

Introduction to Microservices

Peter Dalbhanjan, Solutions Architect, AWS



Pop-up Loft

10 min	Evolution from Monoliths to Microservices
10 min	Core Principle of Microservices Approaches to building microservices on AWS
15 min	Demo: Let's build a microservice!
15 min	Other Architectural Principles
5 min	Additional resources



10 min	Evolution from Monoliths to Microservices
10 min	Core Principle of Microservices Approaches to building microservices on AWS
15 min	Demo: Let's build a microservice!
15 min	Other Architectural Principles
5 min	Additional resources



10 min	Evolution from Monoliths to Microservices
10 min	Core Principle of Microservices Approaches to building microservices on AWS
15 min	Demo: Let's build a microservice!
15 min	Other Architectural Principles
5 min	Additional resources



10 min	Evolution from Monoliths to Microservices
10 min	Core Principle of Microservices Approaches to building microservices on AWS
15 min	Demo: Let's build a microservice!
15 min	Other Architectural Principles
5 min	Additional resources



10 min	Evolution from Monoliths to Microservices
10 min	Core Principle of Microservices Approaches to building microservices on AWS
15 min	Demo: Let's build a microservice!
15 min	Other Architectural Principles
5 min	Additional resources



10 min	Evolution from Monoliths to Microservices
10 min	Core Principle of Microservices Approaches to building microservices on AWS
15 min	Demo: Let's build a microservice!
15 min	Other Architectural Principles
5 min	Additional resources



TECHNICAL & BUSINESS SUPPORT

Support

HYBRID ARCHITECTURE

Integrated Networking



Professional Services



Partner Ecosystem



Training & Certification



Solutions Architects



Account Management



Security & Pricing Reports

MARKETPLACE

Business Apps



Business Intelligence



DevOps Tools



Security



Networking



Databases



Storage

ANALYTICS

- Data Warehousing
- Business Intelligence
- Hadoop/Spark
- Streaming Data Analysis
- Streaming Data Collection
- Machine Learning
- Elastic Search

APP SERVICES

- Queuing & Notifications
- Workflow
- Search
- Email
- Transcoding

MOBILE SERVICES

- API Gateway
- Identity
- Sync
- Mobile Analytics
- Single Integrated Console
- Push Notifications

DEVELOPMENT & OPERATIONS

- One-click App Deployment
- DevOps Resource Management
- Application Lifecycle Management
- Containers
- Triggers
- Resource Templates

IoT

- Rules Engine
- Device Shadows
- Device SDKs
- Device Gateway
- Registry

ENTERPRISE APPS

- Virtual Desktop
- Sharing Collab.
- Corp. Email
- Backup

SECURITY & COMPLIANCE

Identity Management



Access Control



Key Management & Storage



Monitoring & Logs



Configuration Compliance



Web application firewall



Assessment and reporting



Resource Usage

CORE SERVICESCompute
VMs, Auto-scaling, &
Load BalancingStorage
Object, Block, Archive,
Import/Export

CDN

Databases
Relational, NoSQL,
Caching, Migration

Network VPC

INFRASTRUCTURE

Regions



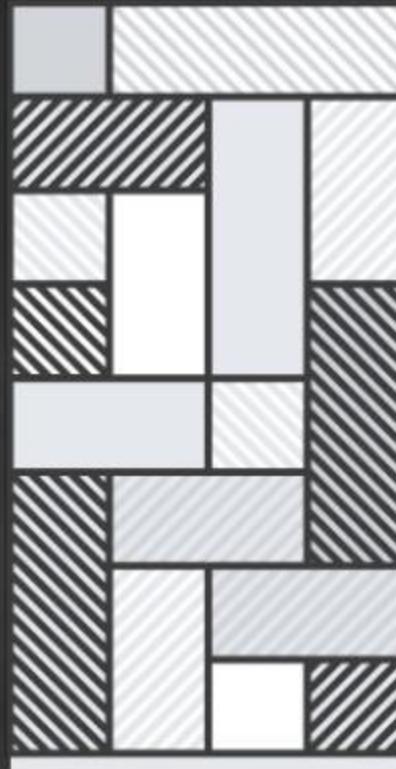
Availability Zones



Point of Presence



“The Monolith”



Challenges with monolithic software

Difficult to scale

Architecture is hard to maintain and evolve

Lack of agility

Long Build/Test/Release Cycles
(who broke the build?)

New releases take months

Lack of innovation

Operations is a nightmare
(module X is failing,
who's the owner?)

Long time to add new features

Frustrated customers

Challenges with monolithic software

Difficult to scale

Architecture is hard to maintain and evolve

Lack of agility

Long Build/Test/Release Cycles
(who broke the build?)

New releases take months

Lack of innovation

Operations is a nightmare
(module X is failing,
who's the owner?)

Long time to add new features

Frustrated customers

Challenges with monolithic software

Difficult to scale

Architecture is hard to maintain and evolve

Lack of agility

Long Build/Test/Release Cycles
(who broke the build?)

New releases take months

Lack of innovation

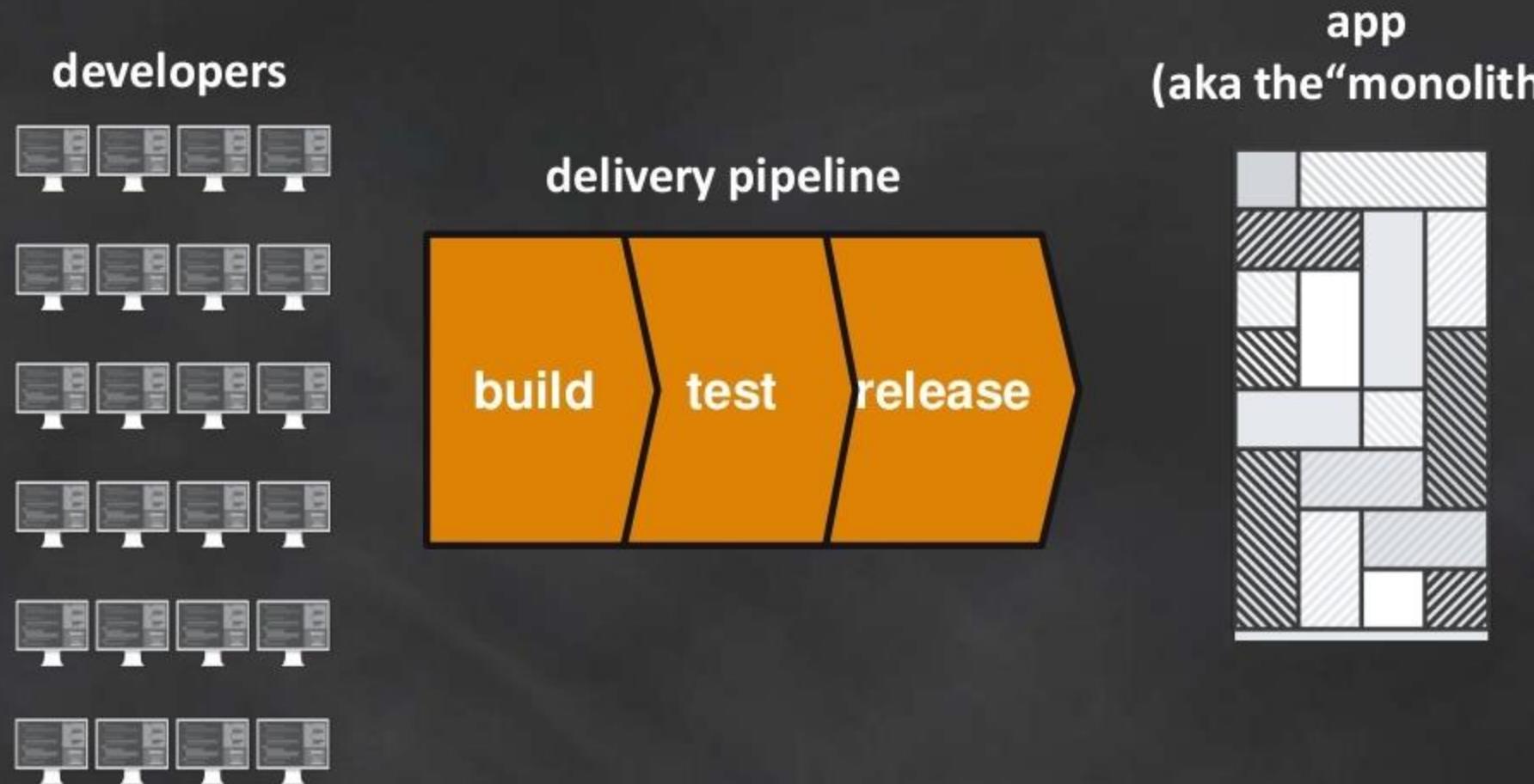
Operations is a nightmare
(module X is failing,
who's the owner?)

Long time to add new features

Frustrated customers



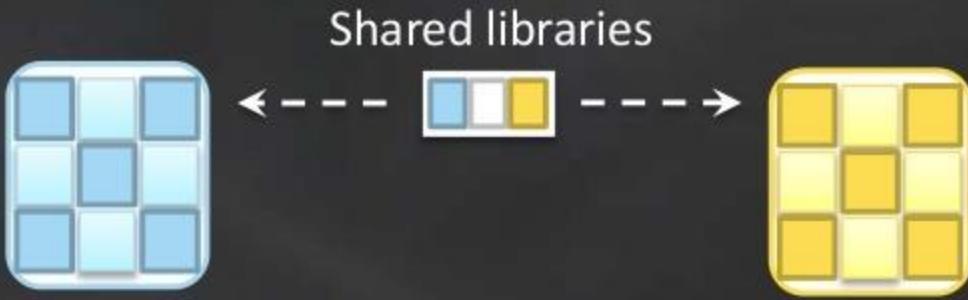
Monolith development lifecycle



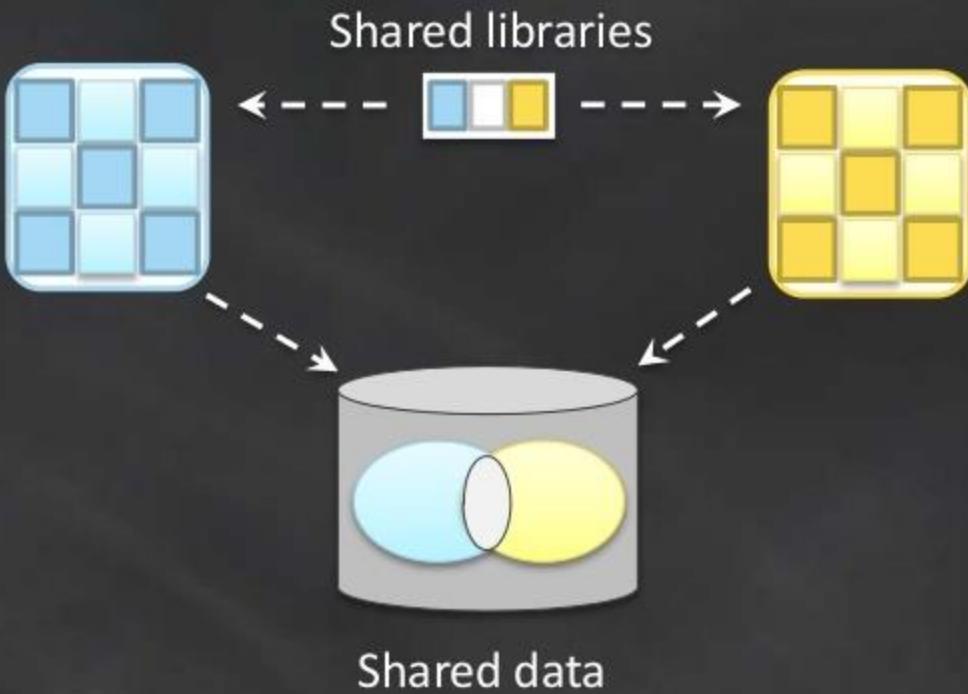
Too much software coupling



Too much software coupling

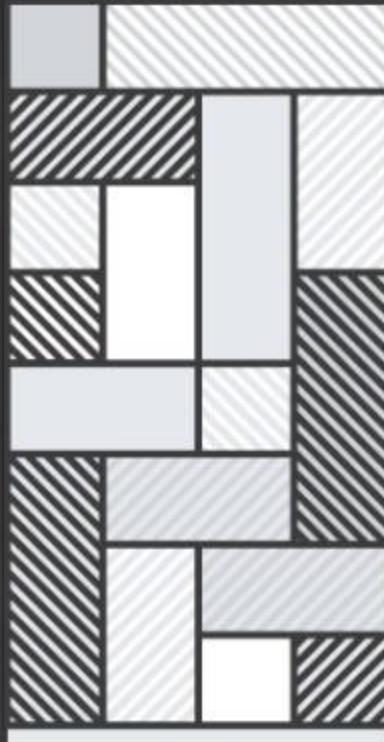


Too much software coupling



CHANGE

For The Better



what are microservices





W Microservices - Wikipedia, ... +

https://en.wikipedia.org/wiki/Microservices dictionary defi →

Non-obvious indic... Microservices We... Employee Q&A | A... AWS Communicati... Terminology | hue...

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search

 WIKIPEDIA The Free Encyclopedia

Microservices

From Wikipedia, the free encyclopedia

Microservices are a more concrete and modern interpretation of service-oriented architectures (SOA) used to build distributed software systems. Like in SOA, services in a microservice architecture are processes that communicate with each other over the network in order to fulfill a goal. Also, like in SOA, these services use technology agnostic protocols.^[1] Microservices architectural style is a first realisation of SOA that has happened after the introduction of DevOps and this is becoming the standard for building continuously deployed systems.^[2]

In contrast to SOA, microservices gives an answer to the question of how big a service should be and how they should communicate with each other. In a microservices architecture, services should be small and the protocols should be lightweight. The benefit of distributing different responsibilities of the system into different smaller services is that it enhances the cohesion and decreases the coupling. This makes it much easier to change and add functions and qualities to the system anytime.^[3] It also allows the architecture of an individual service to emerge through continuous refactoring,^[4] hence reduces the need for a big up front design and allows for releasing the software early

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

**"service-oriented
architecture
composed of
loosely coupled
elements
that have
bounded contexts"**

*Adrian Cockcroft (former Cloud Architect at Netflix,
now Technology Fellow at Battery Ventures)*

**"service-oriented
architecture**

composed of
**loosely coupled
elements**
that have
bounded contexts"



Services communicate with each other over the network

*Adrian Cockcroft (former Cloud Architect at Netflix,
now Technology Fellow at Battery Ventures)*

**"service-oriented
architecture**

composed of

**loosely coupled
elements**

that have

bounded contexts"

You can update the service independently; updating one service doesn't require changing any other service

*Adrian Cockcroft (former Cloud Architect at Netflix,
now Technology Fellow at Battery Ventures)*

**"service-oriented
architecture
composed of
loosely coupled
elements
that have
bounded contexts"**

*Adrian Cockcroft (former Cloud Architect at Netflix,
now Technology Fellow at Battery Ventures)*

Self-contained; you can update the code without knowing anything about the internals of other microservices

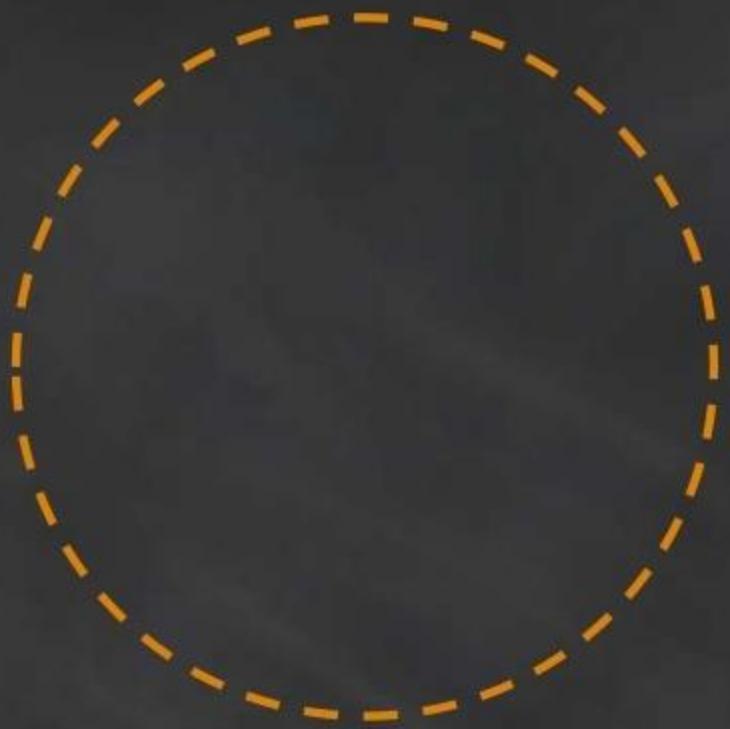
“Do one thing, and do it well”



“Do one thing, and do it well”



