Public Cloud 101

Florian Barth

barth@stocard.de

@der_cthulhu

Matthias Luft

matthias.luft@rational-security.io

@uchi_mata

#whoami – Florian Barth

- Founder & CTO Stocard
- "TeamSecDevOpsOrgNet"

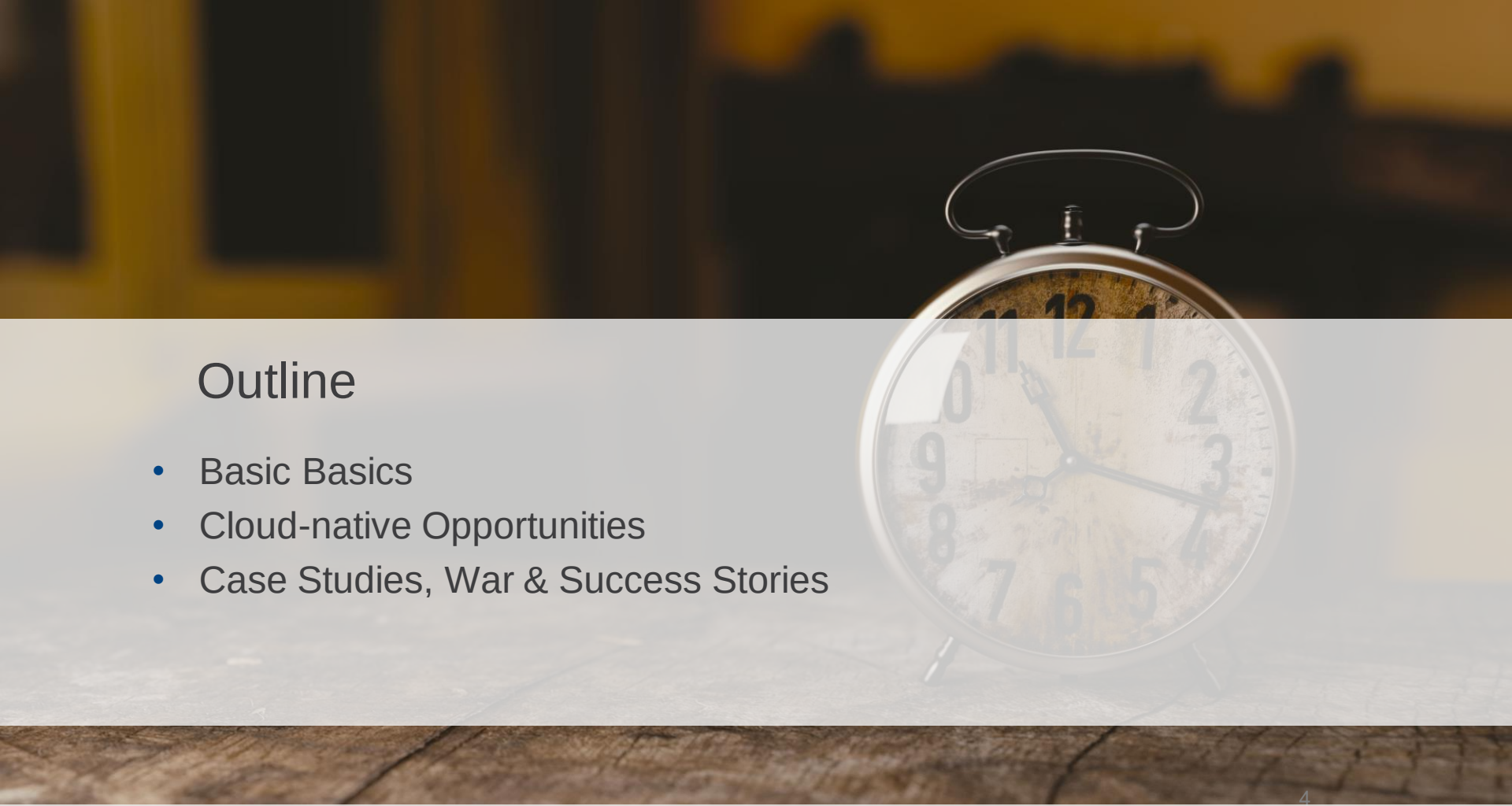
- 11+ years in Software Engineering (academic & professional)

- barth@stocard.de
- @der_cthulhu

STOCARD

#whoami – Matthias Luft

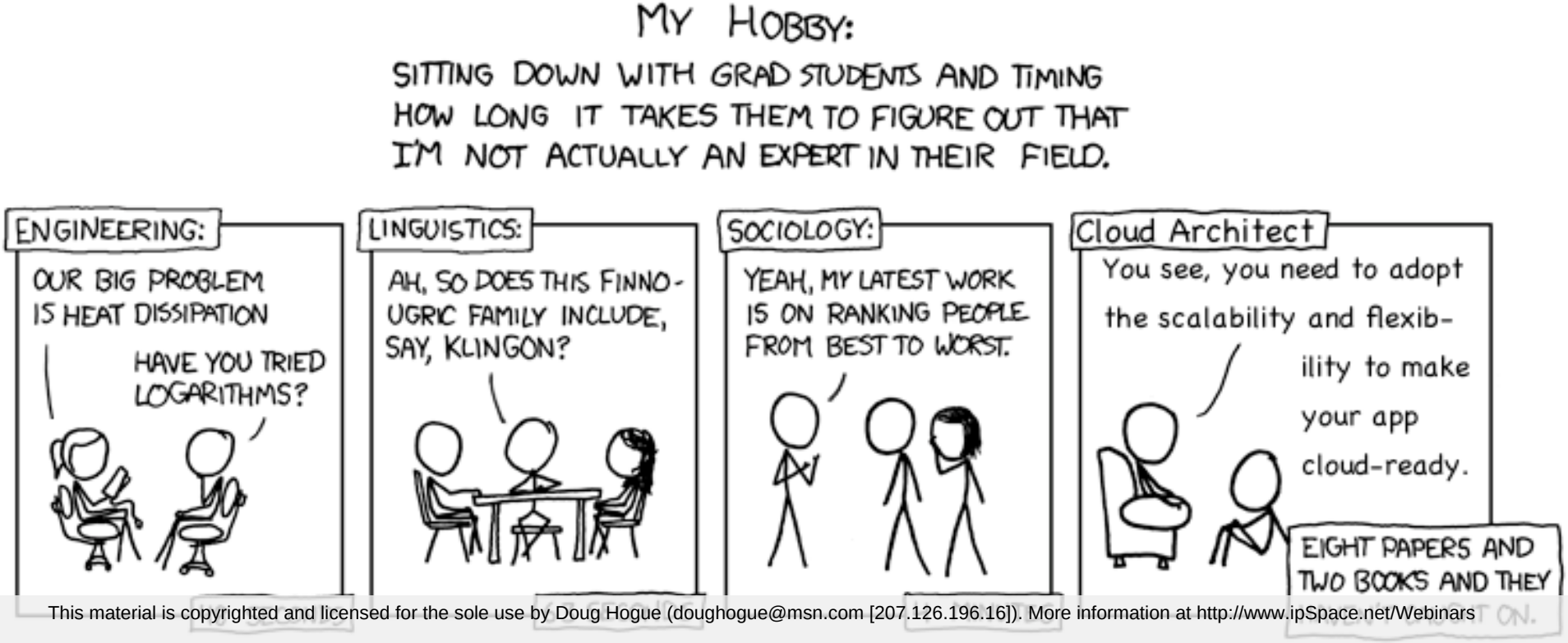
- IT Security Researcher/Engineer/Trainer
  - Day Job: Principal Platform Security Engineer @Heroku Salesforce. Not here in that capacity, opinions are my own.
- 10+ years in IT Security
- Technical background:
  - Pentesting
  - Virtualization/Hypervisor/Network Security
  - Nowadays: Cloud/Container/Agile Security
- matthias.luft@rational-security.io
- @uchi_mata



Outline

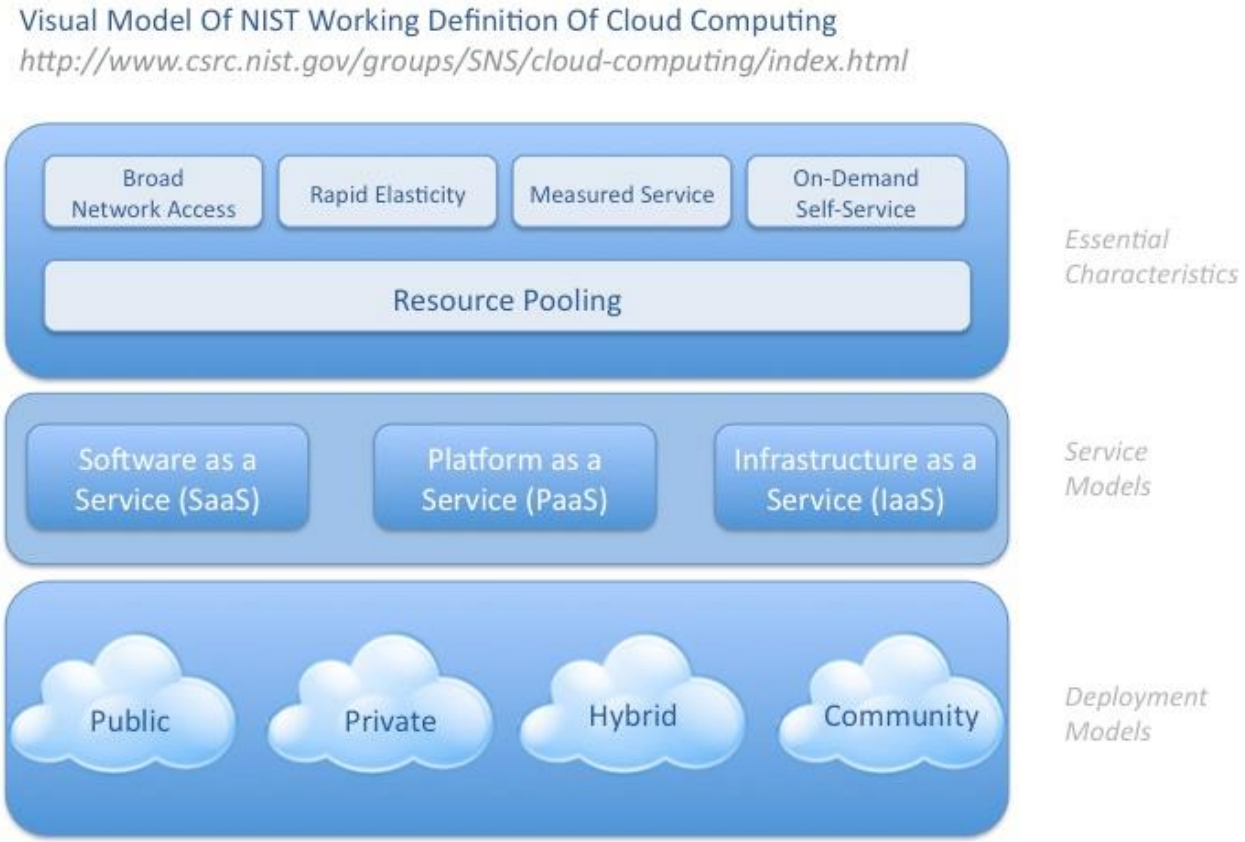
- Basic Basics
- Cloud-native Opportunities
- Case Studies, War & Success Stories

What is the Cloud?



