage or control the underlying cloud infrastructure** including network, servers, operating systems, or storage, but **has control over the deployed applications** and possibly configuration settings for the application-hosting environment.

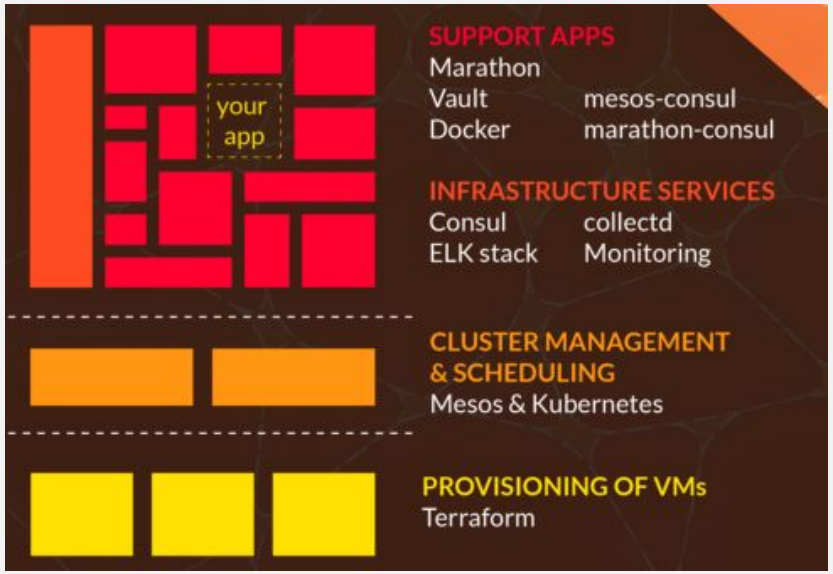
Forrester: A **complete application platform** for multitenant cloud environments that **includes development tools, runtime, and administration** and management tools and services. PaaS **combines an application platform with managed cloud infrastructure** services.

The Big Picture



QA|WARE

Building-Blocks of PaaS-Solutions: Do you see similarities?



Quelle: https://mantl.io

Quelle: https://github.com/yelp/paasta

Note: PaaSTA is an opinionated platform that uses a few un-opinionated tools. It requires a non-trivial amount of infrastructure to be in place before it works completely:

- Docker for code delivery and containment
- Mesos for code execution and scheduling (runs Docker containers)
- Marathon for managing long-running services
- Chronos for running things on a timer (nightly batches)
- SmartStack for service registration and discovery
- Sensu for monitoring/alerting
- Jenkins (optionally) for continuous deployment

Quelle: https://github.com/Capgemini/Apollo

Apollo is built on top of the following components:

- Packer for automating the build of the base images
- Terraform for provisioning the infrastructure
- Apache Mesos for cluster management, scheduling and resource isolation
- Consul for service discovery, DNS
- Docker for application container runtimes
- Weave for networking of docker containers
- HAProxy for application container load balancing

Cloud-ready Software Architecture

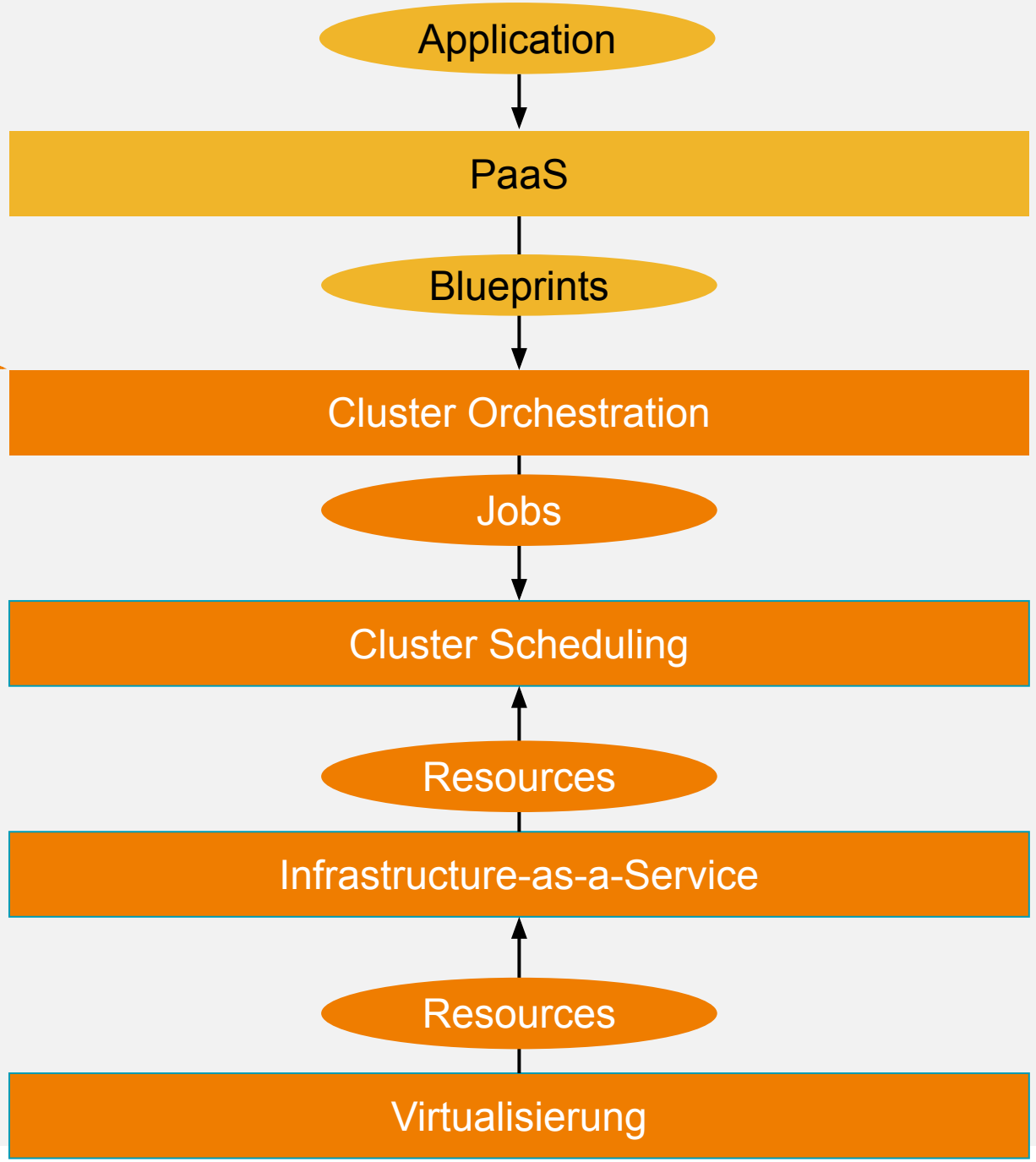
Cluster Orchestration

Cluster Scheduling

The Big Picture

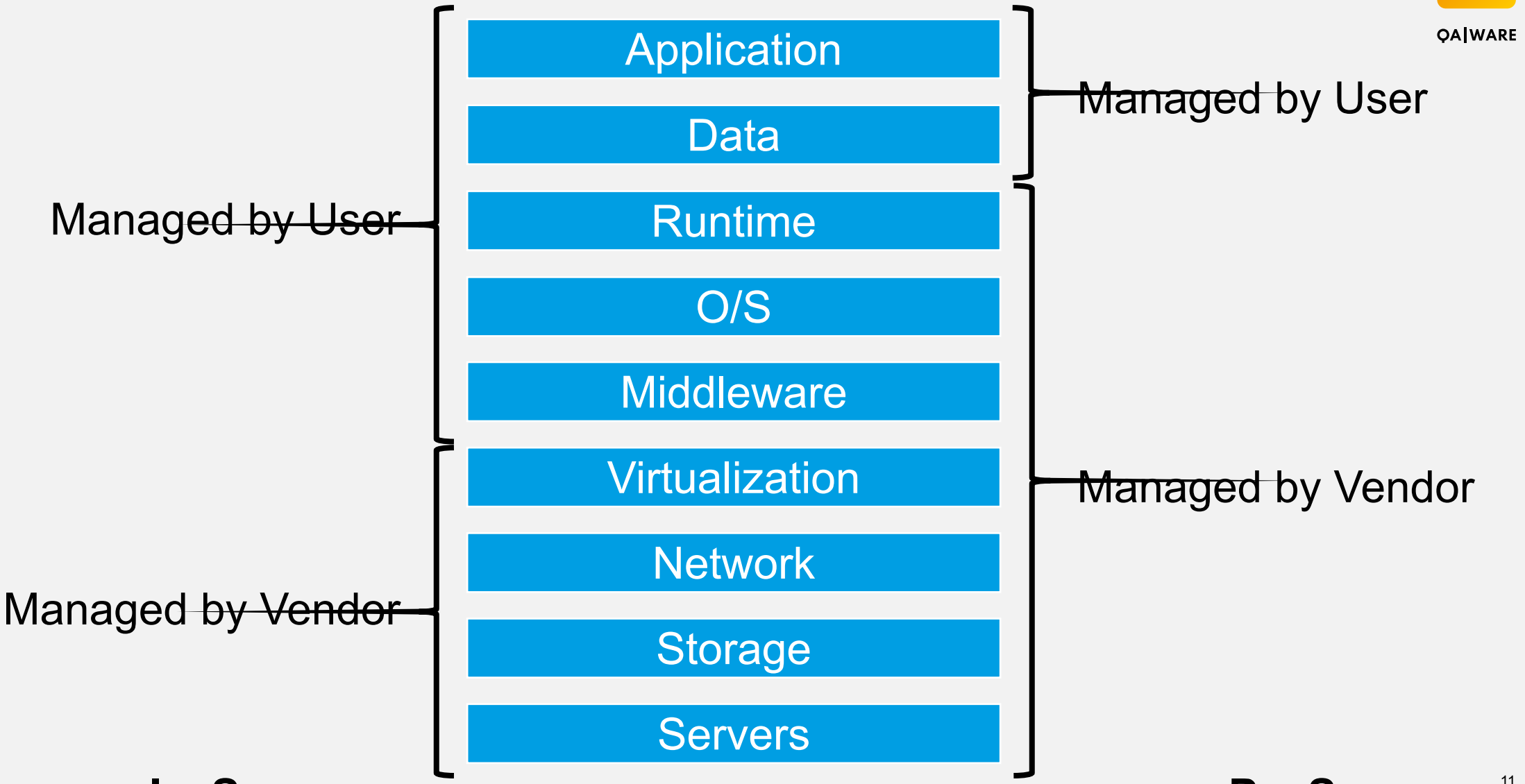
This is already 80% of a PaaS.
What's still missing:

