Building Resilient Serverless Systems

@johnchapin | symphonia.io

John Chapin

- Partner, Symphonia
- Former VP Engineering, Technical Lead
  - Data Engineering and Data Science teams
- 20+ yrs experience in govt, healthcare, travel, and ad-tech
- Intent Media, RoomKey, Meddius, SAIC, Booz Allen

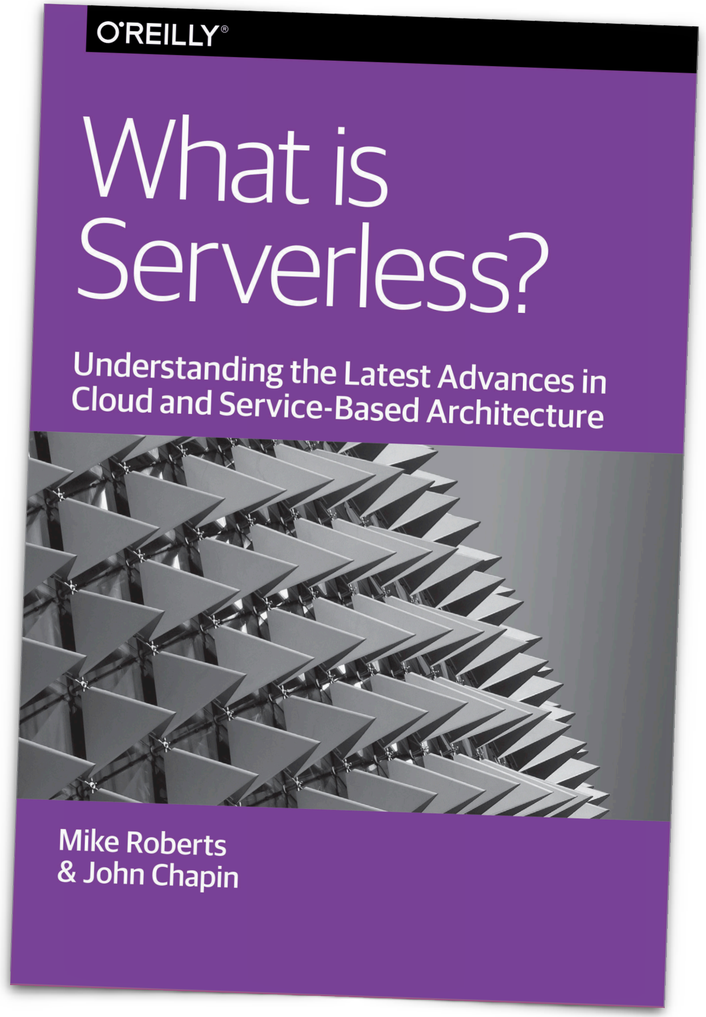
Agenda

- What is Serverless?
- Resiliency
- Demo
- Discussion and Questions

What is Serverless?

Serverless = FaaS + BaaS!