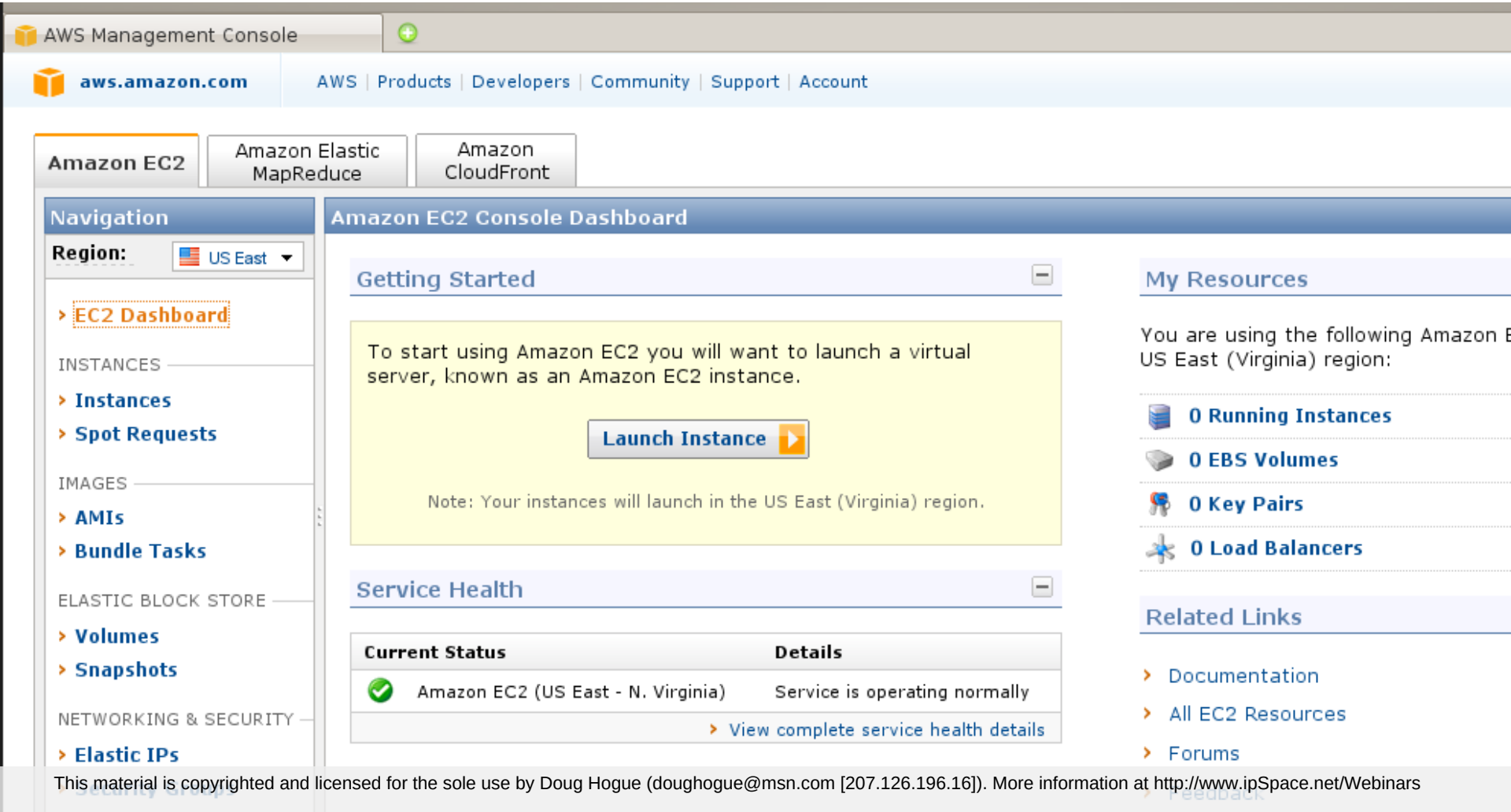
Definition of Cloud Computing

NIST SP 800 145



Demo

IaaS



New Service

Summary | Account | Help and Resources

- Windows Azure
  - ernw_test
- SQL Azure
- AppFabric
- Marketplace

Abonnement-1 | ernw_test

Description Edit

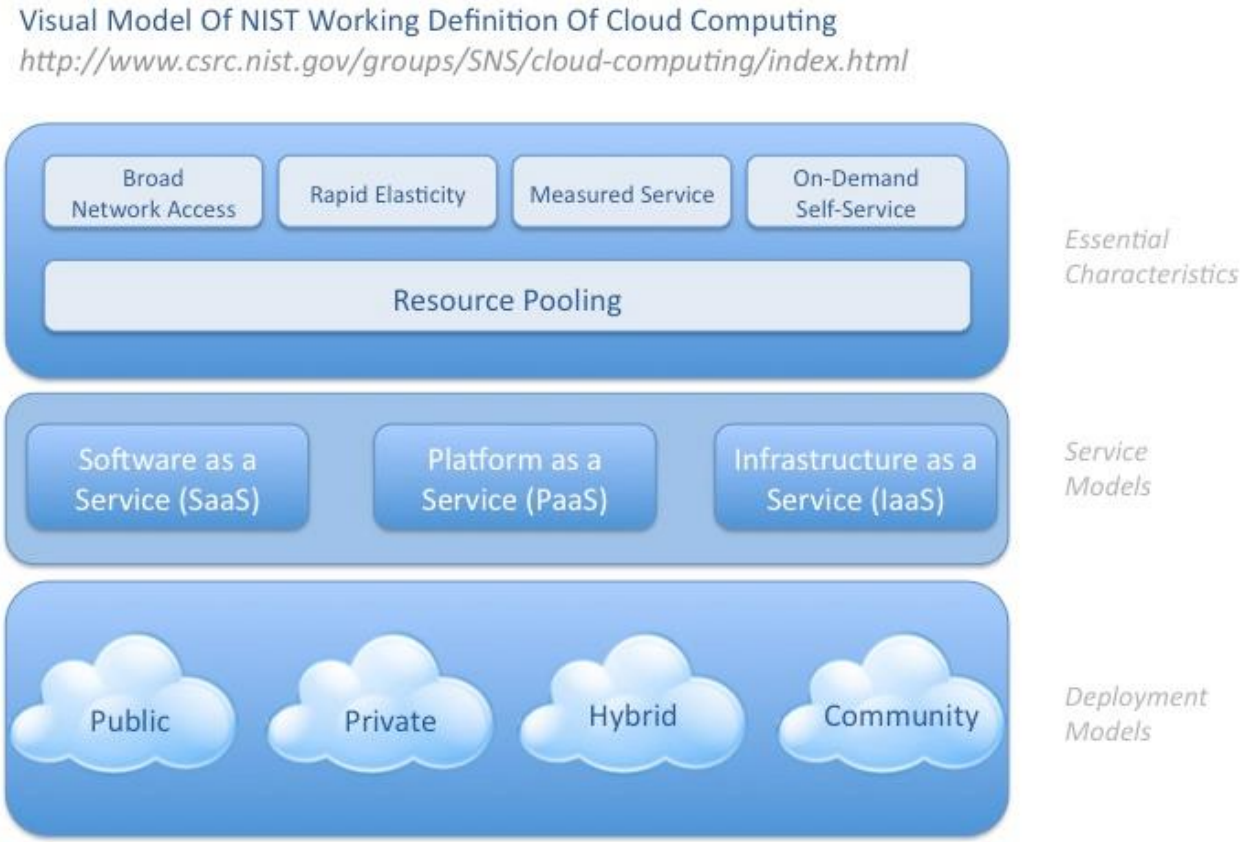
Delete Service

Hosted Service

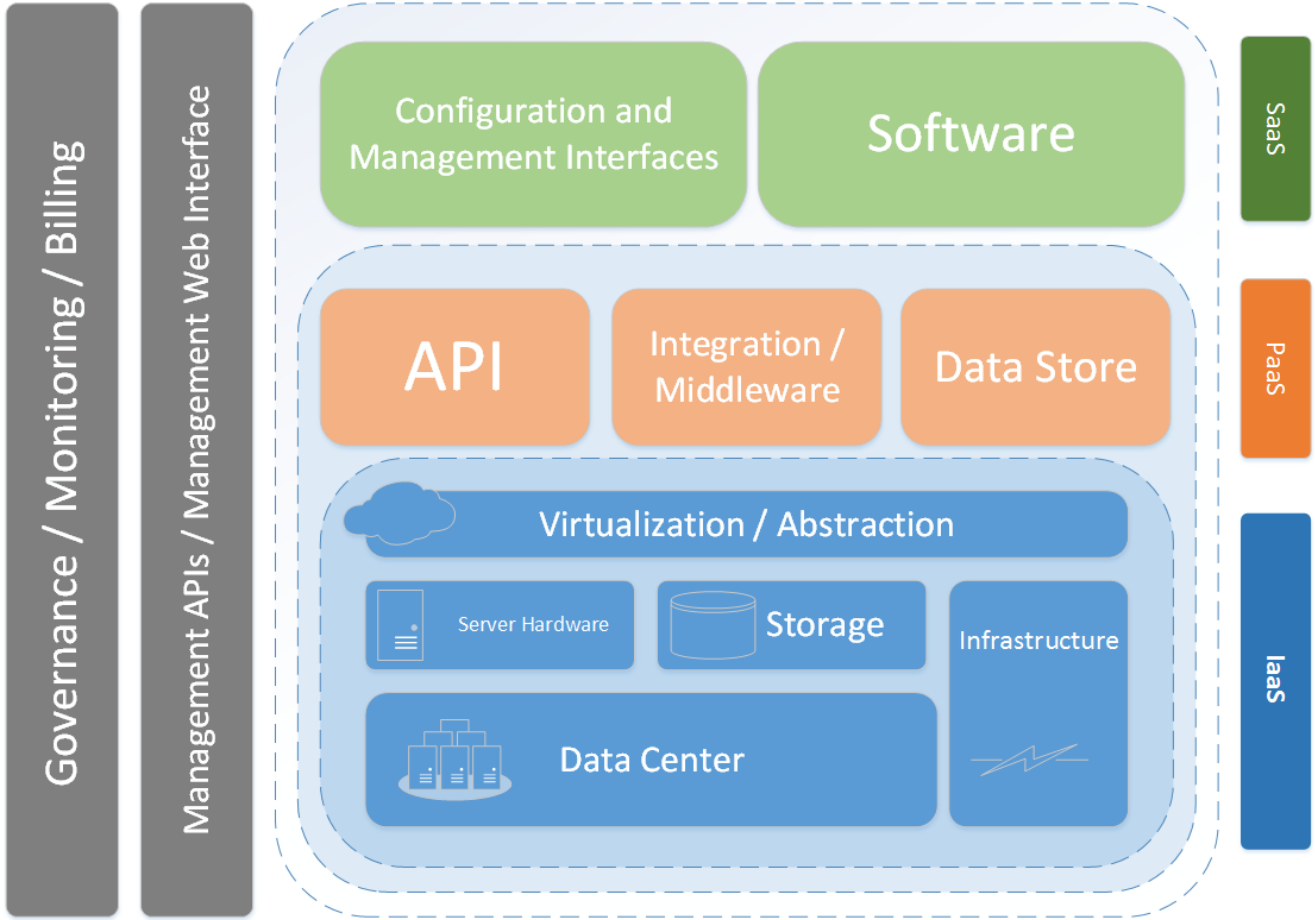
Production

Definition of Cloud Computing

NIST SP 800 145

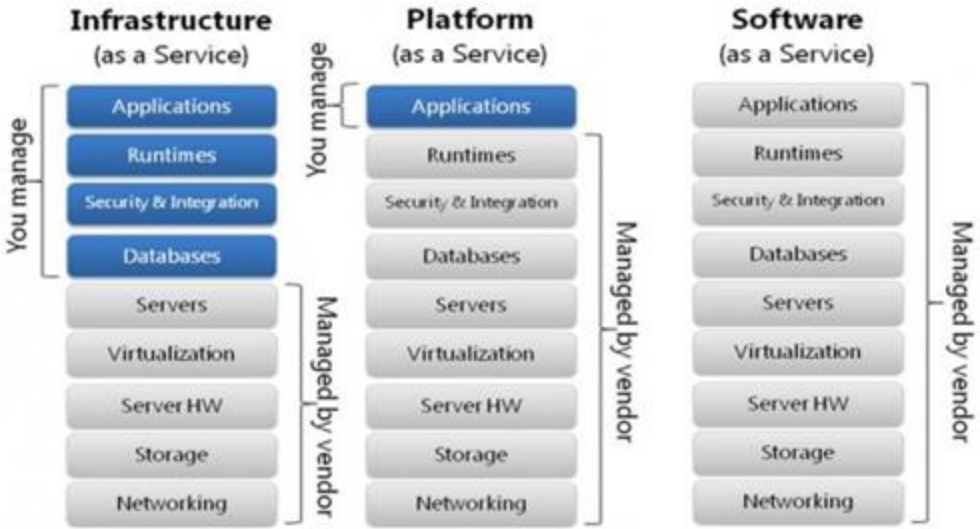


Cloud Layers



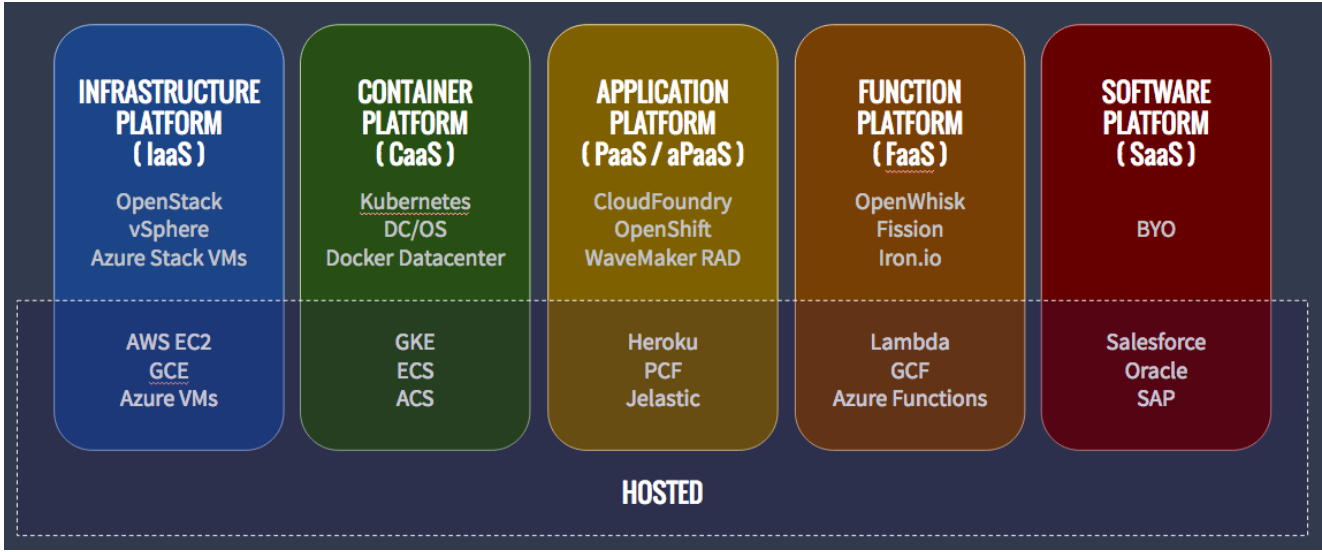
aaS