- Reuse of infrastructure/APIs
- Convenience services for developers

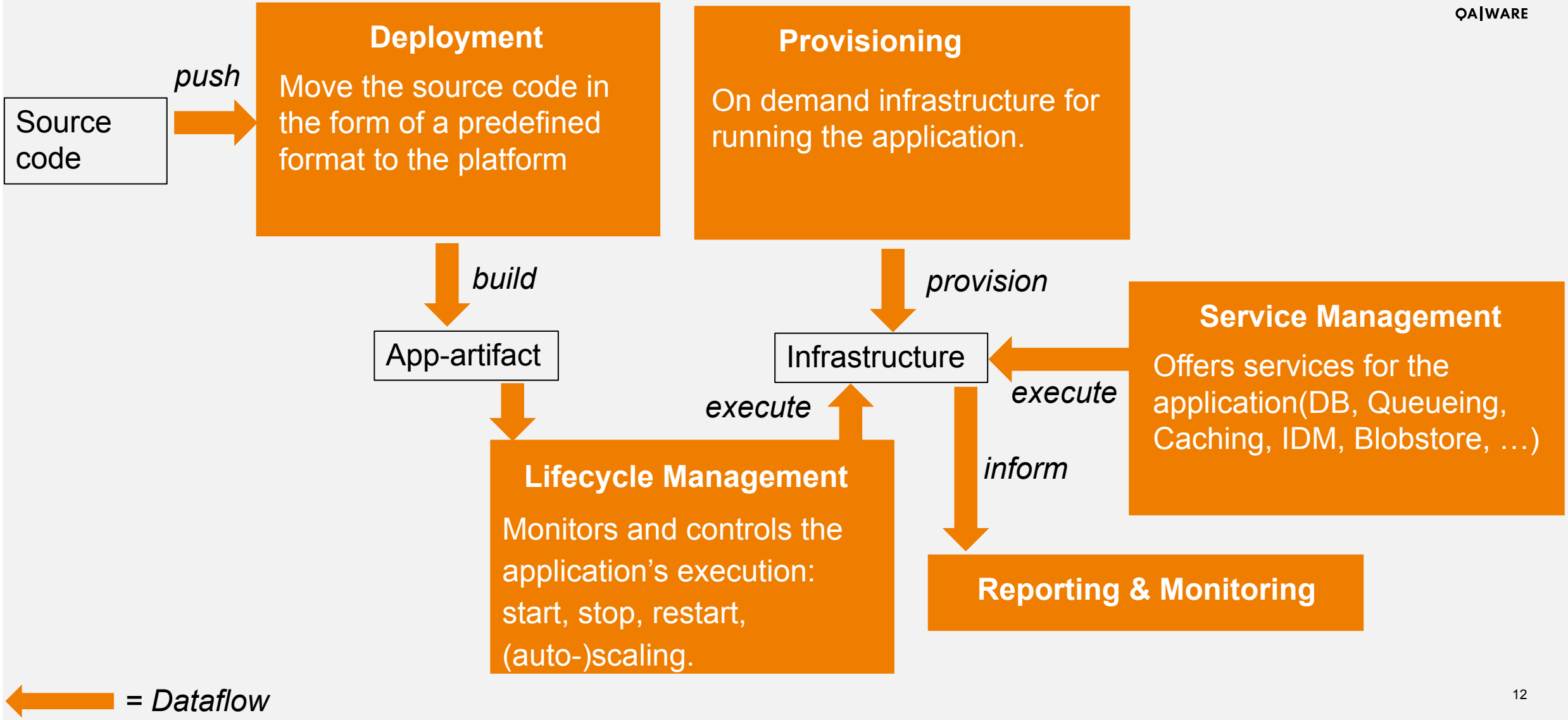


QA|WARE

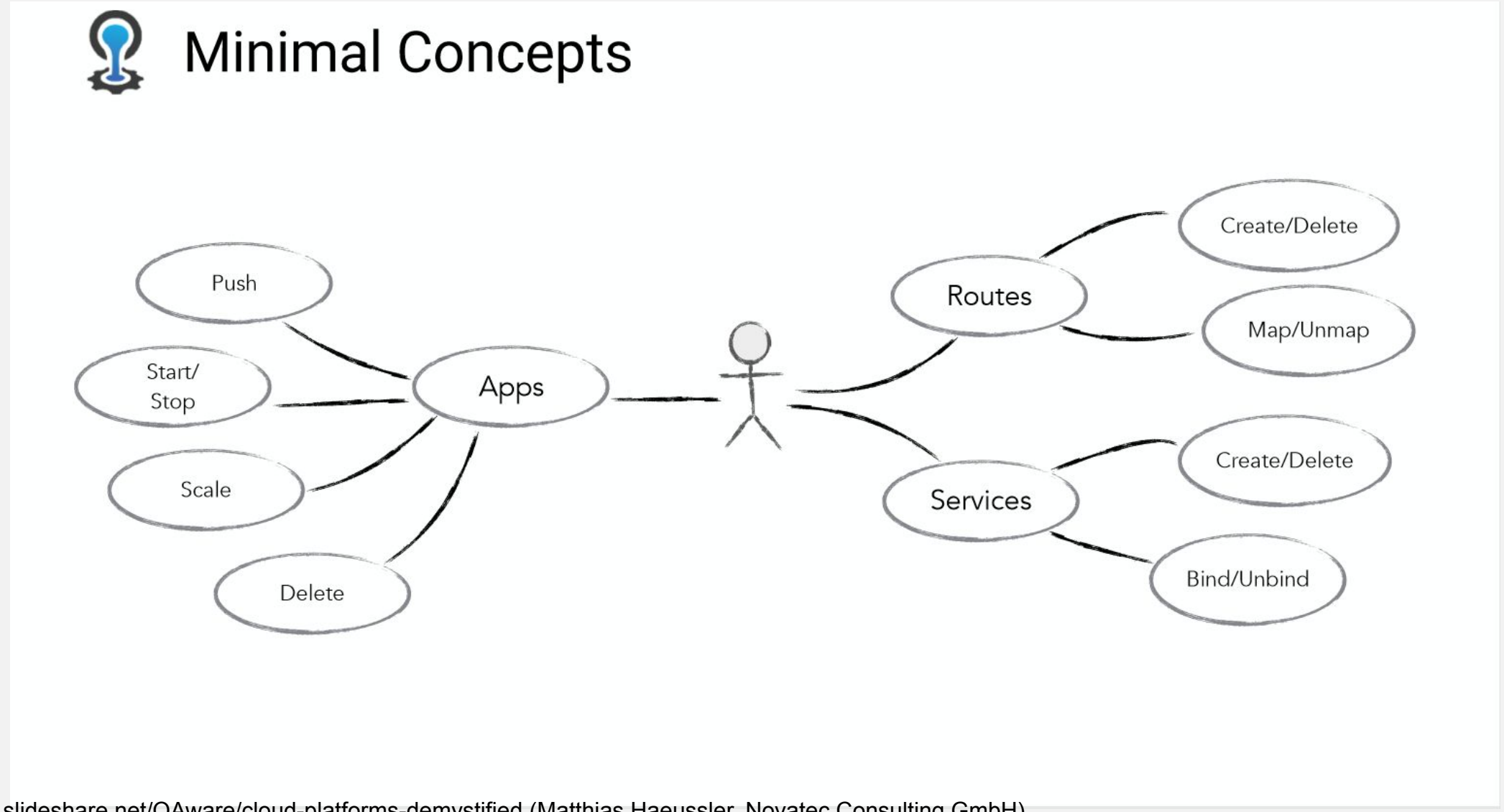
IaaS vs. PaaS


Application
Data
Runtime
O/S
Middleware
Virtualization
Network
Storage
Servers
Managed by User
Managed by Vendor
Managed by User
Managed by Vendor
IaaS
PaaS

The functional building blocks of a PaaS Cloud.

