Jeder Topf braucht seinen Deckel – Cloud Foundry auf OpenStack mit Bosh

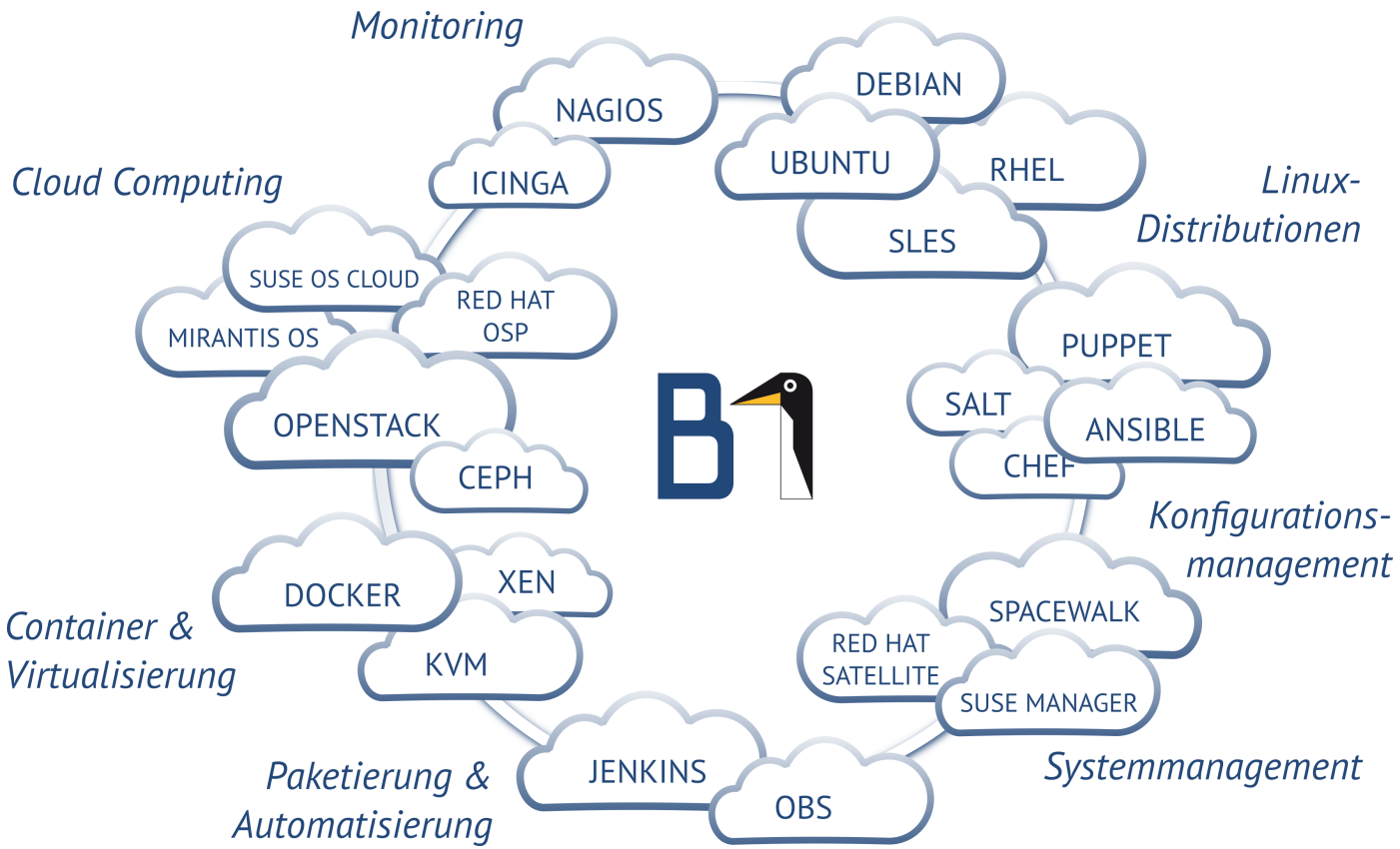
Deutsche OpenStack Tage 2017, München 28. Juni 2017

Michel Rode
Linux/Unix Consultant & Trainer
B1 Systems GmbH
rode@b1-systems.de

Vorstellung B1 Systems

- gegründet 2004
- primär Linux/Open Source-Themen
- national & international tätig
- ca. 100 Mitarbeiter
- unabhängig von Soft- und Hardware-Herstellern
- Leistungsangebot:
  - Beratung & Consulting
  - Support
  - Entwicklung
  - Training
  - Betrieb
  - Lösungen
- Büros in Rockolding, Köln, Berlin & Dresden