[Source]



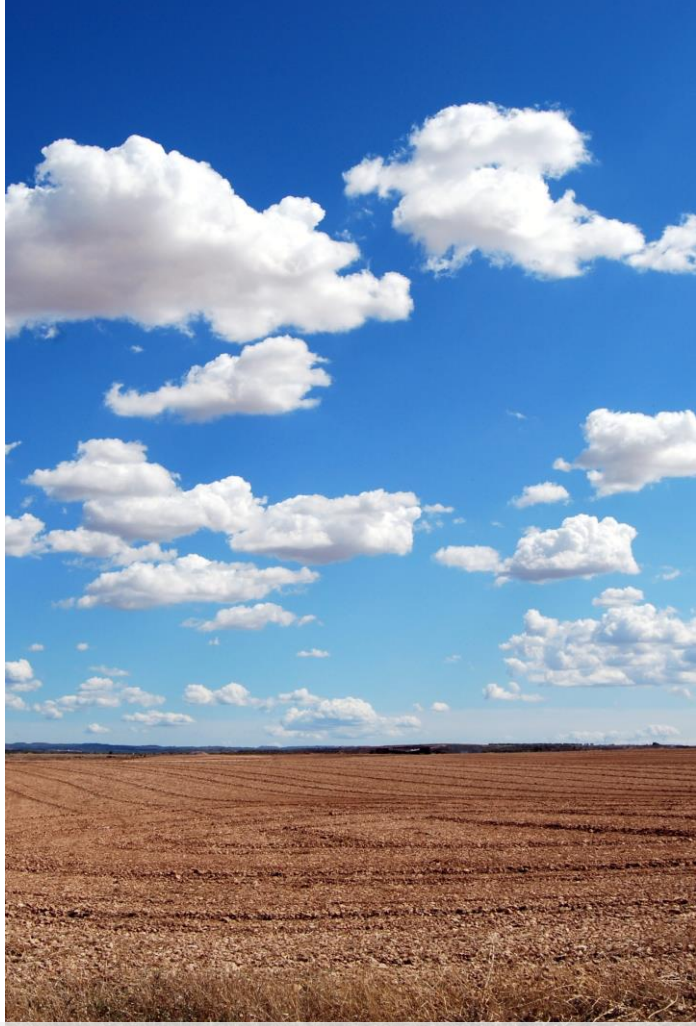
aaS

[Source]



There Is No Cloud

- Cloud is a combination of technologies.
- Large number of *standardized* services that can be *fully orchestrated*
- Services can be understood.
  - Certain proprietary elements, but no black magic.



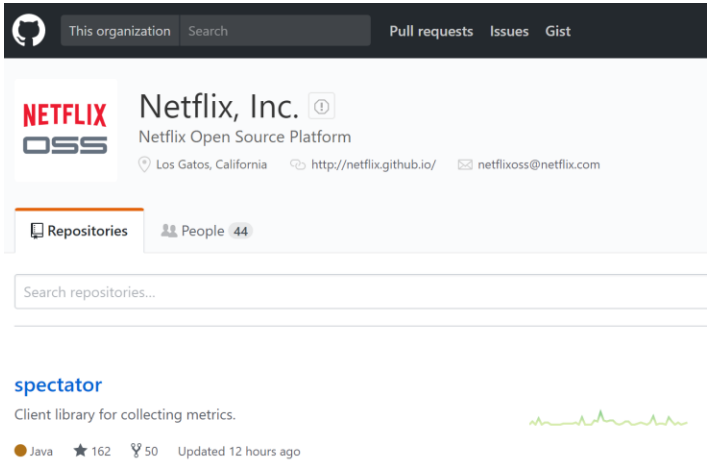
Great Contributions!

DOI:10.1145/1435417.1435432

Building reliable distributed systems at a worldwide scale demands trade-offs between consistency and availability.

BY WERNER VOGELS

Eventually Consistent



So That's It?

