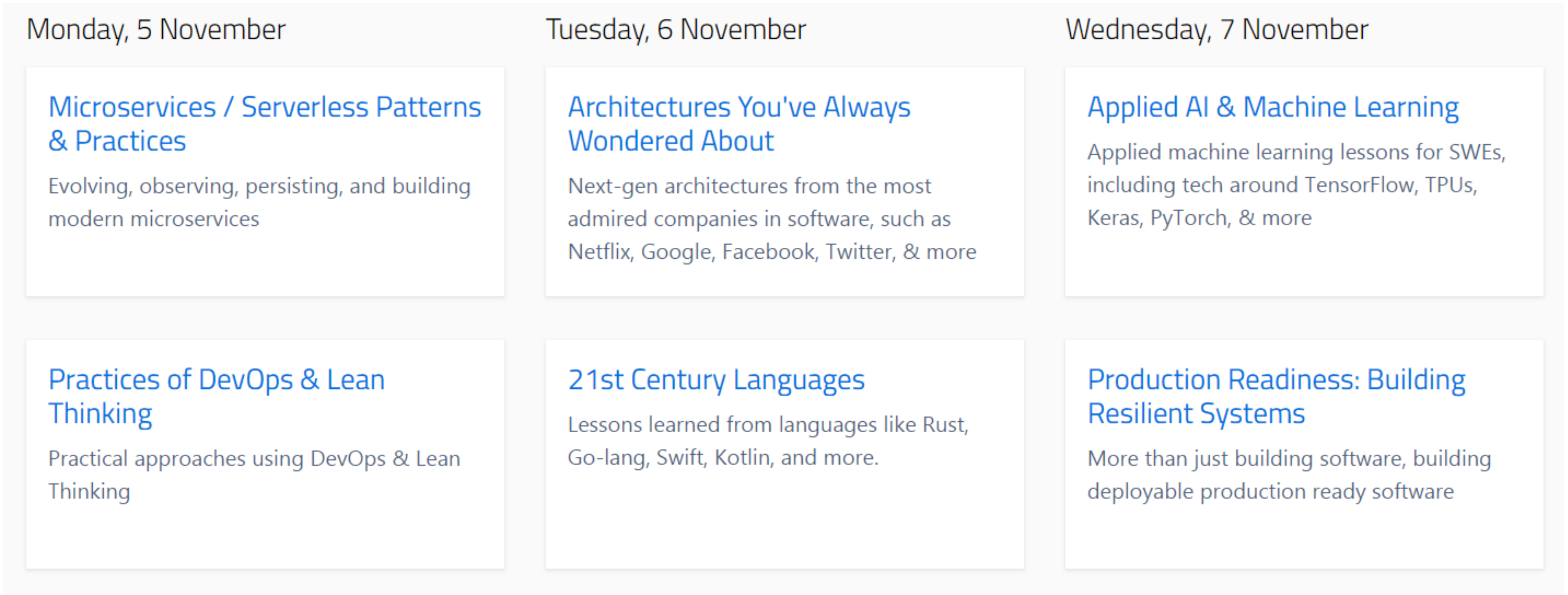
QCon San Francisco 2018

Shared by Ken Aung

November 20, 2018

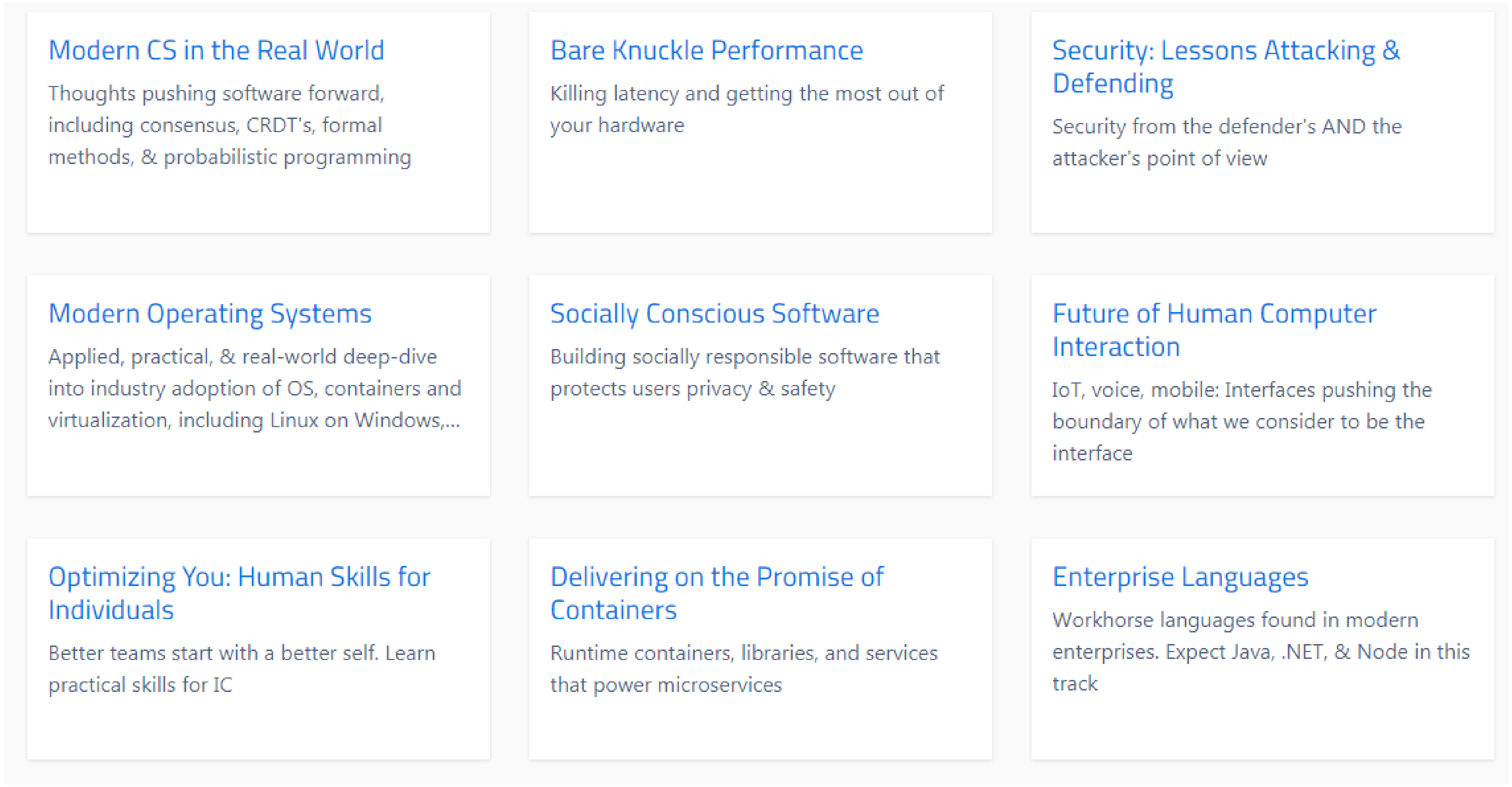
QCon SF by InfoQ

- Bleeding-edge Software Developer Conference for the Enterprise
- 18 editorial tracks across 3 days
- 140+ practitioner speakers



Monday, 5 November	Tuesday, 6 November	Wednesday, 7 November
Microservices / Serverless Patterns & Practices Evolving, observing, persisting, and building modern microservices	**Architectures You've Always Wondered About** Next-gen architectures from the most admired companies in software, such as Netflix, Google, Facebook, Twitter, & more	**Applied AI & Machine Learning** Applied machine learning lessons for SWEs, including tech around TensorFlow, TPUs, Keras, PyTorch, & more
Practices of DevOps & Lean Thinking Practical approaches using DevOps & Lean Thinking	**21st Century Languages** Lessons learned from languages like Rust, Go-lang, Swift, Kotlin, and more.	**Production Readiness: Building Resilient Systems** More than just building software, building deployable production ready software

Tracks



Modern CS in the Real World

Thoughts pushing software forward, including consensus, CRDT's, formal methods, & probabilistic programming

Bare Knuckle Performance

Killing latency and getting the most out of your hardware

Security: Lessons Attacking & Defending

Security from the defender's AND the attacker's point of view

Modern Operating Systems

Applied, practical, & real-world deep-dive into industry adoption of OS, containers and virtualization, including Linux on Windows,...

Socially Conscious Software

Building socially responsible software that protects users privacy & safety

Future of Human Computer Interaction

IoT, voice, mobile: Interfaces pushing the boundary of what we consider to be the interface

Optimizing You: Human Skills for Individuals

Better teams start with a better self. Learn practical skills for IC

Delivering on the Promise of Containers

Runtime containers, libraries, and services that power microservices

Enterprise Languages

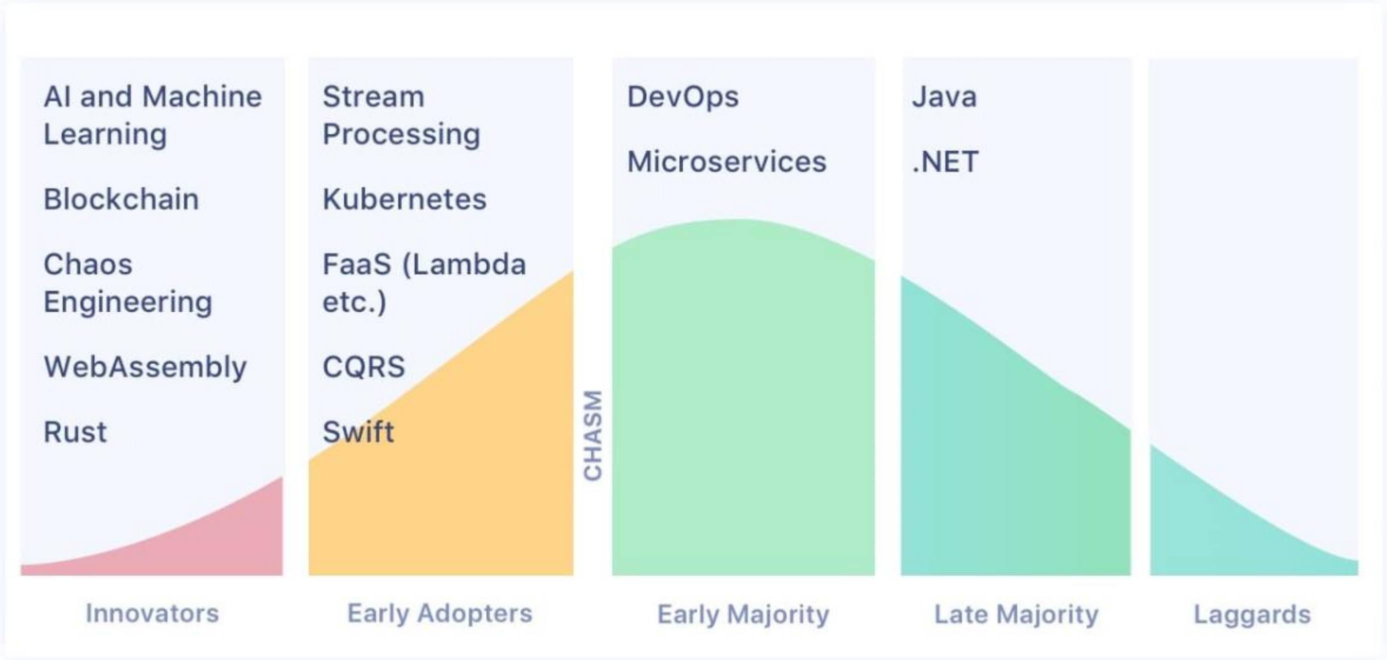
Workhorse languages found in modern enterprises. Expect Java, .NET, & Node in this track

