QAWARE SOFTWARE ENGINEERING

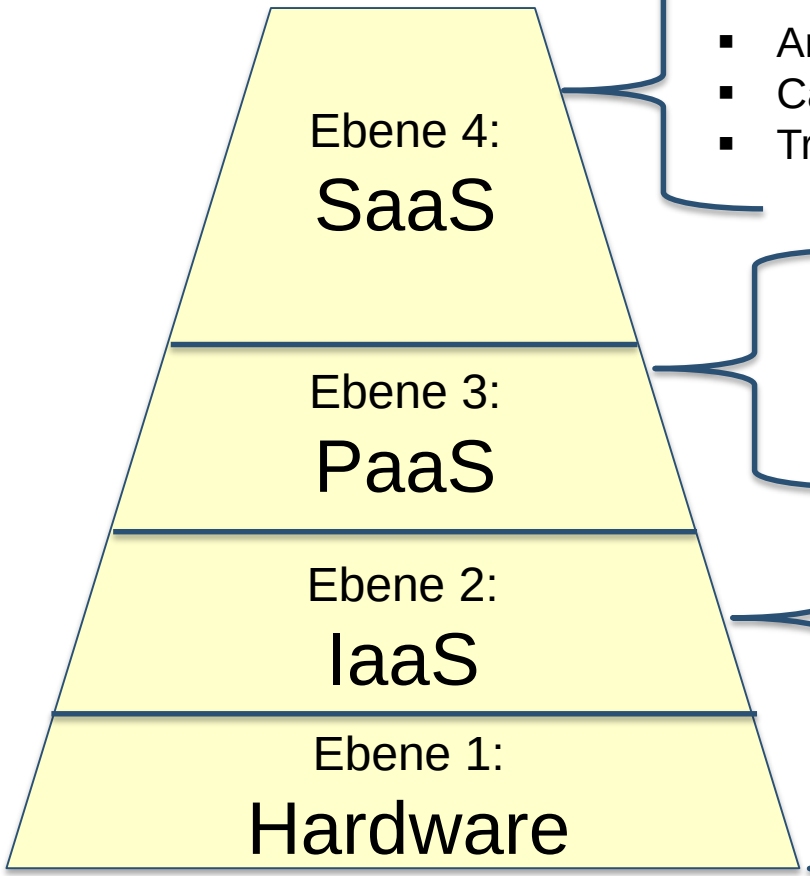
Cloud Computing

Kapitel 10: Platform-as-a-Service

Dr. Josef Adersberger

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 Update

➔ Entwickler

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

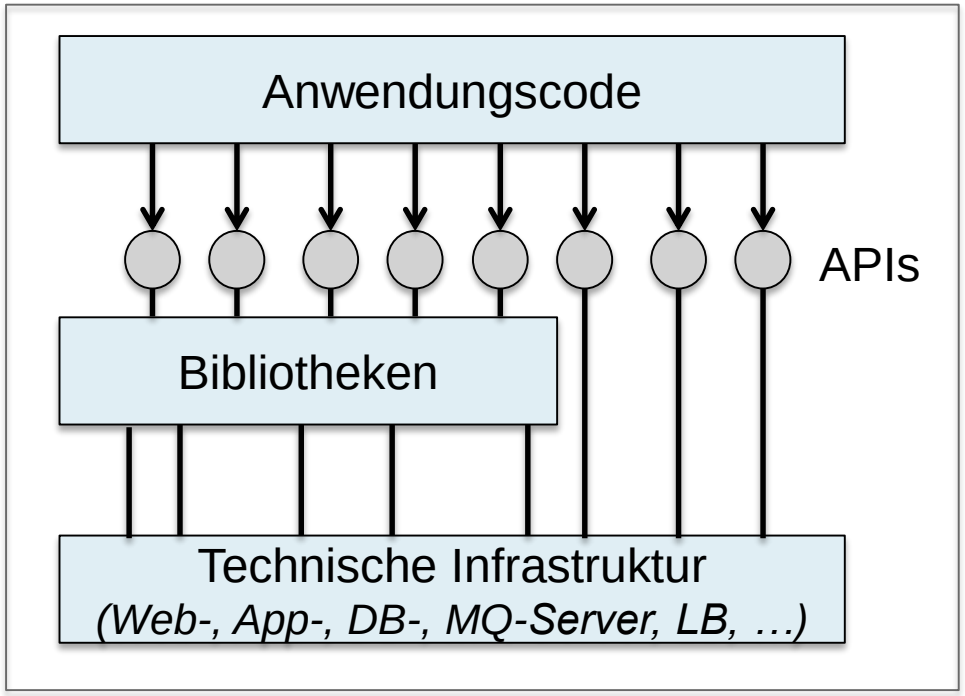
➔ Administratoren

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

- Rechner
- Netzwerk
- Storage

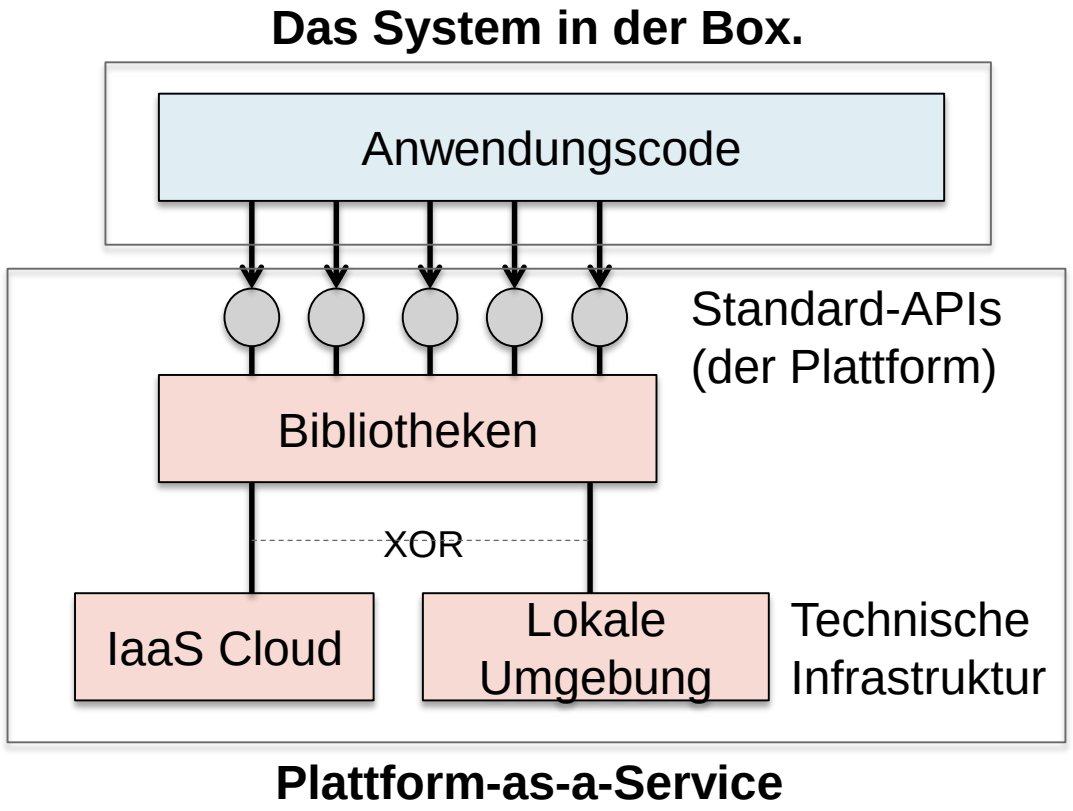
Das Problem: Stovepipe Architecture. Anwendungen aufwändig von Hand verdrahten.

Das System: Mühevoll verdrahtet.



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

- Die Anwendung sieht nur Programmierschnittstellen seiner Laufzeitumgebung.
  „Engine and Operating System should not matter….".
- Die Anwendung wird per Applikationspaket deployed. Es ist kein Image mit Technischer Infrastruktur notwendig.
- Es erfolgt eine automatische Skalierung der Anwendung.
- PaaS repräsentiert einen Vertrag zwischen Entwicklung und Betrieb.
- Die Programmierschnittstellen (APIs) stehen als Bibliotheken oder als (REST-) Webservices zur Verfügung.
- 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.