- Simply standard services that can be ordered automatically?

➲ Orchestration everywhere allows end-to-end deployment and operation.

➲ Cloud services enable *Cloud-native* IT paradigm.

No Free Lunch

- Use of public cloud services will not automatically improve everything.
- You…
  - have to deal with features and complexity like with on-premise software.
  - are still responsible for good design and implementation.
  - have to build new skills and toolchains.

FaaS: AWS Lambda

- High scalability, reliability, availability, ...
- Language support: node.js, python, ruby, Java, Go, .NET
- Custom runtime feature allows to support "every language"
- Tight integration with AWS ecosystem
- Invocation: event-based, SDK, HTTP
- Monitoring and Logging via CloudWatch

Demo

FaaS

But what about…?

- CI/CD
- Infrastructure as Code
- Immutable Infrastructure
- Container
- DevOps

Continuous Integration

"… is a development practice that requires developers to **integrate** code into a **shared repository several times a day**. Each check-in is then **verified by an automated build**, allowing teams to detect problems early."

thoughtworks.com/continuous-integration

Continuous Delivery

“… is a software engineering approach in which teams produce software in **short cycles**, ensuring that the software can be **reliably released** at any time. It aims at **building**, **testing**, and **releasing** software **faster and more frequently**.”

DOI: 10.1109/MS.2015.27

Continuous Deployment

… is often confused with Continuous Delivery and “means that every change goes through the pipeline and **automatically** gets put into production, resulting in **many production deployments every day**.”

https://martinfowler.com/bliki/ContinuousDelivery.html

Infrastructure-as-Code

- Manage und provision **data centers** through **machine-readable** definition files
- Version control your infrastructure

- Documentation & Evolution
- Static/Dynamic analysis

Immutable Infrastructure

- ~~"Servers are cattle not pets"~~
- "Servers are **syringes**"

- Spawn your infra from a template or recipe

- **Update**, **Test** it, **Spawn** new, **Trash** old

DevOps

"DevOps is the philosophy of unifying Development and Operations at the **culture**, **system**, **practice**, and **tool** levels, to achieve **accelerated** and more **frequent delivery** of **value** to the customer, by improving quality in order to increase velocity."

Rob England, 2014

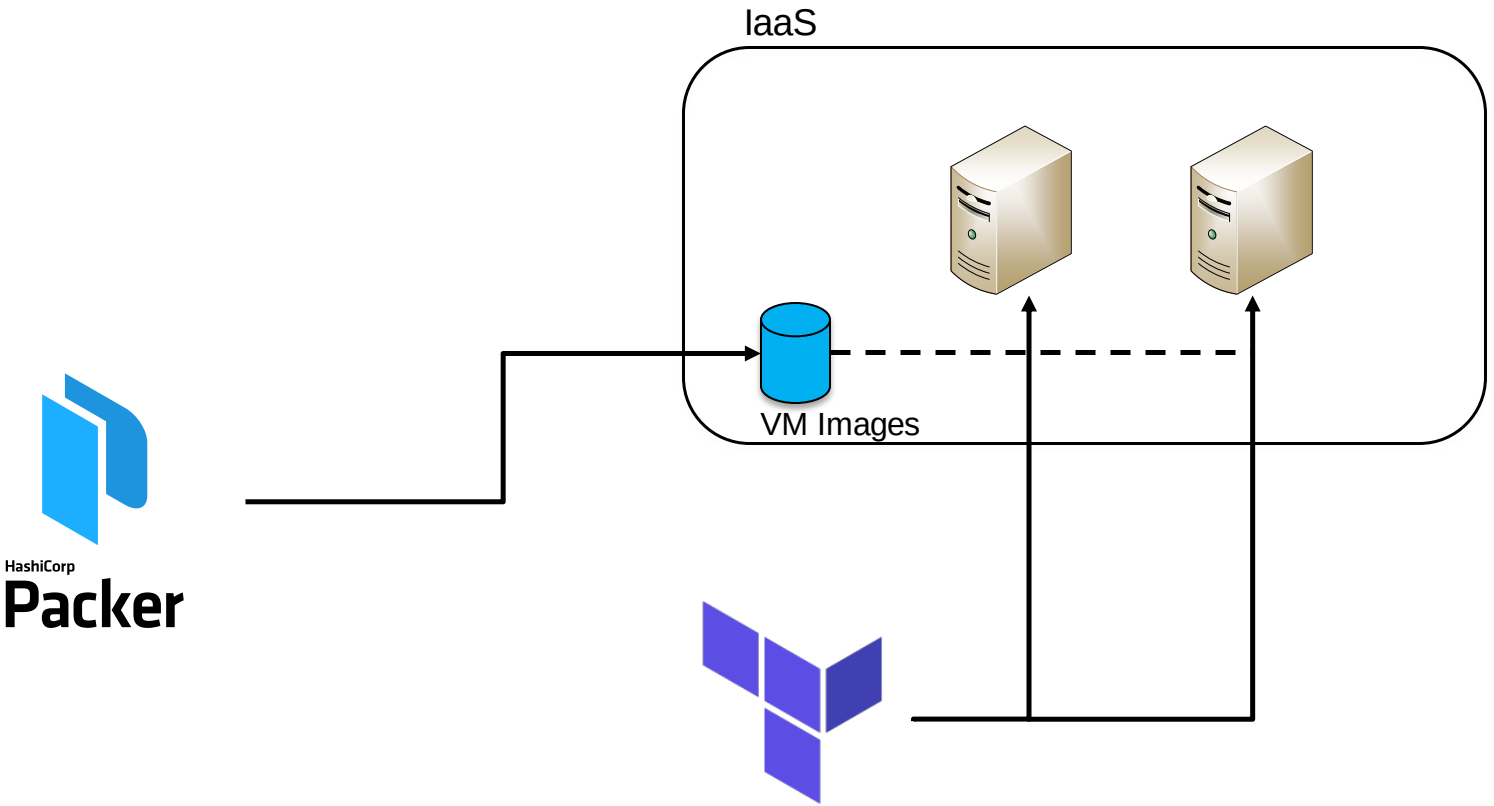
Agile Development

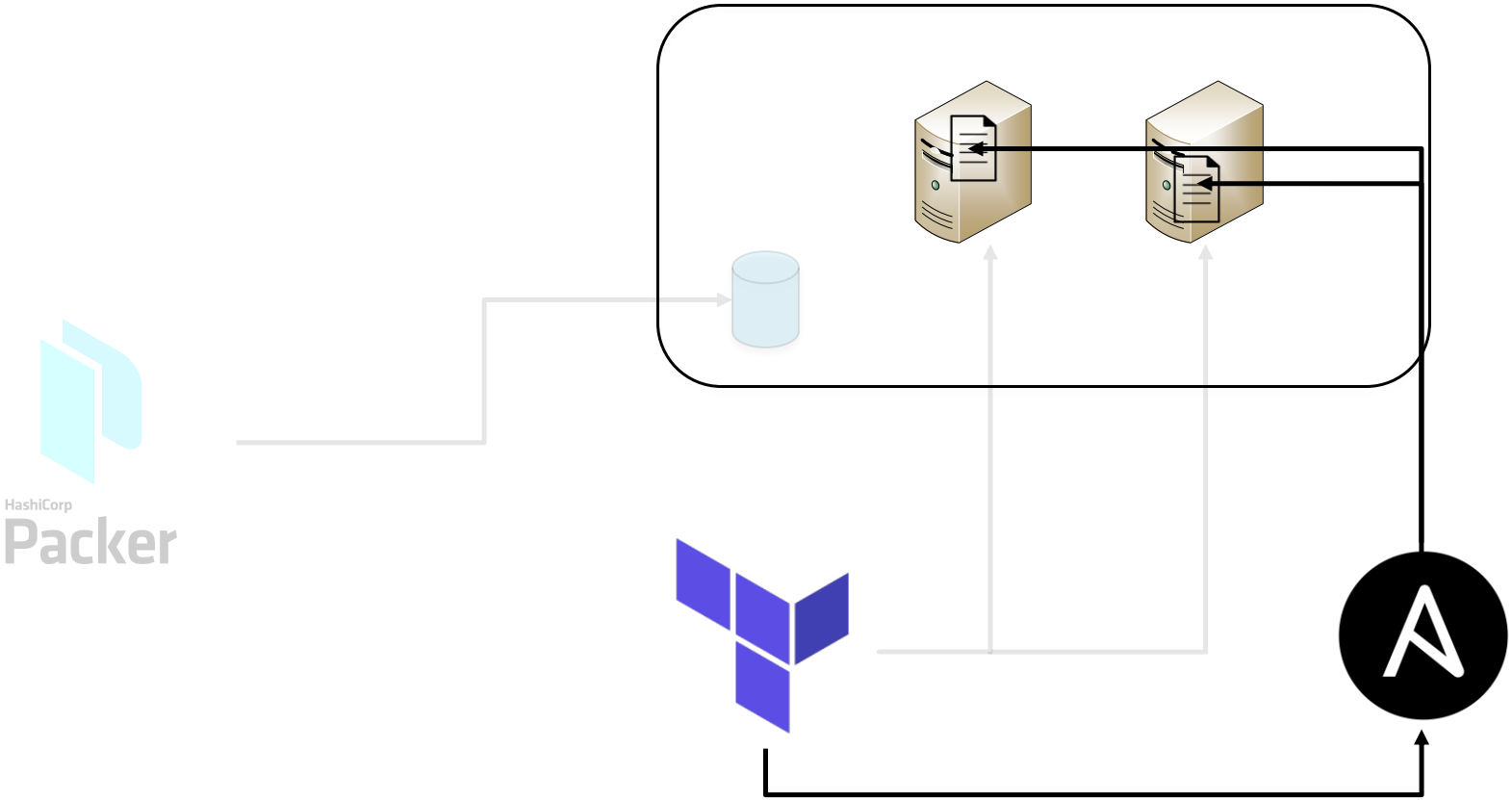
- Agile Methods focus on software development.
- DevOps focusses on software deployment.