Schwerpunkte

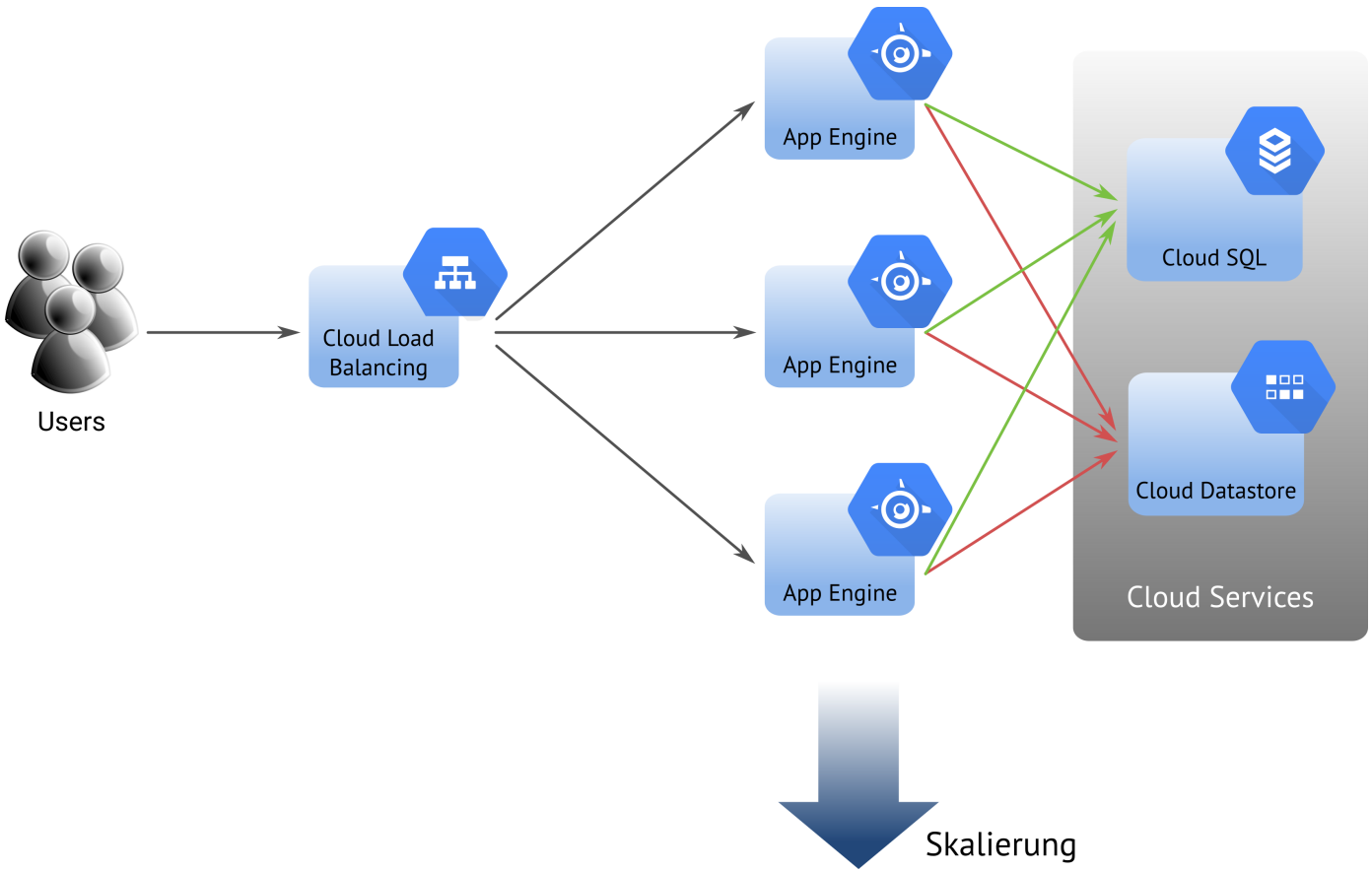


Warum Cloud Foundry auf den OpenStack Tagen?