InfoQ

"**Accelerating** the software side of human technological progress"

"**Stay ahead** of the adoption curve"

Technology Adoption Curve - May 2018



This image is adapted from Geoffrey Moore's book Crossing the Chasm

References

Java

- https://www.infoq.com/java

QCon

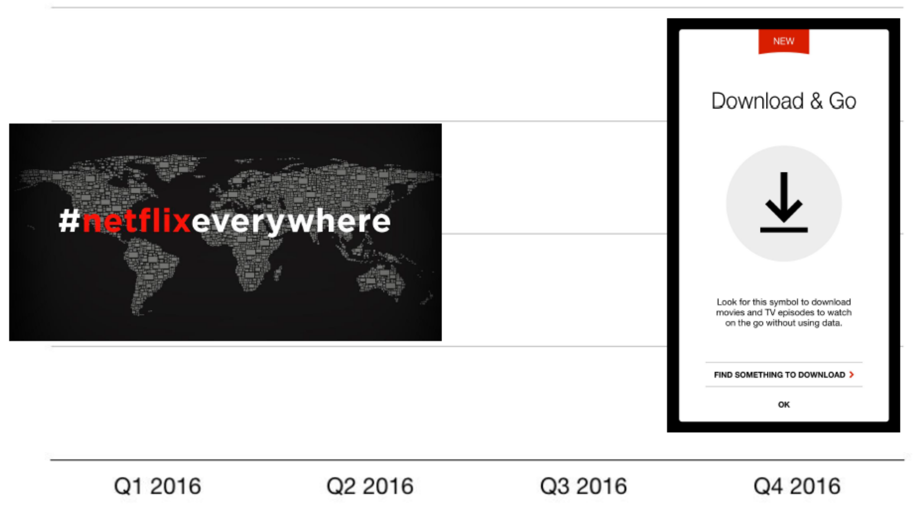
- https://www.infoq.com/qcon/
- https://qconsf.com/volunteers

Netflix Play API

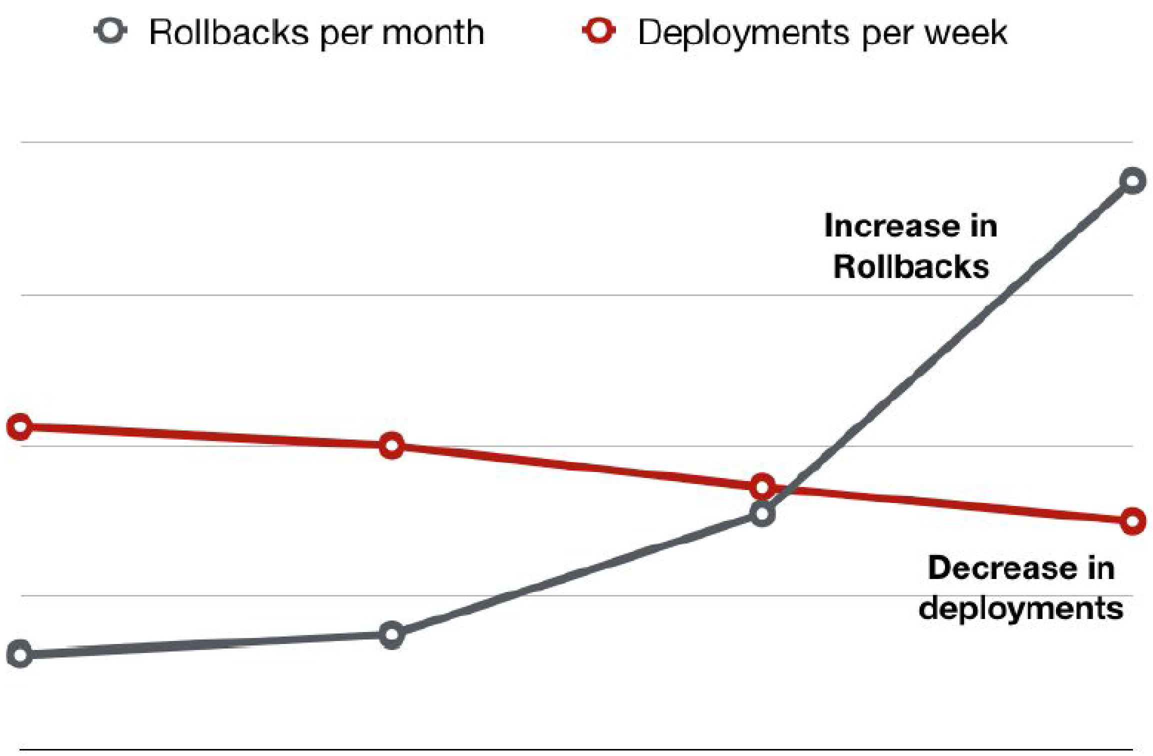
Why we built an Evolutionary Architecture