Anatomy of a Micro-service



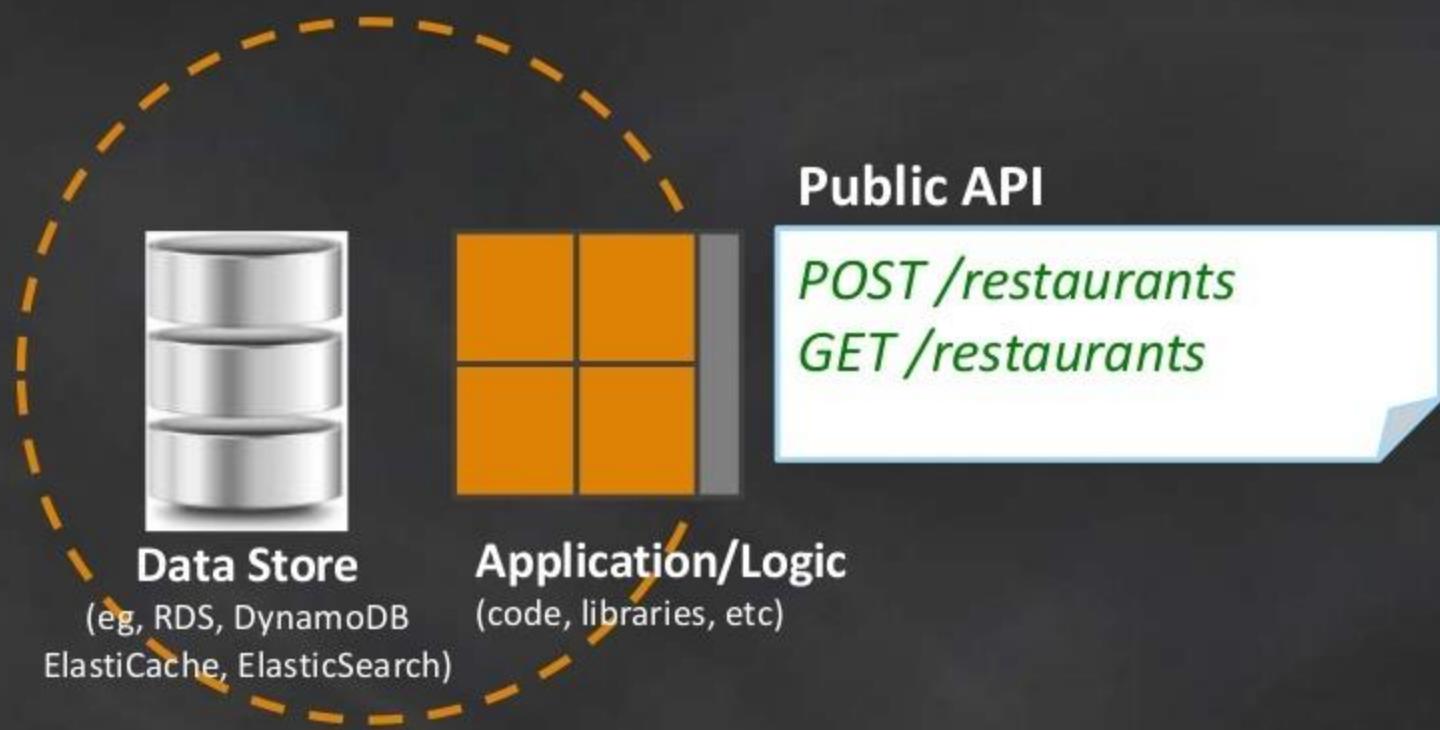
Anatomy of a Micro-service



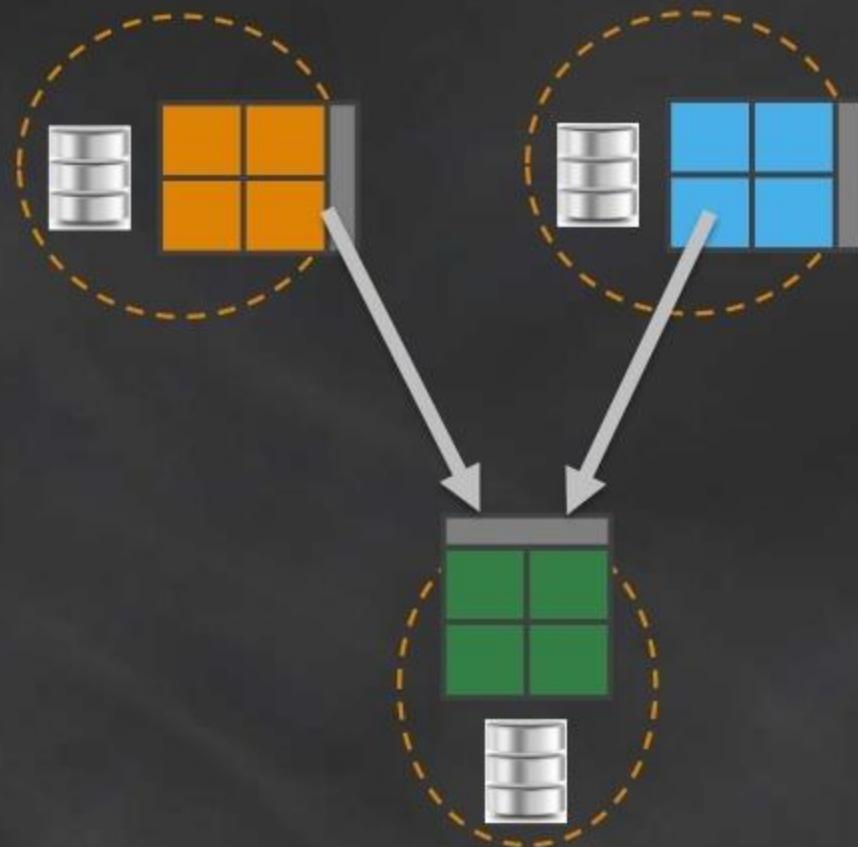
Anatomy of a Micro-service



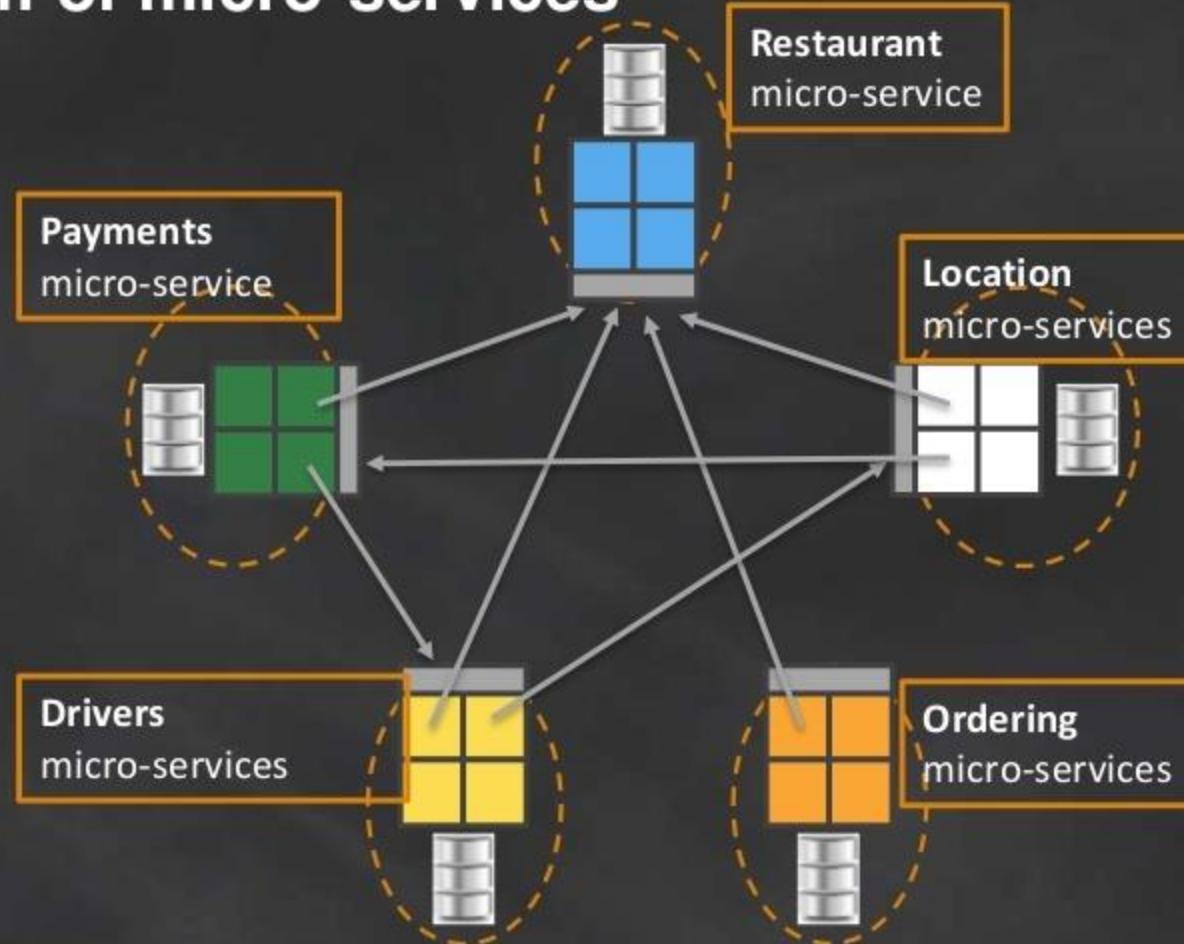
Anatomy of a Micro-service

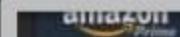


Avoid Software Coupling

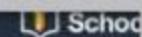


Ecosystem of micro-services





All - don't mess with Texas mug



Departments

Browsing History Jim's Amazon.com

Today's Deals Gift Cards & Registry

Sell Help

Hello, Jim

Your Account

Prime

Lists

Kitchen & Dining

Best Sellers

Wedding Registry

Small Appliances

Kitchen Tools

Cookware

Barware

Cutlery

Dining & Entertaining

Storage & Organization

PrimeTV™



\$10 OFF YOUR FIRST ORDER

Don't Mess With Texas Coffee Mug

Author Unknown

4.5 stars • 11 customer reviews

Price: \$7.99 & Prime

In Stock.

Want it tomorrow, June 24? Order within 14 hrs 24 mins and choose One-Day Shipping at checkout. Details

Sold by Got Attitude and Fulfilled by Amazon. Gift-wrap available.

- 11 OZ Ceramic Mug, Both sides printing
- Microwave and Dishwasher Safe
- Printed on both sides with premium printing

7 new from \$7.99



Click to open expanded view



Save \$10 on George Foreman

when you spend \$60

\$89.99

Find wrong product information here? Help us fix it.

Frequently bought together



Total price: \$23.91

Add all three to Cart

Add all three to List

Share



Qty: 1

Add to Cart

Turn on 1-Click ordering for this browser

Ship to:

Jim Tran-BELLEVUE

Add to List

Add to Wedding Registry

Other Sellers on Amazon

\$12.50

+ Free Shipping

Sold by: iHouse

Add to Cart

\$12.50

+ Free Shipping

Sold by: Sun Coast Max

Add to Cart

\$9.99

+ \$5.49 shipping

Sold by: Asoka Wu Shop

Add to Cart

7 new from \$7.99

Have one to sell?

Sell on Amazon

SIS

Sponsored by Lysol



Cart Subtotal: \$4.99

Want change in Cart

Proceed to checkout

This order contains a gift



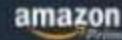
\$4.99

iPhone

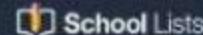


BEAUTIFUL THINGS ON AMAZON UPDATED DAILY

EXPLORE



A+ - don't mess with texas mug



Sponsored by Lysol

Departments

Browsing History

Jim's Amazon.com

Today's Deals

Gift Cards & Registry

Sell Help

Hello, Jim

Your Account

Prime

Lists



Cart Subtotal: \$4.99

1 recent change in Cart

Proceed to checkout

This order contains a gift

Kitchen & Dining
Best Sellers
Wedding Registry
Small Appliances
Kitchen Tools
Cookware
Barware
Cutlery
Dining & Entertaining
Storage & Organization

PrimeNow



1 Back to search results for "don't mess with texas mug"



Click to open expanded view

Don't Mess With Texas Coffee Mug

by Aquiles Creations

4.5 stars • 11 customer reviews

Price: \$7.99

In Stock.

Want it tomorrow, June 24? Order within 14 hrs 24 mins and choose One-Day Shipping at checkout. Details

Sold by Got Attitude and Fulfilled by Amazon. Gift-wrap available.

- 11 OZ Ceramic Mug, Both sides printing
- Microwave and Dishwasher Safe
- Printed on both sides with premium printing

7 new from \$7.99



Save \$10 on George Foreman

when you spend \$60

\$59.99

Find wrong product information here? Help us fix it.

Frequently Bought Together



Total price: \$23.91

Add all three to Cart

Add all three to List

Share



Qty: 1

Add to Cart

Turn on 1-Click ordering for this item

Ship to:

Jim Tran-BELLEVUE

Add to List

Add to Wedding Registry



\$4.99

Prime

Other Sellers on Amazon

\$12.50

+ Free Shipping

Sold by: iHouse

Add to Cart

\$12.50

+ Free Shipping

Sold by: Sun Coast Max

Add to Cart

\$9.99

+ \$5.49 shipping

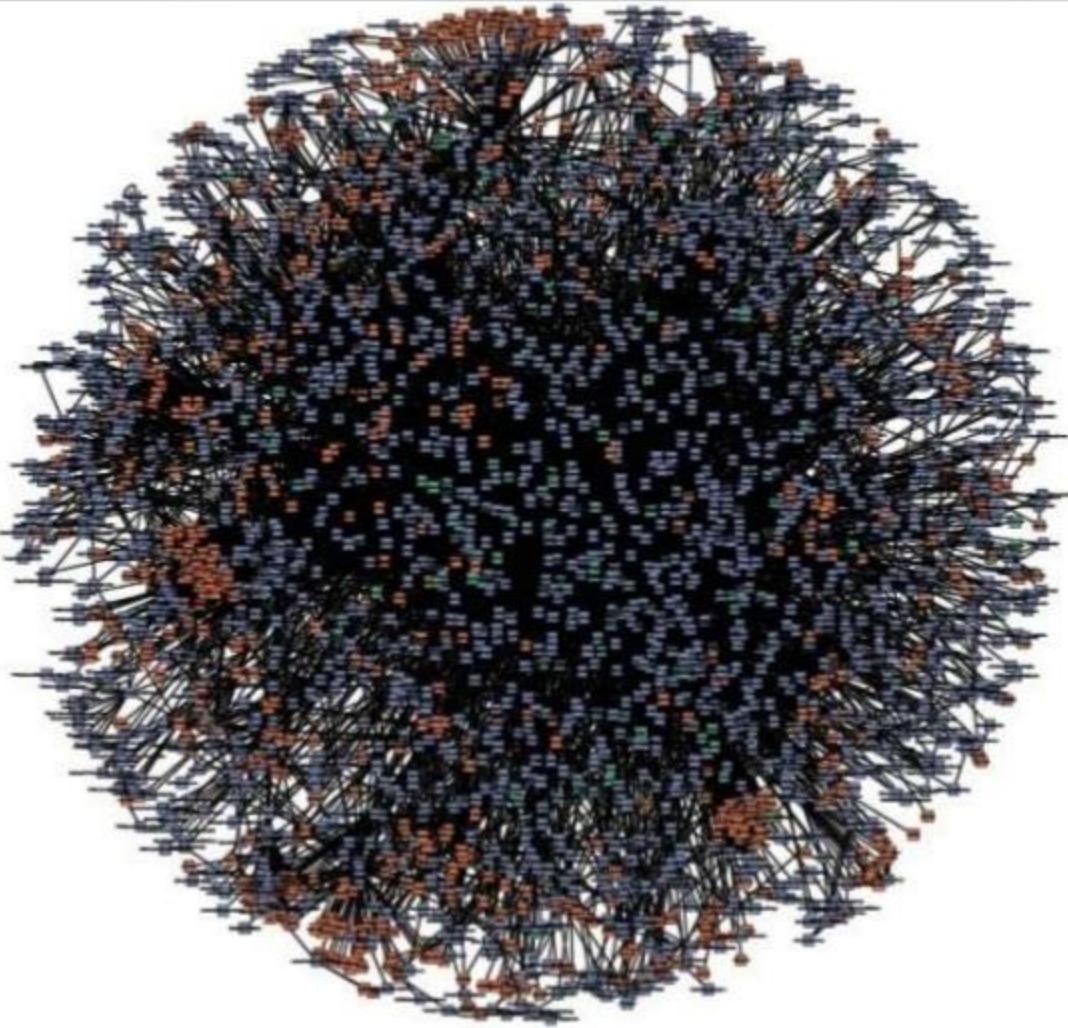
Sold by: Asoka Wu Shop

Add to Cart

7 new from \$7.99

Have one to sell?

Sell on Amazon



Thousands of teams

- × Microservice architecture
 - × Continuous delivery
 - × Multiple environments
-

= 50 million deployments a year

(5708 per hour, or every 0.63 second)

Gilt: Luxury designer brands at members-only prices

The screenshot shows the Gilt website homepage. At the top, there's a navigation bar with links for FEATURED, WOMEN, MEN, KIDS, HOME, CITY, and TRAVEL. Below the navigation is a banner with the text "Big news on returns: Return almost all men's & women's items (with better refunds). Learn More". The main visual features a large image of a teal Prada handbag and a matching clutch bag. To the right of this image, there's a section titled "Today on Women's Prada Accessories" with the subtext "Always impress with the ultimate in Italian luxury". A "Shop this Sale" button is present. Further down the page, there's a "TOP PICKS" section with a smaller image of two pairs of sandals. To the right of this, there's a "The FURNITURE Shop" section with the subtext "Design staples that'll anchor your space". At the bottom of the page, a call-to-action reads "Join Gilt to find the best brands at up to 70% off".