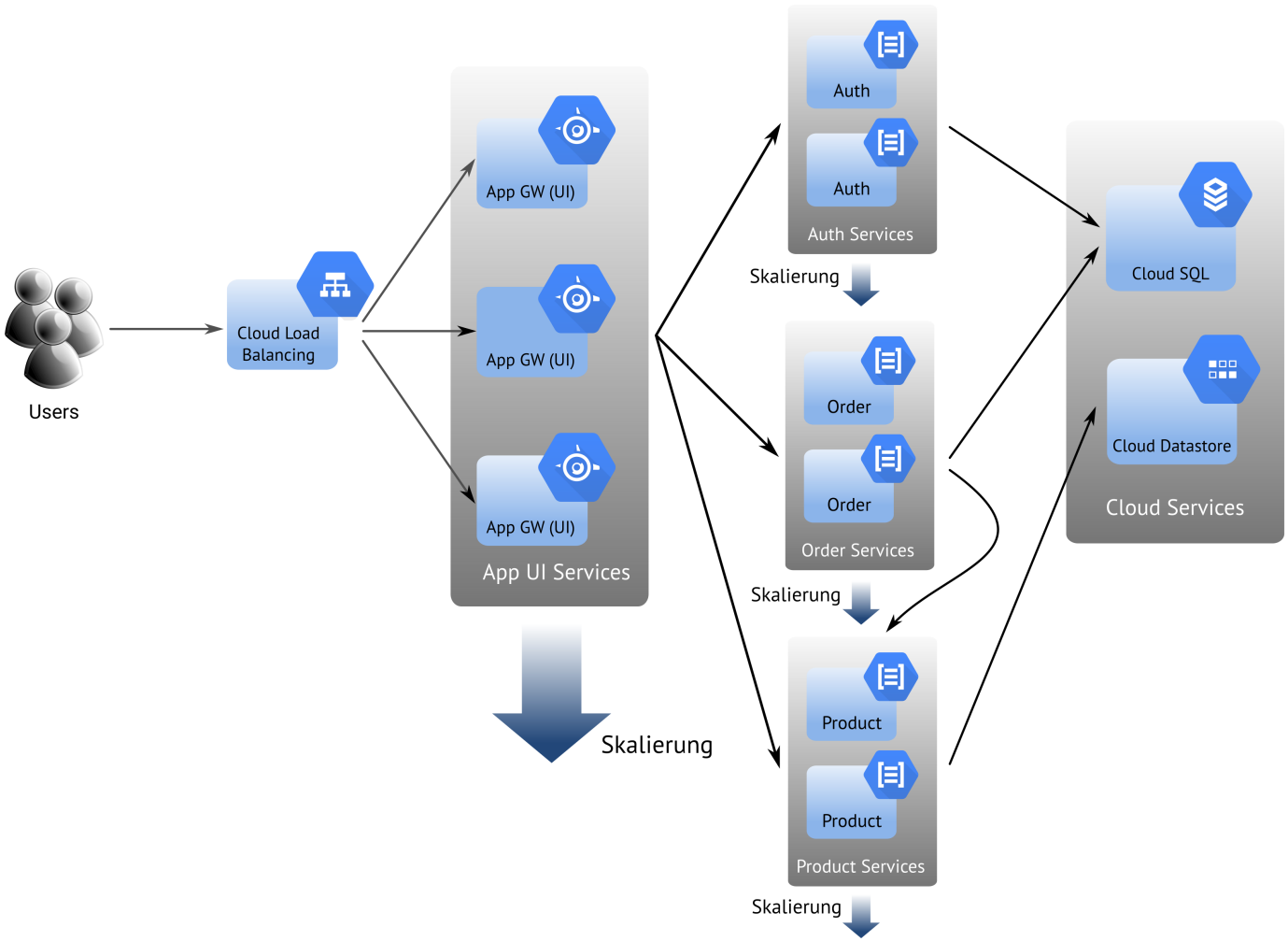
Herausforderungen

- geschätzt bis 2020 über 50 Milliarden Geräte alleine im IoT (*Bernd Heinrichs, Cisco*)
- hohe Varianz und Fluktuation der Auslastung der Infrastruktur
- Infrastruktur muss flexibel sein
- Redesign monolithischer Applikationen
- Erhöhung der Skalierbarkeit der Apps
- Erhöhung der Wartbarkeit der Apps

