Zombie Apocalypse Workshop

Building Serverless Microservices

Presenter Name

Date

amazon web services

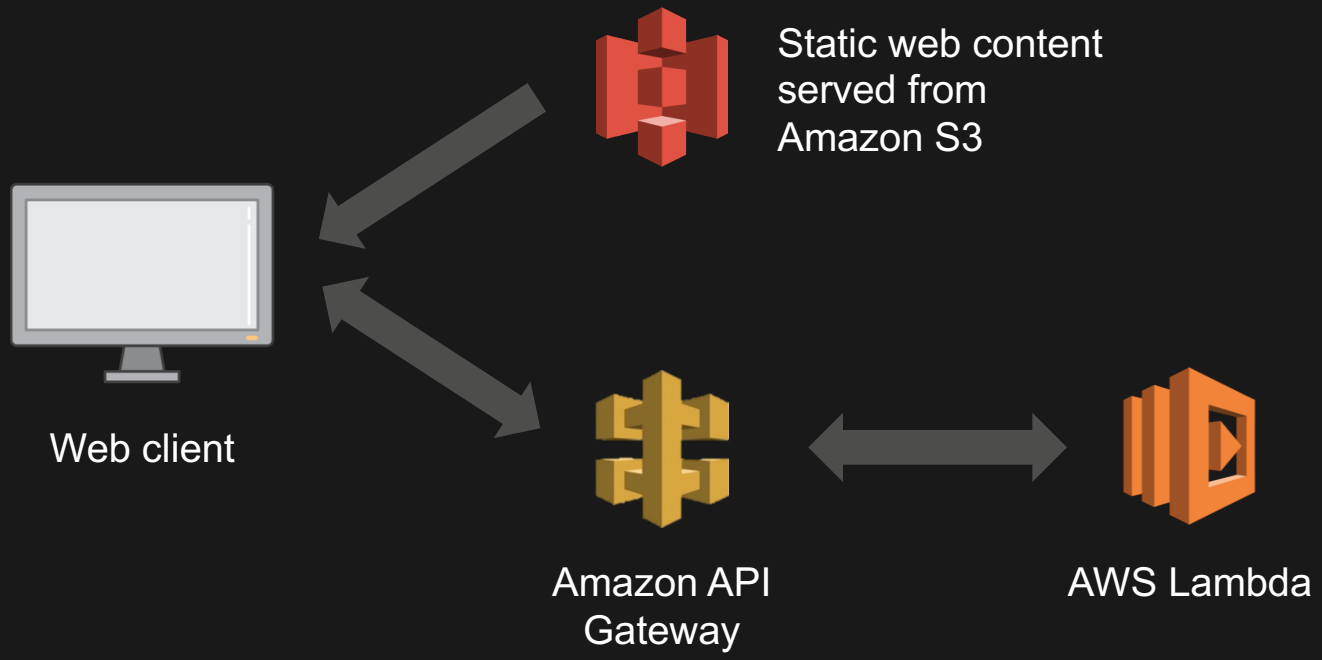
What to expect from this workshop

- Goal of serverless architectures
- Overview of AWS Lambda
- Overview of Amazon API Gateway
- Workshop Breakout – Time to build!
- Wrap-up/Q&A

Why serverless architectures?

- No servers to manage and scale
- Run at scale
- Respond quickly to events
- Only pay for compute time that you use
- Developer productivity

Serverless microservice architecture



AWS Lambda

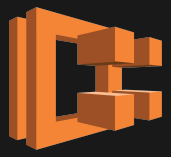
Your feedback helped create Lambda!

- No direct responsibility for infrastructure resources
- Quick and simple deployments
- Highly available and scalable apps with zero administration
- Costs that are closely aligned to application usage

AWS compute offerings

Amazon EC2

Resizable virtual servers in the cloud



Amazon ECS

Container management service for running Docker on EC2

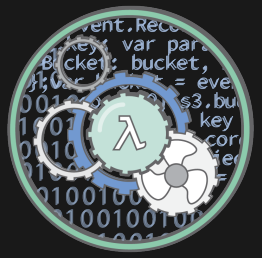
AWS Lambda

Serverless compute, run code in response to events

Benefits of using Lambda

1

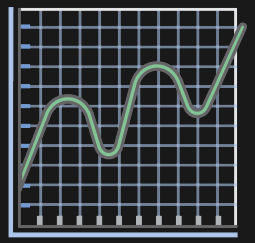
No Servers to Manage



Lambda automatically runs your code without requiring you to provision or manage servers. Just write the code and upload it to Lambda.

2

Continuous Scaling



Lambda automatically scales your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.

3

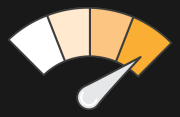
Subsecond Metering

With Lambda, you are charged for every 100 ms your code executes and the number of times your code is triggered. You don’t pay anything when your code isn’t running.

AWS Lambda – How it works

Bring your own code

- Node.js, Java, Python
- Java = Any JVM based language such as Scala, Clojure, etc.
- Bring your own libraries



Simple resource model

- Select memory from 128MB to 1.5GB in 64MB steps
- CPU & Network allocated proportionately to RAM
- Reports actual usage

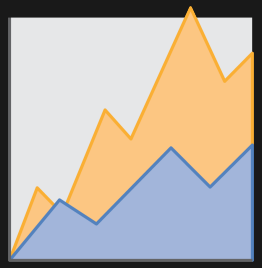
Flexible invocation paths

- Event or RequestResponse invoke options
- Existing integrations with various AWS services

Fine grained permissions

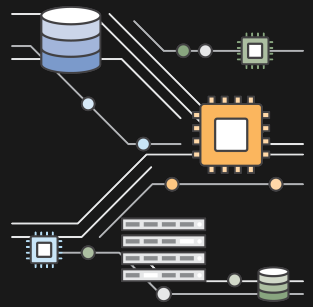
- Uses IAM role for Lambda execution permissions
- Uses Resource policy for AWS event sources

AWS Lambda – Use Cases



Data Processing
Execute code in response to changes in data, shifts in system state, or actions by users

Backends
Execute backend logic to handle requests for web, mobile, IoT, and 3rd APIs



Control Systems
Customize responses and response workflows to state and data changes within AWS

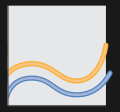
Amazon API Gateway

Your feedback

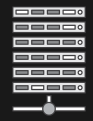
Managing multiple versions and stages of an API is difficult

Monitoring third-party developers' access is time consuming

Access authorization is a challenge



Traffic spikes create an operational burden



What if I don't want servers at all?