Gilt Top Brands for Women... [+/-](#)

www.gilt.com

gilt daily deals

Sign In | Register

Cart

GILT

FEATURED WOMEN MEN KIDS HOME CITY TRAVEL

Big news on returns: Return almost all men's & women's items (with better refunds). [Learn More](#)

Today on Women's

Prada Accessories

Always impress with the ultimate in Italian luxury

[Shop this Sale](#)

Tuesday, June 28th 2011

TOP PICKS

The FURNITURE Shop

Design staples that'll anchor your space

Join Gilt to find the best brands at up to 70% off

... Sale every day at noon EST

Gilt Top Brands for Women... > www.gilt.com gilt daily deals Sign In | Register Cart

GILT FEATURED WOMEN MEN KIDS HOME CITY TRAVEL Search

Big news on returns: Return almost all men's & women's items (with better refunds). Learn More



Today on Women's
Prada Accessories
Always impress with the ultimate in Italian luxury.

[Shop this Sale](#)

Tuesday, June 28th 2011

TOP PICKS

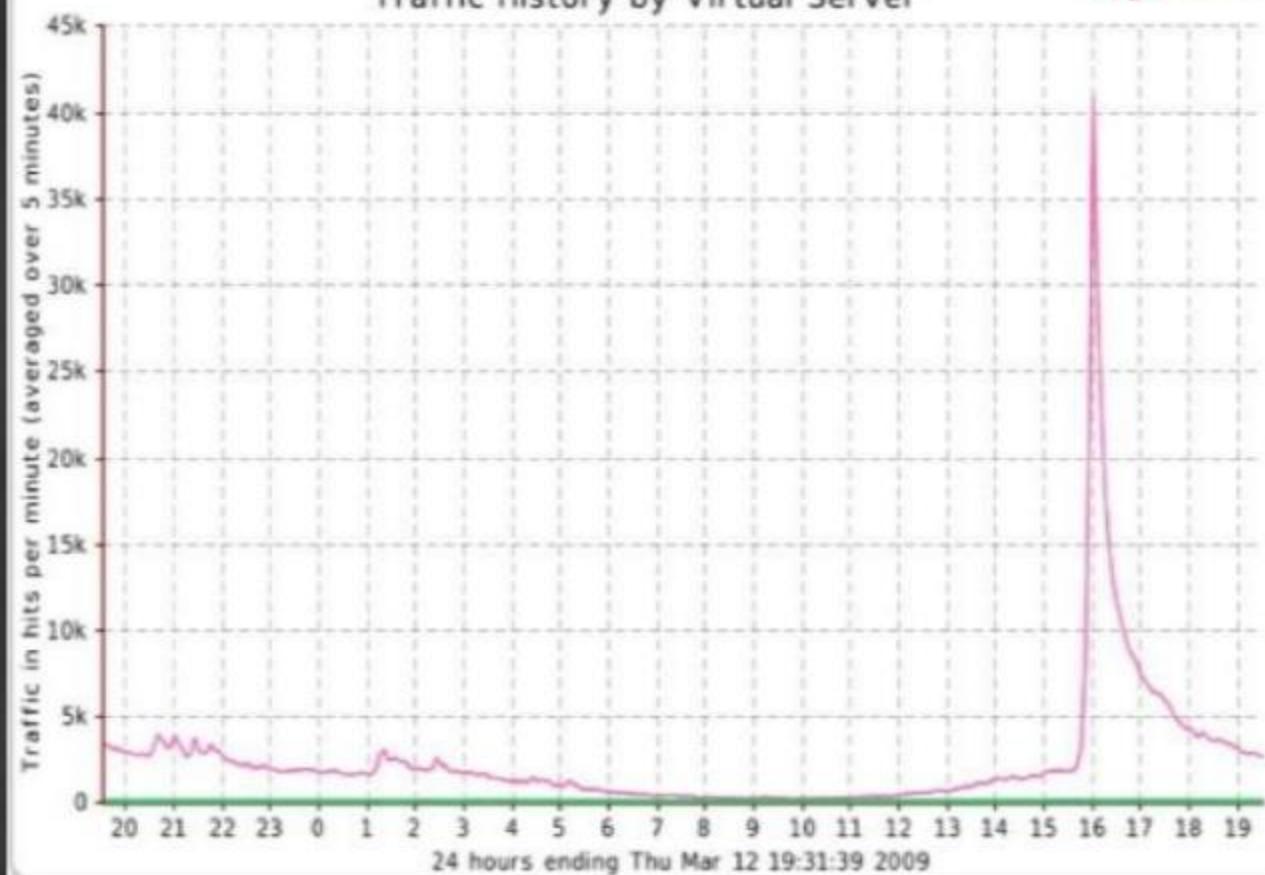


The
FURNITURE
Shop
Design staples that'll anchor your space

Join Gilt to find the best brands at up to 70% off



Traffic History by Virtual Server





AWS
re:Invent

CONSUMER
DIGITAL TECH

ARC 306

Nike's Journey to Microservices

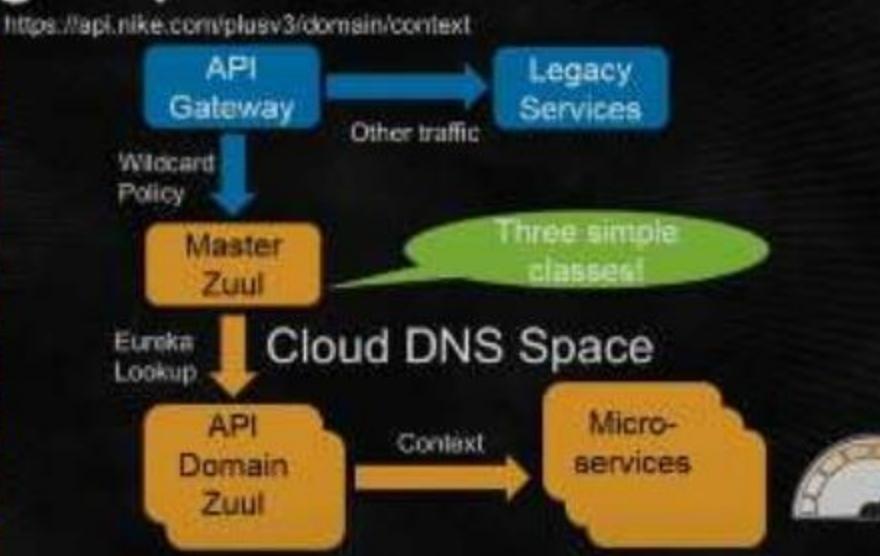
Jason Robey, Consumer Digital Technology (CDT), Nike, Inc.

November 12, 2014 | Las Vegas, NV



Getting to production fast

- AMI
- AWS Console
- Conformity Monkey
- Asgard
- Ribbon
- Eureka
- Archaius
- Servo
- ASG



Principle 1

Micro-services only rely
on each other's public API



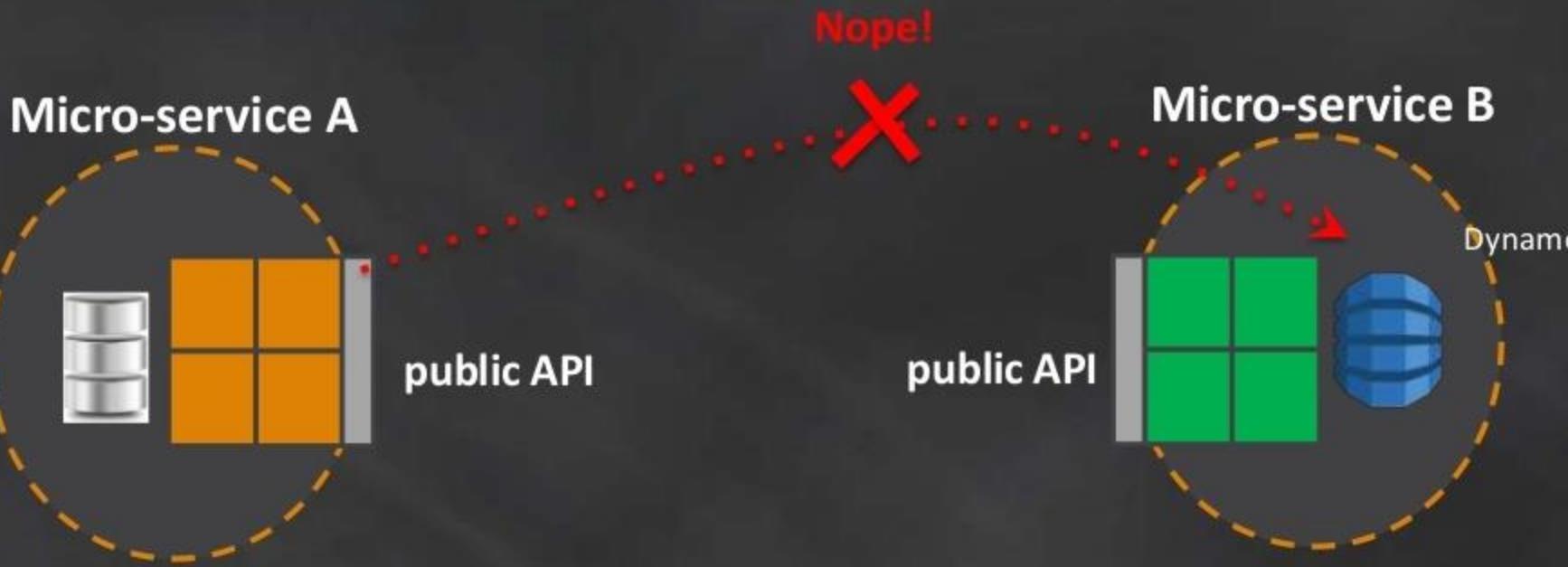
Principle 1: Microservices only rely on each other's public API



Principle 1: Microservices only rely on each other's public API (Hide Your Data)



Principle 1: Microservices only rely on each other's public API (Hide Your Data)



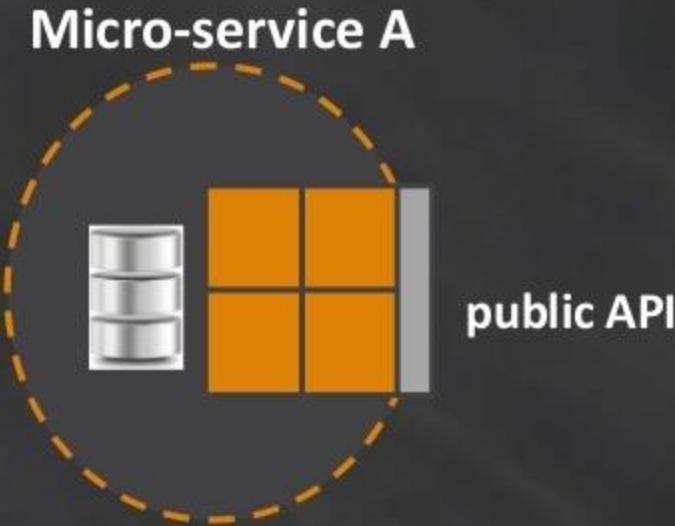
Principle 1: Microservices only rely on each other's public API (Hide Your Data)



Principle 1: Microservices only rely on each other's public API (Evolve API in backward-compatible way...and document!)



Principle 1: Microservices only rely on each other's public API (Evolve API in backward-compatible way...and document!)



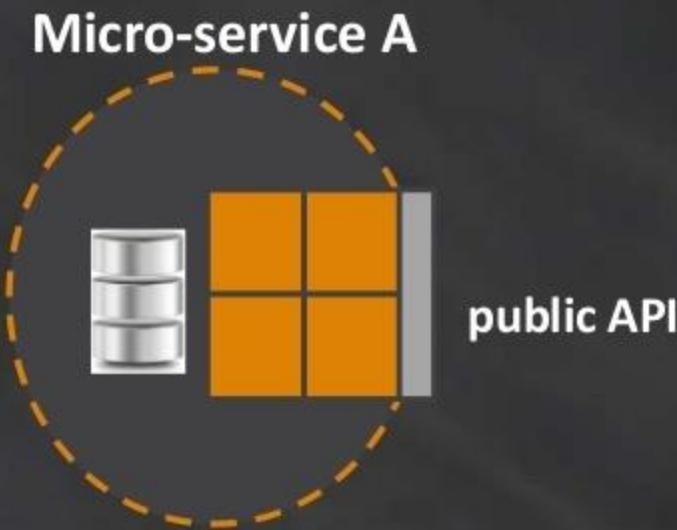
Version 1.0.0

storeRestaurant (id, name, cuisine)

Version 1.1.0

storeRestaurant (id, name, cuisine)
*storeRestaurant (id, name,
arbitrary_metadata)*
addReview (restaurantId, rating, comments)

Principle 1: Microservices only rely on each other's public API (Evolve API in backward-compatible way...and document!)



Version 1.0.0

storeRestaurant (id, name, cuisine)

Version 1.1.0

storeRestaurant (id, name, cuisine)
*storeRestaurant (id, name,
arbitrary_metadata)*
addReview (restaurantId, rating, comments)

Version 2.0.0

*storeRestaurant (id, name,
arbitrary_metadata)*
addReview (restaurantId, rating, comments)

A photograph of a wooden pegboard in a workshop. The board is covered with various hand tools. On the top left, there is a set of wrenches of different sizes. To the right of the wrenches, there are several hammers of varying head sizes and handle materials (wood and metal). Below the hammers, there are two pairs of pliers. The tools are arranged in a somewhat organized manner, though some overlap. The background is a plain, light-colored wall.

Principle 2

Use the right tool for the job

Principle 2: Use the right tool for the job (Embrace polyglot persistence)



Principle 2: Use the right tool for the job (Embrace polyglot persistence)



Principle 2: Use the right tool for the job (Embrace polyglot persistence)



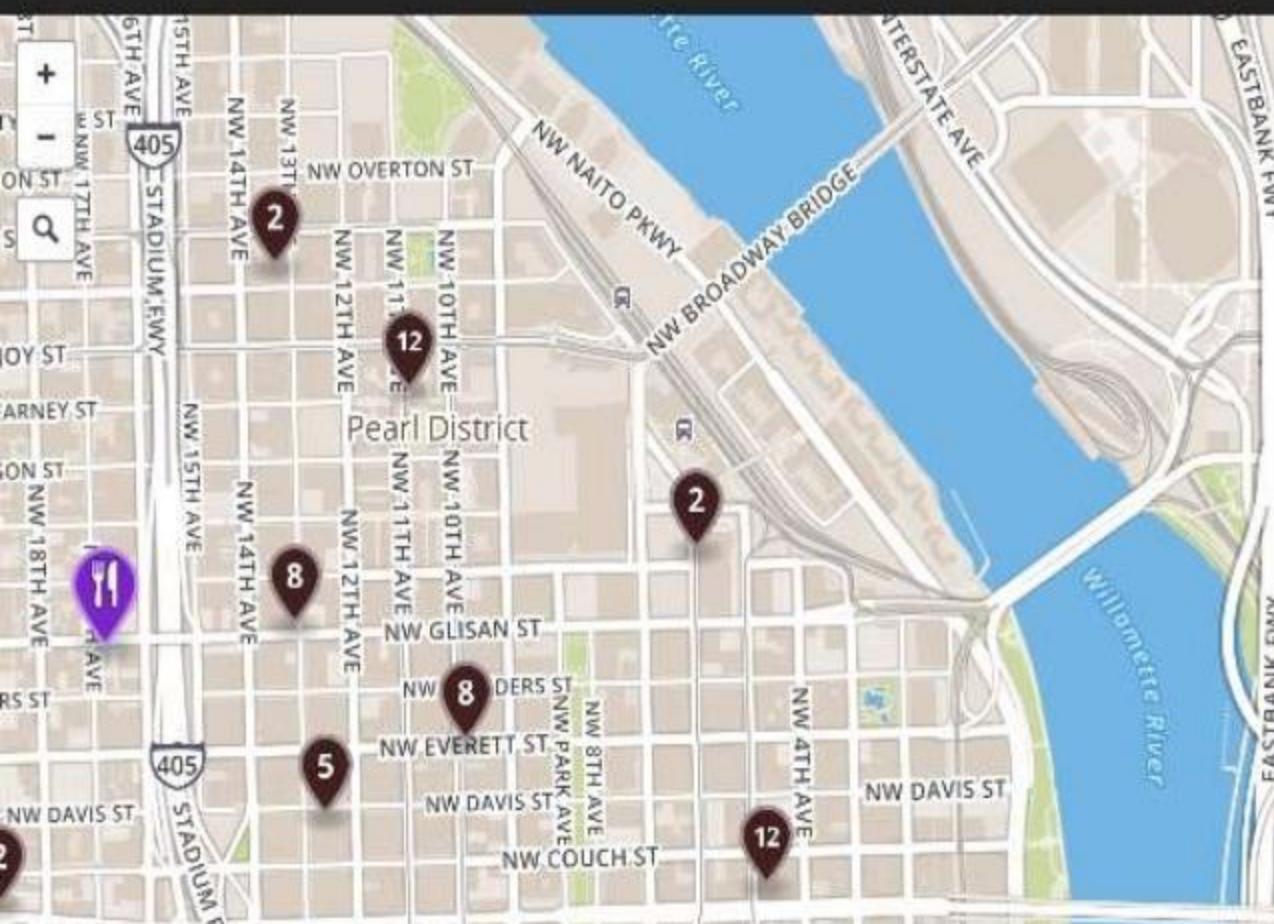
Principle 2: Use the right tool for the job (Embrace polyglot programming frameworks)



Principle 2: Use the right tool for the job (Embrace polyglot programming frameworks)