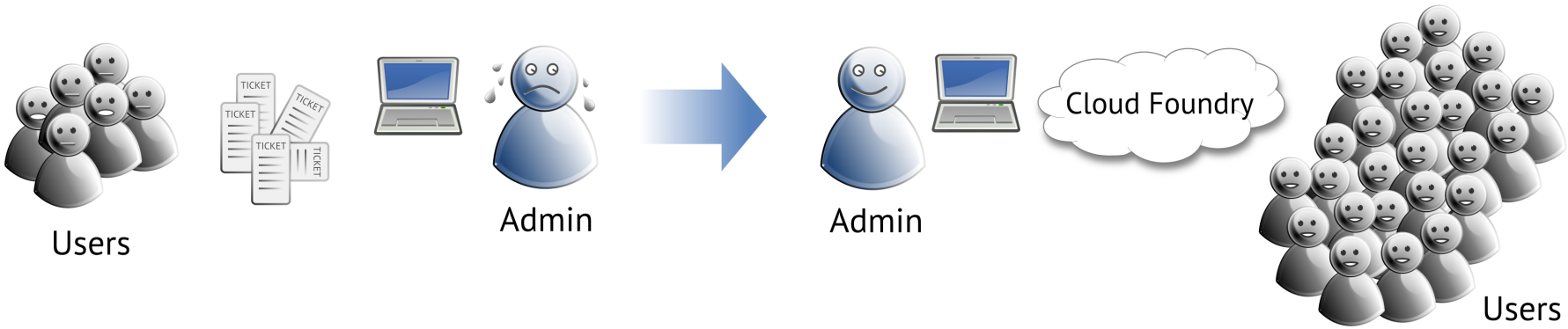
App monolithisch



App Micro Service

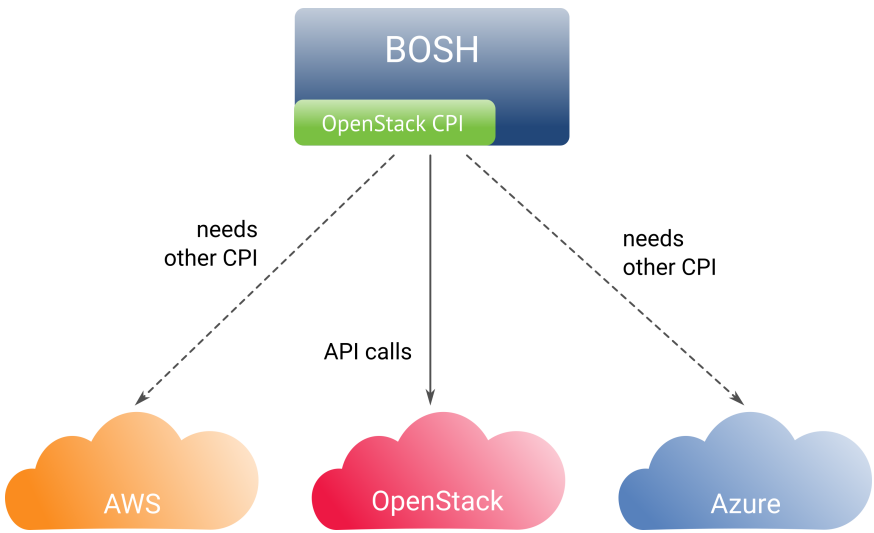


Änderung der Arbeitsweise Admin und Dev



- Beziehung Admin (Provider) vs. Kunden (Dev) ändert sich
- Request- & Response-Zeiten zwischen Admin und Dev fallen nicht mehr ins Gewicht
- Entwickler fokussiert sich auf App
- Stage und Live Systeme absolut identisch durch ECO-System
- heterogene Systeme schwerer zu pflegen $\rightarrow$ Admin langsam

Cloud Foundry ist IaaS-unabhängig



- schneller & einfacher Wechsel zu AWS, Azure, OpenStack, GCP, vSphere, ...
- Kunden können entscheiden, welcher IaaS eingesetzt wird
  → größere Zielgruppe
- minimale Konfigurationsänderungen bei IaaS-Wechsel
- Diversität der zu wartenden Plattform auf verschiedenen IaaS Layern nimmt dramatisch ab

Was ist eigentlich Cloud Foundry?

