qaware.de

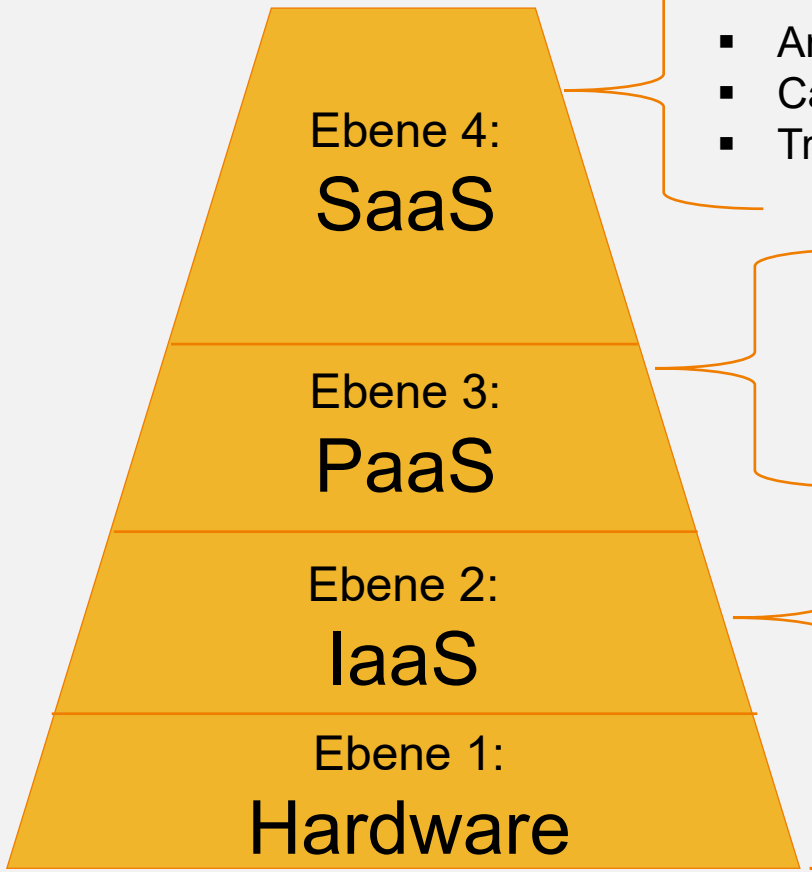
QA|WARE
SOFTWARE ENGINEERING

Platform-as-a-Service (PaaS)

Franz Wimmer
franz.wimmer@qaware.de

Grundlagen zu einer PaaS-Cloud

Das Schichtenmodell des Cloud Computing: Vom Blech zur Anwendung.



➔ Kunden, Endnutzer

- Anpassbare Softwaredienste
- CaaS: Component-as-a-Serivce (z.B. Google Charts)
- Transparente Updates

➔ Entwickler

- **Programmierschnittstellen (APIs)**
- **Plattformdienste**
- **Abstraktion der technischen Infrastruktur**

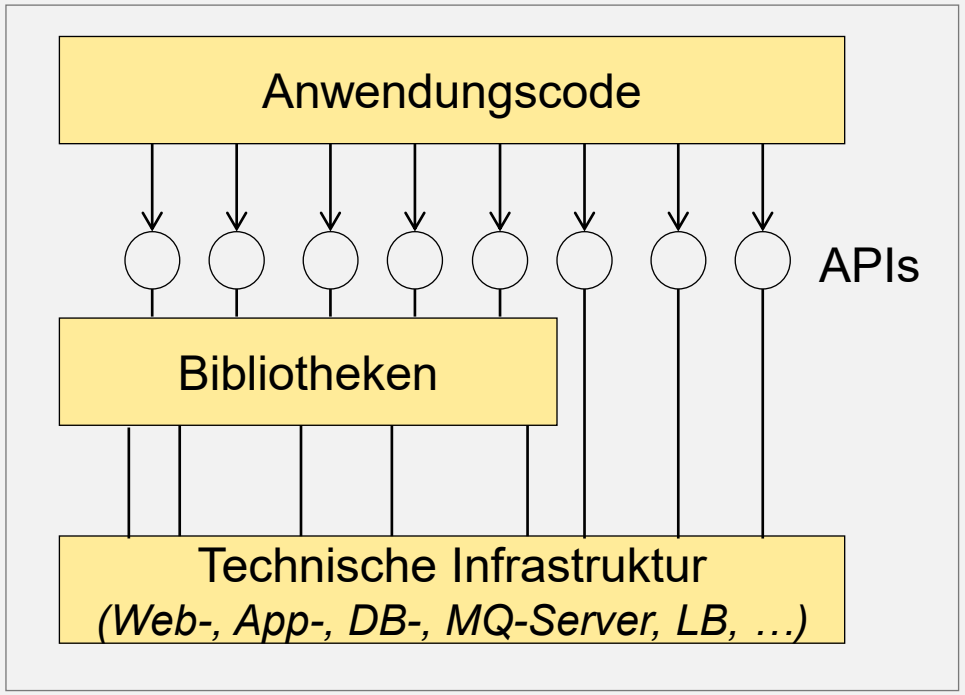
➔ Administratoren

- Elastizität
- Virtuelle Ressourcenpools
- Technische Infrastruktur:
  Maschinen, Server (DNS, DHCP, LB, NAS, …)

- Rechner
- Netzwerk
- Storage

Das Problem: Stovepipe Architecture.
Anwendungen aufwändig von Hand verdrahten.

Das System: Mühevoll verdrahtet.

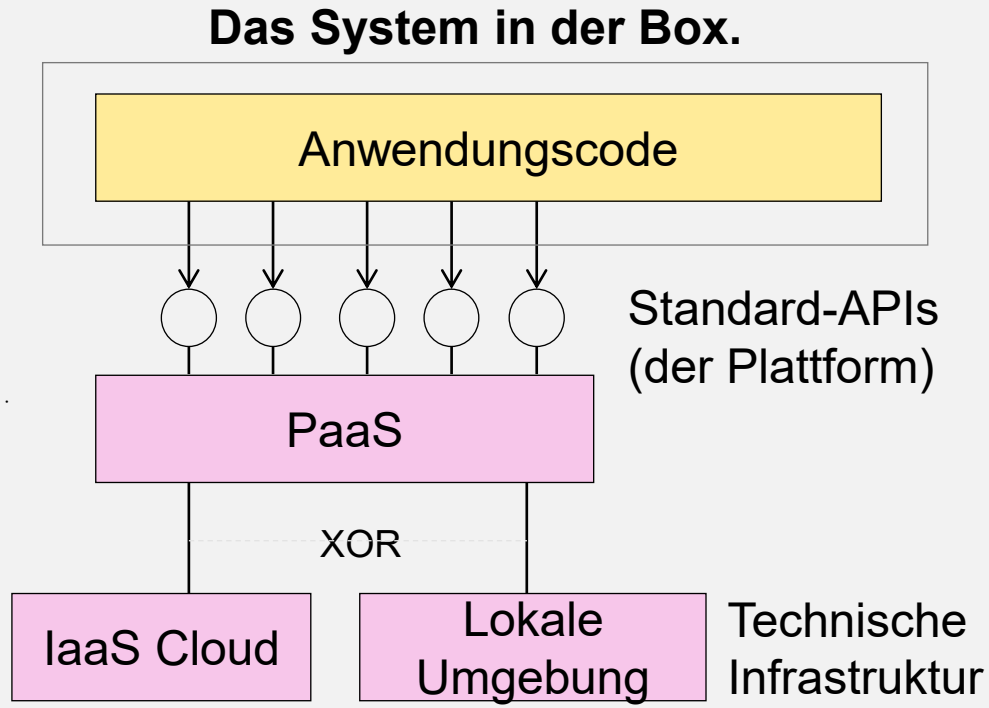


Die Lösung: Plattform-as-a-Service bietet eine ad-hoc Entwicklungs- und Betriebsplattform.

Die Anwendung wird per Applikationspaket oder als Quellcode deployed. Es ist kein Image mit Technischer Infrastruktur notwendig.

Die Anwendung sieht nur Programmier- oder Zugriffsschnittstellen seiner Laufzeitumgebung. „Engine and Operating System should not matter…".