*Let's build
a microservice!*

**RESTAURANT NAME****LATITUDE****LONGITUDE****CUISINE**

Asian

PRICE

Affordable (\$)

Add restaurant

Restaurant Micro-service

POST /restaurants

GET /restaurants

Restaurant Micro-service

Approach #1

EC2

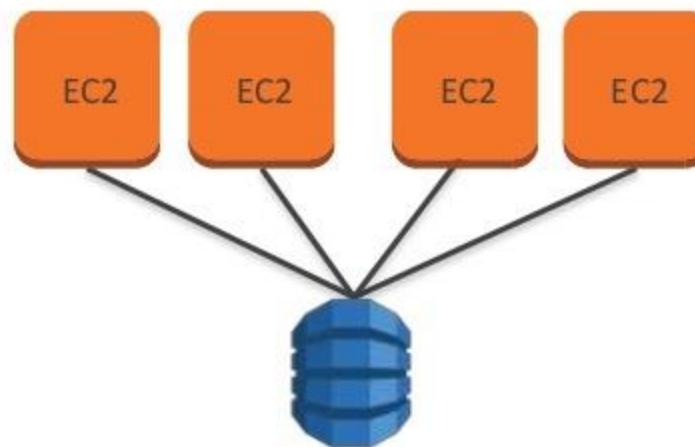
Restaurant Micro-service

EC2

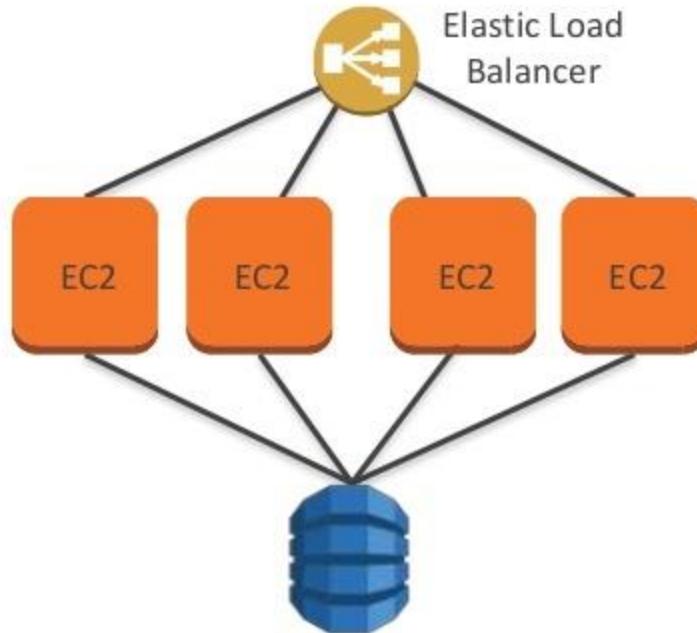
Restaurant Micro-service



Restaurant Micro-service



Restaurant Micro-service



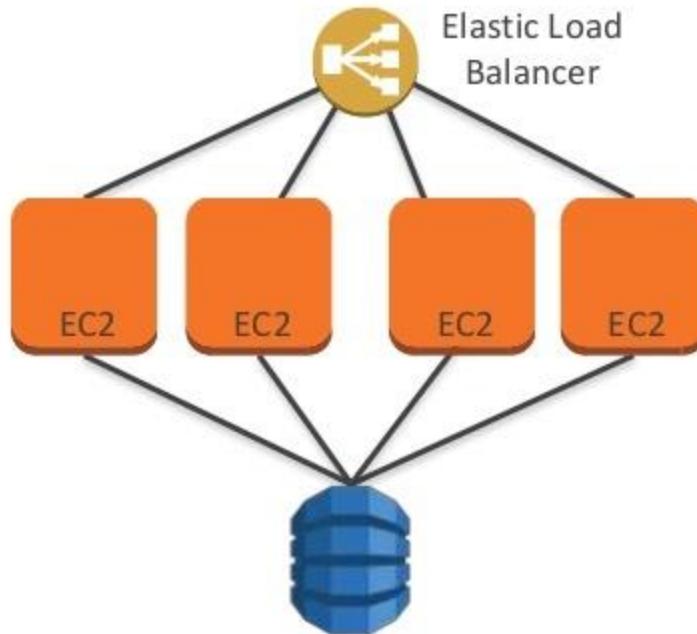
Restaurant Micro-service

Approach #2

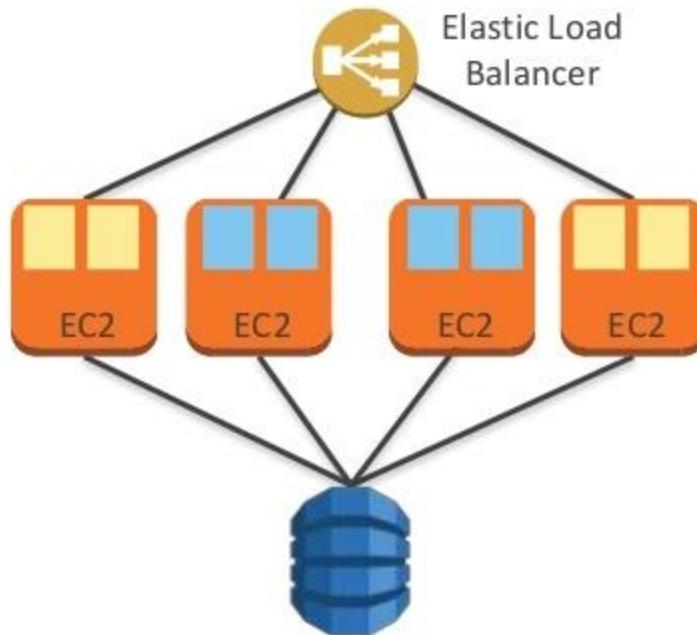
Containers

Using ECS

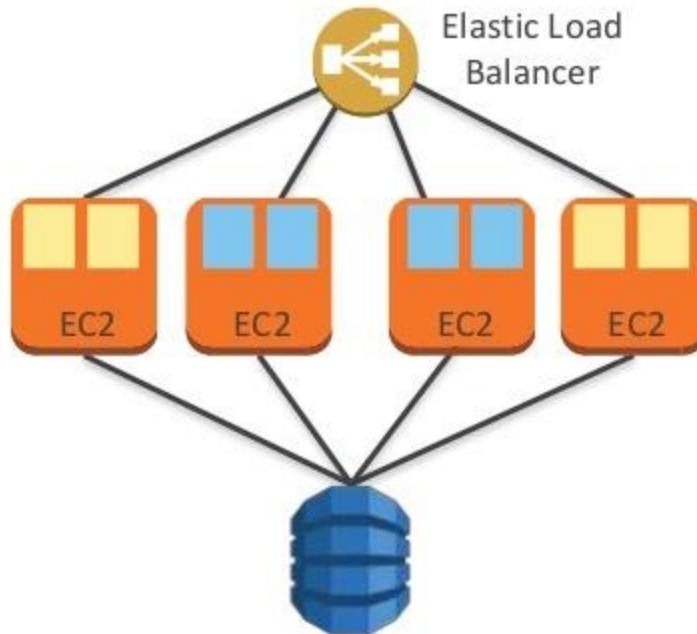
Restaurant Micro-service



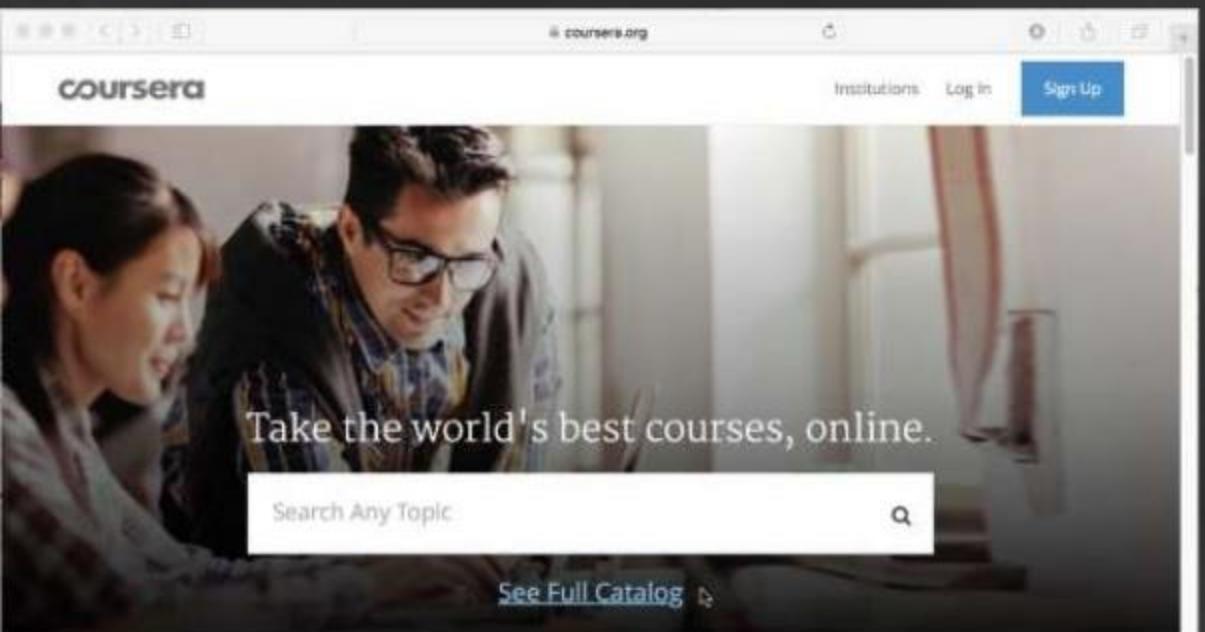
Restaurant Micro-service



Restaurant Micro-service



Amazon
EC2 Container
Service (ECS)
to manage
containers



The screenshot shows the Coursera homepage. At the top, there's a navigation bar with icons for search, refresh, and user profile, followed by the URL "coursera.org". To the right are links for "Institutions", "Log In", and a blue "Sign Up" button. Below the navigation is a large banner image of two people looking at a laptop screen. Overlaid on the banner is the text "Take the world's best courses, online." A search bar with the placeholder "Search Any Topic" and a magnifying glass icon is positioned below the banner. Underneath the search bar is a blue button labeled "See Full Catalog". At the bottom of the page, there are logos for several partner institutions: University of Pennsylvania (Penn), Johns Hopkins University, University of Michigan, Stanford University, and UC San Diego.

Coursera

13 million users from 190 countries
1,000 courses from 119 institutions

- Prototype in less than 2 months
- Deployment time: hours → minutes
- Each team can now develop its respective applications independently

Restaurant Micro-service

Approach #3

API Gateway

+ Lambda

Restaurant Micro-service

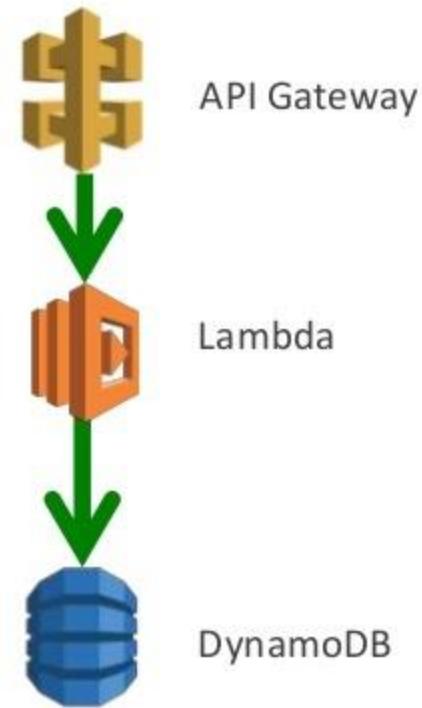


DynamoDB

Restaurant Micro-service



Restaurant Micro-service

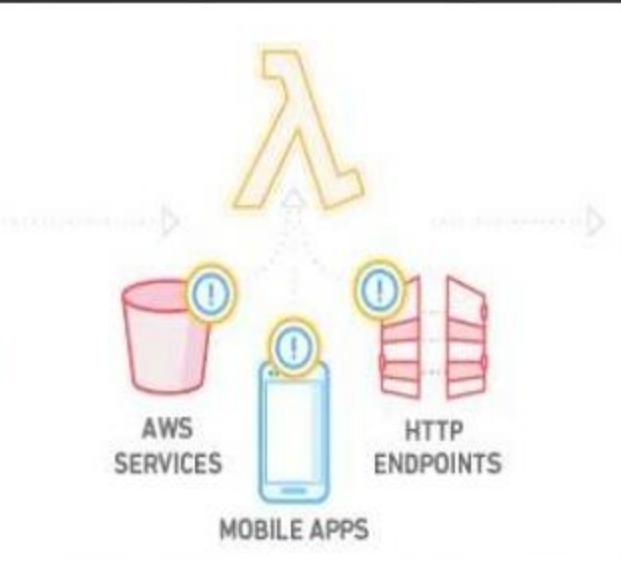




AWS Lambda
lets you run code
without managing servers

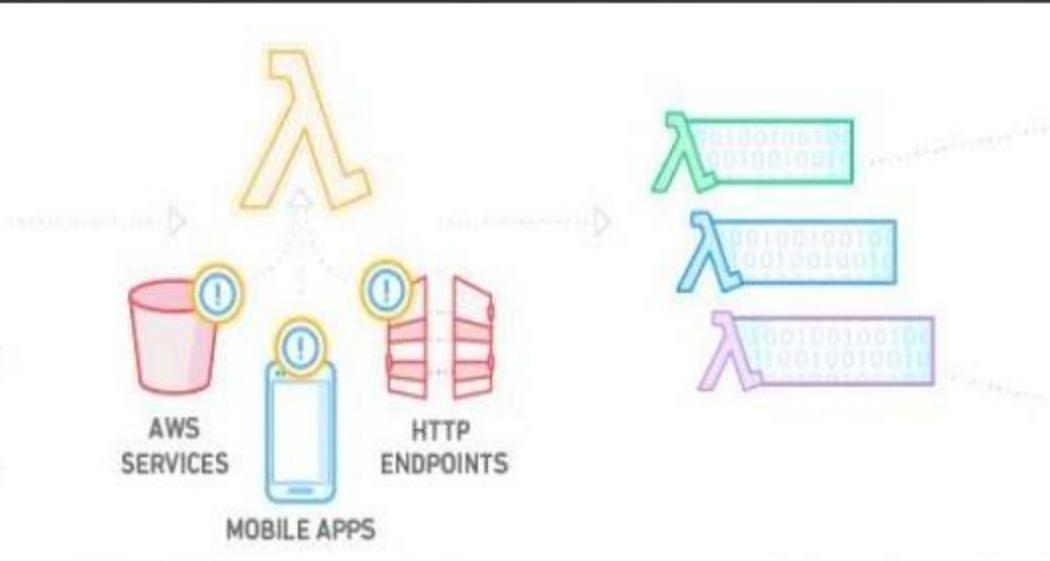


Upload your code
(Java, JavaScript,
Python)



Upload your code
(Java, JavaScript,
Python)

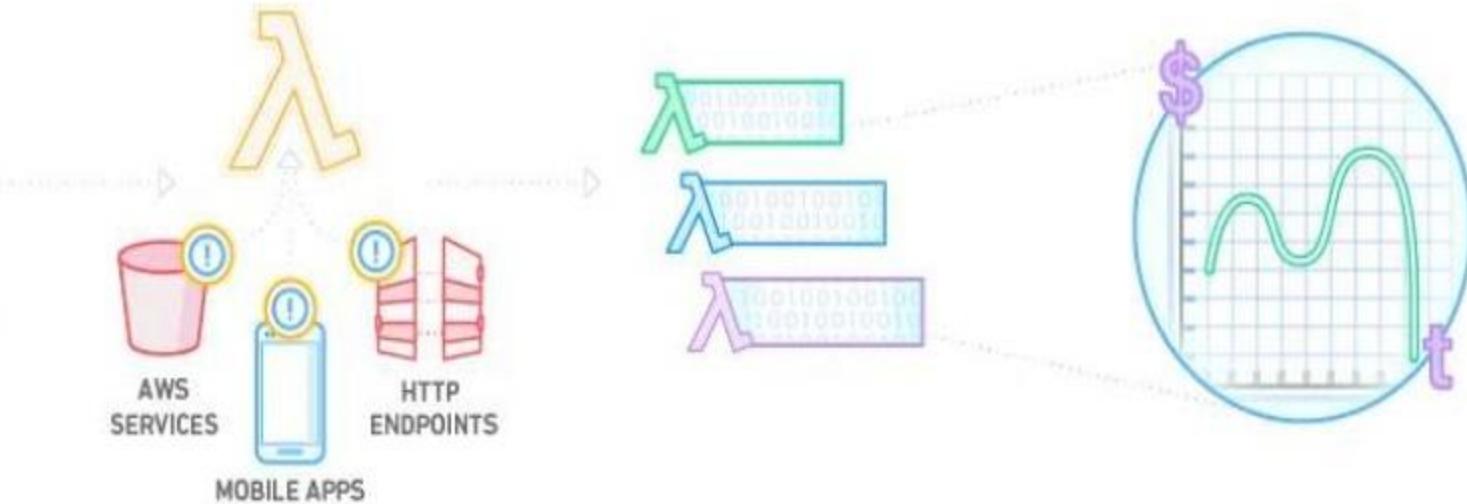
Set up your code to
trigger from other AWS
services, webservice
calls, or app activity



Upload your code
(Java, JavaScript,
Python)

Set up your code to
trigger from other AWS
services, webservice
calls, or app activity

