
serverless
What is it? Why should I care?
OBSERVIAN
Do Cloud Better.

Who am I?

Currently: Co-Founder & VP, Engineering @Observian

Past Life:

- VP, Engineering & DevOps @Ghostery
- Software Architect / Dev Lead @Autopoint
- Team Lead @DST, Inc

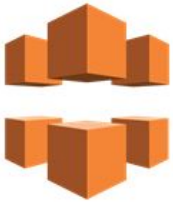
Current focus:

- Engineering at Scale
- Cloud Computing
- Serverless
- Dev Culture
- Containers
- Software Delivery

What is it?

WEB/STORAGE

S3



CloudFront

CONTAINERS

Fargate

COMPUTING

API Gateway

Lambda

Lambda function

DATABASES

Aurora

DynamoDB

ANALYTICS

Athena

MESSAGING

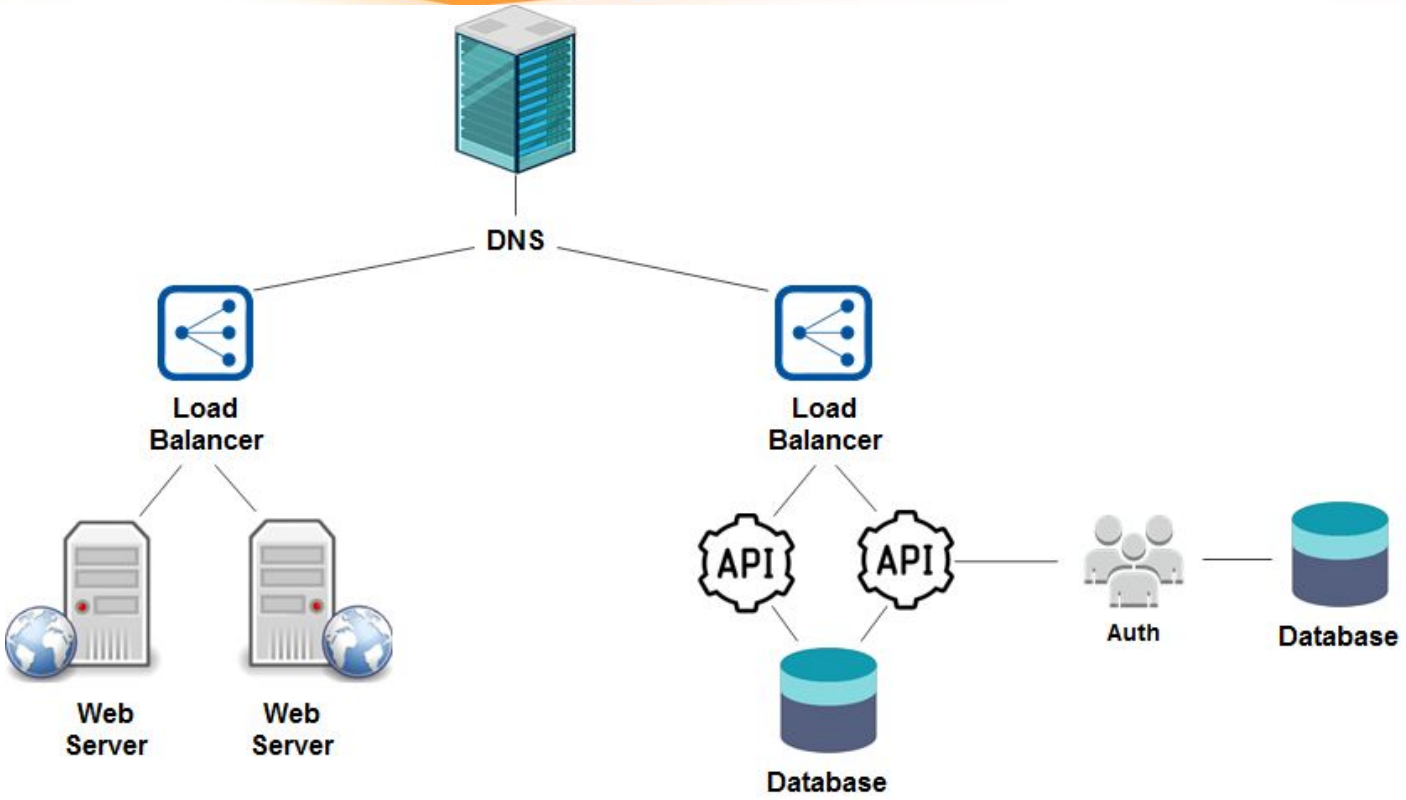
SQS

SNS

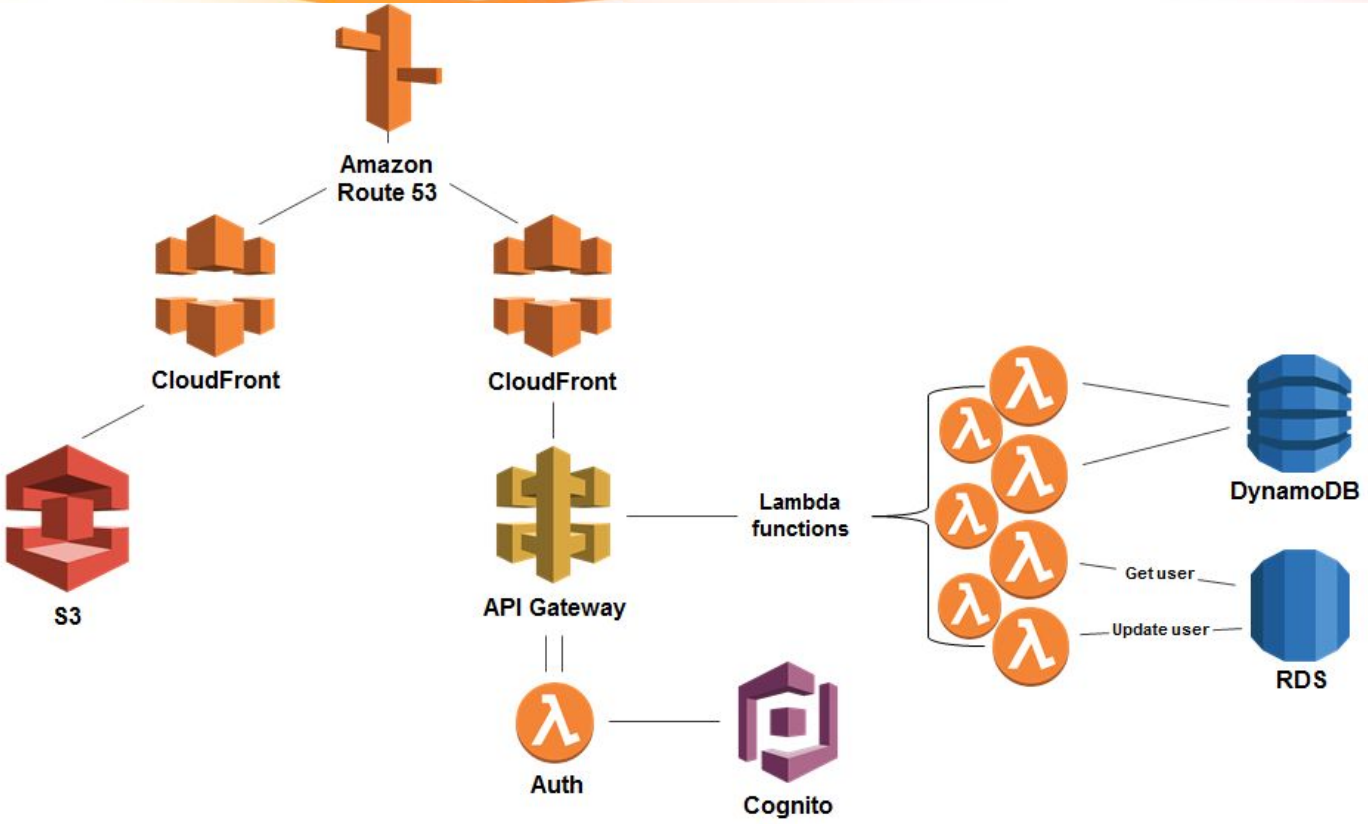
Who's doing it?

**(Sorta)

How do I get there? (Before)



How do I get there? (After)



Serverless Framework & You

You - Focus on the Code

AWS Lambda / Managed Services enables you to launch applications at record speed and cost

You get to:
Rapidly build apps that handle production-ready traffic

You don't have to:

- Actively manage scaling for your applications
- Provision servers regularly
- Pay for resources that go unused

Zero Server Admin | Auto-Scaling | Pay-per-use

Framework - Auto Config Cloud Vendor Settings

Build and deploy applications to any and every cloud provider with a consistent experience

- **Increase development speed** - build, test, deploy in the same environment
- **Avoid cloud vendor lock-in** - single package application deployed across all providers
- **Infrastructure as code –** formalize and standardize entire infrastructure
- **Existing ecosystem –** Serverless Framework is already widely adopted

Design Patterns -- Microservice “Nano”

What is it?