Die Eckdaten

- erste Version 2011 von VMware, spätere Auslagerung zu Pivotal
- die meisten Komponenten in Ruby geschrieben
- Redesign einiger Komponenten in GO
- 2 Versionen verfügbar:
  - PCF: Pivotal Cloud Foundry, kostenpflichtig inkl. Support
  - CCF: Community Cloud Foundry, Open Source
  - beide Versionen nutzen die gleiche Codebasis
- wird deployed und überwacht mit Bosh
- *Platform as a Service*, vergleichbar mit Google App Engine, Heroku oder IBM Bluemix

Feature-Spektrum

- individuelle Gestaltung der Größe und Komplexität meines Environments
  - Anzahl und Größe der App-Instanzen (Container)
  - Welche Services benötigt meine App (Redis, DB, . . .)?
  - Welche Spaces benötige ich (dev, testing, prod)?
- Services können On-Demand eingebunden oder entfernt werden
- Log-Management für die App-Container
- Paketversionen und Programmiersprache spielen keine Rolle (Buildpacks)
- einfacher Umzug aller Apps auf einen anderen IaaS

Bosh

Was ist Bosh?

- Orchestrierung der Deplyoments
- Verwalten der Releases
- Verwalten der Stemcells
- Überwachung & Wiederherstellung der Deployment-Komponenten
- Zero-Downtime Update-Mechanismus

Deployment

- Beschreibung aller Komponenten in einer/mehreren YAML-Datei(en):
  - Passwörter
  - Netzwerk
  - Volumes
  - IaaS-Spezifika (Login, API-URL, . . .)
  - Welche CF-Komponenten auf welche VM?
  - Verteilung der VMs
- Deployment ist die oberste Abstraktionsebene

Releases

- Deployment multipler Releases möglich (CF, PostgreSQL, Redis, ...)
- Release enthält:
  - Binaries/Source Code
  - Scripts
  - Konfigurationstemplates
- Binaries werden in speziellen VMs kompiliert und im zentralen Blobstore abgelegt

Stemcell

- Stemcell repräsentiert das Basis-Image
- ist optimiert für jeweiligen IaaS
- Stemcells werden von Pivotal bereitgestellt
- Stemcell-Version korreliert mit Cloud Foundry Version
- kann selbst erstellt werden, wenn Voraussetzung erfüllt
- kann in CI-Prozess eingebettet werden

Bosh – Aufbau

Director

- Orchestrierungspart von Bosh
- Kommunikation mit dem IaaS via CPI (*Cloud Provider Interface*)
- nimmt Action/Commands von Admins via Bosh CLI entgegen
- steuert geplante Prozesse wie Backups
- verwendet Task Queue für alle Aufgaben
- Erstellung von VMs, Volumes, Netzwerk, usw.

CPI – *Cloud Provider Interface*

- keine eigenständige Bosh-Komponente
- Bindeglied zwischen IaaS und Bosh Director
- jeder IaaS hat eigenen Bosh CPI
- repräsentiert nur Basiskommandos
  - Create/Delete VMs, Volumes, NICs
  - Attach/Detach Volumes, NICs
- stellt spezielle IaaS-Parameter bereit
  - Availability Zone
  - Volume Thin Provisioning
  - Security Groups
  - ...
- OpenStack CPI maintained von SAP
- CPI nutzt Ruby Cloud Library „Fog“