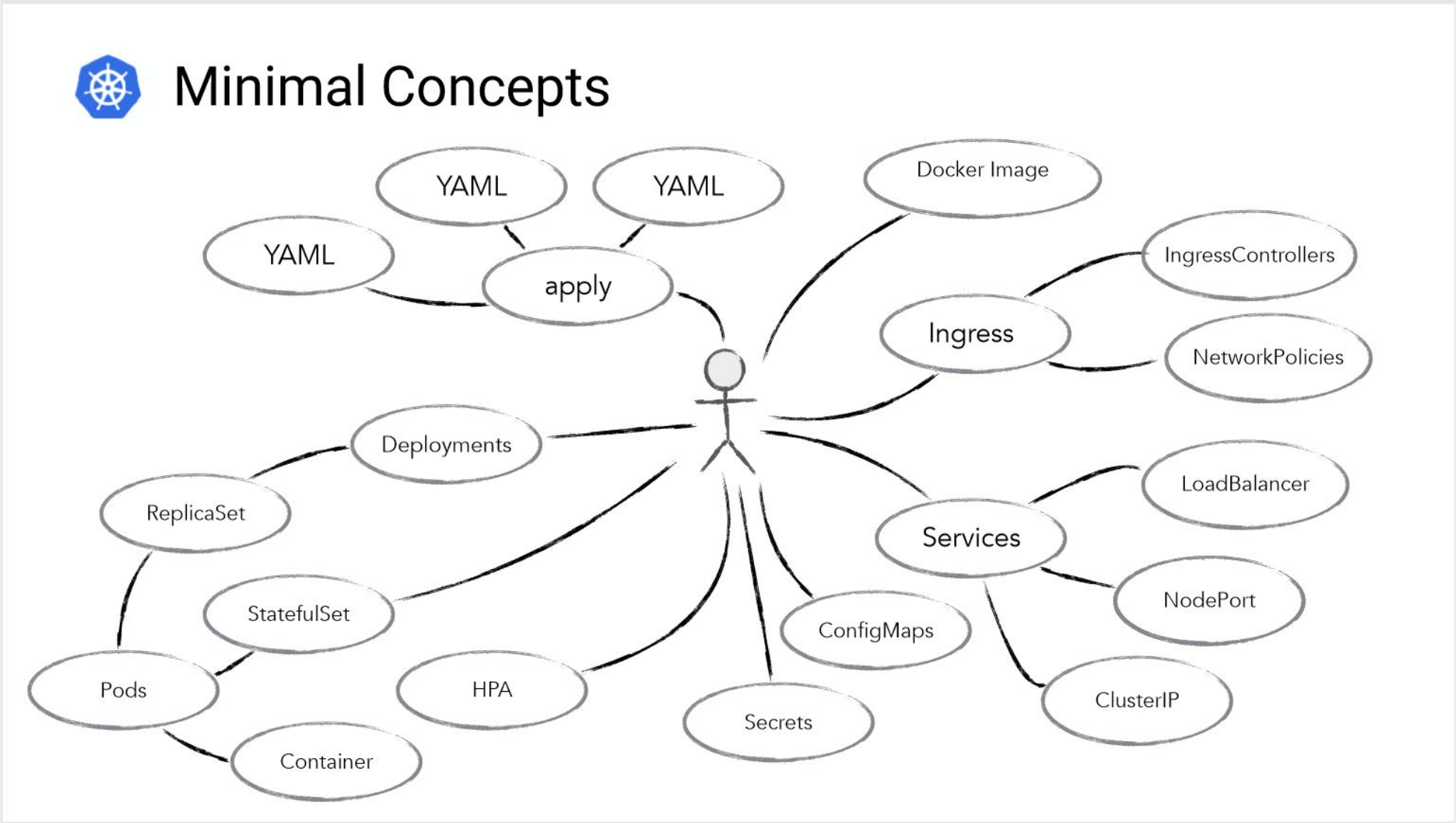


Example: Cloud Foundry



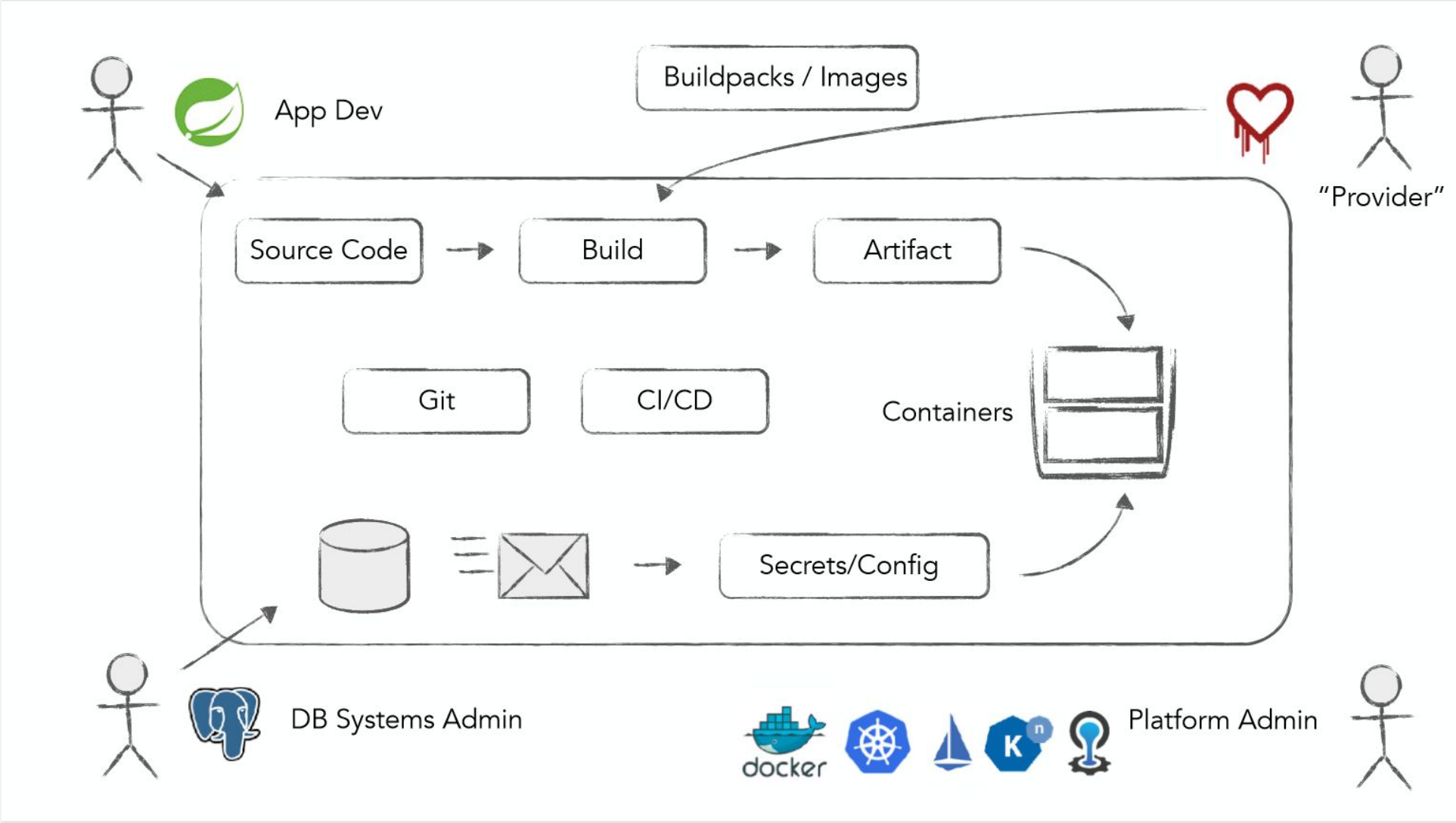
https://www.slideshare.net/QAware/cloud-platforms-demystified (Matthias Haeussler, Novatec Consulting GmbH)