Suudhan Rangarajan (@suudhan)
Senior Software Engineer

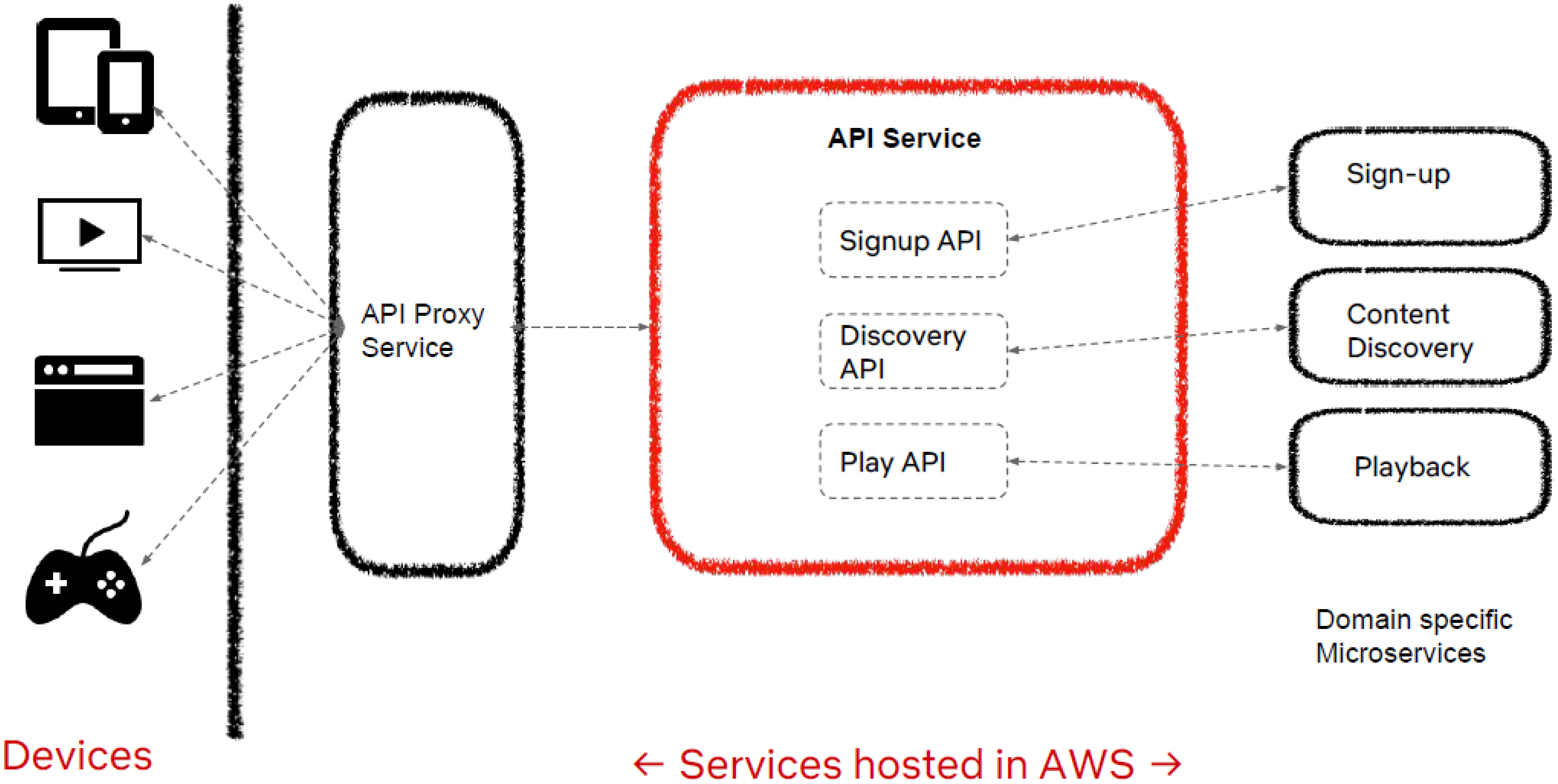
NETFLIX

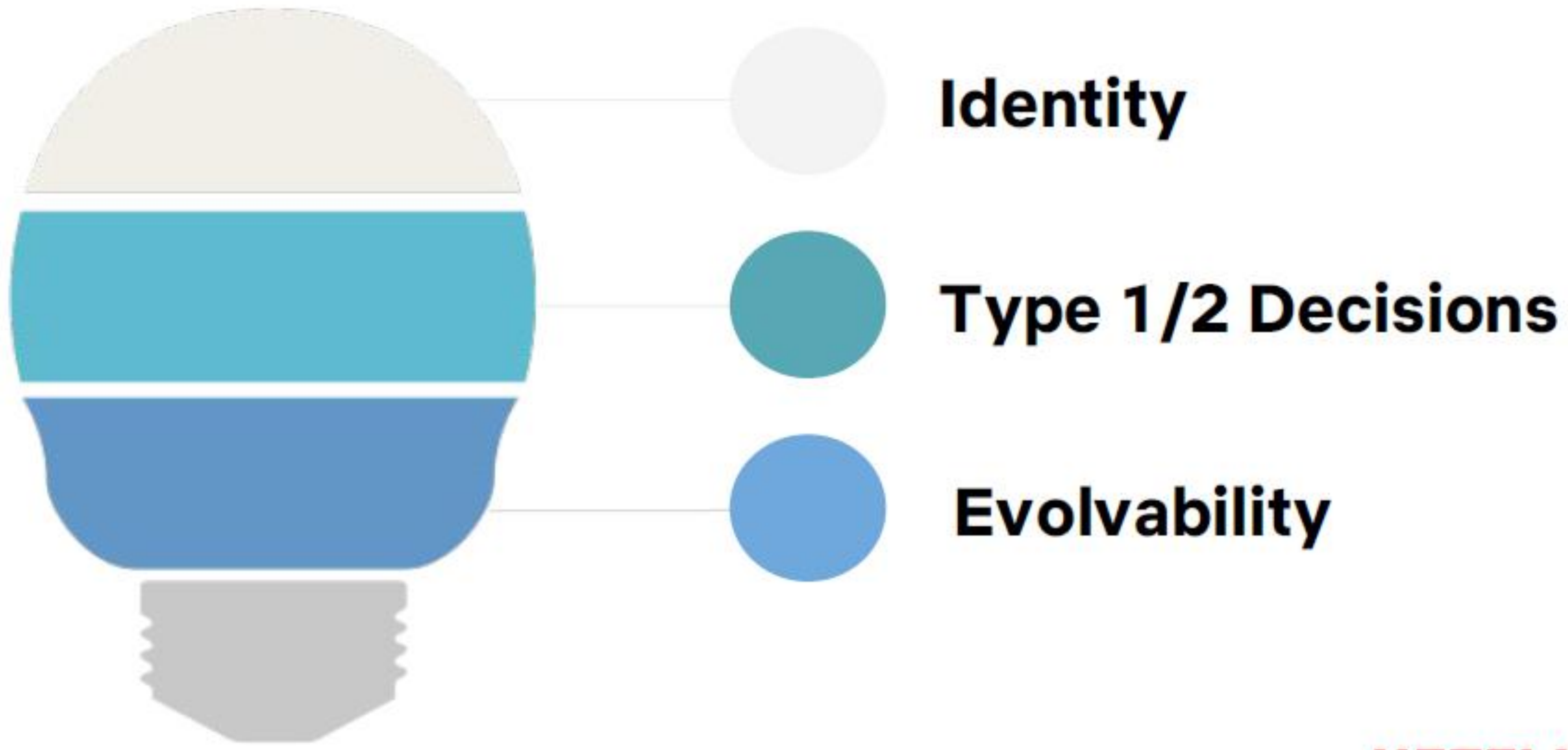


Look for this symbol to download movies and TV episodes to watch on the go without using data.