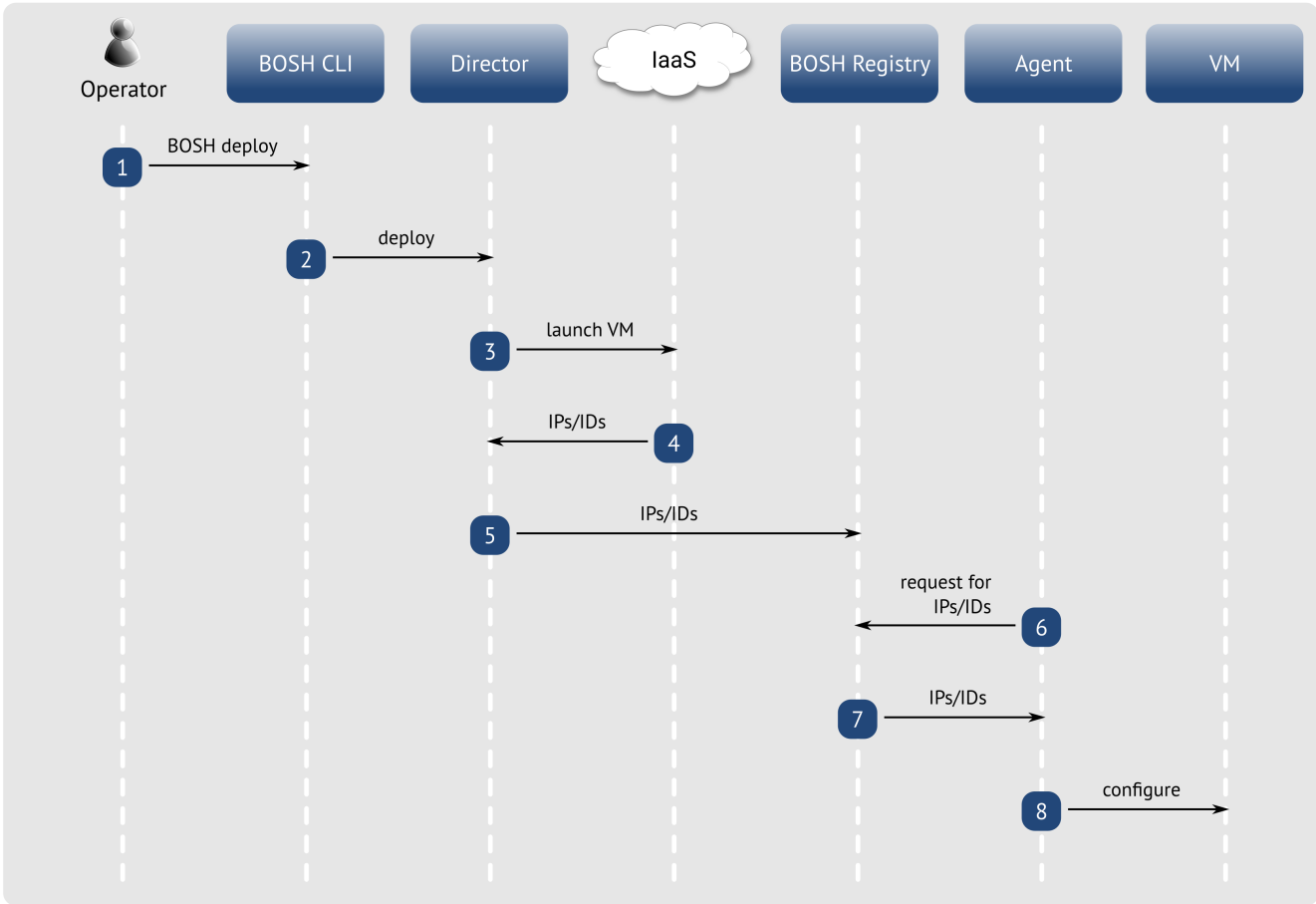
Agent

- läuft auf jeder VM
- meldet Status und Metrics an Bosh Director
- ersetzt kein detailliertes Monitoring durch 3rd-Party-Tools (Icinga, Datadog, . . .)
- erhält Informationen von Bosh Registry
  - Um welche VM handelt es sich?
  - Wie muss die VM eingerichtet werden?
  - Welche Templates und Binaries sollen verwendet werden?

Registry

- erster Call eines Bosh Agent geht an Registry
- Informationen für das Bootstrapping der VMs
- Konfigurationsinformationen für die VM

Beispiel



Quelle: https://bosh.io/docs/bosh-components.html

Database

- Single-Node PostgreSQL Database (SPOF)
- mögliche Auslagerung zu einem Cluster
- beinhaltet:
  - DNS-Informationen
  - Informationen zu Deployment, Release, Stemcell
  - Status-Informationen
  - Task Queue

Health Monitor

- betrachtet Zustand der VMs
- verwendet Status und Alerts von den Agents
- Resurrector (Auto Heal)
  - Plugin von Health Monitor
  - kann deaktiviert werden
  - bei Fehler wird VM neu gebaut

DNS-Server

- interne Domain-Auflösung
- wird benötigt für die interne Kommunikation
- auch die Service-Endpoint-Domains werden hier gespeichert
- SPOF!!