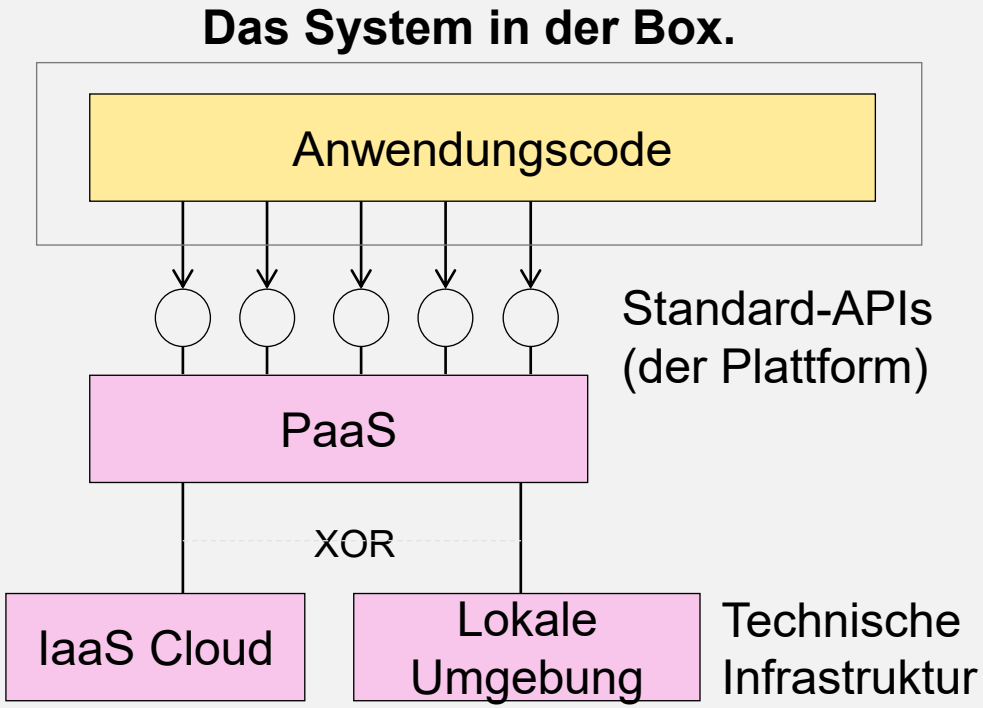
Es erfolgt eine automatische Skalierung der Anwendung.



Die Lösung: Plattform-as-a-Service bietet eine ad-hoc Entwicklungs- und Betriebsplattform.

Entwicklungswerkzeuge (insb. Plugins für IDEs und Buildsysteme sowie eine lokale Testumgebung) stehen zur Verfügung: „deploy to cloud".

Die Plattform bietet eine Schnittstelle zur Administration und zum Monitoring der Anwendungen.

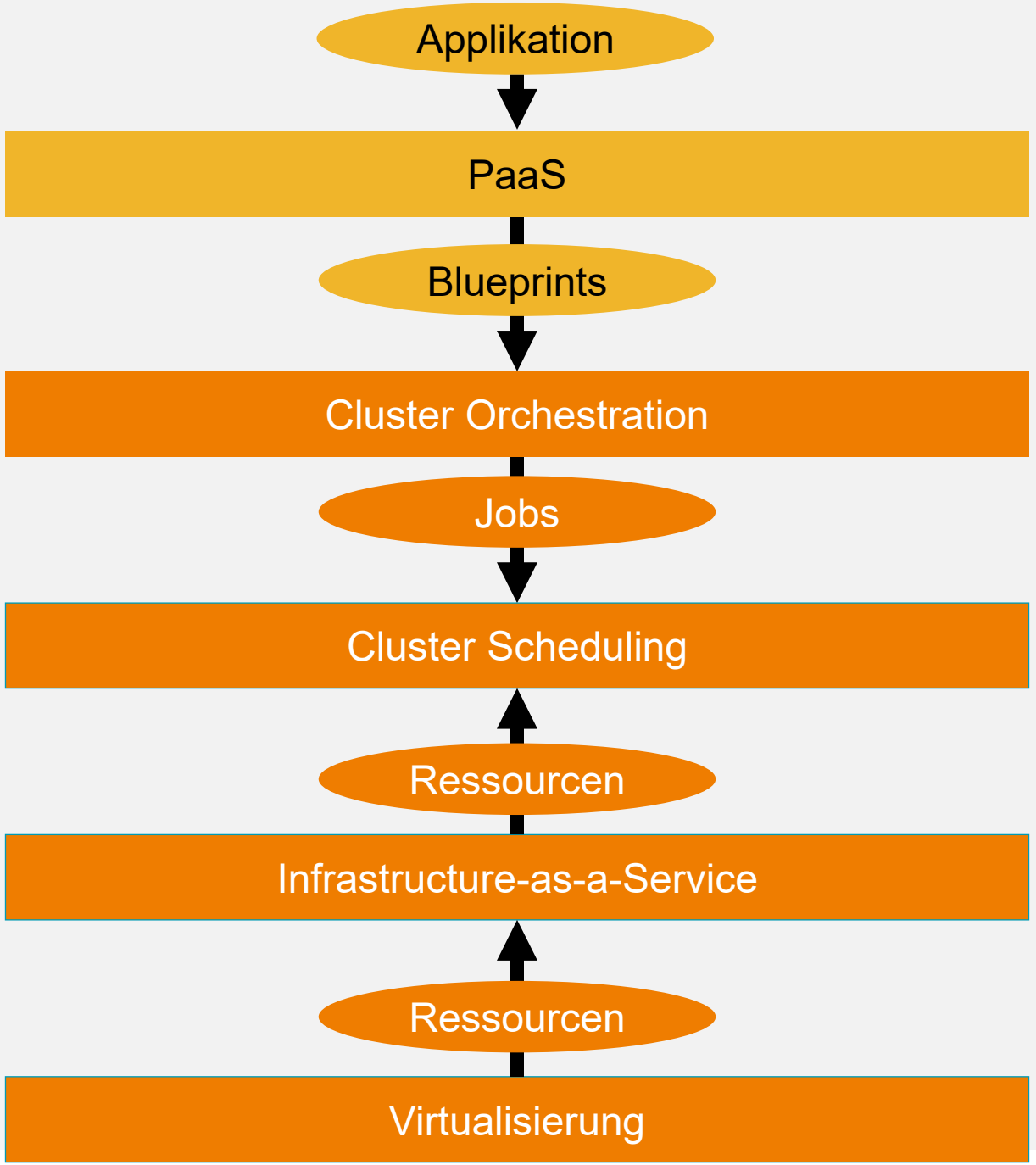


PaaS: Definitionen

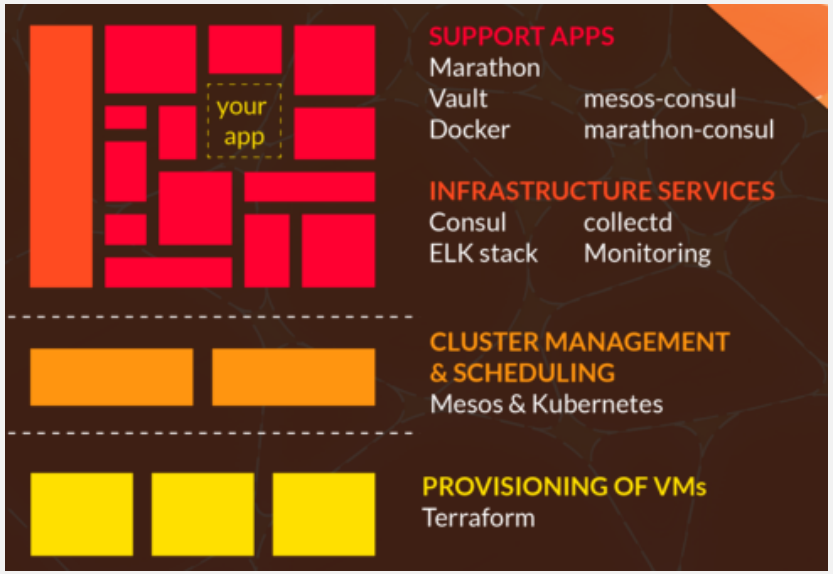
NIST: The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure 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.

Das Big Picture



Die technischen Building-Blocks von PaaS-Lösungen: Sehen sie die Gemeinsamkeiten?