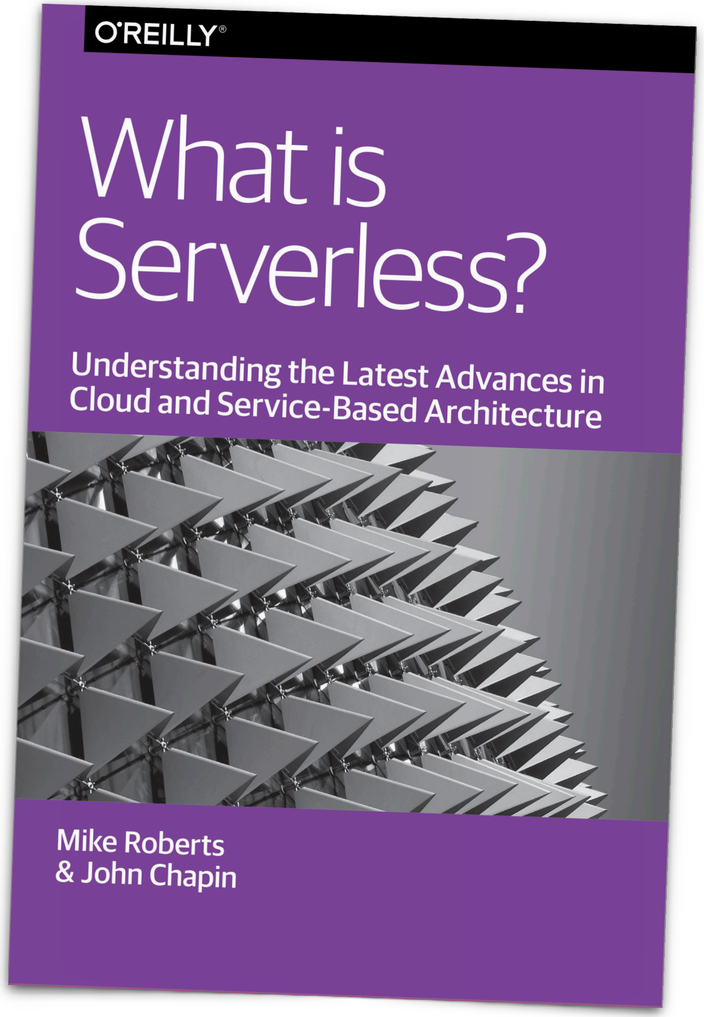
- **FaaS** = **F**unctions **a**s **a** **S**ervice
  - AWS Lambda, Auth0 Webtask, Azure Functions, Google Cloud Functions, etc...
- **BaaS** = **B**ackend **a**s **a** **S**ervice
  - Auth0, Amazon DynamoDB, Google Firebase, Parse, Amazon S3, etc...



go.symphonia.io/what-is-serverless

Serverless attributes

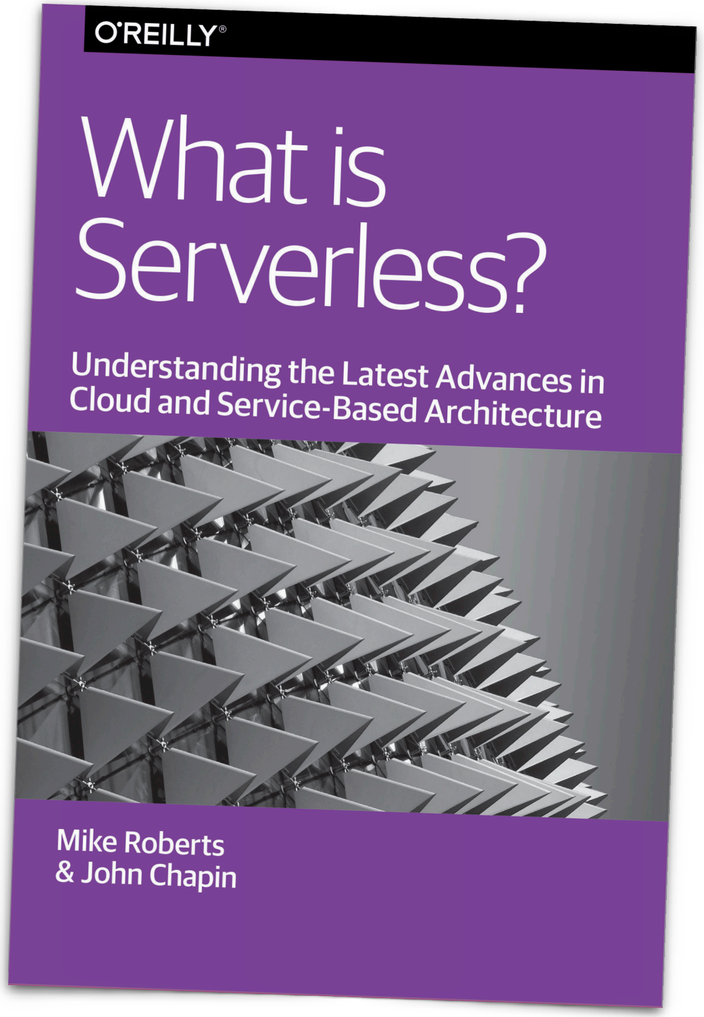
- No managing of hosts or processes
- Self auto-scaling and provisioning
- Costs based on precise usage (down to zero!)
- Implicit high availability



