Zombie Apocalypse Workshop

Building Serverless Microservices

aws manufacturing

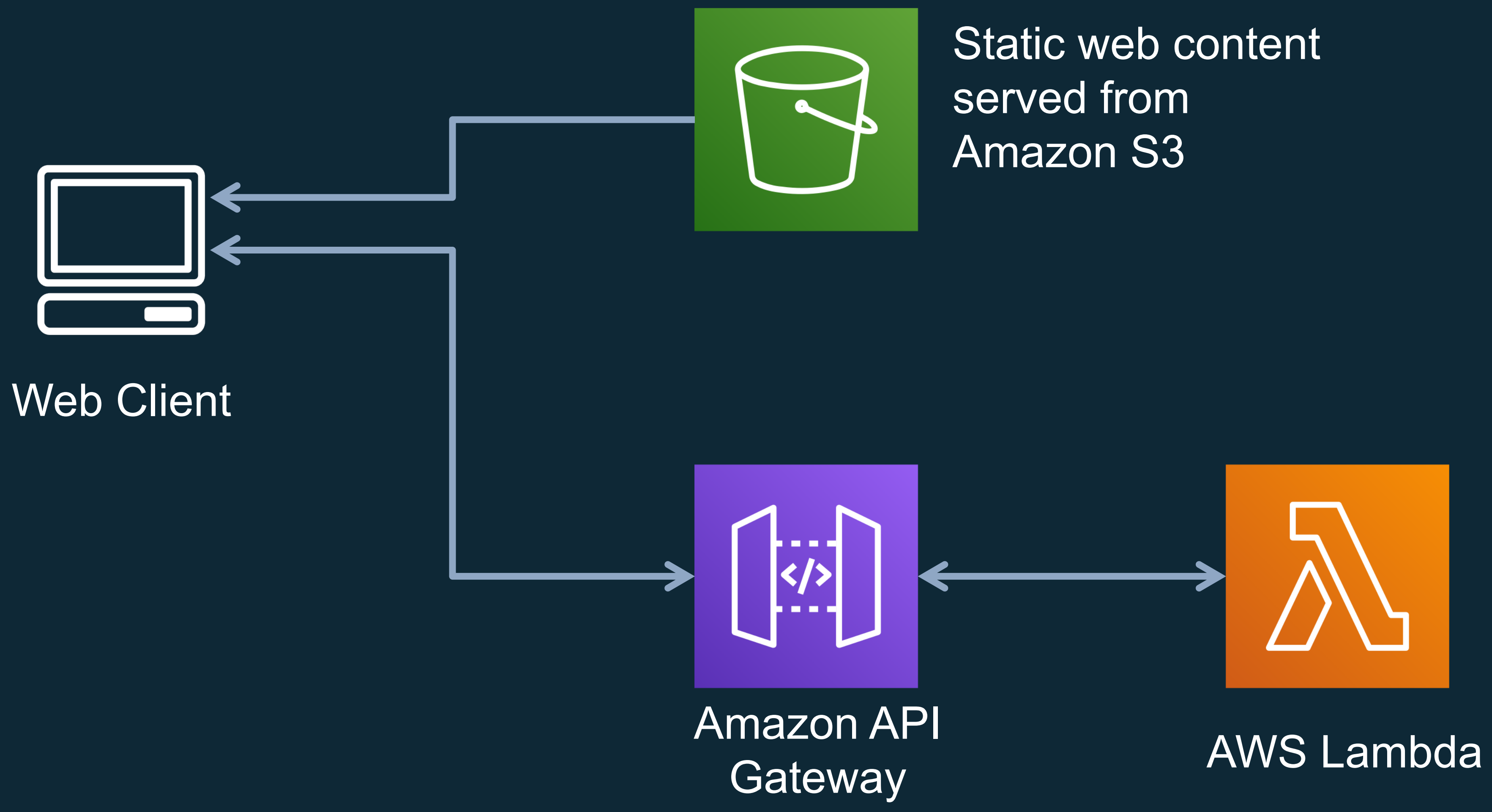
What to expect from this workshop

- 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



Static web content served from Amazon S3

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

Amazon Elastic Kubernetes Service
Fully managed K8s

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, Ruby, Go, .NET
- Java = Any JVM based language such as Scala, Clojure, etc.
- Bring your own libraries