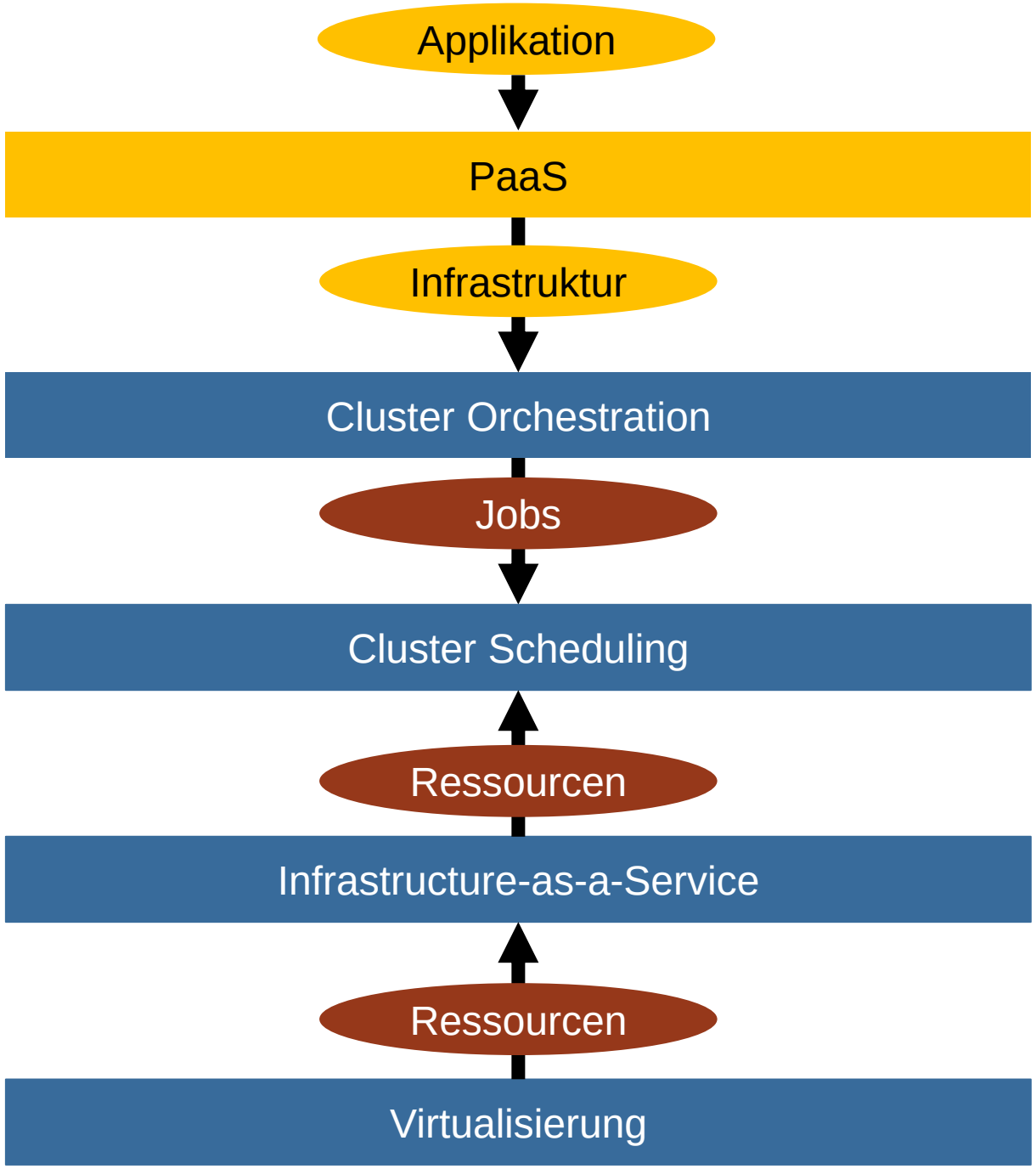


PaaS: Definitionen

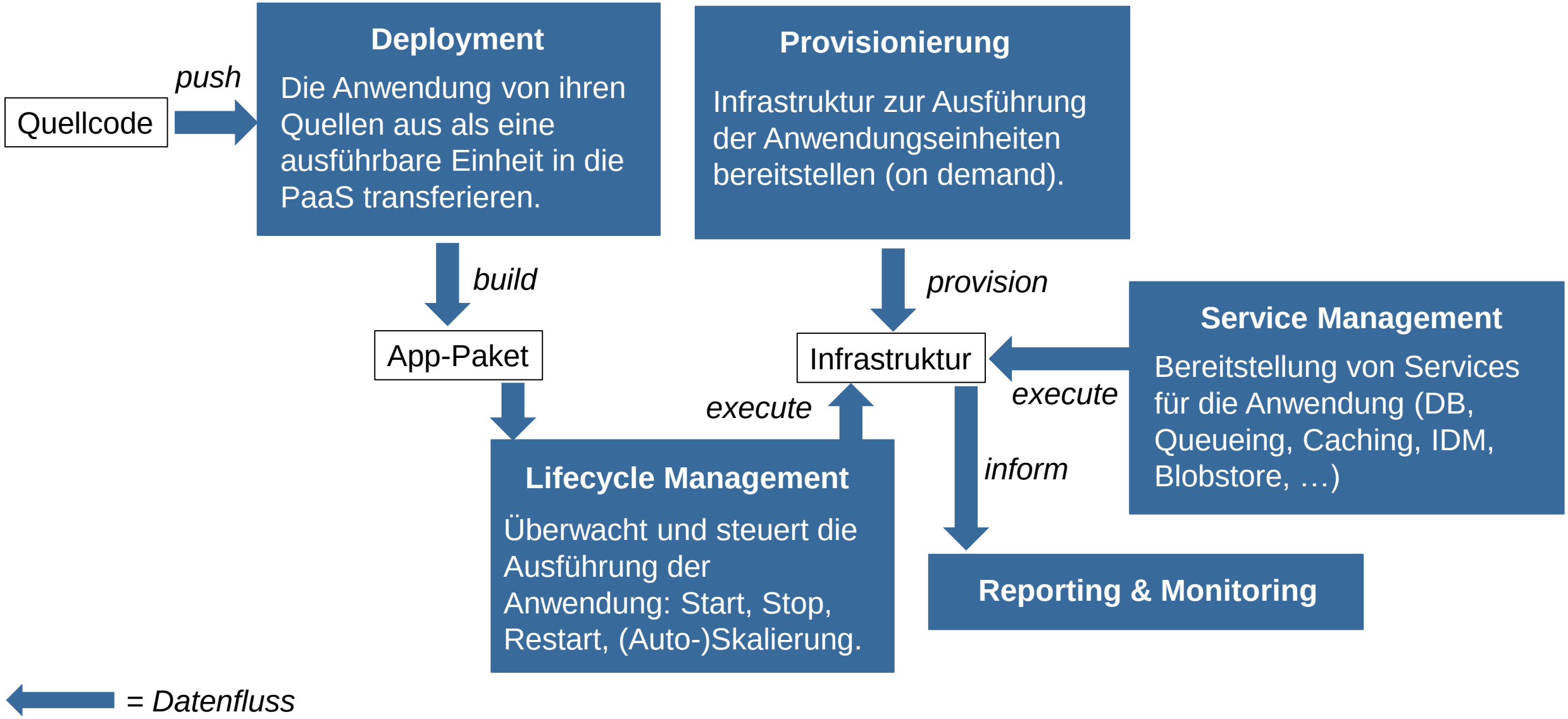
- NIST: The capability provided to the consumer is to **deploy onto the cloud infrastructure** consumer-created or acquired applications created **using programming languages, libraries, services, and tools supported by the provider**. **The consumer does not manage or control the underlying cloud infrastructure**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