go.symphonia.io/what-is-serverless

Serverless benefits

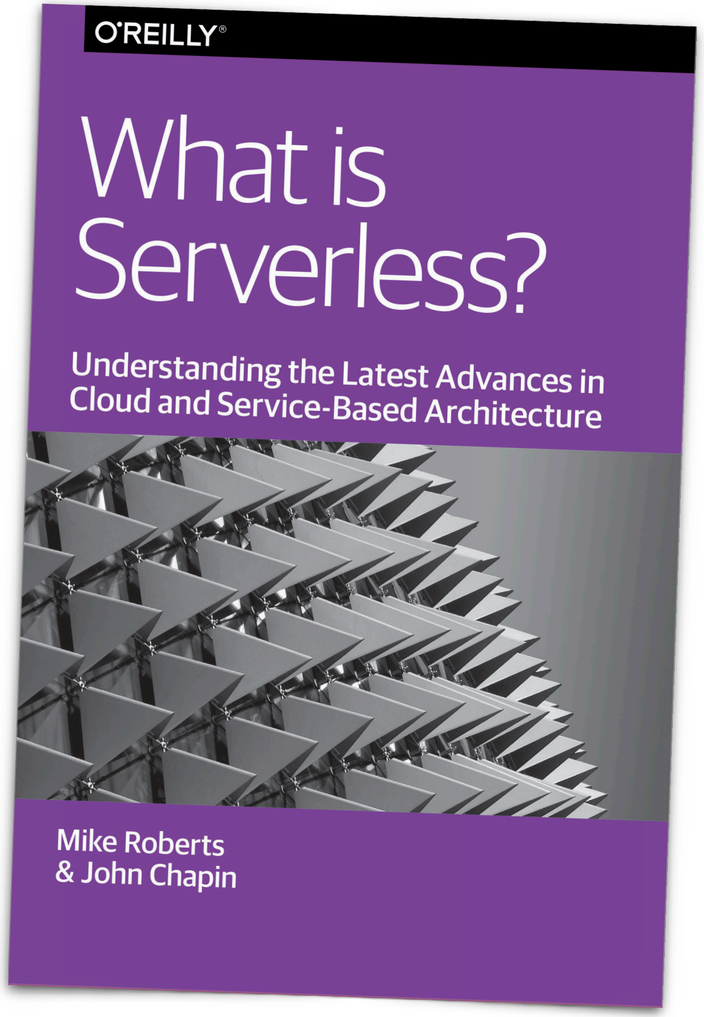
- Cloud benefits ++
  - Reduced TCO
  - Scaling flexibility
  - Shorter lead time



go.symphonia.io/what-is-serverless

Loss of control

- Limited configuration options
- Fewer opportunities for optimization
- Hands-off issue resolution



go.symphonia.io/what-is-serverless

Resiliency

“Failures are a given and everything will eventually fail over time …”

–Werner Vogels

(https://www.allthingsdistributed.com/2016/03/10-lessons-from-10-years-of-aws.html)

Werner on Embracing Failure

- Systems will fail. At scale, systems will fail a lot.
- Embrace failure as a natural occurrence.
- Limit the blast radius of failures.
- Keep operating.
- *Recover quickly (automate!)*

