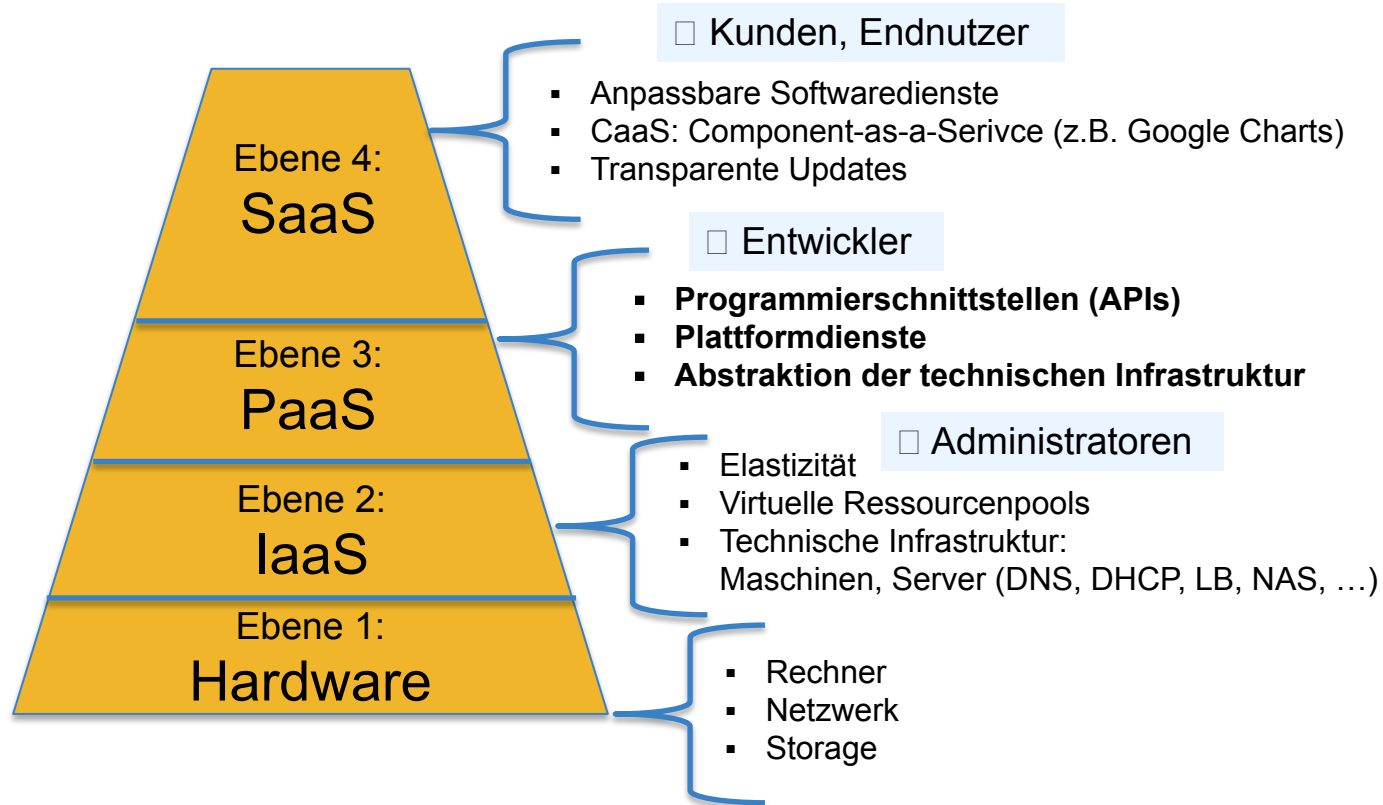


Cloud Computing Platform as a Service

Grundlagen einer PaaS Cloud

Erinnerung: PaaS im Schichtenmodell

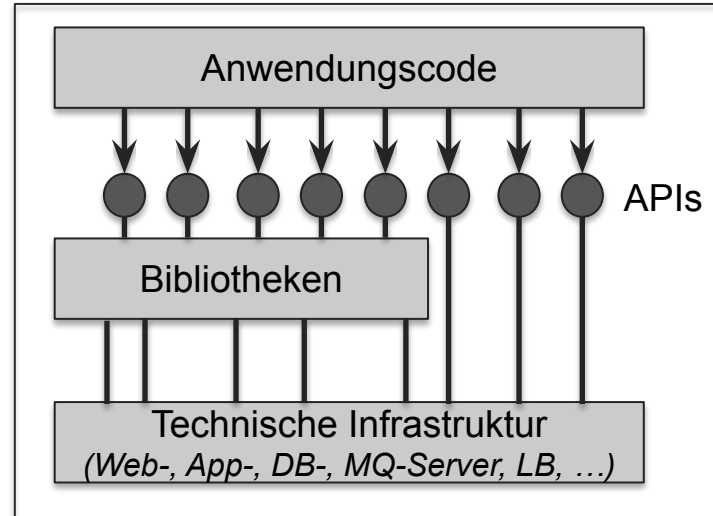


Ein Problem mit IaaS: Sie müssen Anwendungen aufwändig von Hand verdrahten.



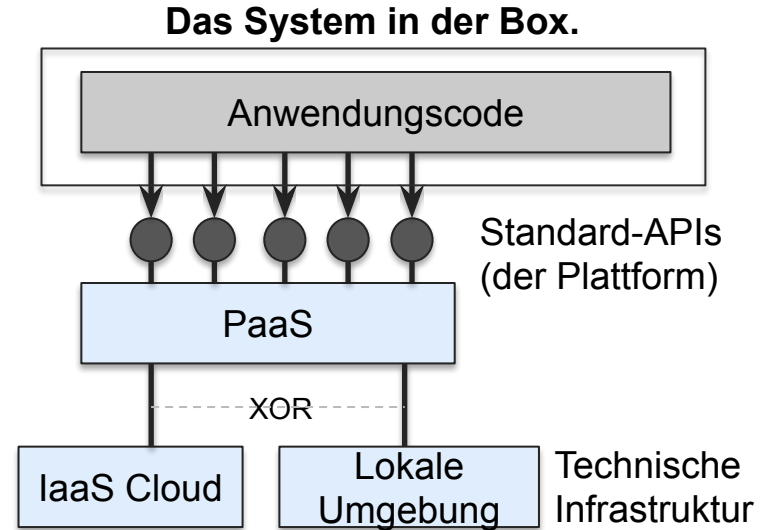
“Stovepipe
Architecture.”

Das System: Mühevoll verdrahtet.



Lösung: Plattform-as-a-Service bietet eine Entwicklungs- und Betriebsplattform mit vorgegebenen APIs.

