



Serverless architecture: Functions as a Service

#LvivJS2017



Dzmitry Varabei
EPAM, Chief Software
Engineer



#rollingscopes
#rsschool



#rsconf



Часть #1. Предисловие

A woman with short brown hair, seen from the back, wears a dark fur-trimmed coat and looks towards a man in a hooded coat. They are standing in front of large, dark wooden doors with metal rivets. The scene has a dramatic, cinematic feel.

– Кем вы видите себя в нашей
компании через 5 лет?
– Никем.

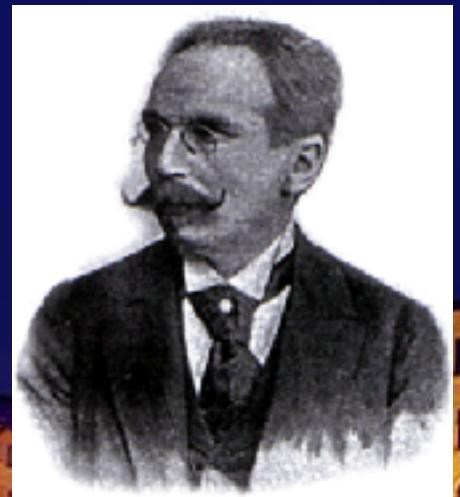


CEO?

Manager?

Team Lead?

Senior?



Кто-нибудь хочет быть Архитектором?

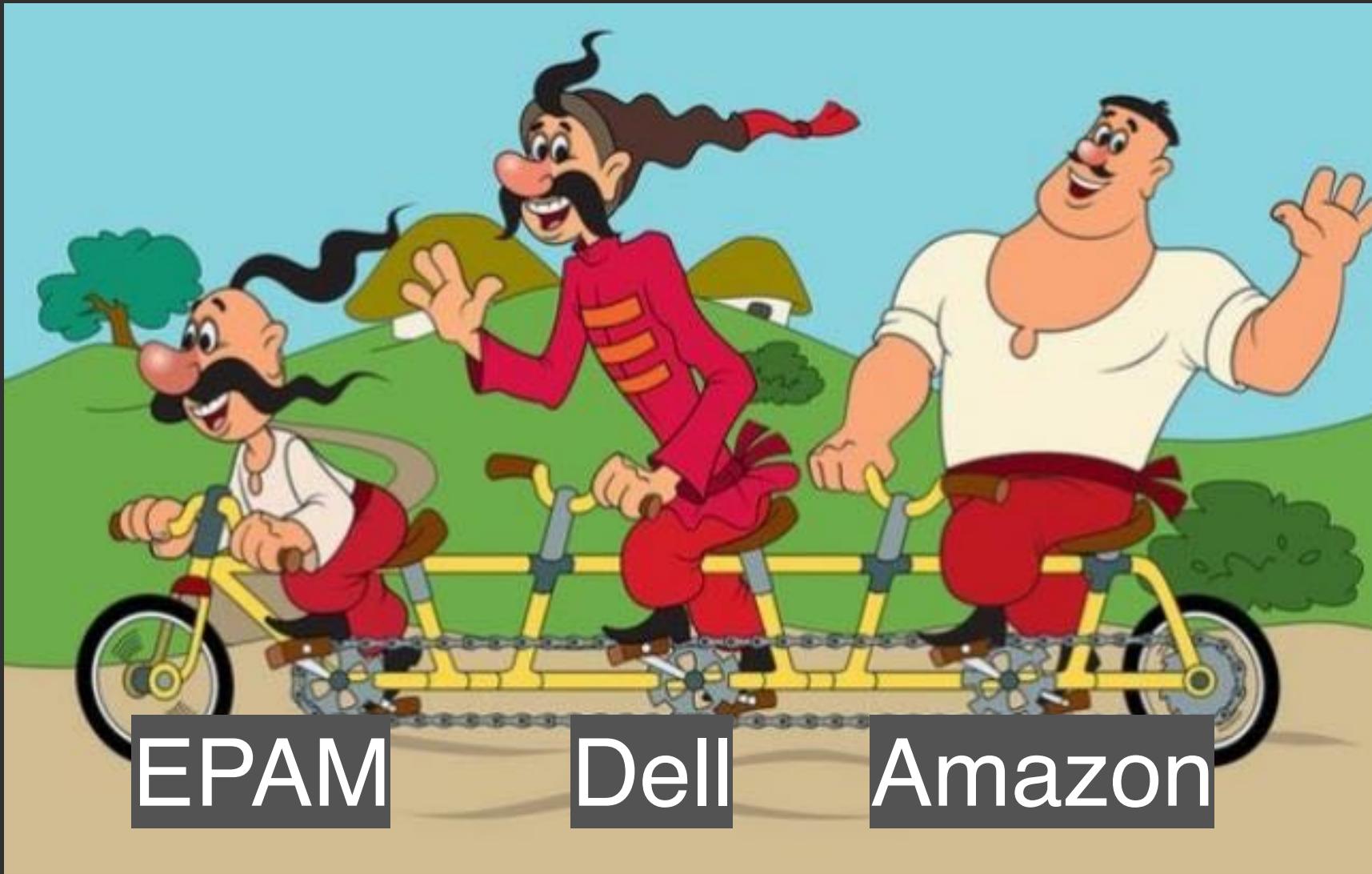


Точнее Software Architect

Часть #2.

Исследование на тему
“Кто такой Software/Solution Architect”

Поговорил с архитекторами

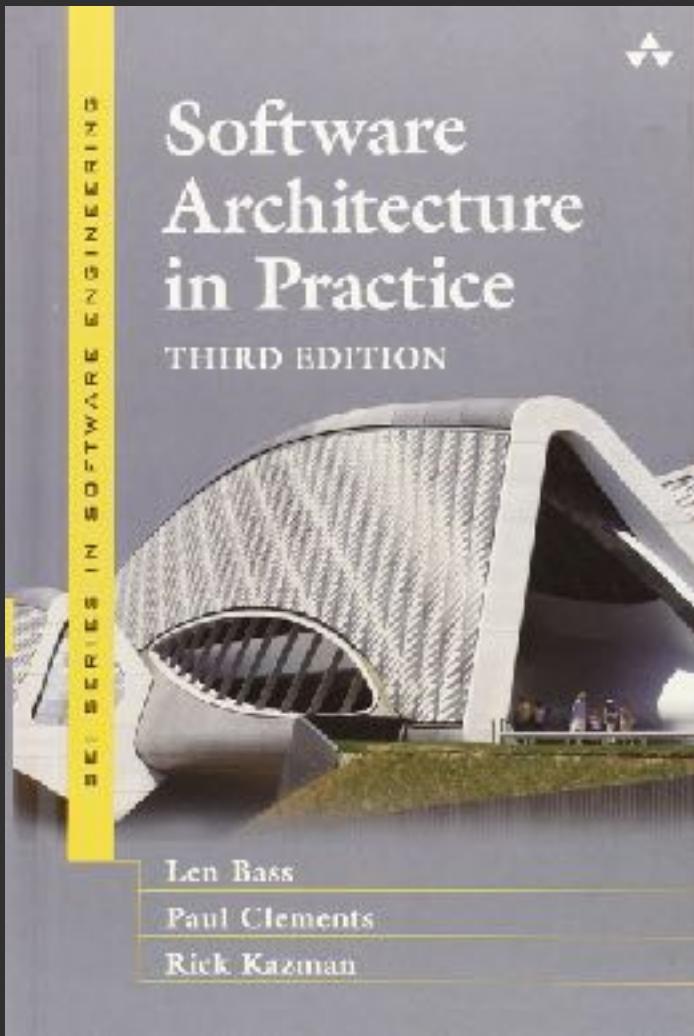


EPAM

Dell

Amazon

Почитал книгу



Сходил на курсы



Аналогия №1

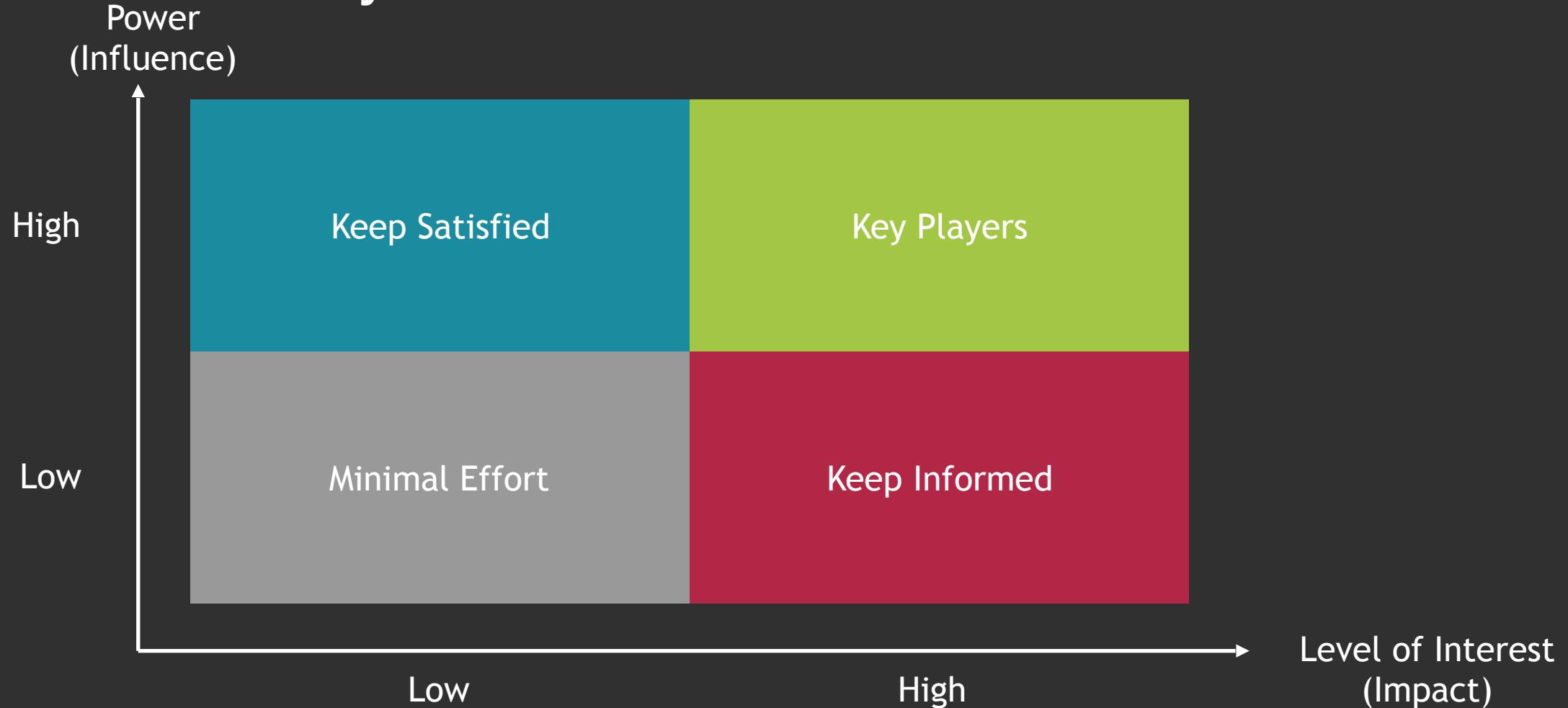


Аналогия №2

TDD & Test First

- Много командировок (до 25% рабочего времени)
- Много общения с заказчиком
- Много общения с заказчиками заказчиков
- Много общения с sales отделом
- Много общения с разработчиками
- Много эстимирования, прогнозирования и т.д.
- Много подготовки различной документации
- Много рисования диаграмм
- ГДЕ КОД???????

Классификация и приоритизация заказчиков или кого слушать и что же всё-таки делать



- Попытаться понять реальные бизнес цели
- Понять требования (иногда противоречивые)
- Определить основные не функциональные требования (ASR)

Байка №1

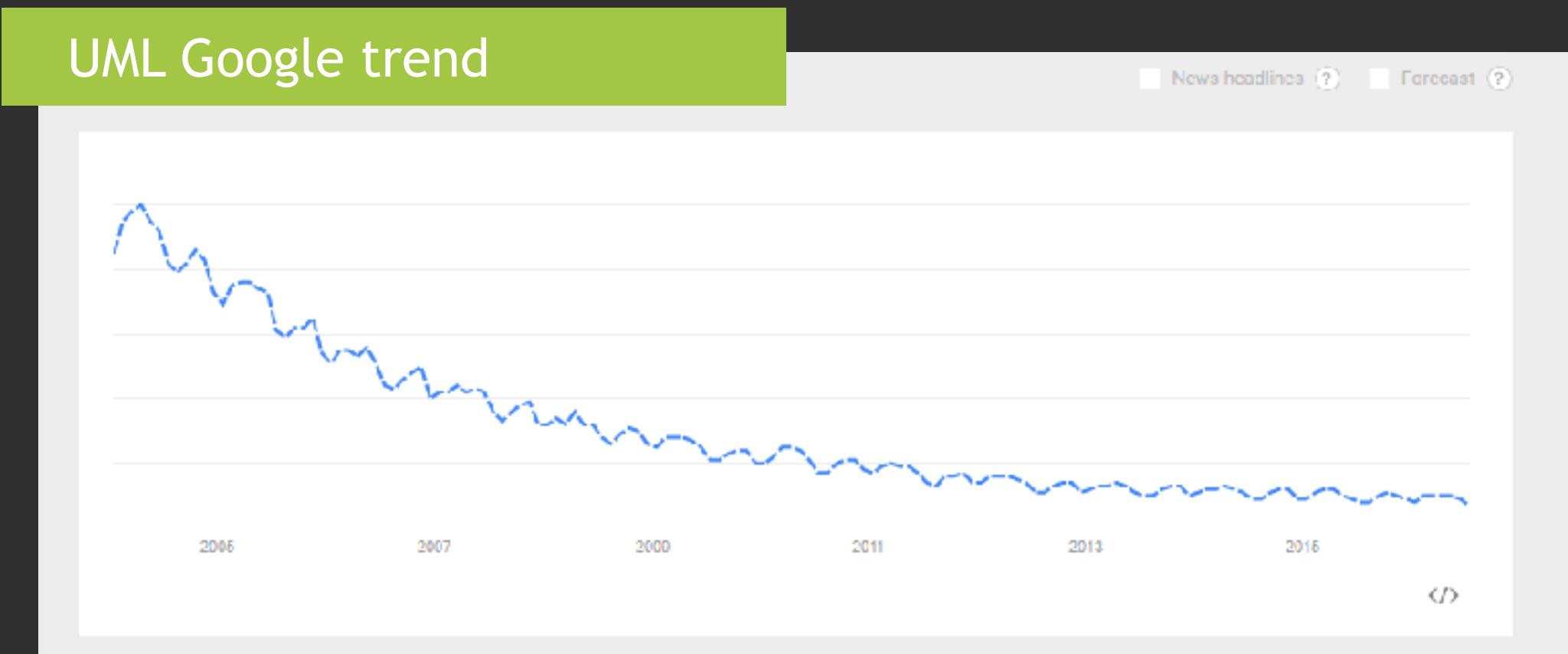
0.6 sec

- Work breakdown structure
- Estimations
- Resource plan
- Release schedule

Вас когда-нибудь просили подготовить
документацию?

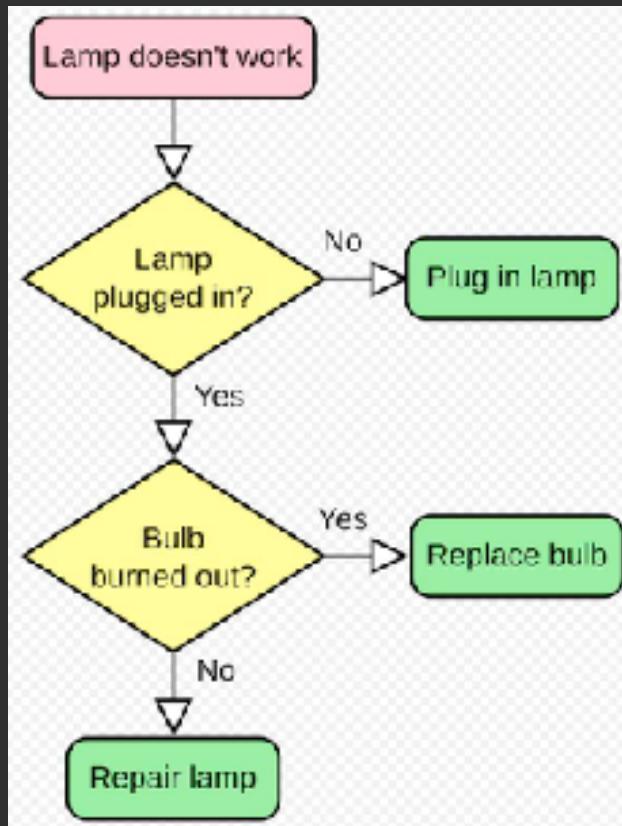
Нарисовать пару диаграмм?

Хорошие новости



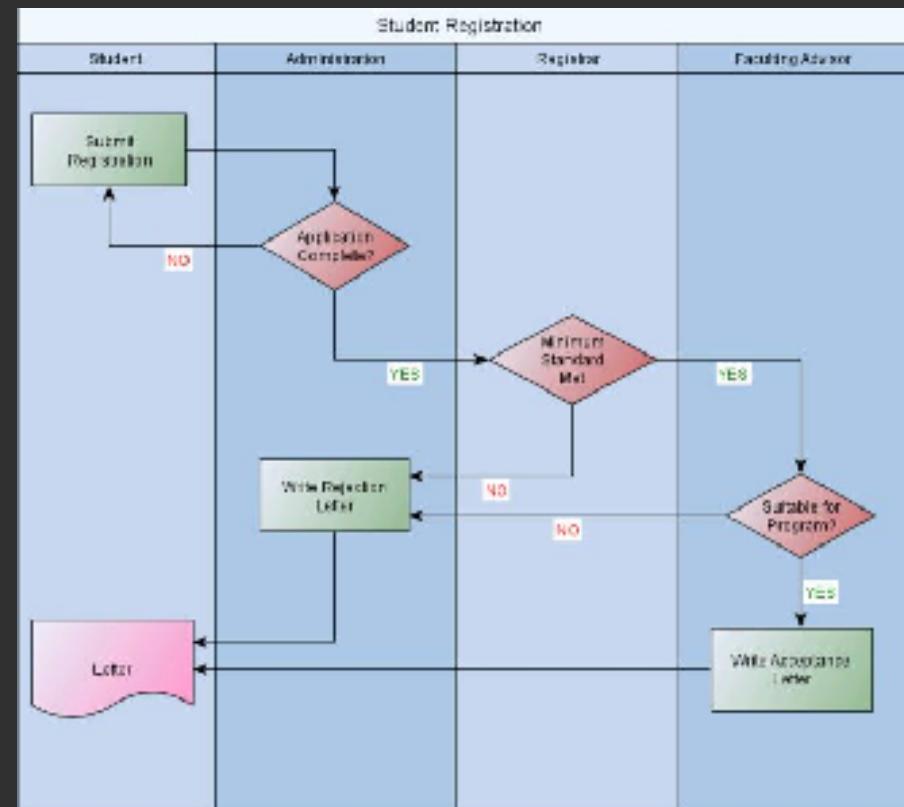
В природе 150+ классифицированных
видов диаграмм

Flowchart



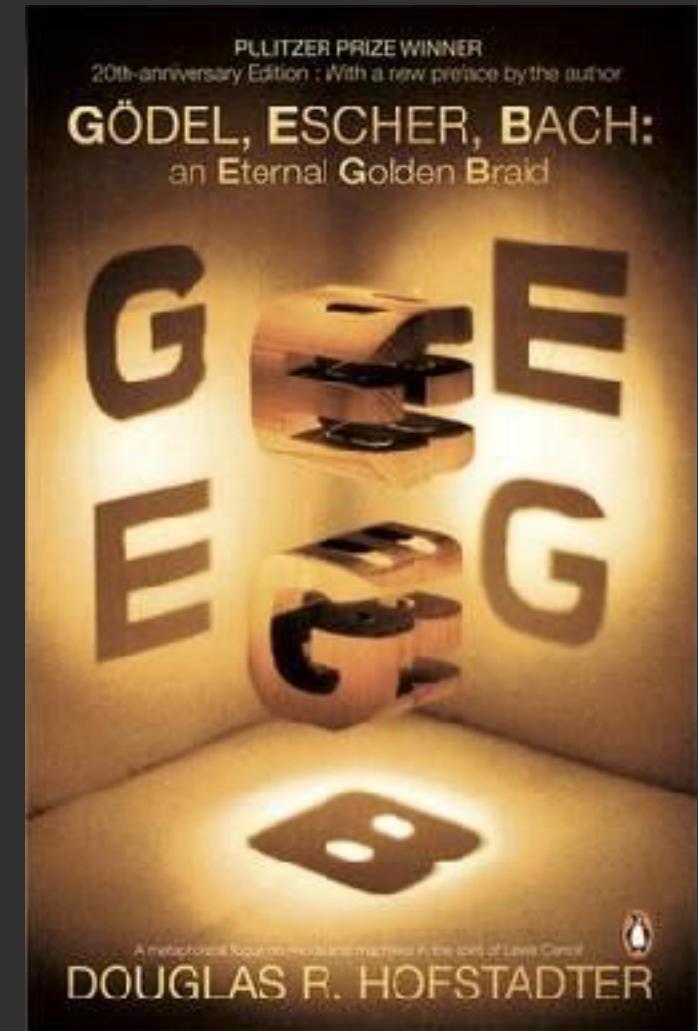
Например

Swimlane



В это время на реальном проекте:

“линии и блоки
в powerpoint” без легенды
и текстового описания



Architecture views

- 4 + 1 View Model
- Siemens architecture view
- Zachman framework
- TOGAF
- C4 (From developers to developers)

C4 MODEL

C4 Model

SYSTEM CONTEXT DIAGRAM

Starting point for diagramming and documenting, allows to look at the big picture. Draw a simple block diagram showing your system as a box in the centre, surrounded by its users and the other systems that it interfaces with.