K.C. Green, Gunshow #648

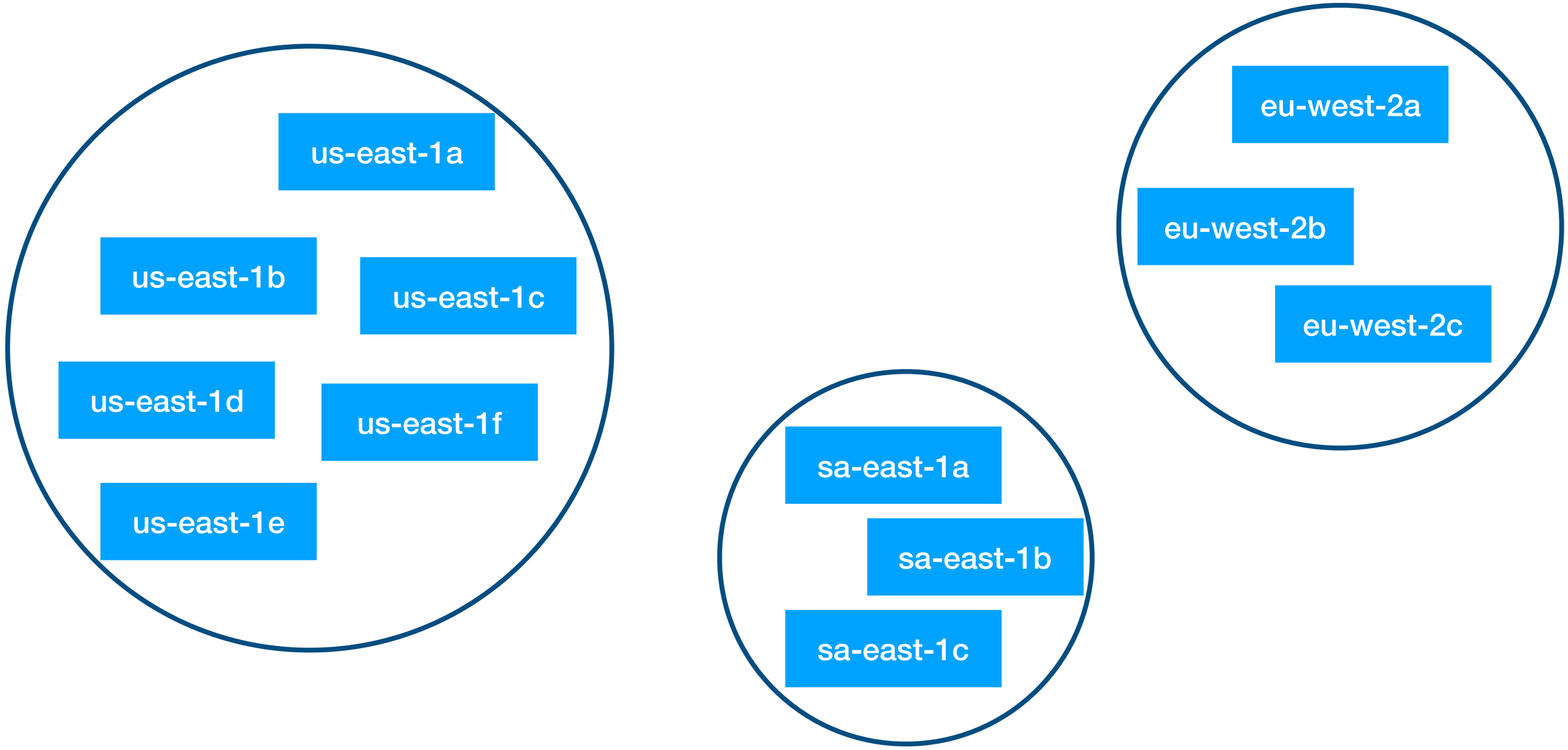
Failures in Serverless land

- Serverless (or Serviceful) is all about using vendor-managed services.
- Two broad classes of failures:
  - **Application failures** (your problem, your resolution)
  - **Service failures** (your problem, but not your resolution)
- What happens when those vendor-managed services fail?

Mitigation through architecture

- No control over resolving acute vendor failures.
- Plan for failure, architect and build applications to be resilient.
- Take advantage of:
  - Vendor-designed isolation mechanisms (like AWS regions).
  - Vendor services designed to work across regions (like Route 53).
- Take advantage of vendor-recommended architectural practices, like the AWS Well-Architected Framework's Reliability Pillar: https://d1.awsstatic.com/whitepapers/architecture/AWS-Reliability-Pillar.pdf

