Serverless Cloud Functions

CS199 - ACC

Prof. Robert J. Brunner

Ben Congdon
Tyler Kim
Bhuvan Venkatesh

“Serverless” Computing

What is serverless?

- What it's NOT:
  - The absense of servers (We're still in the cloud!)

- What it IS:
  - Ephemeral compute capabilities
  - Usually leverages some type of containerization
  - Short-lived (lives usually less than 1 second, almost always less than 5 minutes)

What is serverless?

- Think of “serverless” as a “cloud function”:
  - A function does one specific thing
  - It takes an input, does something, and has some output

- Only now, we can scale our “cloud function” to run in dozens or hundreds of parallel instances

Why “Serverless”?

- Lots of companies have moved to a “microservices” model
  - Lots of tiny servers to serve services instead of few beefy “monoliths”

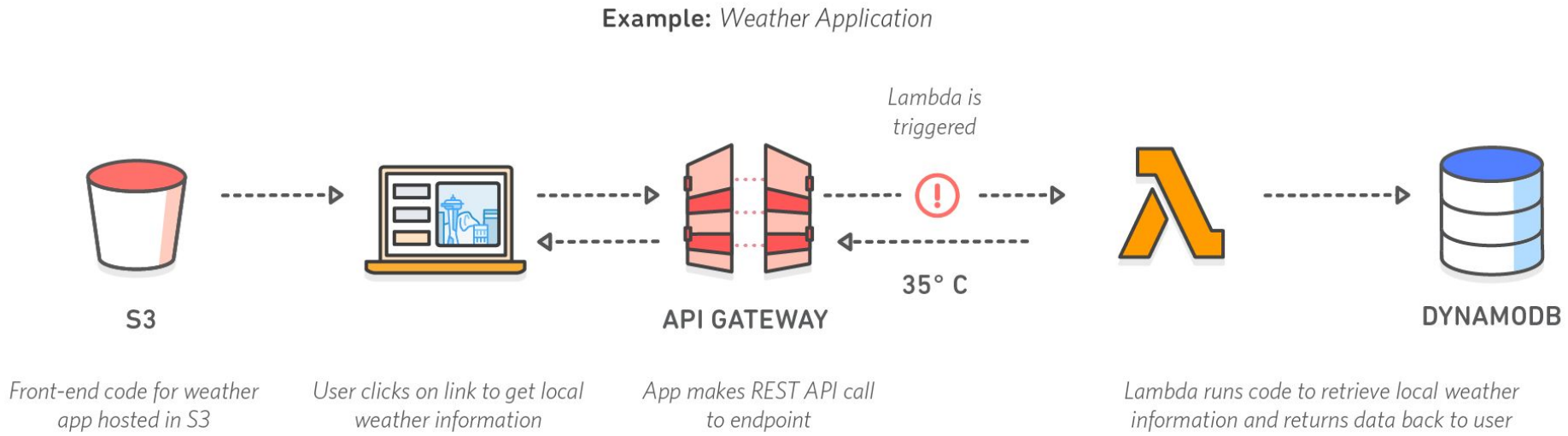
- What happens when you have a really low-traffic service?
  - Have to pay for a server to be idle. Leads to waste.

- Rise of ephemeral, event-driven computing in the cloud
  - We want to be able to respond to events in the cloud
  - i.e. A user uploads a new file, a new record is written to a database, an IoT device sends some data to the cloud
  - We can “spin up” a compute instance that only lives for the duration of that request