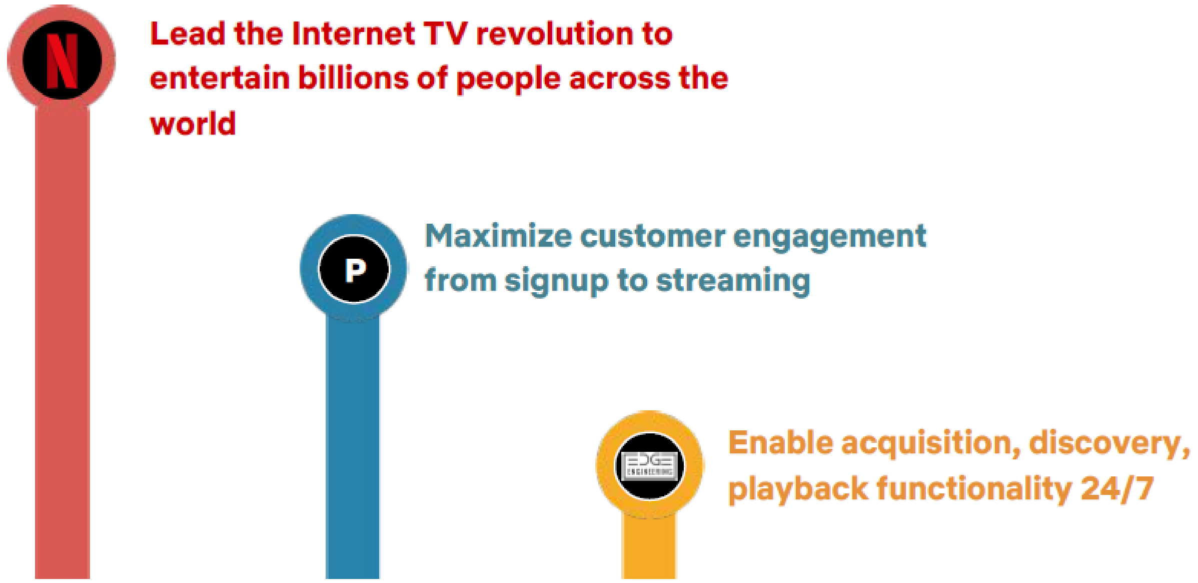


Previous Architecture




Identity
Type 1/2 Decisions
Evolvability

Start with WHY: Ask why your service exists

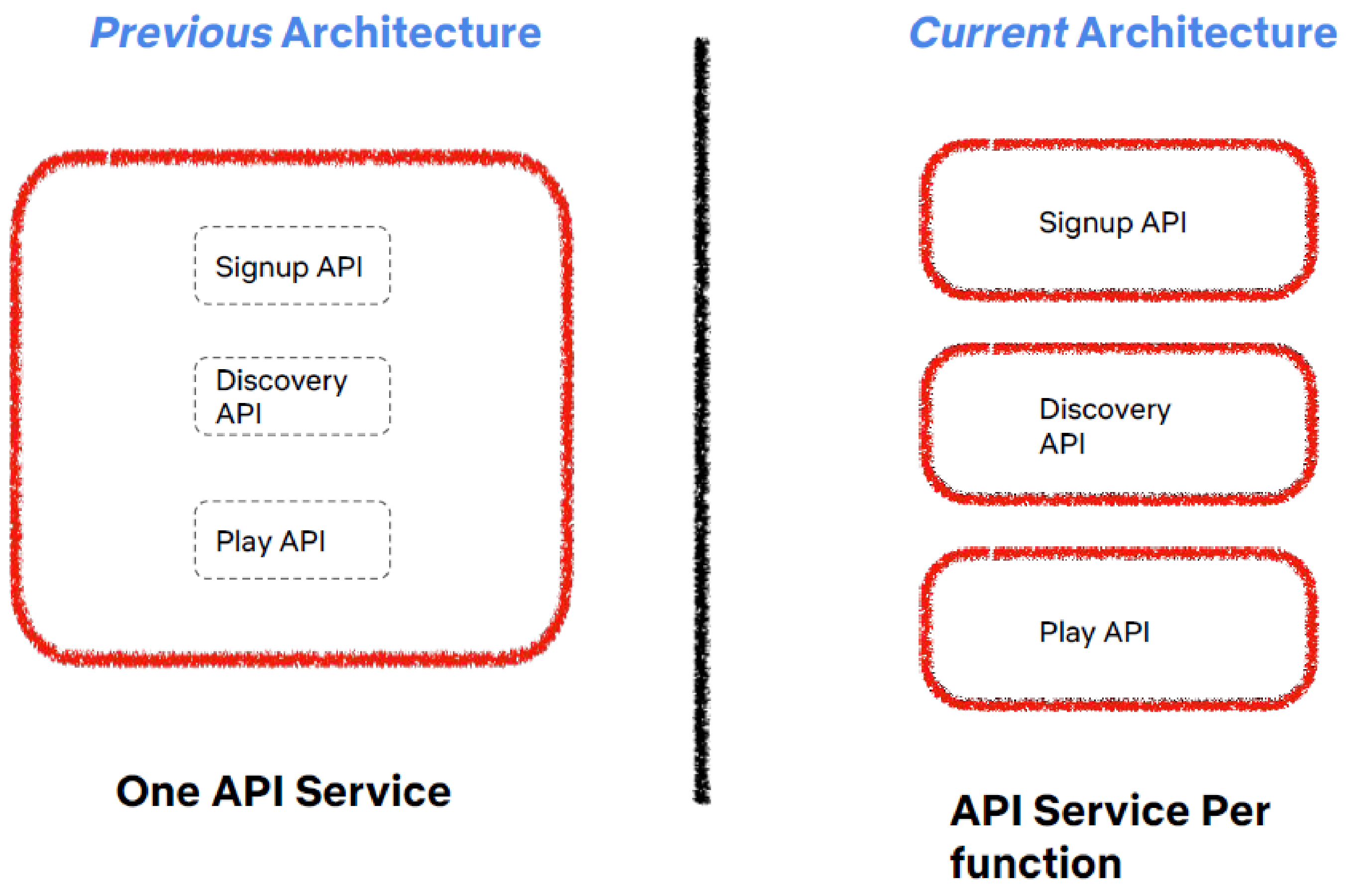

Lead the Internet TV revolution to entertain billions of people across the world

Maximize customer engagement from signup to streaming

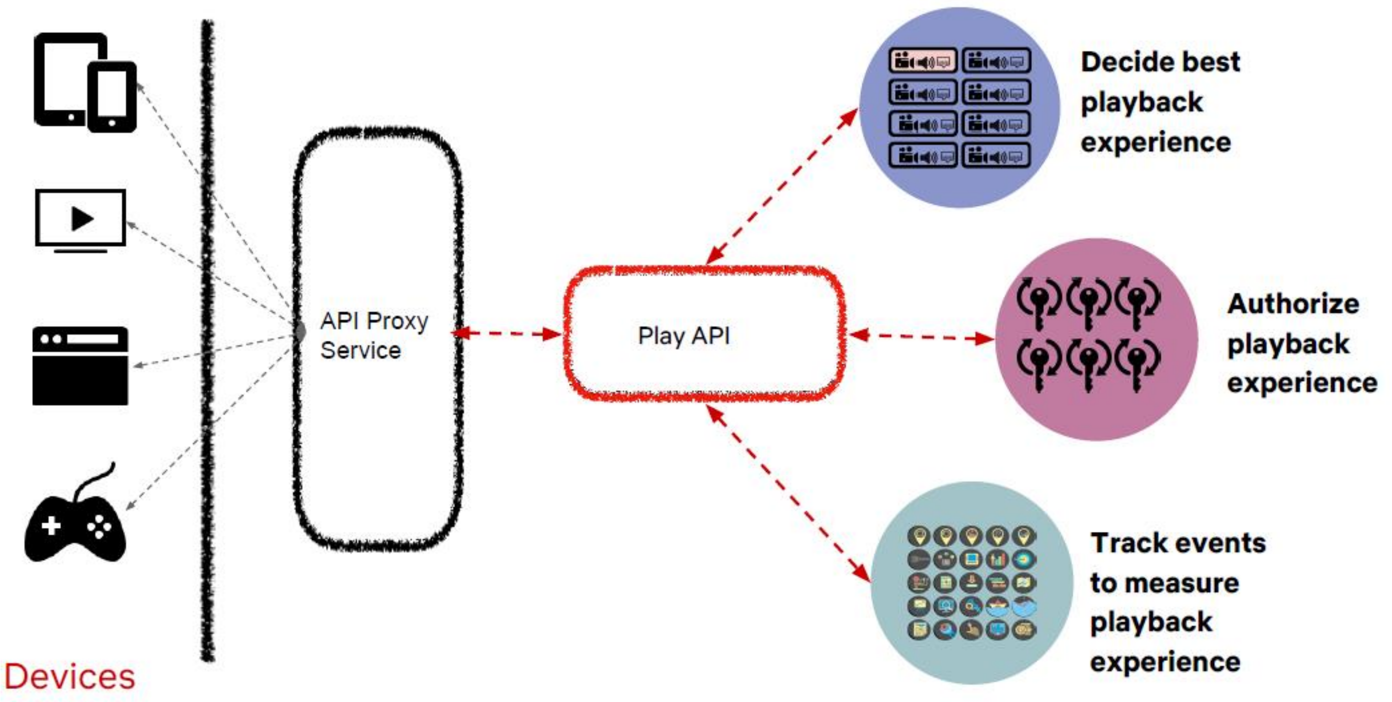
Enable acquisition, discovery, playback functionality 24/7

API Identity: Deliver Acquisition, Discovery and Playback functions with high availability

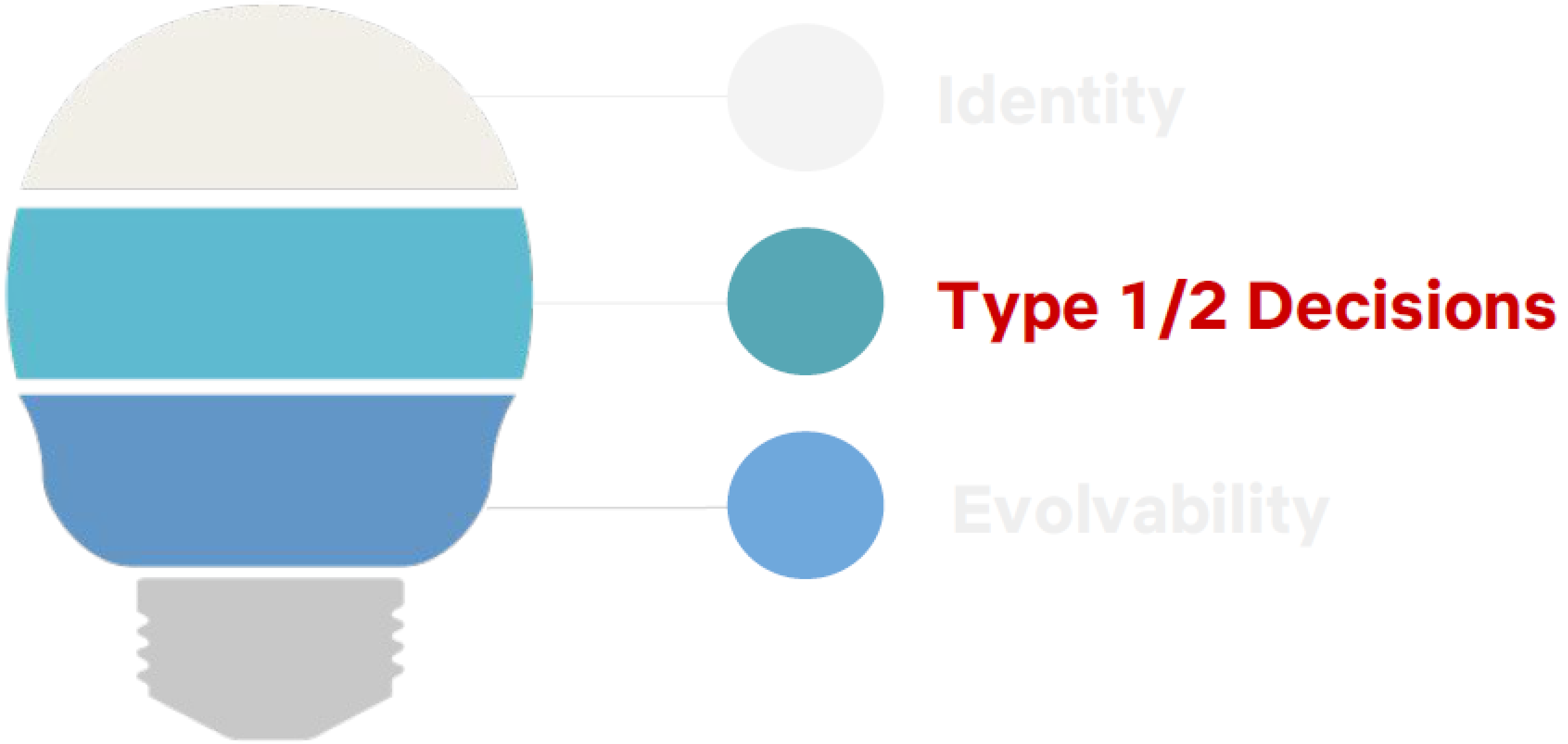
Single Responsibility Principle: Be wary of multiple-identities rolled up into a single service



Play API Identity: Orchestrate Playback Lifecycle with stable abstractions


API Proxy Service
Play API
Decide best playback experience
Authorize playback experience
Track events to measure playback experience
Devices

Guiding Principle: We believe in a simple singular identity for our services. The identity relates to and complements the identities of the company, organization, team and its peer services



NETFLIX

“Some decisions are consequential and irreversible or nearly irreversible – one-way doors – and these decisions must be made methodically, carefully, slowly, with great deliberation and consultation [...] We can call these Type 1 decisions…”

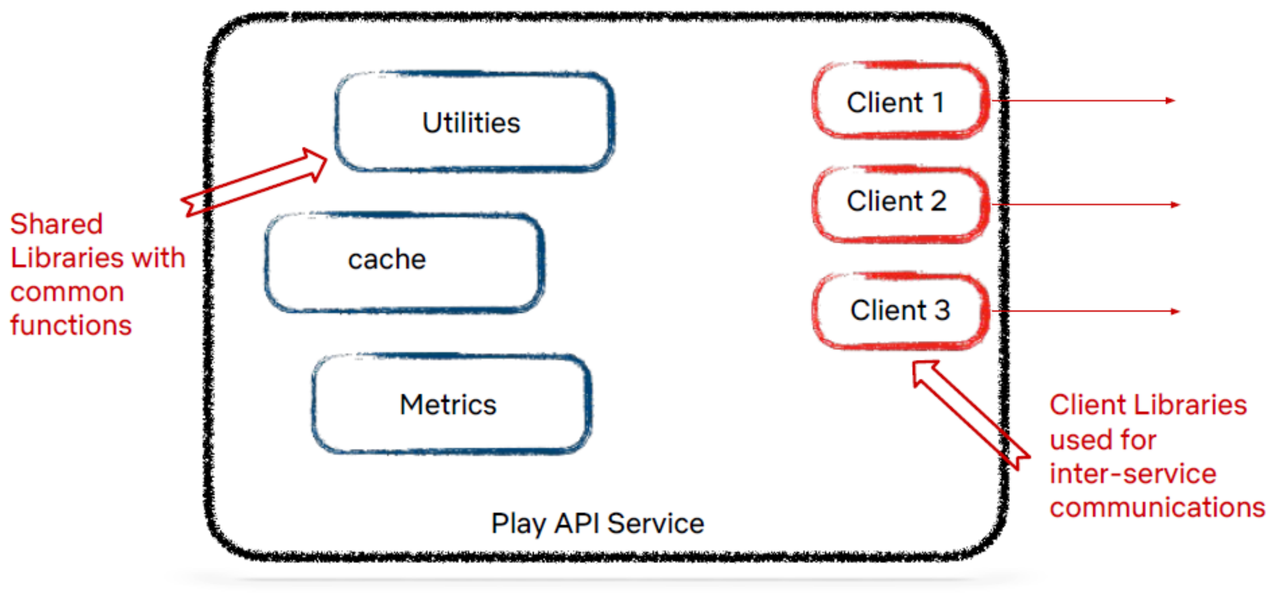
“...But most decisions aren’t like that – they are changeable, reversible – they’re two-way doors. If you’ve made a suboptimal Type 2 decision, you don’t have to live with the consequences for that long [...] Type 2 decisions can and should be made quickly by high judgment individuals or small groups.”

Quote from Jeff Bezos

Three Type 1 Decisions to Consider

- Appropriate Coupling
- Synchronous & Asynchronous
- Data Architecture

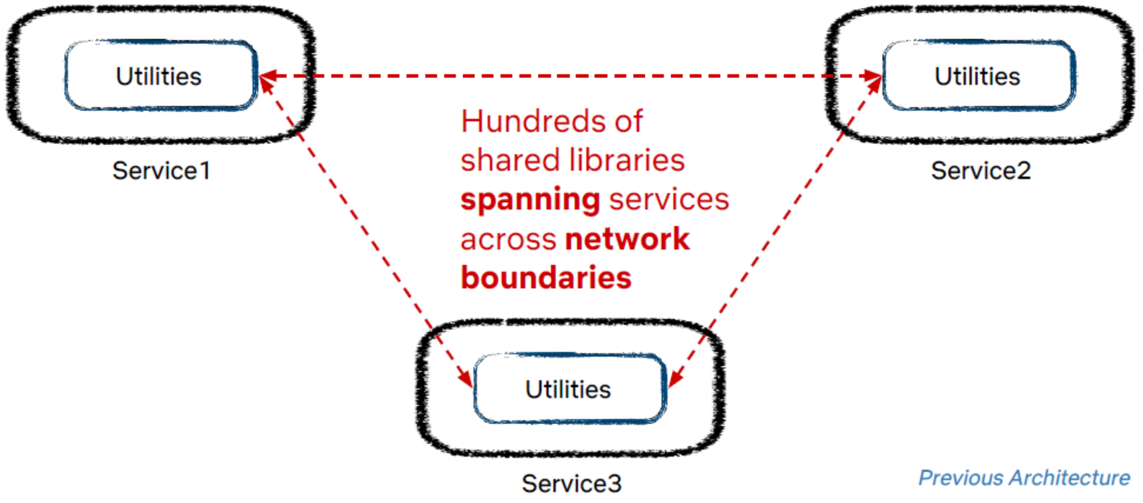
Two types of Shared Libraries


Utilities
cache
Metrics
Client 1
Client 2
Client 3
Play API Service
Shared Libraries with common functions
Client Libraries used for inter-service communications