CONTAINER DIAGRAM

Next step can be to illustrate the high-level technology choices with a containers diagram. By "container" I mean something like a web application, desktop application, mobile app, database, file system, etc.

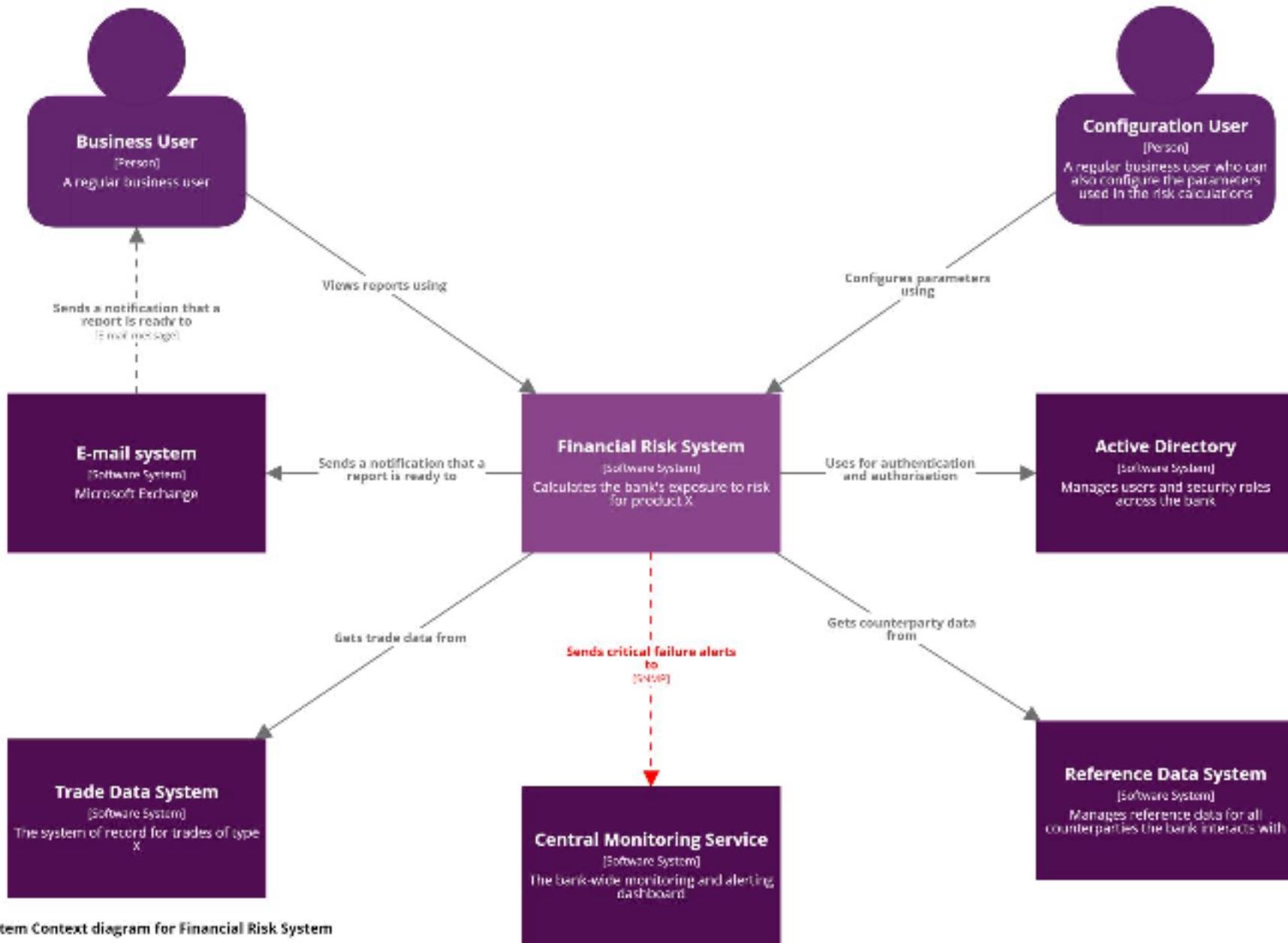
COMPONENT DIAGRAM(S)

Zoom in and decompose each container further. However you decompose your system is up to you, but I tend to identify the major logical components and their interactions.

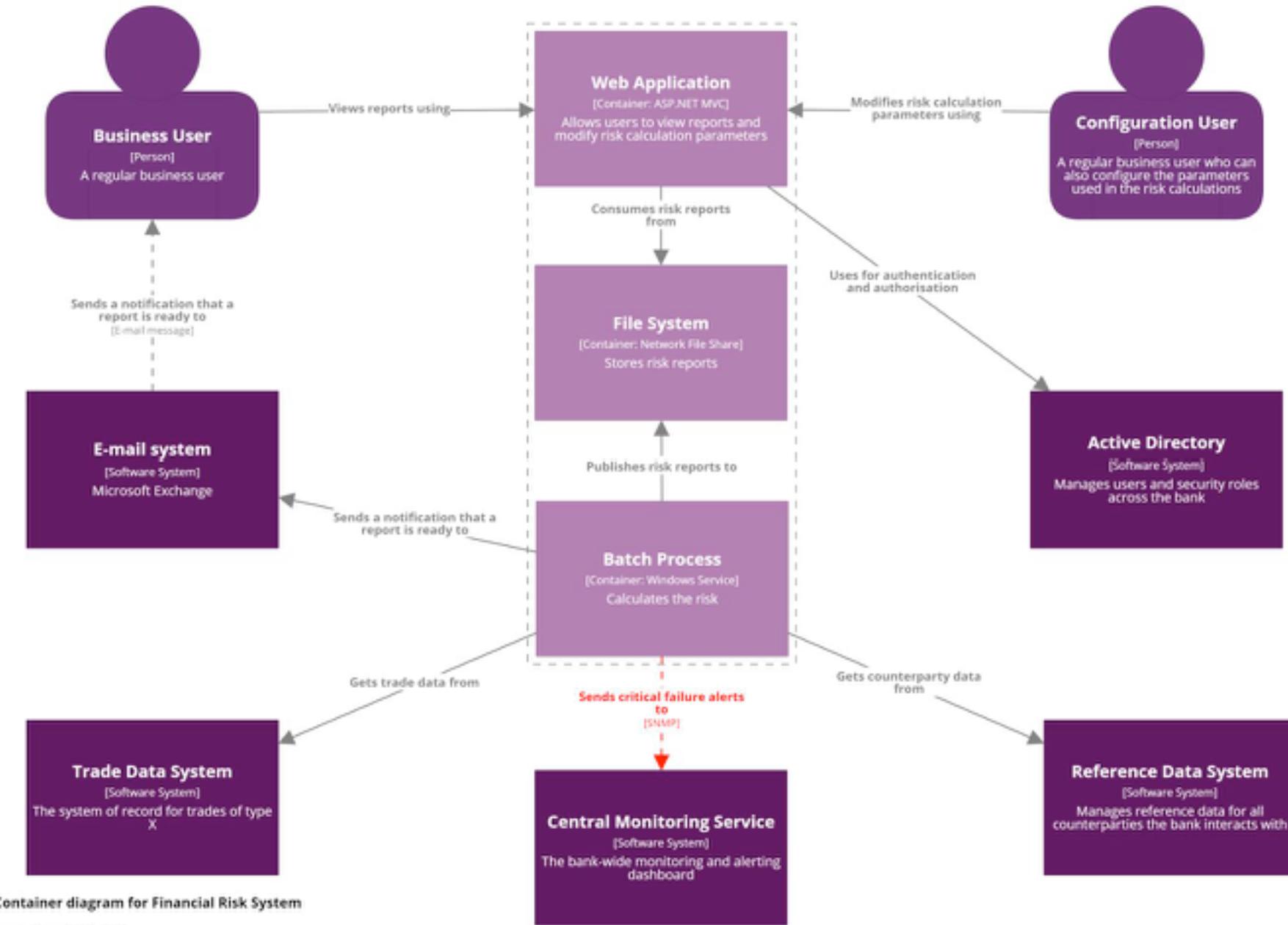
CLASS DIAGRAM(S)

This is optional, but I sometimes draw one or more UML class diagrams if I want to illustrate specific component

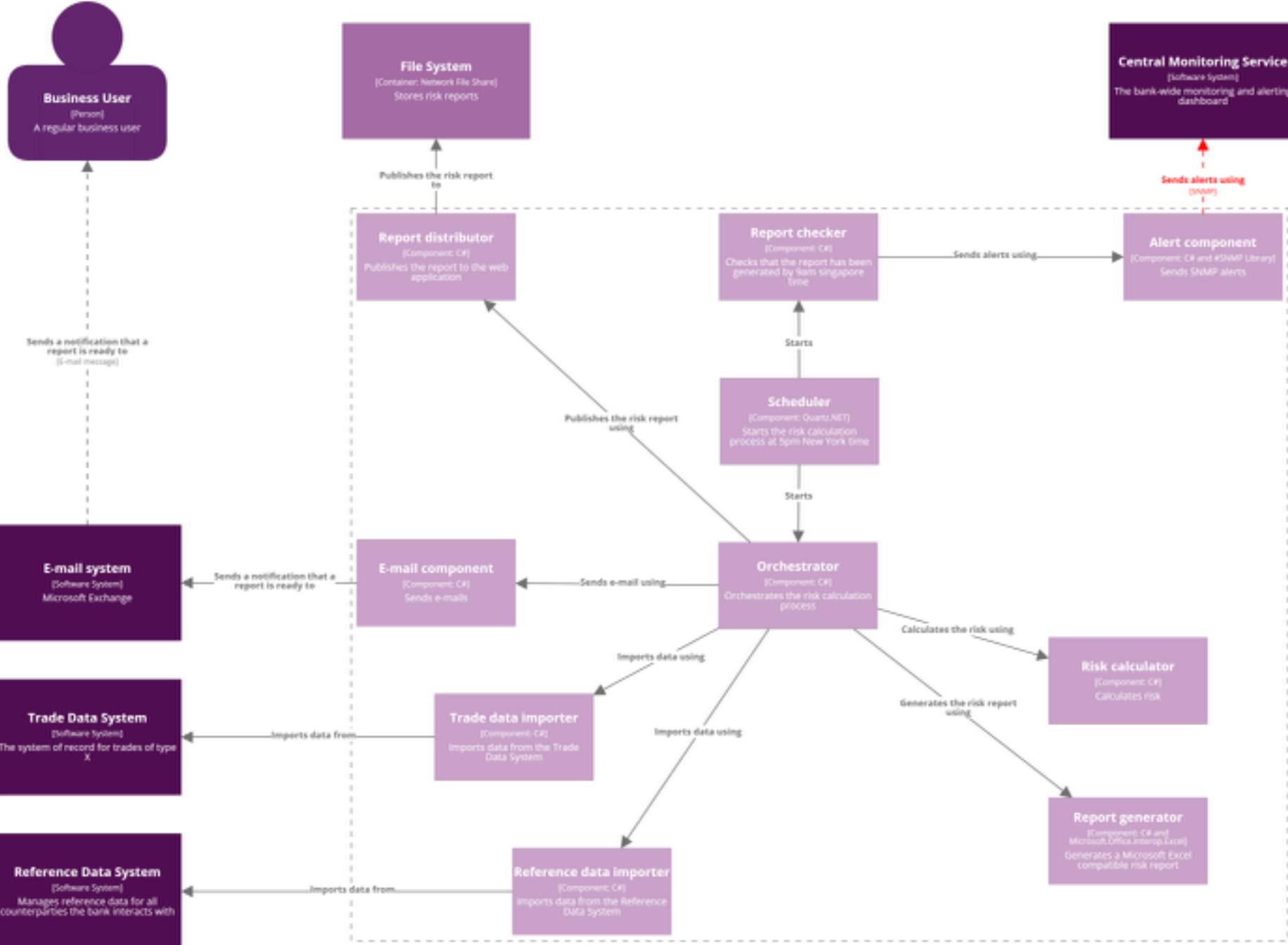
SYSTEM CONTEXT DIAGRAM



CONTAINER DIAGRAM

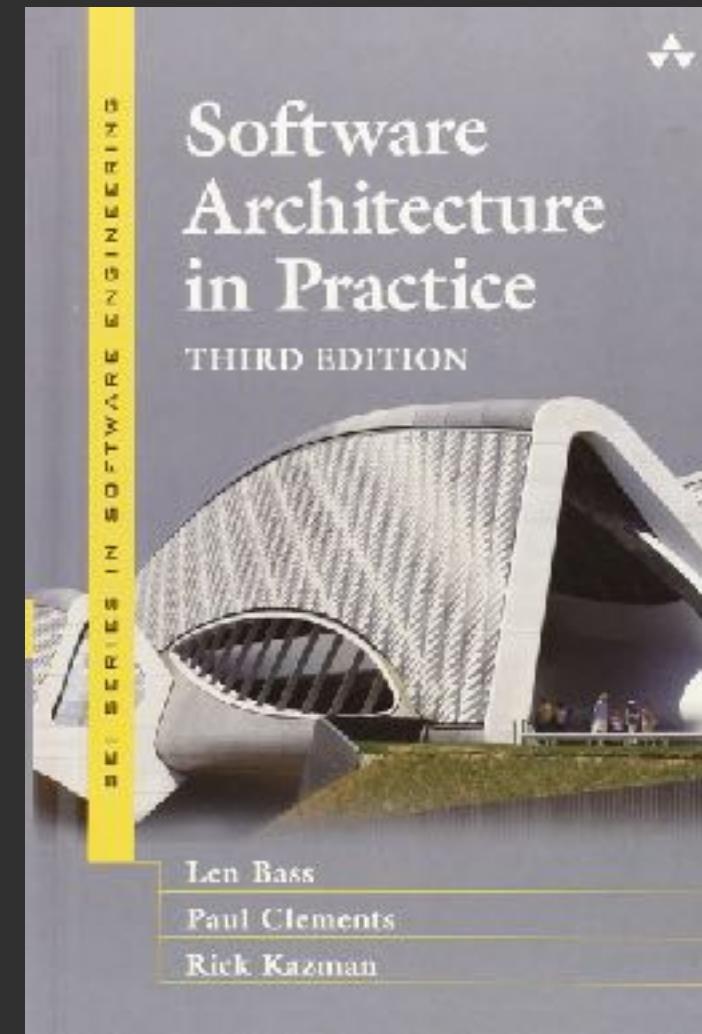


COMPONENT DIAGRAM



Основные атрибуты качества

Availability
Interoperability
Modifiability
Performance
Security
Testability
Usability



Availability

Availability refers to a property of software that it is there and ready to carry out its task when you need it to be. It is linked with:

- Security : if hacked it won't be available
- Performance : available but response time is 2 days???
- Fault-tolerance : can a system recover from failures

A common way to express availability is in terms of a number of nines:

Number of nines	Percentage	Minutes of downtime (year)
Three 9s	99.9%	525.6
Four 9s	99.99%	52.56
Five 9s	99.999%	5.256
Six 9s	99.9999%	0.5256

WEB

TECH

AMAZON

How a typo took down S3, the backbone of the internet

40

*Hello, operator*by [Casey Newton](#) | [@CaseyNewton](#) | Mar 2, 2017, 1:24pm EST

SHARE

TWEET

LINKEDIN

In 2013, Gmail was available
99.978 percent of the time
(~2 hours per year)

И еще много интересных паттернов

Pattern: Leaky Bucket Counter

Level: Detection

Context/Risk

Your system is intended to recognize and correct errors automatically. Not all errors are so critical that the first occurrence of them stimulates error processing.

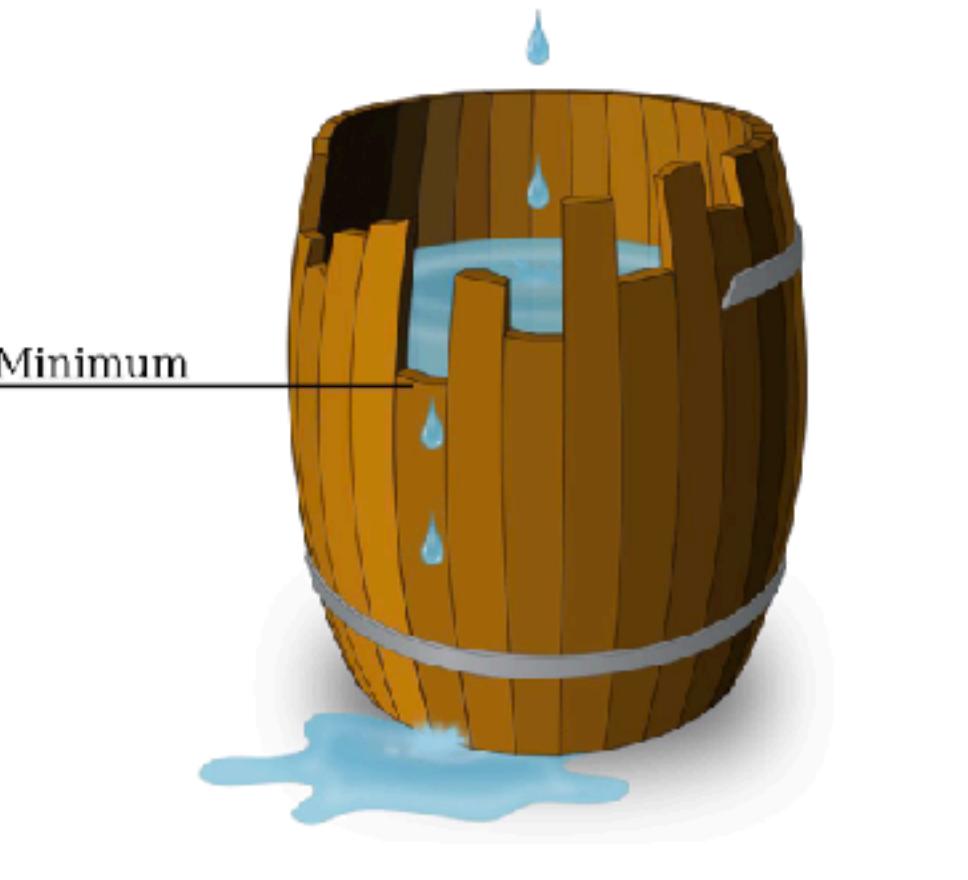
Intention

You need to know if an error is transient or intermittent.