In comparison: Kubernetes



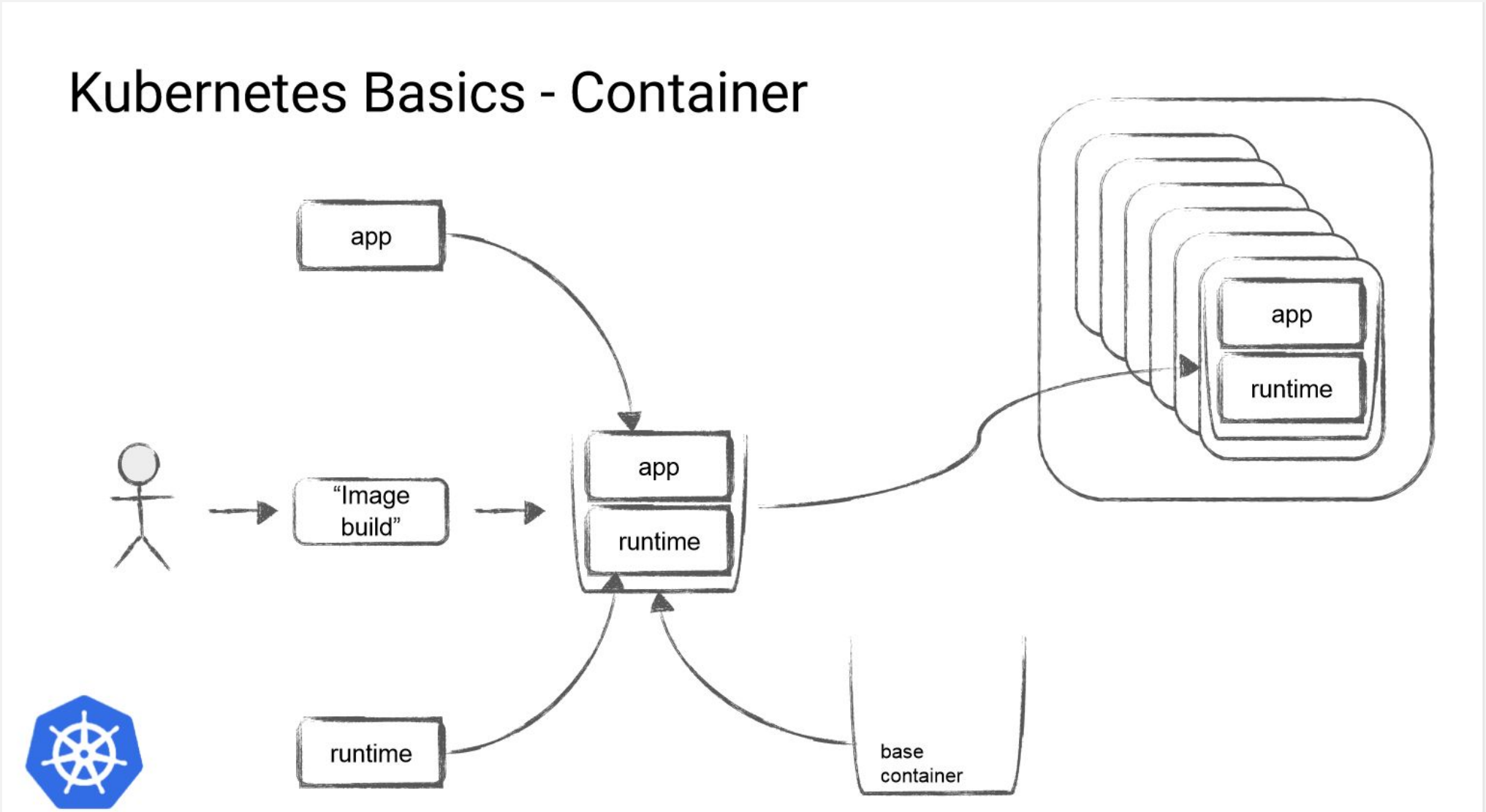
https://www.slideshare.net/QAware/cloud-platforms-demystified (Matthias Haeussler, Novatec Consulting GmbH)

Example: Cloud Foundry - Workflow



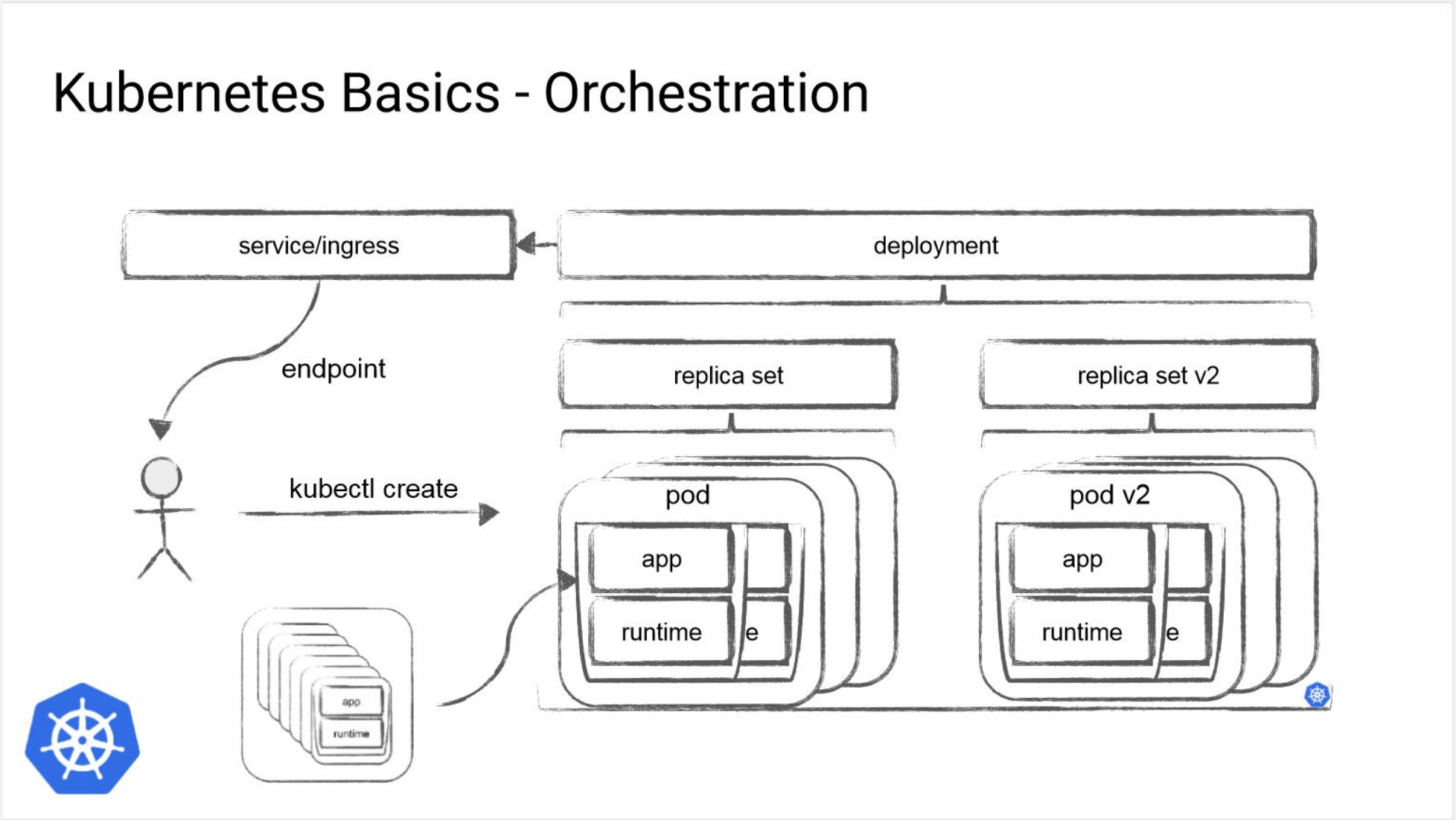
https://www.slideshare.net/QAware/cloud-platforms-demystified (Matthias Haeussler, Novatec Consulting GmbH)

In comparison: Kubernetes – Workflow (1/2)