"The evils of too much coupling between services are far worse than the problems caused by code duplication"

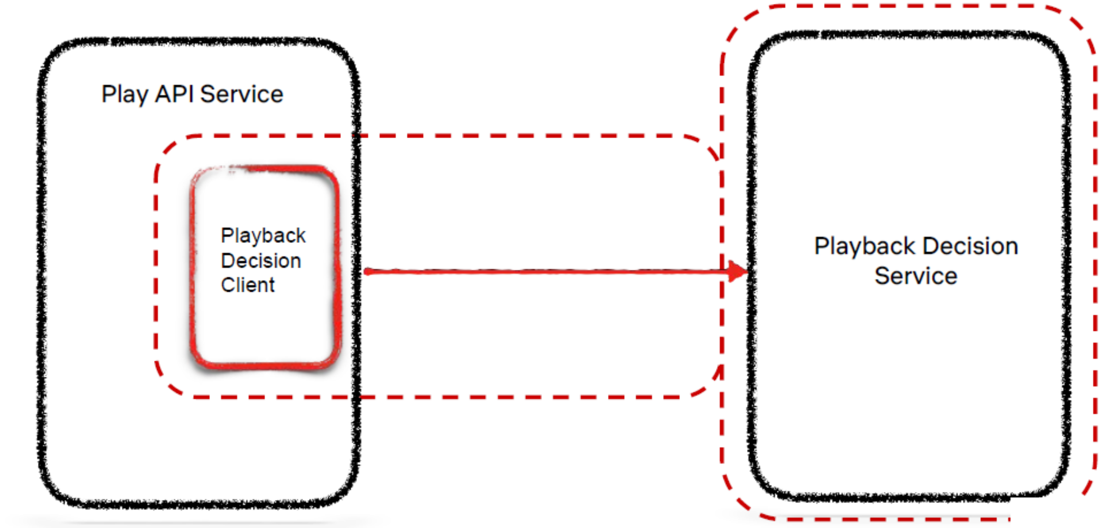
- **Sam Newman (Building Microservices)**

Binary coupling => Distributed Monolith


Utilities
Service1
Utilities
Service2
Hundreds of shared libraries **spanning** services across **network boundaries**
Utilities
Service3

Previous Architecture

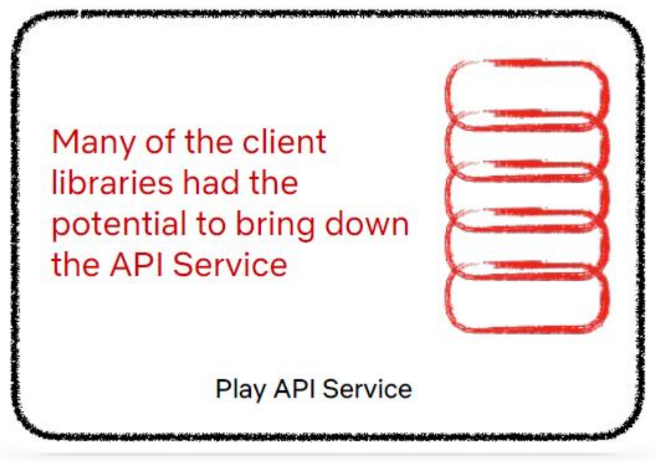
2) Operational Coupling



Previous Ar

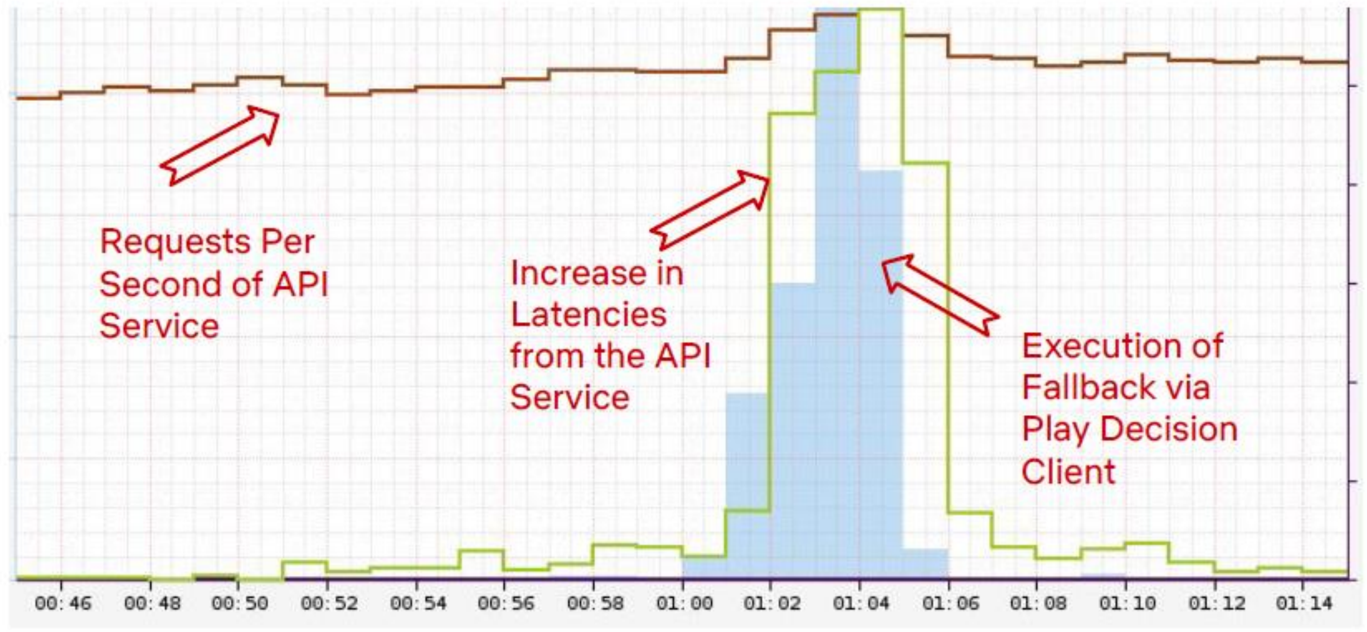
"Operational Coupling" might be an ok choice, if some services/teams are not yet ready to own and operate a highly available service.

Operational Coupling impacts Availability


Many of the client libraries had the potential to bring down the API Service
Play API Service

Previous Architecture

Clients with heavy Fallbacks



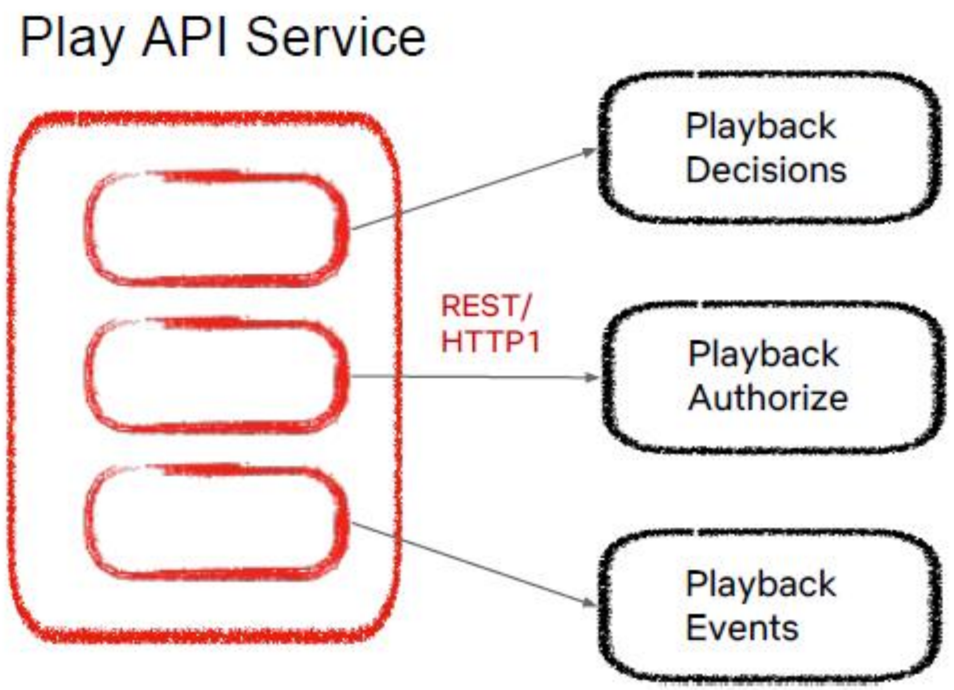
Requirements

Operationally “thin” Clients	No or limited shared libraries
Auto-generated clients for Polyglot support	Bi-Directional Communication

REST vs RPC

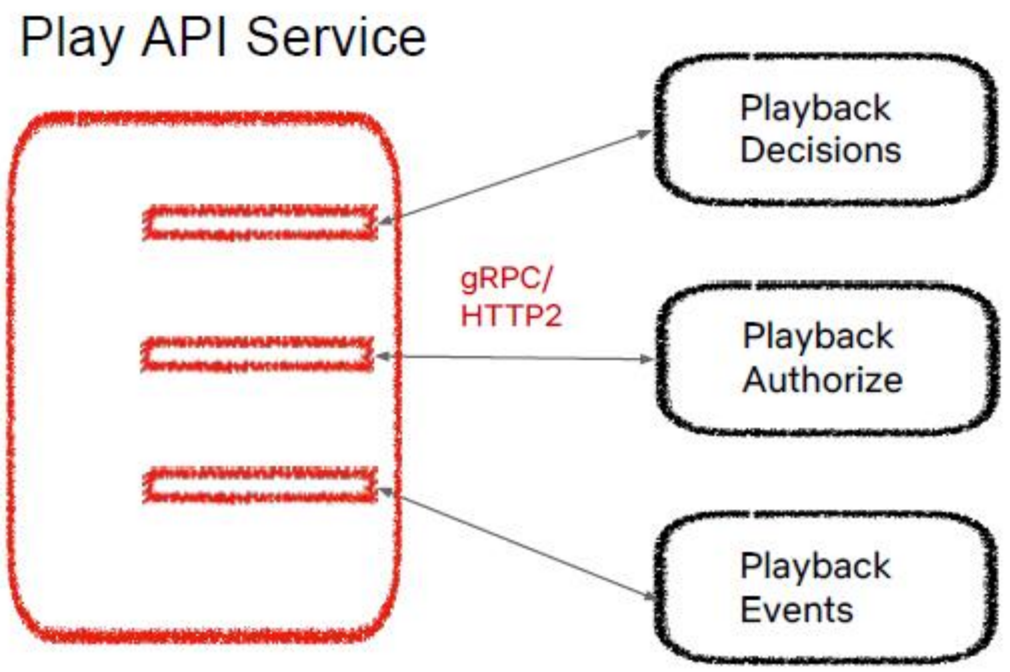
- At Netflix, most use-cases were modelled as Request/Response
  - REST was a simple and easy way of communicating between services; so choice of REST was more incidental rather than intentional
- Most of the services were not following RESTful principles.
  - The URL didn’t represent a unique resource, instead the parameters passed in the call determined the response - effectively made them a RPC call
- So we were agnostic to REST vs RPC as long as it meets our requirements