Solution:

Define your leaky bucket counter

- Initial value at the system start-up
- Incrementing when specified issue occurs
- Decrementing over specified period but never less than initial value
- If exceeding a predetermined upper threshold, handle the error as permanent



И еще много интересных паттернов



HYSTRIX
DEFEND YOUR APP

Hystrix: Latency and Fault Tolerance for Distributed Systems

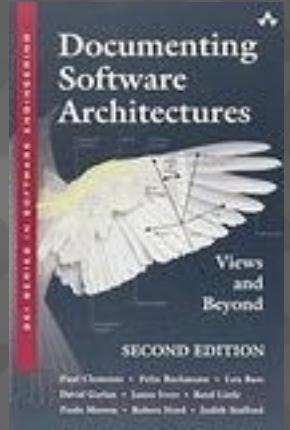
OSS Lifecycle inaccessible build failing maven central 1.5.12 License Apache 2

Hystrix is a latency and fault tolerance library designed to isolate points of access to remote systems, services and 3rd party libraries, stop cascading failure and enable resilience in complex distributed systems where failure is inevitable.

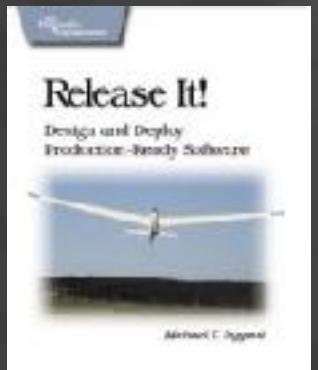
BOOKS



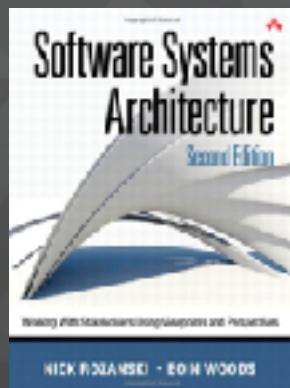
Software Architecture in Practice
(3rd Edition)



Documenting Software Architectures:
Views and Beyond (2nd Edition)

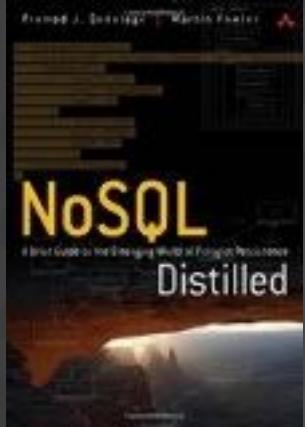


Release It! Design and Deploy
Production-Ready Software

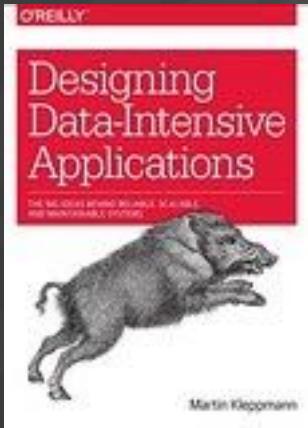


Software Systems Architecture:
Working With Stakeholders Using
Viewpoints and Perspectives (2nd
Edition)

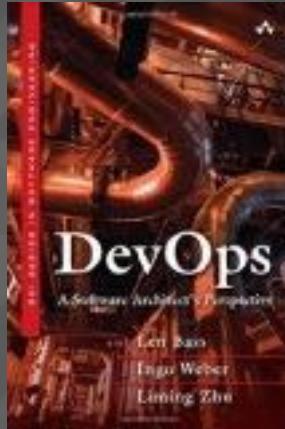
BOOKS



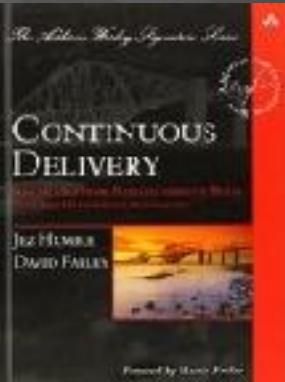
NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence



Designing Data-Intensive Applications



DevOps: A Software Architect's Perspective (SEI Series in Software Engineering)



Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation

Что осталось за кадром?

Architecture styles

SOLUTION LEVEL

- MONOLITH
- SERVICE-ORIENTED
- MICROSERVICES
- EVENT-DRIVEN

COMPONENT LEVEL

- LAYERED
- MODULE-BASED
- REST

Techical Domains

SQL

Web

Cloud

Mobile

Data Streaming

BigData

Cache

NoSQL

Search

Message oriented middleware

Machine Learning

и на десерт...

Infrastructure Provisioning & Deployment

Вывод:

Читайте книги, ходите на митапы и т.д.
Не забывайте про закон Парето.

`fs.writeFileSync()`

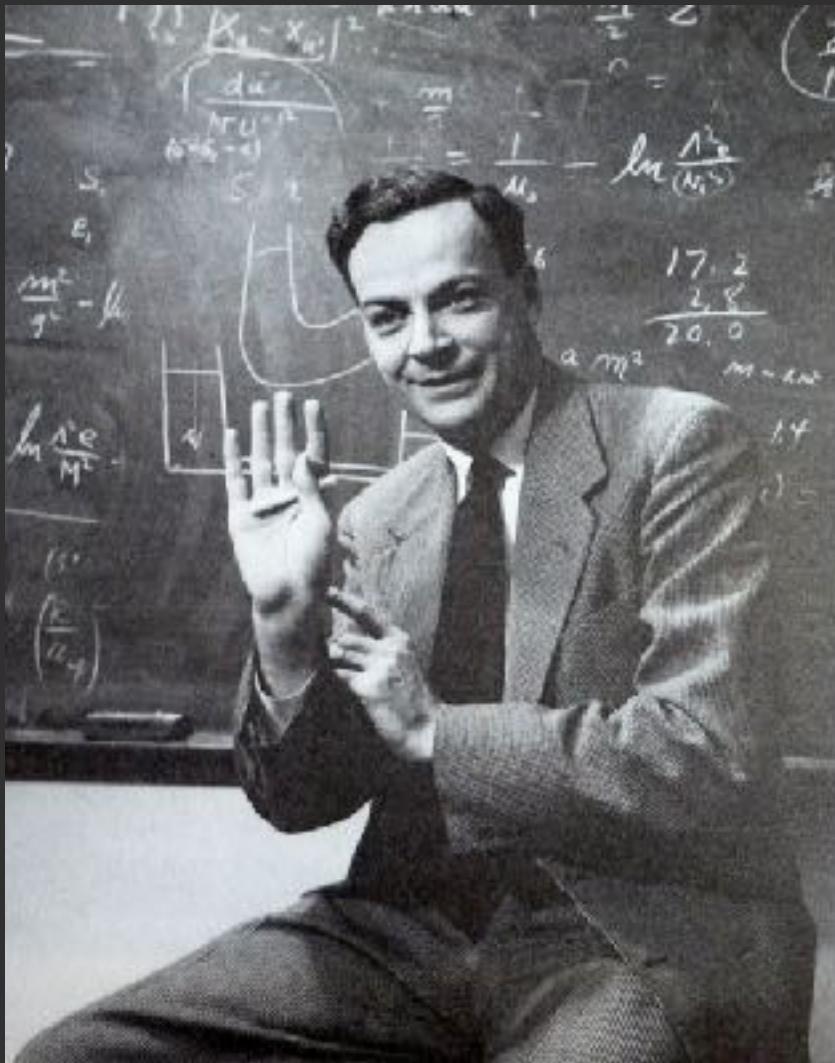
`fs.fsync()`

`fs.fsyncSync()`

Скучно?

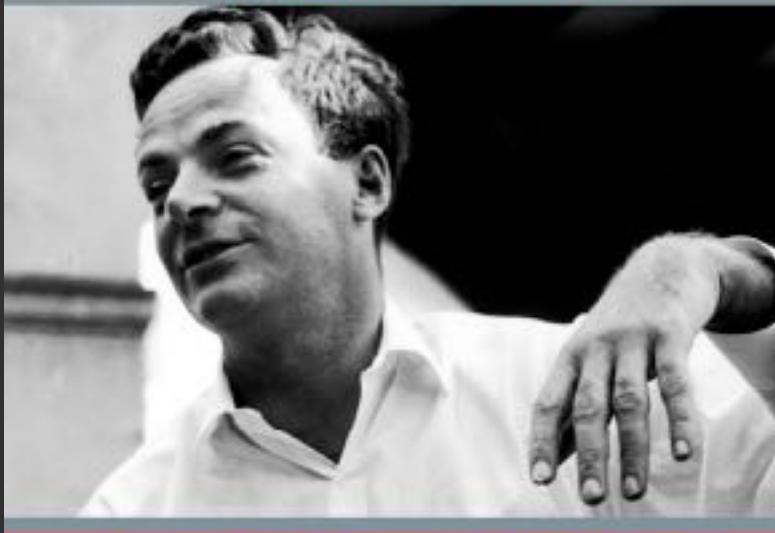
Скучно





НОБЕЛЕВСКИЙ ЛАУРЕАТ О ВРЕМЕНИ,
НАУКЕ И О СЕБЕ

ВЫ, КОНЕЧНО, ШУТИТЕ, МИСТЕР ФЕЙНМАН!



Это произведение – лучшее доказательство того, что гении могут заставить вас
одновременно смеяться и задумываться из весьма серьезные темы.

«New York Times»

РИЧАРД ФЕЙНМАН

озон.ru



Serverless architecture: Functions as a Service

#LvivJS2017

ThoughtWorks®

TECHNOLOGY RADAR APRIL '16

Our thoughts on the
technology and trends that
are shaping the future

Chapter I: The hardest problem

The hardest problem in computer science

- ...is, of course, **naming**.

Serverless architecture!

Just right for front-end devs?

02 WINS

60

Angular.js developer

CAGE

Front-end WINS
FATALITY ???

DevOps ?
Java dev?



Fashionable Naming

Cloud Computing

Infrastructure as a service (IaaS)

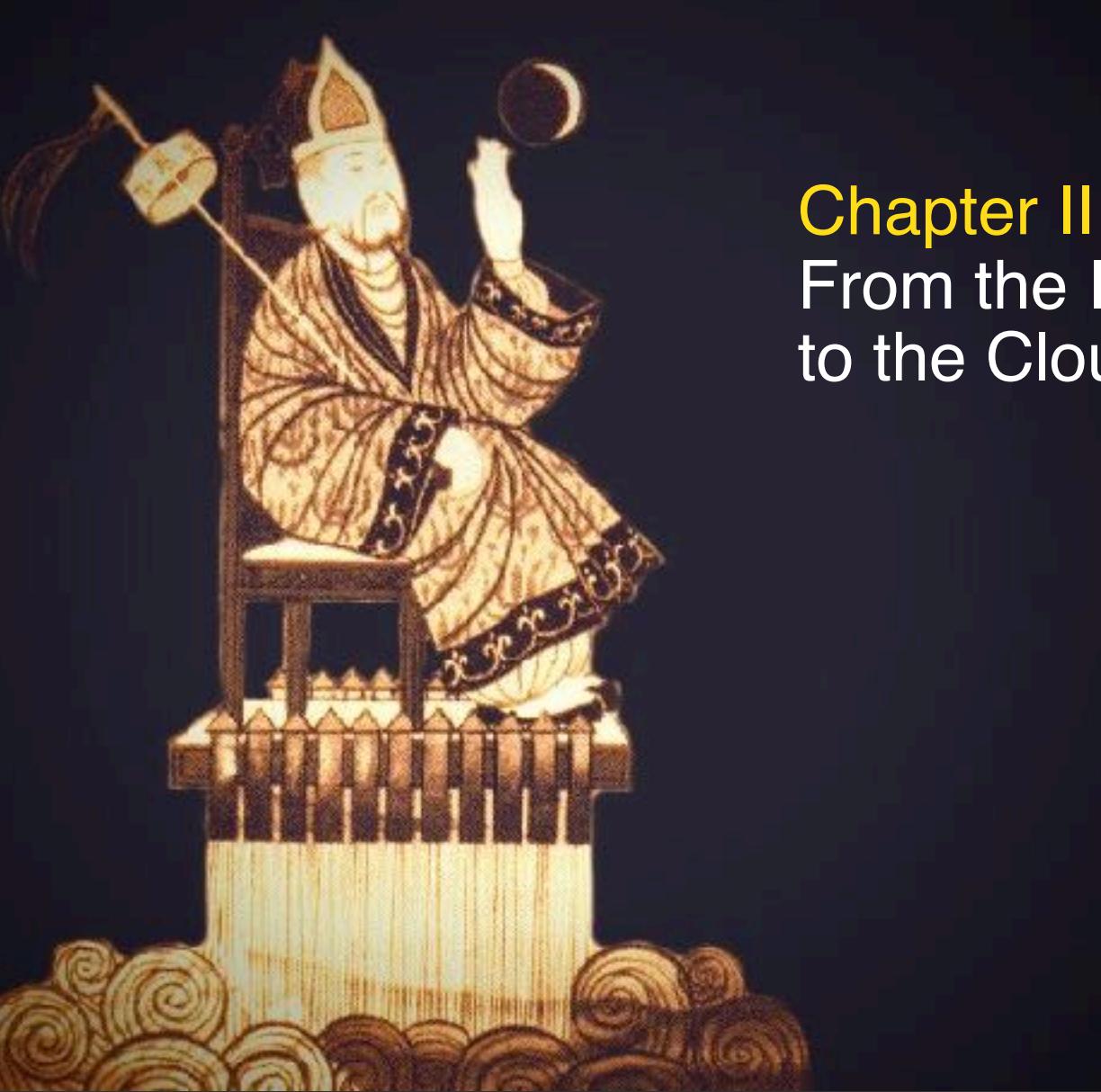
Platform as a service (PaaS)

Function as a service (FaaS)

Serverless architecture

AWS Lambda, Phoenix Environments, API Gateway,

.....



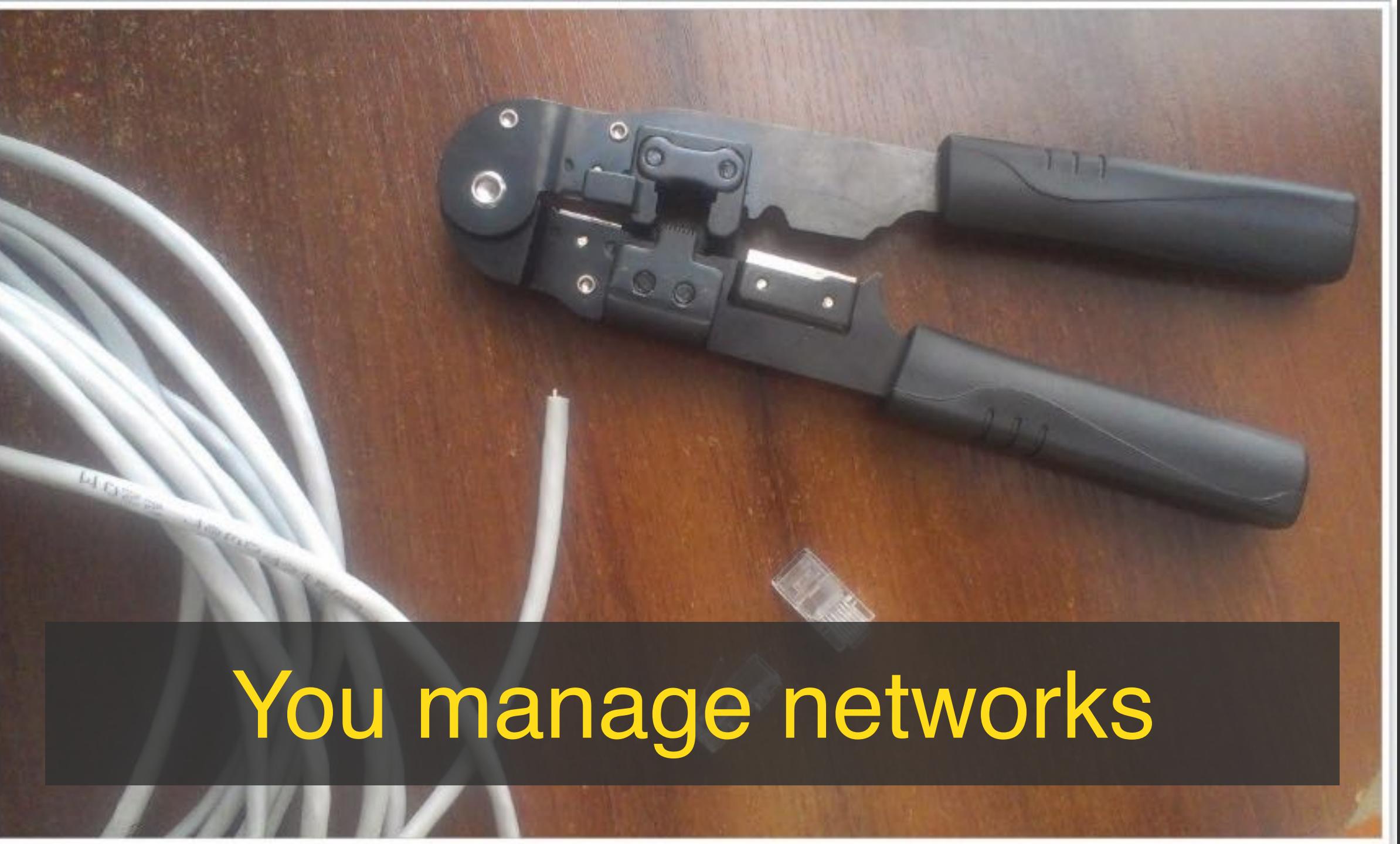
Chapter II:

From the Iron Age to the Cloud Age

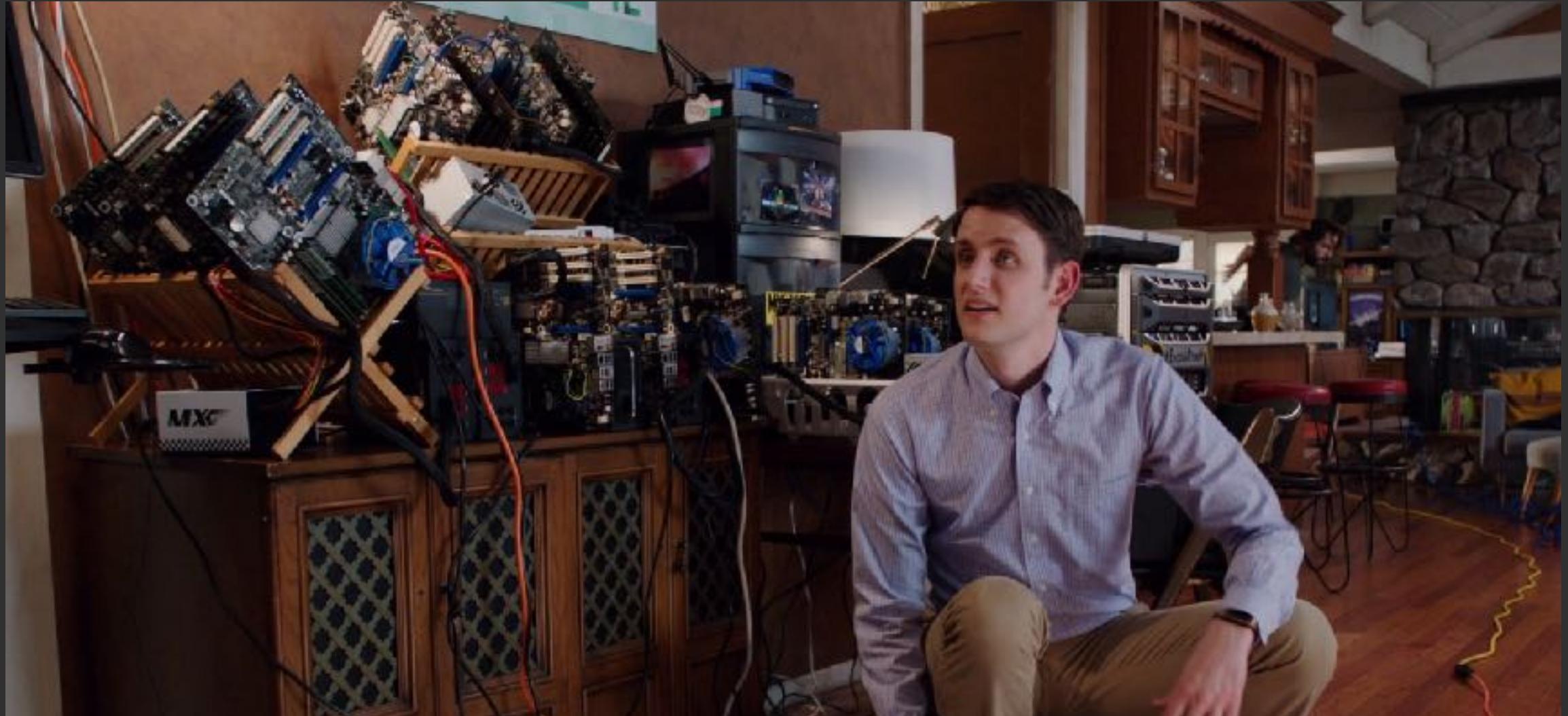
**Self-hosting
(aka On-premise)**



You buy and install hardware



You manage networks



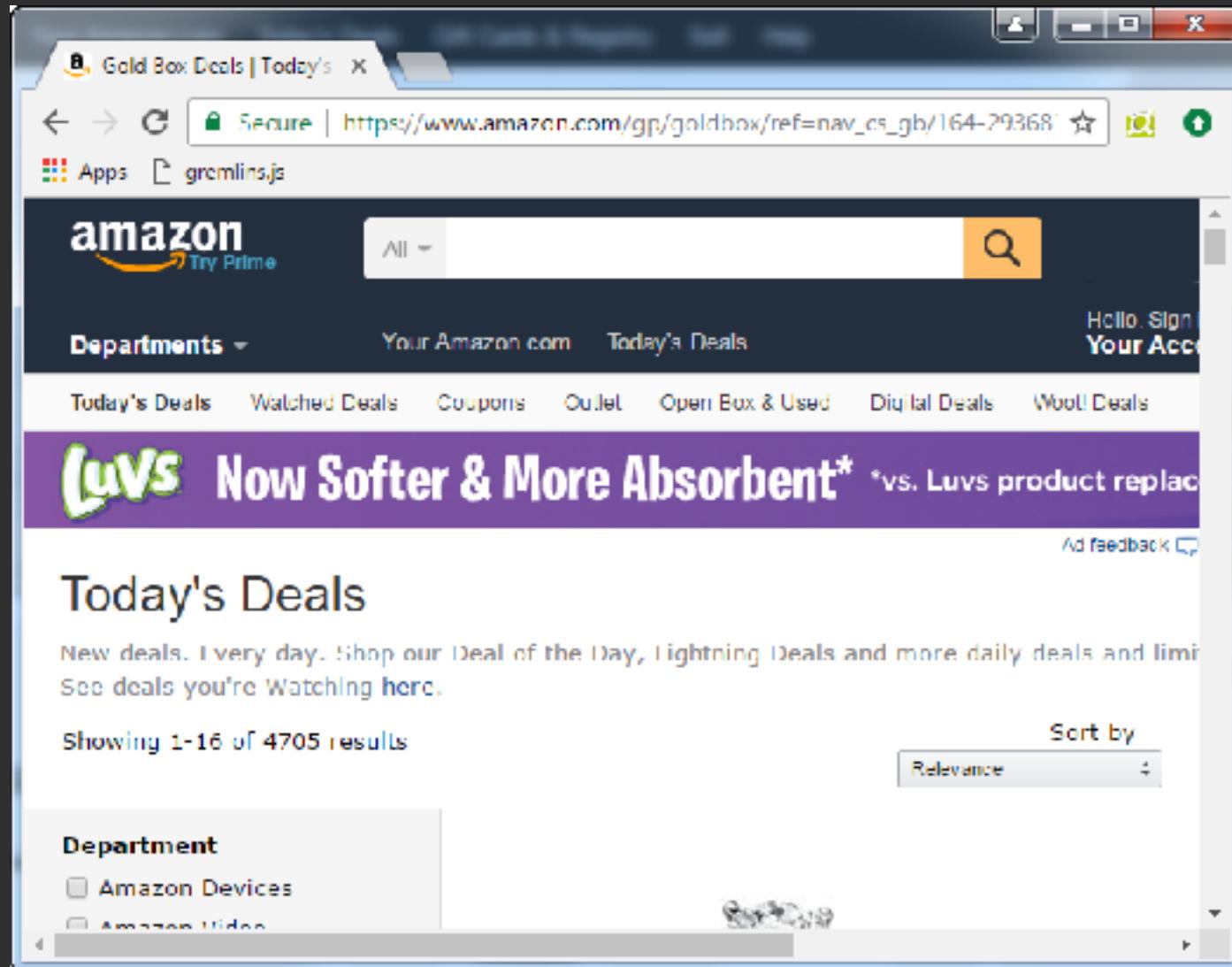
You scale

[Silicon Valley](<http://www.imdb.com/title/tt2575988/>)



You maintain

In Amazon we trust



Survey

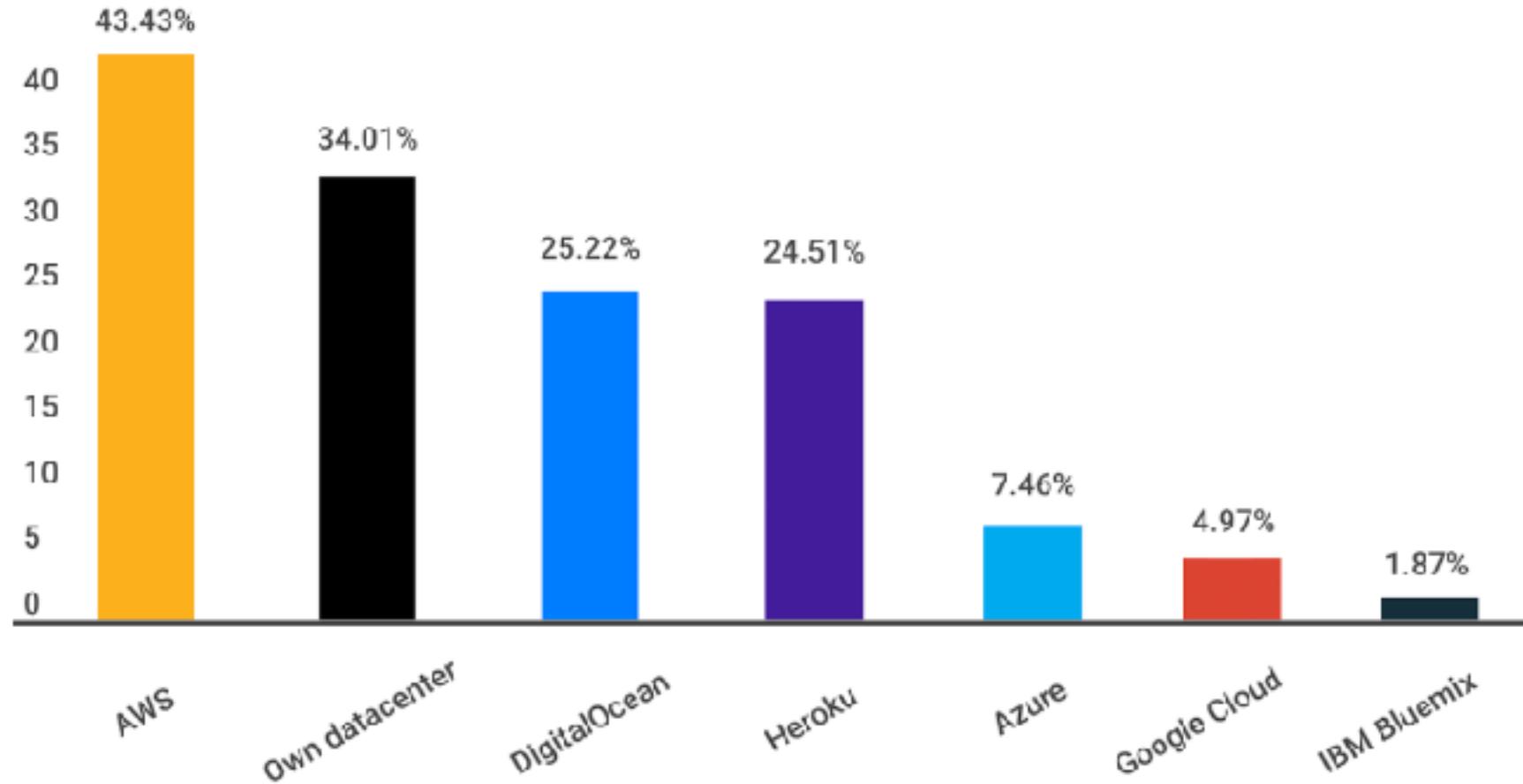
“How Developers use Node.js”

1126 Respondents

<https://blog.risingstack.com/node-js-developer-survey-results-2016/>

Where do you run your Node.js apps?

1126 respondents - multiple choice answers



A painting depicting three medieval or Viking-style figures on horseback in a pastoral landscape. The figure on the left is mounted on a white horse, wearing a conical helmet and holding a sword. The central figure is mounted on a dark horse, wearing a rounded helmet and holding a spear. The figure on the right is mounted on a light-colored horse, wearing a helmet with a tall plume and holding a sword. They are positioned in a field with rolling hills and a cloudy sky in the background.

Three DevOps Survey

Results: 80-90% AWS/Hybrid/ Private cloud



Private cloud

PayPal OpenStack Private Cloud

Physical servers: 8000+

Total cores: 400 000+

Number of VMs: 82000+

Storage: 2 petabytes

Processed \$228 billion in payments last year



Hybrid cloud

Chapter III:

Cloud computing services

Infrastructure as a service (IaaS)

is a form of cloud computing that provides virtualized computing resources over the Internet.

Self Hosting

YOU MANAGE

Applications
Data
Runtime
Middeware
O/S
Virtualization
Servers
Storage
Networking

VS

Infrastructure as a service

YOU MANAGE

Applications
Data
Runtime
Middeware
O/S

VENDOR

Virtualization
Servers
Storage
Networking

YOU MANAGE

Self Hosting

Applications

Data

Runtime

Middeware

O/S

Virtualization

Servers

Storage

Networking

VS

Infrastructure as a service

YOU MANAGE

VENDOR



Virtualization
Servers
Storage
Networking

40 servers to 5000 in 3 days



Case Study

Number of instances

Steady state of ~40 instances

4/12/2008 4/13/2008 4/14/2008 4/15/2008 4/16/2008 4/17/2008 4/18/2008 4/19/2008 4/20/2008

EC2 scaled to peak of 5000 instances

"Techcrunched"

Launch of Facebook modification

IAAS benefits

- Infrastructure in minutes
- Reduced operational cost
- Pay as you go
- Worldwide
- Scalability and flexibility
- Anytime, Anywhere, Access

“add dev/prod env, env in japan, add 1000 servers or remove 1000 servers - by clicking a button”

IAAS: INAPPROPRIATE USAGE EXAMPLES

- Where regulatory compliance makes the offshoring or outsourcing of data storage and processing difficult
- Where the highest levels of performance are required, and on-premise or dedicated hosted infrastructure has the capacity to meet the organization's needs

 Compute	 Developer Tools	 Analytics	 Application Services
EC2	CodeStar	Amazon Athena	Step Functions
Amazon Container Service	CodeCommit	Amazon EMR	AWS Lambda
Amazon Lightsail	CodeBuild	Amazon CloudSearch	Amazon API Gateway
Amazon Elastic Beanstalk	CodeDeploy	Amazon Elasticsearch Service	Amazon Kinesis
Amazon Lambda	CodePipeline	Amazon Kinesis Data Pipeline	Amazon Transcribe
Amazon Batch	X-Ray	Amazon QuickSight	
 Storage	 Management Tools	 Artificial Intelligence	 Messaging
Amazon S3	Amazon CloudWatch	Amazon Lex	Amazon Simple Queue Service
Amazon EBS	Amazon CloudFormation	Amazon Polly	Amazon Simple Notification Service
Amazon Glacier	Amazon CloudTrail	Amazon Rekognition	Amazon SES
Amazon Storage Gateway	Amazon Config	Amazon Machine Learning	
 Database	Amazon Service Catalog	 Internet Of Things	 Business Productivity
Amazon RDS	Amazon Trusted Advisor	AWS IoT	Amazon WorkDocs
Amazon DynamoDB	Amazon Managed Services	AWS Greengrass	Amazon WorkMail
Amazon ElastiCache		 Contact Center	Amazon Amazon Chime
Amazon Redshift		Amazon Connect	
 Networking & Content Delivery	 Security, Identity & Compliance	 Game Development	 Desktop & App Streaming
Amazon VPC	Amazon IAM	Amazon GameLift	Amazon WorkSpaces
Amazon CloudFront	Amazon Inspector		Amazon AppStream 2.0
Amazon Direct Connect	Amazon Certificate Manager		
Amazon Route 53	Amazon Directory Service		
	Amazon WAF & Shield		
	Amazon Compliance Reports		
 Migration		 Mobile Services	
Amazon Application Discovery Service		Amazon Mobile Hub	
Amazon DMS		Amazon Cognito	
Amazon Server Migration		Amazon Device Farm	
Amazon Snowball		Amazon Mobile Analytics	
		Amazon Pinpoint	

AWS Spot Market

“At the time of writing this (October 2016), an On-Demand r3.2xlarge instance (with 8 vCPUs and 61 GB RAM) costs \$0.665 per hour in the US East region. However, the current Spot price for that instance is only \$0.10... a cost savings of over 80%!”

<https://databricks.com/blog/2016/10/25/running-apache-spark-clusters-with-spot-instances-in-databricks.html>

AWS Spot Market

“Foursquare has started using the service, performing analytics across more than three million daily checkins and has reduced analytics costs by more than 50%. They have decreased processing time for urgent data-analysis, all without requiring additional application development or adding risk to the company’s analytics.”

<https://siliconangle.com/blog/2011/08/18/amazon-web-services-offers-spot-market-capabilities-for-managing-hadoop-clusters/>