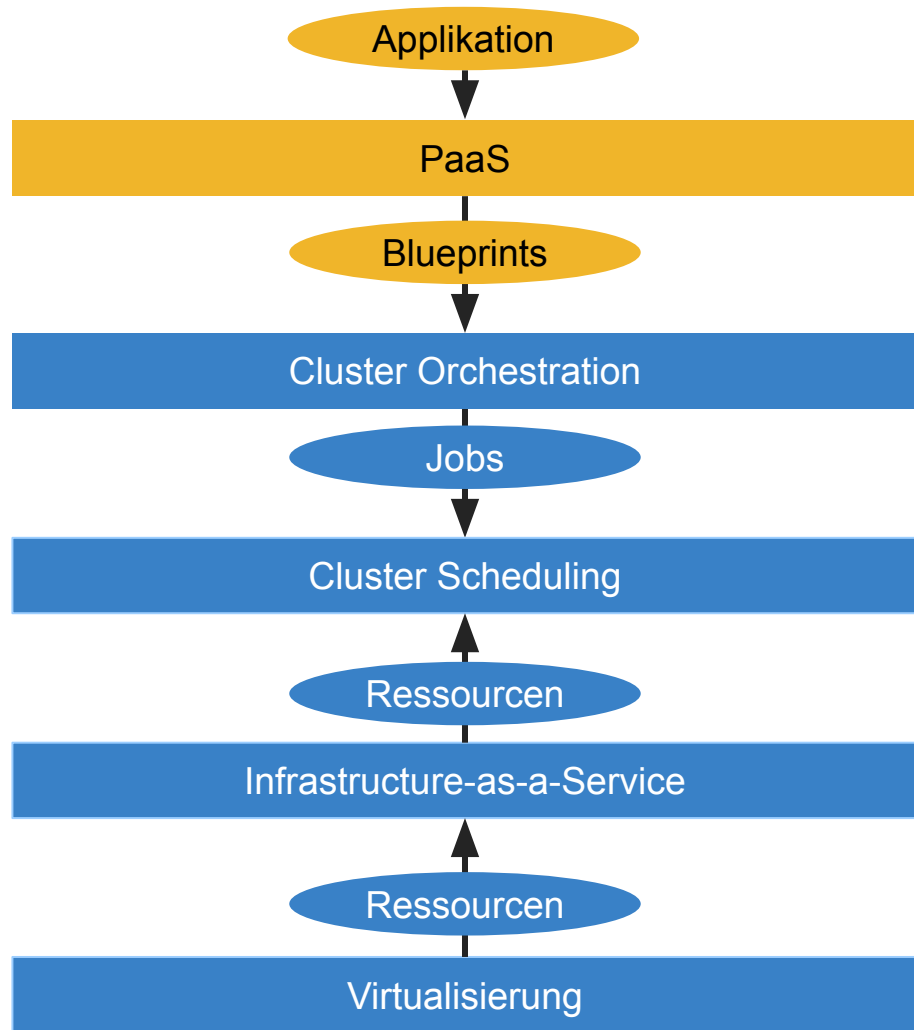
- Die Anwendung wird per Applikationspaket oder als Quellcode deployed. Es ist kein Image mit Technischer Infrastruktur notwendig.
- Die Anwendung sieht nur Programmier- oder Zugriffsschnittstellen seiner Laufzeitumgebung.
„Engine and Operating System should not matter....“.
- Es erfolgt eine automatische Skalierung der Anwendung.
- 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**.