AWS isolation mechanisms



Serverless resiliency on AWS

- Regional high-availability = services running across multiple availability zones in one region.
  - With EC2 (and other traditional instance-based services), it's our problem.
  - With Serverless (Lambda, DynamoDB, S3, etc), AWS handle it for us.
- Global high-availability = services running across multiple regions.
  - **We can architect our systems for global high-availability.**
  - **The Serverless cost model is a huge advantage!**

Serverless resiliency on AWS

- **Event-driven** Serverless systems with externalized state mean:
  - Little or no data in-flight when a failure occurs
  - Data persisted to reliable stores (like DynamoDB or S3)
- Serverless **continuous deployment** means:
  - No persistent infrastructure to re-hydrate
  - Highly likely to be a portable, infrastructure-as-code approach

Not just resiliency

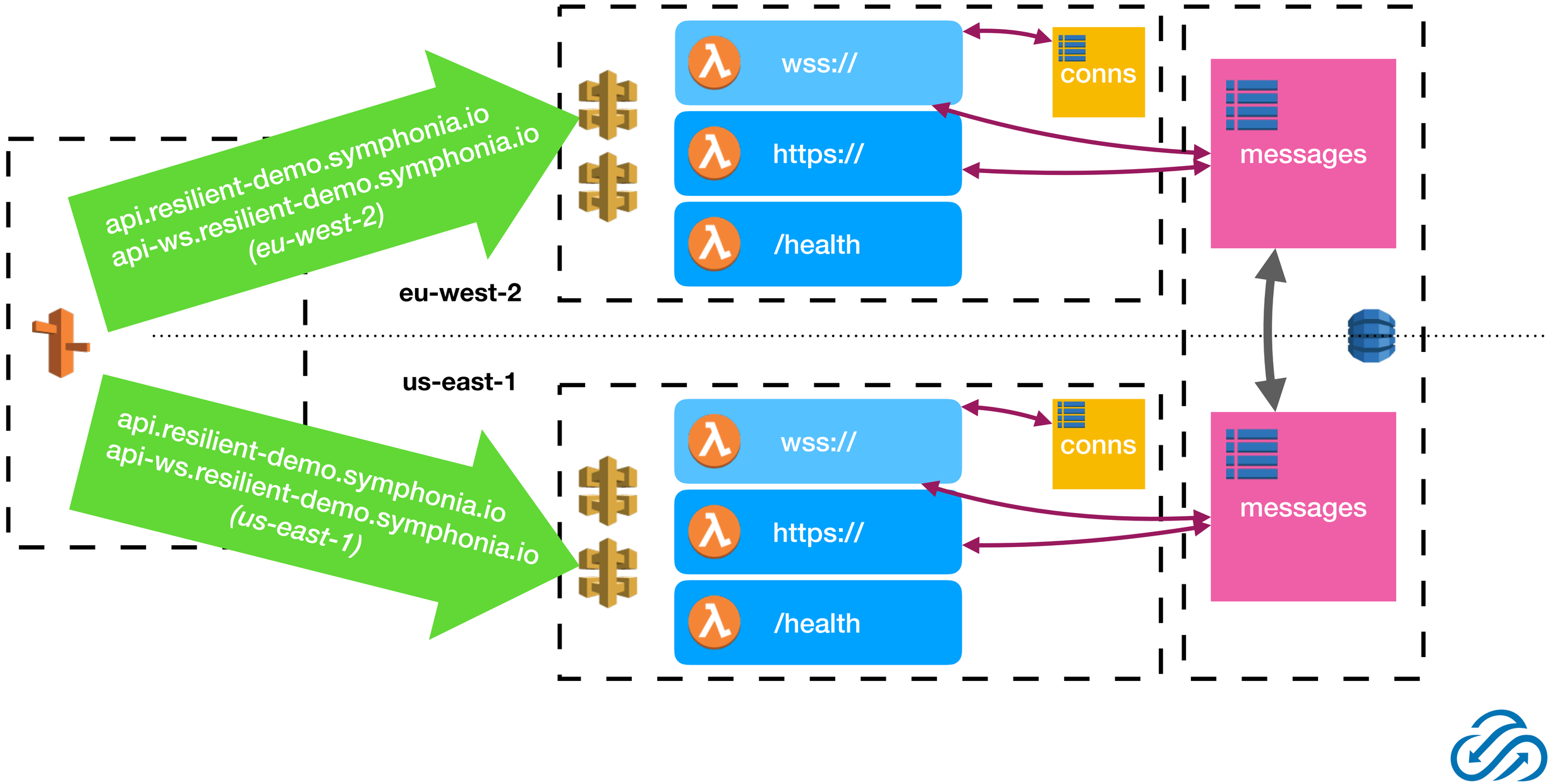
- Regional infrastructure is closer to regional users
- Because Serverless is "pay per request", total costs are similar
- Infrastructure-as-code minimizes incremental work in deploying to new region
- Automated multi-region deployment keeps infrastructure up-to-date