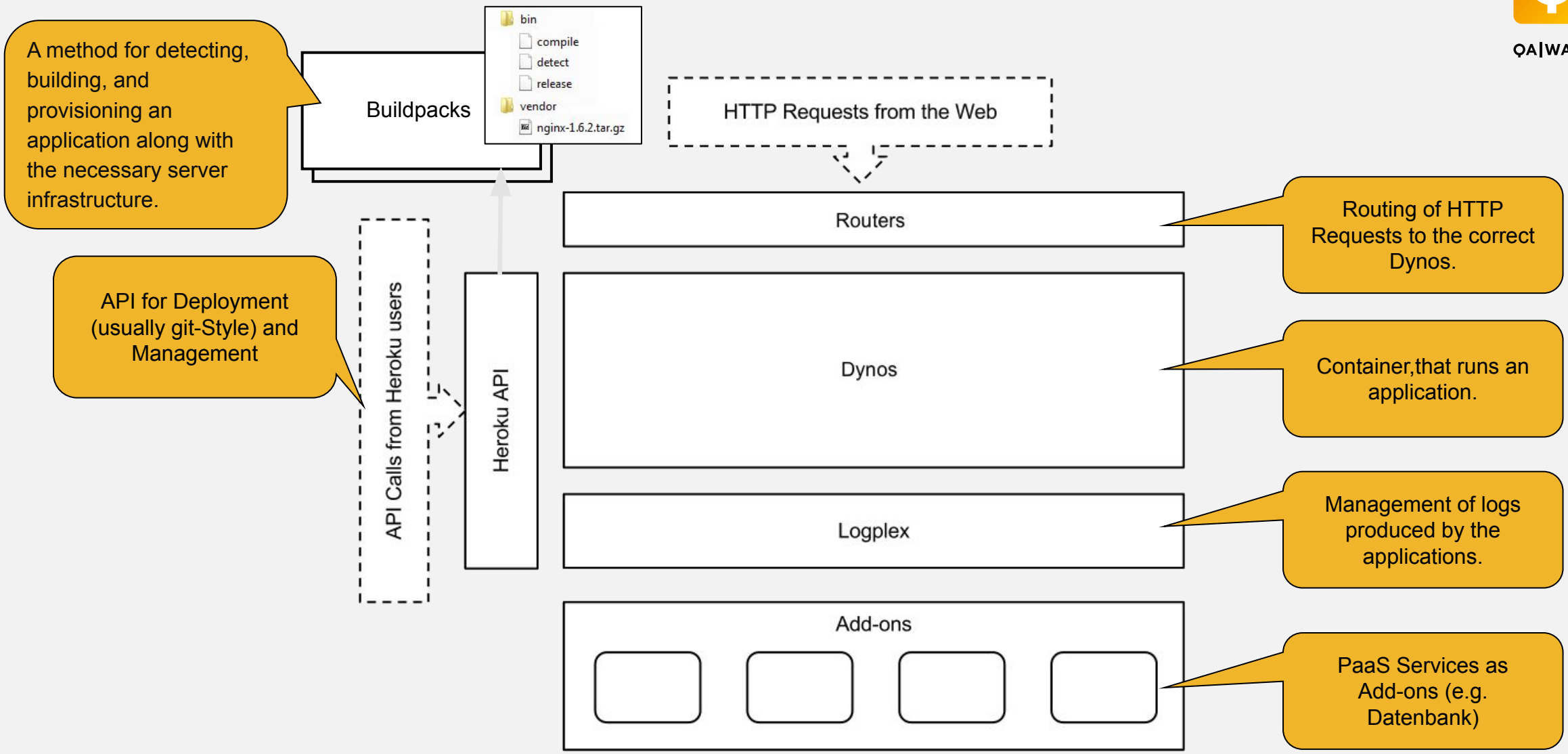
https://www.slidesnare.net/QAware/cloud-platforms-demystified (Matthias Haeussler, Novatec Consulting GmbH)

In comparison: Kubernetes – Workflow (2/2)



https://www.slideshare.net/QAware/cloud-platforms-demystified (Matthias Haeussler, Novatec Consulting GmbH)

High-Level Architecture of a PaaS, looking at Heroku.