Simple resource model

- Select memory from 128MB to 3GB 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

aws manufacturing

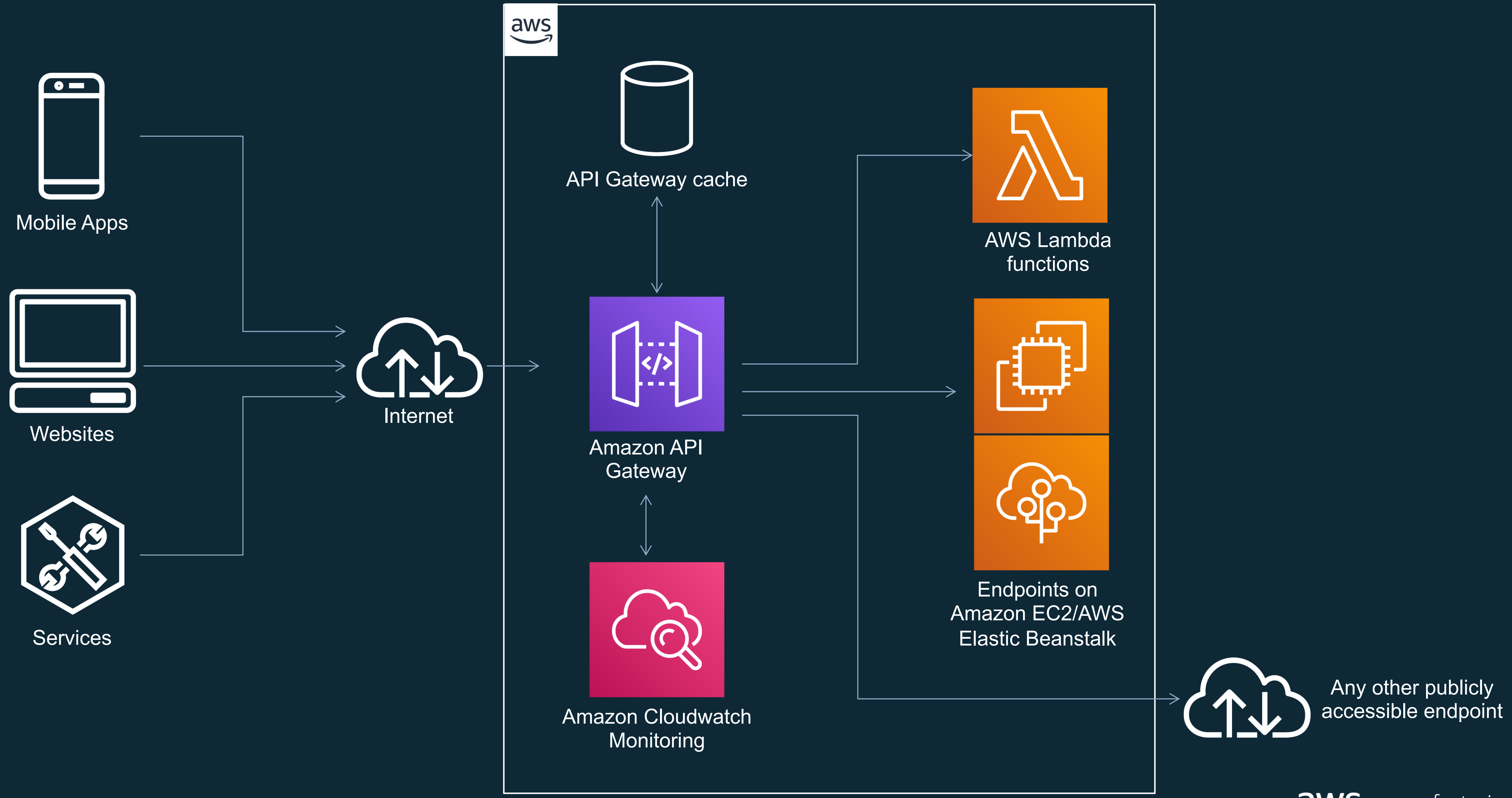
API Gateway - Capabilities

- 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



aws
Mobile Apps
Websites
Services
Internet
API Gateway cache
Amazon API Gateway
Amazon Cloudwatch Monitoring
AWS Lambda functions
Endpoints on Amazon EC2/AWS Elastic Beanstalk
Any other publicly accessible endpoint