The nature of Serverless systems makes it ~~easy~~possible to architect for resiliency to vendor failures.

Demo

Overview

- Global, highly-available API
- https://github.com/symphoniacloud/resilient-serverless-systems
  - Serverless Application Model (SAM) template
  - Lambda code (Typescript)
  - Build system (NPM + shell)
  - Elm front-end


api.resilient-demo.symphonia.io
api-ws.resilient-demo.symphonia.io
(eu-west-2)
eu-west-2
wss://
https://
/health
conns
messages
us-east-1
api.resilient-demo.symphonia.io
api-ws.resilient-demo.symphonia.io
(us-east-1)
wss://
https://
/health
conns
messages

Request flow

- DNS lookup for **api.resilient-demo.symphonia.io**
- Route 53 responds with IP address for
  - lowest latency regional API Gateway endpoint
  - that has a passing health check (HTTP 2xx or 3xx from **/health** endpoint)
- Request traverses regional API Gateway to regional Lambda
- Regional Lambda writes to regional DynamoDB table
- DynamoDB replicates data to **all** replica tables in other regions, last write wins

Simulating failure

- Alter **us-east-1** health check to return HTTP error status
- Observe HTTP request routed to **eu-west-2** instead
- Observe DynamoDB writes propagated from **eu-west-2** back to **us-east-1**

Rough edges

- DynamoDB Global Tables not available in CloudFormation
- API Gateway WebSockets + Custom Domains not available in CloudFormation
- Can't add new replicas to DynamoDB global tables after inserting data
- SAM not compatible with CloudFormation Stack Sets

Additional approaches

- Multi-region deployment via Code Pipeline
  https://github.com/symphoniacloud/multi-region-codepipeline

- CloudFront Origin Failover
  https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/high_availability_origin_failover.html

- Global Accelerator (for ELB, ALB, and EIP)
  https://aws.amazon.com/global-accelerator/

AWS Resources

- James Hamilton's "Amazon Global Network Overview"
  https://www.youtube.com/watch?v=uj7Ting6Ckk

- Rick Houlihan's DAT401: Advanced Design Patterns for DynamoDB
  https://www.youtube.com/watch?v=HaEPXoXVf2k

- https://aws.amazon.com/blogs/compute/building-a-multi-region-serverless-application-with-amazon-api-gateway-and-aws-lambda/
  (Magnus Bjorkman, November 2017)

- https://aws.amazon.com/blogs/database/how-to-use-amazon-dynamodb-global-tables-to-power-multiregion-architectures/
  (Adrian Hornsby, December 2018)

- https://aws.amazon.com/blogs/compute/announcing-websocket-apis-in-amazon-api-gateway/ (Diego Magalhaes, December 2018)

Symphonia resources

- **What is Serverless?** Our 2017 report, published by O'Reilly.
- **Programming AWS Lambda** - Our upcoming full-length book with O'Reilly.
- **Serverless Architectures** - Mike's de facto industry primer on Serverless.
- **Learning Lambda** - A 9-part blog series to help new Lambda devs get started.
- **Serverless Insights** - Our email newsletter covering Serverless news, event, etc.
- **The Symphonium** - Our blog, featuring technical content and analysis.

Stay in touch!

john@symphonia.io

@johnchapin

@symphoniacloud

symphonia.io/events

blog.symphonia.io