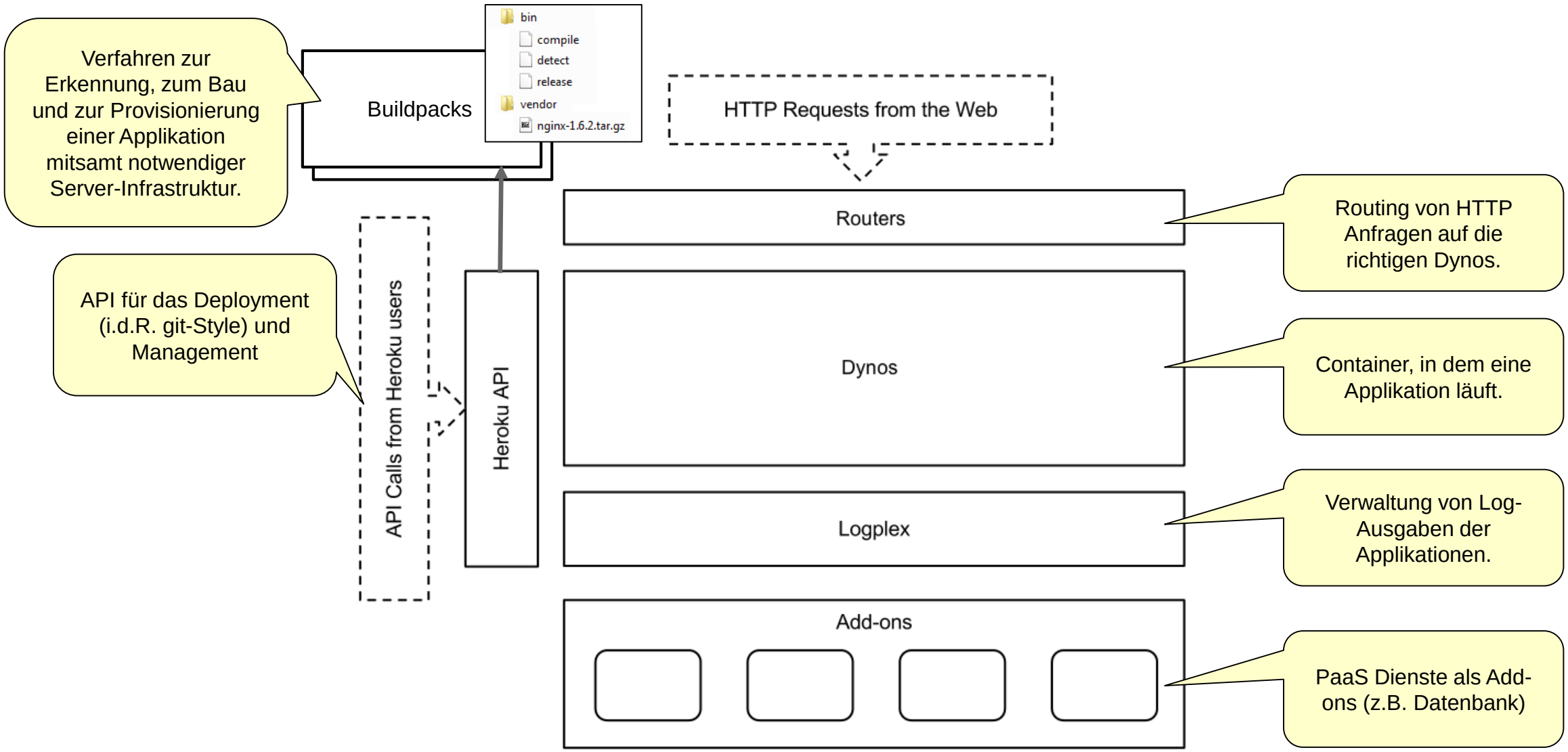
Das Big Picture



Die funktionalen Building Blocks einer PaaS Cloud.

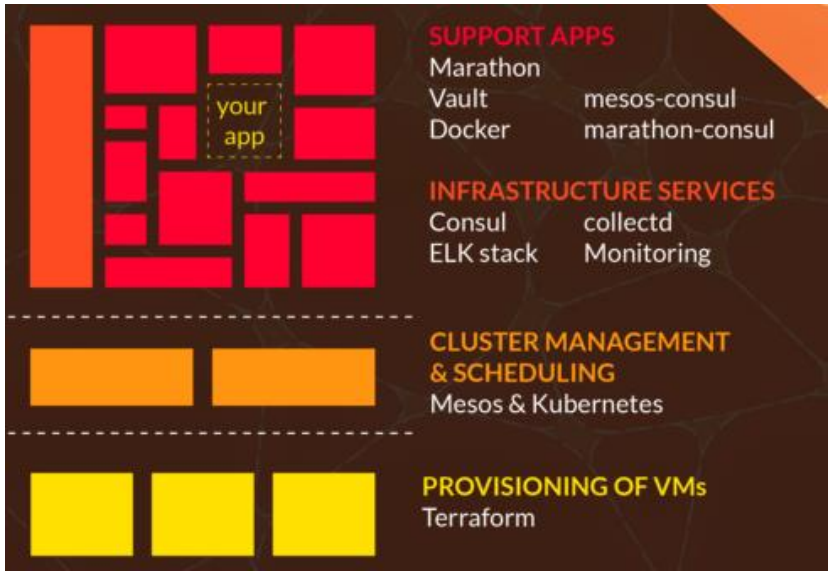


High-Level Architektur einer PaaS am Beispiel Heroku.