Lambda
automatically
scales



Upload your code
(Java, JavaScript,
Python)

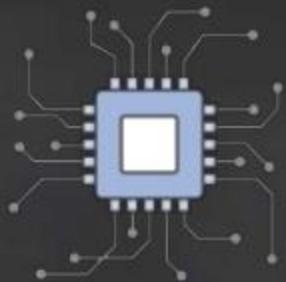
Set up your code to
trigger from other AWS
services, webservice
calls, or app activity

Lambda
automatically
scales

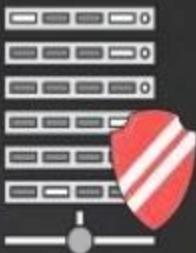
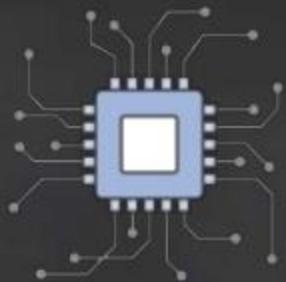
Pay for only the
compute time you
use (sub-second
metering)



AWS API Gateway
is the easiest way to
deploy micro-services

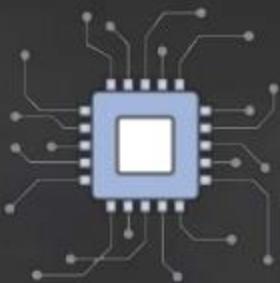


Create a unified
API frontend for
multiple
micro-services

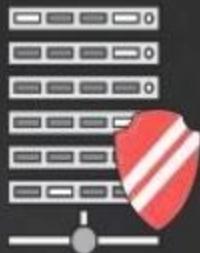


Create a unified
API frontend for
multiple
micro-services

Authenticate and
authorize
requests



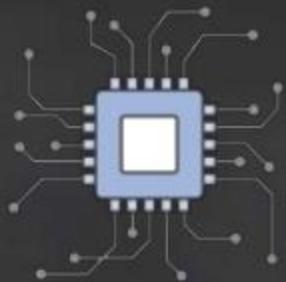
Create a unified
API frontend for
multiple
micro-services



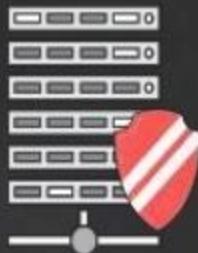
Authenticate and
authorize
requests



Handles DDoS
protection and
API throttling



Create a unified API frontend for multiple micro-services



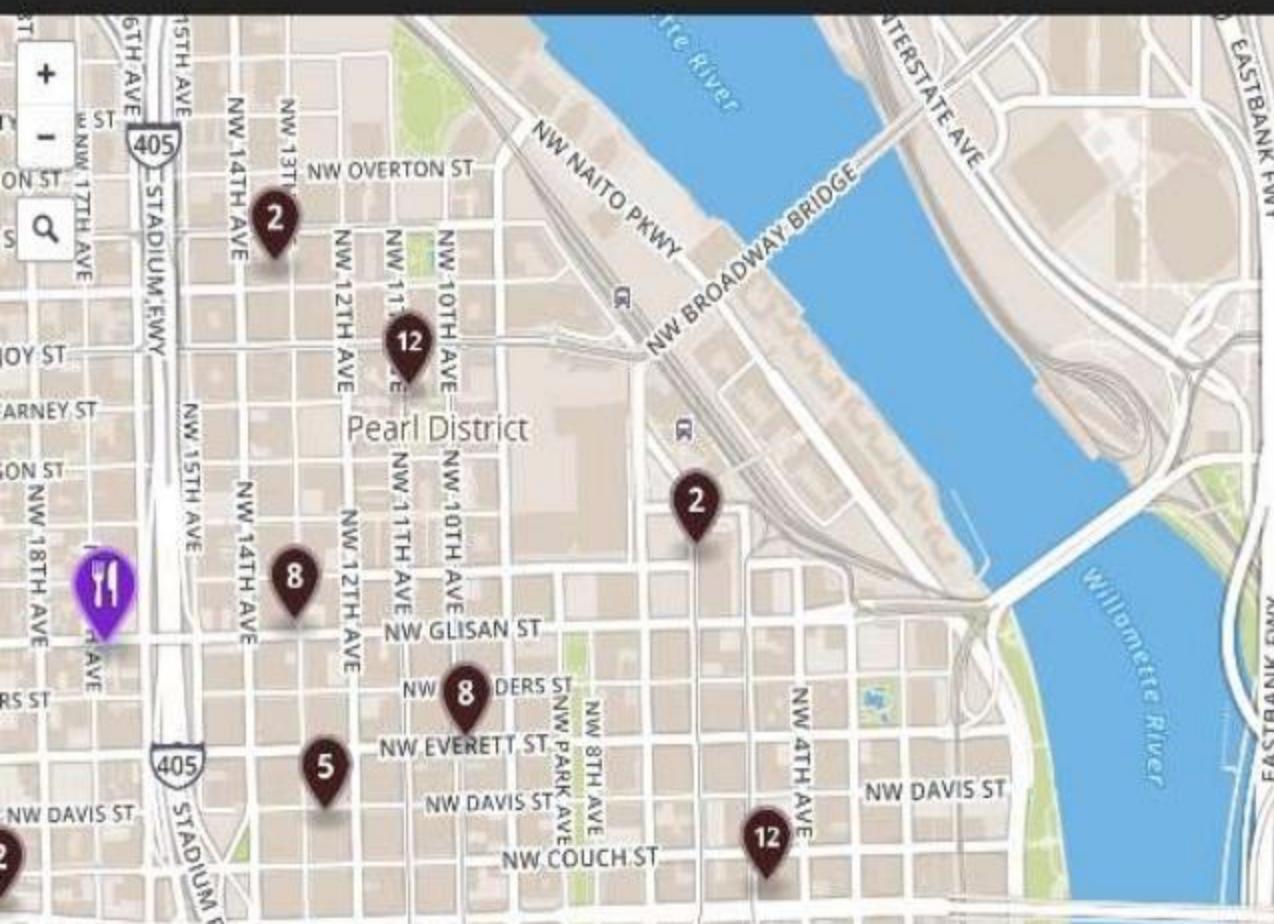
Authenticate and authorize requests



Handles DDoS protection and API throttling



...as well as monitoring, logging, rollbacks, client SDK generation...

**RESTAURANT NAME****LATITUDE****LONGITUDE****CUISINE**

Asian

PRICE

Affordable (\$)