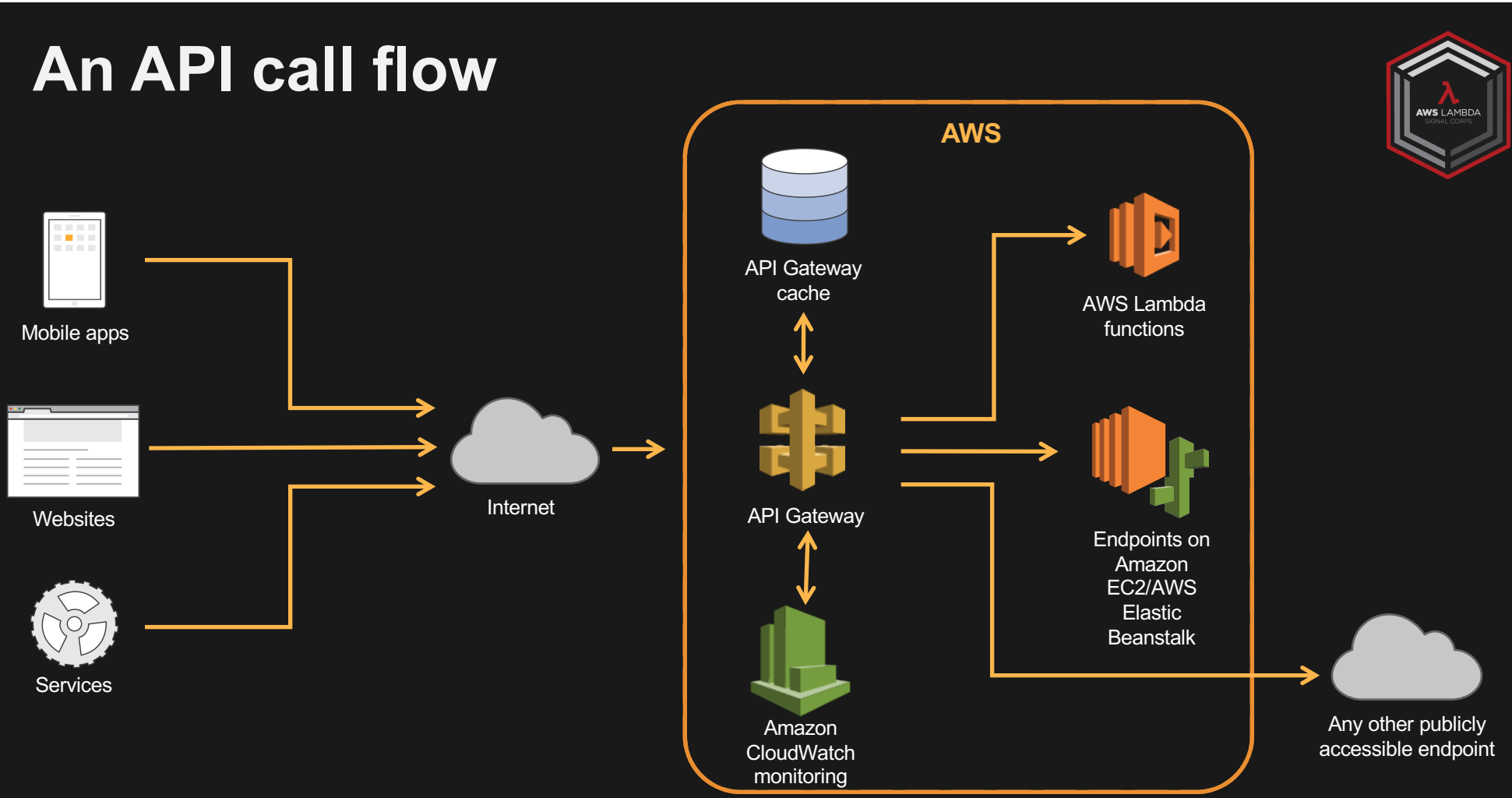
API Gateway - Capabilities

- Host multiple versions and stages of your APIs
- Create and distribute API keys to developers
- Leverage signature version 4 to authorize access to APIs
- Throttle and monitor requests to protect your backend
- Utilize Lambda as a backend

Benefits of API Gateway

- Managed cache to store API responses
- Reduced latency and distributed denial of service (DDoS) protection through Amazon CloudFront
- SDK generation for iOS, Android, and JavaScript
- Swagger support
- Request and response data transformation

An API call flow

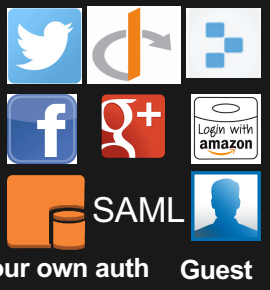


Amazon Cognito

Amazon Cognito Identity

Cognito User Pools

You can easily and securely add sign-up and sign-in functionality to your mobile and web apps with a fully-managed service that scales to support 100s of millions of users.



Federated User Identities

Your users can sign-in through social identity providers such as Facebook, Twitter and SAML providers and you can control access to AWS resources from your app.