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

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

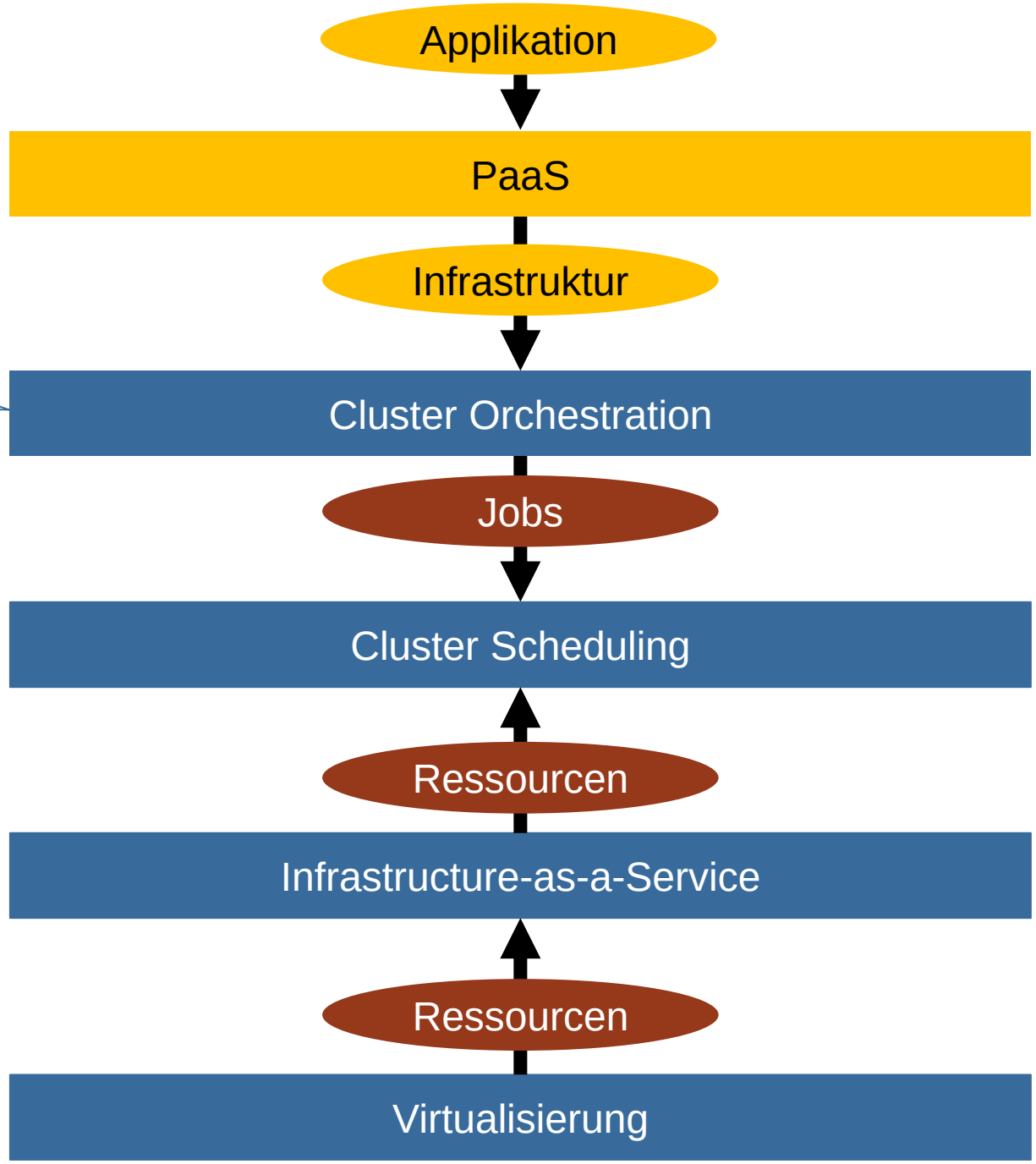
Cluster Orchestration

Cluster Scheduling

Das Big Picture

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

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



Private PaaS Clouds am Beispiel Flynn

Flynn

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

In a Nutshell, Flynn.io...

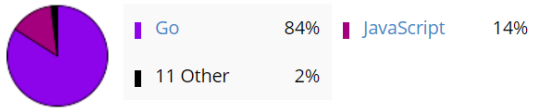
... has had 6,154 commits made by 104 contributors
representing 409,138 lines of code

... is mostly written in Go