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-fähige Softwarearchitektur

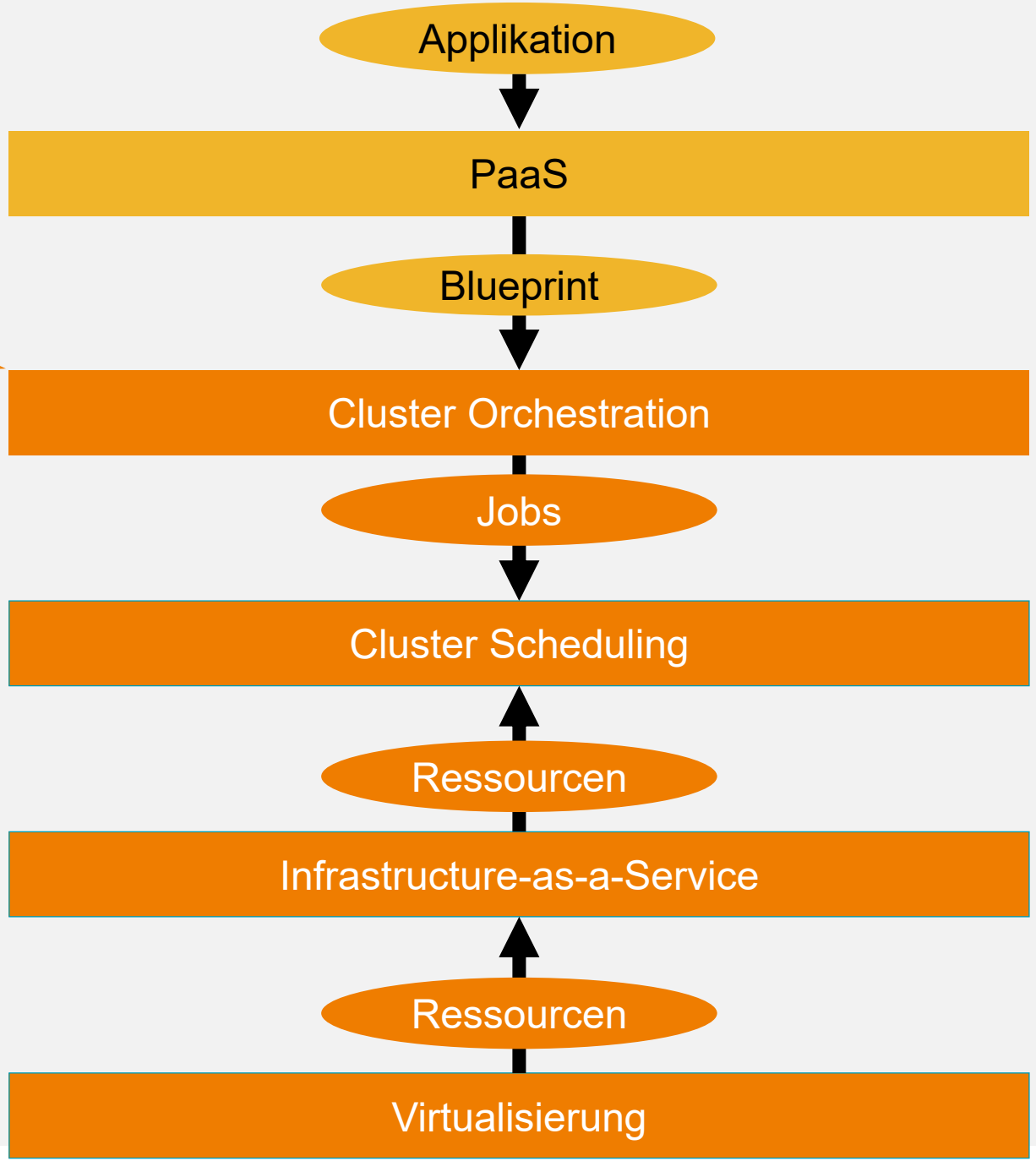
Cluster Orchestration

Cluster Scheduling

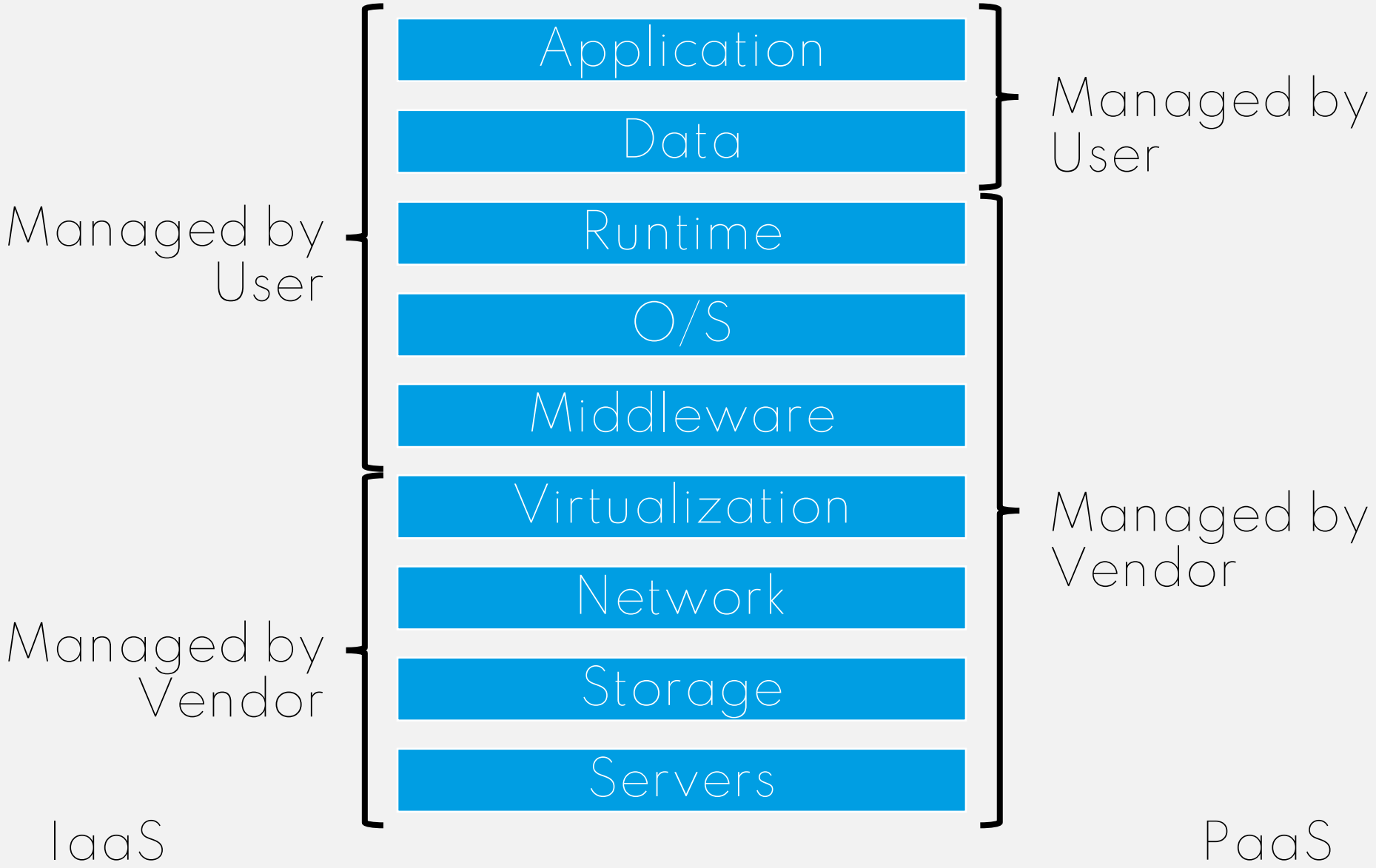
Das Big Picture

Hier ist man bereits bei 80% einer PaaS. Was noch fehlt:

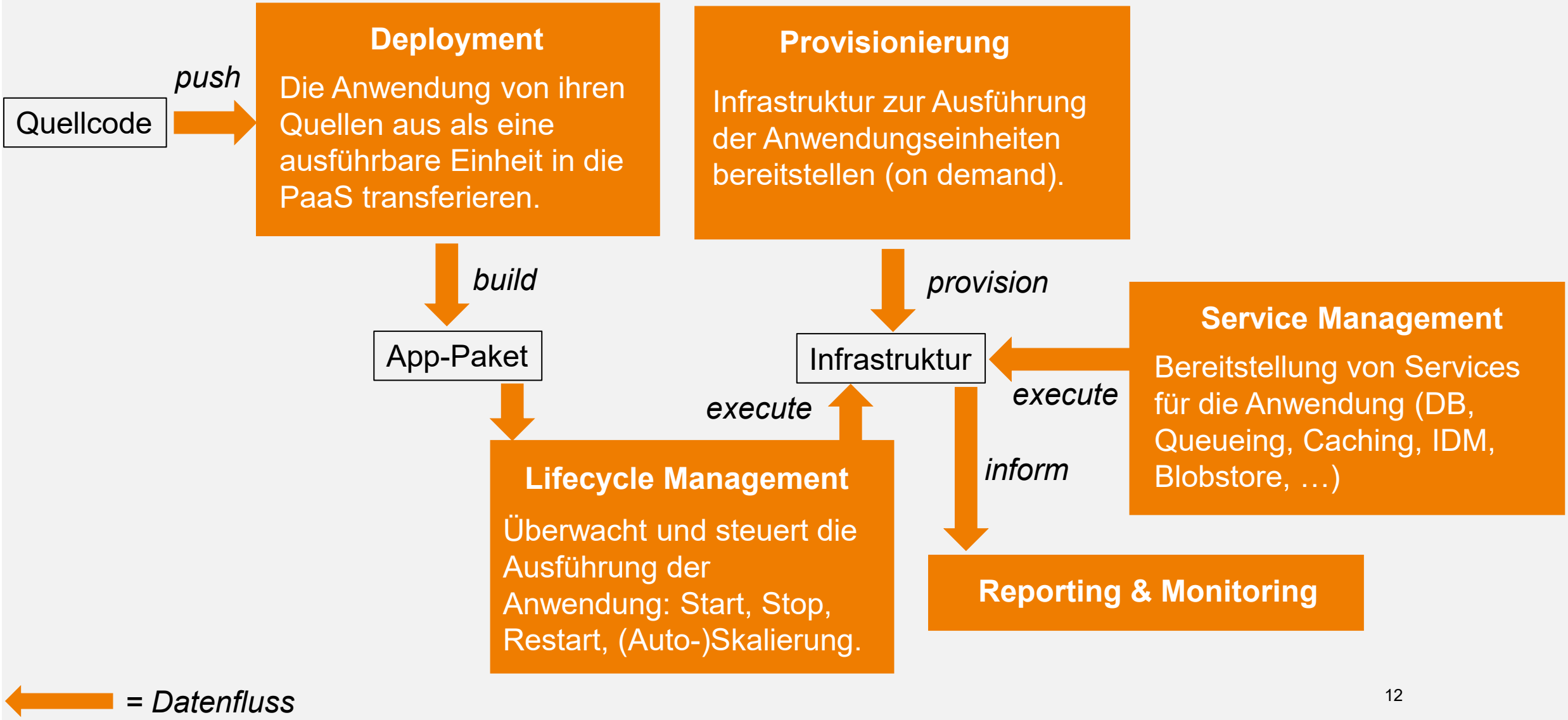
- Wiederverwendung von Infrastruktur / APIs
- Komfort-Dienste für Entwickler