Amazon Cognito User Pools

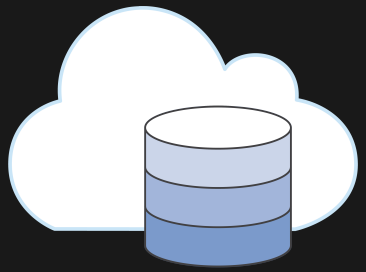
1

Serverless Authentication and User Management

Add user sign-up and sign-in easily to your mobile and web apps without worrying about server infrastructure

2

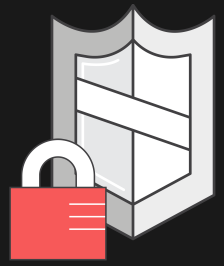
Managed User Directory



A simple, secure, low-cost, and fully managed service to create and maintain a user directory that scales to 100s of millions of users

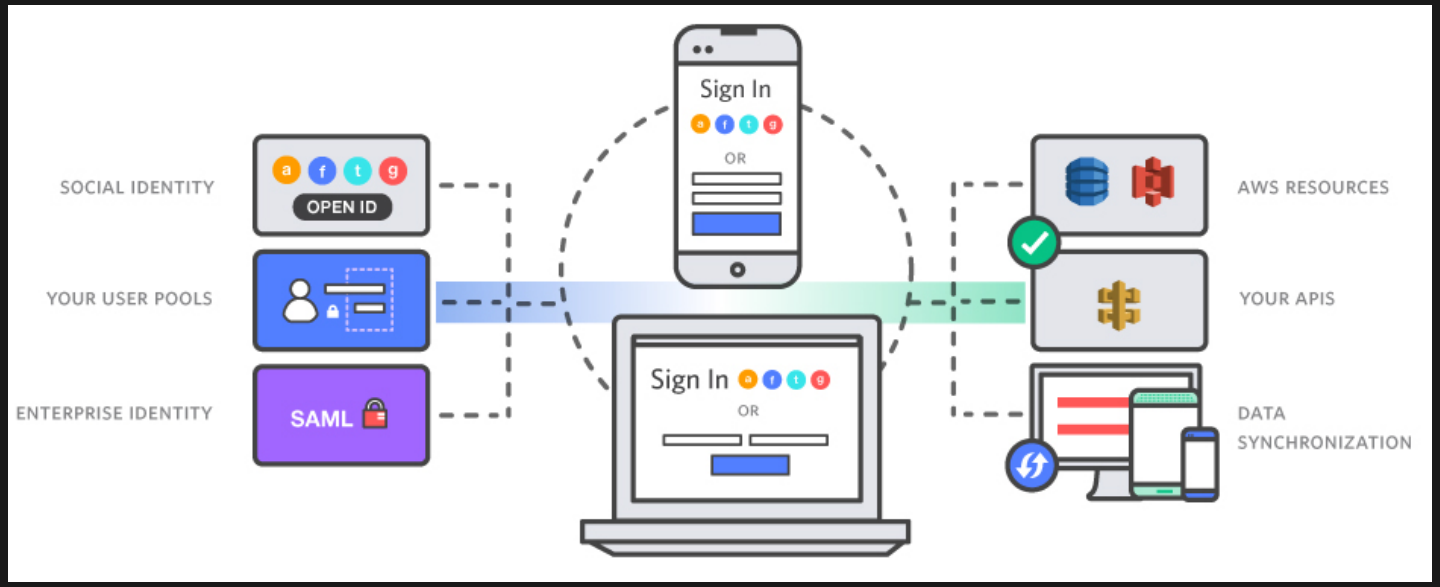
3

Enhanced Security Features



Verify phone numbers and email addresses and offer multi-factor authentication

Comprehensive Support for Identity Use Cases

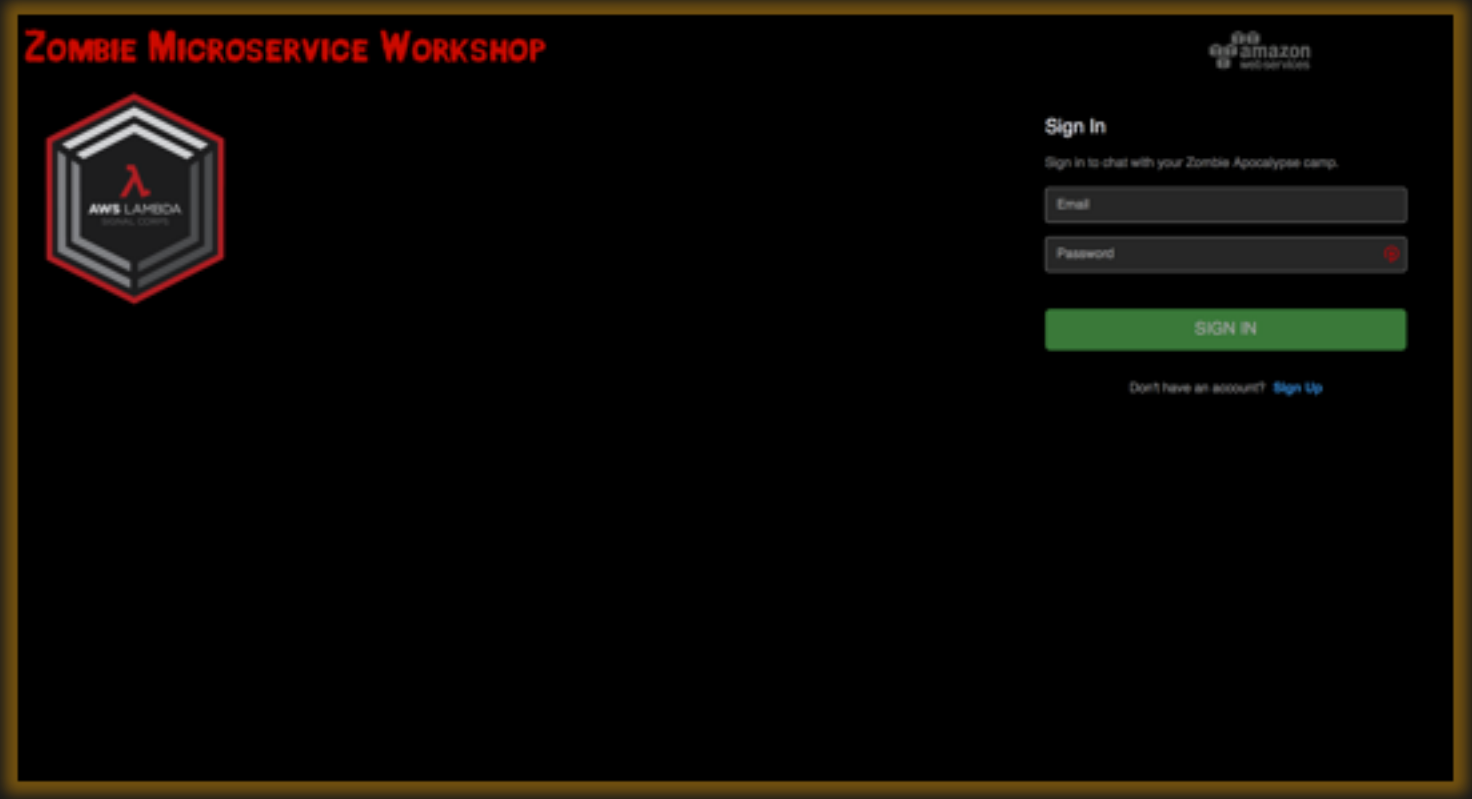


The Zombie Apocalypse Survival

ZOMBIES!