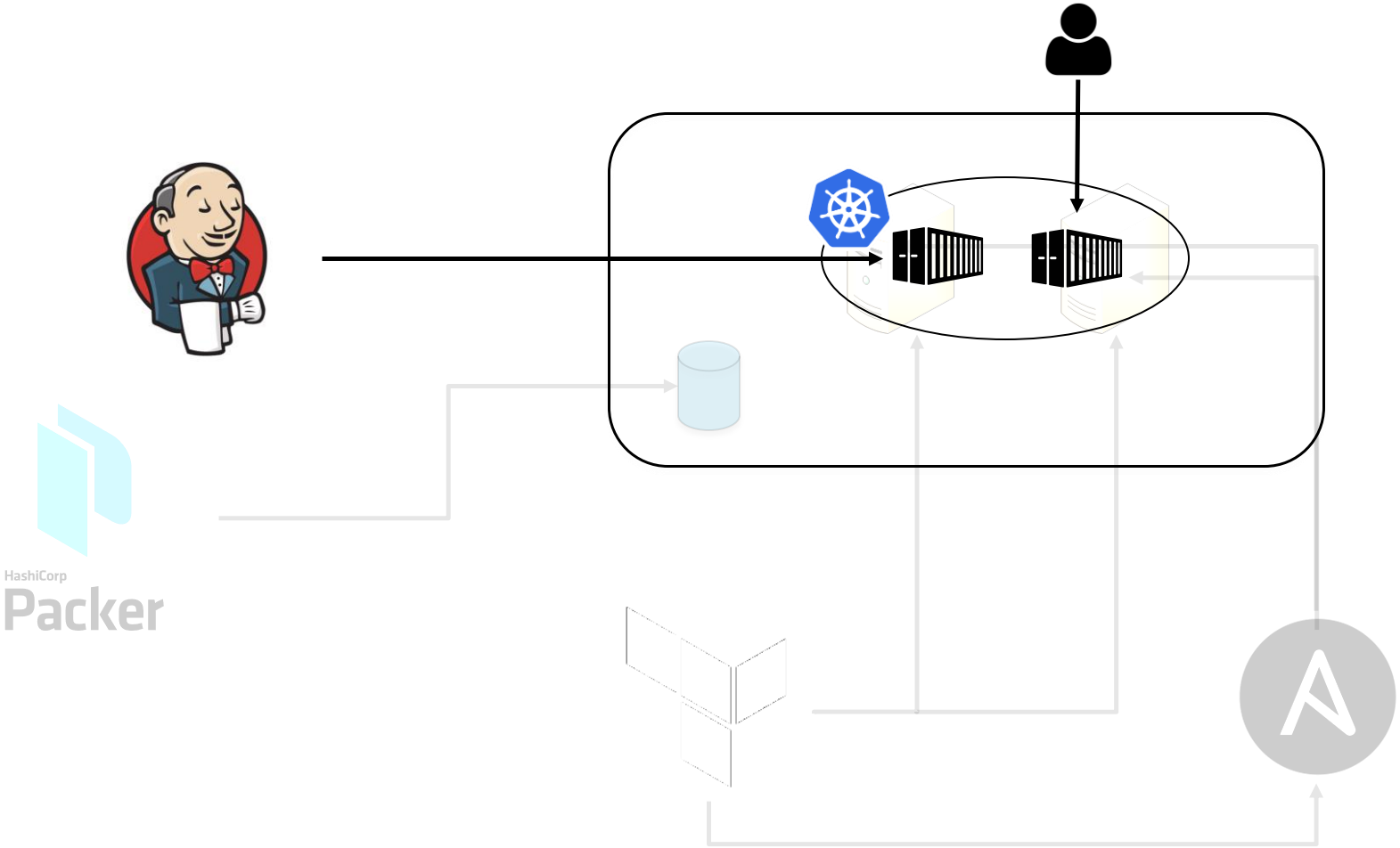
- Both share many approaches, ideas and tools!

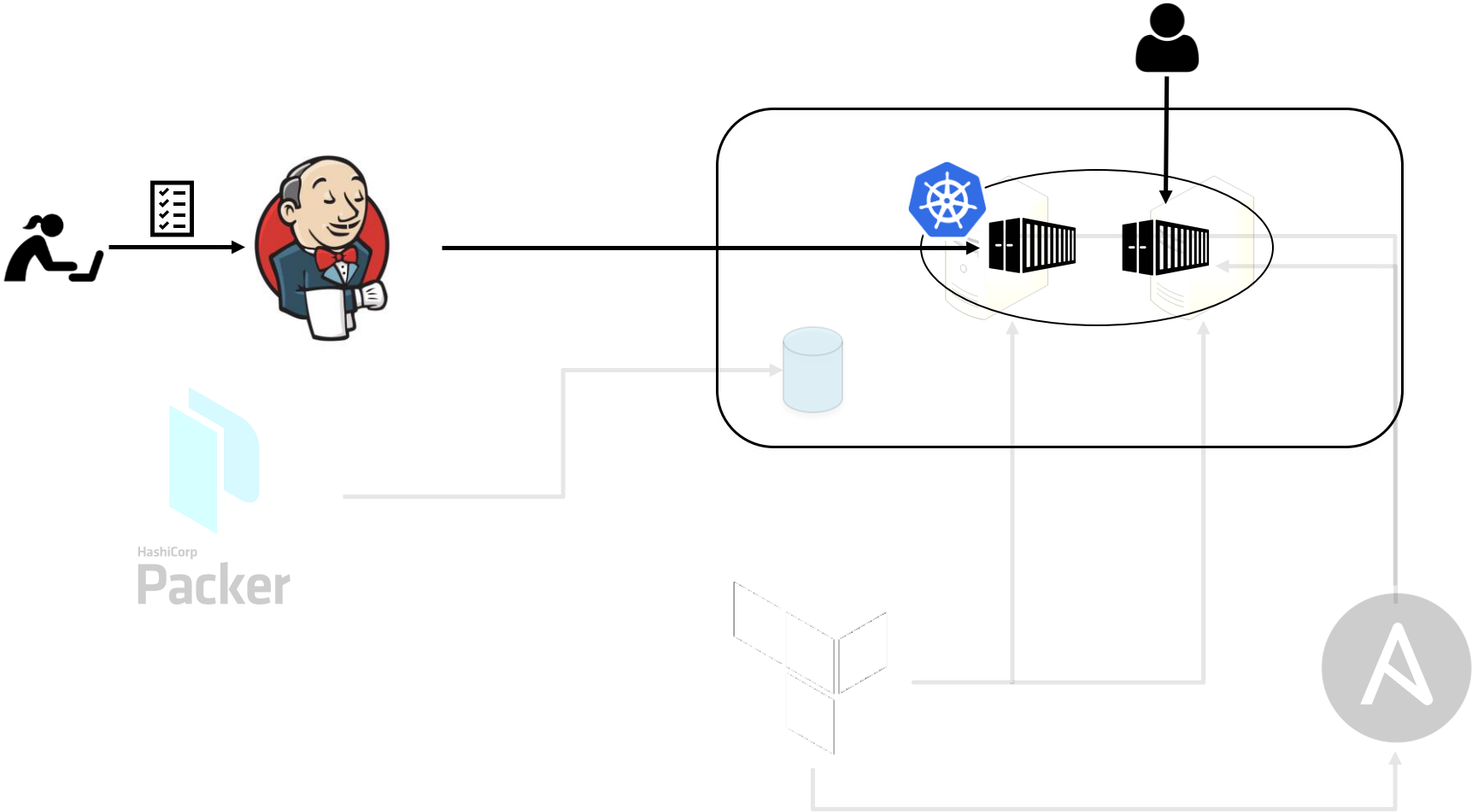
Bringing It All Together

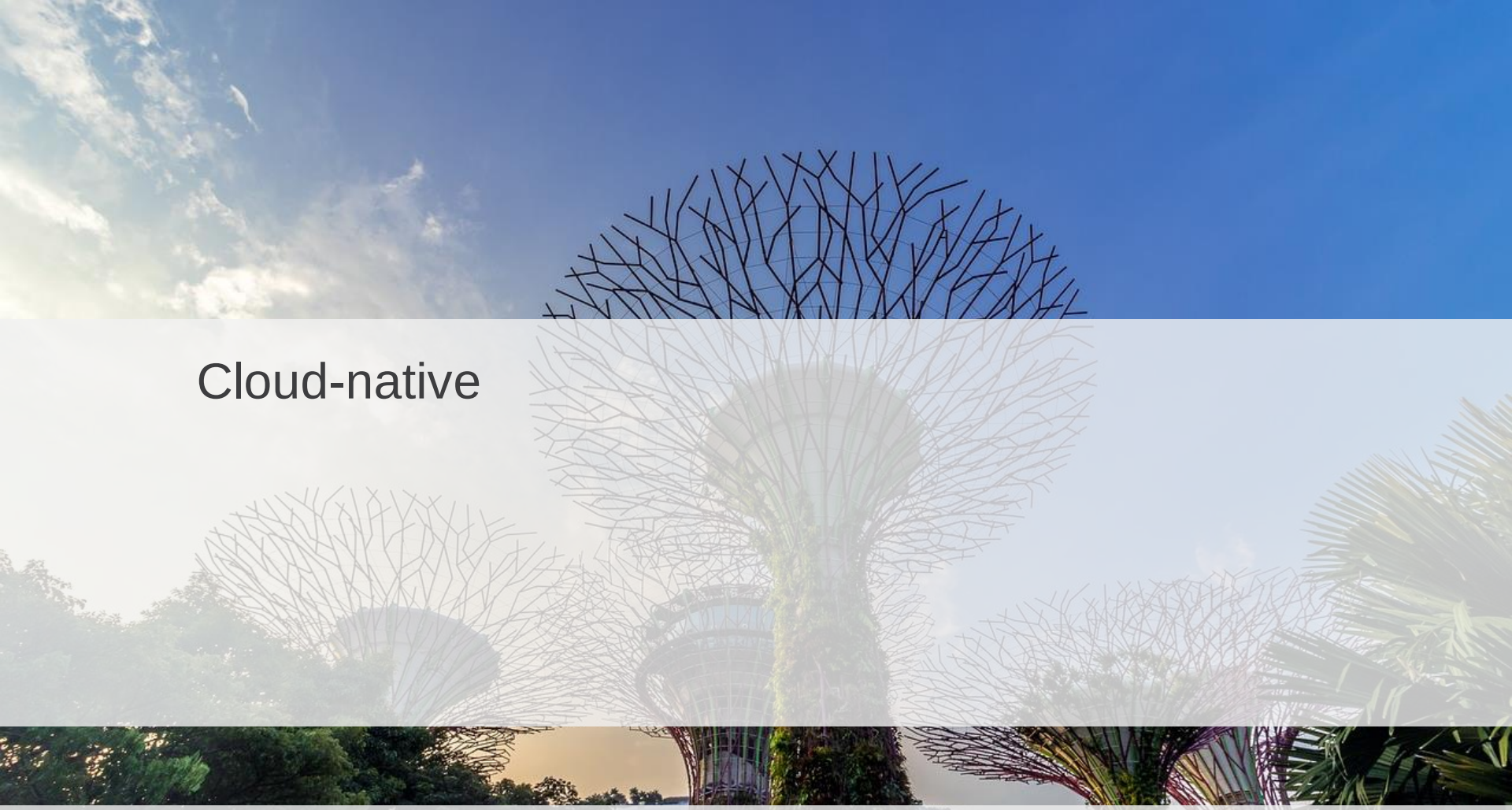
- **Agile Development**
  - produces software.
- **Continuous Delivery**
  - is a paradigm for development that each sprint (or even more granular tasks) must result in deployable software.
- **Continuous Deployment**
  - is the automated deployment of software to production.
- **DevOps**
  - is the approach to complement Agile Development with deployment/infrastructure/operational aspects and provide the technology required to deploy fast and often.