Previous Architecture



1) Operationally Coupled Clients
2) High Binary Coupling
3) Only Java
4) Unidirectional communication

Current Architecture



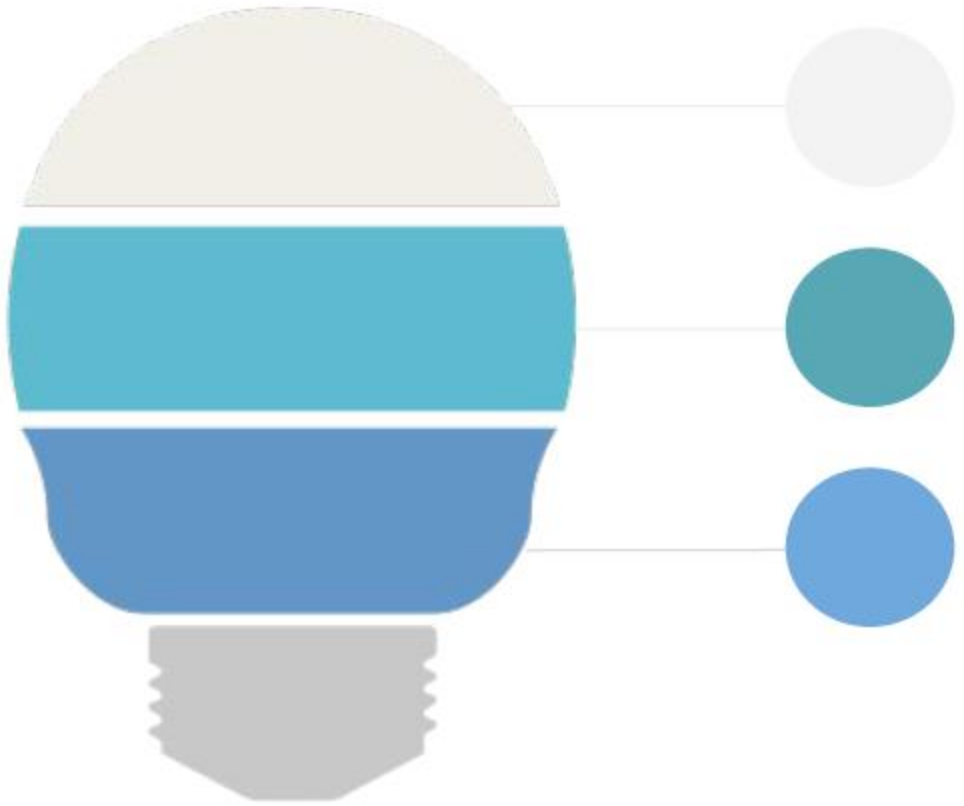
1) Minimal Operational Coupling
2) Limited Binary Coupling
3) Beyond Java
4) Beyond Request/ Response

Type 1 Decision: Appropriate Coupling

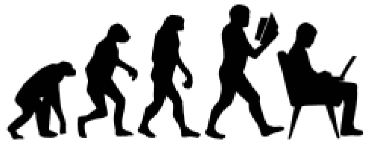
Consider “thin” auto-generated clients with bi-directional communication and minimize code reuse across service boundaries

For Type 2 decisions, choose a path, experiment and iterate

Guiding Principle: Identify your Type 1 and Type 2 decisions; Spend 80% of your time debating and aligning on Type 1 Decisions



NETFLIX



An Evolutionary Architecture supports guided and incremental change as first principle among multiple dimensions

- **ThoughtWorks**

Choosing a microservices architecture with appropriate coupling allows us to evolve across multiple dimensions

How evolvable are the Type 1 decisions

Known Unknowns

Change Play API	**Previous Architecture**	**Current Architecture**
Asynchronous?	▬▬▬▬	▬▬▬▬▬▬▬▬
Polyglot services?	▬	▬▬▬▬▬▬
Bidirectional APIs?	▬	▬▬▬▬▬▬▬▬
Additional Data Sources?	▬▬▬▬▬	▬▬▬▬▬▬▬

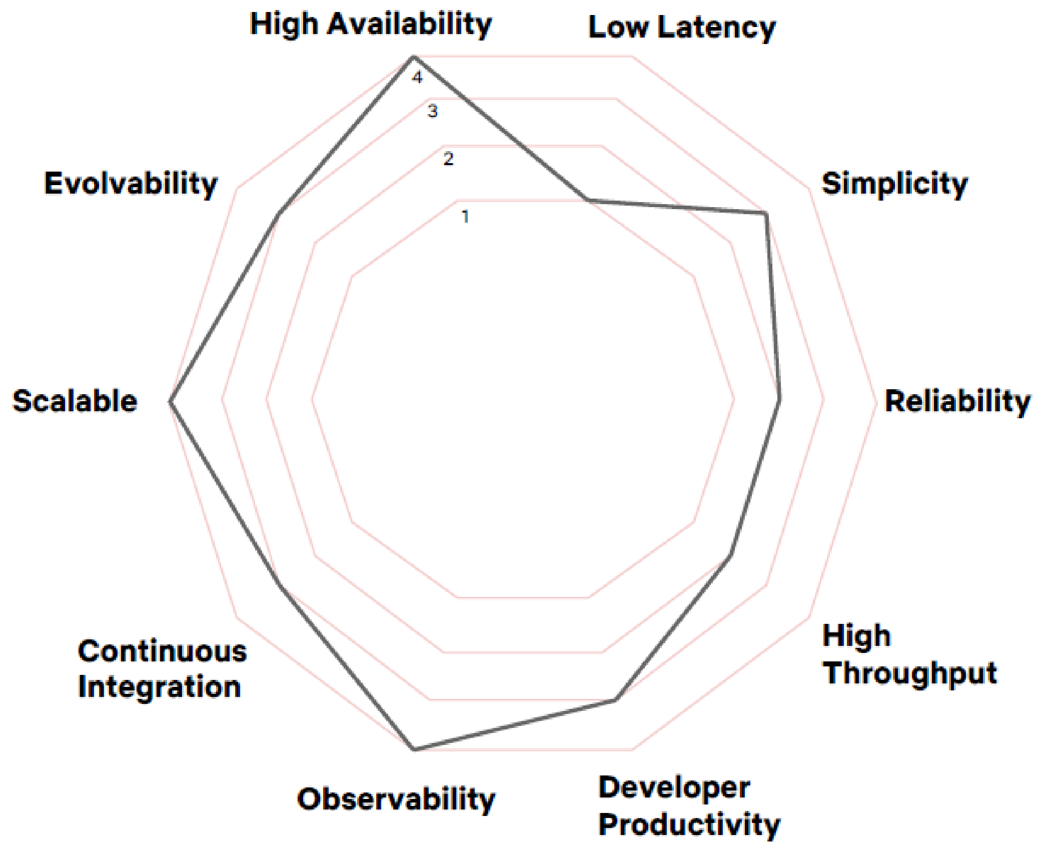
Potential Type 1 decisions in the future?

Change Play API	**Previous Architecture**	**Current Architecture**
Containers?	▬	?
Serverless?	▬	?

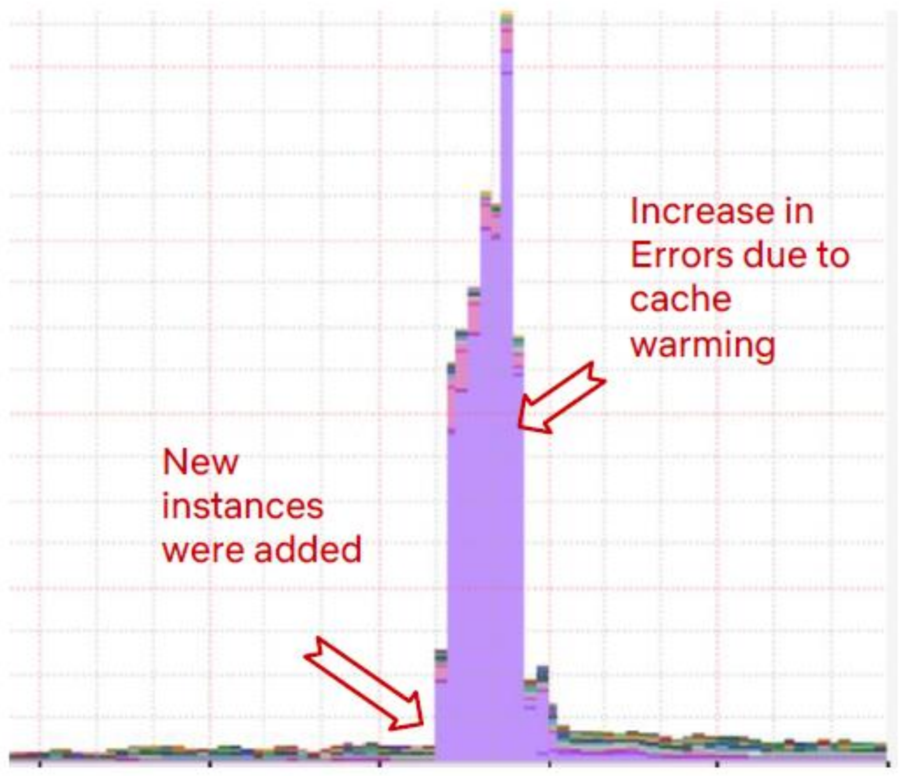
And we fully expect that there will be Unknown Unknowns

As we evolve, how to ensure we are not breaking our original goals?