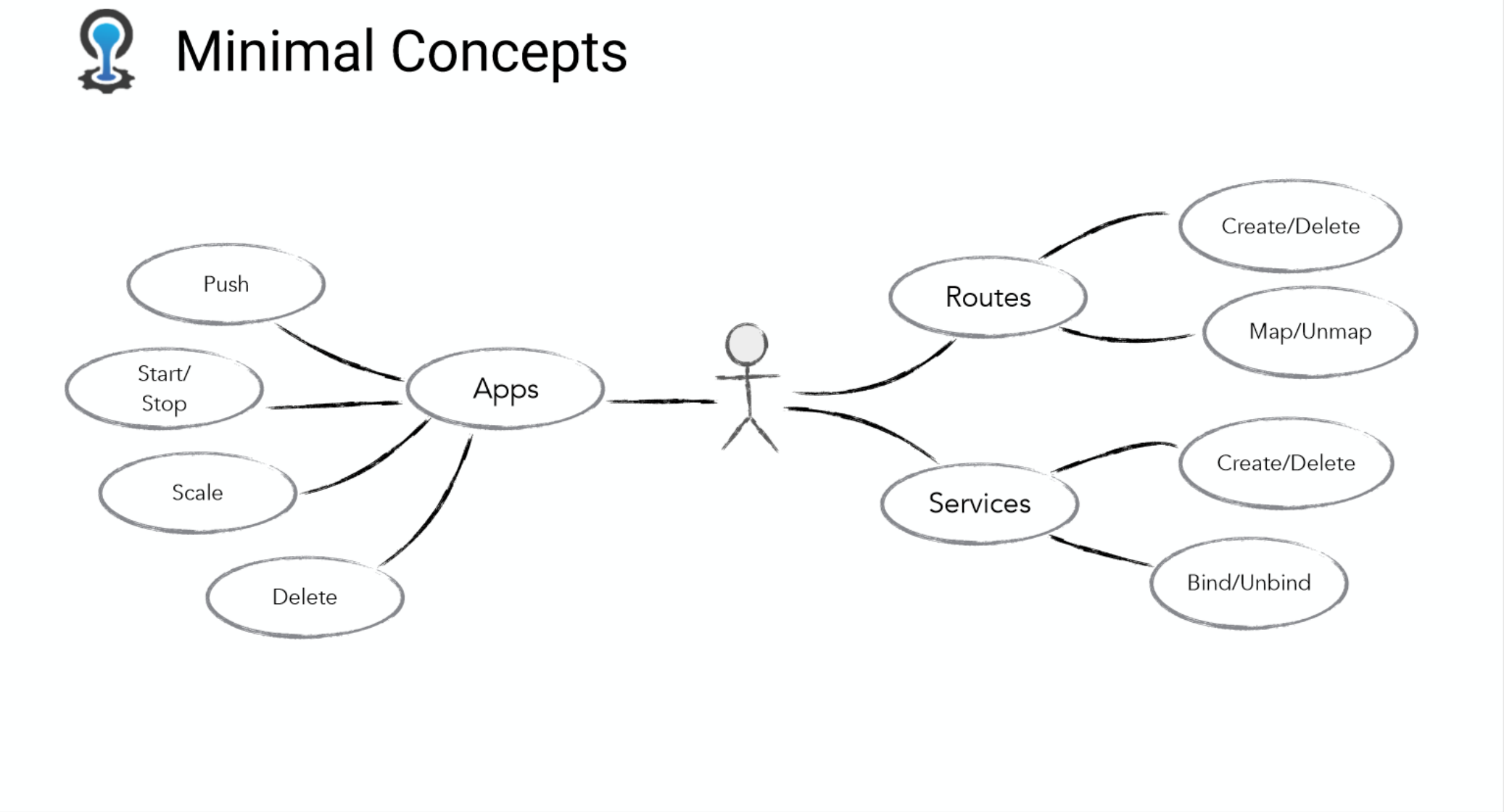
IaaS vs. PaaS


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

Die funktionalen Bausteine einer PaaS Cloud.