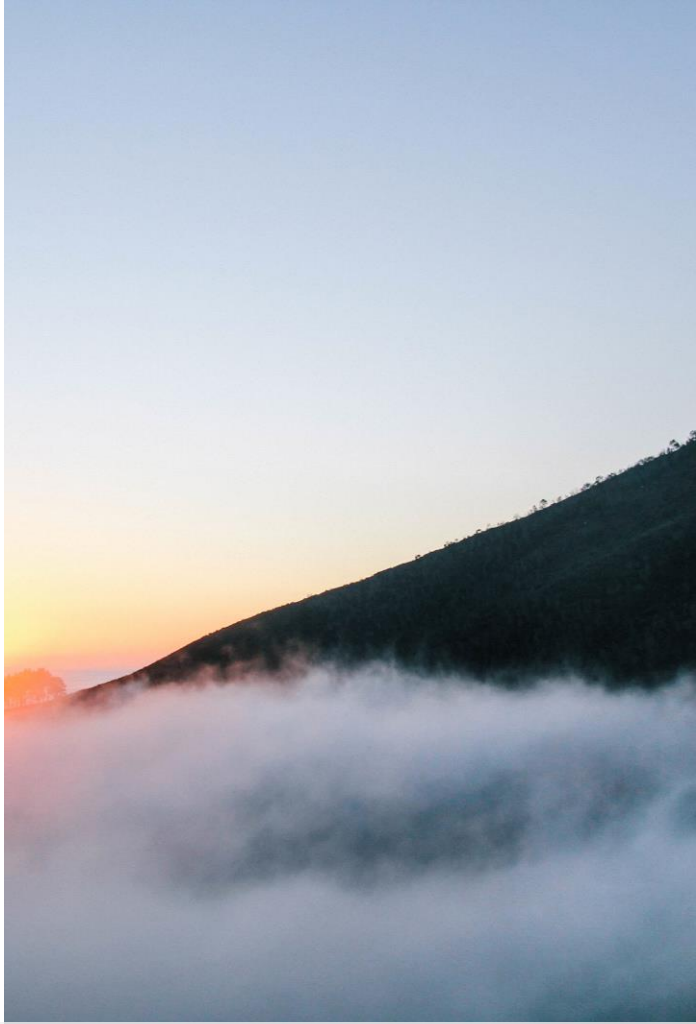




Cloud-native

Cloud-native?

- “Cloud native is a term used to describe container-based environments”
  - **Strong Disagree!**
- Our Take:
  - Happening in Application Design
  - Use standard services & best suited tools/languages for each job
  - Leveraging a fully orchestrate-able infrastructure
  - End-to-end responsible two pizza product teams
  - Offered services in turn fully orchestrate-able



Container?

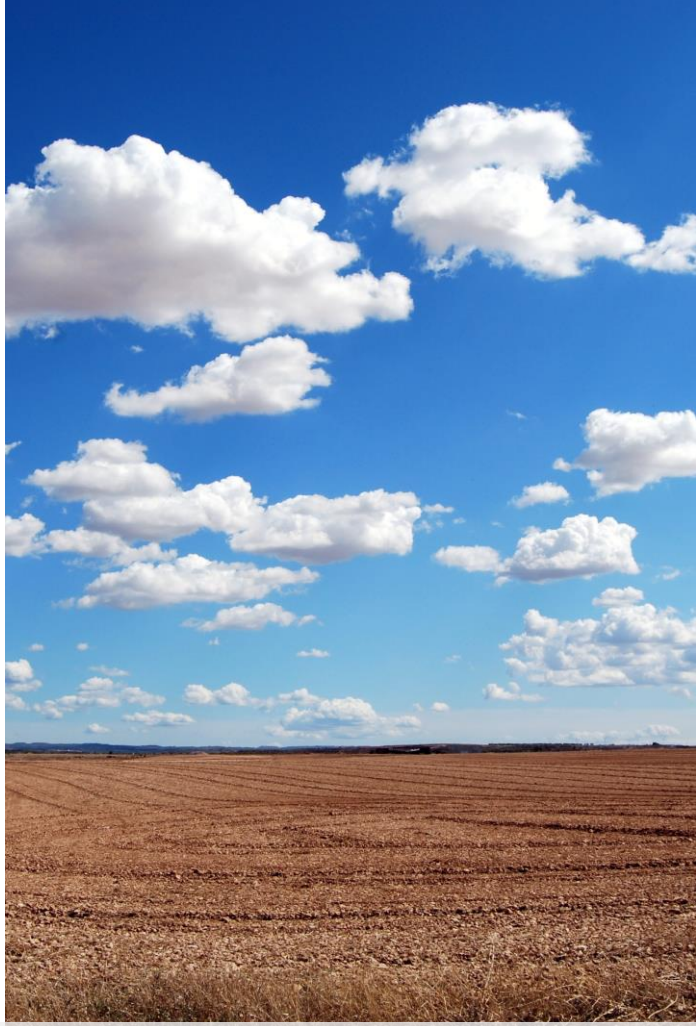
- For details: See Container content.
- **OS Container**
  - Group of processes isolated/controlled via OS functionality
- **Application Container**
  - One or more processes running in an **OS container** based on an **Application Container Image**
    - E.g. also “runtime instantiation of an application container image”
- **Application Container Image**
  - Portable base file system and configuration for an **OS container**.
    - Containing a self-contained application including all of its dependencies.

Container Advantages

- **Comprehensive Ecosystem:**
  - Strong software packaging, management, and distribution capabilities.
- **Self Sustainability:**
  - Container image contains everything an application needs to run.
- **Host Independence:**
  - A container runs the same way on any container host.
  - I.e. developer can spin up test environment on local computer that is identical to the prod environment.
    - Some pitfalls/additional requirements ;-)

Cloud-native
Application Design & Operation

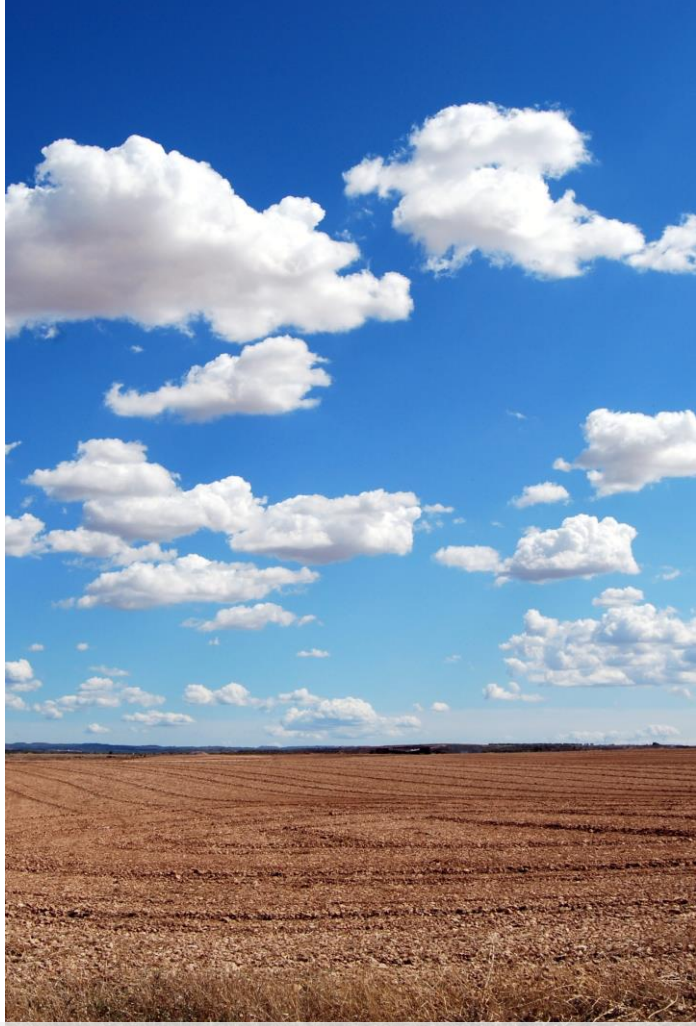
- Loose coupling - Microservices / FaaS
- Stateless application logic
- Persistence-aaS, Optimistic Locking
- Message- / Event-based communication
- CQRS, Event Sourcing, ...
- Automated deployment & operation
- Chaos Engineering