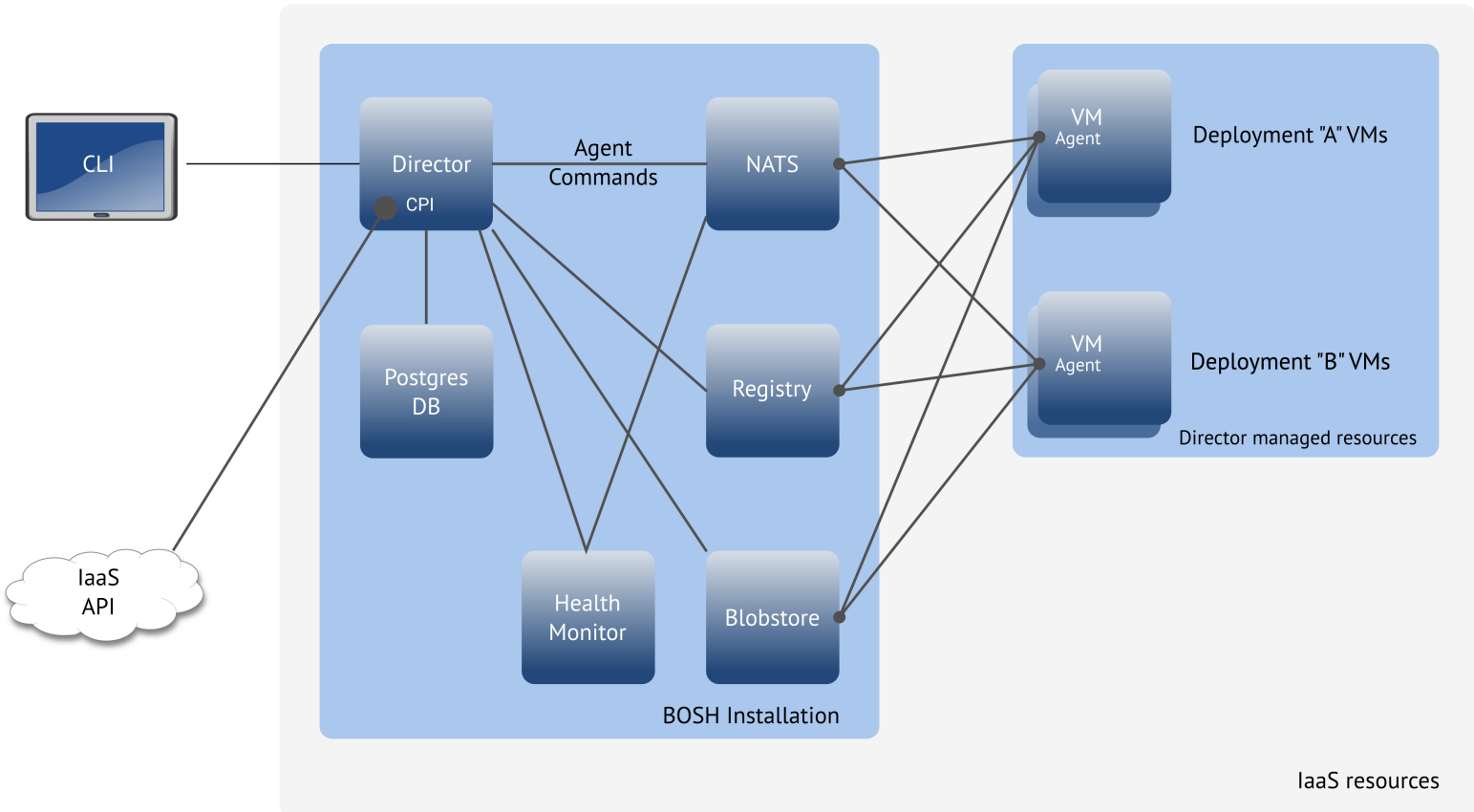
Blobstore

- beinhaltet alle vorkompilierten Binaries, Logs und andere Assets
- kann auf verschiedenen Storagetypen betrieben werden:
  - S3
  - Swift
  - persistentes Volume
  - ...

NATS

- zentraler Messaging Bus
- in Golang geschrieben
- Instruktionen für die Bosh Agents
- Informationen für Health Manager