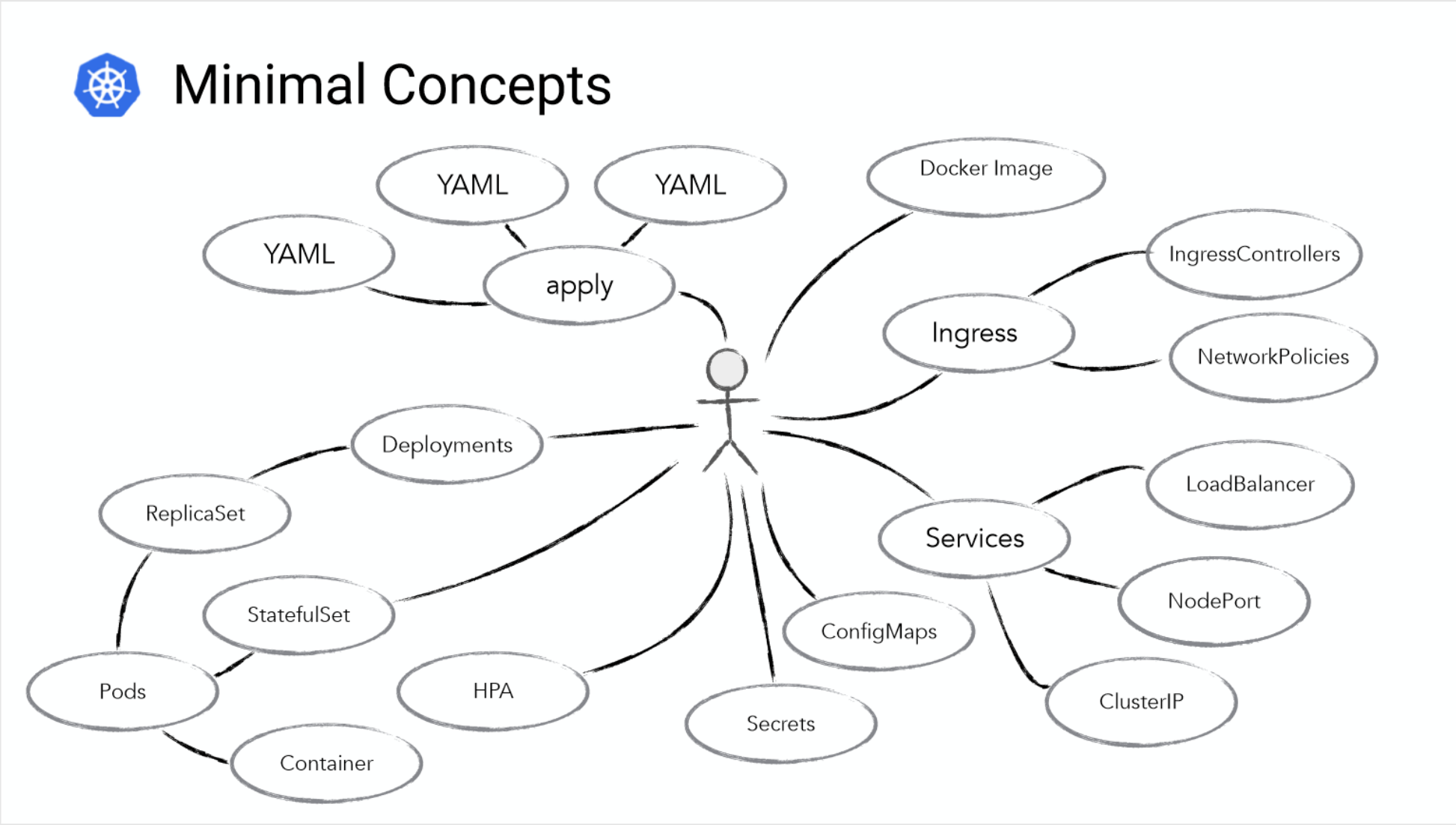


Beispiel: Cloud Foundry



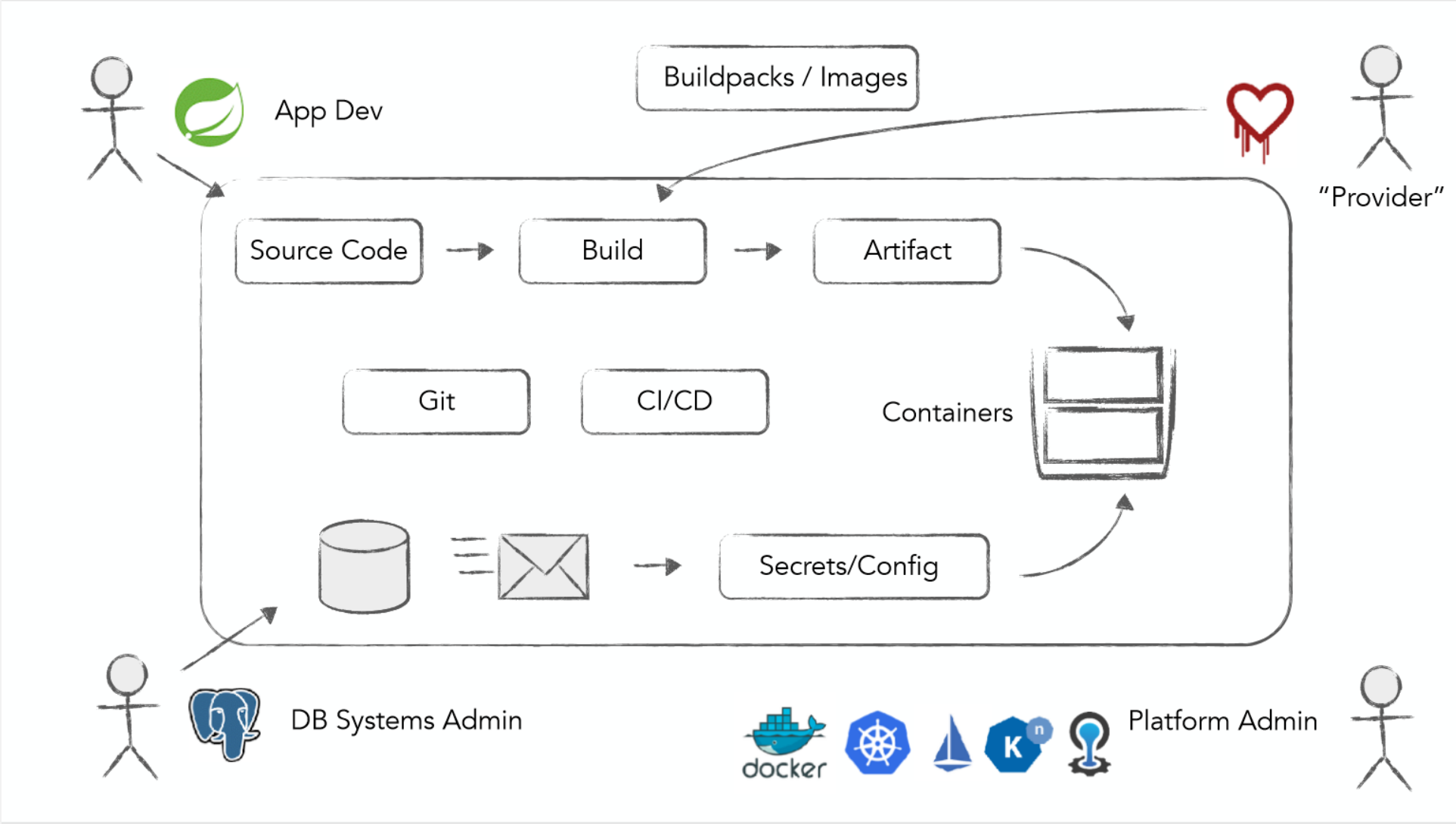
https://www.slideshare.net/QAware/cloud-platforms-demystified (Matthias Haeussler, Novatec Consulting GmbH)

Im Vergleich: Kubernetes



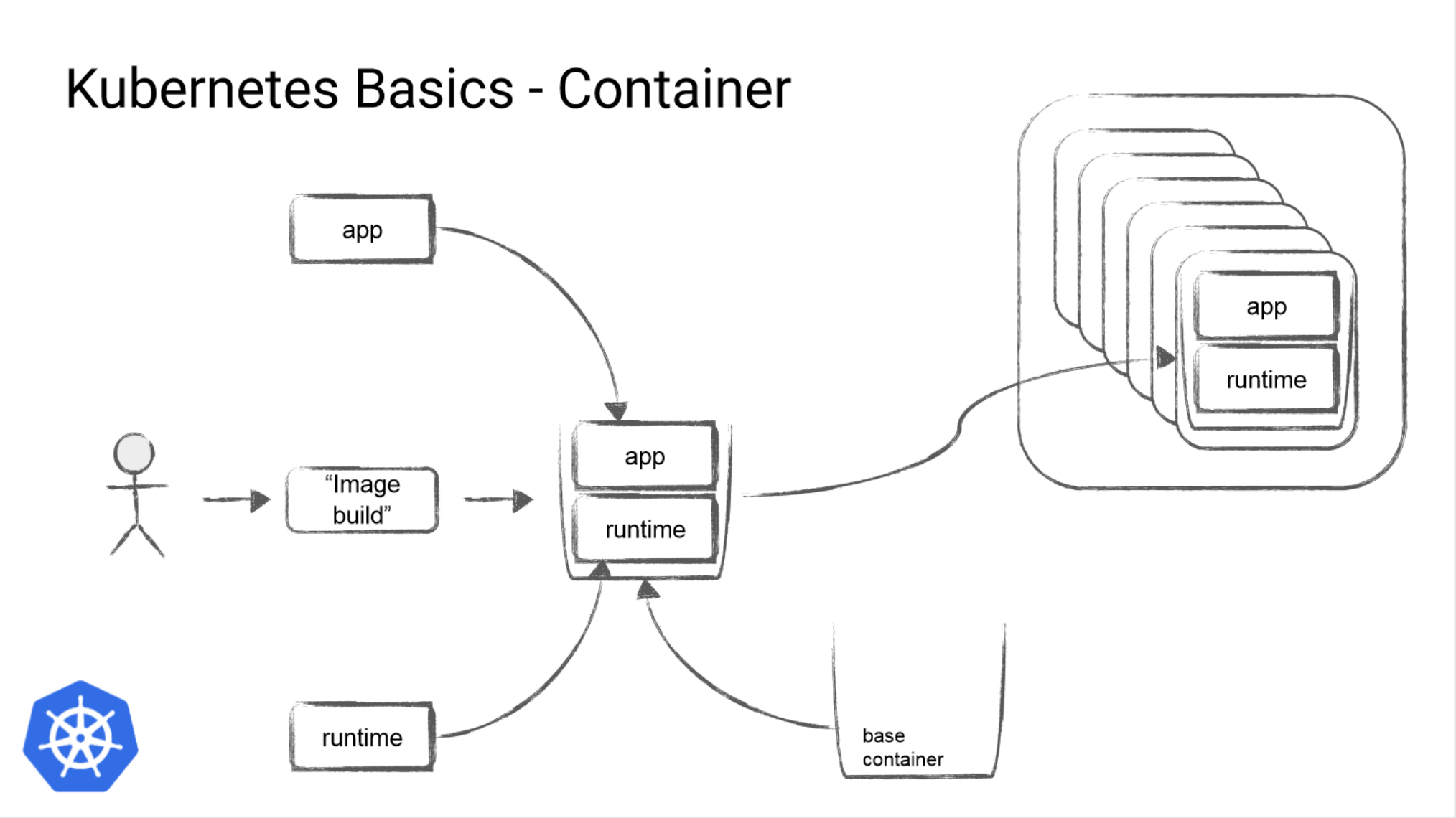
https://www.slideshare.net/QAware/cloud-platforms-demystified (Matthias Haeussler, Novatec Consulting GmbH)