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 & 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

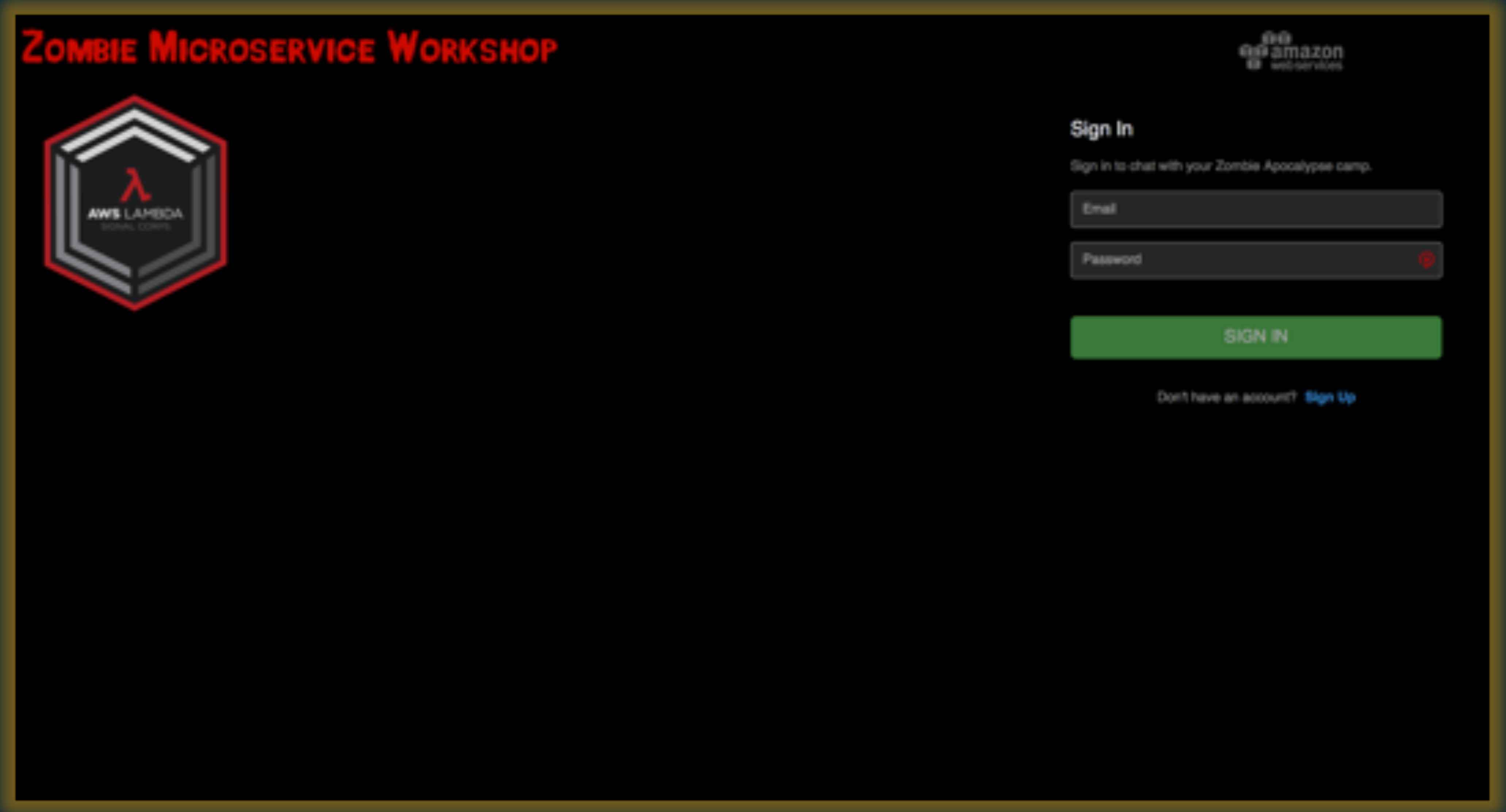
Verify phone numbers and email addresses and offer multi-factor authentication