Zombies have taken over major metropolitan areas. The AWS Lambda Signal Corps has built a communications system to connect remaining survivors. Come learn how AWS Lambda provides a platform for building event-driven microservices, all without the need to provision, manage, and scale servers. In this workshop, we will introduce the basics of using AWS Lambda to run code in response to events from Amazon DynamoDB, S3, and API Gateway. You'll work within a team to build a secure, scalable, fault-tolerant chat service with global reach from scratch using blueprints provided by us. Unfortunately, the blueprints provided only describe a very rudimentary communications system (the engineers of the project got mysteriously ill). We are looking to you and your team to add additional real-time life saving features (e.g., food cache locations, zombie motion detectors, undead counters) to the chat platform using Lambda functions.

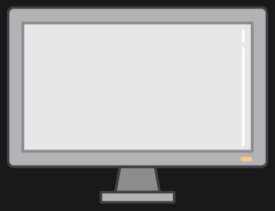
Engineers got this far…



High-level Zombie Chat Architecture

Amazon Cognito
User Authentication

Amazon S3
Static Content/Chat Web App



Web client
End user

Amazon API Gateway
REST Interface

AWS Lambda
Backend Logic

Amazon DynamoDB
Messages Data Store

Zombie Chat implementation

S3

A new S3 bucket with single-page HTML5 web app

API Gateway

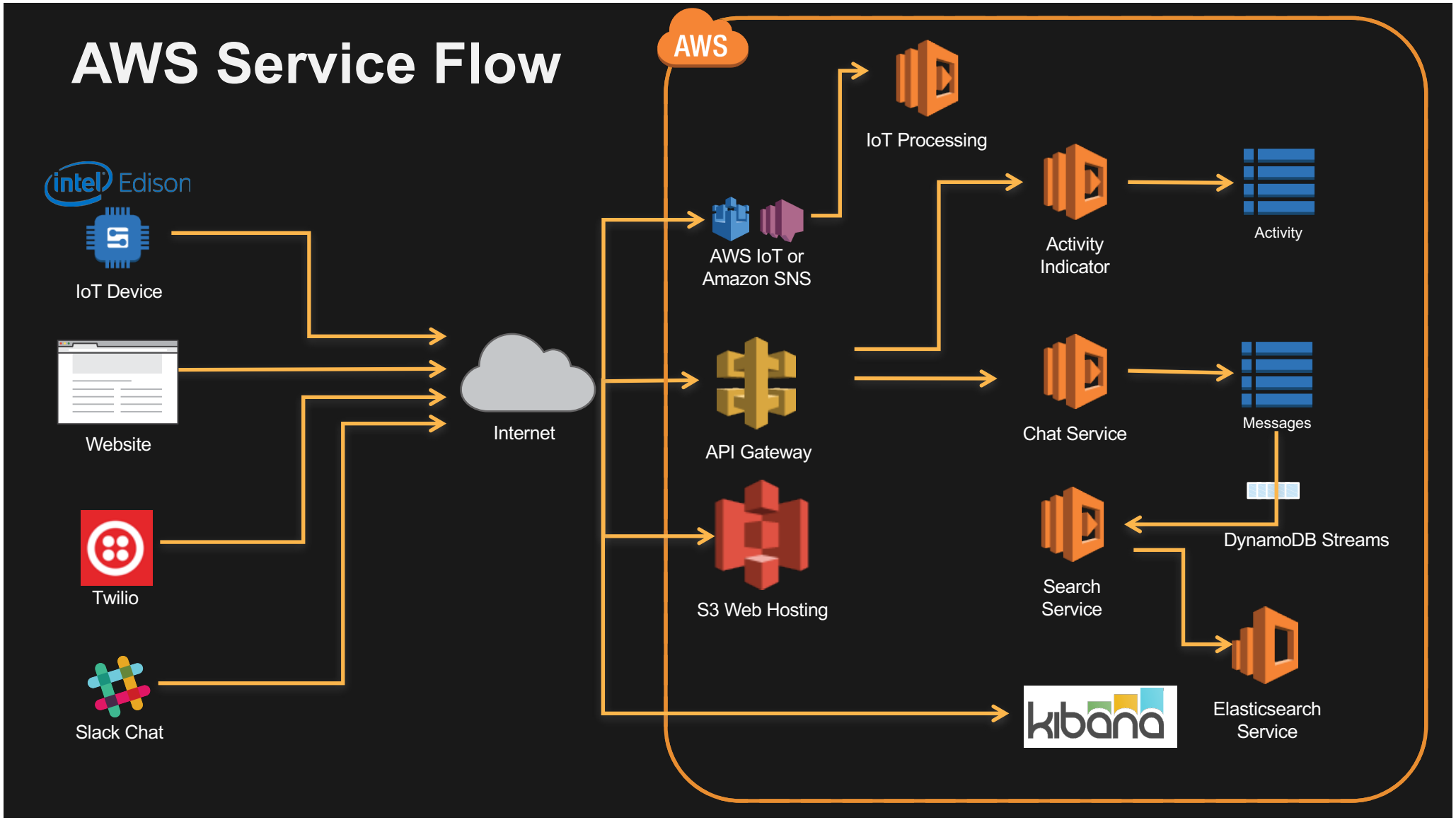
/zombie/messages API with GET and POST methods