Beispiel: Cloud Foundry - Workflow



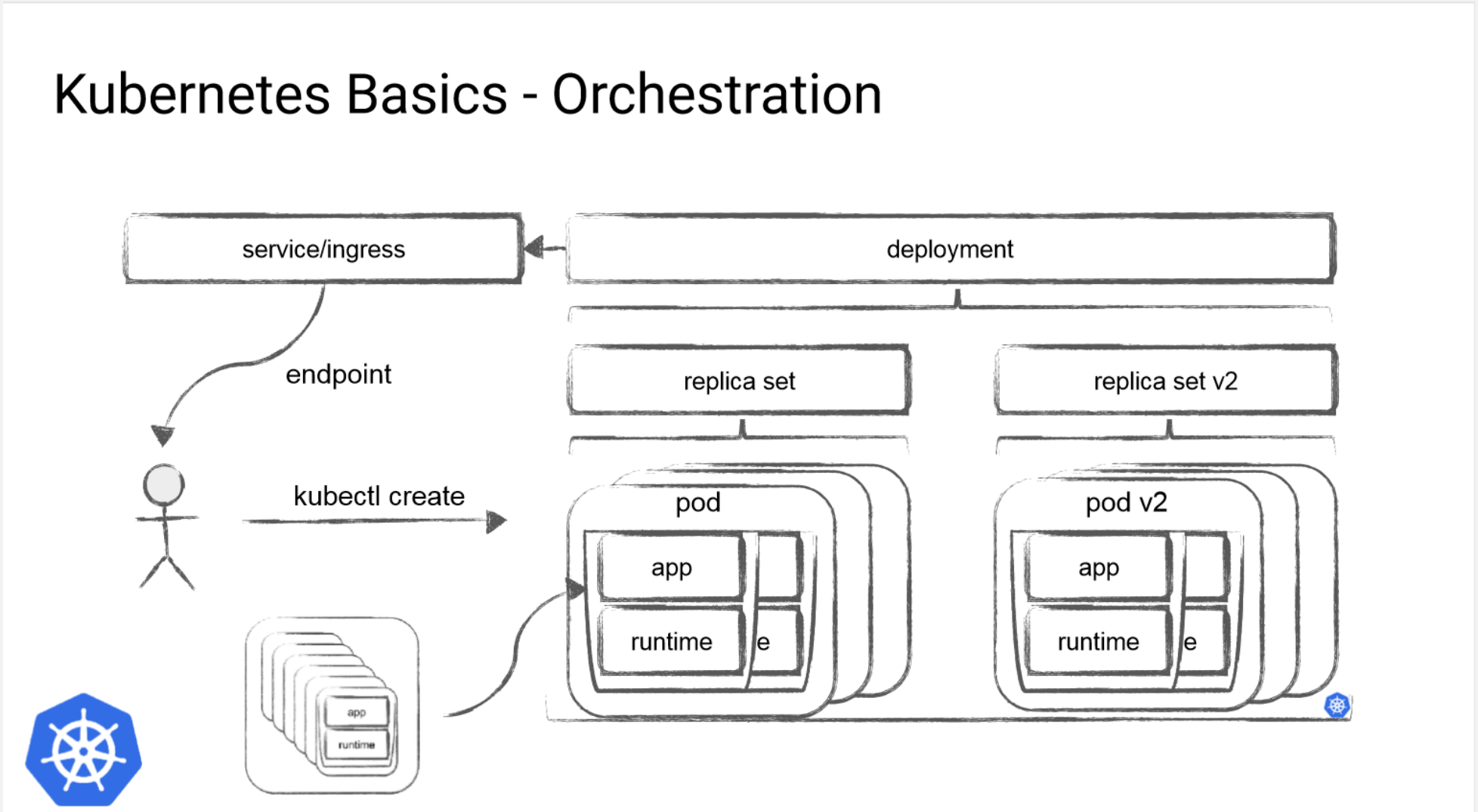
https://www.slideshare.net/QAware/cloud-platforms-demystified (Matthias Haeussler, Novatec Consulting GmbH)

Im Vergleich: Kubernetes – Workflow (1/2)



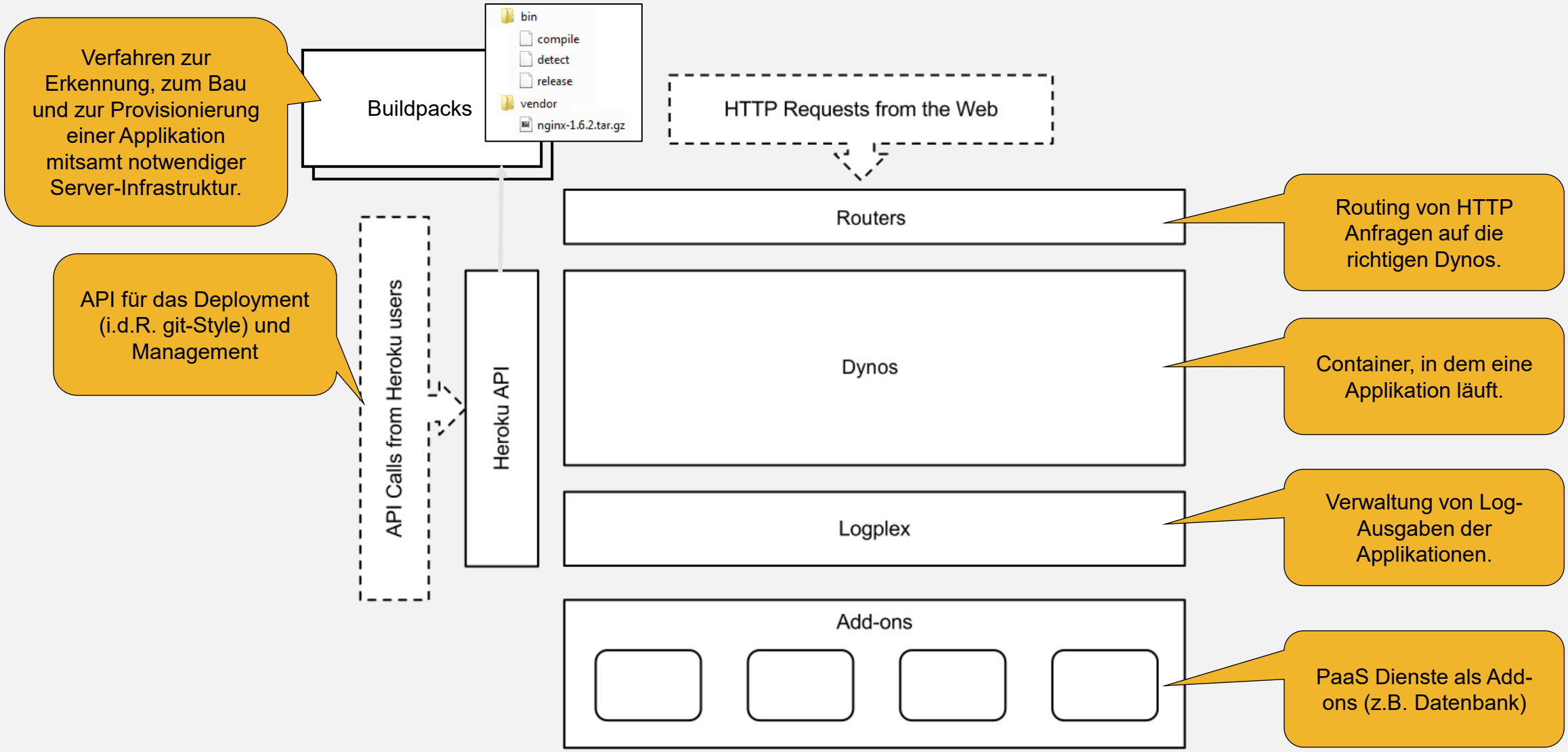
https://www.slideshare.net/QAware/cloud-platforms-demystified (Matthias Haeussler, Novatec Consulting GmbH)

Im Vergleich: Kubernetes – Workflow (2/2)



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

High-Level Architektur einer PaaS am Beispiel Heroku.