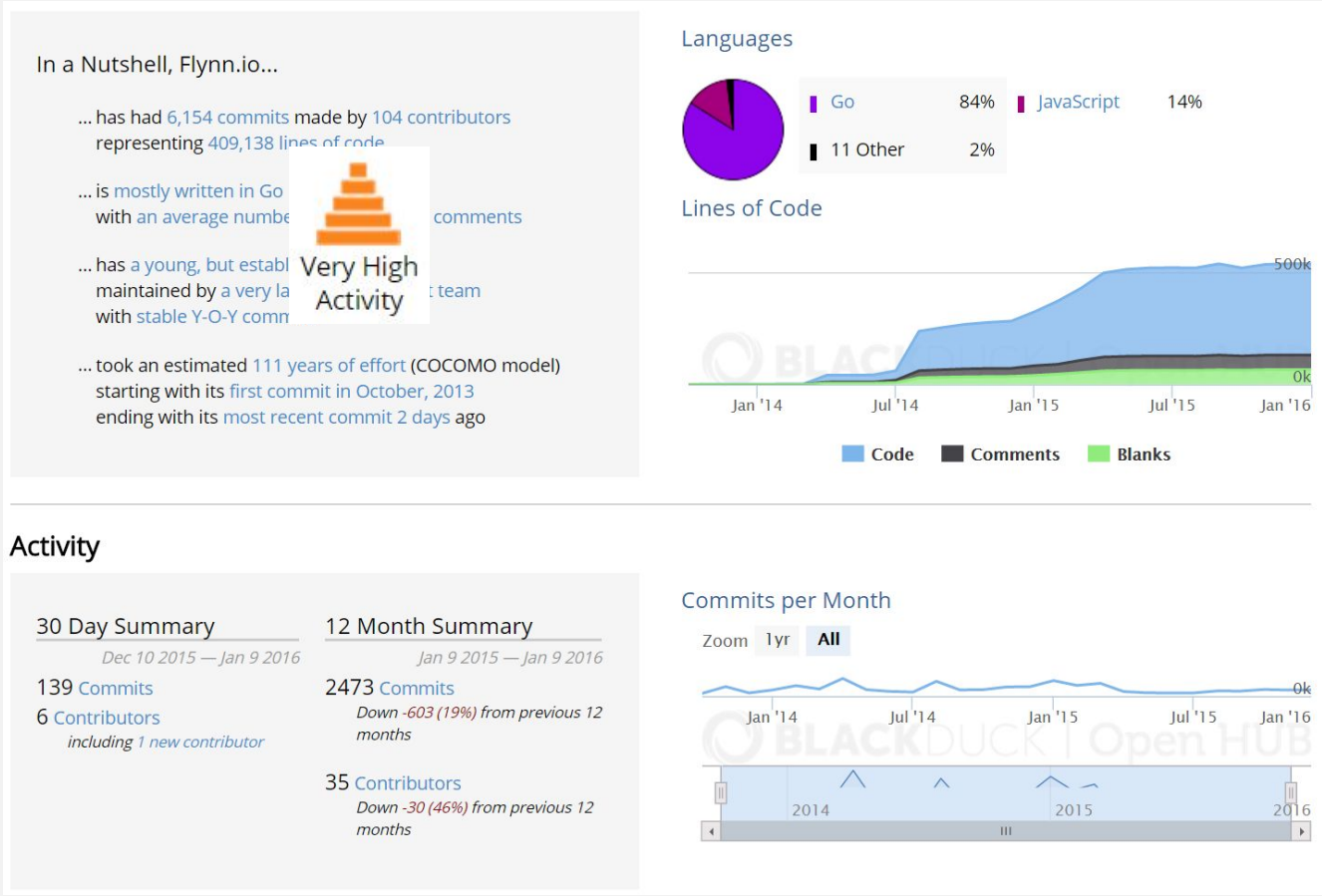


Erweitert von: http://chimera.labs.oreilly.com/books/1234000000018/ch02.html

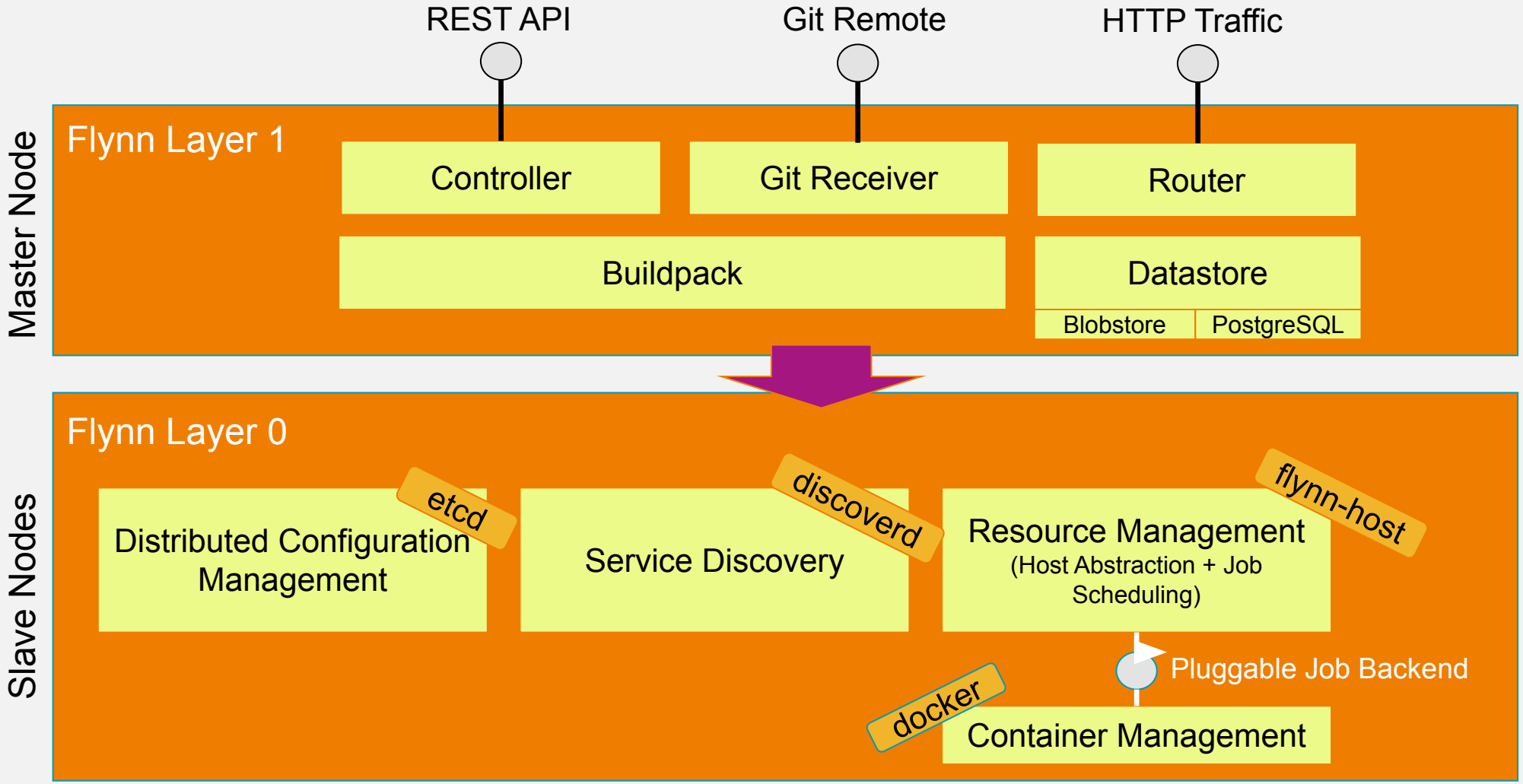
Private PaaS Clouds

Beispiel: Flynn

- Private PaaS auf Basis Docker
- Open-Source-Projekt unter einer BSD Lizenz



Die Architektur von Flynn

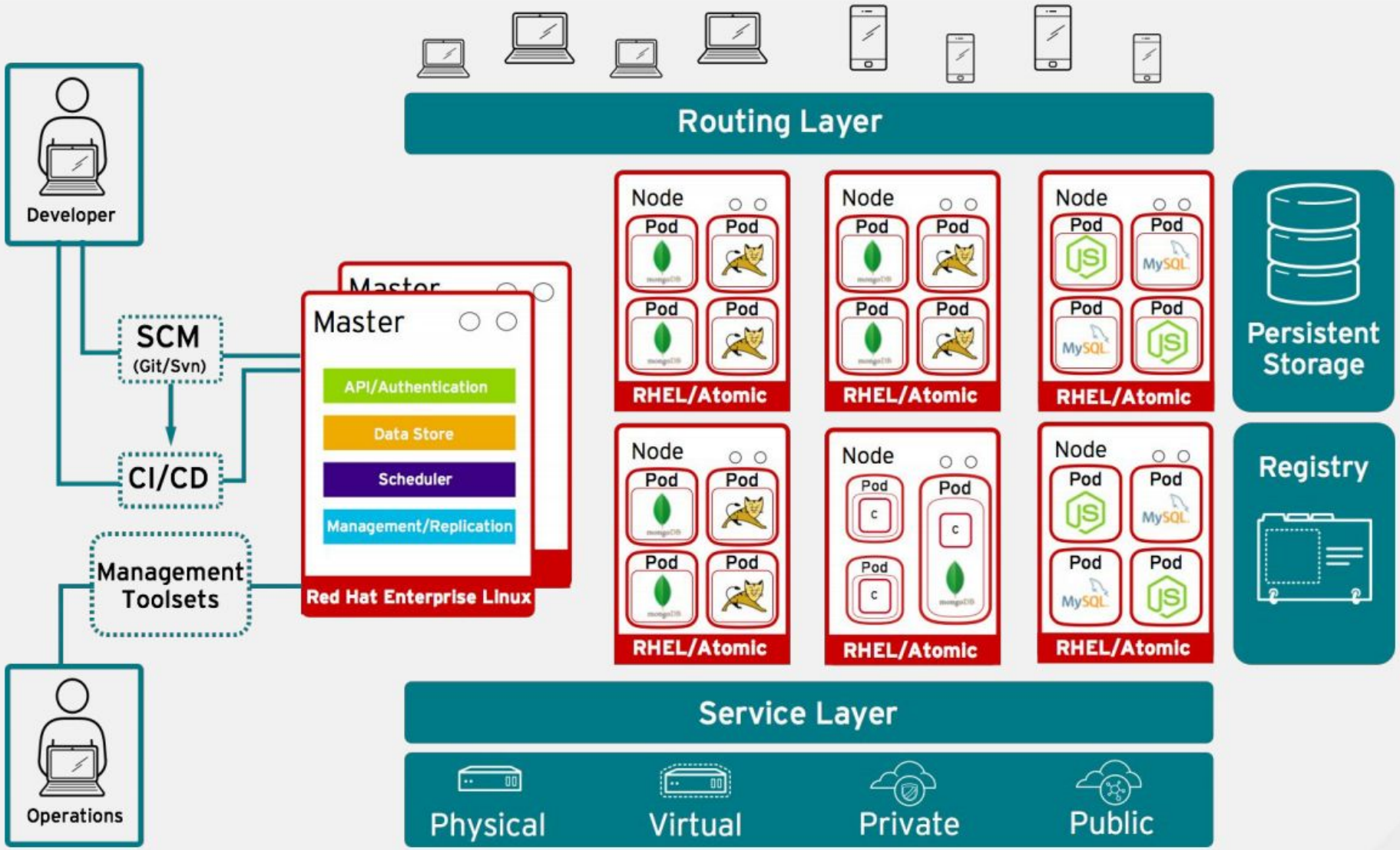


Alternative Private PaaS Clouds

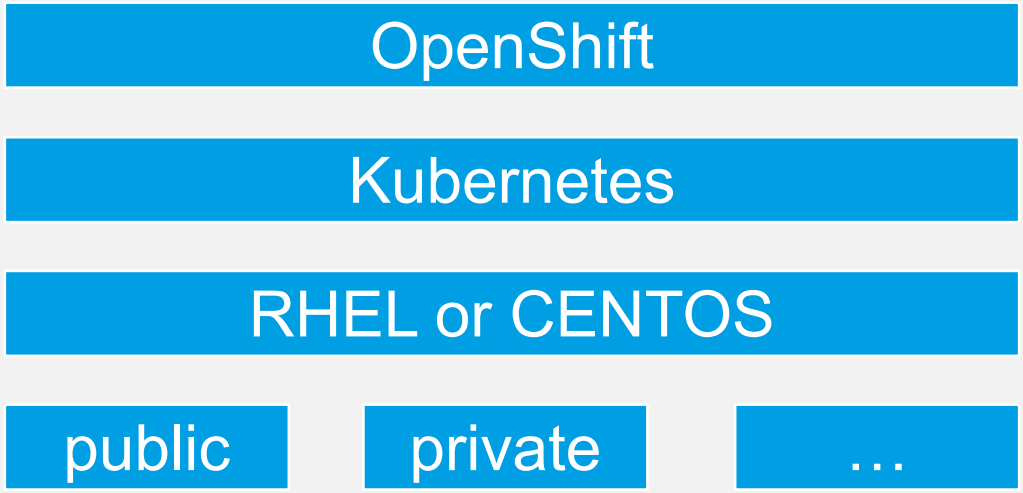
- OpenShift (https://www.openshift.com, PaaS created by Red Hat, based on Kubernetes)
- CloudFoundry (http://www.cloudfoundry.org, PaaS based on Kubernetes
- PaaSTA (https://github.com/yelp/paasta). Open-Source private PaaS based on Kubernetes (originally Mesos and Marathon)

Example: OpenShift.