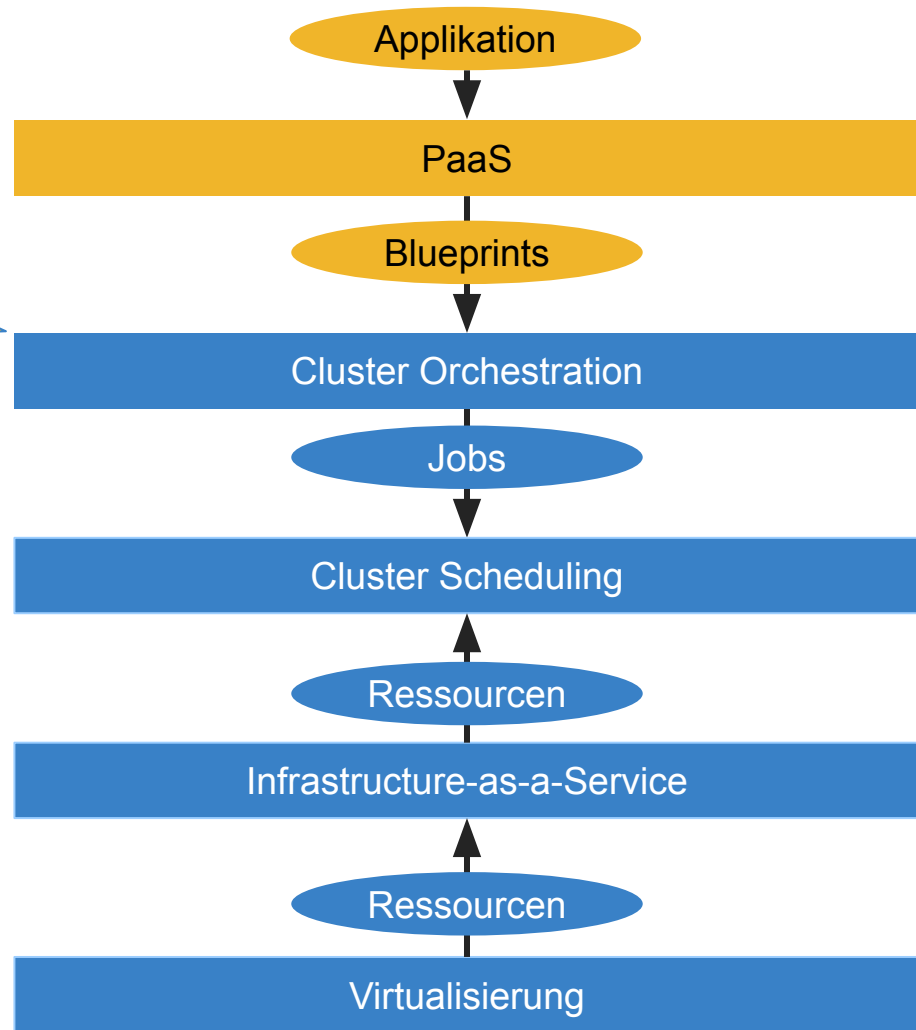
Das Big Picture



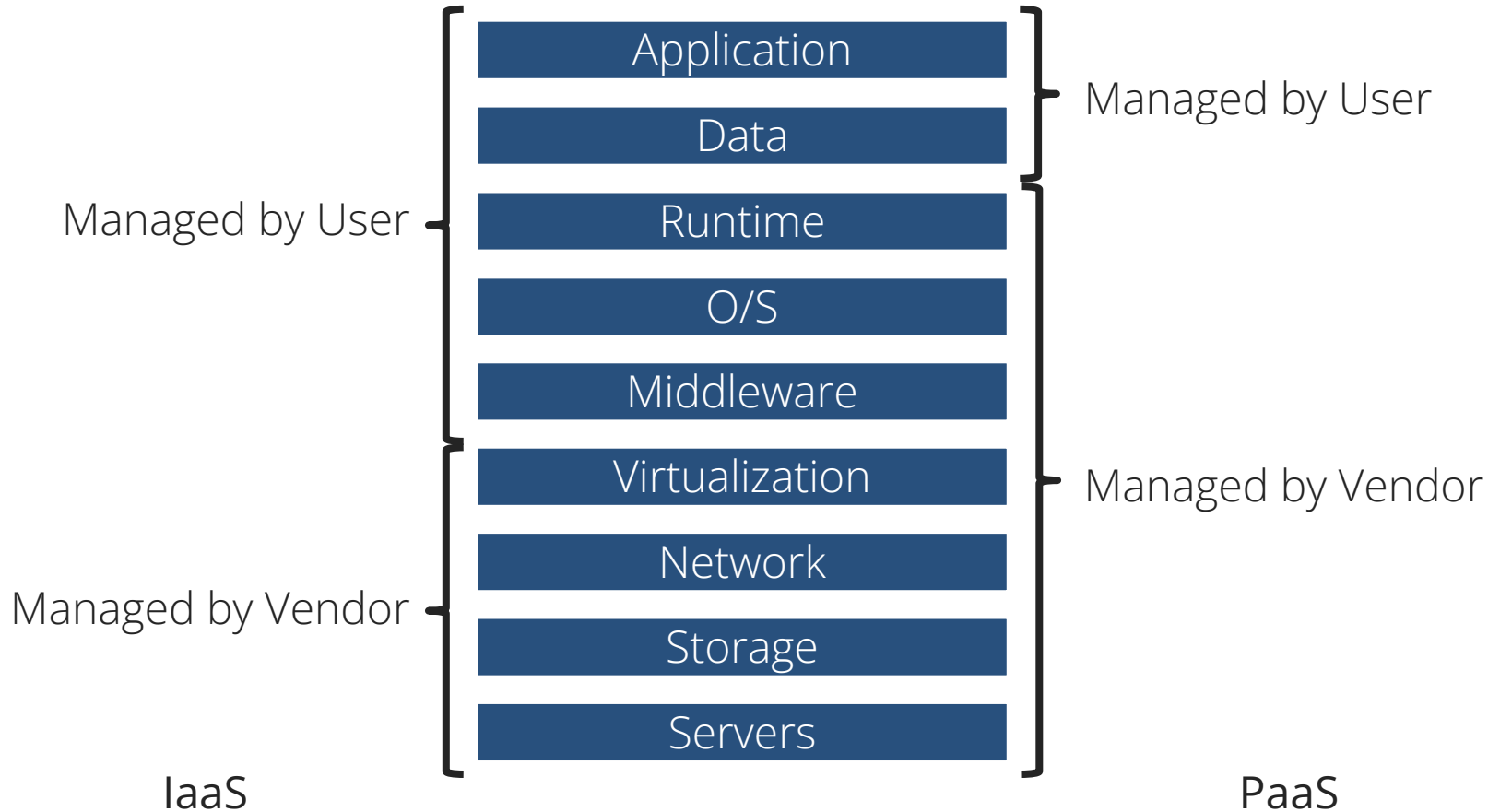
Das Big Picture

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

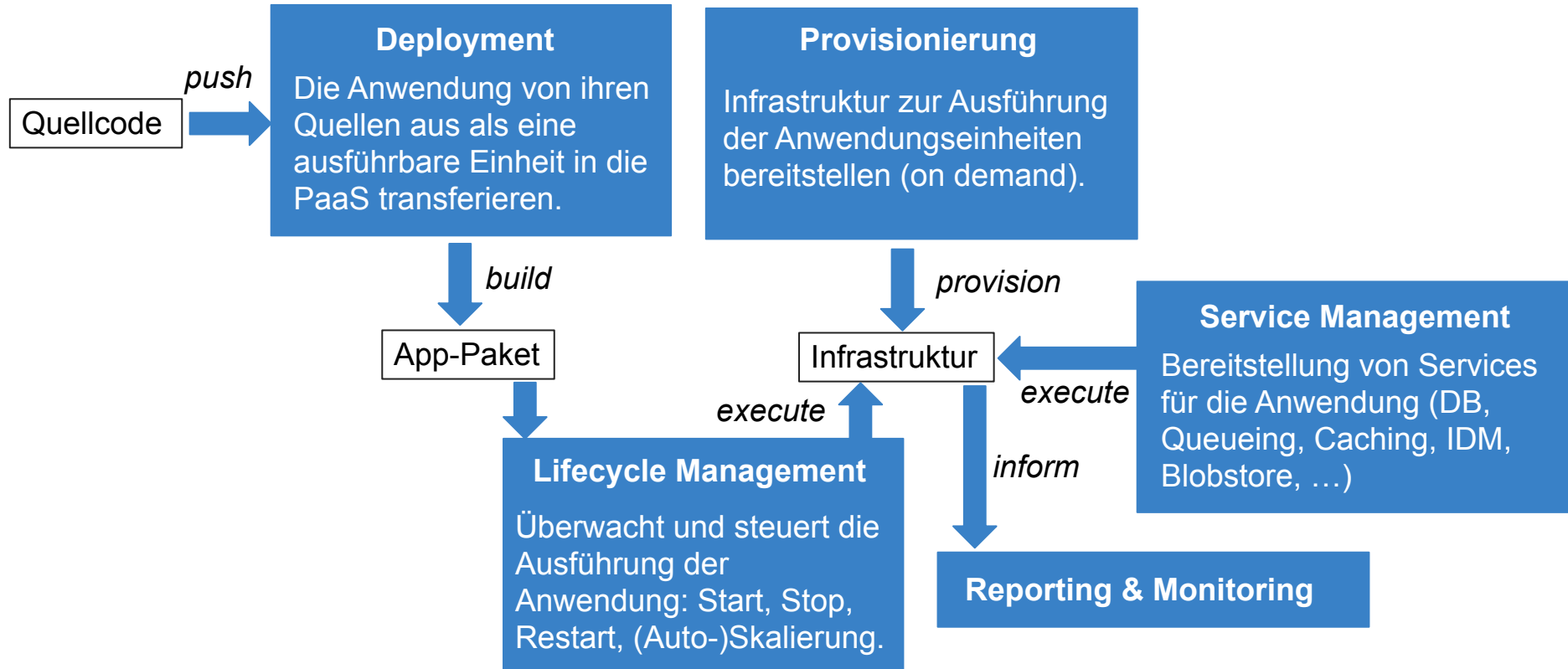
- Wiederverwendung von Infrastruktur / APIs
- Komfort-Dienste für Entwickler