Platform as a service

Platform as a service

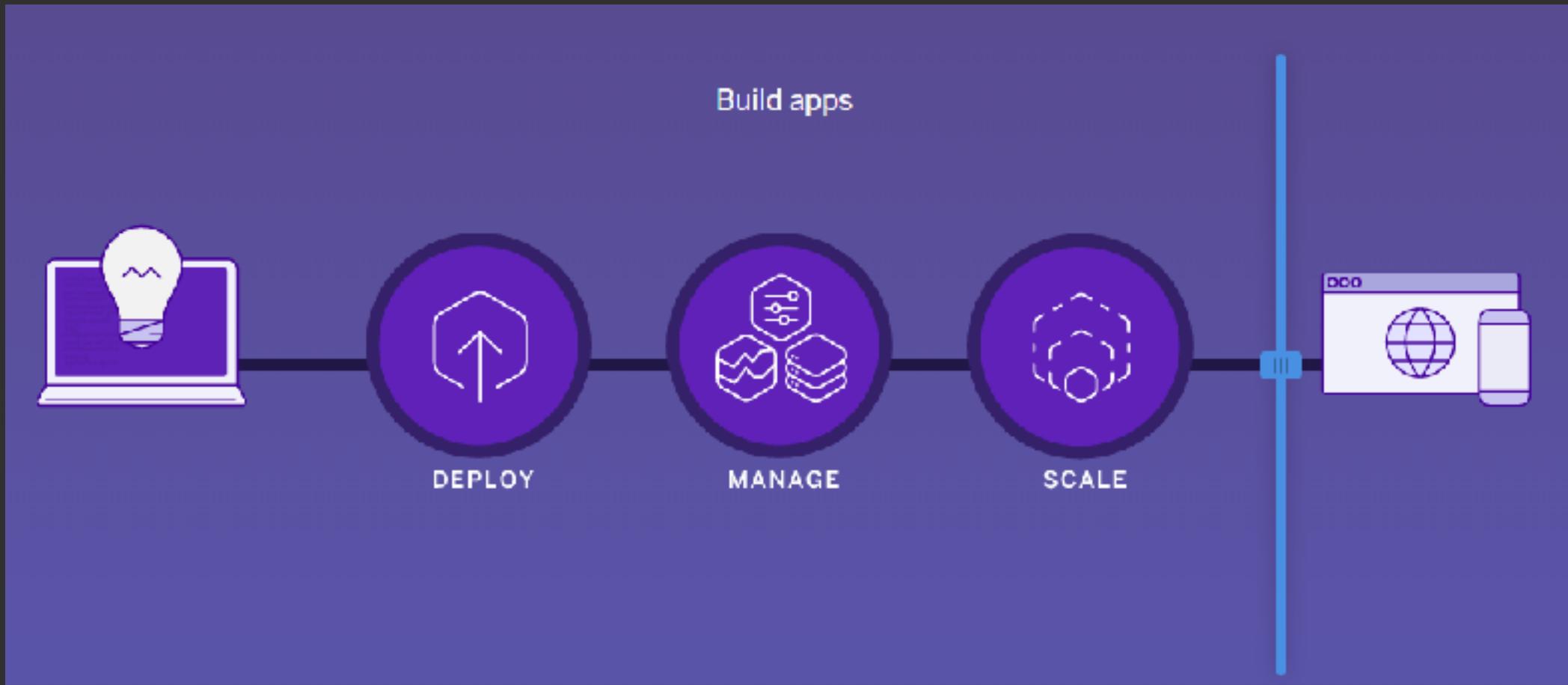
Applications
Data

Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking

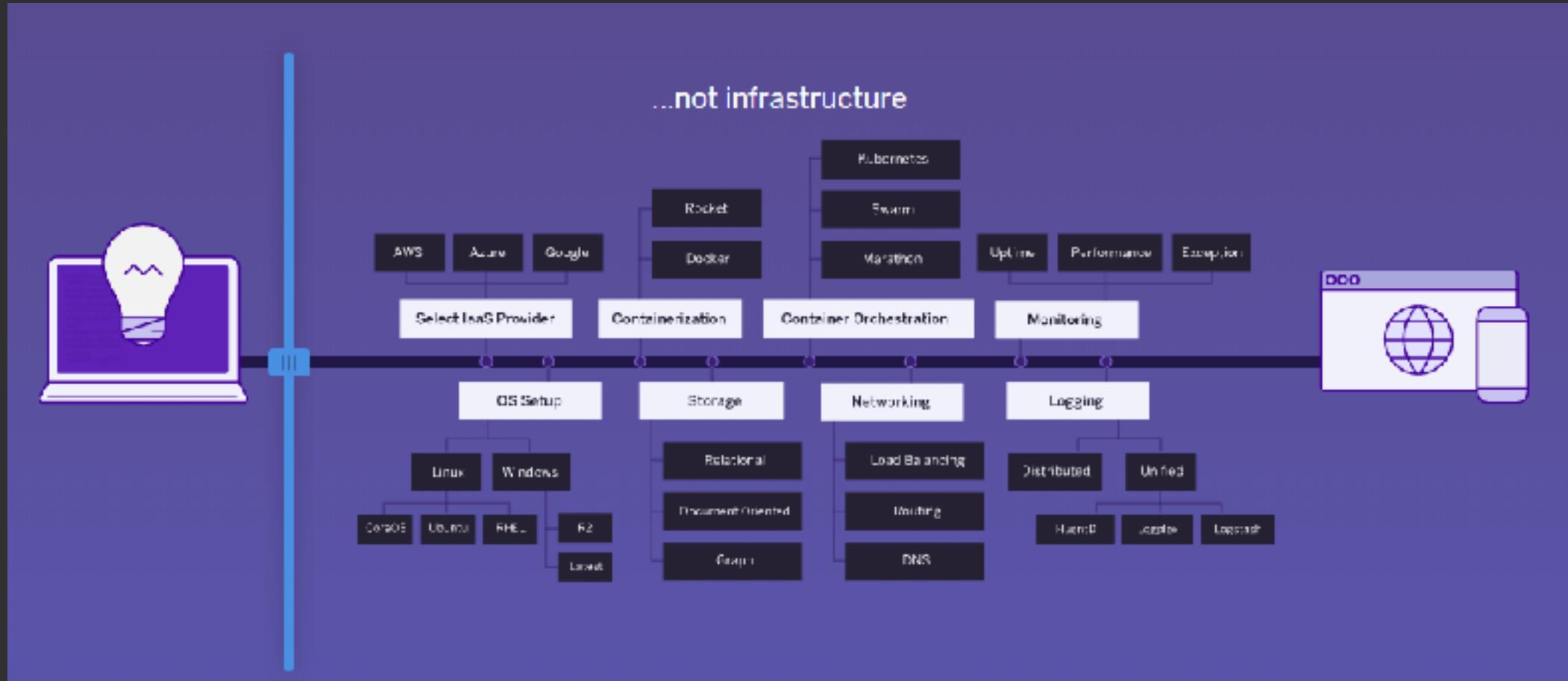
YOU

VENDOR MANAGE

Heroku: “Build apps...



Heroku: “... not infrastructure”



Chapter III:

Scalability and cost basics

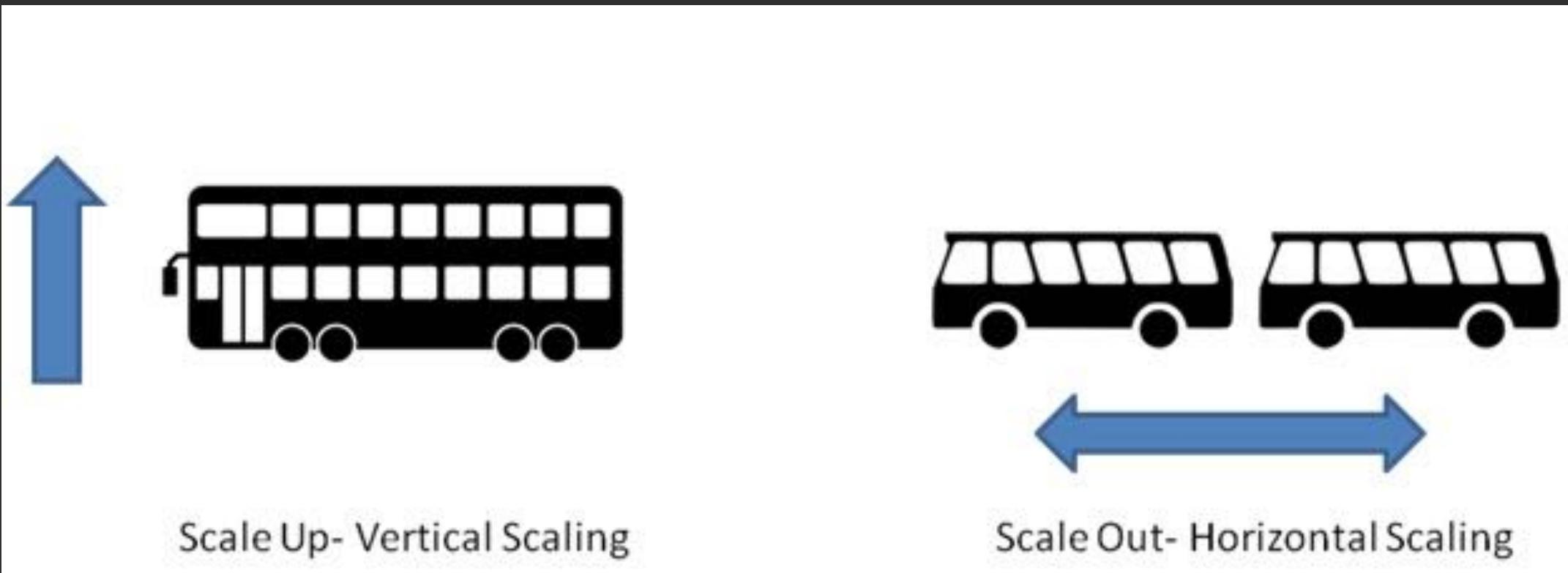
Classic Client-Server Architecture



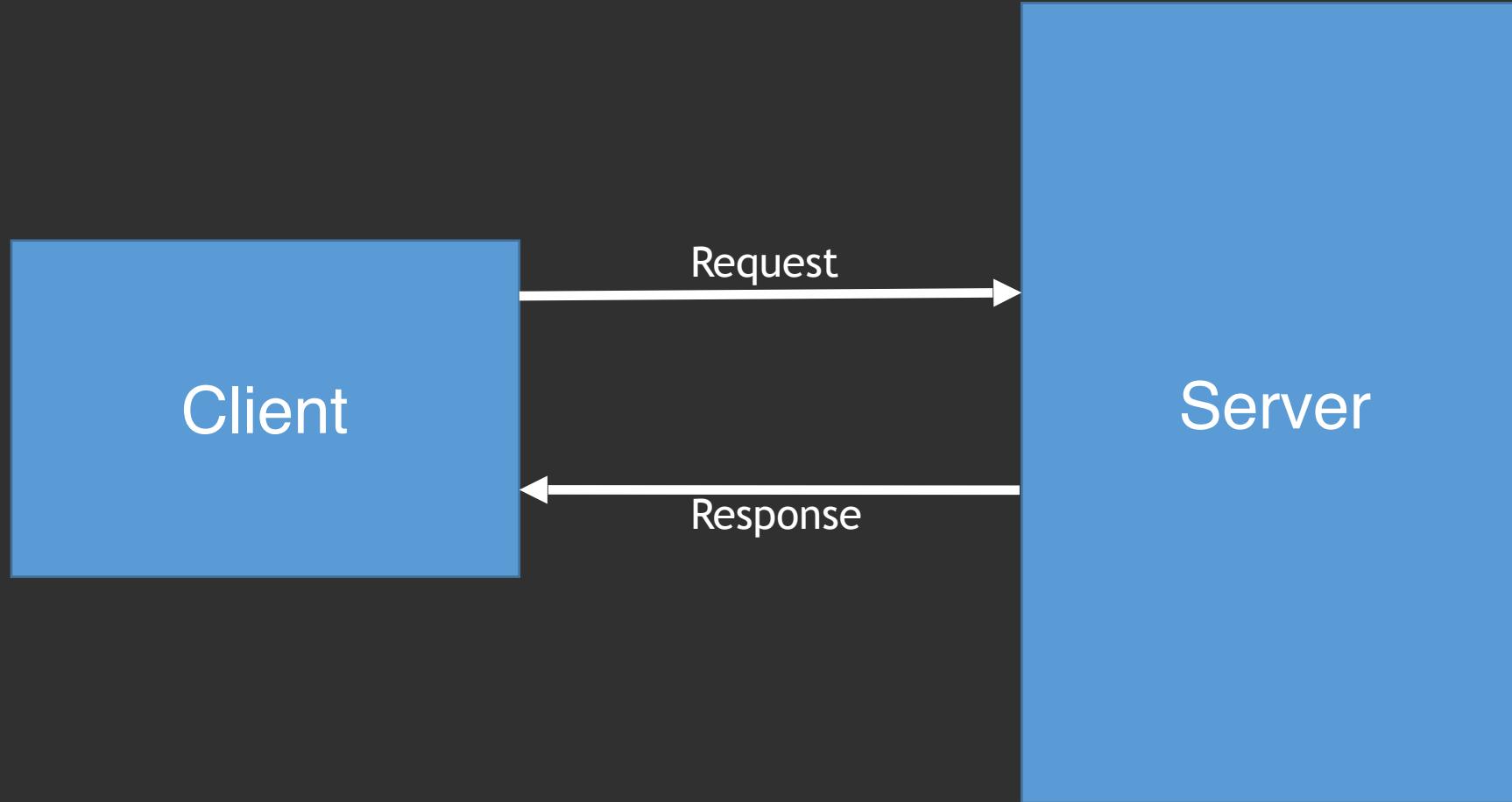
What about scaling?

10 users -> 10000+ users

Vertical scaling

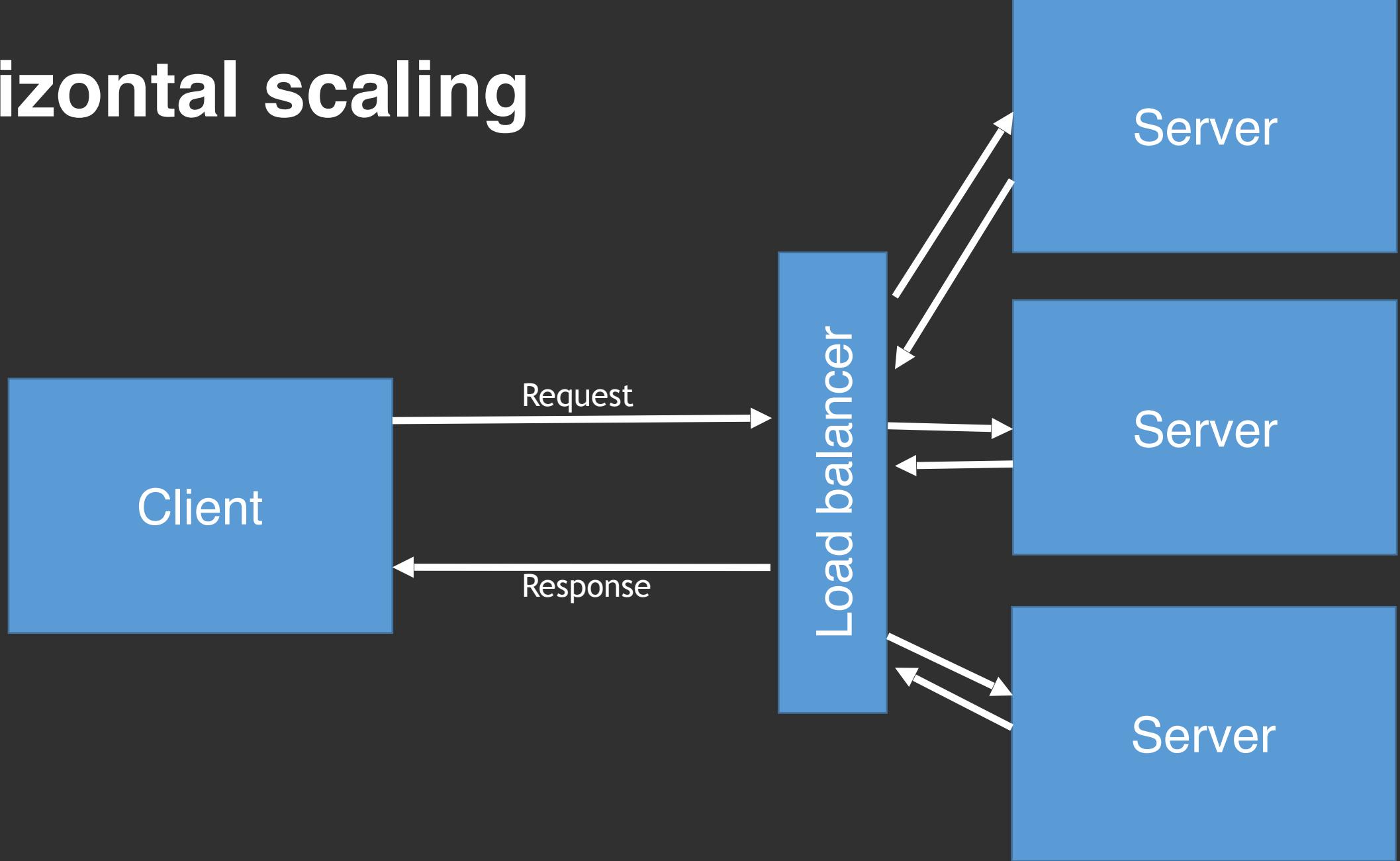


Vertical scaling



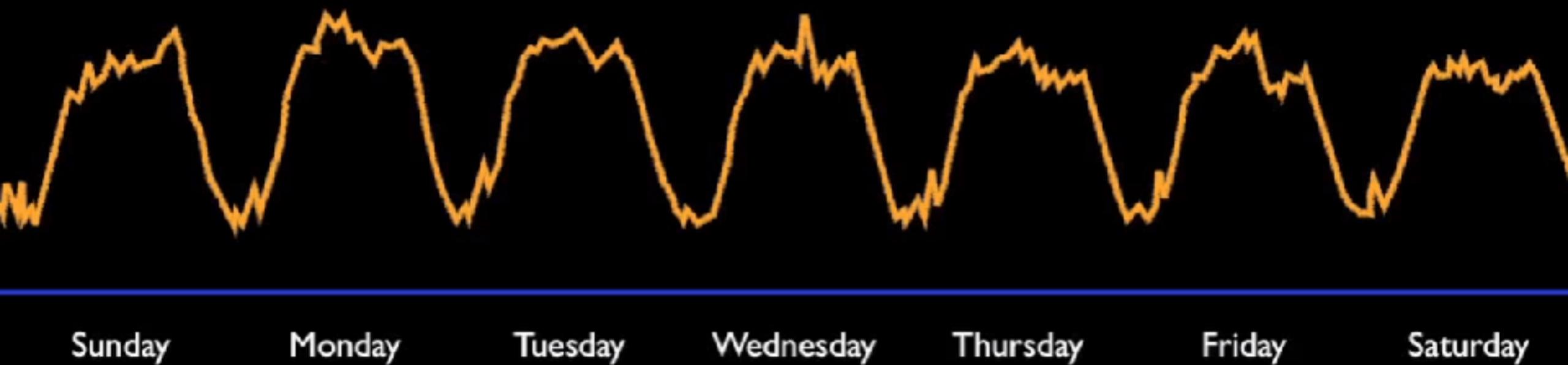
That's enough for 90 percent of projects

Horizontal scaling



Traffic story

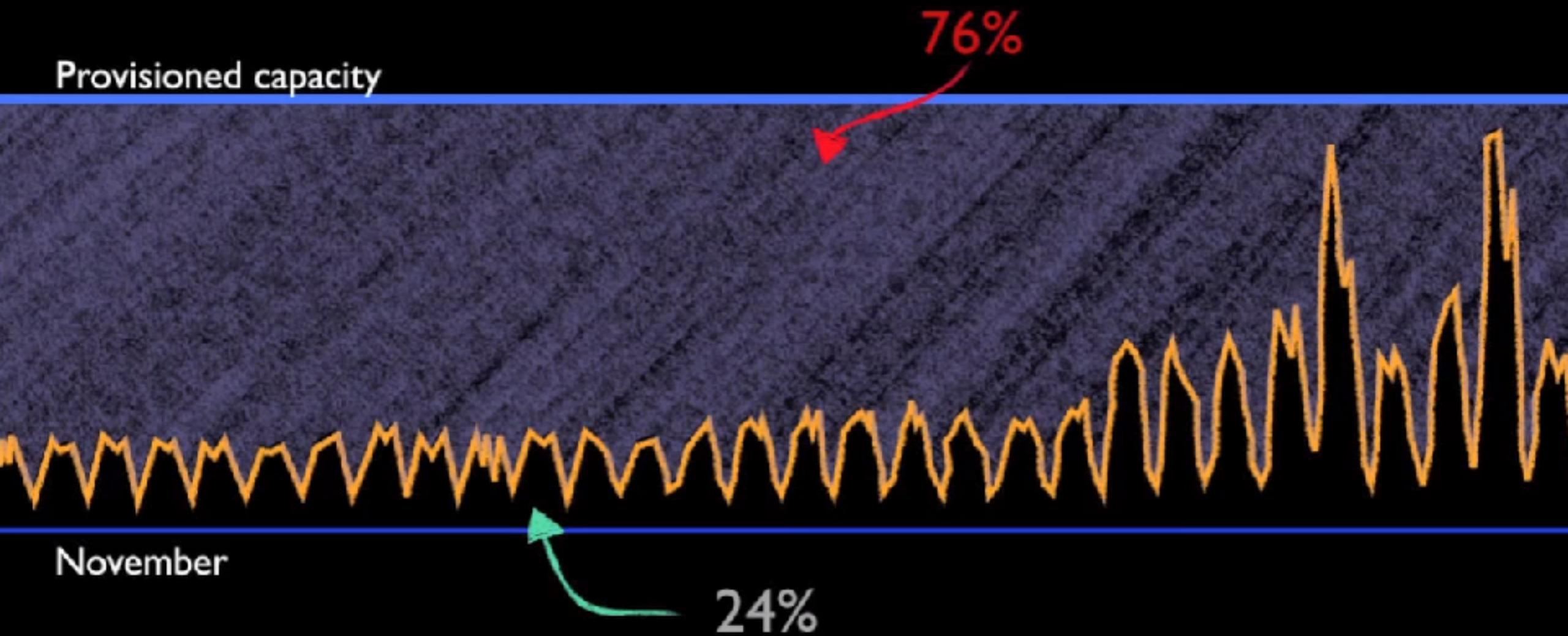
Typical weekly traffic to Amazon.com



<https://www.youtube.com/watch?v=DERzYnthq1s>

Black Friday

November traffic to Amazon.com



Chapter IV: Function as a service

Serverless architecture

1. Infinite scaling
2. Only pay for the compute that you need
3. No servers to manage

Doug McIlroy distilled the Unix Philosophy down to
*“Write programs that do one thing and do it well. Write programs
to work together. Write programs to handle text streams, because
that is a universal interface.”*

Adapting the Unix Philosophy for the serverless world is simply a matter of changing the interface.