Lambda

Functions GetMessagesFromDynamoDB and WriteMessagesToDynamoDB

DynamoDB

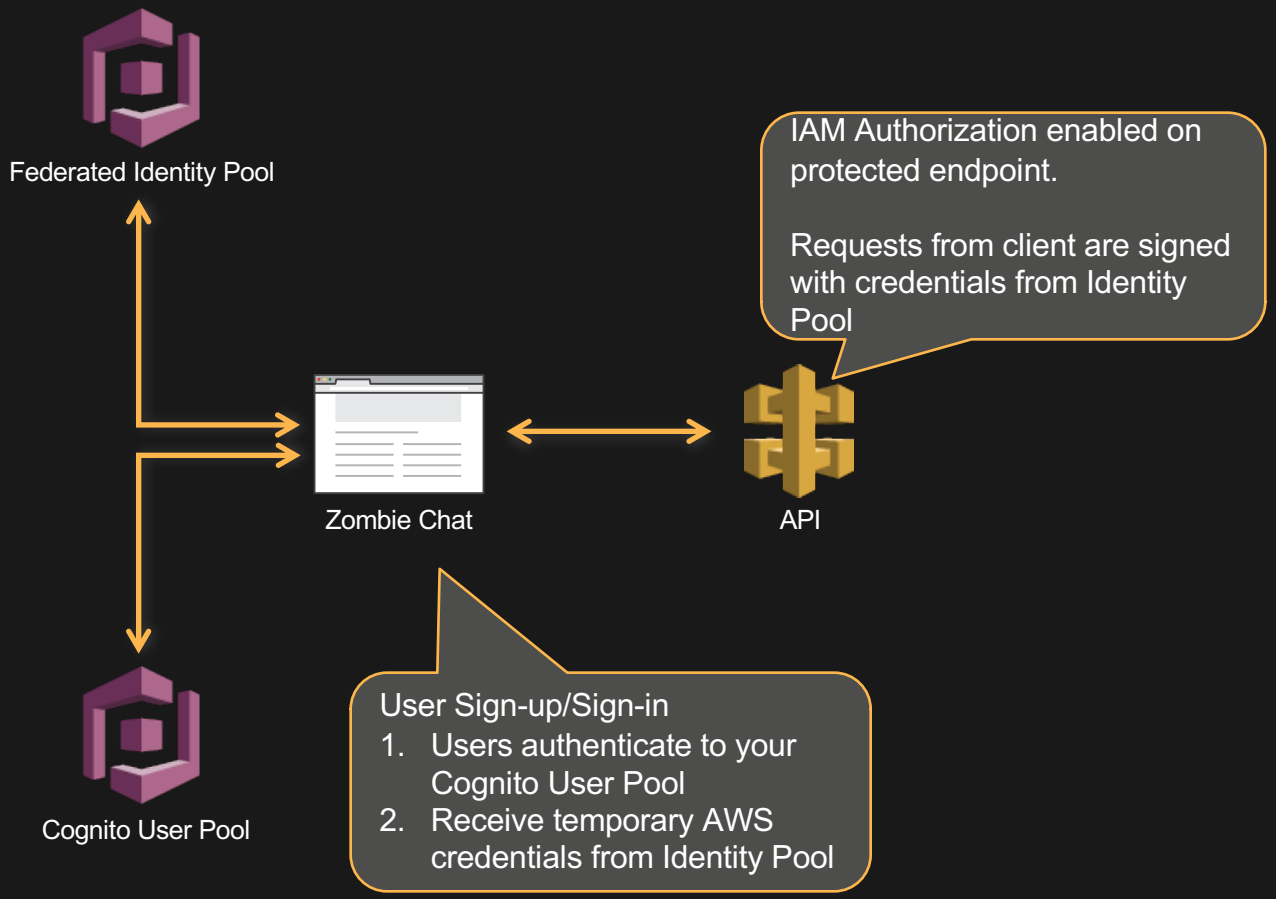
A ‘messages’ table to track *channel, timestamp, message, and name*


AWS Service Flow
AWS
IoT Processing
intel Edison
IoT Device
AWS IoT or
Amazon SNS
Activity
Indicator
Activity
Website
Internet
API Gateway
Chat Service
Messages
DynamoDB Streams
Twilio
S3 Web Hosting
Search
Service
Slack Chat
kibana
Elasticsearch
Service