Add restaurant

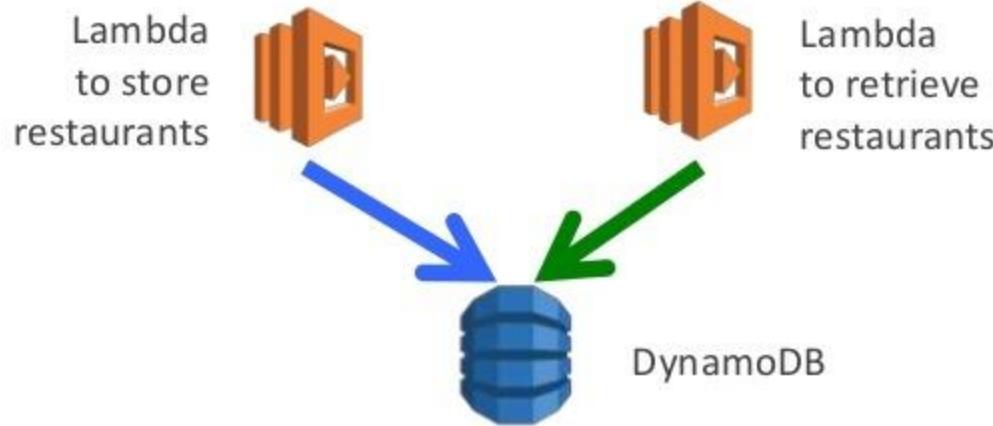
Restaurant Microservice

Restaurant Microservice

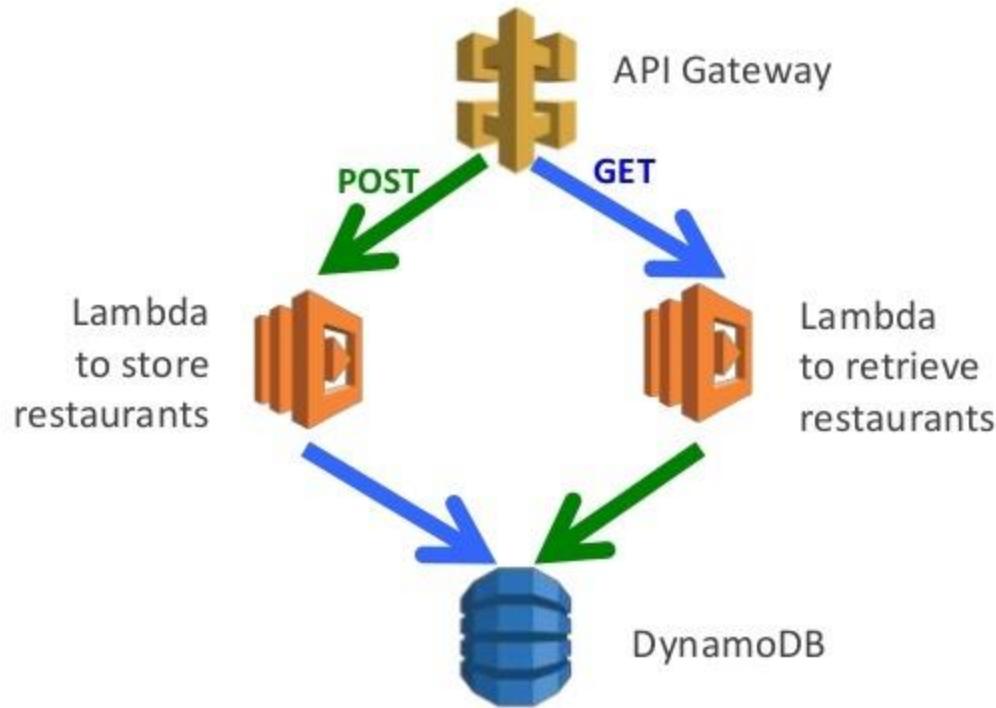


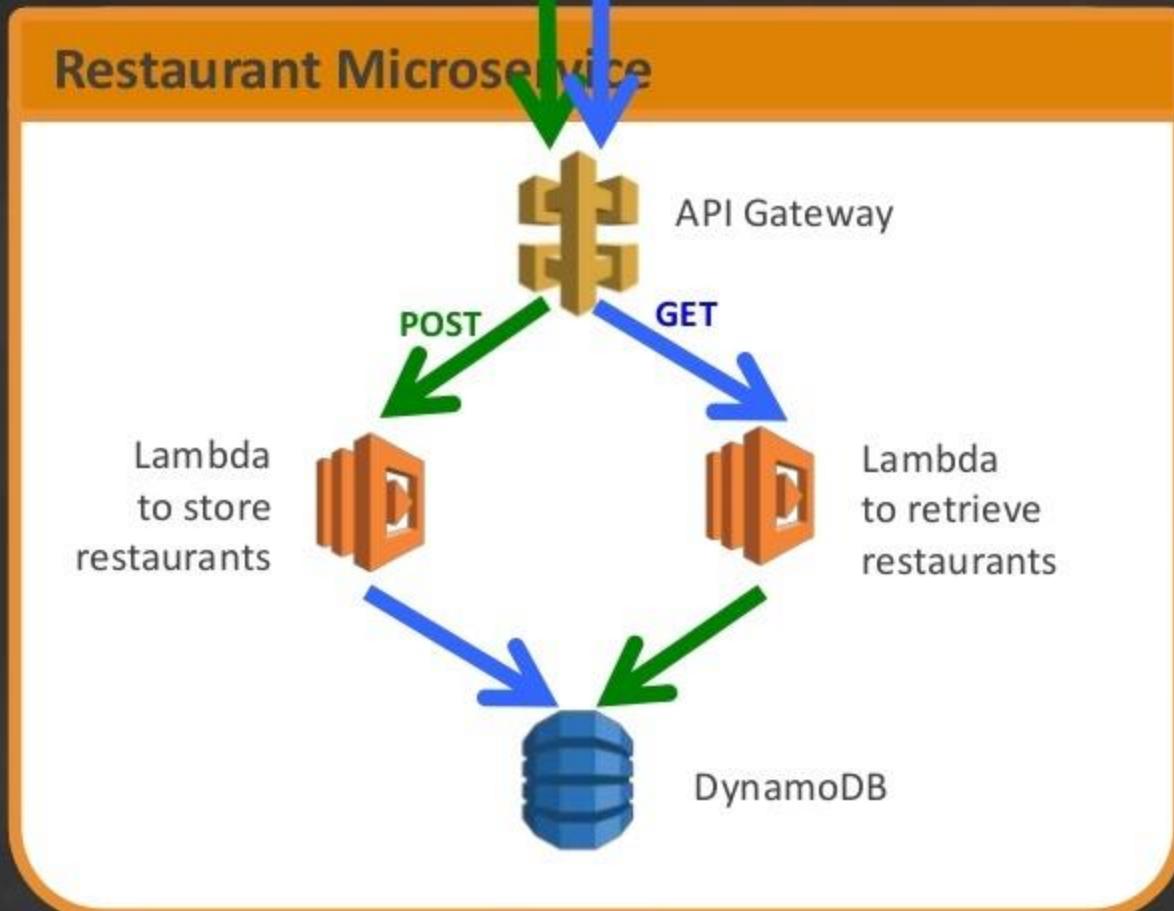
DynamoDB

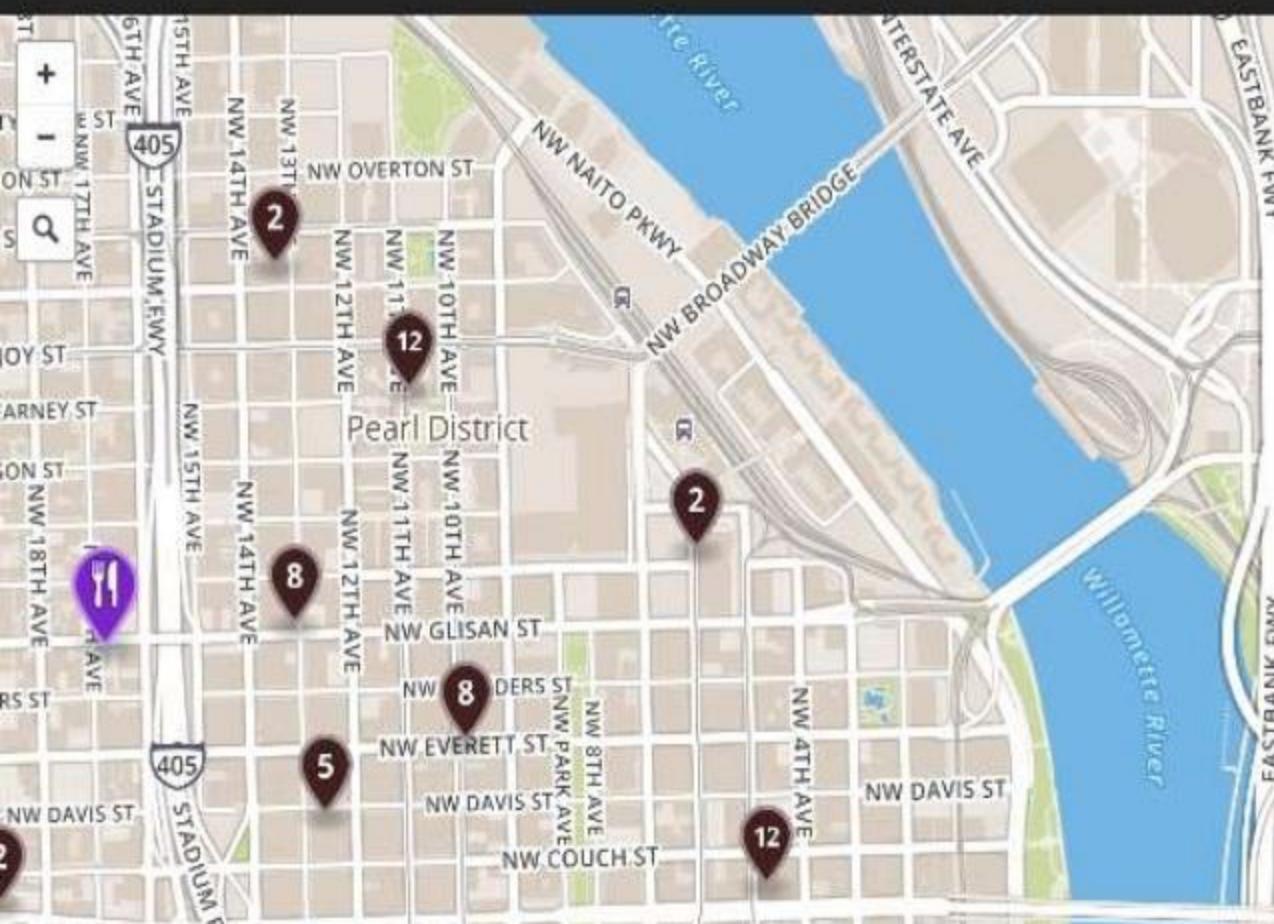
Restaurant Microservice



Restaurant Microservice





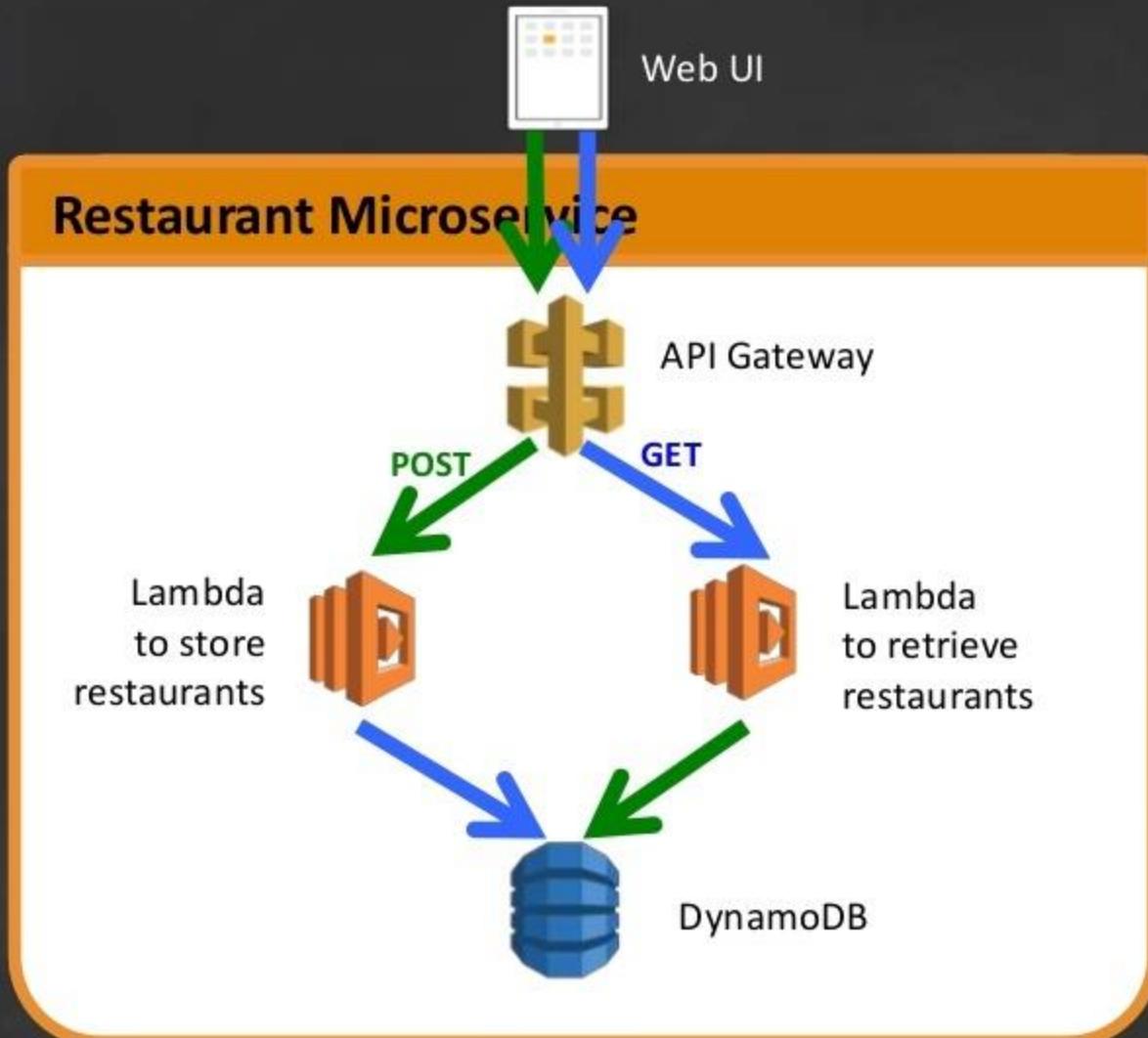
**RESTAURANT NAME****LATITUDE****LONGITUDE****CUISINE**

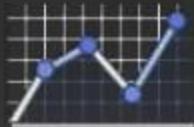
Asian

PRICE

Affordable (\$)

Add restaurant





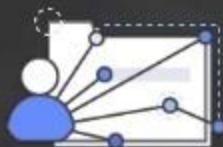
Highly Scalable

- Inherently scalable



Secure

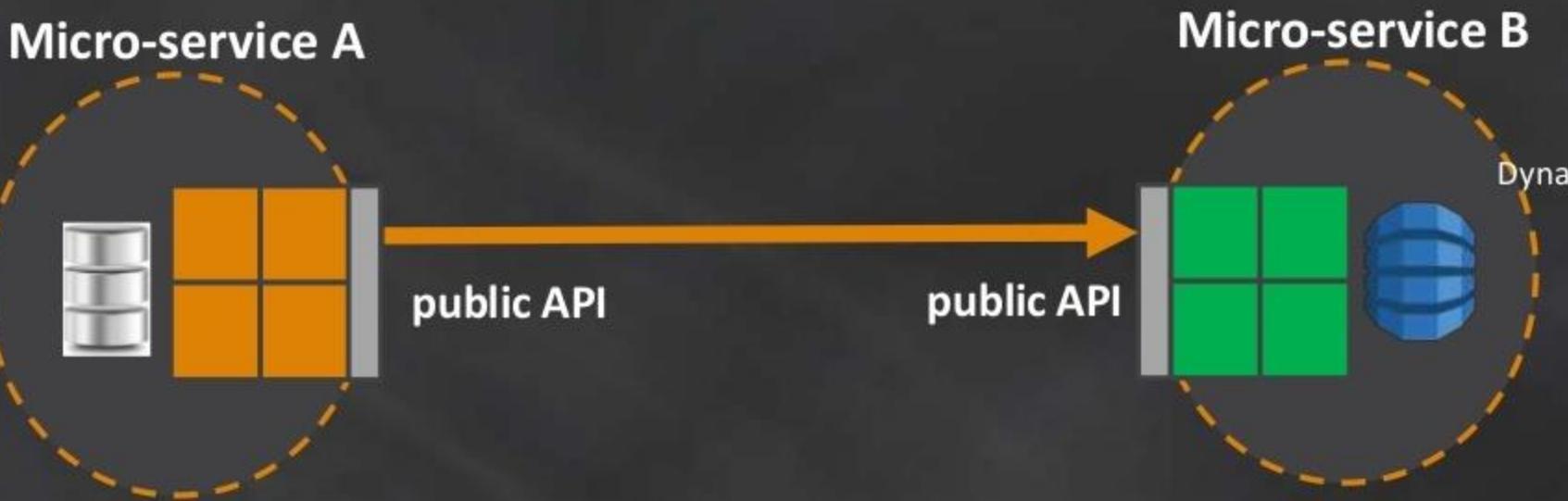
- API Gateway acts as “front door”
- Can add authN/authZ; or throttle API if needed
- S3 bucket policies
- IAM Roles for Lambda invocations



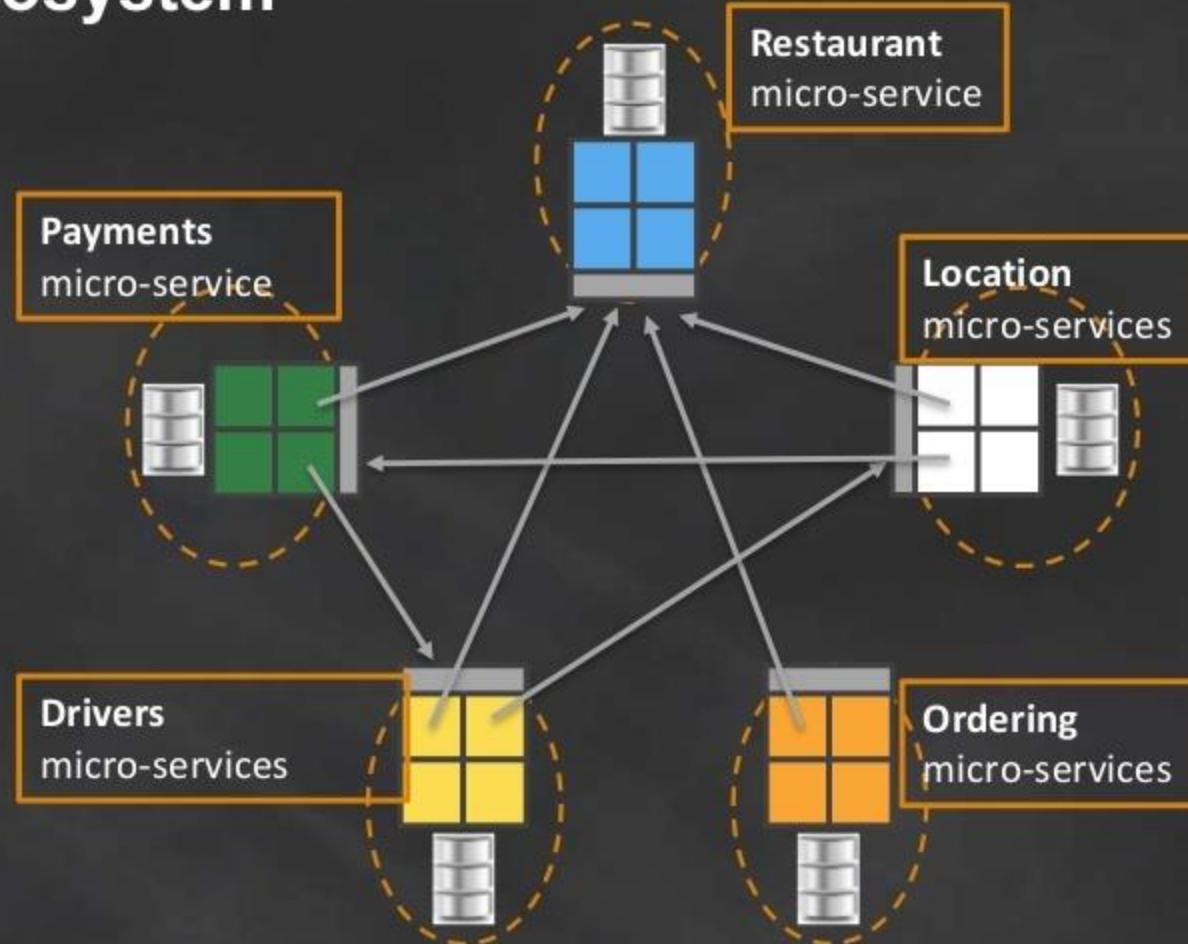
Cost-efficient

- Only pay for actual microservice usage

From a few...



...to an ecosystem

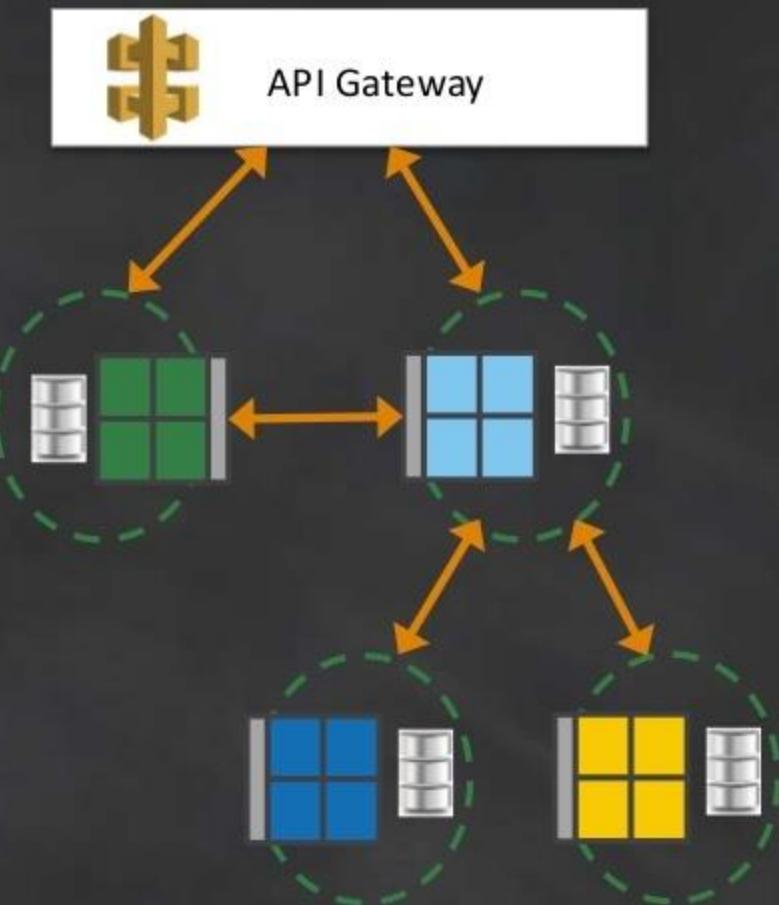




Principle 3

Secure Your Services

Principle 3: Secure Your Services



- **Defense-in-depth**

- Network level (e.g. VPC, Security Groups, TLS)
- Server/container-level
- App-level
- IAM policies

- **Gateway** ("Front door")

- **API Throttling**

- **Authentication & Authorization**

- Client-to-service, as well as service-to-service
- API Gateway: custom Lambda authorizers
- IAM-based Authentication
- Token-based auth (JWT tokens, OAuth 2.0)

- **Secrets management**

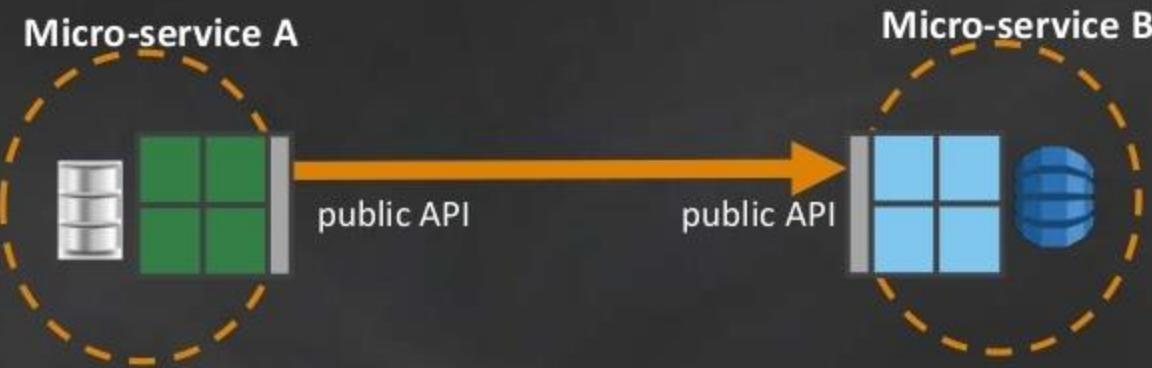
- S3 bucket policies + KMS + IAM
- Open-source tools (e.g. Vault, Keywhiz)



Principle 4

**Be a good citizen
within the ecosystem**

Principle 4: Be a good citizen within the ecosystem



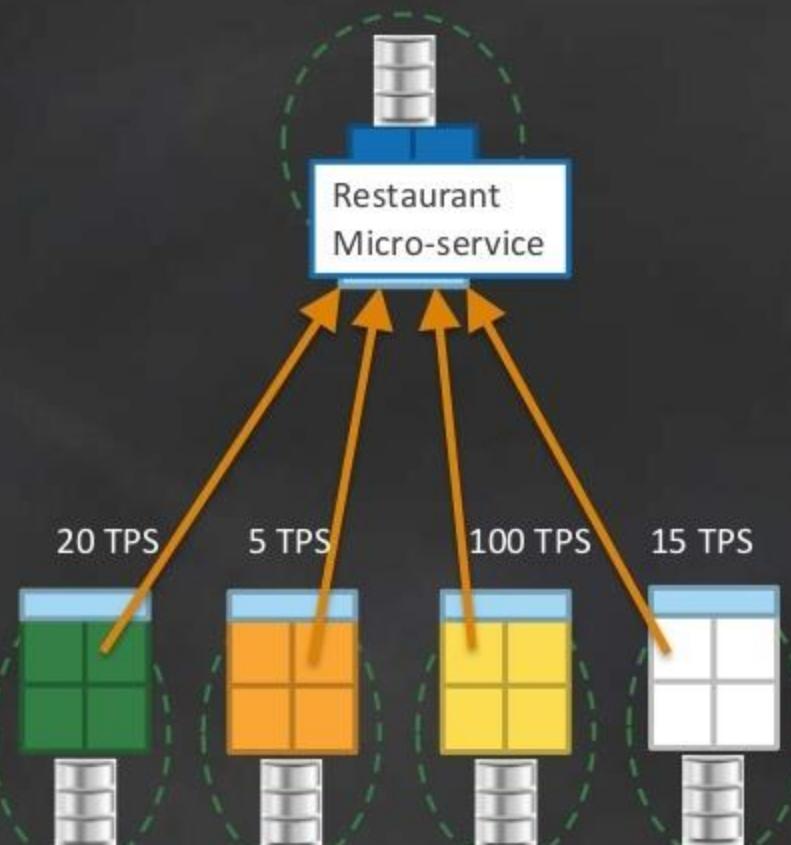
Hey Sally, we need to call your micro-service to fetch restaurants details.

Sure Paul. Which APIs you need to call? Once I know better your use cases I'll give you permission to register your service as a client on our service's directory entry.

Principle 4: Be a good citizen within the ecosystem (Have clear SLAs)



Before we let you call our micro-service we need to understand your use case, expected load (TPS) and accepted latency



Principle 4: Be a good citizen within the ecosystem (Distributed monitoring, logging and tracing)

Distributed monitoring and tracing

- “Is the service meeting its SLA?”
- “Which services were involved in a request?”
- “How did downstream dependencies perform?”