IaaS vs. PaaS



Die funktionalen Bausteine einer PaaS Cloud

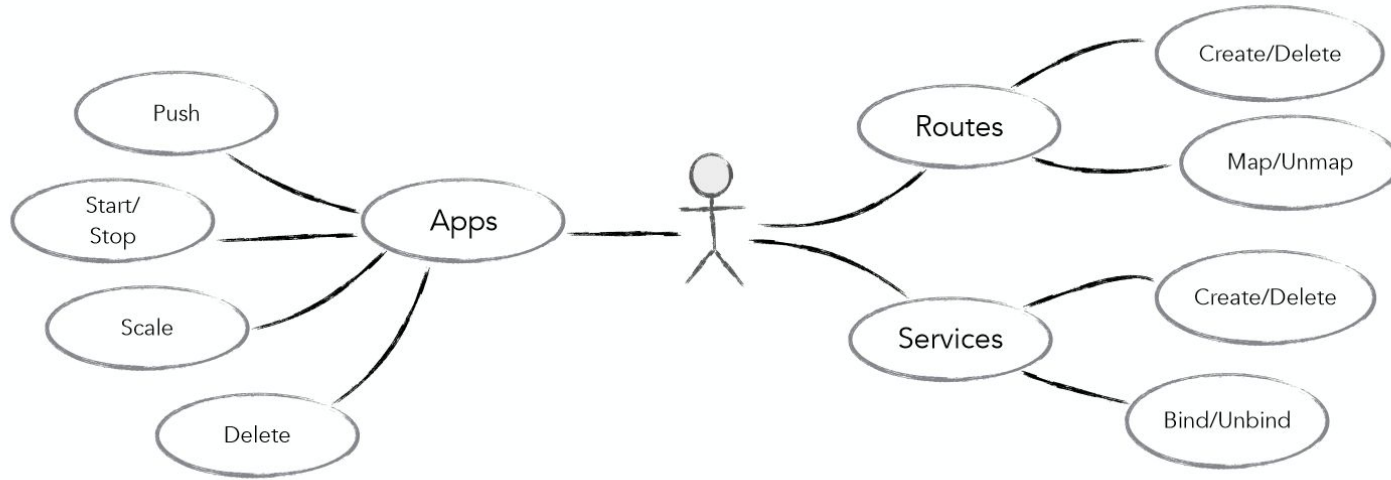


← = Datenfluss

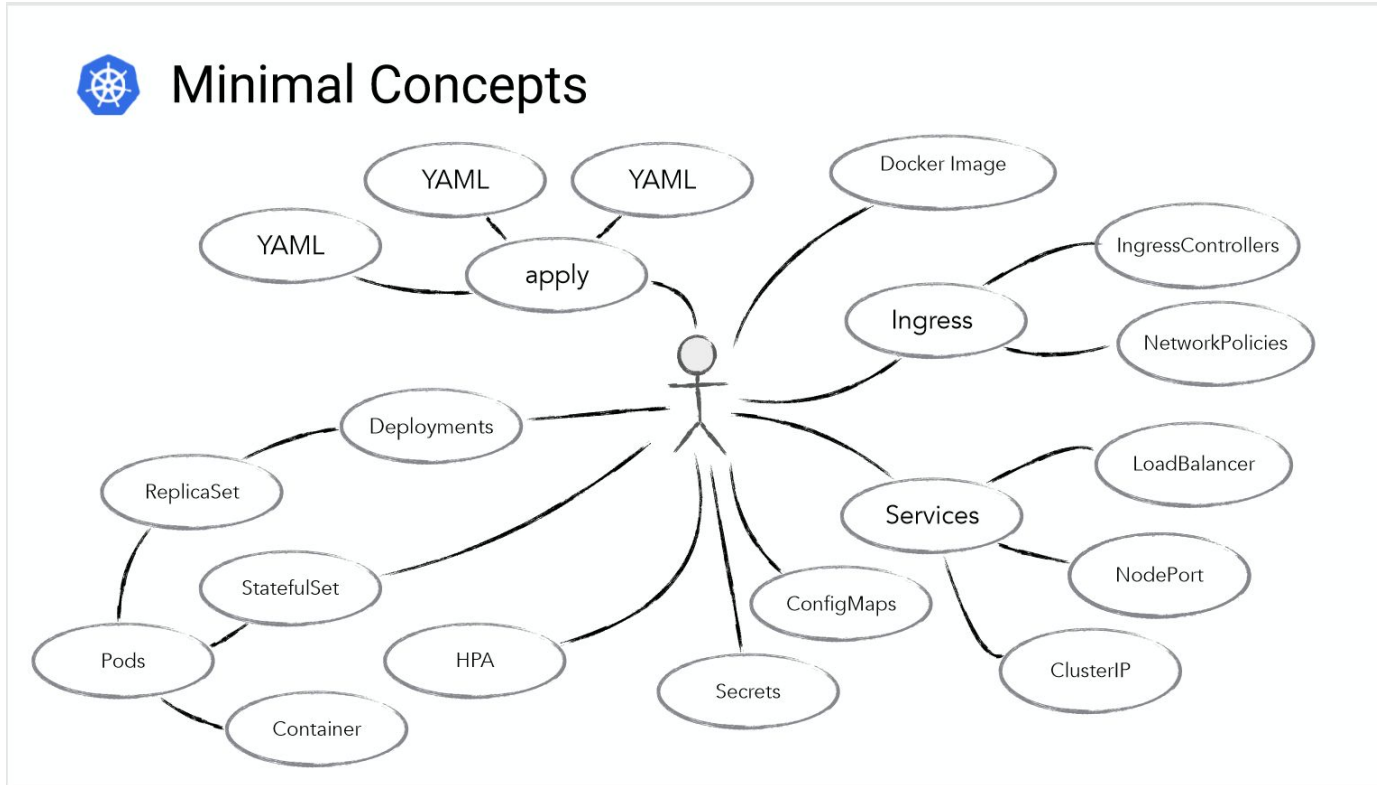
Beispiel: Cloud Foundry



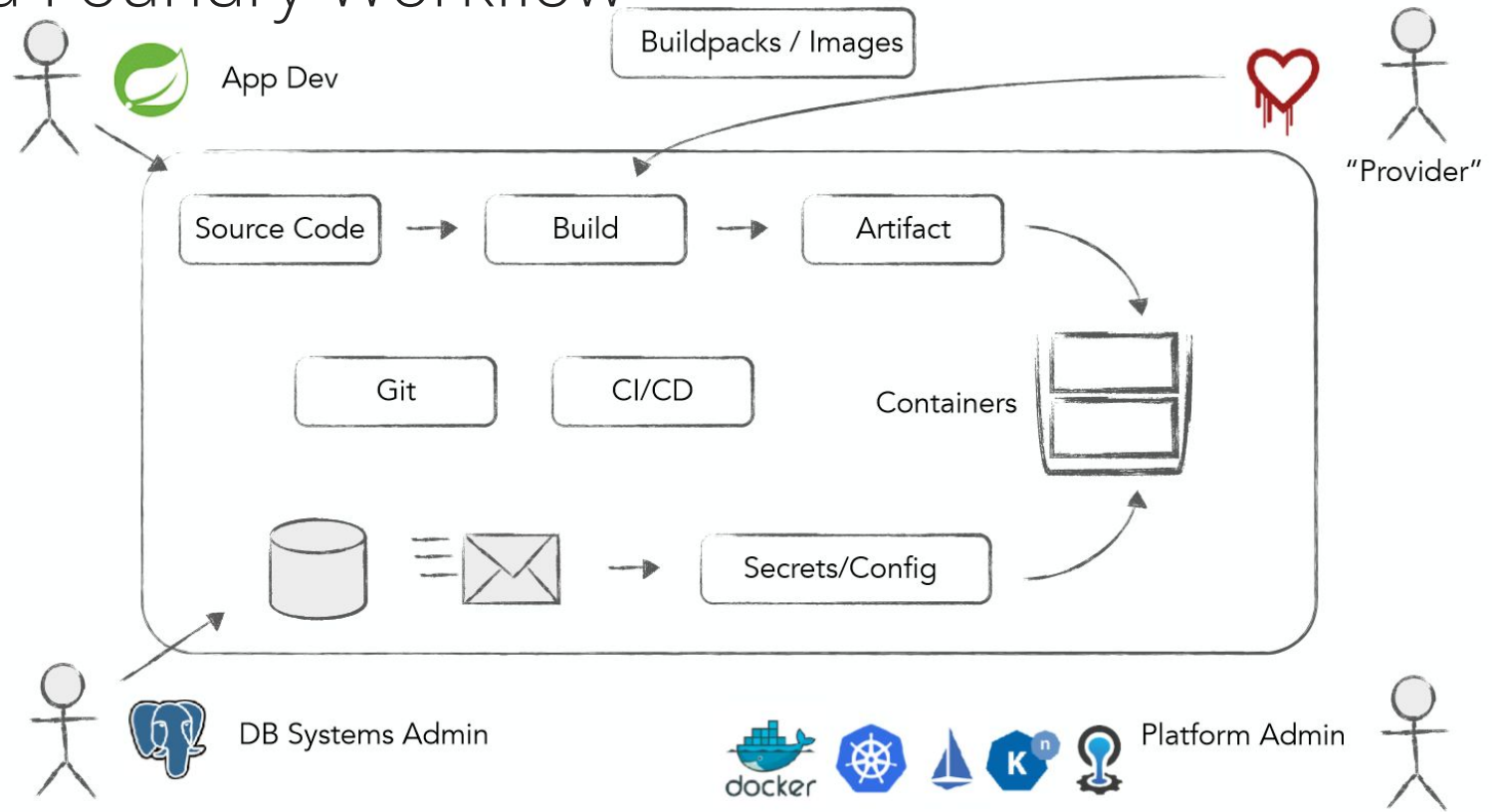
Minimal Concepts



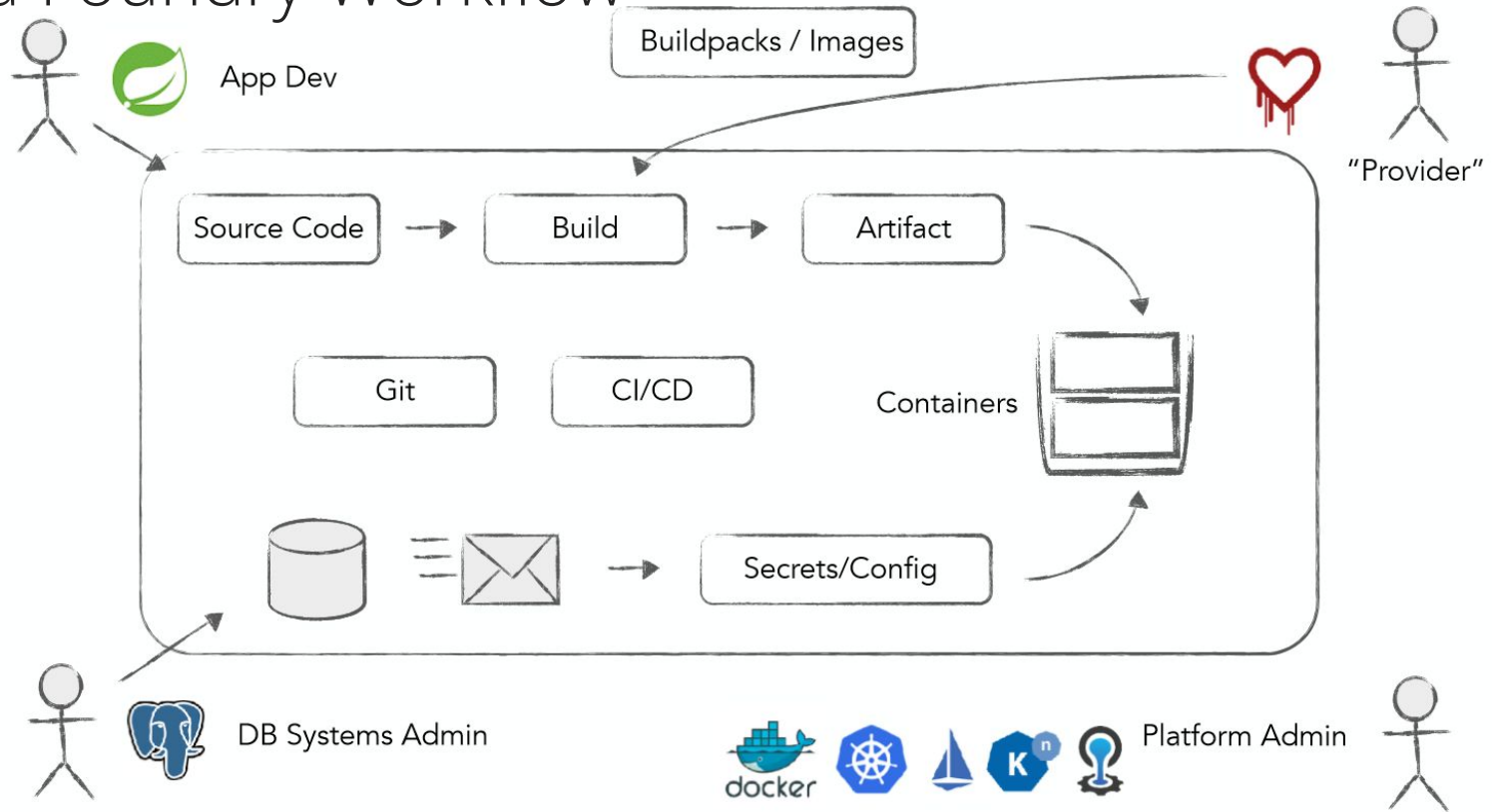
Kontrast: Kubernetes



Cloud Foundry Workflow

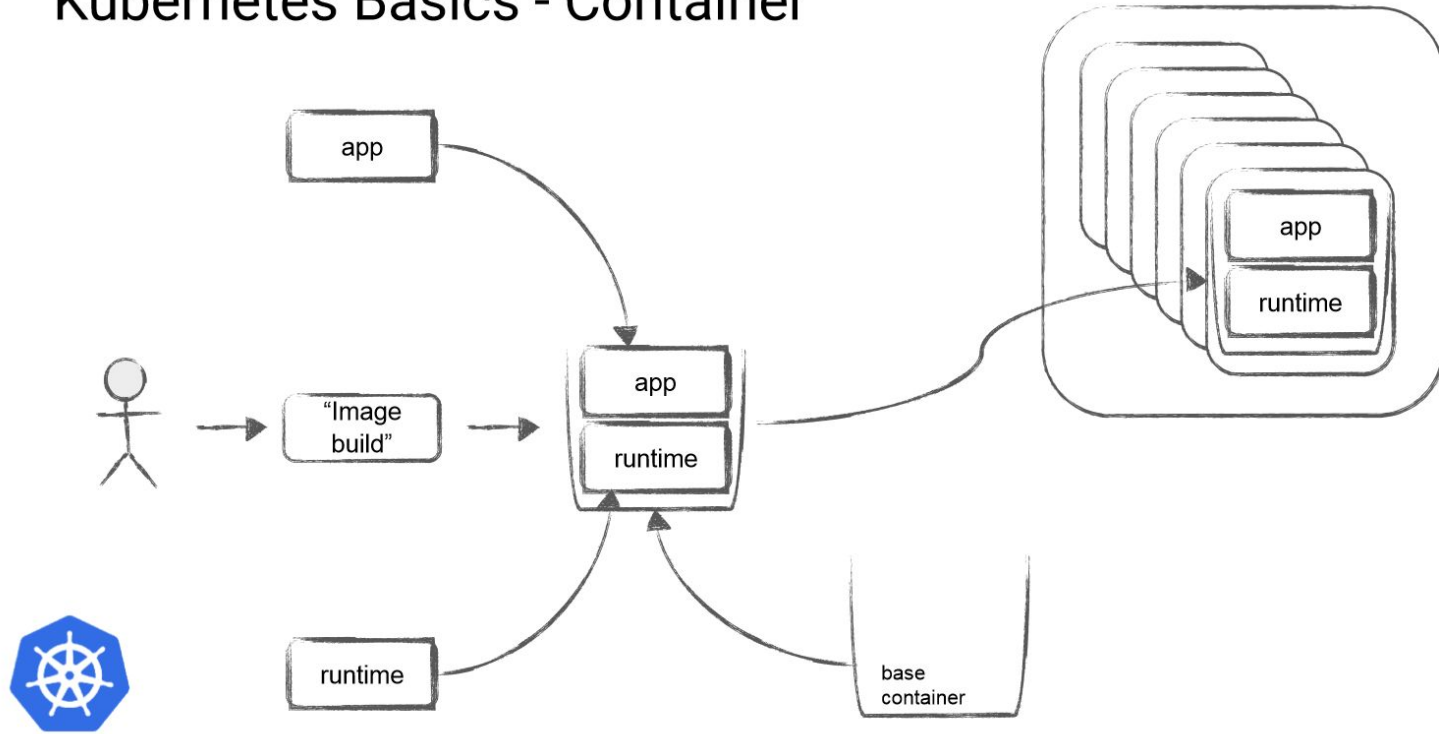


Cloud Foundry Workflow



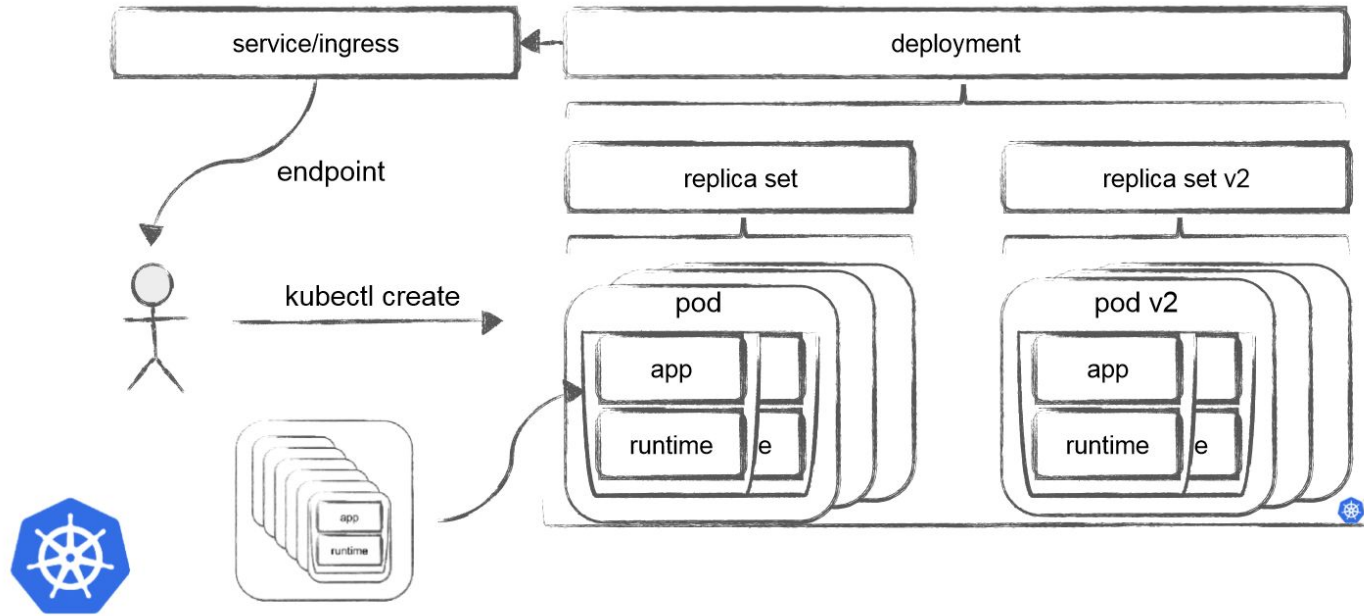
In Kubernetes

Kubernetes Basics - Container



Kubernetes Workflow