with an average number of source code comments

... has a young, but established codebase
maintained by a very large development team
with stable Y-O-Y commits

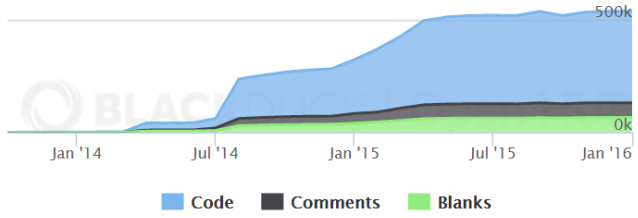
... took an estimated 111 years of effort (COCOMO model)
starting with its first commit in October, 2013
ending with its most recent commit 2 days ago

Languages



Very High Activity

Lines of Code

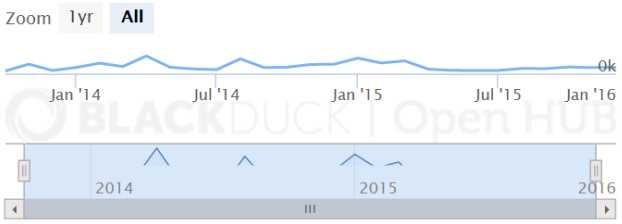


Activity

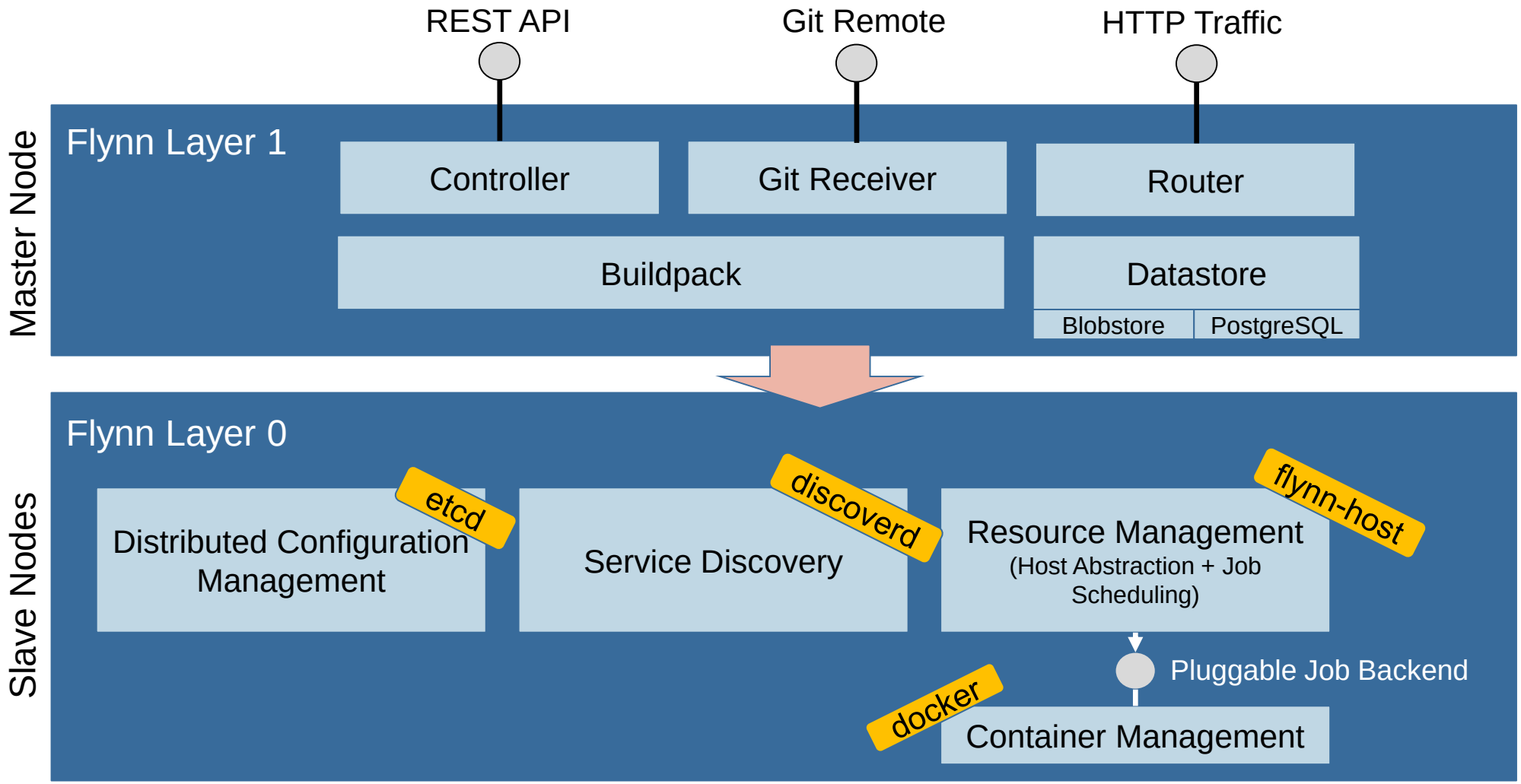
30 Day Summary
Dec 10 2015 — Jan 9 2016
139 Commits
6 Contributors
including 1 new contributor