Use Fitness Functions to guide change



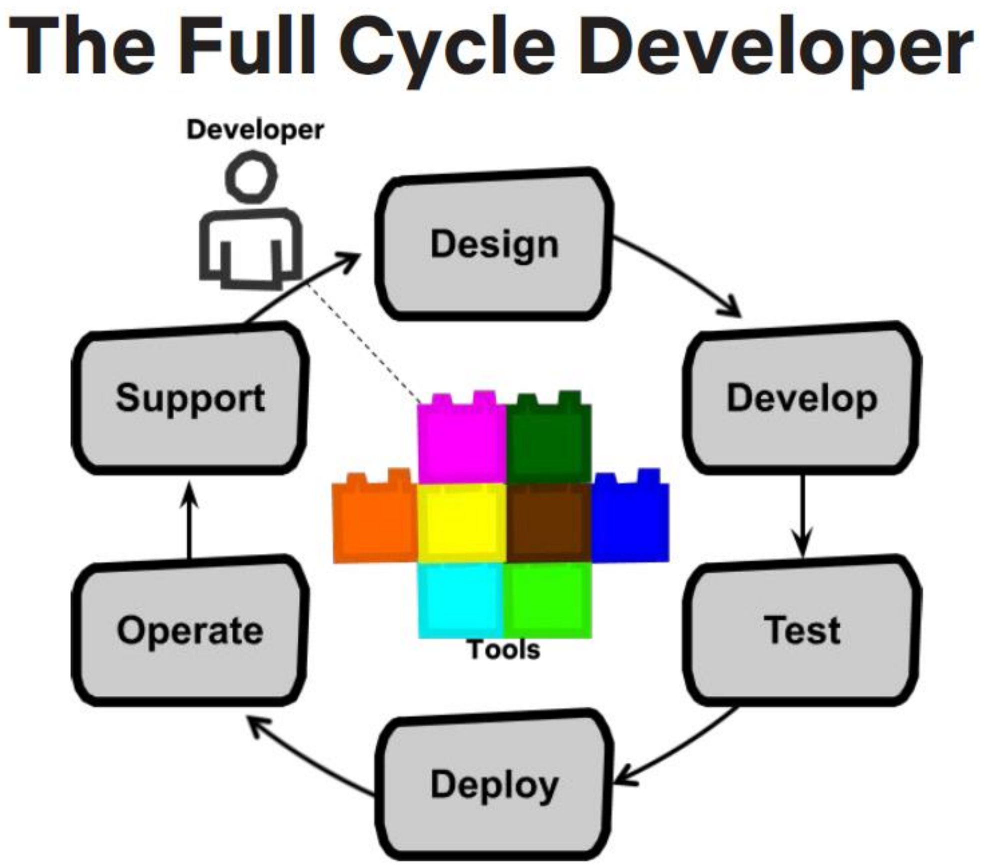
Why Scalability over Throughput?



Guiding Principle: Define Fitness functions to act as your guide for architectural evolution

Previous Architecture	*Current* Architecture
Multiple Identities	Singular Identities
Operational Coupling	Operational Isolation
Binary Coupling	No Binary Coupling
Synchronous communication	Asynchronous communication
Only Java	Beyond Java
Data Monolith	Explicit Data Architecture
	Guided Fitness Functions

Full Cycle Developers @ Netflix by Greg Burrell



If You Don’t Know Where You’re Going, It Doesn’t Matter How Fast You Get There
by Jez Humble, Nicole Forsgren

2018

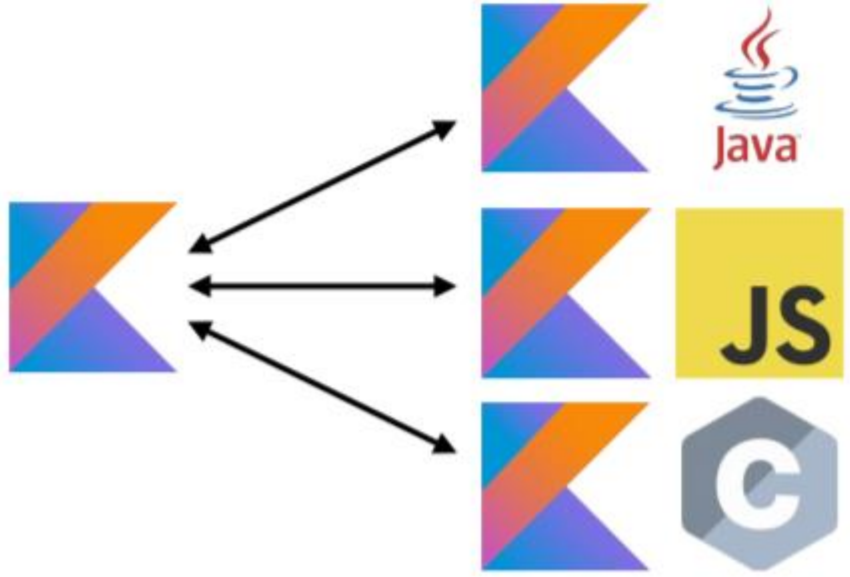
Accelerate:
State of DevOps
Strategies for a New Economy

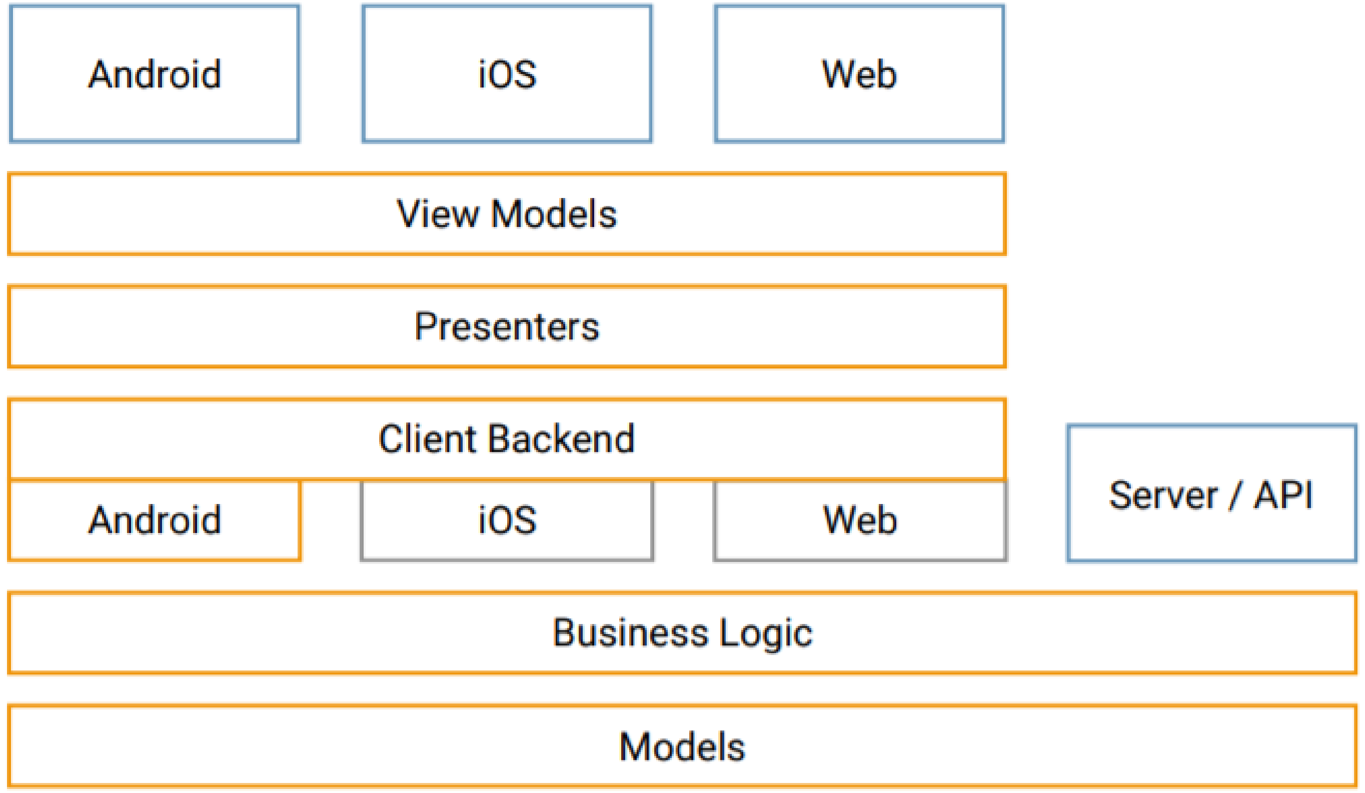
Presented by
DORA
DEVOPS RESEARCH & ASSESSMENT

Diamond Sponsor
Google Cloud

Gold Sponsors
Deloitte. XebiaLabs pagerduty redgate REV
cloudbees Datical sumologic TRICENTIS GitLab
Electric Cloud Microsoft Azure Pivotal aws ca technologies