Kubernetes Basics - Orchestration



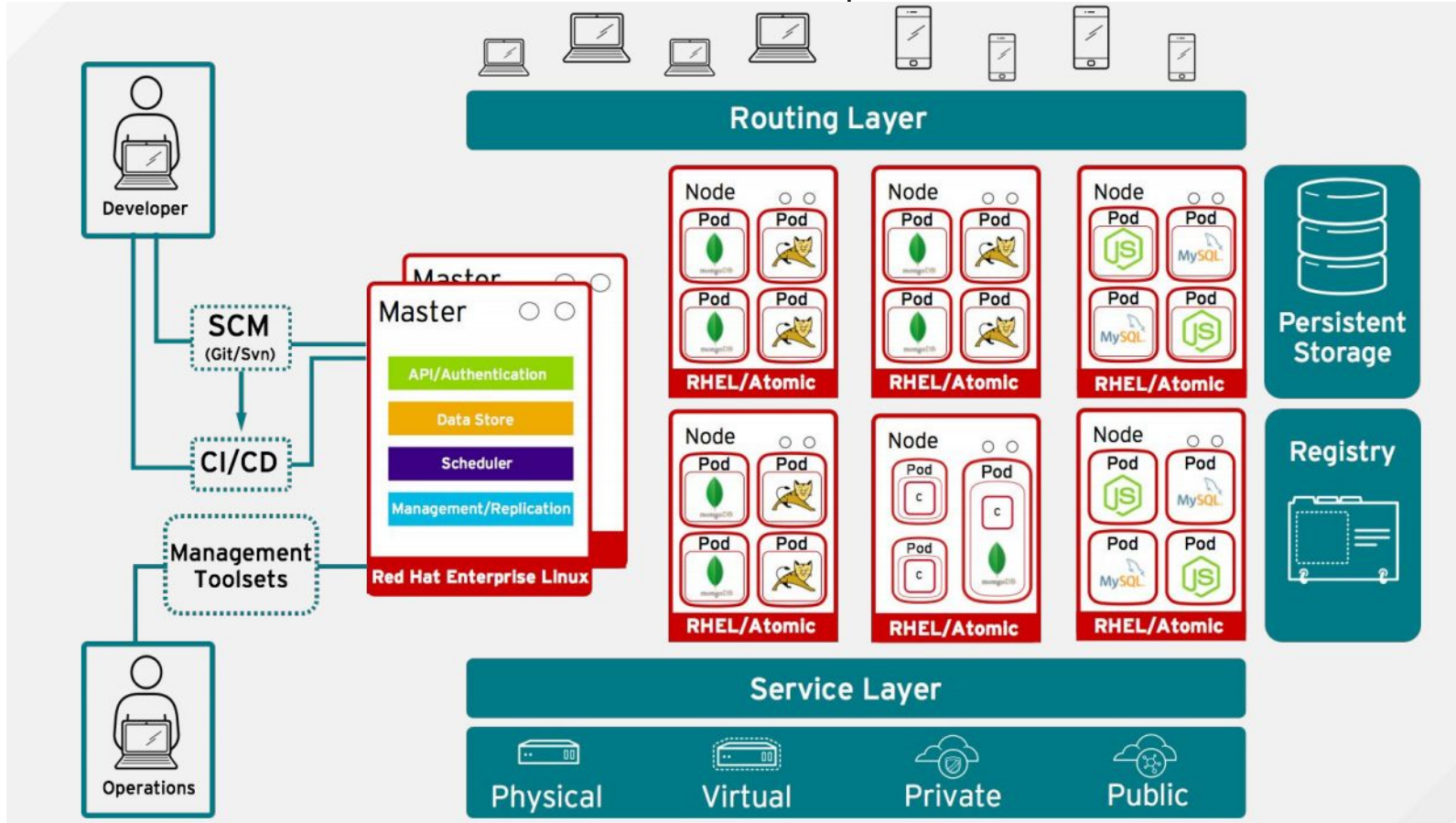


Private PaaS: Kurze Übersicht

Private PaaS - Übersicht

- ~~Flynn~~ (<https://github.com/flynn/flynn>, basiert auf Docker, abandoned)
- ~~DEIS~~ Hefhy (<https://github.com/teamhephy/workflow>, vergleichbar, originales Team wurde von Microsoft acquired)
- 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 HP Helion (Private PaaS von ActiveState, kommerziell, eingestellt).
- PaaSSTA (<https://github.com/yelp/paasta>). Open-Source, auf Basis von Mesos oder Kubernetes und Marathon, von Yelp
- ~~VAMP~~ (<http://vamp.io>). Leichtgewichtige Open-Source private PaaS ausgelegt auf Microservices. Läuft auf Basis Mesos oder Kubernetes. (abandoned)
- ~~Apollo~~ (<https://github.com/Capgemini/Apollo>). Open-Source private PaaS auf Basis Mesos von Capgemini (abandoned)
- ~~Mantl~~ (<http://mantl.io>). Open-Source private PaaS auf Basis von Mesos von Cisco (abandoned)
- Dokku (<https://github.com/dokku/dokku>, Docker-powered Mini-Heroku (in Bash))
- ...