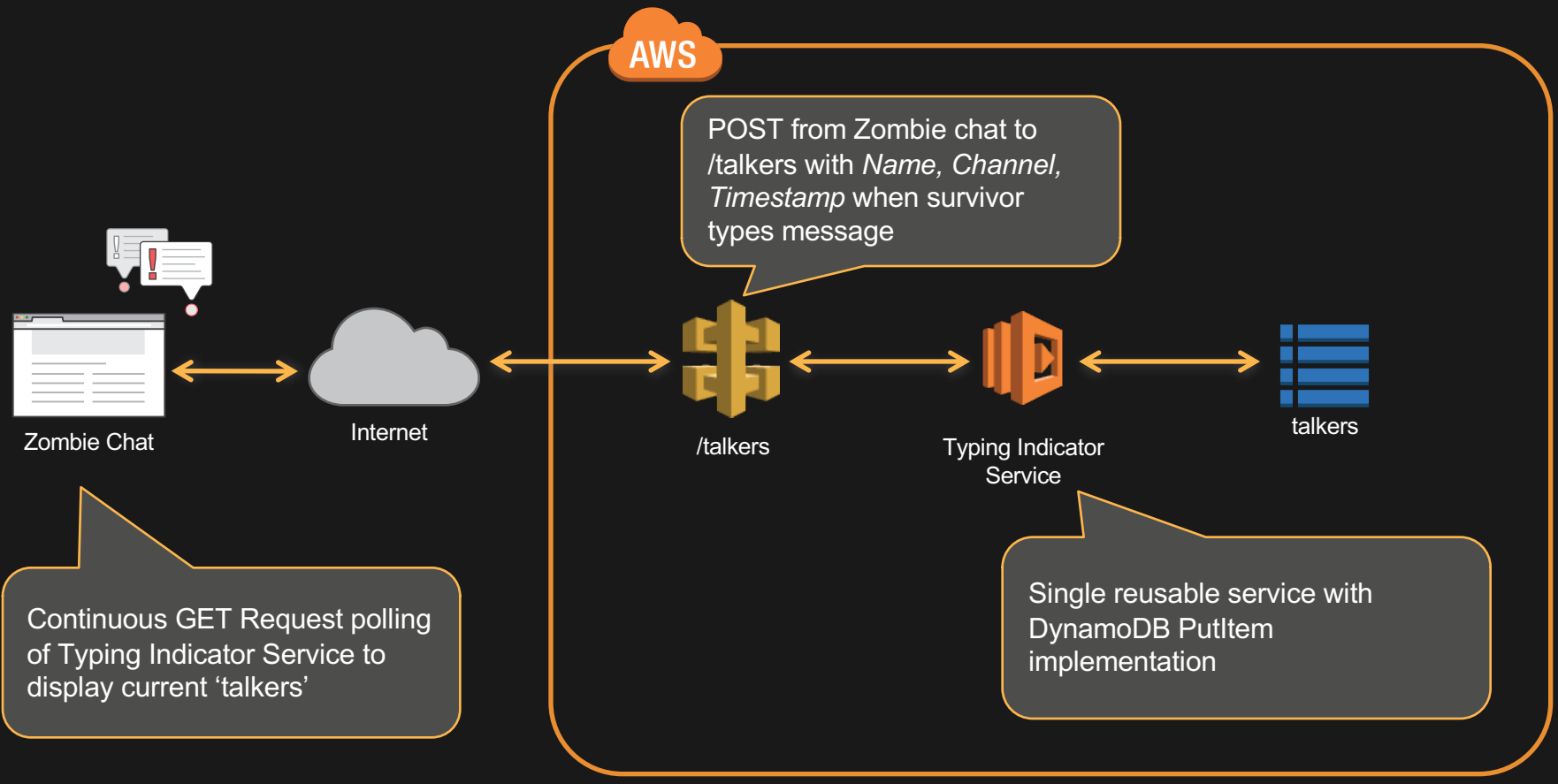
Lab Architectures

What you'll build today!

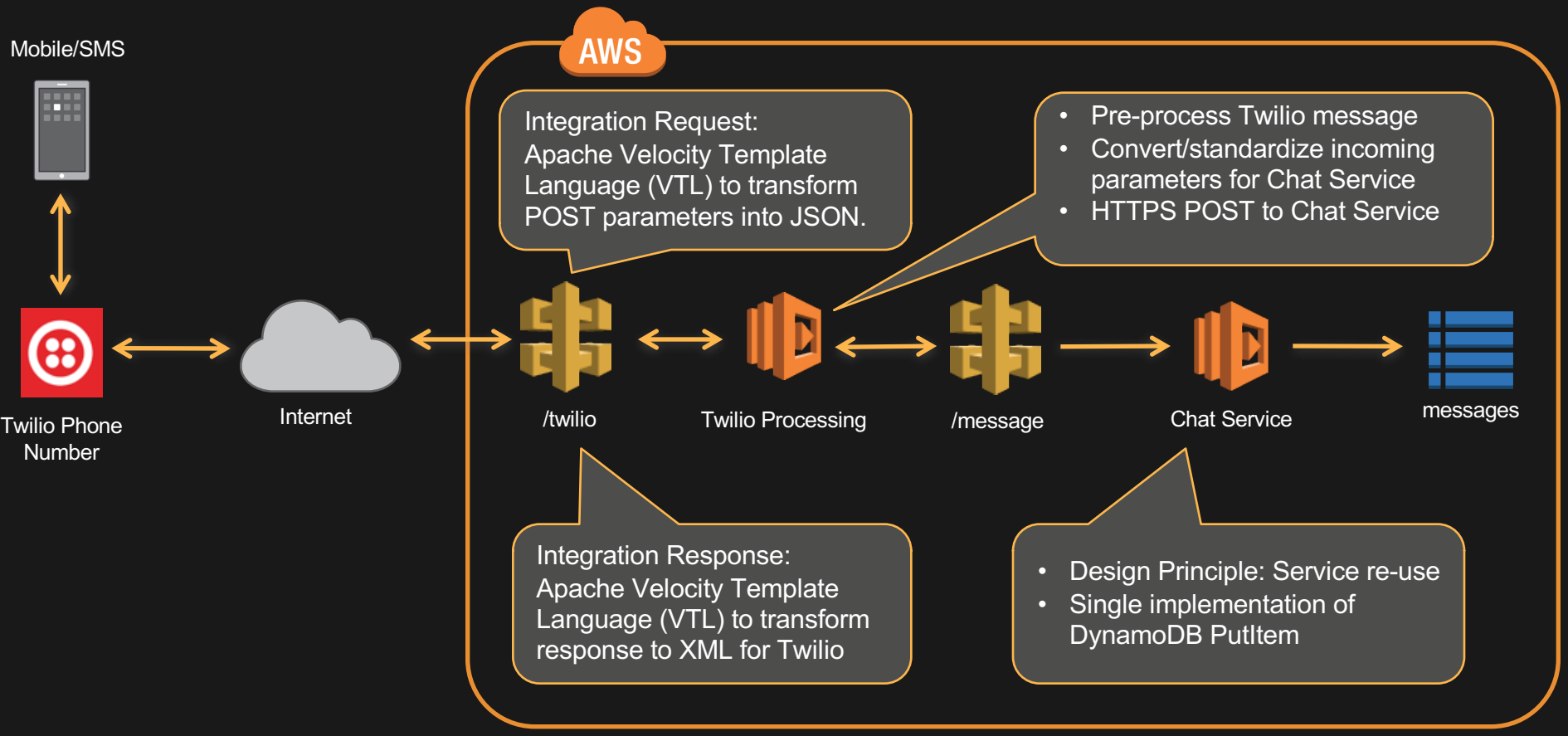
Setup Authentication w/ Cognito User Pools