12 Month Summary
Jan 9 2015 — Jan 9 2016
2473 Commits
Down -603 (19%) from previous 12 months
35 Contributors
Down -30 (46%) from previous 12 months

Commits per Month



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.

Public PaaS Clouds am Beispiel Google App Engine

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)	

Die Google App Engine

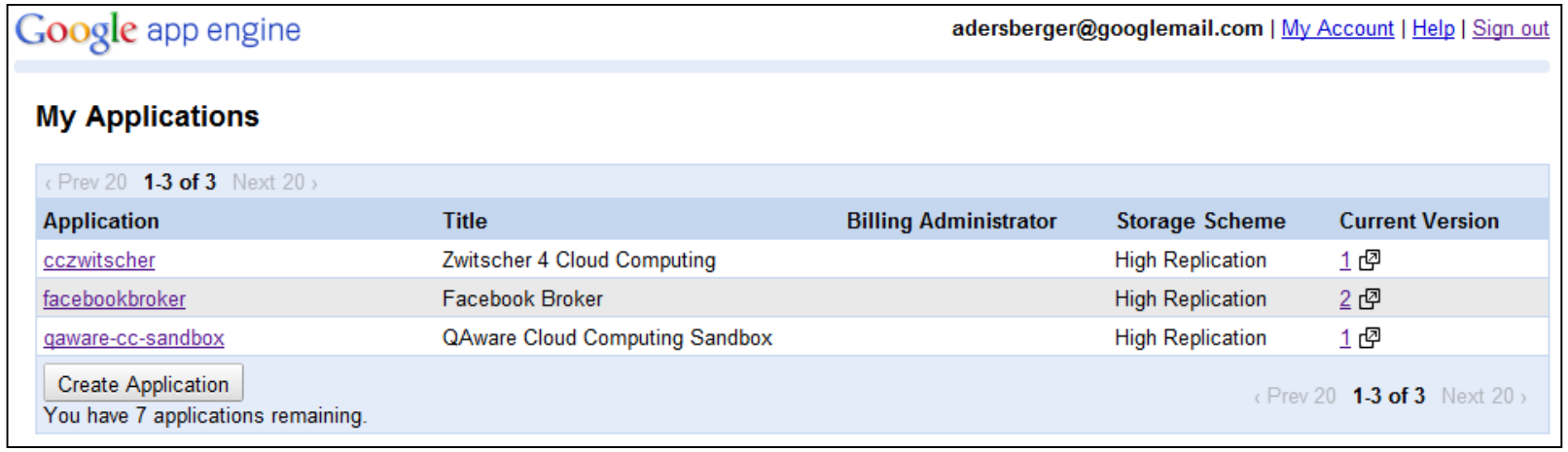
- Die Google App Engine (GAE) ist das PaaS-Angebot von Google.
- Anwendungen laufen innerhalb der Google Infrastruktur.
- Der Betrieb der Anwendungen ist innerhalb bestimmter Quoten kostenfrei. Danach fallen Kosten u.A. auf Basis von Service-Aufrufen, Storage-Volumen und real genutzten CPU-Sekunden an.
- Unterstützte Sprachen:
  - Python (seit 2008)
  - Java (seit 2009)
  - Go - „Python meets C++“ (seit 2011)
  - PHP (seit 2013)
- Integrationen in alle gängigen IDEs stehen zur Verfügung (Eclipse, intelliJ, Netbeans).

Einschränkungen der Google App Engine.