Requirements

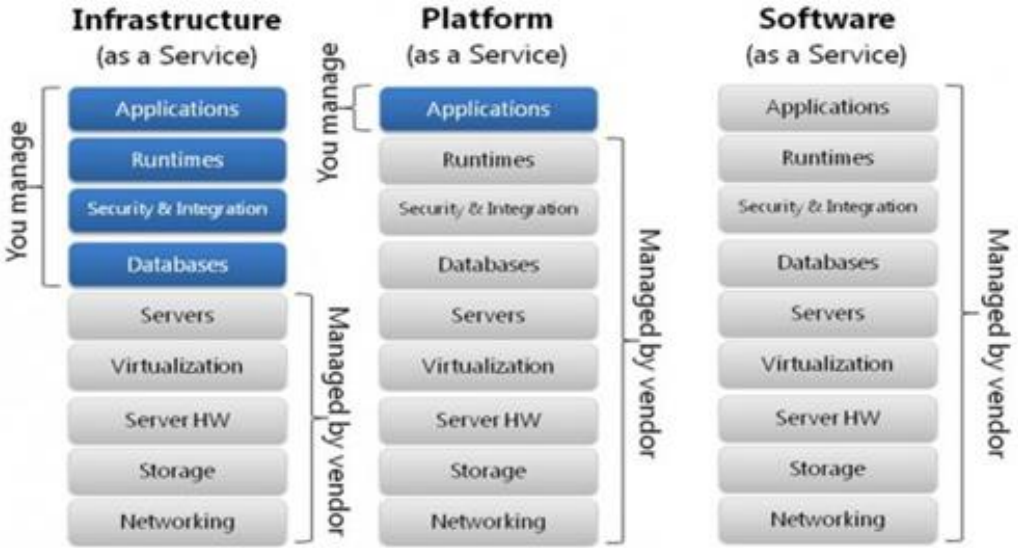
- Log Management
- Monitoring
- Service Discovery
- CI/CD'ability
- Persistence-as-a-service
- Secret / Config Management
- Inventory Management
- Network Functionality

➲ All of those controllable via API



Case Studies

Shared Responsibilities



Responsibility	On-Prem	IaaS	PaaS	SaaS
Data classification & accountability	■	■	■	■
Client & end-point protection	■	■	■	◪
Identity & access management	■	■	◪	◪
Application level controls	■	■	◪	□
Network controls	■	◪	□	□
Host infrastructure	■	◪	□	□
Physical security	■	□	□	□

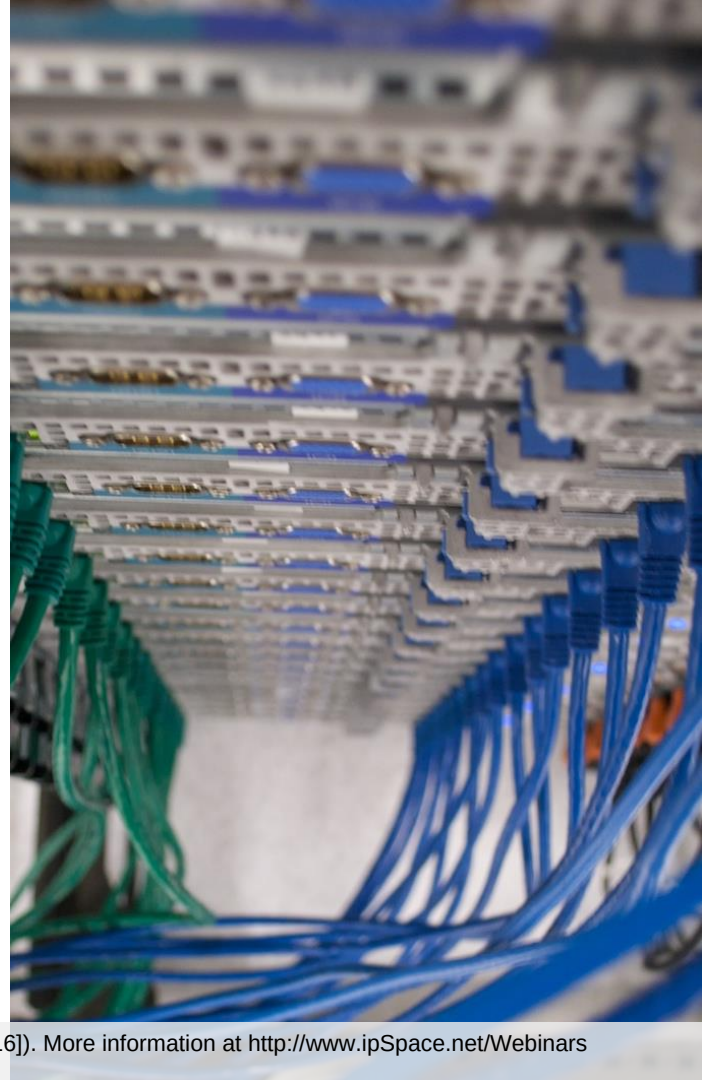
■ Cloud Customer ◪ Cloud Provider

Shared Responsibilities

- You are still responsible for different operational aspects in Cloud environments.
- Important to be clear about those.
- Fulfilling those operational aspects should leverage Cloud functionality!
  - E.g. automating your tasks.

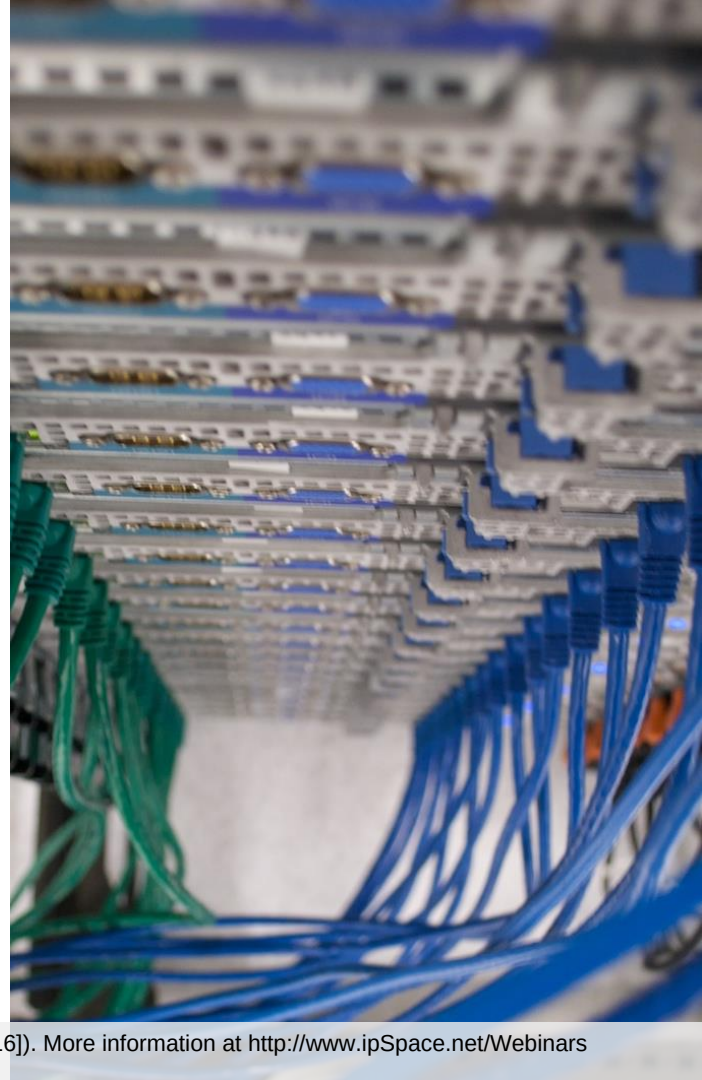
Shared Responsibilities

- Digression: Shared responsibility models are very relevant for on-premise environments as well!
  - Barely any environment has them…
- Example: Network filtering.



Shared Responsibilities: Network Filtering

- Central/provider responsibility:
  - Operation of firewalls, implementation of underlay network/network transport.
  - Expertise can be built centralized
- Tenant responsibility:
  - Maintenance of proper filtering rules.
  - Expertise requires application domain knowledge!
    - Cannot efficiently be centralized.

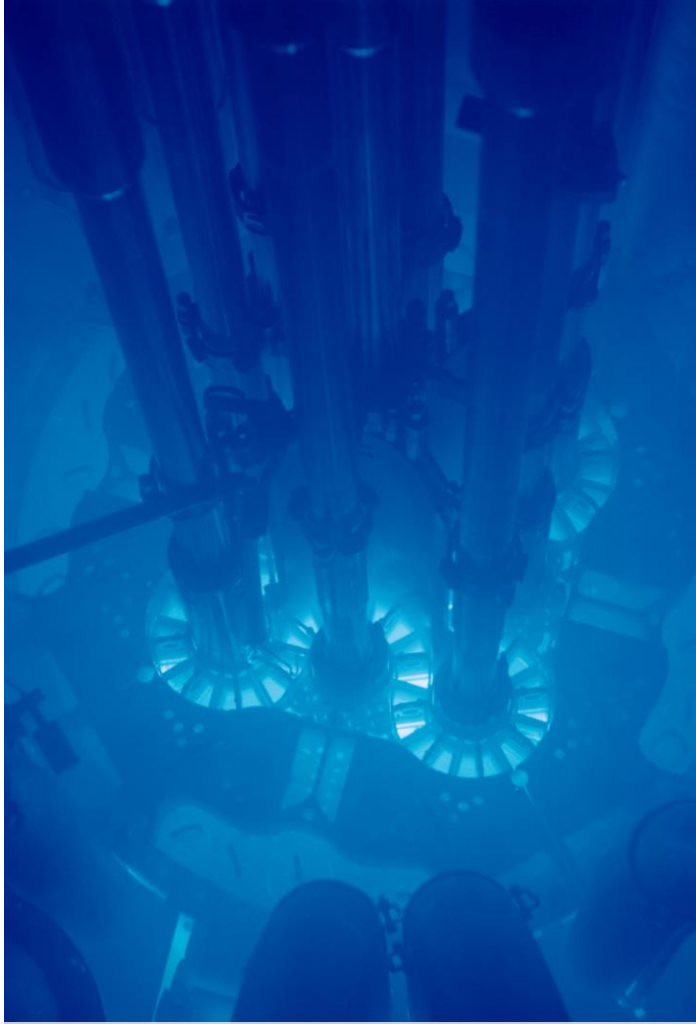


IaaS/IaC: PacketWars™

- “PACKETWARS(TM) is an intense, real-time **Cyber Operations Simulation**. Unlike other “capture the flag” games, the battlegrounds featured in PACKETWARS(TM) are designed to simulate **real-world** engagements and campaigns.” – packetwars.com
- Traditionally:
  - Build on-premise infra with servers, switches, VMs – 2 to 8 hour effort before games.