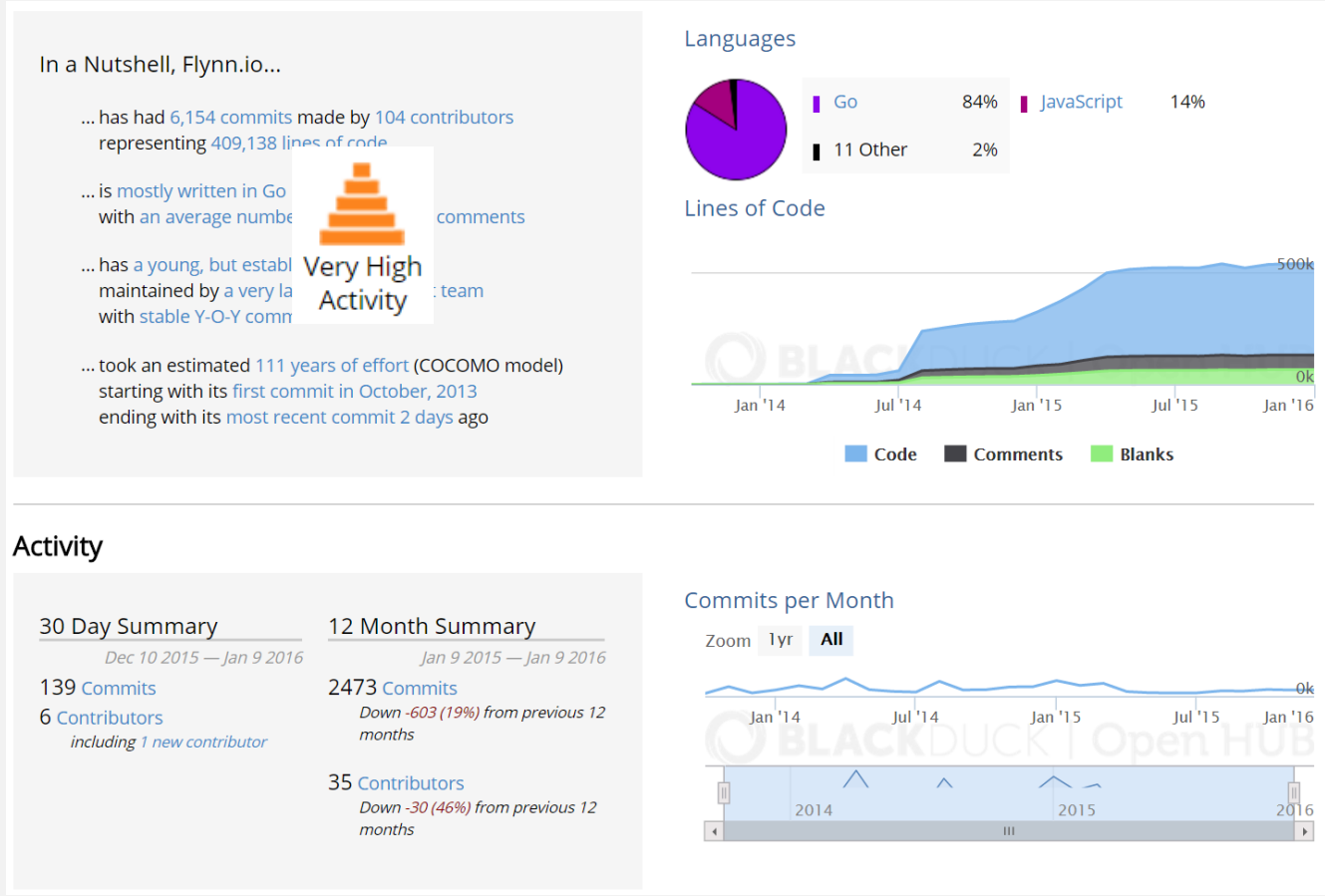


Erweitert von: http://chimera.labs.oreilly.com/books/1234000000018/ch02.html

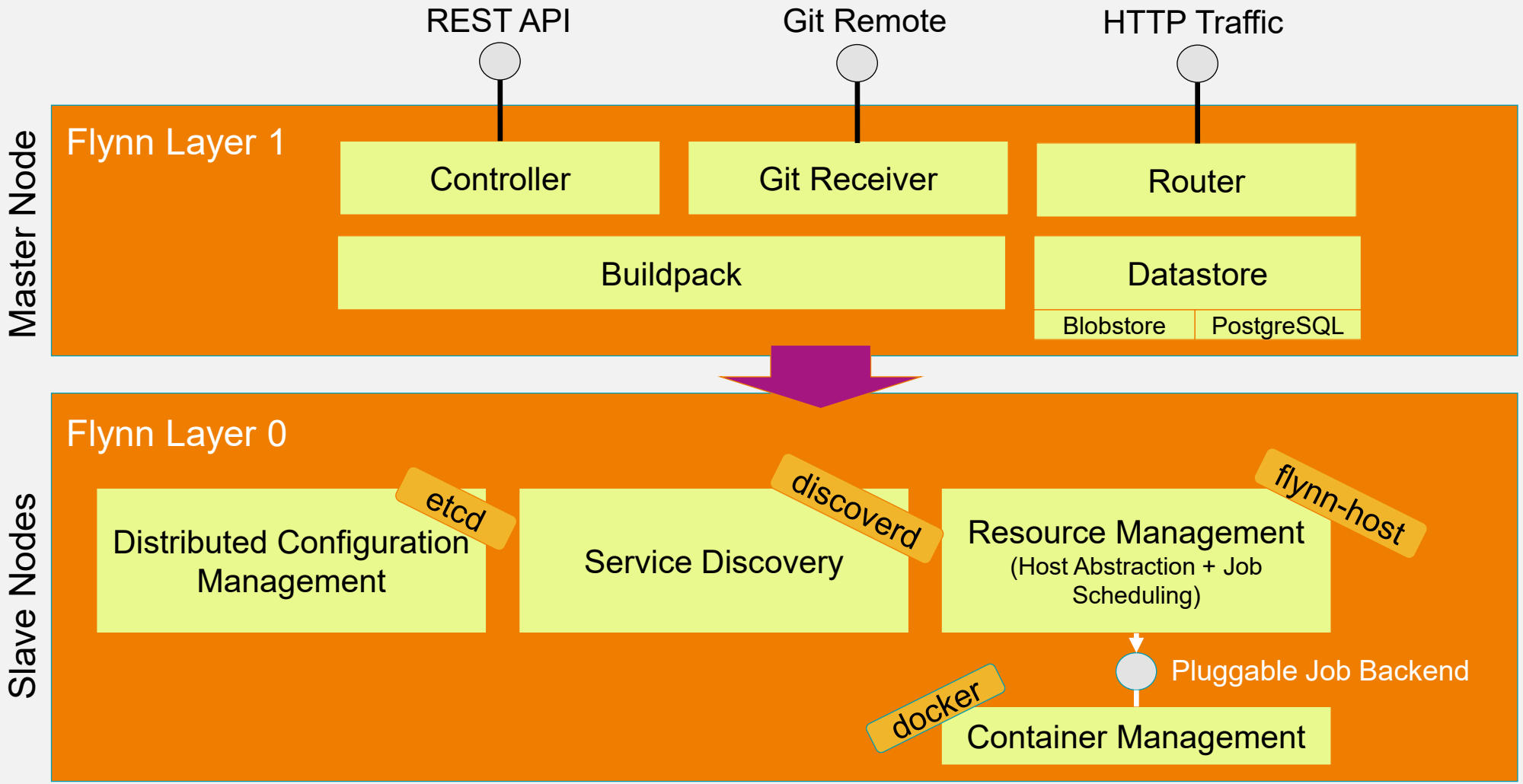
Private PaaS Clouds

Beispiel: Flynn

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



Die Architektur von Flynn

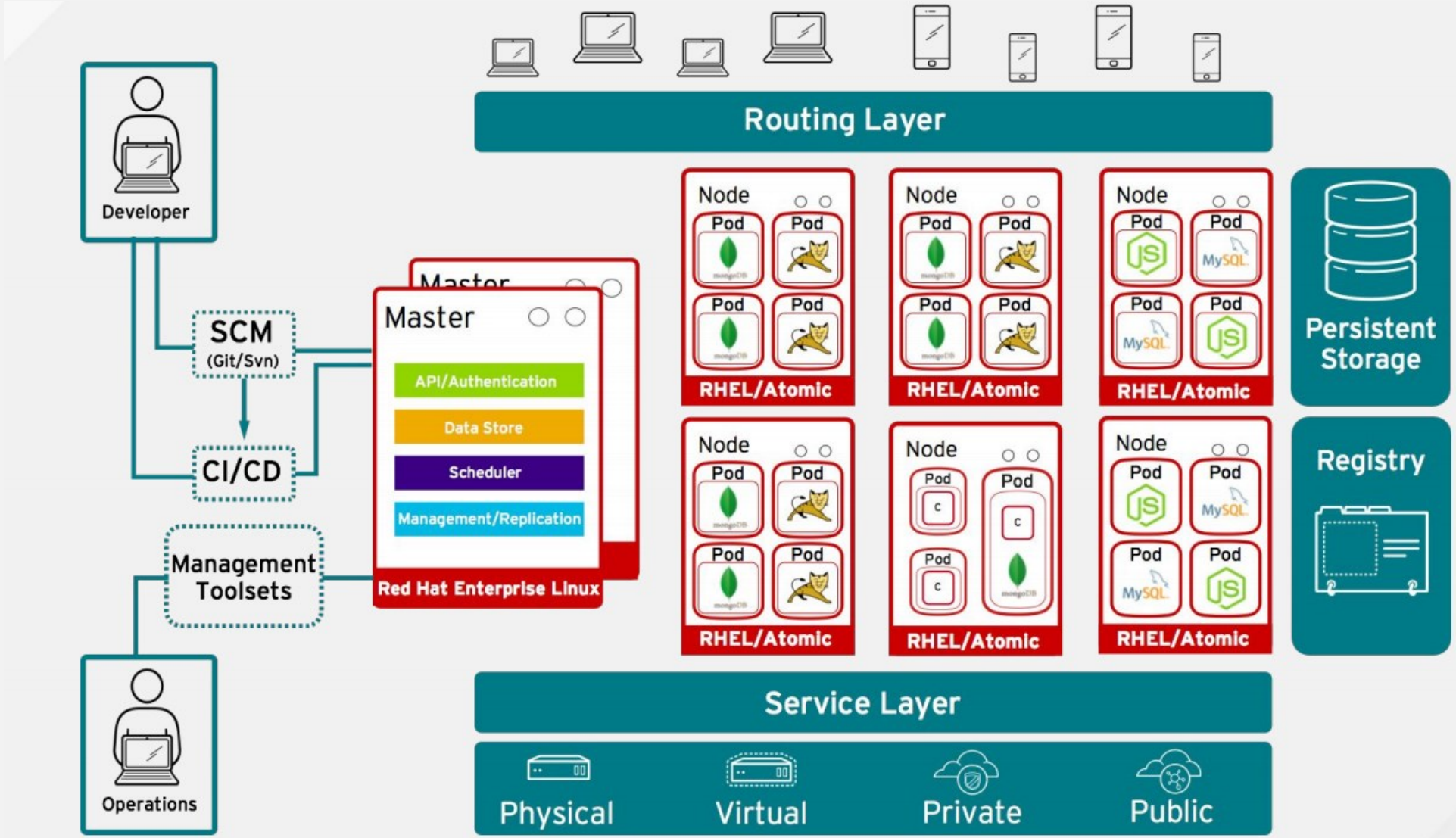


Alternative Private PaaS Clouds

- DEIS (http://deis.io, zu Flynn vergleichbarer Ansatz auf Basis von Docker)
- OpenShift (https://www.openshift.com, PaaS mit Schwerpunkt JEE von Red Hat)
- CloudFoundry (http://www.cloudfoundry.org, produktionserprobte PaaS von Pivotal mit breiter Unterstützung aus der Industrie)
- Stackato (http://www.activestate.com/stackato). Private PaaS von ActiveState (kommerziell).
- PaaSTA (https://github.com/yelp/paasta). Open-Source private PaaS auf Basis von Mesos und Marathon.
- VAMP (http://vamp.io). Leichtgewichtige Open-Source private PaaS ausgelegt auf Microservices. Läuft auf Basis Mesos oder Kubernetes.
- Apollo (https://github.com/Capgemini/Apollo). Open-Source private PaaS auf Basis Mesos von Capgemini.
- Mantl (http://mantl.io). Open-Source private PaaS auf Basis von Mesos von Cisco.

Beispiel: OpenShift.

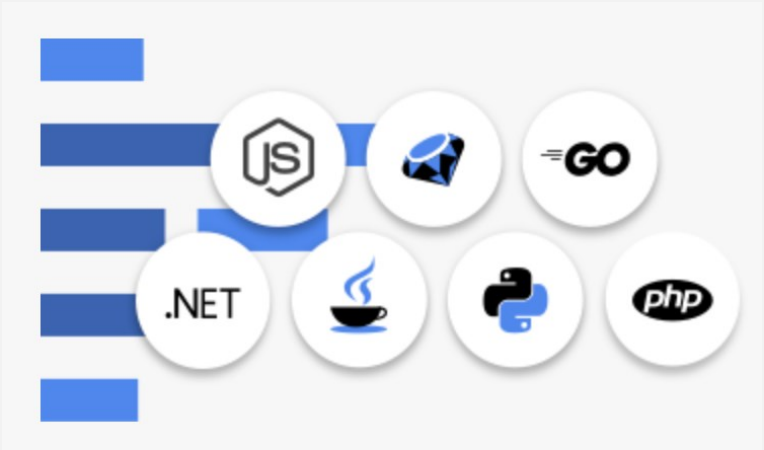
https://www.redhat.com/cms/managed-files/OpenShift.



Public PaaS Clouds

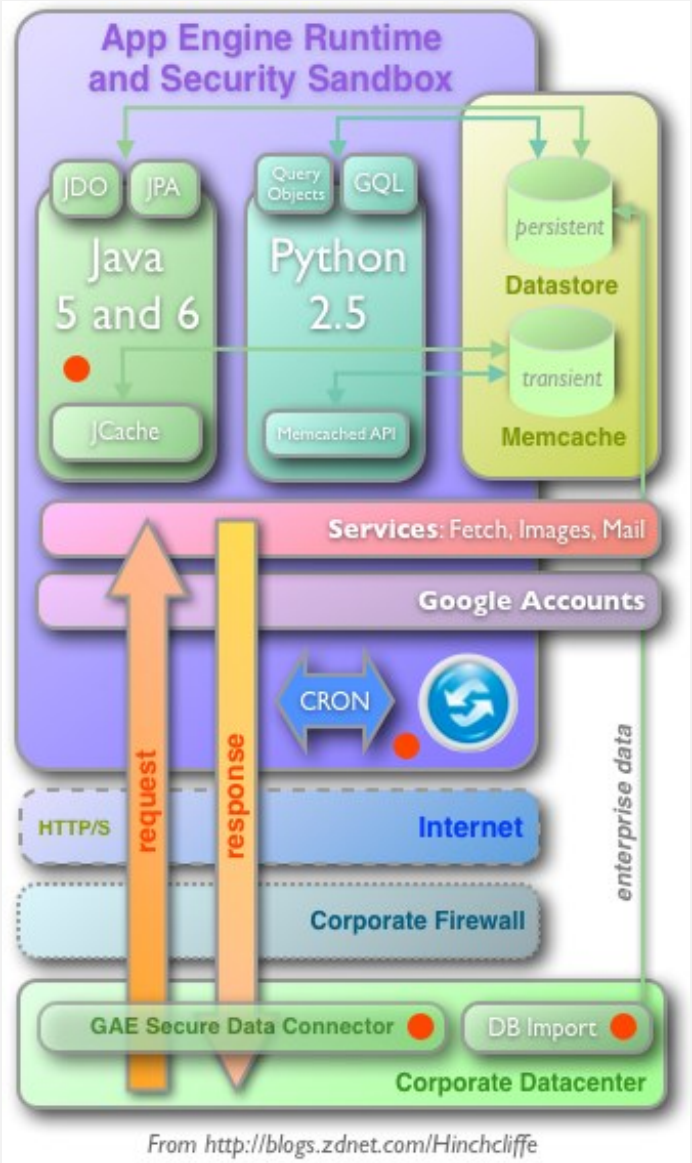
Die Google App Engine

- Die Google App Engine (GAE) ist das PaaS-Angebot von Google.
- Anwendungen laufen innerhalb der Google Infrastruktur.
- Der Betrieb der Anwendungen ist innerhalb bestimmter Quoten kostenfrei. Danach fallen Kosten u.A. auf Basis von Service-Aufrufen, Storage-Volumen und real genutzten CPU-Sekunden an.
- Unterstützte Sprachen:



- Integrationen in alle gängigen IDEs stehen zur Verfügung (Eclipse, intelliJ, Netbeans).

Die Google App Engine im Überblick .



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

Ausgewählte GAE Services (1/2)

Datastore

- Persistenter Speicher, realisiert als Key/Value-Datenbank.
- Transaktionen sind atomar. Schreibvorgänge sind stark konsistent. Abfragen sind eventuell konsistent.
- Definition, Abfrage und Manipulation von Daten erfolgt über eine eigene Sprache, die GQL (Google Query Language, nah an SQL).