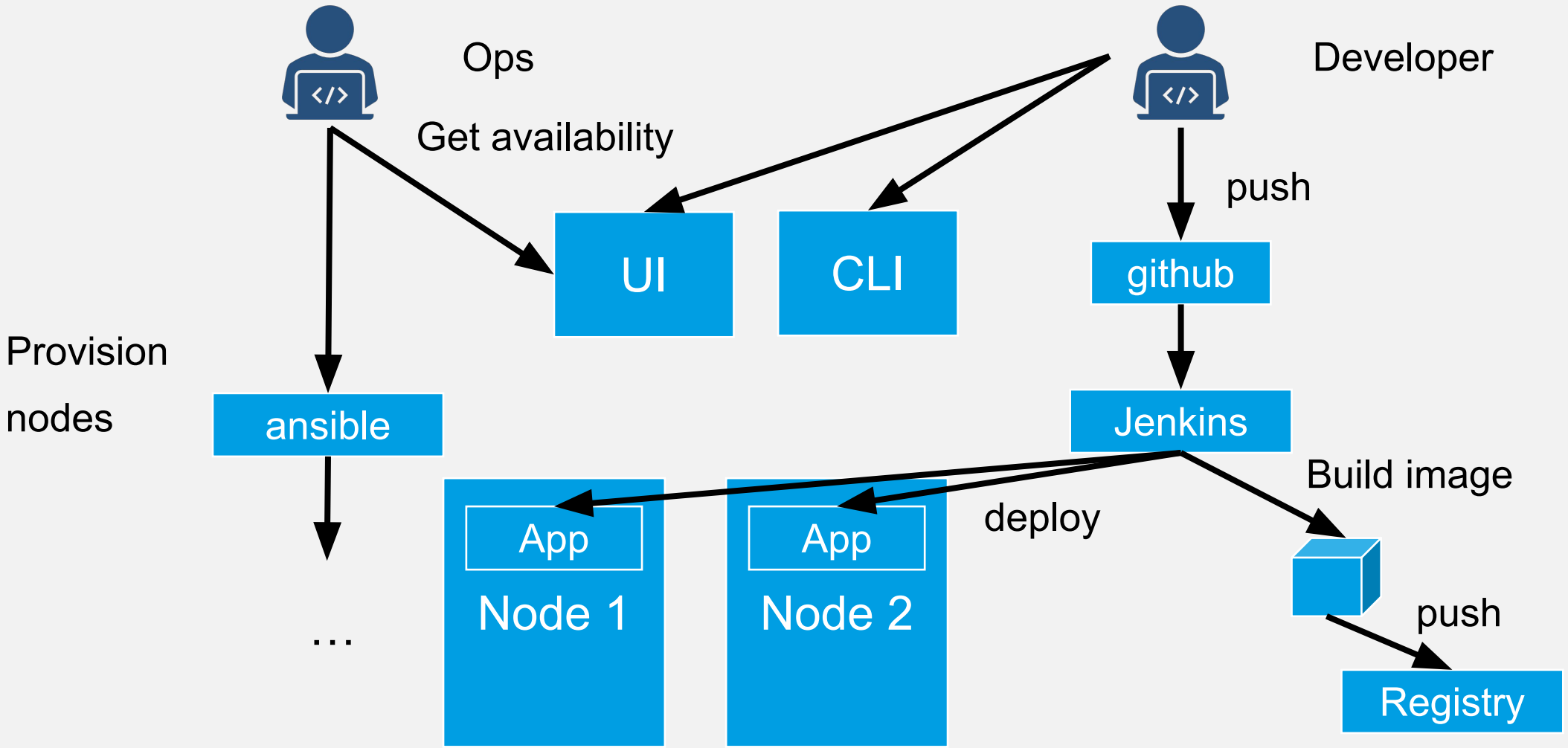
https://www.redhat.com/cms/managed-files/OpenShift.pdf



OpenShift (Whiteboard)



OpenShift (Whiteboard)



Public PaaS Clouds

Ein PaaS-Vergleich über die angebotenen APIs und Services.

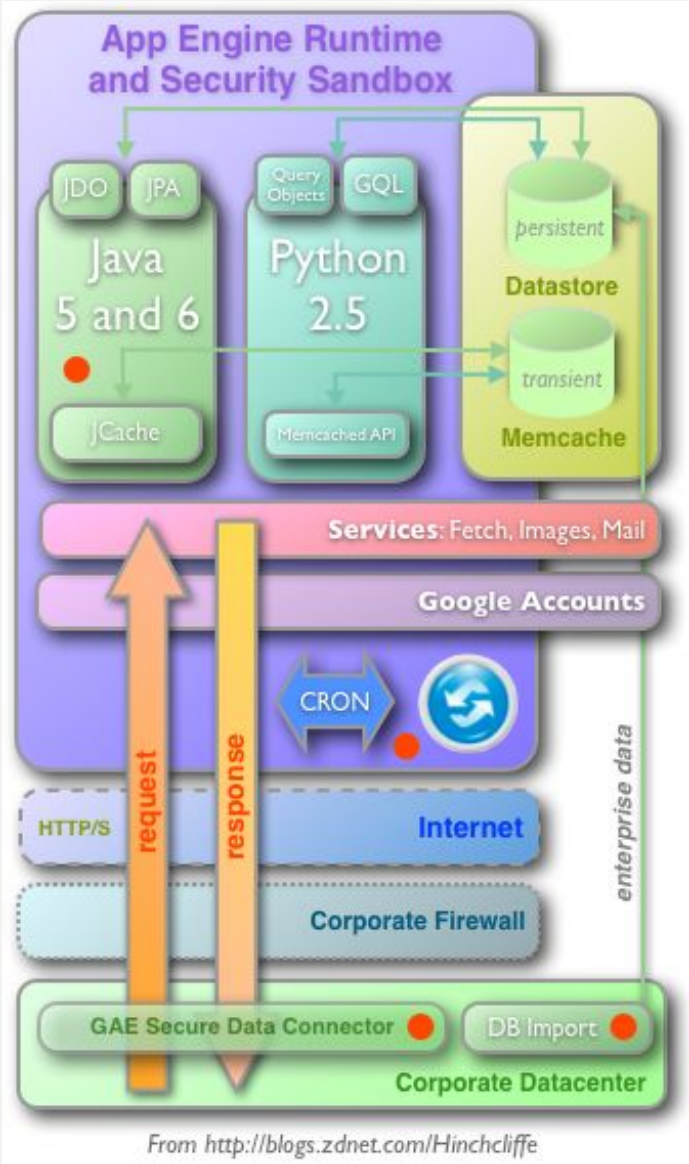
	GAE-J	AWS
Datenspeicher	App Engine Datastore (Key/Value mit JDO und JPA API) Cloud Storage (Objekte) Blobstore (Dateien), Cloud SQL (relational)	DynamoDB (Key/Value), S3 (Objekte und Dateien), RDS (relational)
Messaging	Mail (mit javax.mail API), XMPP, Channel (Push-API)	SES (E-Mails), SNS (Notifications), SQS (Message Queuing)
Engine	Servlet Engine, Capabilities, LogService	Elastic Beanstalk (Servlet Engine)
Integration	URLFetch, App Identity, OAuth	
Parallele Verarbeitung	Task Queue	Elastic MapReduce
Volltextsuche	Search, Prospective Search	CloudSearch
Cache	Memcache mit JCache-API	ElastiCache
User-Authentifzierung	Google Accounts, OpenID	IAM
SaaS-APIs	Google Data API, Images, Conversion	SWF (Workflows)
Mandantenfähigkeit	Multitenancy (Namespaces API)	

The Google App Engine

The Google App Engine (GAE) is Google's PaaS offering.
Applications run within Google's infrastructure.
Running applications is free within certain quotas. Beyond that, costs are incurred based on factors such as service calls, storage volume, and actual CPU seconds used.

Supported languages:

The Google App Engine - Overview



From http://blogs.zdnet.com/Hinchcliffe

Some GAE Services (1/2)

Datastore

- Persistent storage implemented as a key/value database.
- Transactions are atomic. Write operations are strongly consistent. Queries are eventually consistent.
- Definition, querying, and manipulation of data are done through a custom language called GQL (Google Query Language, similar to SQL).
- High-level APIs such as JDO and JPA are available. These are standardized within the Java/JEE ecosystem. The API is implemented using the DataNucleus framework.

Memcache

- High-performance temporary data storage in memory (in-memory data grid).
- Each entry is stored with a unique key.
- Each entry is limited to 1 MB.
- An expiration time in seconds is specified, after which the entry will be removed from the Memcache.
- Data may also be evicted earlier, depending on Memcache usage.
- The JCache API is available as a high-level API.

Some GAE Services (2/2)

URL Fetch

- Access to content on the internet.
- Supported methods: GET, POST, PUT, DELETE, and HEADER.
- Access is allowed on ports within the ranges 80-90, 440-450, and 1024-65535.
- Requests and responses are each limited to 1 MB.

Users

- Integration with a single sign-on system.
- Supports Google Accounts and OpenID accounts.
- The JAAS API is used as the high-level API.

XMPP

- Messages can be sent to and received from any XMPP-compatible messaging system.
- Each application has a unique XMPP username.

All APIs:

- App Identity
- Blobstore
- Google Cloud Storage
- Capabilities
- Channel
- Conversion
- Images
- Mail
- Memcache
- Multitenancy
- OAuth
- Prospective Search
- Search
- Task Queues
- URL Fetch
- Users
- XMPP

Constraints when running inside of Google App Engine (1 / 2)

A GAE application runs in a sandbox that restricts the application's behavior. This is done to ensure efficient processing and to protect the infrastructure during auto-scaling.