Moving to the Cloud

- Develop infrastructure as IaC
- Perfect use case:
  - Have pre-built images that are immutable.
  - Clear network communication relationships.
    - IaC frameworks can be limiting here if complex relationships are required.
  - Scaling from 2 to 20 teams within minutes by changing a variable.
- Limitations:
  - Individual configs or similar per team still requires more elaborate script surroundings/scaffolding.
    - => Use the right tool for the job ☺



Demo

IaaS IaC

Migrating into the cloud

- CouchDB - "**Seamless** multi-master sync, that scales from **Big Data** to **Mobile,** with an **Intuitive** HTTP/JSON API and designed for **Reliability.**“
- Migration from bare metal onto virtualised metal
- AWS EC2 & AWS EBS

Running out of space

- CouchDB uses an append-only data model
- Compaction needed for clean-up
- Compaction needs space
- …
- Guess what‘s next...

Demo

EBS upsize

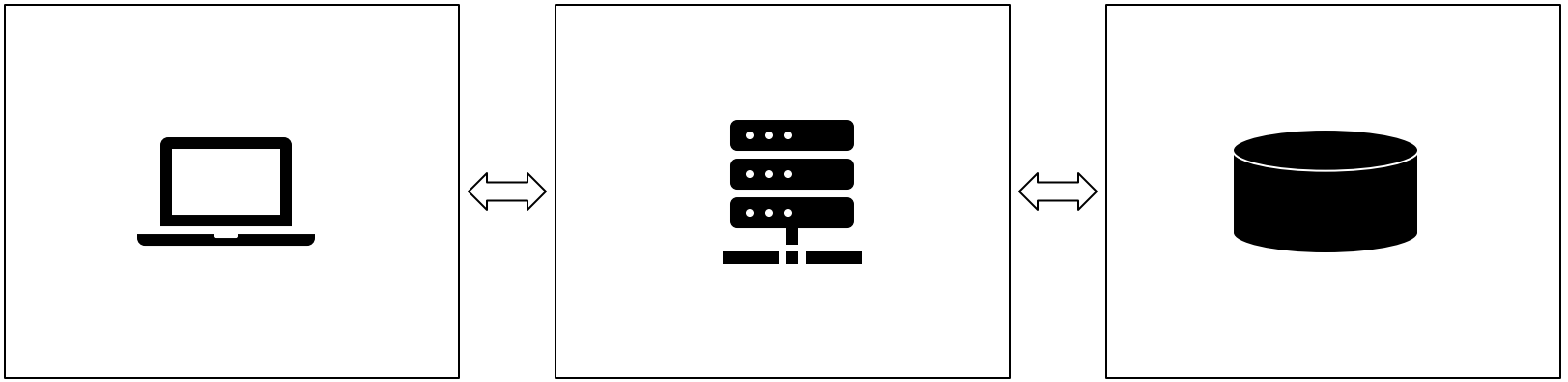
Yay, IaaS powers!

- Sizing up an active, virtual hard disk in normal operation
- On the fly vertical / horizontal scaling
- A lot of operational tooling
- Backup, Monitoring, Logging, Network ACL & Firewall as a Service

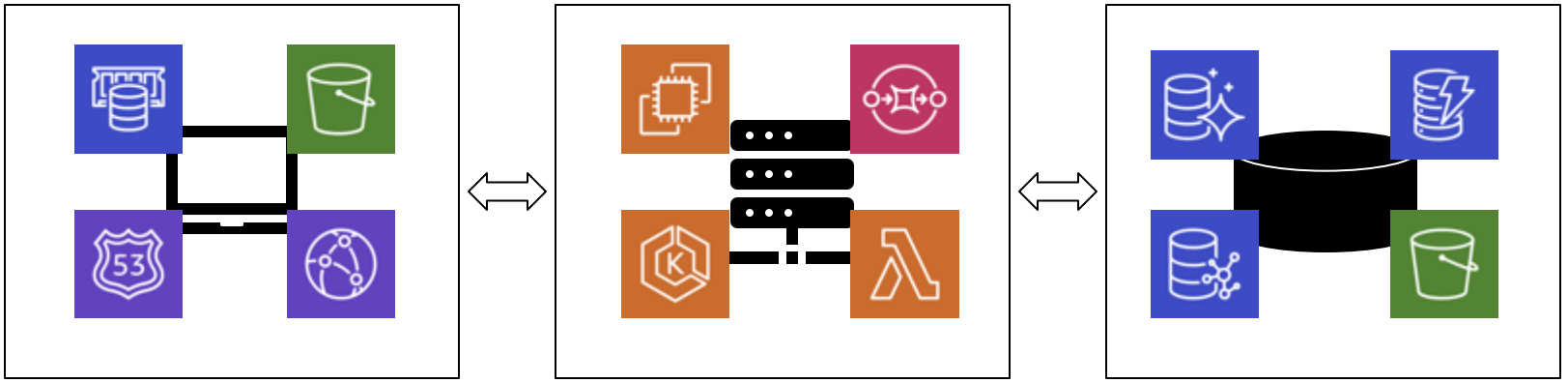
But wait…

- Why would you want to operate a self-maintained database?
- Managed services for persistence
- Updates, Upgrades, Scaling, High-Availability
- Focus on things that you can do best
- Let others do things that would steal that focus

3-tier architecture: AWS Style



3-tier architecture: AWS Style



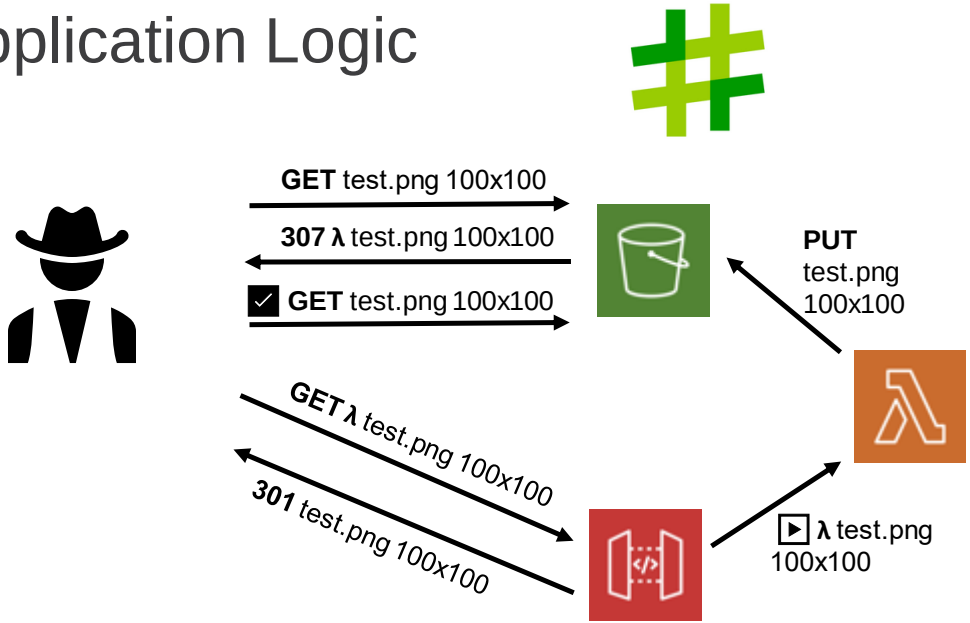
Serverless Image Handling

- API Gateway for orchestration and routing
- Image processing via Lambda
- Persistence of image data via S3
- Support dynamic image resizing

AWS API Gateway

- API as a Service
- REST or WebSocket APIs
- Import Swagger or OpenAPI definitions
- Transform Requests & Responses
- API Versioning, Monitoring, Caching

Application Logic



Demo

Serverless

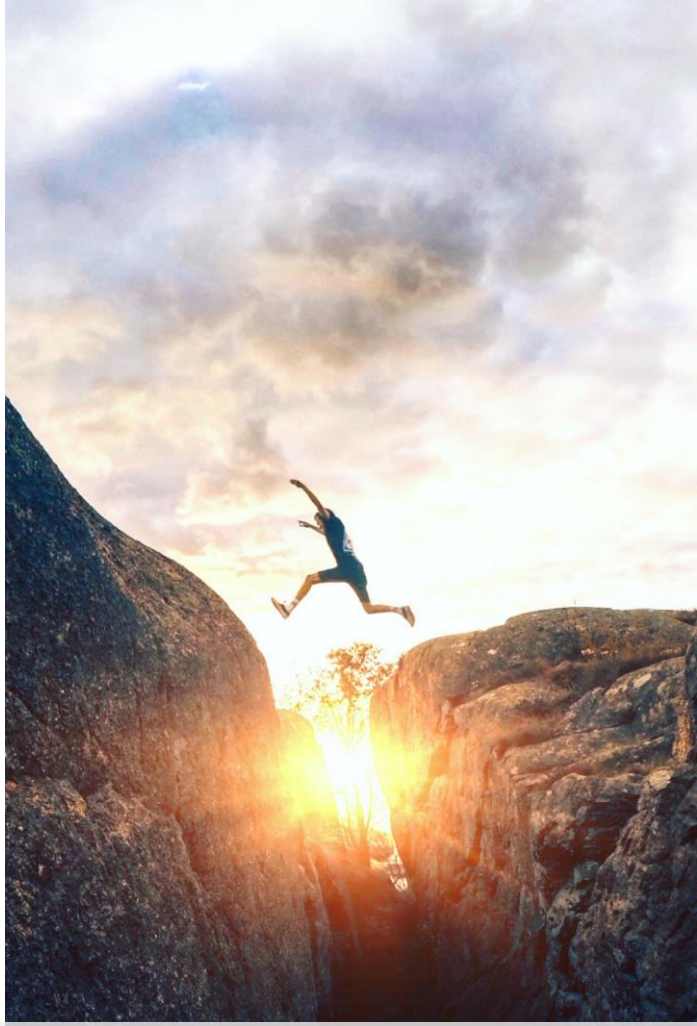
Goodbye, Microservices?

- Serverless & “Serviceless” applications
- Containers vs Serverless
- Trade-off decision
- Hybrid architectures

Summary

- “Cloud” can mean many different things
- Migrating to the Cloud (at least in a somewhat Cloud-native way) requires fundamental IT re-design
  - And not just using cloud resources instead of on-prem resources.

Thank you for your Attention!



matthias.luft@rational-security.io

@uchi_mata