Serverless Use Cases

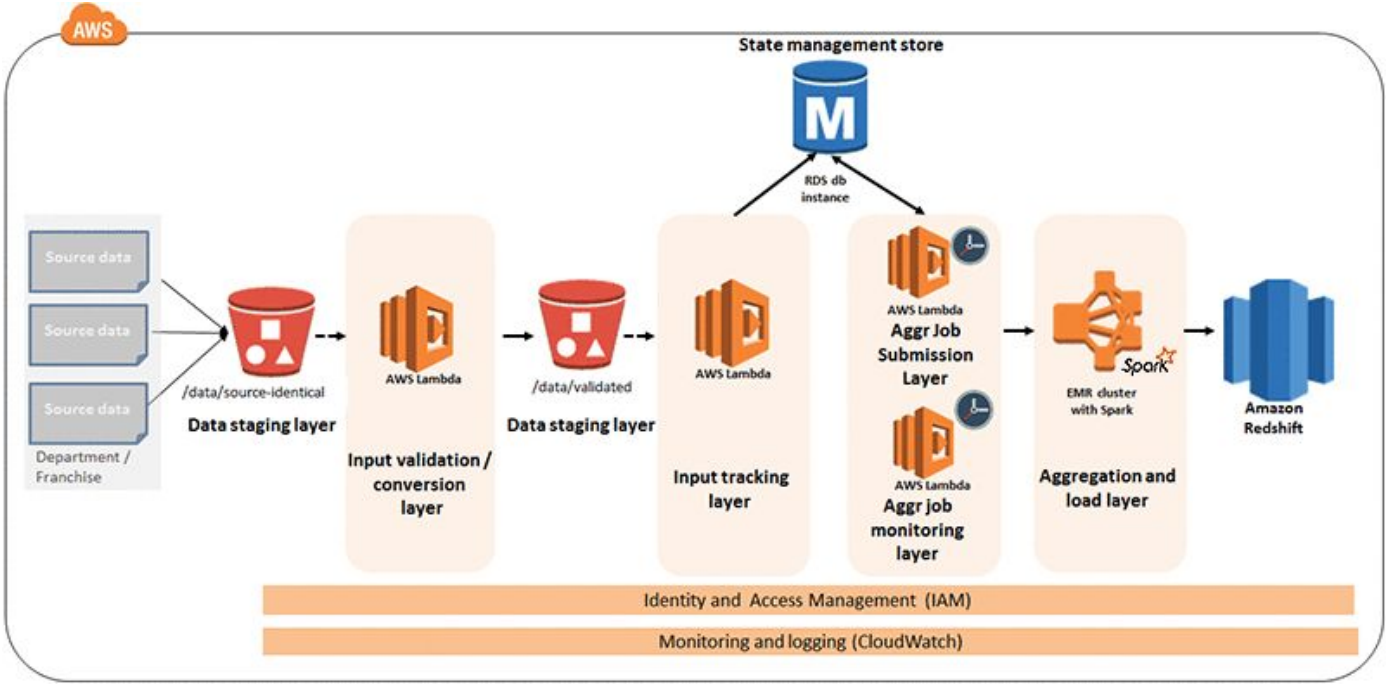
- Event-Driven Computation
  - Often heavily integrated with other Cloud services (i.e. message queues, storage buckets)
- Small Batch Data Processing
  - “Cheap” way to handle streaming data
- Low to Medium Traffic APIs
  - Serverless compute can act as
- IoT Backends
  - All of the above

… as an API / Web App



https://aws.amazon.com/serverless/

… for batch processing



https://aws.amazon.com/blogs/big-data/building-event-driven-batch-analytics-on-aws/

How does it actually work?

1. You upload your code to the serverless platform.
2. When triggered, the platform starts up a container running your code.
3. Your code executes, usually after being given information about the trigger event.
4. Your function completes, and yields control back to the framework
5. The framework might keep around the container to reduce latencies on subsequent requests

Benefits of Serverless Architecture

- Cost efficiency
  - Pay per request, instead of per server
- “Infinite scalability”
  - The unit of compute is more granular, so you can scale up and down dynamically with load
- Tightly integrated with other cloud services you’re probably already using
- Computation “glue” between other cloud services

Drawbacks of Serverless Architecture

- Higher first-time latencies
  - The framework needs to create and start a container for your code
- Limited environment options
  - You're restricted to the languages and tools that your framework/platform support
  - Using custom libraries (especially libraries that use dynamically-linked binaries) can be really difficult
- Assumption of "statelessness"
  - Cannot rely on anything being in memory, or on local disk
  - Have to rely on databases or other means for *all* persistence between requests

Serverless Platforms

AWS Lambda

- Pay by the 100ms
- Supports Node.js, Python, Java, and C#
- Heavily integrated with other AWS Services
  - API Gateway for web endpoints
  - DynamoDB Streams
  - S3 events

GCP Cloud Functions

- Metered per 100ms of compute usage
- Javascript Only (as of 2017)
- Integrations with:
  - Firebase
  - GCloud Pub/Sub (Similar to Kafka and AWS Kinesis)

Other Providers

- Azure Functions
- IBM Cloud Functions / OpenWhisk
- Kubeless
  - Kubernetes-based serverless framework

Serverless Frameworks

What does a serverless framework do?

- Helps you package up dependencies with your code
- Some provide a platform agnostic interface
  - Like Terraform. This can reduce “lock-in”
- Some provide testing tools so you can test your code
- Easier CLI tools for deploying your cloud functions

- Supports many languages (depending on platform)
  - Python, Node.js, Java, Scala, C#, F#, Groovy, Kotlin, PHP, Swift
- Supports many serverless platforms
  - AWS, GCP, Azure, IBM OpenWhisk, Kubeless, Spotinst, Webtasks
- Large plugin ecosystem
- Arguably the most popular serverless framework

Zappa

- Allows you to deploy traditional Python web applications on AWS Lambda
  - Django, Flask, and others work “out of the box”
- Supports event-driven execution
- CLI tools for deploying code to the cloud
- Works only with Python on AWS Lambda

Other Noteworthy Frameworks

- Apex
  - AWS Lambda specific; supports Golang and Closure, in addition to the Lambda-supported languages
- Chalice
  - AWS Lambda specific; designed to be used to create Python microservices