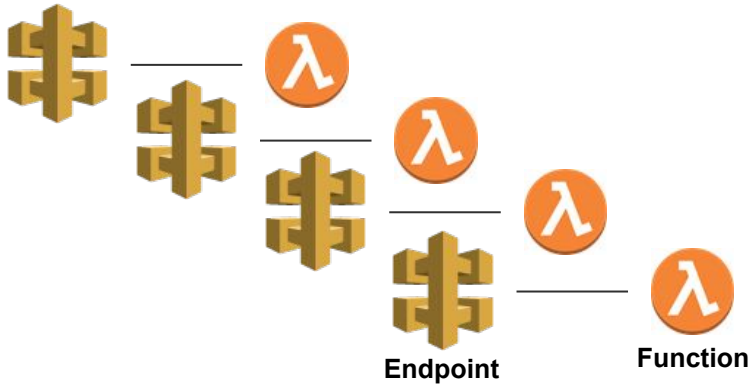
- Single Lambda per Function
- Single API endpoint per Function

Pros

- Separation of concerns down to the function level
- Easy to debug an individual function
- Easy testing
- Easy to deploy changes to a single function

Cons

- A LOT of functions as you grow
- Risk of hitting Cloud Formation limits
- Full environment deploys take longer
- **HIGHEST probability of a cold start**



Design Patterns -- Service “Micro”

What is it?

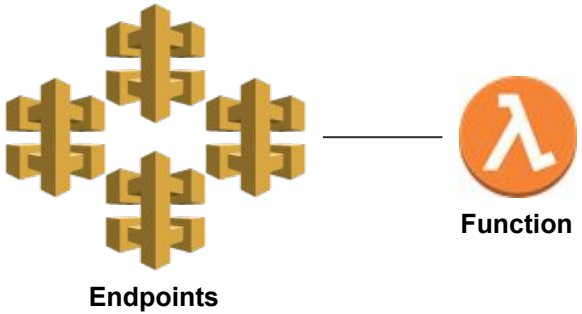
- Single Lambda handles multiple scoped Functions
- Multiple API endpoints per Function

Pros

- Separation of concerns down to the service level
- Easy to deploy single service
- Reduced cold starts

Cons

- Requires custom router
- Increased debugging complexity
- Harder to test
- Larger function size = more risk to changes



Design Patterns -- Monolithic

What is it?

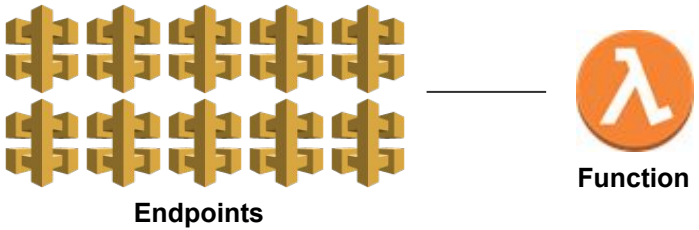
- Single Lambda for everything
- All API endpoints in one Function

Pros

- Feels more MVC like
- Faster deployment of entire system
- **LOWEST probability of a cold start**

Cons

- Requires a complex custom router
- Risk of hitting Cloud Formation limits
- Complex debugging
- Complex testing
- **Performance monitoring is all over the place Invocation run times are inconsistent**



Design Patterns -- Graph

What is it?

- Single Lambda for everything
- Very few API Endpoints required

Pros

- Low cost of ownership / scalable Graph Api
- Faster deployment of entire system
- **LOWEST probability of a cold start**

Cons

- Requires learning GraphQL
- Risk of hitting Cloud Formation limits
- Complex debugging
- Complex testing
- **Performance monitoring is all over the place Invocation run times are inconsistent**

Deployment & Secrets

Deploy

- Command line deploys
- Automated orchestration via Cloudformation
- Easily deploy to multiple regions
- Support for stages aka environments
    - Dev / Test / Staging / Prod etc

Configure

- ENV variables can be set in serverless.yml -- meh
- Serverless-secrets-plugin -- eh?
- AWS Secrets Manager -- expensive af
- SSM Parameter Store -- o'rly
- Observian Secret AWSome -- ya'rly

https://observian.com/tools/secret-awsome/

Demo Time

Future Meetup Topics

My plan:

- Serverless CI/CD Pipeline
- Monitoring Serverless Apps
- Securing Serverless Apps
- Multi-Region Serverless Apps
- Lambda Authorizers
- Leveraging AWS Fargate (Containers) for long running jobs
- How to build / package static S3 hosted sites

What do **YOU** guys wants to talk about?