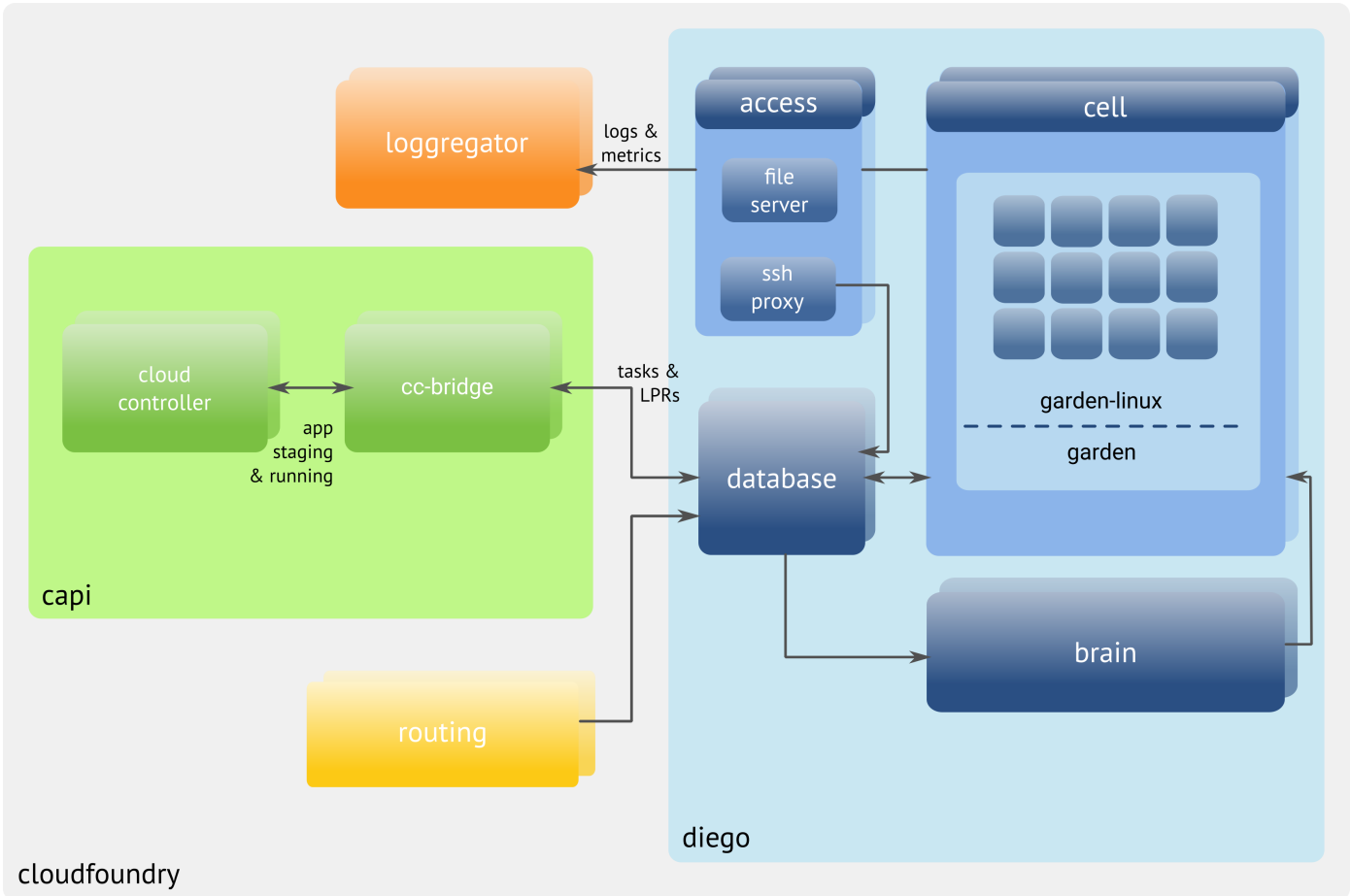
Architektur-Übersicht



Quelle: https://bosh.io/docs/bosh-components.html

Cloud Foundry

Schlüsselkomponenten



Einsatzgebiete

- Voraussetzung für Anwendungen: 12-Factor-Manifest
- Beispiele:
  - Industrie 4.0
  - autonomes Fahren
  - einfache Anwendungen/Web-Applikationen
  - große Applikationskonstrukte
- nicht geeignet für:
  - Batch Jobs
  - (Cluster-)Services
  - monolithische Anwendungen

Probleme bei Cloud Foundry auf OpenStack

- API-Endpoints müssen verfügbar sein, speziell Metadata
- bei eingeschaltetem Resurrector:
  - Compute Node Failure → Volume Detach funktioniert nicht
  - kein Auto Heal der VMs möglich
  - VM kann nicht von Bosh neu gebaut werden
  - Major Issue bei Betrieb von Cloud Foundry auf OpenStack
- API Rate Limit hoch setzen
- Reaktionszeit Cinder und Nova Agent

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de
oder +49 (0)8457 - 931096