Private PaaS - Architektur von Openshift

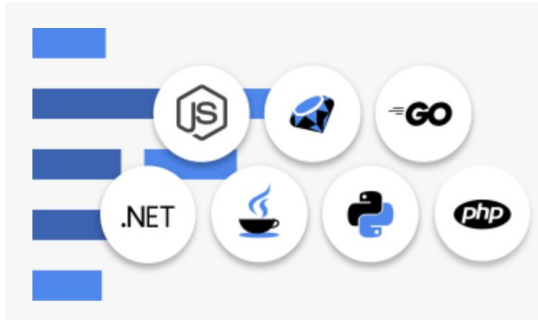




PaaS in der Public Cloud: Beispiel Google App Engine

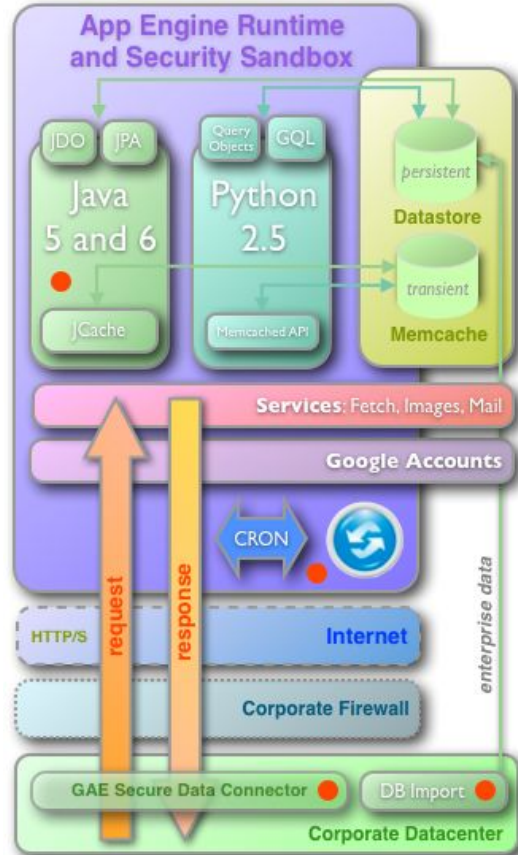
Google App Engine

- 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).

Google App Engine im Überblick



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

Google App Engine API (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 angegeben, 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.

Google App Engine API (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 diesem empfangen werden.
- Jede Anwendung besitzt einen eindeutigen XMPP-Benutzernamen.

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



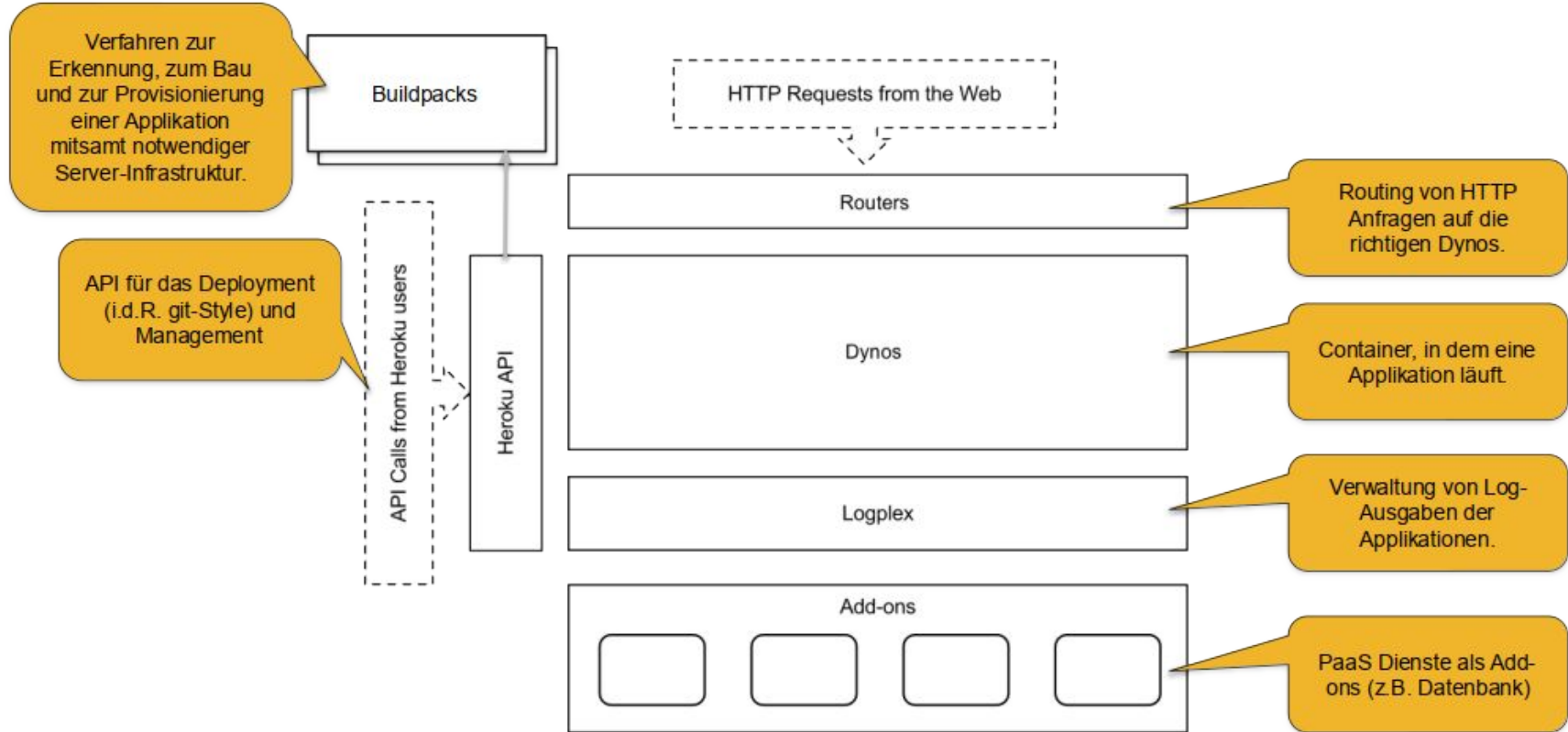
PaaS in der Praxis: Heroku

Heroku

- Eine der ersten PaaS-Plattformen
- Erstes Auftreten 2007
- 2010 von Salesforce gekauft
- Initialer Fokus auf Ruby, mittlerweile wird auch Java, Node.js, Go, Scala, Clojure, Python und PHP unterstützt
- Komplettes kostenloses Starterangebot inklusive PostgreSQL
- GitOps-Ansatz: Deployments werden über Git gesteuert



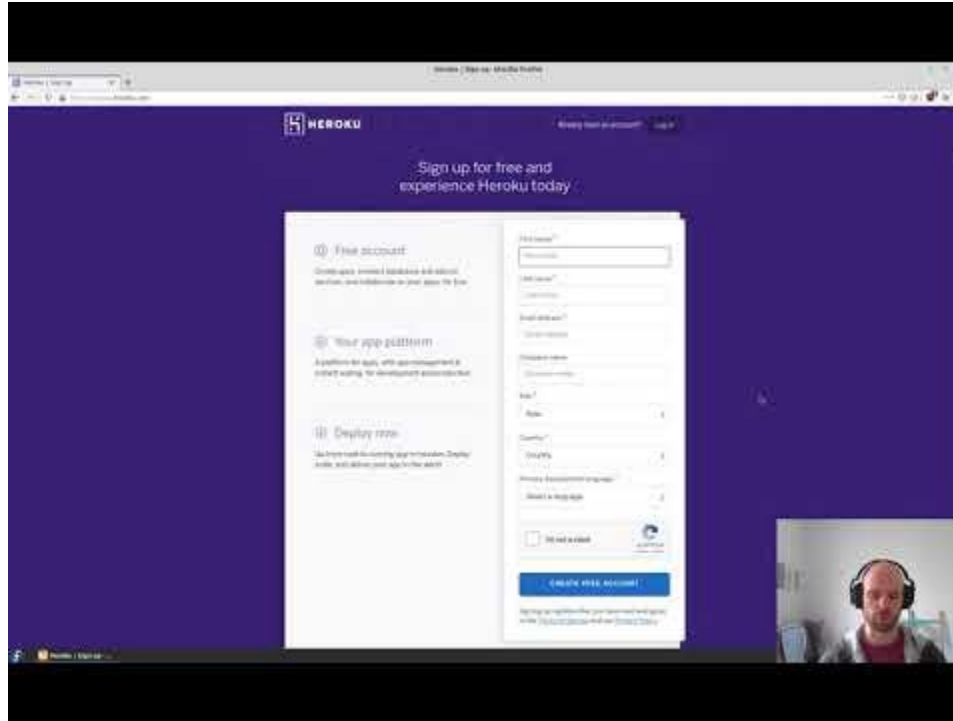
Architektur von PaaS - Beispiel Heroku



The background of the slide is a dark blue color. Overlaid on this background is a complex, abstract pattern of thin white lines and small white dots. These lines and dots form a network of interconnected geometric shapes, primarily triangles and polygons, creating a mesh-like or molecular structure that spans the entire width and height of the image. The pattern is more dense in some areas and more sparse in others, giving it a dynamic, organic feel.