- Als High-Level API sind die JDO und JPA APIs verfügbar. Diese sind im Rahmen von Java/JEE standardisiert. Die API wird durch das DataNucleus-Framework implementiert.

Memcache

- Hochperformanter temporärer Datenspeicher im Hauptspeicher (In-Memory Data-Grid).
- Jeder Eintrag wird mit einem eindeutigen Schlüssel abgelegt.
- Jeder Eintrag ist auf 1 MB beschränkt.
- Es wird eine Verfallszeit in Sekunden angeben, wann der Eintrag aus dem Memcache entfernt werden soll.
- Daten werden je nach Auslastung des Memcache auch bereits früher verdrängt.
- Als High-Level API ist die JCache API verfügbar.

Ausgewählte GAE Services (2/2)

URL Fetch

- Zugriff auf Inhalte im Internet.
- Unterstützte Methoden: GET, POST, PUT, DELETE und HEADER.
- Es darf auf Ports in den Bereichen 80-90, 440-450 und 1024-65535 zugegriffen werden.
- Anfragen und Antworten sind auf jeweils 1 MB beschränkt.

Users

- Anbindung eines Single-Sign-On Systems.
- Es werden Google Accounts und OpenID Accounts unterstützt.
- Als High-Level-API wird JAAS genutzt.

XMPP

- Nachrichten können an jedes XMPP-kompatibles Nachrichtensystem gesendet und von dieser werden.
- Jede Anwendung besitzt einen eindeutigen XMPP-Benutzernamen.

Alle APIs:

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

Einschränkungen der Google App Engine (1 / 2)

Eine GAE-Applikation läuft in einer Sandbox, die das Verhalten der Applikation einschränkt. Dies geschieht mit dem Ziel, die Verarbeitung effizient zu halten und die Infrastruktur im Auto-Scaling zu schützen.

Es dürfen nicht alle Klassen der Standardbibliothek genützt werden

- Keine eigenen Threads öffnen
- Kein Zugriff auf die Laufzeitumgebung und z.B. ihre Classloader
- http://code.google.com/p/googleappengine/wiki/WillItPlayInJava

Einschränkungen der Google App Engine (2 / 2)

Kommunikation mit anderen Web-Anwendungen oder Servern nur über URL Fetch, XMPP oder Email

- Anfragen und Antworten dürfen maximal 1MB groß sein
- Web-Hooks als allgemeines Architekturmittel für eingehende Kommunikation. Angestoßen bei Ereignissen (Warmup), Messages oder Cron-gesteuert.

Alle Requests an eine GAE-Anwendung werden nach 60 Sekunden beendet

Diverse Einschränkungen zu Datenvolumina und Anzahl von Service-Aufrufen