“Write programs that do one thing and do it well. Write programs to work together. Write programs to handle https, because that is a universal interface”.

Function as a service

YOU

VENDOR MANAGE

Applications
Data

Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking

Function as a Service

“Magic computer power that comes into existence on request and disappears immediately after use”

(c) ThoughtWorks Radar

Function as a Service

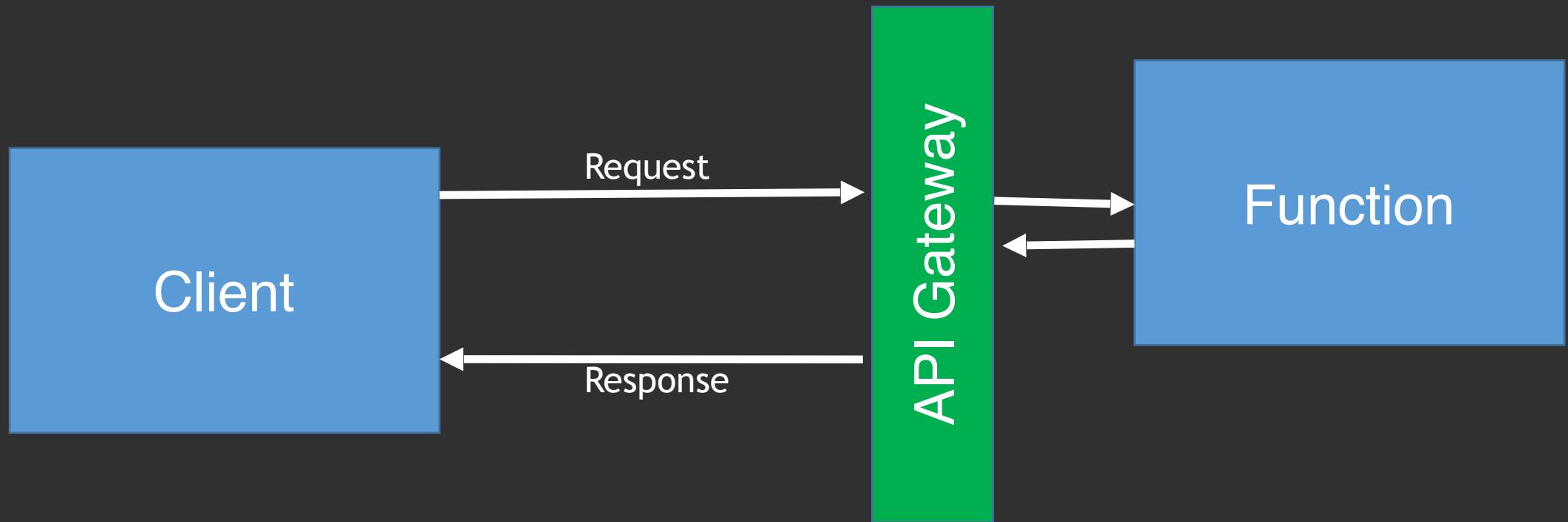


Amazon API Gateway

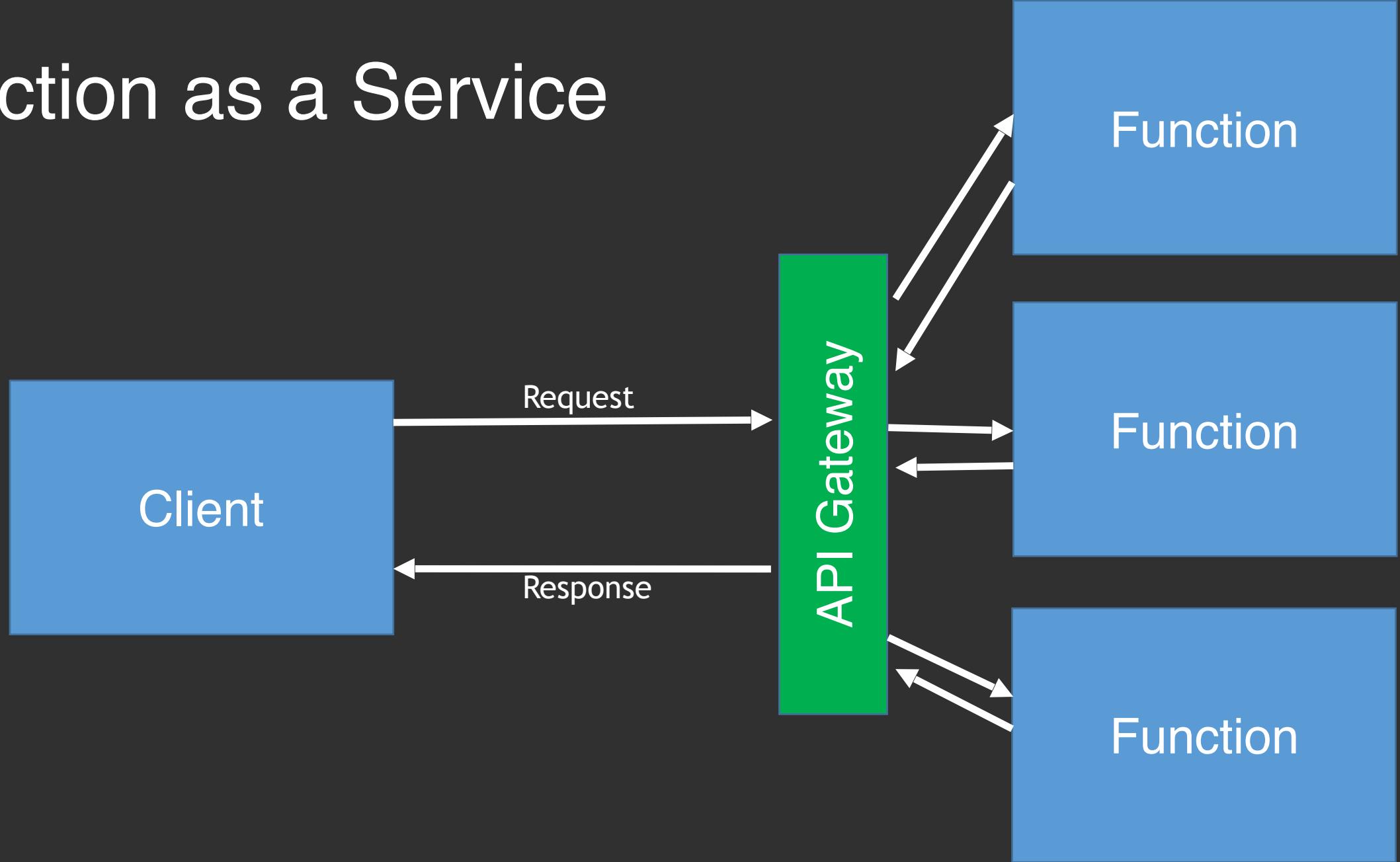
 Fully managed and scalable **RESTful**
API gateway service

 Powered by our **content delivery
network** via 59 global edge locations

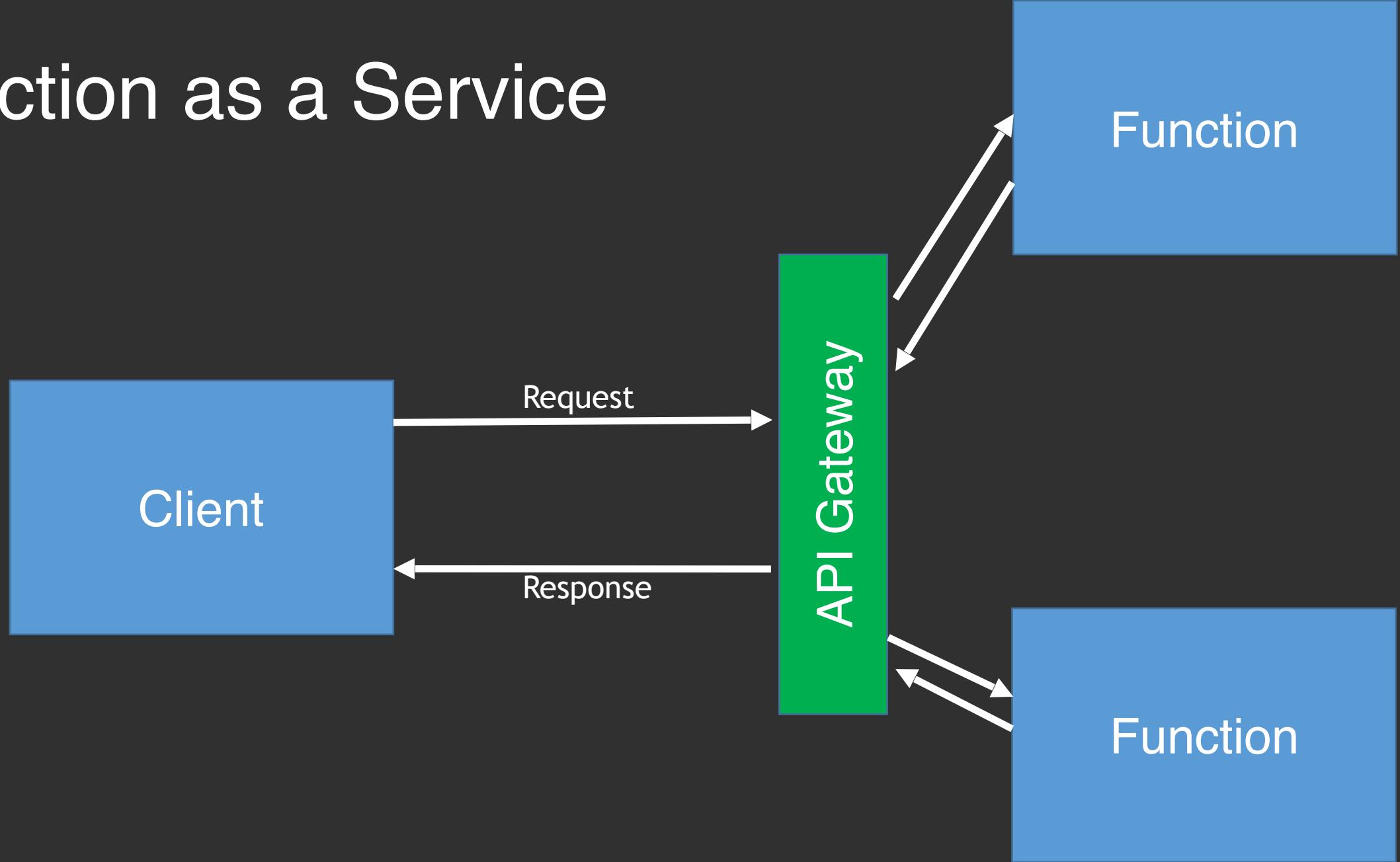
Function as a Service



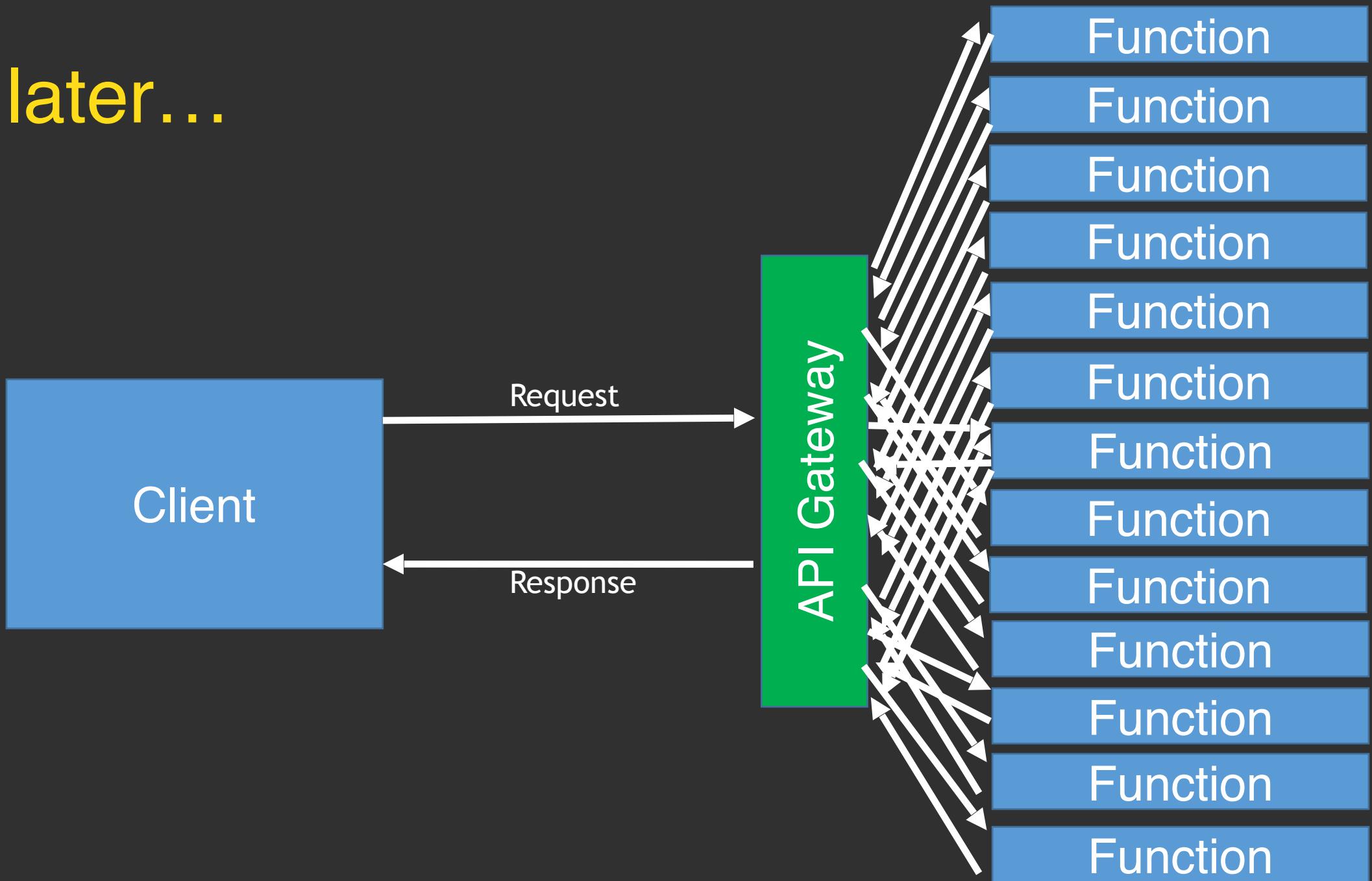
Function as a Service



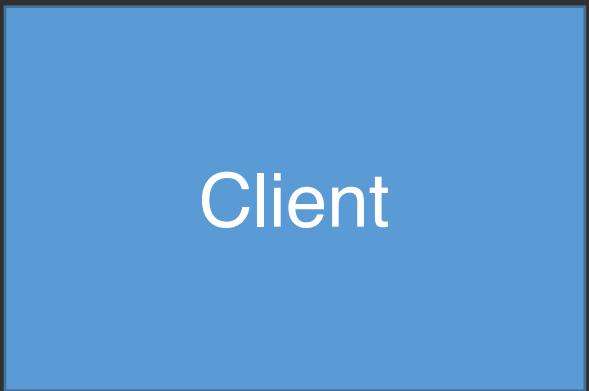
Function as a Service



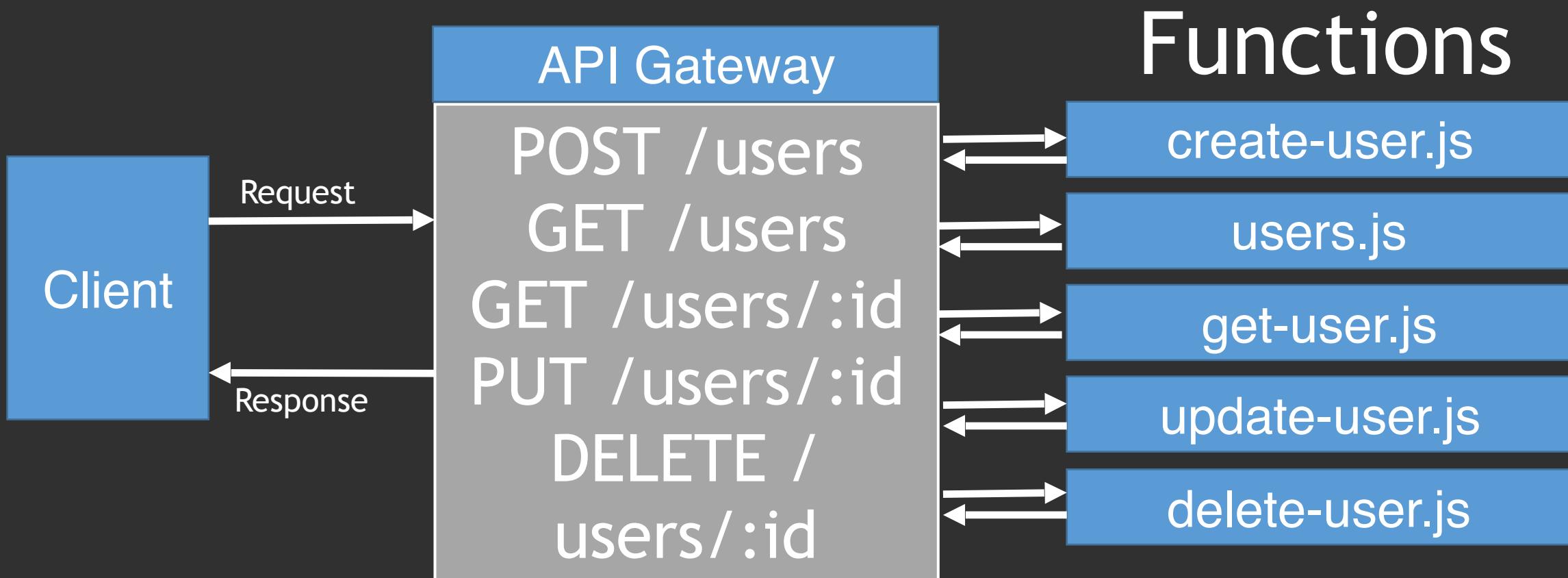
30s later...