Shared metrics

- e.g. request time, time to first byte

Distributed tracing

- e.g. Zipkin, OpenTracing

User-experience metrics





Principle 5

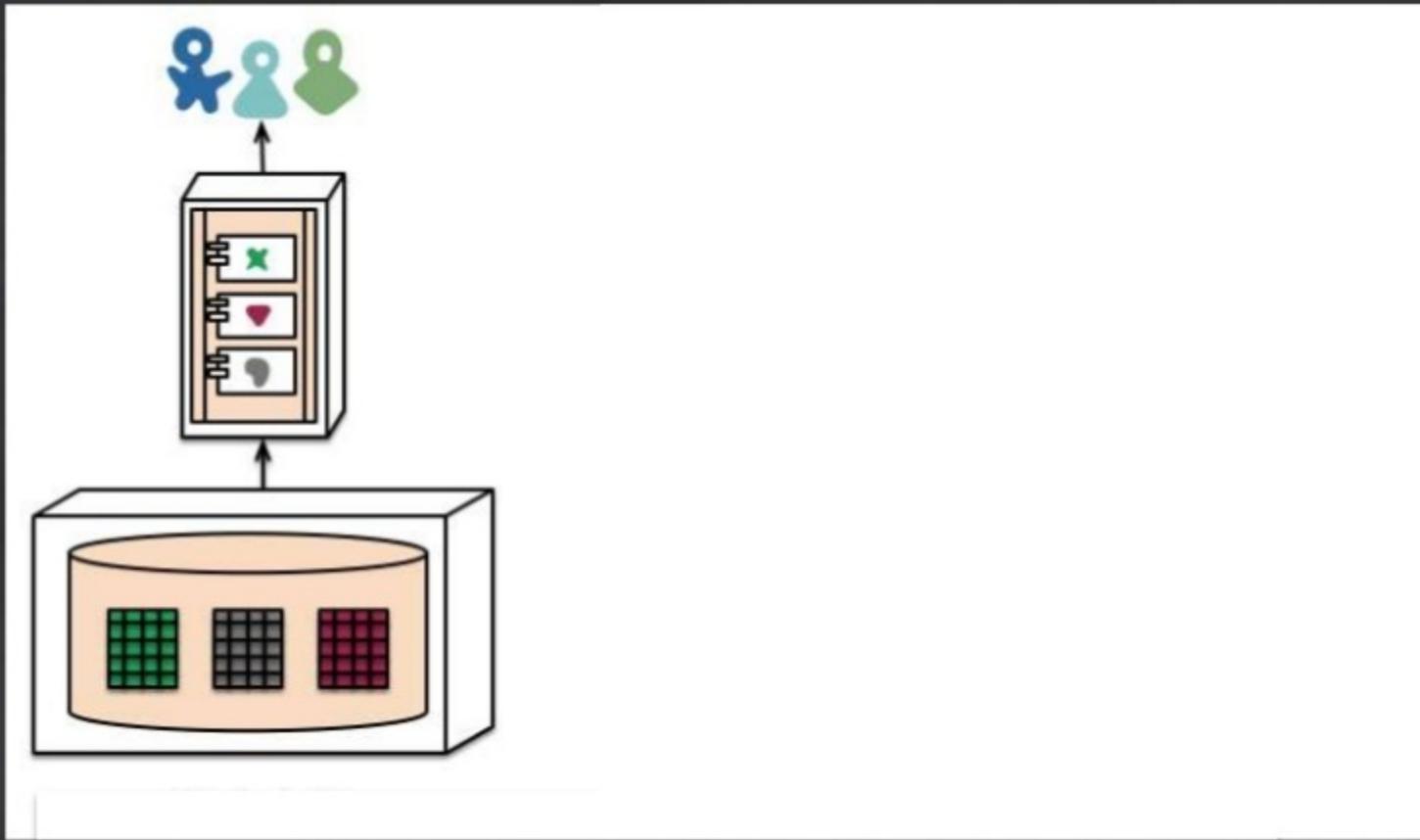
More than just
technology transforma

Conway's Law

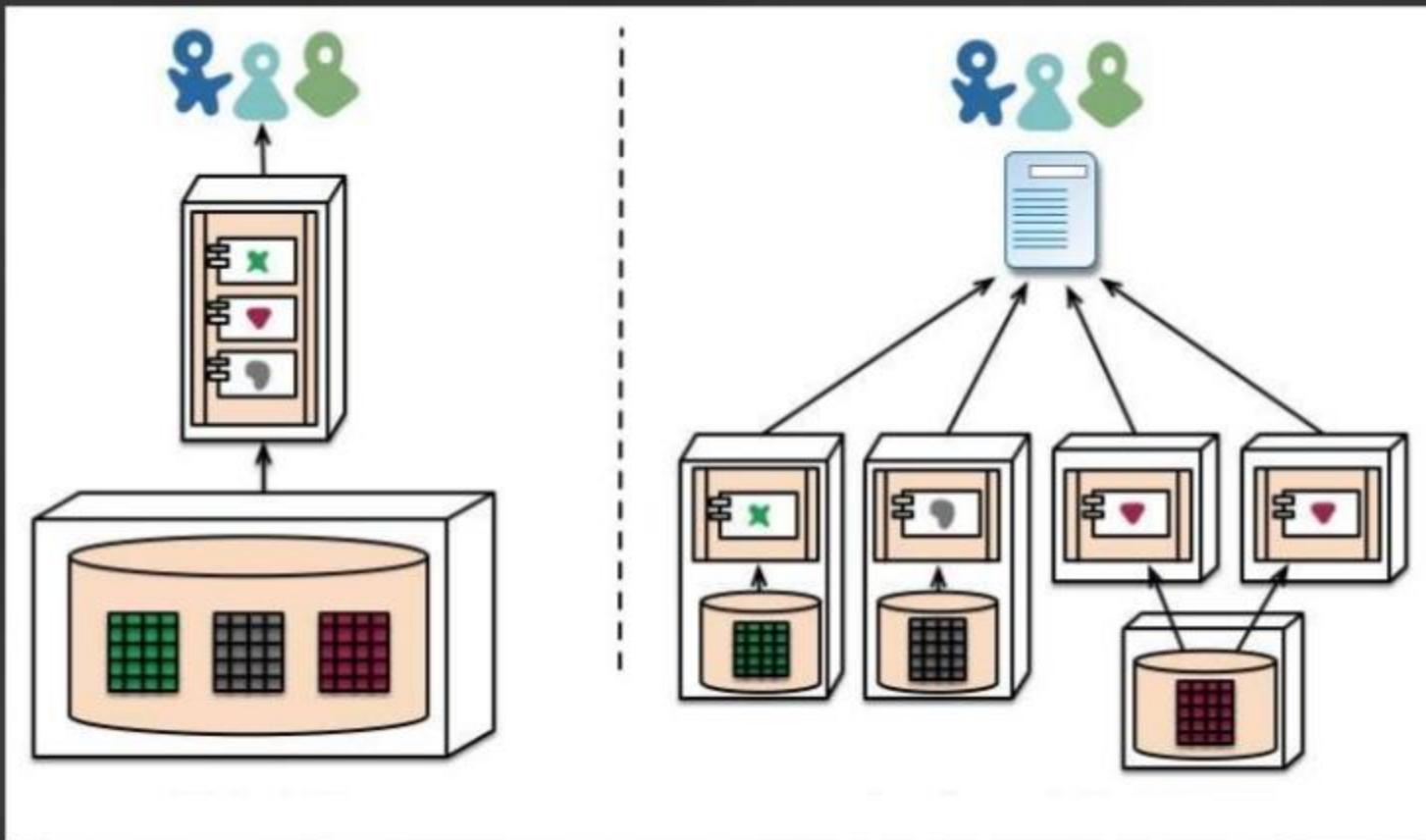
“Any organization that designs a system will inevitably produce a design whose structure is a copy of the organization’s communication structure.”

Melvin E. Conway, 1967

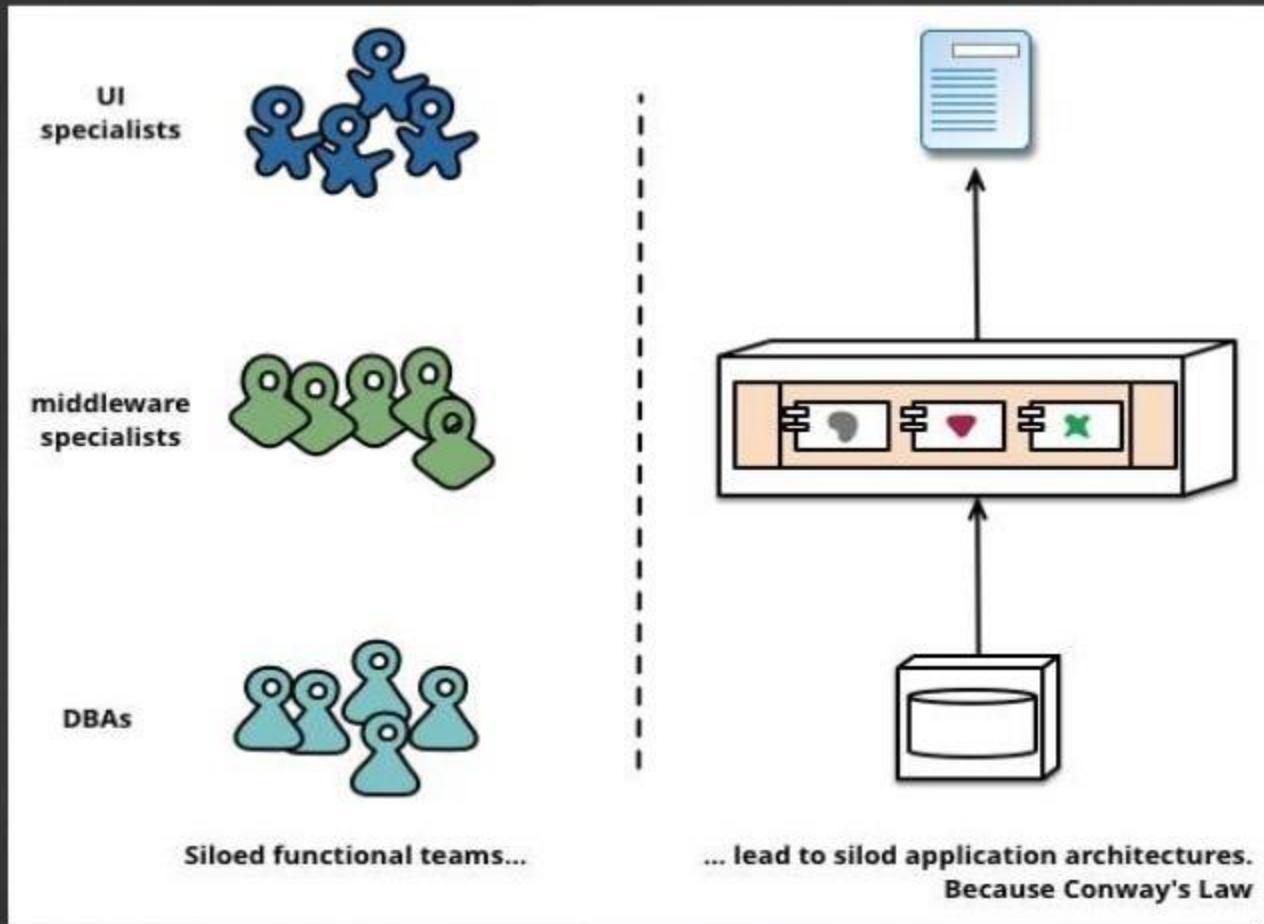
Decentralize governance and data management



Decentralize governance and data management

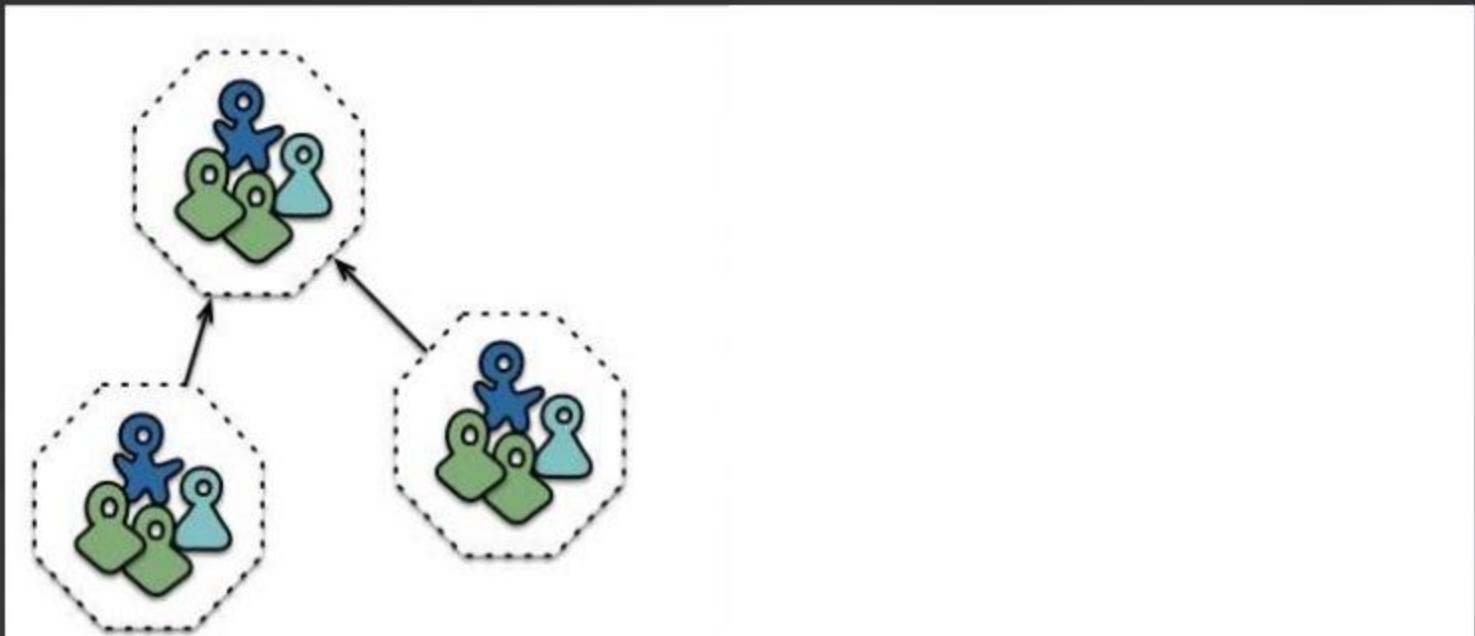


Silo'd functional teams → silo'd application architectures

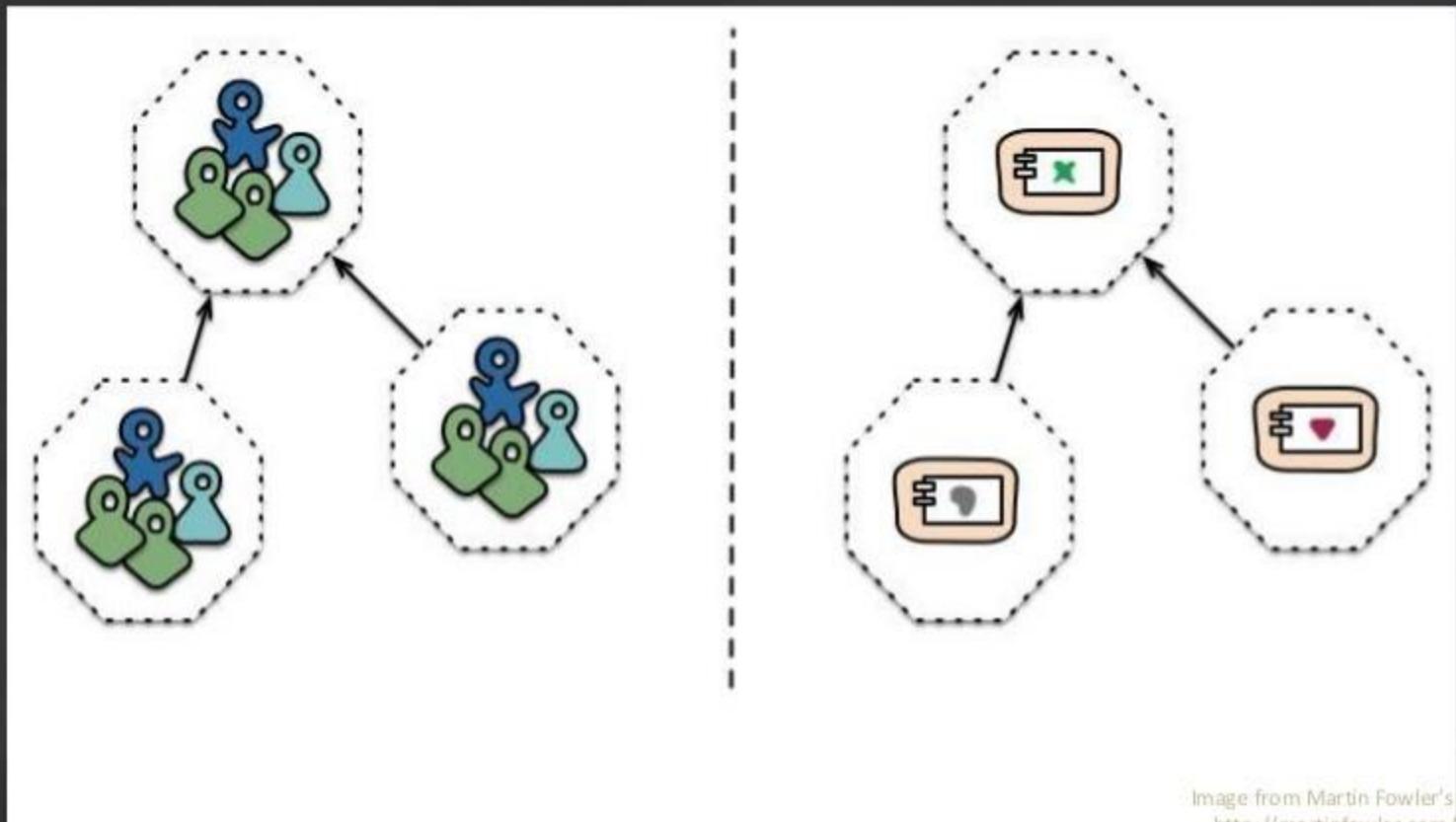


e from Martin Fowler's article on micro
[p://martinfowler.com/articles/microse](http://martinfowler.com/articles/microse)