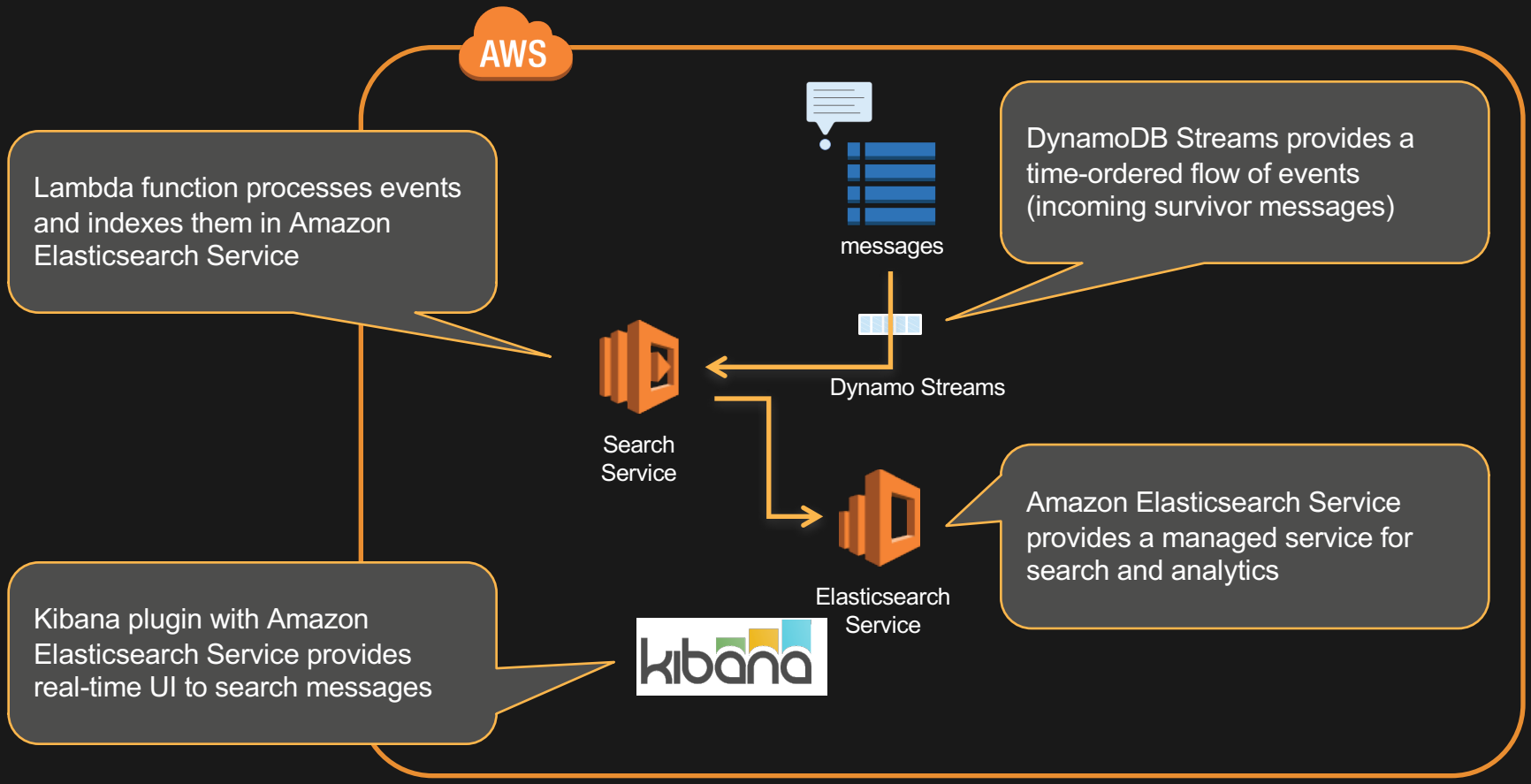
Lab 1: Typing Indicator



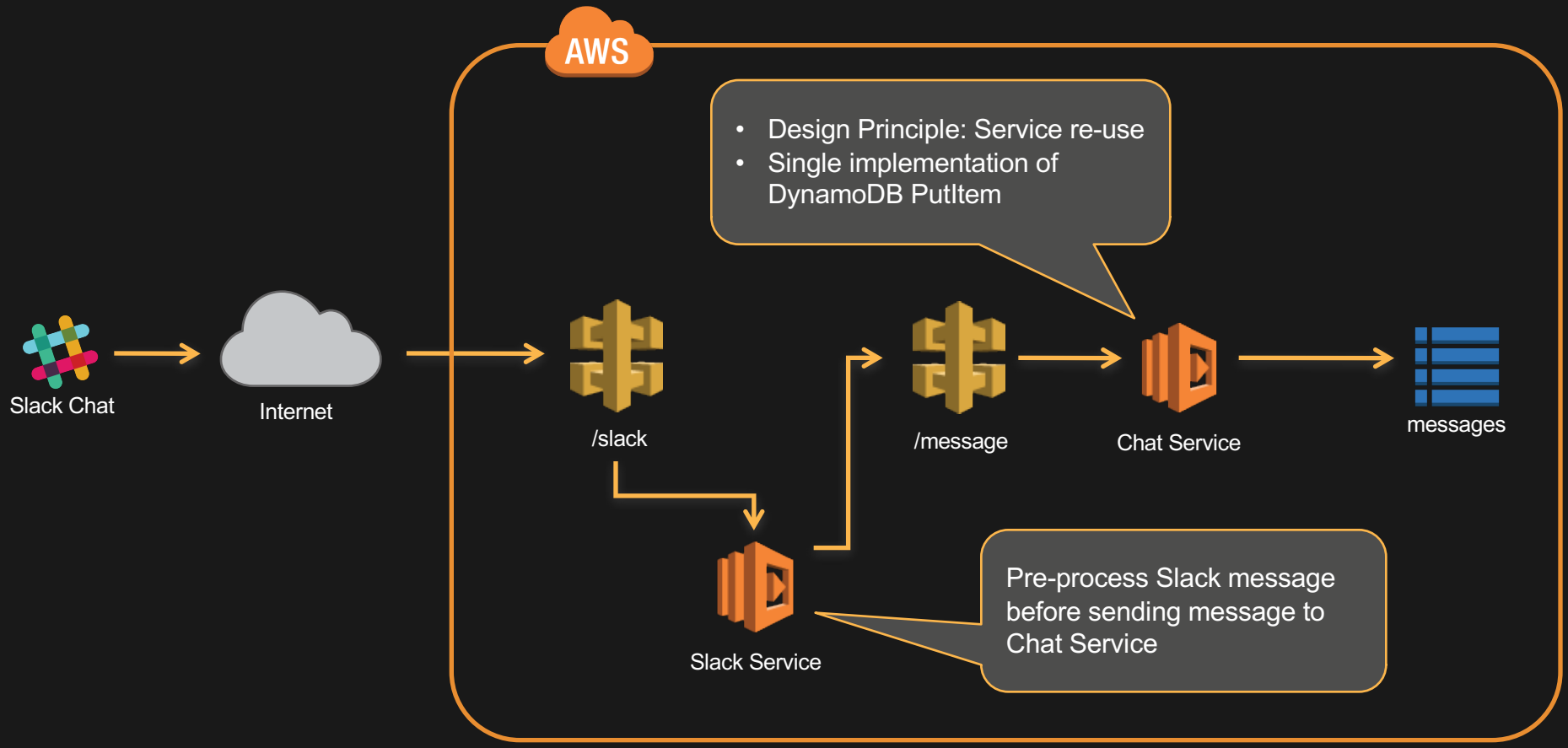
Lab 2: SMS Integration with Twilio



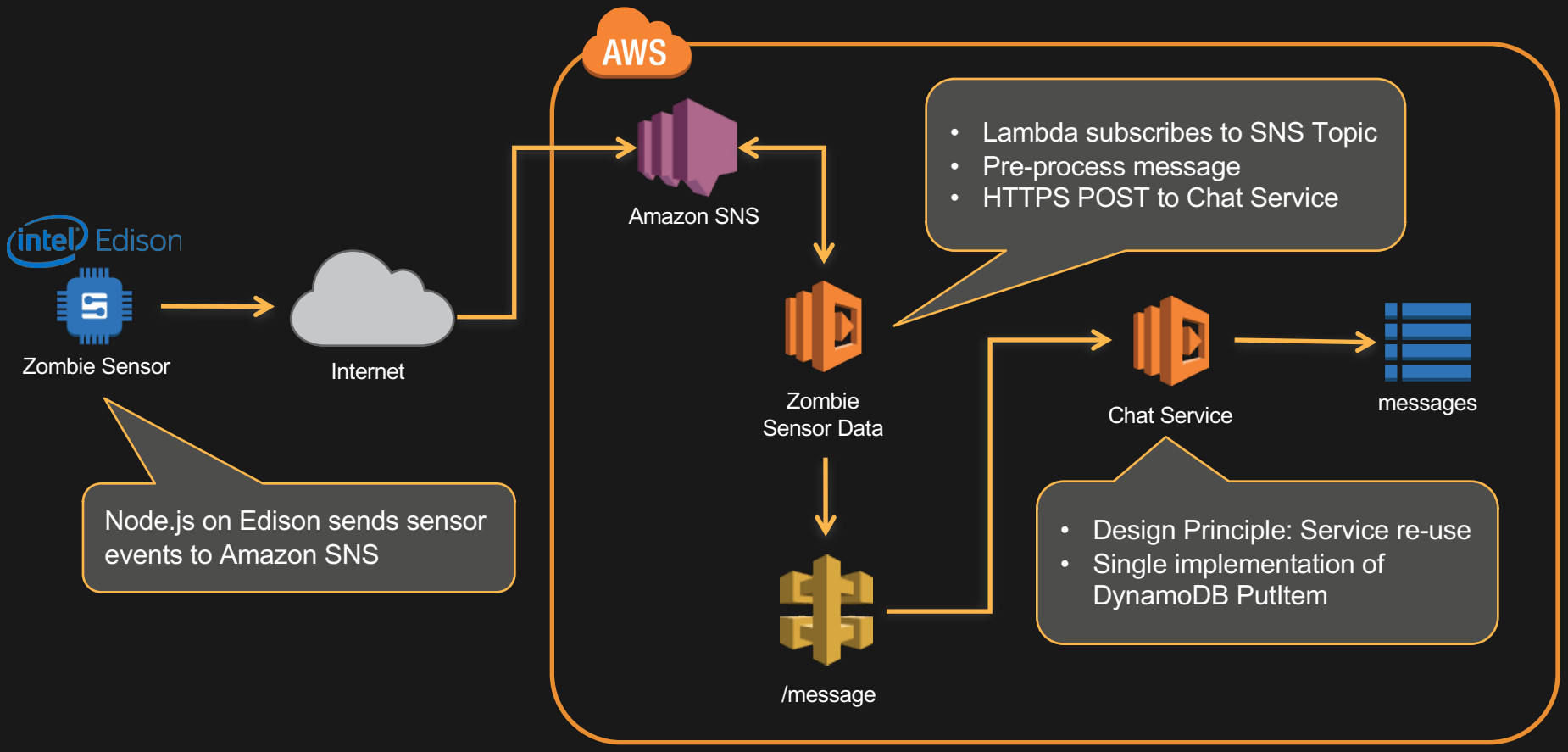
Lab 3: Search with Elasticsearch Service



Lab 4: Send Messages from Slack



Lab 5: Zombie Sensor with Intel Edison



Your challenge

Base Challenge

1. Implement chat add-ons with the steps from the Lab Guide

Extra Credit Challenges

1. Implement channel functionality for different chat rooms/private chats
2. Data store for weapons/food caches & bot to notify survivors of cache levels
3. Build your own challenges and share your design with us!

Steps to get started

- Break into groups (less than 5 people) or work solo!.
- Select a leader to launch the CloudFormation Stack.
- Complete add-ons from Lab Guide.
- Decide on other challenges you'll build!
- Share your designs with fellow survivors!

Workshop available at:

https://github.com/awslabs/aws-lambda-zombie-workshop

Thank You!

Run Apps for Pennies!

Cost estimate to run this 3 hour workshop!

Estimated Cost ~ $0.6555

- Lambda: **FREE**
  - 1st 1m requests are free each month! Duration pricing will be sub-1penny!
- DynamoDB: **$0.0585**
  - $0.0065/hr for every 10 read units provisioned - 75 units provisioned/hr
  - $0.0065/hr for every 50 write units provisioned – 75 units provisioned/hr
  - DynamoDB Streams: 2.5m reads free per month
- ElasticSearch Service: **$0.282**
  - M3.medium with instance storage - $0.094/hr
- API Gateway: **$0.035**
  - $3.50/million API calls. Assume 10,000 calls made per hour during lab
- CloudWatch Logs: **$0.25**
  - $0.50 per GB/month ingested. Super high end estimate of 500MB of log data during workshop
- S3: **$0.03**
  - $0.03 per GB for first 1TB a month.
- Data Transfer: **FREE**
  - First 1GB/month out is free. Data transfer in is free!