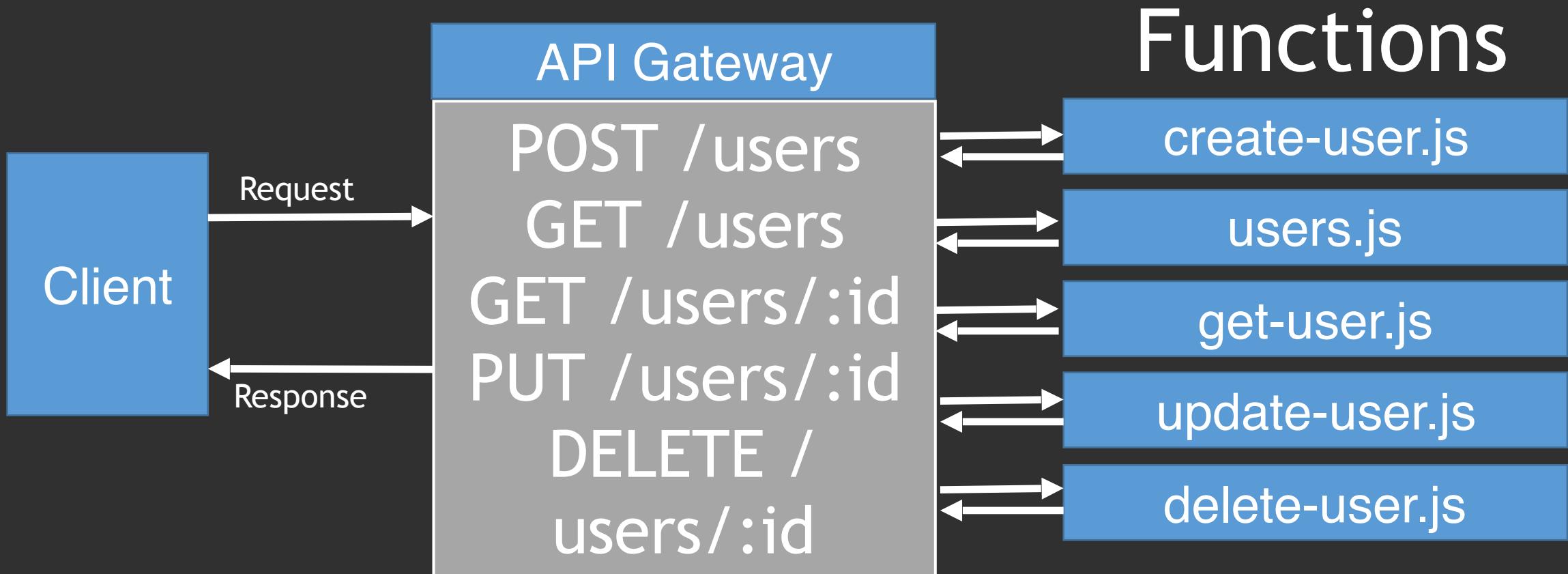
60s later...



Nanoservices



Granular Deploy, Update, Scale, Reliability, etc.



Use case #1

The screenshot shows a web browser window with the title "Rolling Scopes School". The address bar displays "Github, Inc. [US] https://github.com/rolling-scopes-school/". The page content includes the Rolling Scopes School logo, the text "Rolling Scopes School organisation", location "Minsk", and links for "Repositories" and "People". A search bar at the bottom is set to "Find a repository...".

The screenshot shows a Google Sheets spreadsheet titled "Score". The sheet contains the following data:

	A	B	C	D	E	F	G
1	H						
2	Status	Name	Github				
3	magic factor			1	0.1		
4	Slytherin - Stanislau Zubovitch						
5	зачислен(а)	Maksim Shestsel	Sheste	25	100		
6	зачислен(а)	Kartin Dmitry	KartinDmitry	20	100		
7	зачислен(а)	VARVARA LURAVIIS	Iuravlis		100		
8	зачислен(а)	Vladislau Averyn	Averin-Vladislav	30	100		
9	зачислен(а)	Natalia Sasinovich	nsasinovich		100		

GitHub
Webhooks

Request →
← Response

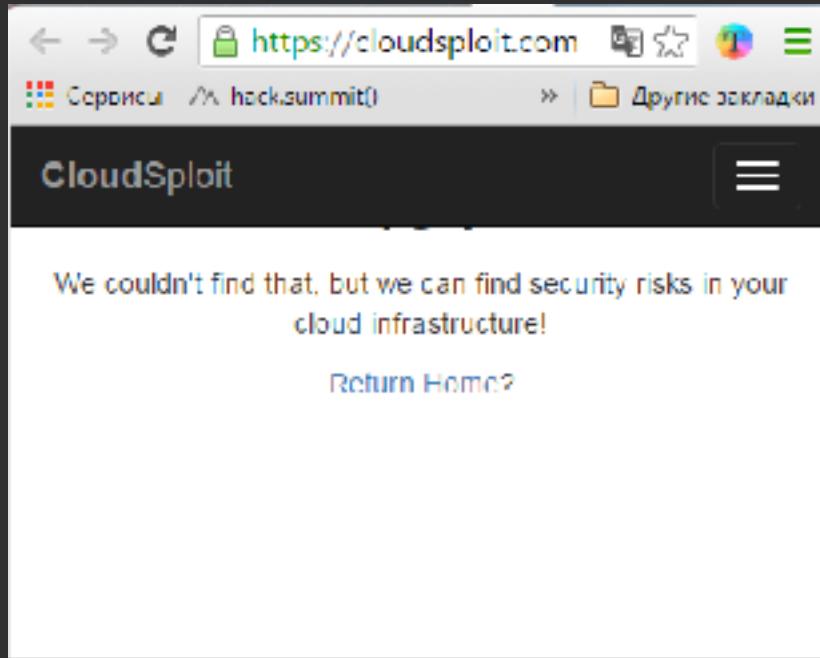
API Gateway

Functions
set-mark.js

Google Spreadsheets Data API

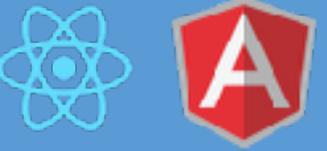
Use case #2

Story: We Made the Whole Company “Serverless”



<https://blog.cloudsploit.com/we-made-the-whole-company-serverless-5a91c27cd8c4#.asm49udxm>

Client



React/Angular



AWS S3

API Gateway

Cloud

users.js

search.js

etc.js

<https://github.com/hoppula/react-apig-lambda>

GitHub, Inc. [US] | https://github.com/hoppula/react-apig-lambda

Apps | [cremlins.js](#)

webpack.apexconfig.js	Initial commit	5 days ago
webpack.config.js	Initial commit	5 days ago

[README.md](#)

react-apig-lambda

Render React.js on-demand with CDN caching.

Minimal example on how to render React & React Router v4 with Amazon API Gateway, AWS Lambda and CloudFront.

Online demo

<https://test.es6.li>

Basic example app from React Router documentation. Initial server-side render and acts as SPA from there.

Dependencies

Function as a Service

Functions are the unit of deployment and scaling

No machines, VMs or containers visible in the programming model

Scales per request, Users cannot over- or under-provision capacity

Service Providers

Amazon Lambda

IBM OpenWhisk

Google Cloud Functions

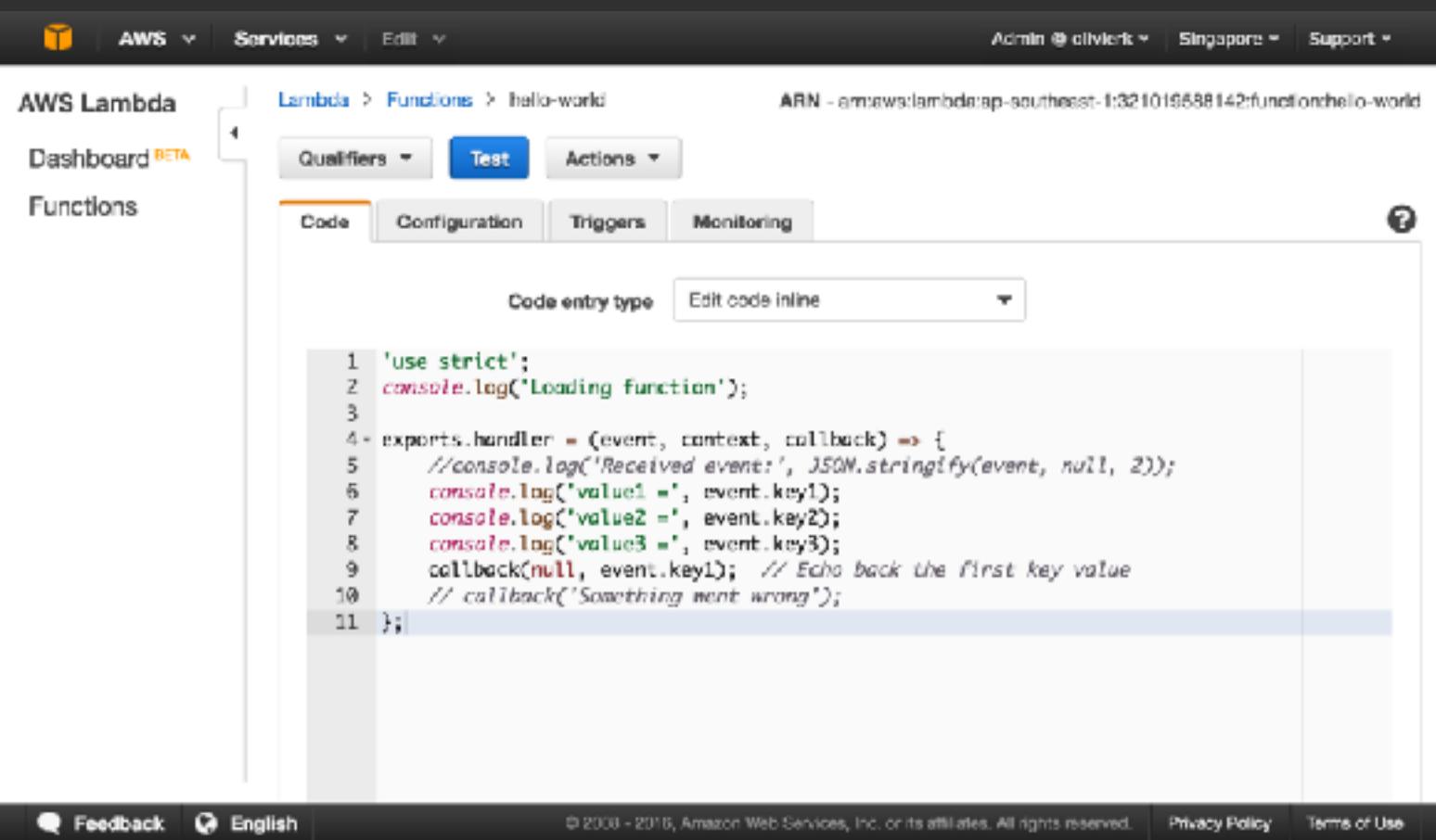
Microsoft Azure Functions

....



AWS Lambda

Supports JavaScript, Java Python

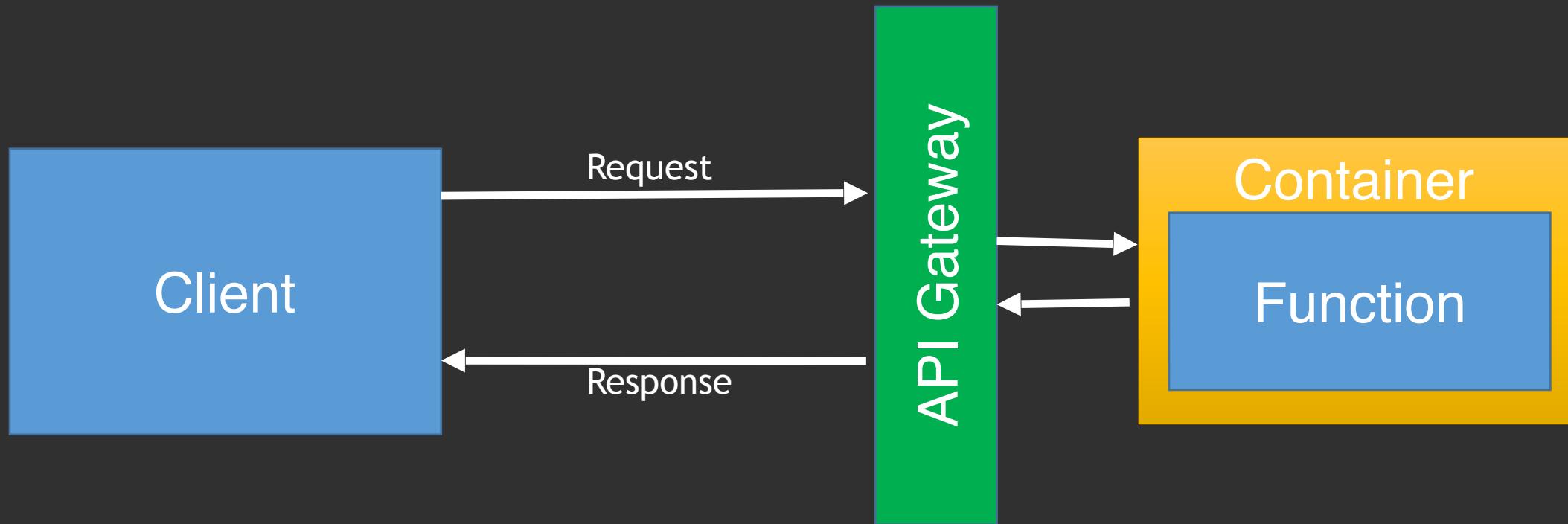


The screenshot shows the AWS Lambda Functions console. The top navigation bar includes the AWS logo, Services dropdown, Edit dropdown, Admin @ chivtek, Singapore, and Support. The main header displays 'Lambda > Functions > hello-world' and the ARN 'arn:aws:lambda:ap-southeast-1:321019588142:function:hello-world'. Below the header, there are tabs for Qualifiers, Test, Actions, Code (which is selected), Configuration, Triggers, and Monitoring. A dropdown menu for 'Code entry type' is open, showing 'Edit code inline'. The code editor contains the following JavaScript code:

```
1 'use strict';
2 console.log('Loading function');
3
4 exports.handler = (event, context, callback) => {
5   //console.log('Received event:', JSON.stringify(event, null, 2));
6   console.log('value1 =', event.key1);
7   console.log('value2 =', event.key2);
8   console.log('value3 =', event.key3);
9   callback(null, event.key1); // Echo back the first key value
10  // callback('Something went wrong');
11};
```

At the bottom of the page, there are links for Feedback, English, Copyright notice (© 2006 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.), Privacy Policy, and Terms of Use.

Lambda container



Start-up latency

Node 10-100 ms

Java ??? ms

Cold start / Hot start / Container reuse

If start time is crucial, create a scheduled function to keep the Lambda function “warm” (c)

AWS

What is Lambda container?

Resource	Default Limit
Ephemeral disk capacity	512 MB
<u>Invoke</u> request body payload size (RequestResponse)	6 MB
<u>Invoke</u> response body payload size (RequestResponse)	6 MB

<http://docs.aws.amazon.com/lambda/latest/dg/limits.html>

Maximum
execution
duration
per
request:
5 min



AWS Lambda Deployment Limits

MAX 50Mb (zip) per Function

Default concurrent executions limits Per Region Per Account

1000

NoOps?

Serverless != DevOpsLess

AWS CloudFormation



Phoenix Environments

Using automation, we can create whole environments - including network configuration, load balancing and firewall ports - for example by using [CloudFormation](#) in AWS. We can then prove that the process works by tearing the environments down and recreating them from scratch on a regular basis.

(c) ThoughtWorks Radar

Fine-Grained Pricing



Free Tier
1M requests and 400,000 GBs of
compute.
Every month, every customer.

-  Compute time in 100ms increments
 -  Low request charge
 -  No hourly, daily or monthly
minimums
 -  No per-device fees
- Never pay for idle!**

Gigabytes per second (GBps)

Price =
function's RAM size **X** execution time (sec)

Example: 1GB RAM **X** 5 min = 0.005001\$

ROLLINGSCOPES.COM 😊

Billing: \$11.43/month

Instance utilization: 4%/month 😞

Fond of performance optimization?

Direct relationship between cost and
performance 😊

Serverless architecture

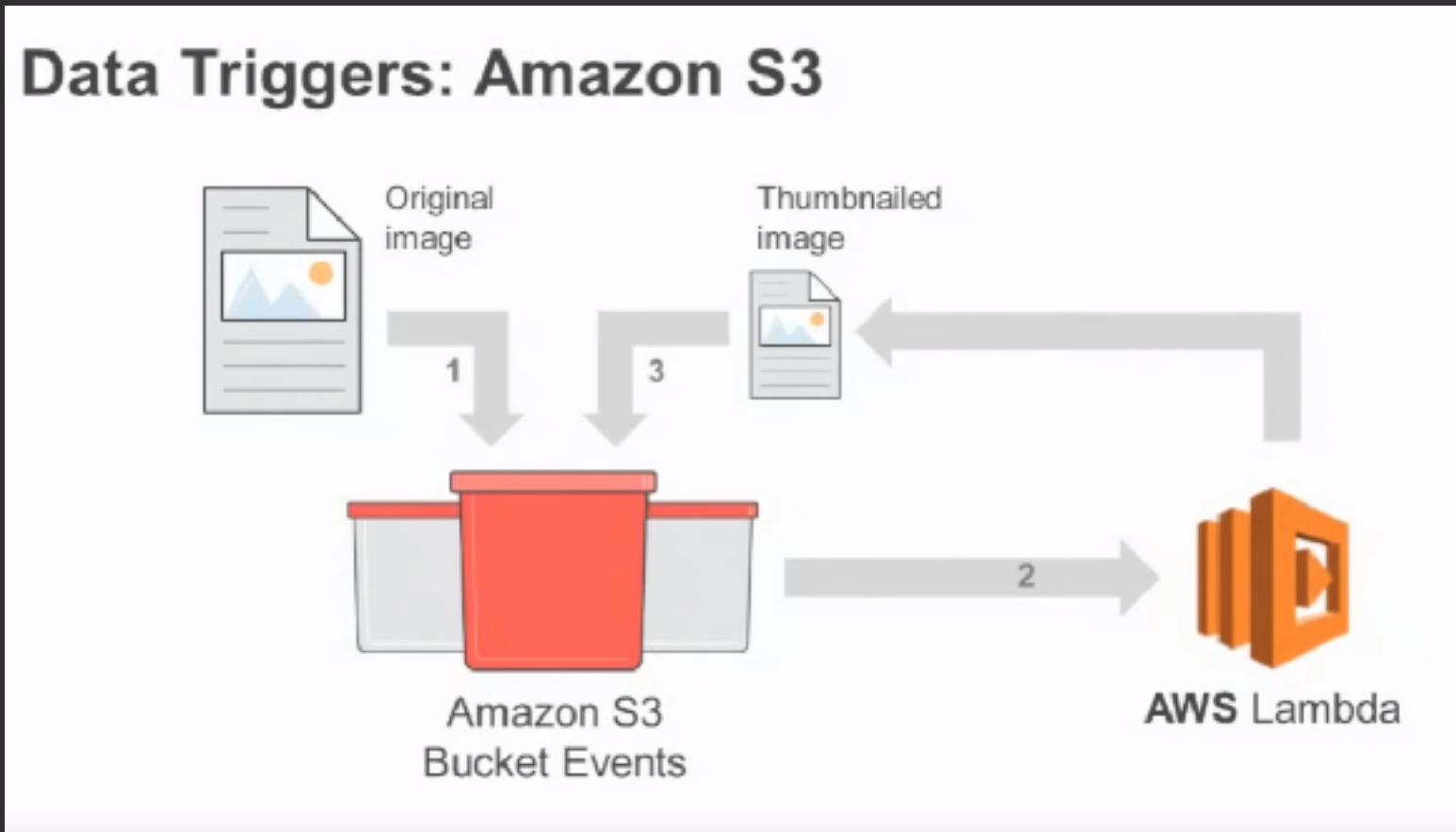
Serverless was first used to describe applications that **significantly or fully depend on 3rd party applications / services** ('in the cloud') to manage server-side logic and state.