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…



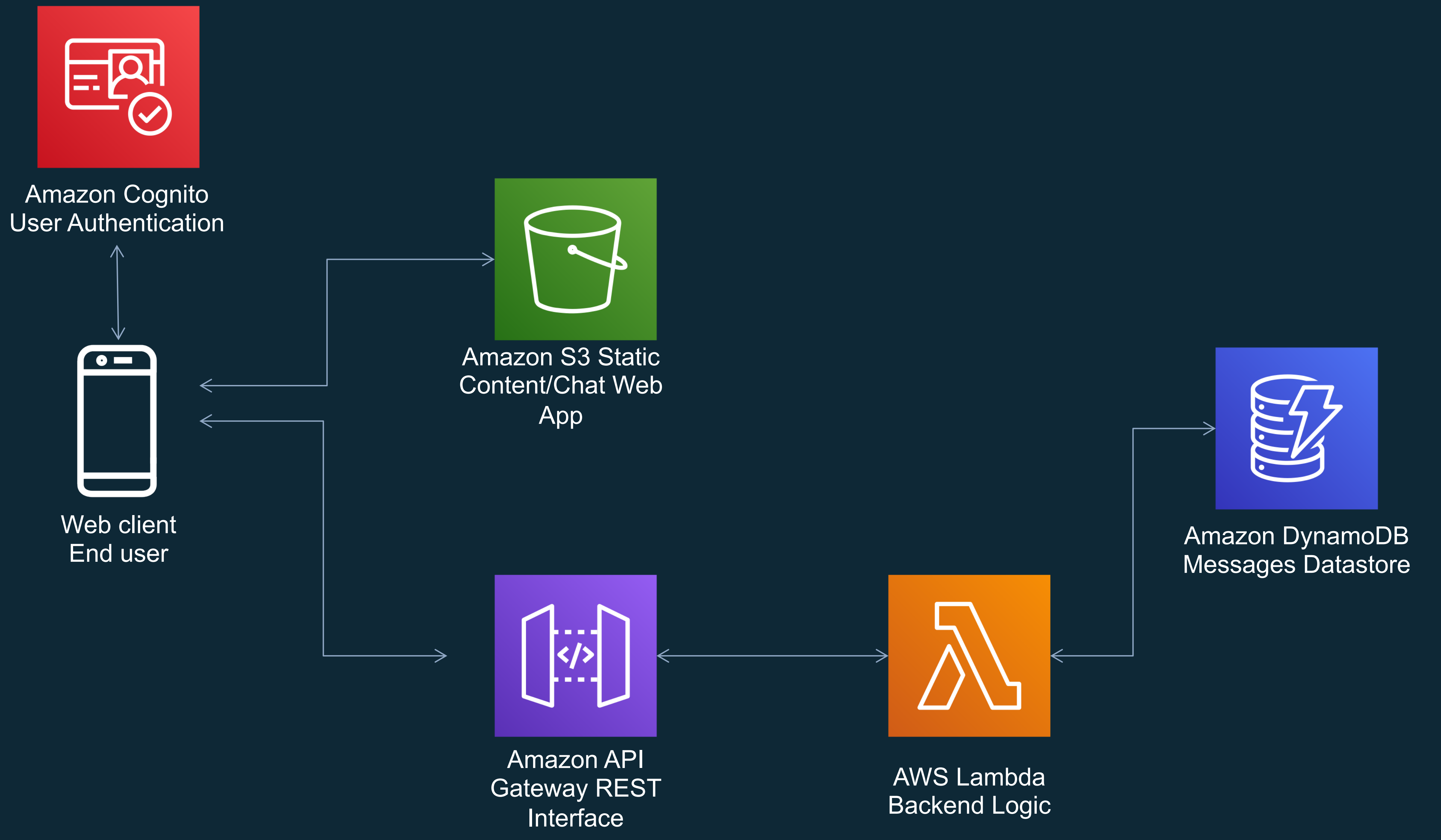
Zombie Microservice Workshop

Sign In

Sign in to chat with your Zombie Apocalypse camp.