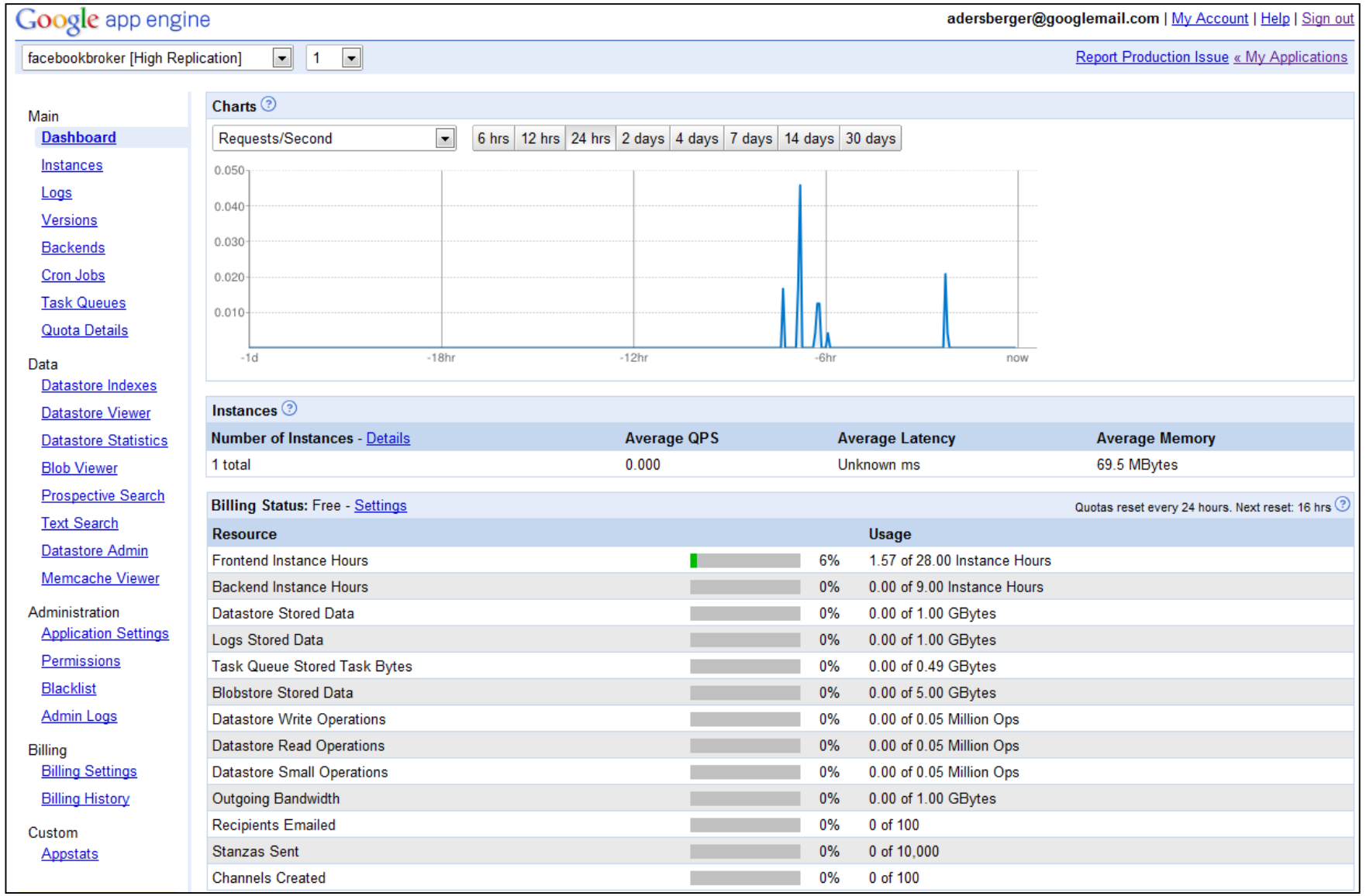
Eine GAE-Applikation läuft in einer Sandbox, die das Verhalten der Applikation einschränkt. Dies geschieht mit dem Ziel, die Verarbeitung effizient zu halten und die Infrastruktur im Auto-Scaling zu schützen.

- 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
- 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

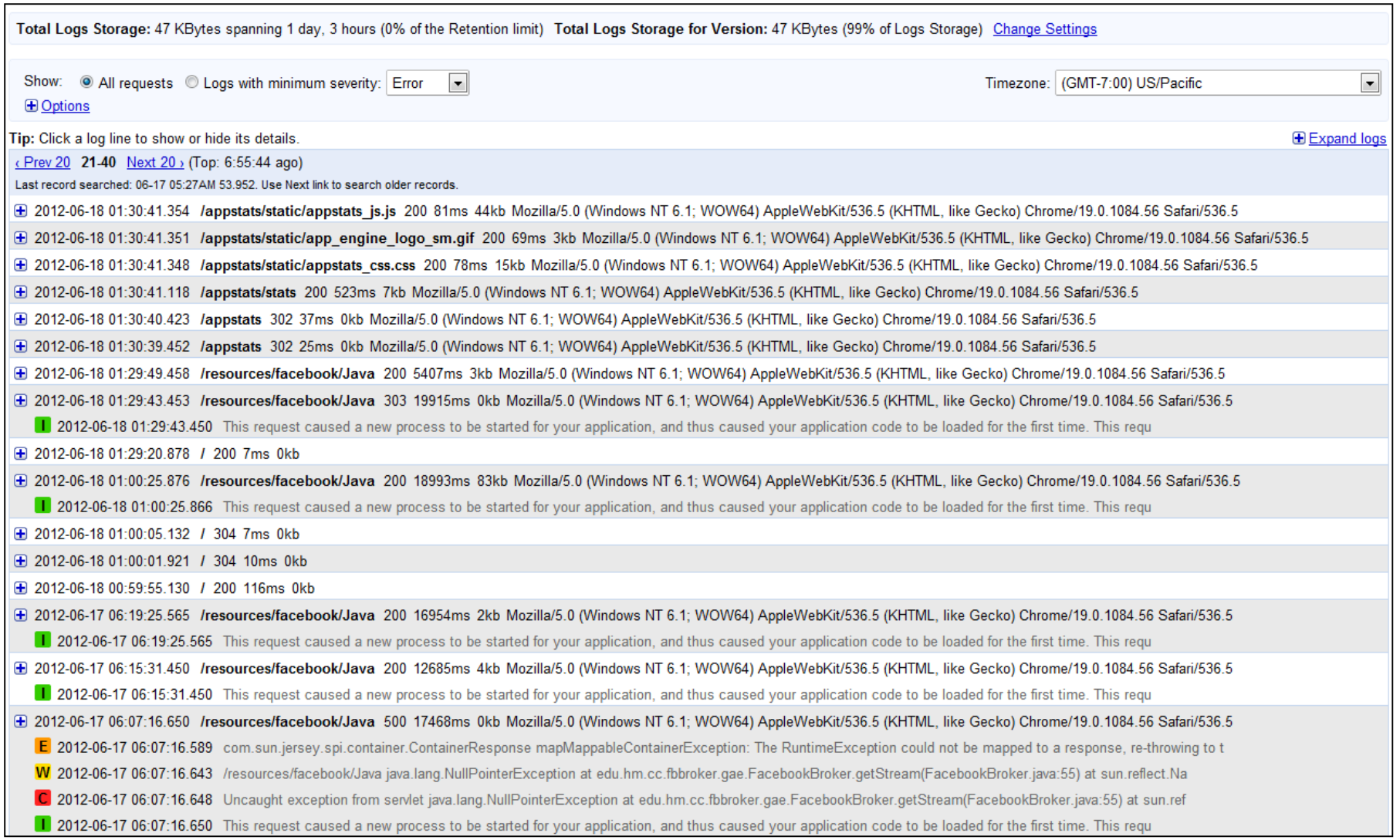
Funktionen der Admin-UI: Die Applikationsübersicht.