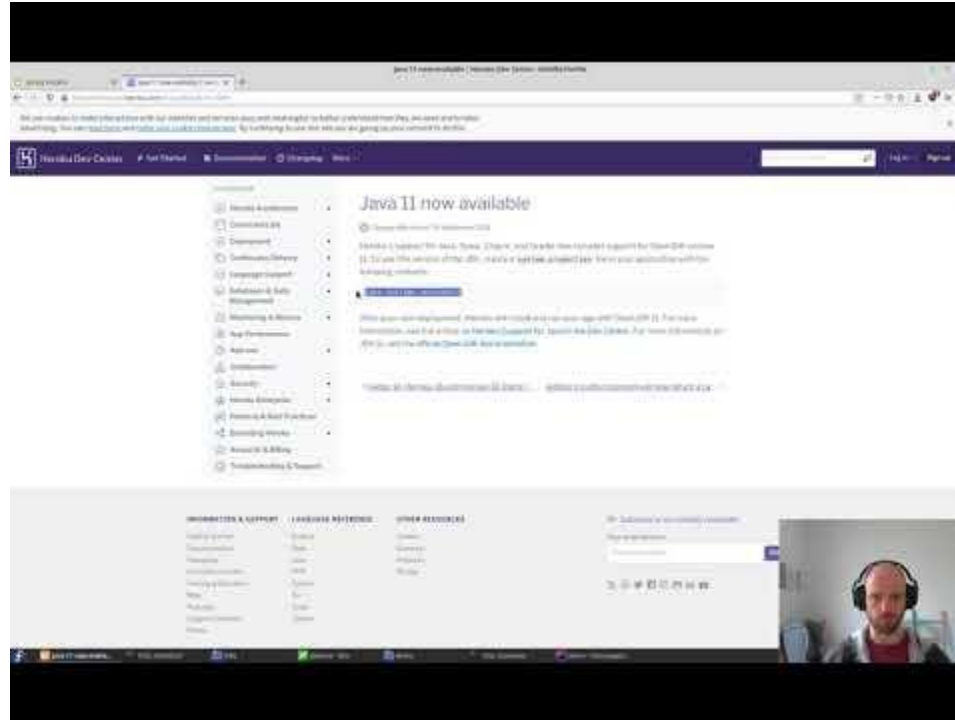
Heroku im Praktikum

Demo: Installation der Heroku CLI



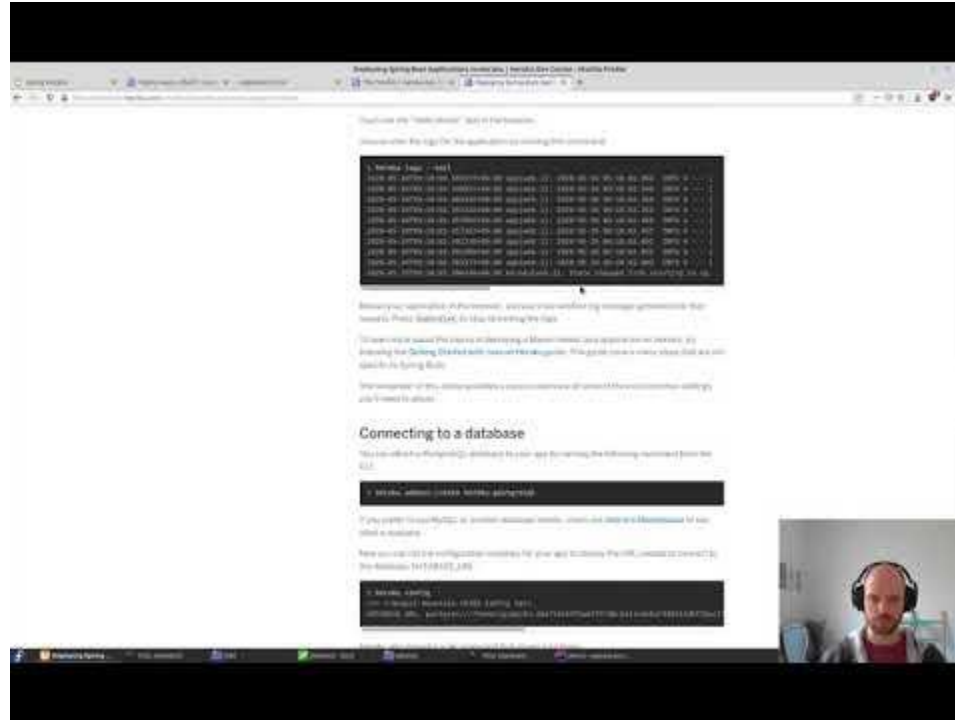
<https://www.youtube.com/watch?v=IPWTS1NTN78>

Demo: Hello World Anwendung



<https://www.youtube.com/watch?v=L8UNtUFaYTA>

Demo: Lokale Entwicklung



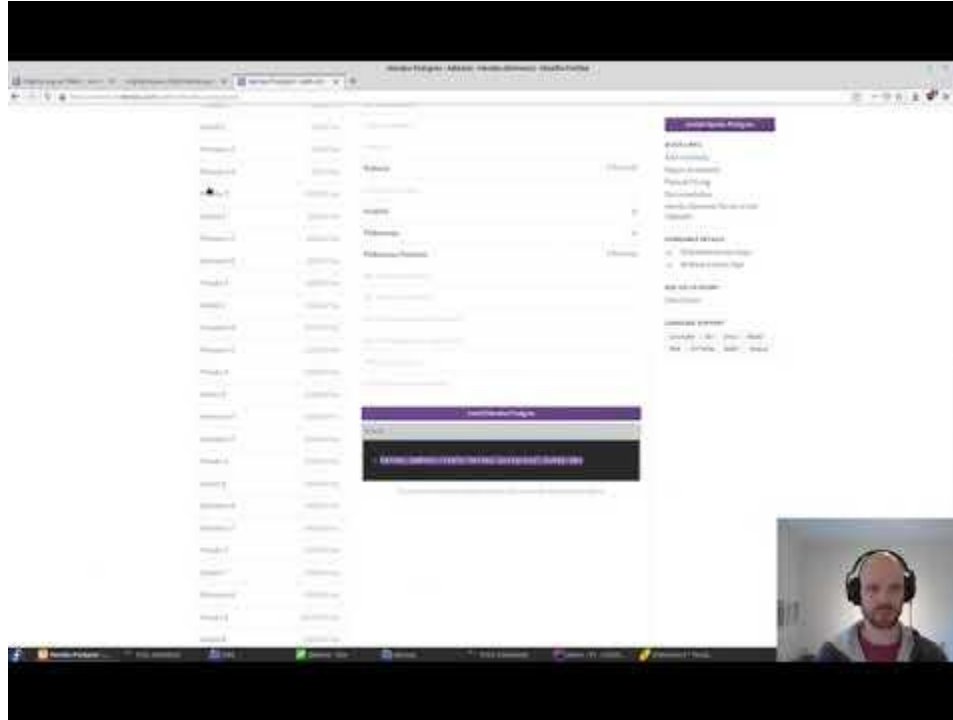
<https://www.youtube.com/watch?v=kbWZdqZduvE>

Demo: Anwendungskonfiguration



https://www.youtube.com/watch?v=VpXcB9gN_9Q

Demo: Datenbanken mit PostgreSQL



<https://www.youtube.com/watch?v=63sGKGhUx-w>