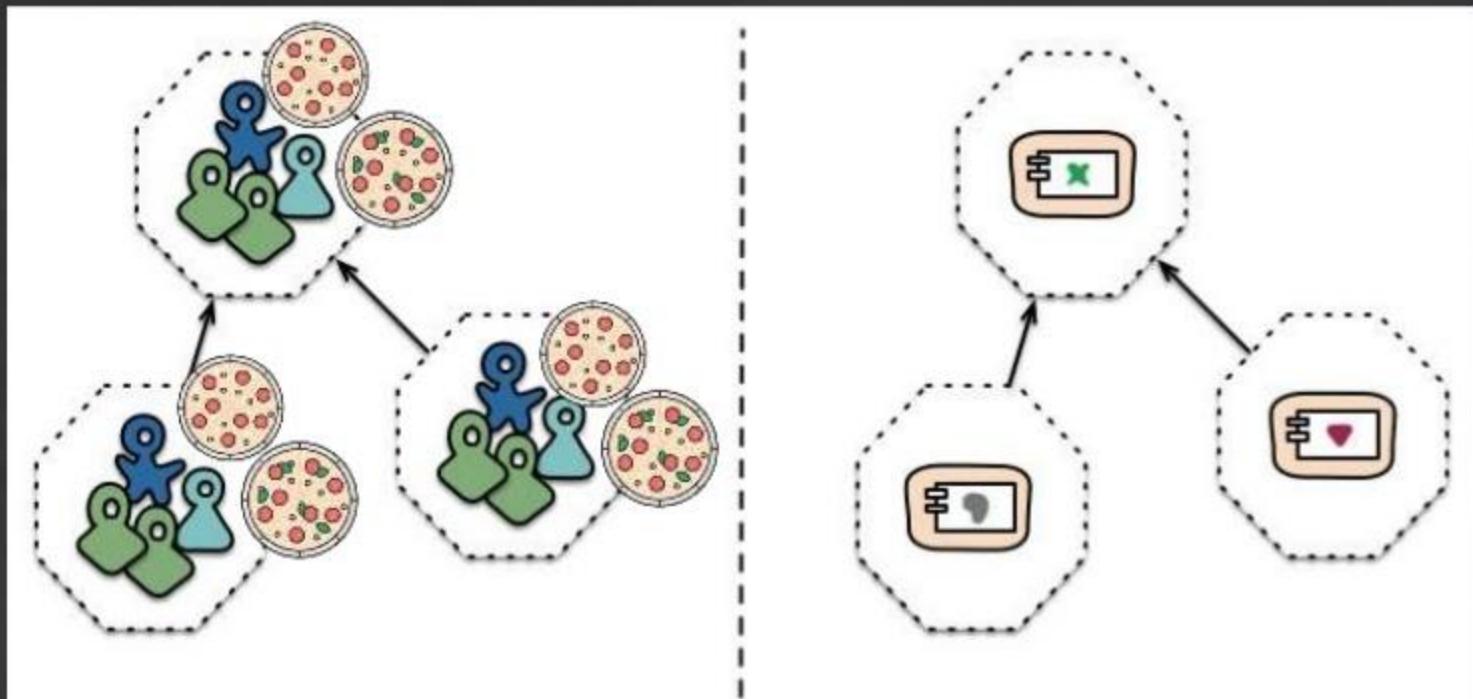
Cross functional teams → self-contained services



Cross functional teams → self-contained services



Cross functional teams → self-contained services ("Two-pizza teams" at Amazon)



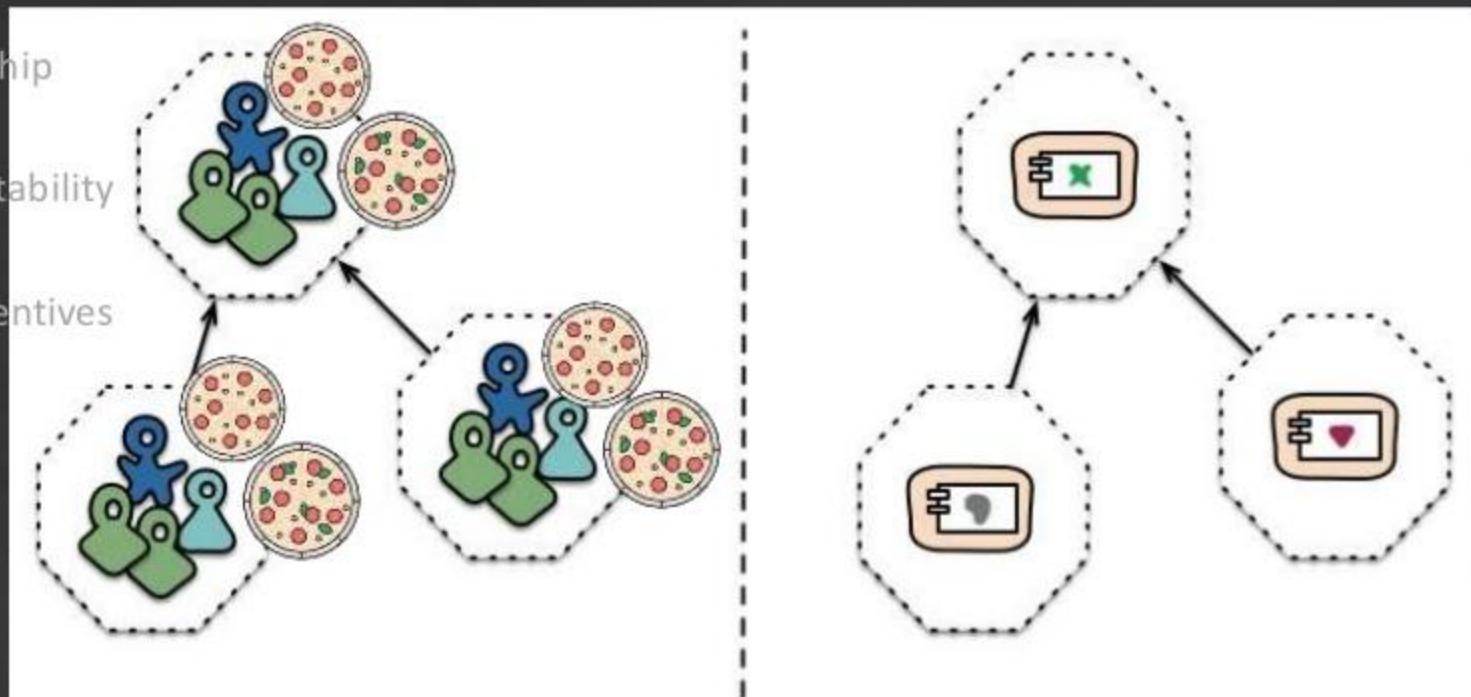
Cross functional teams → self-contained services ("Two-pizza teams" at Amazon)

Full ownership

Full accountability

Aligned incentives

"DevOps"



Principle 6

Automate Everything



Focused agile teams

2-pizza team



delivery pipeline

build

test

release



service



Focused agile teams

2-pizza team



delivery pipeline

build

test

release

service



build

test

release



Focused agile teams

2-pizza team



delivery pipeline

build

test

release

service



build

test

release



build

test

release



Focused agile teams

2-pizza team

delivery pipeline

service



Focused agile teams

2-pizza team

delivery pipeline

service

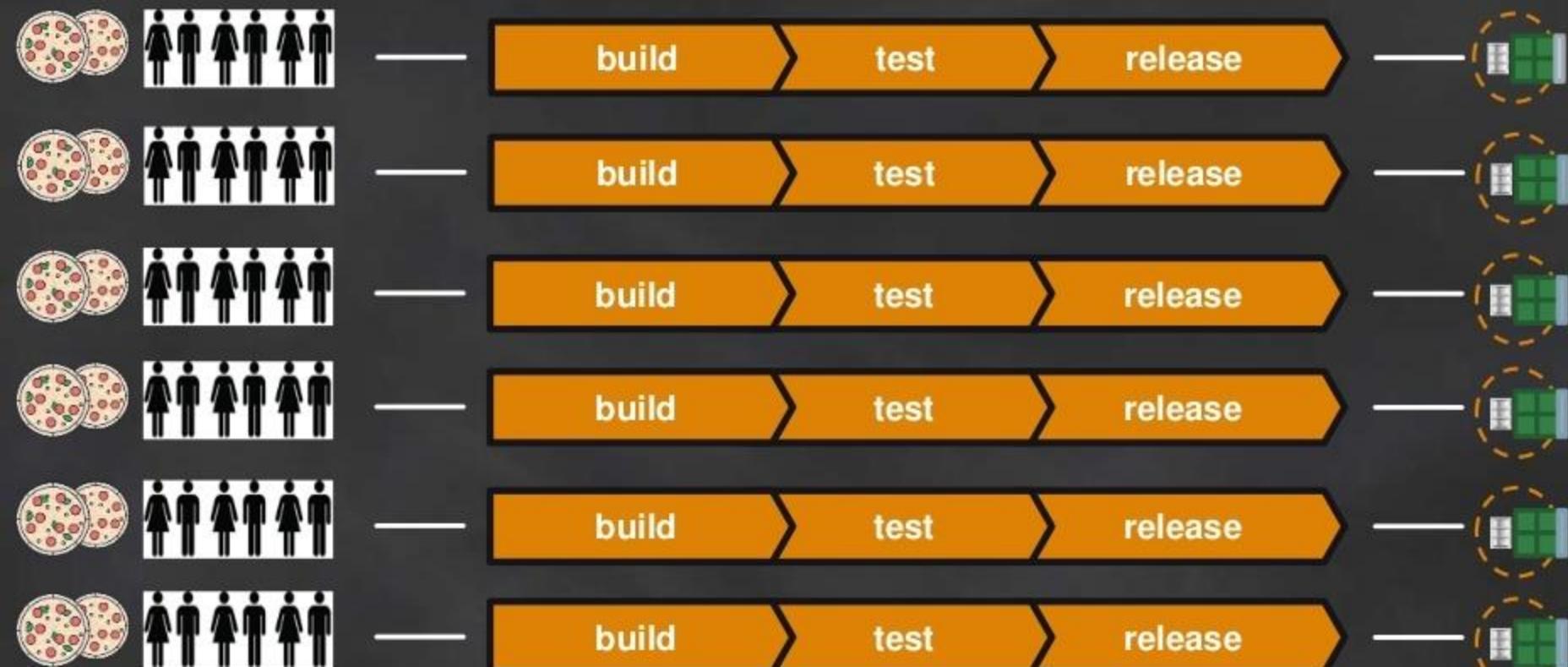


Focused agile teams

2-pizza team

delivery pipeline

service



Principle 6: Automate everything



AWS
CodeCommit



AWS
CodePipeline



AWS
CodeDeploy



RDS



DynamoDB



ElastiCache



CloudWatch



Cloud Trail



API Gateway



SQS



SWF



EC2



ECS



Lambda



Auto
Scaling



ELB



Elastic
Beanstalk



SES

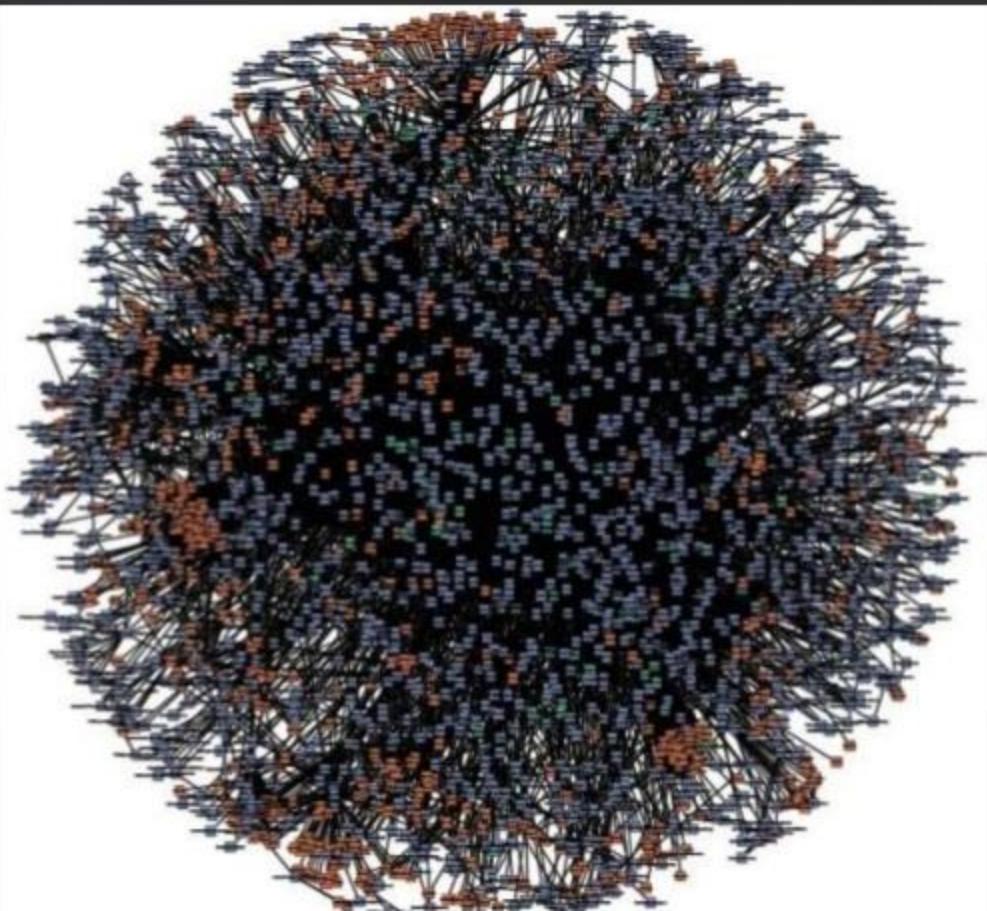


SNS



Kinesis

It's a journey...



Expect challenges along the way...

- Understanding of business domains
- Coordinating txns across multiple services
- Eventual Consistency
- Service discovery
- Lots of moving parts requires increased coordination
- Complexity of testing / deploying / operating a distributed system
- Cultural transformation

Principles of Microservices

1. Rely only on the public API

- Hide your data
- Document your APIs
- Define a versioning strategy

4. Be a good citizen within the ecosystem

- Have SLAs
- Distributed monitoring, logging, tracing

2. Use the right tool for the job

- Polygot persistence (data layer)
- Polyglot frameworks (app layer)

5. More than just technology transformation

- Embrace organizational change
- Favor small focused dev teams

3. Secure your services

- Defense-in-depth
- Authentication/authorization

6. Automate everything

- Adopt DevOps

Benefits of microservices

Easier to scale
each
individual
micro-service

Rapid
Build/Test/Release
Cycles

Clear ownership and
accountability

Benefits of microservices

Easier to scale
each
individual
micro-service

Easier to
maintain and
evolve system

Rapid
Build/Test/Release
Cycles

New releases
take minutes

Clear ownership and
accountability

Short time to add
new features

Benefits of microservices

Easier to scale
each
individual
micro-service

Easier to
maintain and
evolve system

Increased agility

Rapid
Build/Test/Release
Cycles

New releases
take minutes

Faster innovation

Clear ownership and
accountability

Short time to add
new features

Delighted customers

Additional resources

<https://aws.amazon.com/devops/>

AWS provides a set of flexible services designed to enable companies to more rapidly and reliably build and deliver products using AWS and DevOps practices. These services simplify provisioning and managing infrastructure, deploying application code, automating software release processes, and monitoring your application and infrastructure performance.

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity, evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

[Learn more about DevOps -](#)

Jump to an overview of our DevOps tooling -

Additional AWS resources:

- **Zombie Microservices Workshop:**
<https://github.com/awslabs/aws-lambda-zombie-workshop>
- **Serverless Webapp - Reference Architecture:**
<https://github.com/awslabs/lambda-refarch-webapp>
- **Microservices with ECS:**
<https://aws.amazon.com/blogs/compute/using-amazon-api-gateway-with-microservices-deployed-on-amazon-ecs>
- **Serverless Service Discovery:**
<https://aws.amazon.com/blogs/developer/serverless-service-discovery-part-1-get-started/>
- **ECS Service Discovery:**
<https://aws.amazon.com/blogs/compute/service-discovery-an-amazon-ecs-reference-architecture>
- **Microservices without the Servers**
<https://aws.amazon.com/blogs/compute/microservices-without-the-servers>

Popular open-source tools:

- **Serverless** – <http://serverless.com>
- **Apex** - <http://apex.run/>

Thank you!

Peter Dalbhanjan
dalbhanj@amazon.com