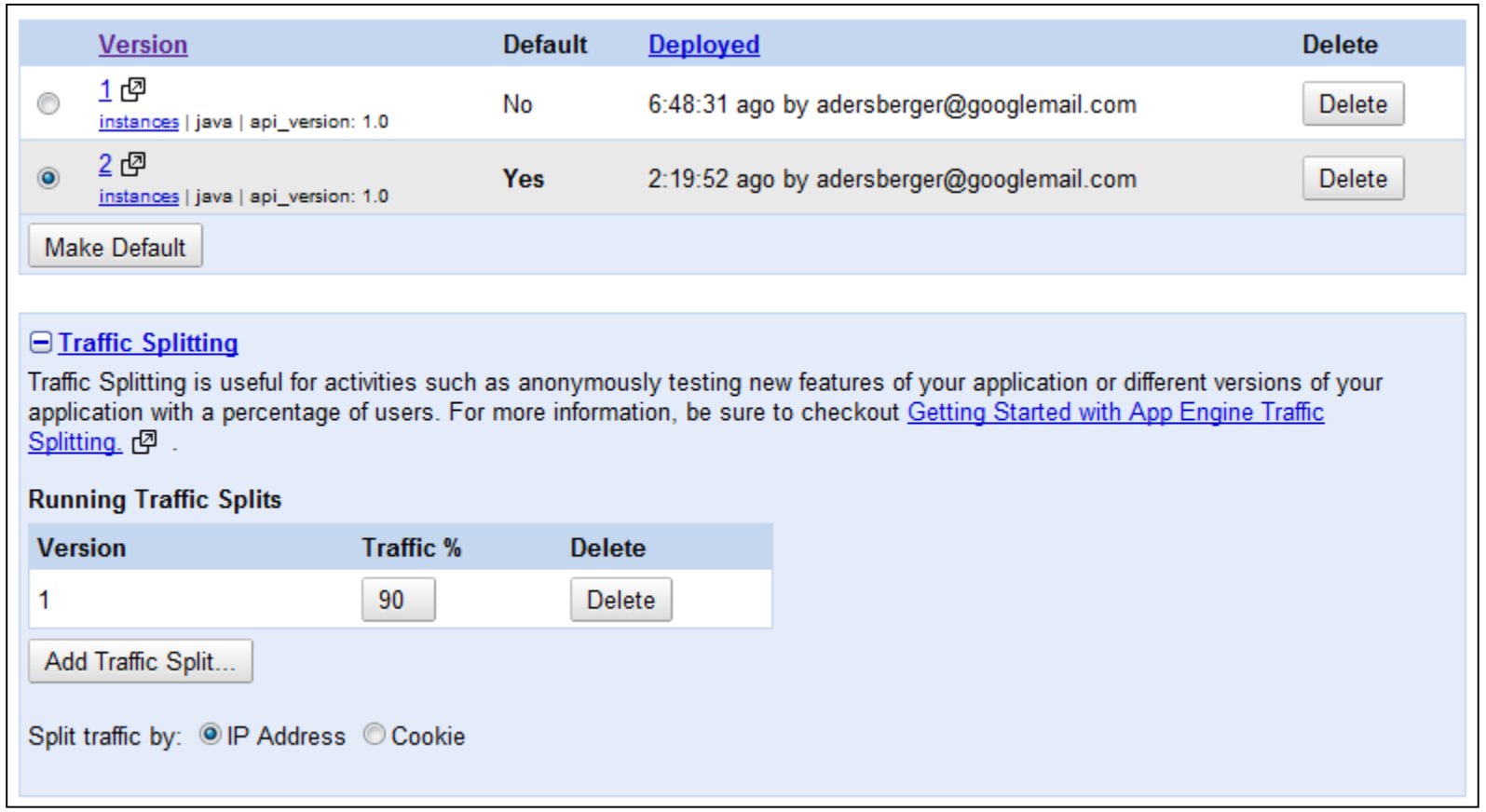
Funktionen der Admin-UI: Das Applikations-Dashboard.



Funktionen der Admin-UI: Zugriff und Analyse der Log-Einträge.



Funktionen der Admin-UI: Mehrere Versionen einer Applikation gleichzeitig betreiben.



Funktionen der Admin-UI: Verwaltung der Plattform-Dienste (z.B. Datastore).