Not all classes of the standard library may be used.

- No custom threads may be created.
- No access to the runtime environment, such as its class loaders.

Constraints when running inside of Google App Engine (2 / 2)

- Communication with other web applications or servers is only possible via URL Fetch, XMPP, or email.
- Requests and responses must not exceed 1 MB in size.
- Webhooks are used as a general architectural mechanism for incoming communication, triggered by events (e.g., warmups), messages, or cron jobs.
- All requests to a GAE application are terminated after 60 seconds.
- Various restrictions apply to data volumes and the number of service calls.

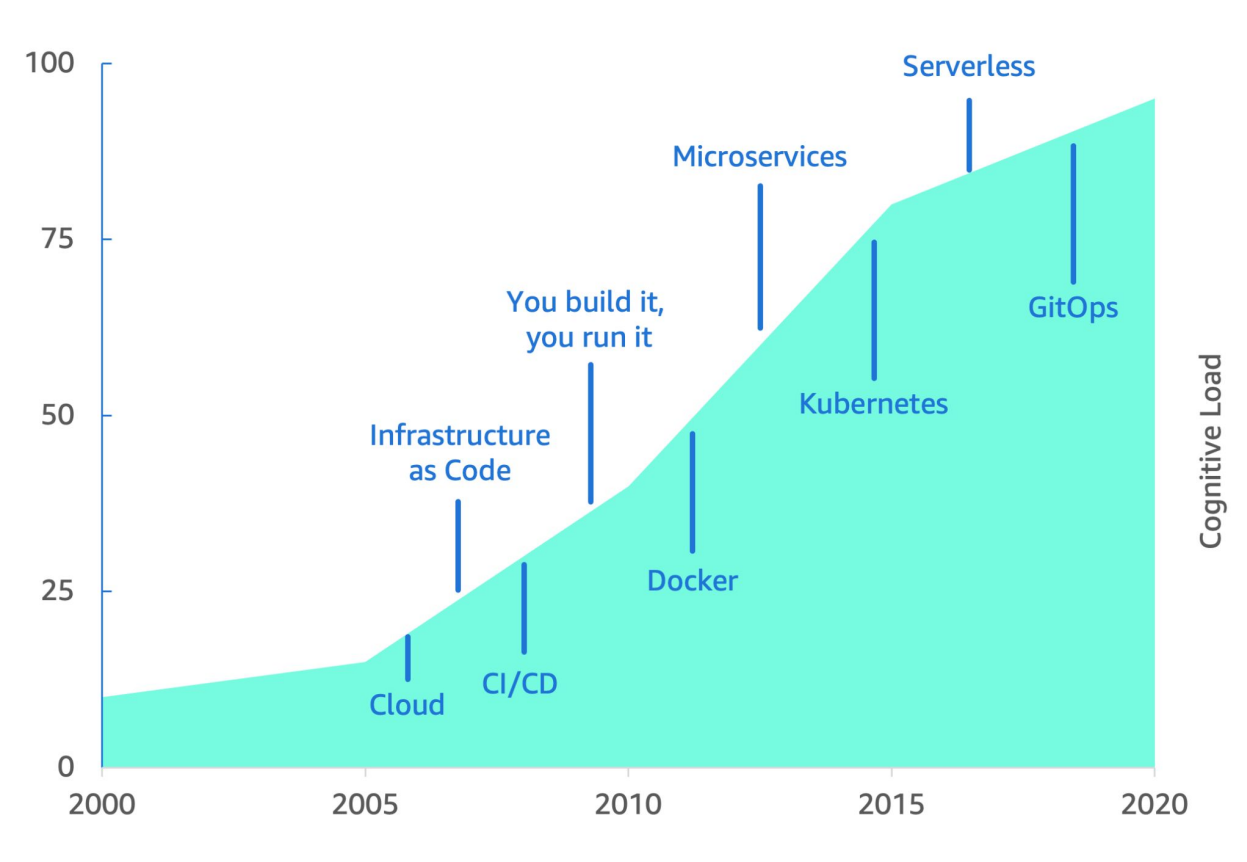
Problems with most PaaS solutions

- Very opinionated, that’s why they work so great in greenfield projects
- Hard to integrate into the brownfield environment of existing enterprises

Enterprises, however, want to profit from the PaaS idea as well.

This led to a engineering specialization called *platform engineering.*

Cloud native resulted in massive cognitive load 🚀



Source: Amazon Web Services

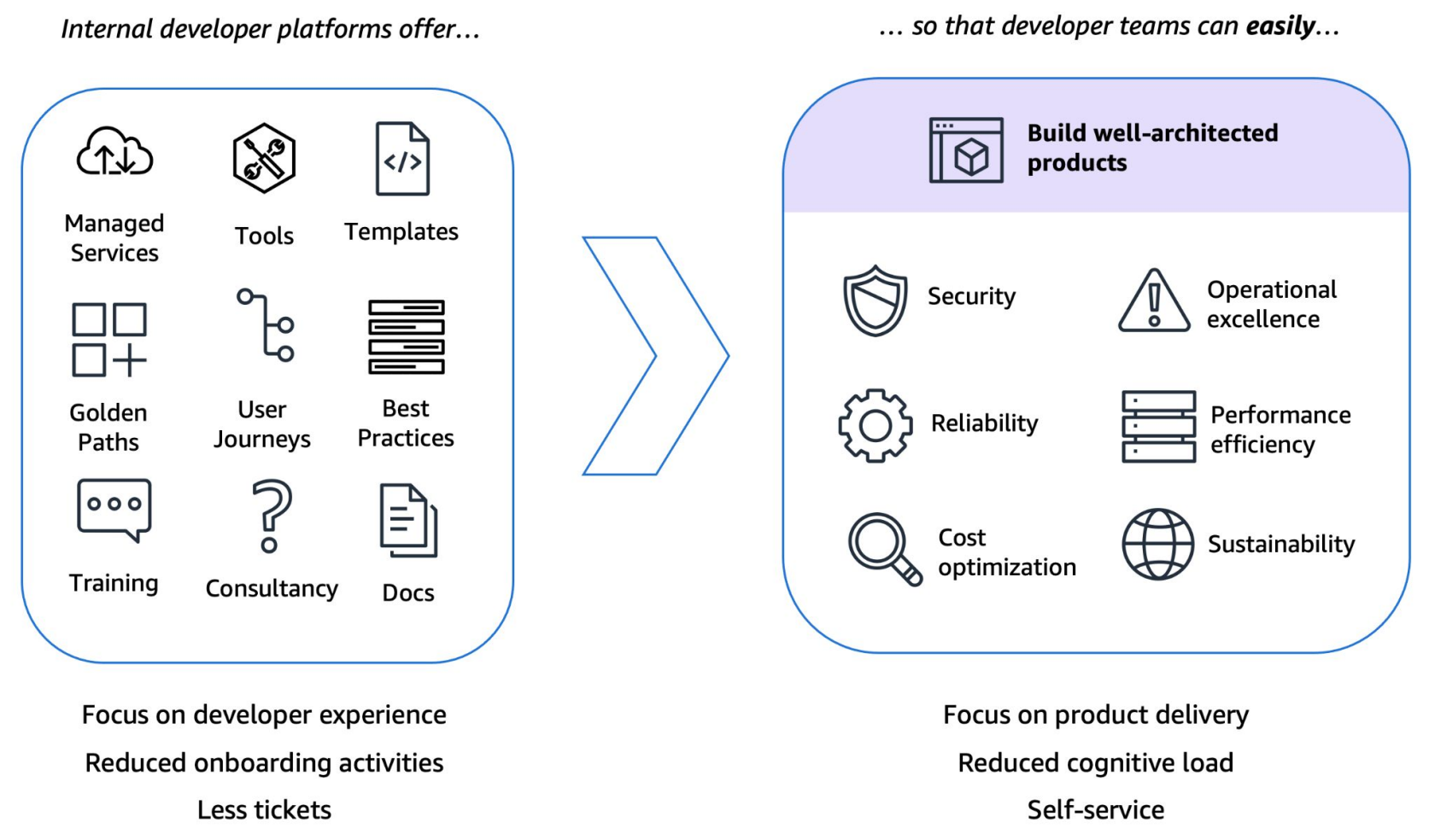
Platform Engineering solves that issue

"Platform engineering is the discipline of designing and building ***toolchains and workflows*** *that enable* ***self-service capabilities for software engineering*** *organizations in the cloud-native era. Platform engineers provide an* ***integrated*** *product most often referred to as an "****Internal Developer Platform****" covering the operational necessities of the* ***entire lifecycle*** *of an application."*

Humanitec

- Specialization of roles, resulting in reduced cognitive load.
- Still DevOps, with a central interface: the platform.
- Fosters reuse and organizational scaling.
- Automated integration means more time for actal software engineering, i.e. creating business value.

Internal Developer Platforms



Source: Amazon Web Services