Kotlin: Write Once, Run (Actually) Everywhere
by Jake Wharton

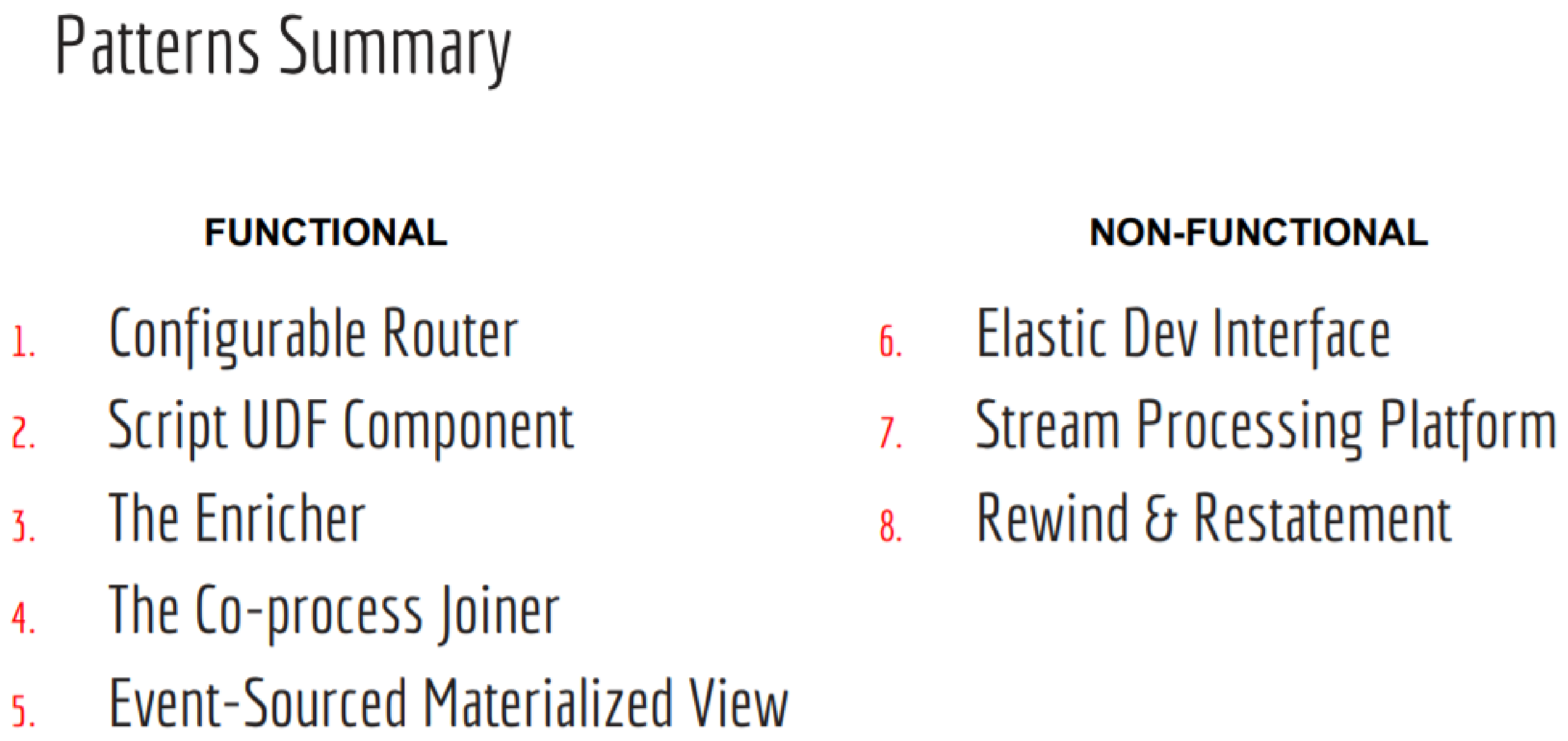




Building Production-Ready Applications by Michael Kehoe

1	Stability & Reliability
2	Scalability & Performance
3	Fault Tolerance and DR
4	Monitoring
5	Documentation

Patterns of Streaming Applications by Monal Daxini



Patterns Summary

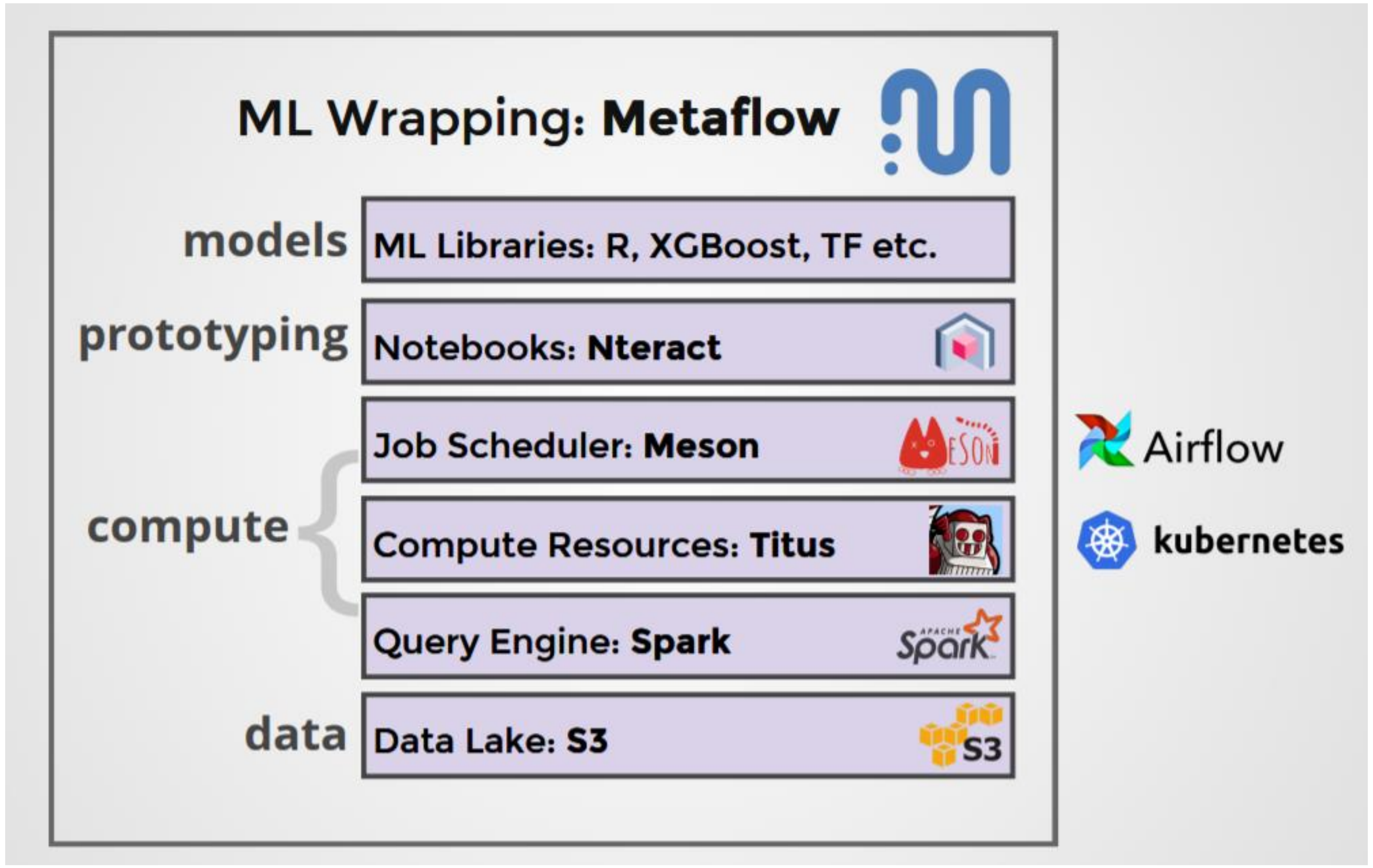
FUNCTIONAL

1. Configurable Router
2. Script UDF Component
3. The Enricher
4. The Co-process Joiner
5. Event-Sourced Materialized View

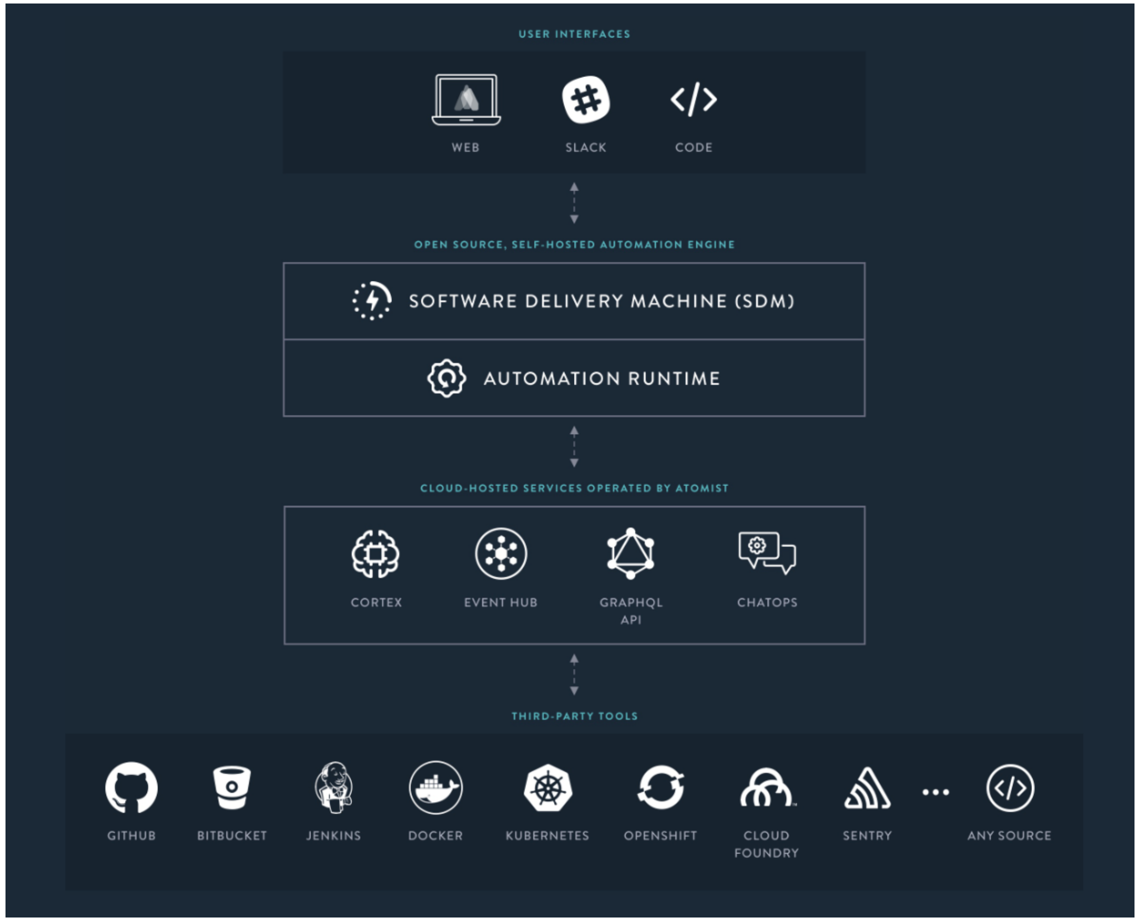
NON-FUNCTIONAL

6. Elastic Dev Interface
7. Stream Processing Platform
8. Rewind & Restatement

Human Centric Machine Learning Infrastructure @Netflix by Ville Tuulos



Atomist - A Platform Built For Delivering Modern Cloud Native Application



Q&A

- What is gRPC?
  - "gRPC is a modern, open source remote procedure call (RPC) framework that can run anywhere. It enables client and server applications to communicate transparently, and makes it easier to build connected systems."
  - "The main usage scenarios:
    - Low latency, highly scalable, distributed systems.
    - Developing mobile clients which are communicating to a cloud server.
    - Designing a new protocol that needs to be accurate, efficient and language independent.
    - Layered design to enable extension eg. authentication, load balancing, logging and monitoring etc."

- What language is used in developing Atomist's Software Delivery Machine (SDM)?
  - "…SDM is in TypeScript (or JavaScript works too), and comes with a framework designed for software delivery and development automation. Write functions to make decisions or take action, with access to all the code plus the context of the push or build or issue event. All of this is open source."

ResMed Open Role!!!

Check them out at: careers.resmed.com

- ✓ Sr. V&V Engineer
- ✓ Agile Project Manager
- ✓ Senior Manager, Technical Product Manager
- ✓ Systems Analyst
- ✓ Manager, Platform Engineer
- ✓ Software Engineer
- ✓ Associate Software Engineer
- ✓ Senior Software Development Engineer in Test
- ✓ Associate Software Development Engineer in Test
- ✓ Project Manager, Advanced Analytics

And many more…

ResMed

Thank You!