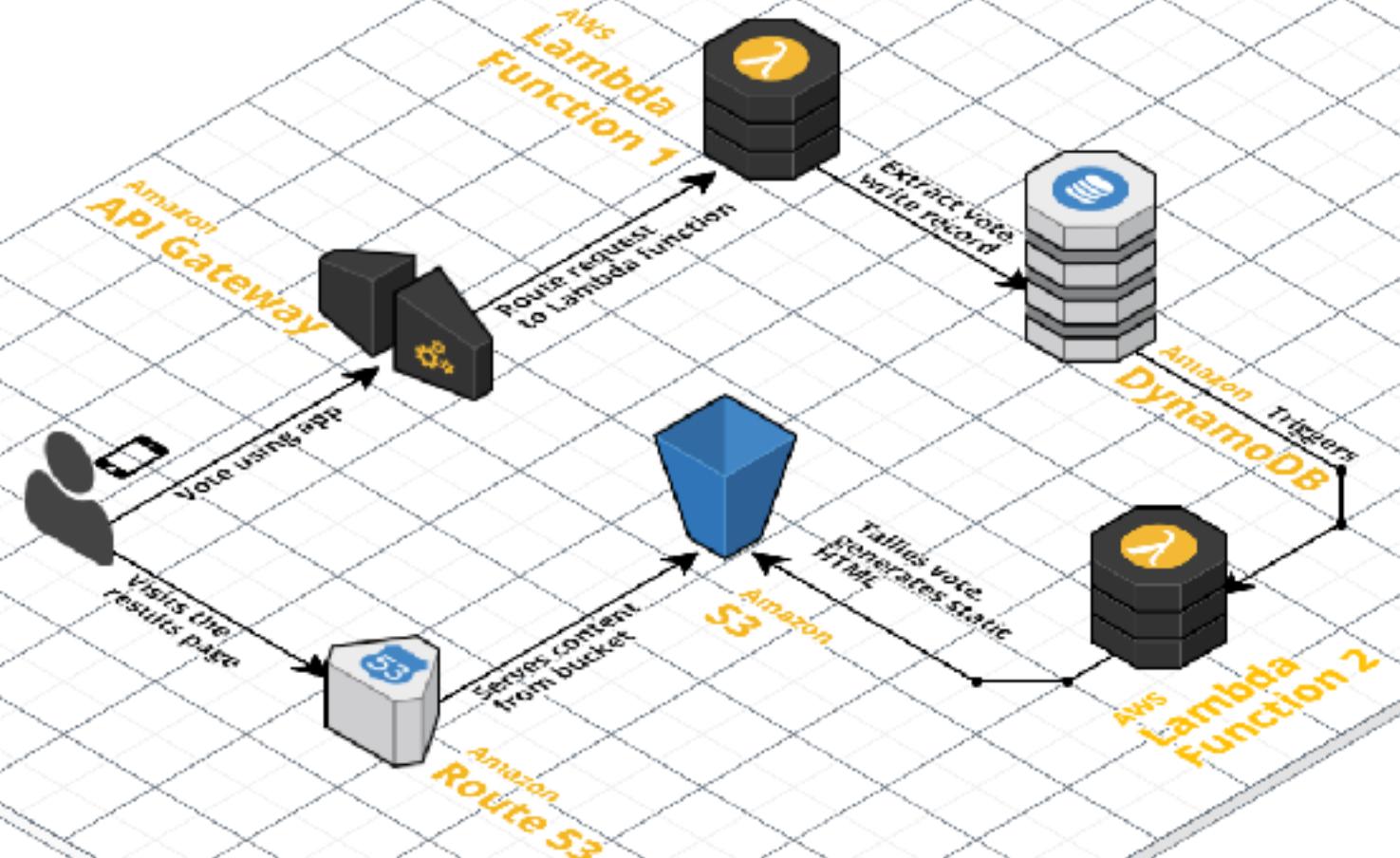
<http://martinfowler.com/articles/serverless.html>

Usual scenario (for web)

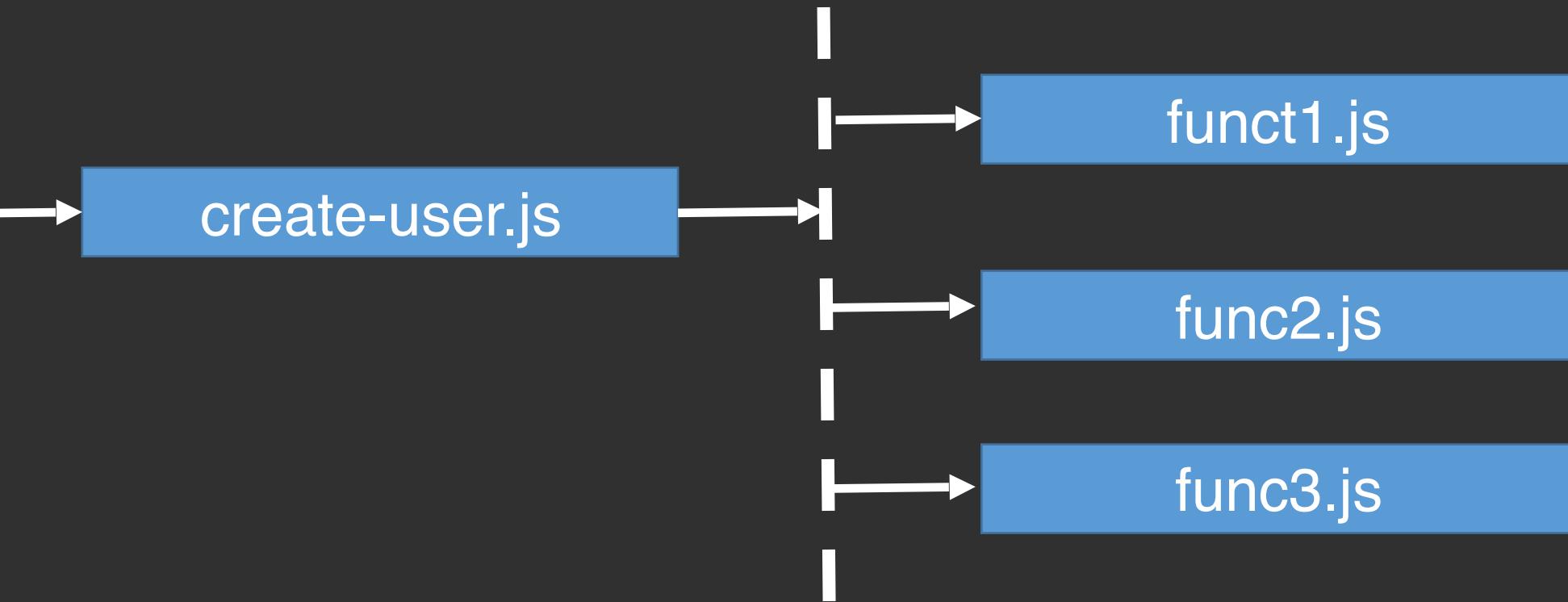


Event-driven Compute in the Cloud





SNS Events



Topic name

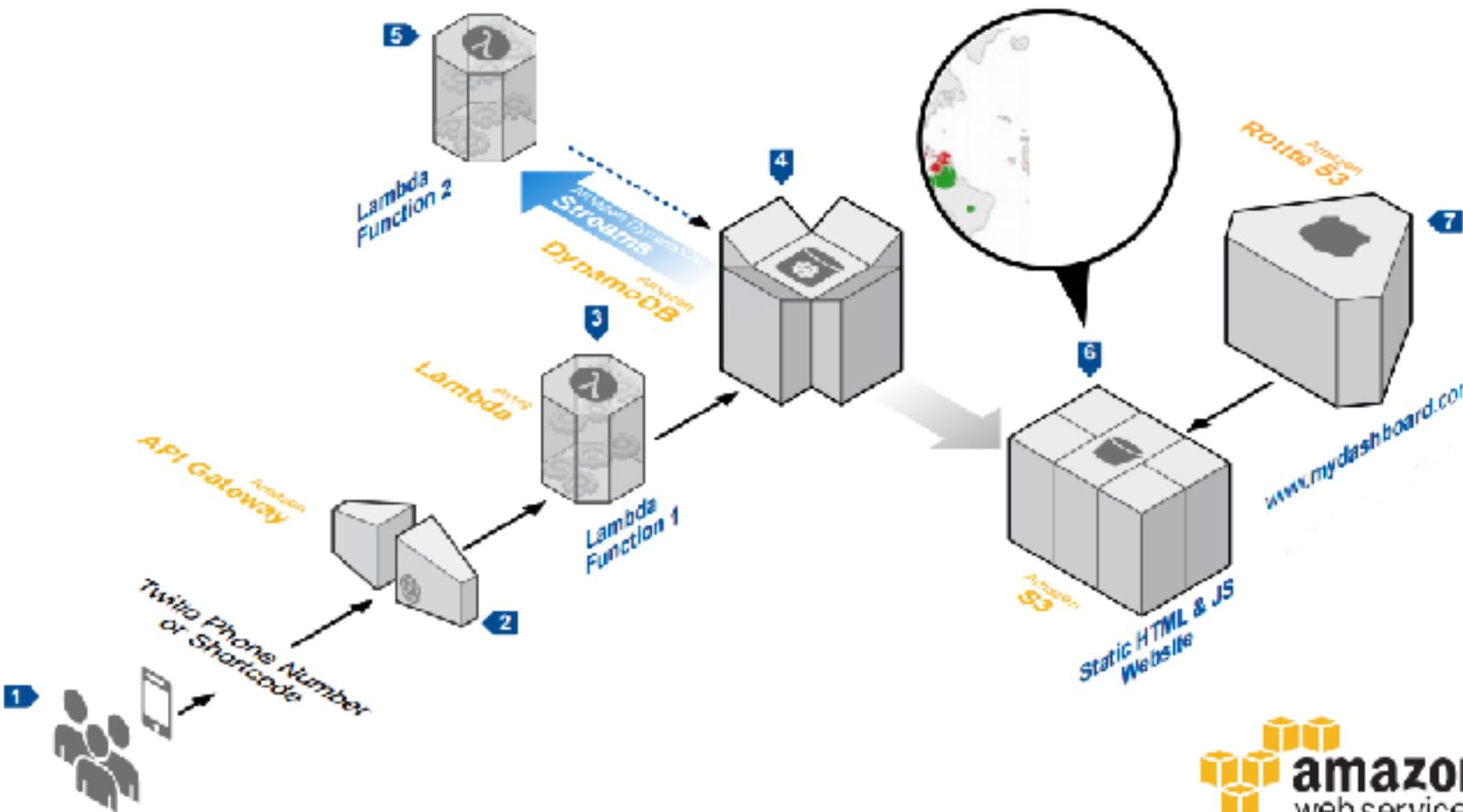
Serverless Reference Architectures with AWS Lambda

- [Web Applications Serverless Reference Architecture](#)
- [Mobile Backend Serverless Reference Architecture](#)
- [Real-time File Processing Serverless Reference Architecture](#)
- [IoT Backend Serverless Reference Architecture](#)
- [Real-time Stream Processing Serverless Reference Architecture](#)

AWS LAMBDA: REAL-TIME VOTING APPLICATION

Consider a dynamic web application that receives votes in real-time. Traditionally, architecting such applications meant building out your infrastructure to support both "spike" and sustained usage over a finite amount of time. In most cases, this required your operator's team to over-provision resources, leading to waste outside of high-volume voting periods.

By combining AWS Lambda with other core AWS services such as Amazon API Gateway, you can build a powerful, highly available web application that automatically scales up and down to handle large amounts of concurrent votes - all with zero administrative effort required. You can also store and analyze your data in well-known services like Amazon DynamoDB and Amazon Simple Storage Service.



System

- 1 Users send a vote to a phone number or shortcode provided by a third party like Twilio.

- 4 metadata into a table in Amazon DynamoDB. This table has DynamoDB Streams enabled, which allows us to react directly to mutations on a rolling basis.

- 6 A dashboard to display a summary of votes is created using HTML and JavaScript, and hosted as a static storage service (Amazon S3). Javascript SDK to query the and display the voting results in

<https://github.com/awslabs/lambda-refarch-webapp>

This function extracts the vote from the message content and writes the result and any

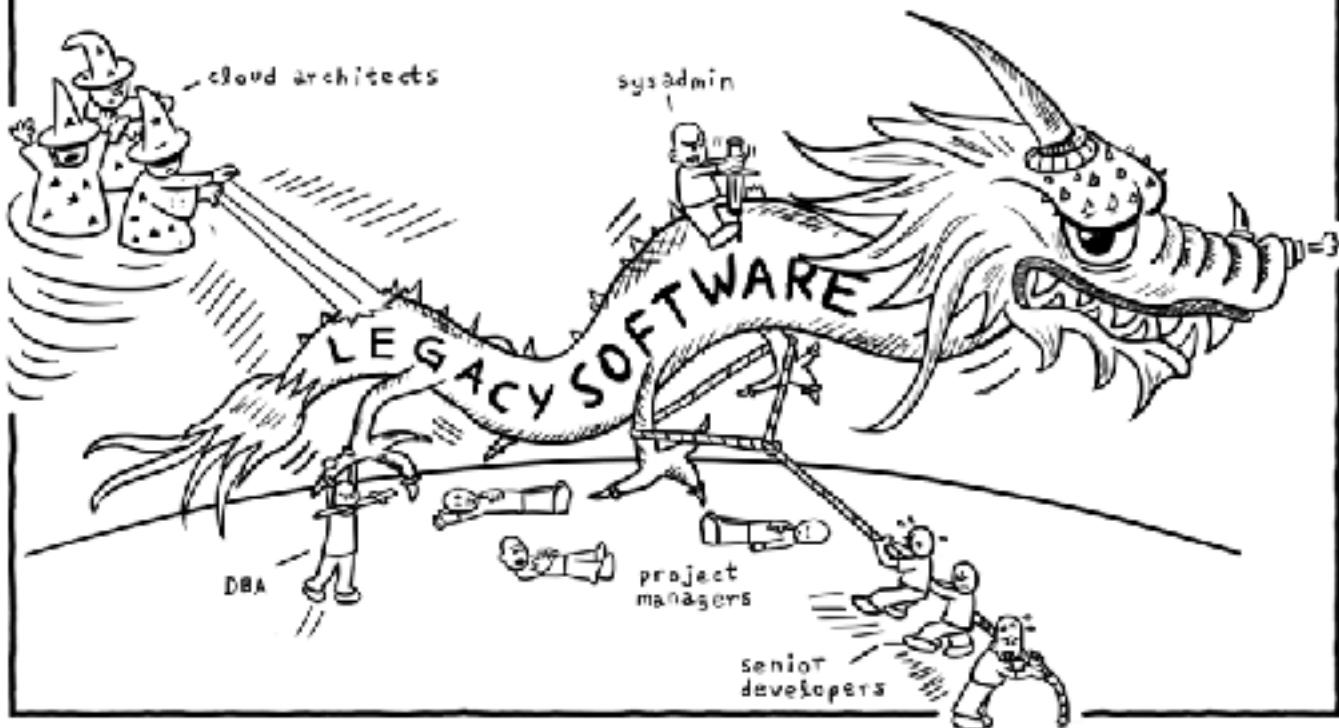
binary. Amazon Route 53 is used as a DNS provider to create a hosted zone pointing a custom domain name to the Amazon S3 bucket.

Drawbacks



Massive Vendor Lock-In

The Enterprise Journey to Cloud



Startups Journey to Cloud



How to debug?

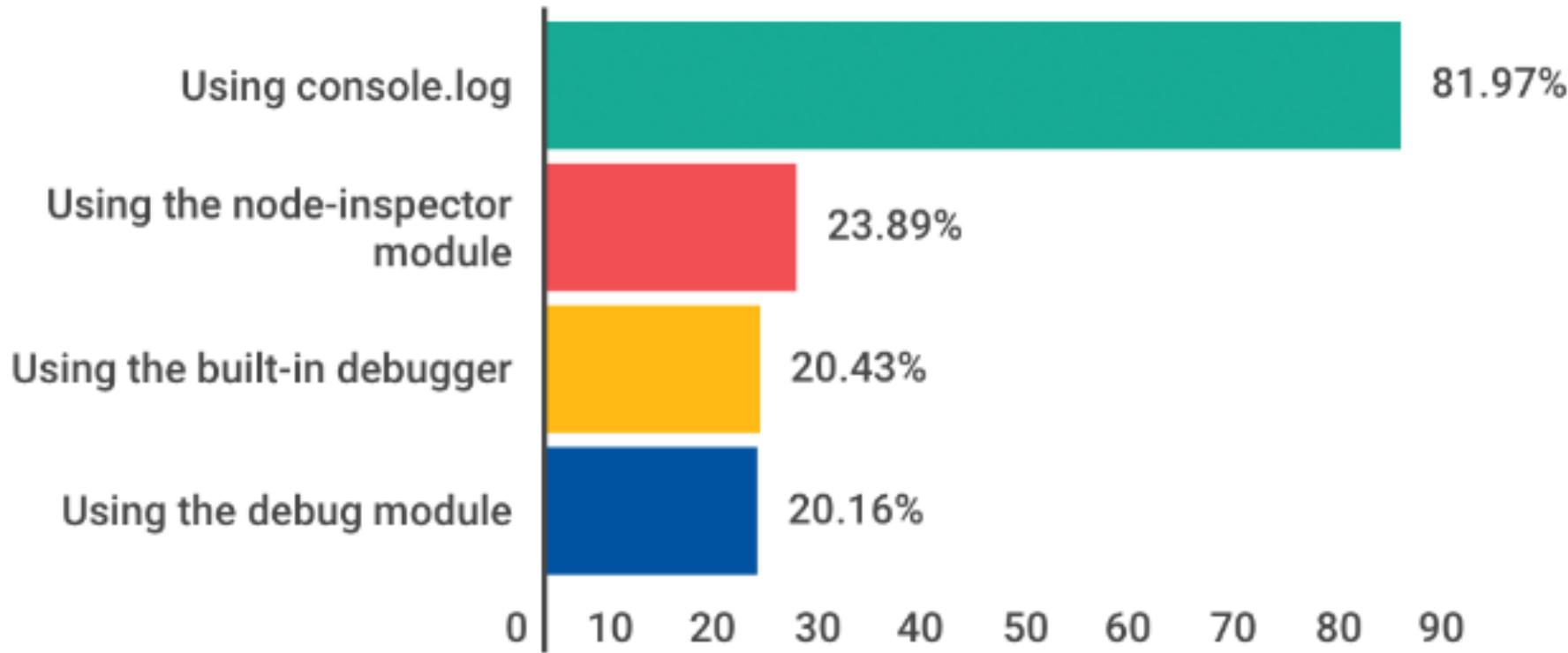
“It is also INCREDIBLY irritating and challenging to debug in a practically infinite number of insanity-inducing ways”

“You cannot ssh into your Lambda.”

“You cannot inspect your program as it’s running. Forget debugging techniques that we’ve accumulated over the years, they don’t apply in this post-Kansas serverless world. Forget dtrace, forget perf, forget [flamegraphs](#). Your debugging arsenal is limited to good old printf debugging. No, scratch that, I should’ve said “good old CloudWatch” debugging, which is a few notches below printf-debugging, because it lags. Hooray!”

How do you debug your applications?

1126 respondents - multiple choice answers



Serverless === Stateless

No in-proc caching or state

Limited Environments

- No local env
- One account for QA, UAT, PROD
- Integration testing difficulties
- Self DDOS 😊

Thank you!

Useful links

AWS Free Tier

<https://aws.amazon.com/ru/free/>

AWS Lambda Developer Guide

<http://docs.aws.amazon.com/lambda/latest/dg/lambda-dg.pdf>

Serverless Architectures

<http://martinfowler.com/articles/serverless.html>

Useful links

What is AWS?

<https://www.youtube.com/watch?v=DERzYnthq1s>

Your First Week on Amazon Web Services

<https://www.youtube.com/watch?v=7CiHBcqw6zc>

<https://github.com/JustServerless/awesome-serverless>

<https://github.com/donnemartin/awesome-aws>

<https://acloud.guru/course/serverlessconf-nyc-2016/dashboard>

Useful links

AWS May 2016 Webinar Series - Deep Dive on Serverless Web Applications

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

AWS April Webinar Series - AWS Lambda: Event-driven Code for Devices and the Cloud

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