Don't have an account? Sign Up

High-level Zombie Chat Architecture



Amazon Cognito
User Authentication

Amazon S3 Static
Content/Chat Web
App

Web client
End user

Amazon DynamoDB
Messages Datastore

Amazon API
Gateway REST
Interface

AWS Lambda
Backend Logic

Zombie Chat implementation

S3

A new S3 bucket with single-page HTML5 web app

API Gateway

/zombie/messages API with GET and POST methods

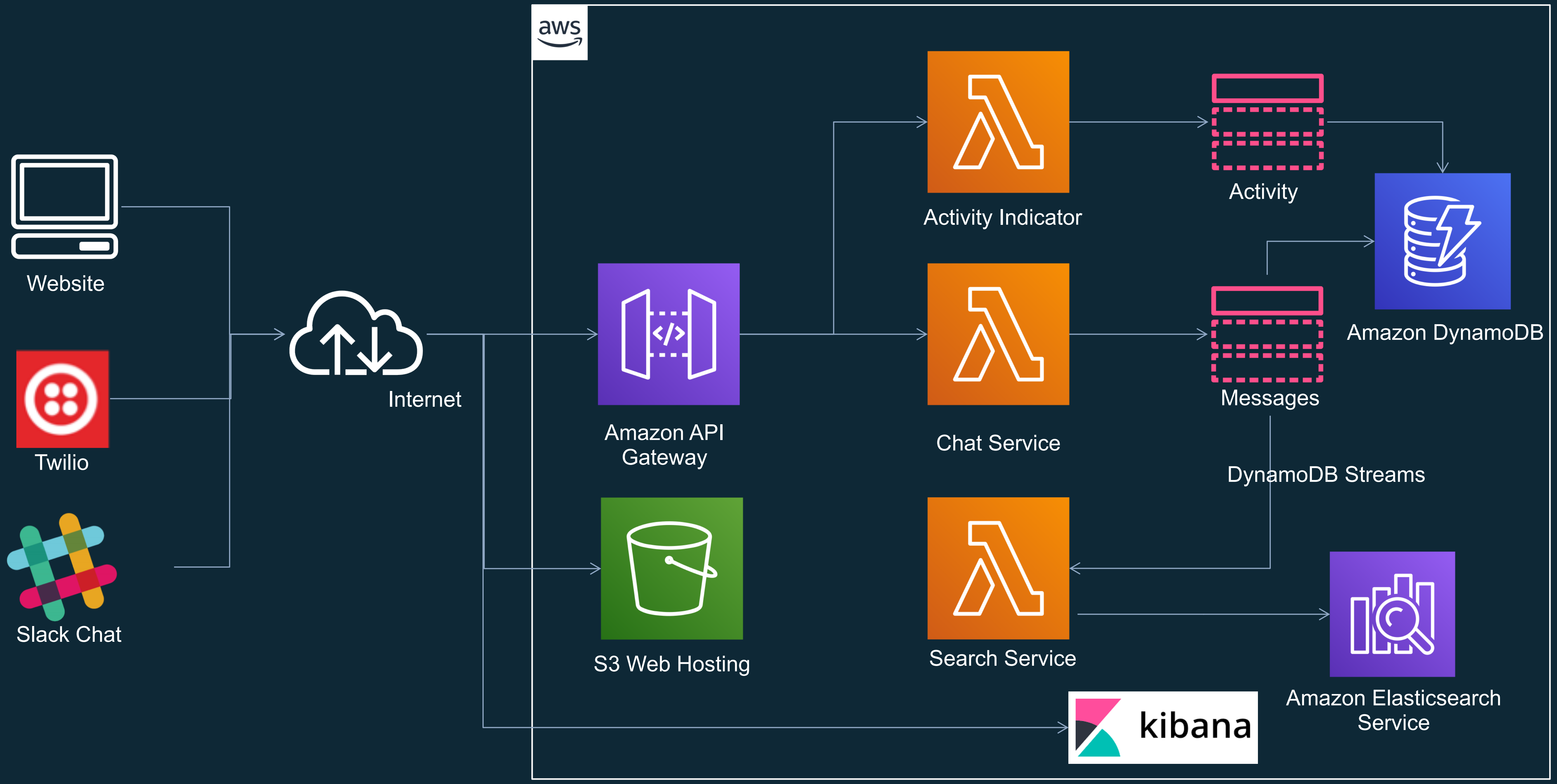
Lambda

Functions GetMessagesFromDynamoDB and WriteMessagesToDynamoDB

DynamoDB

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

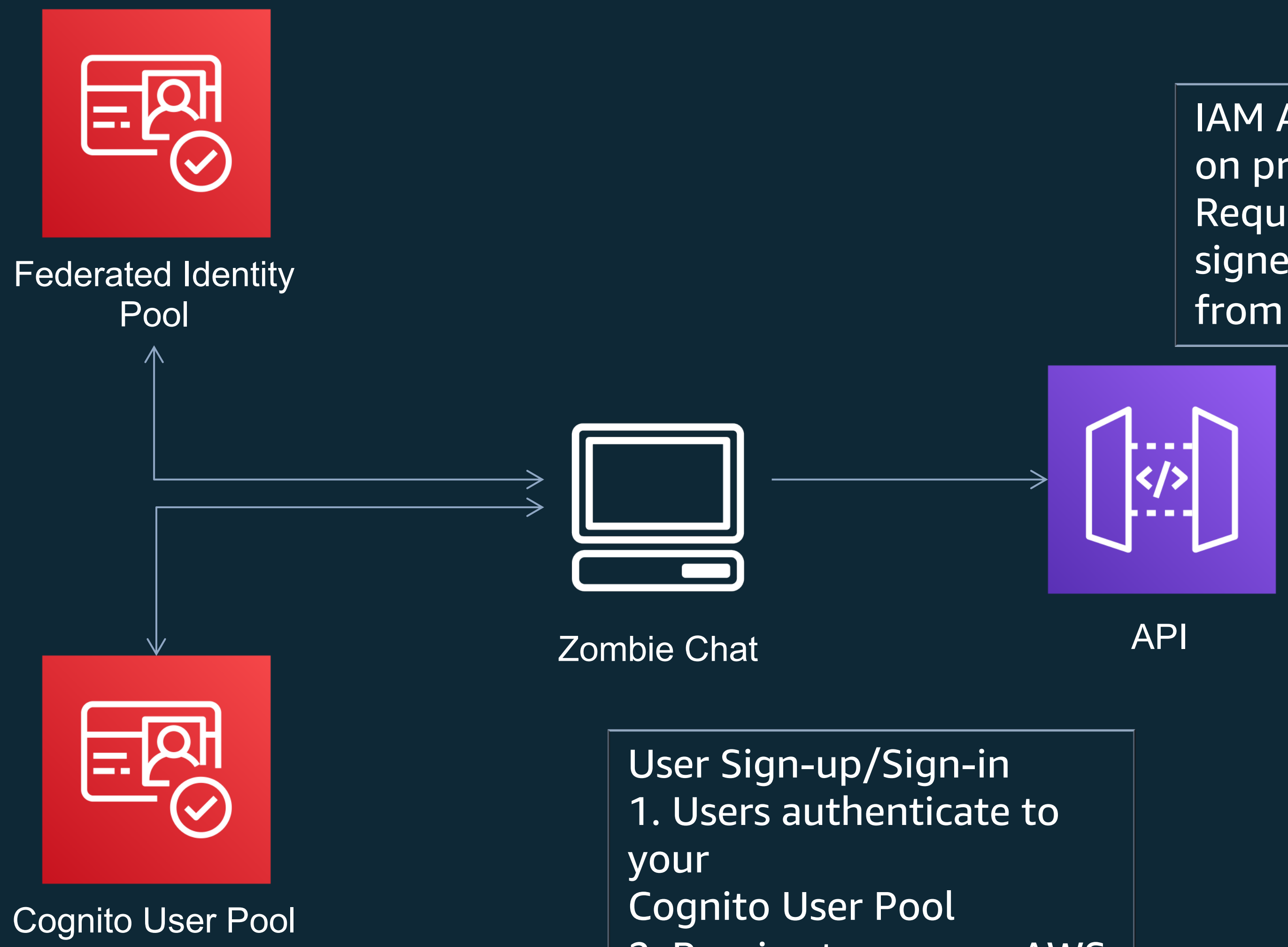
AWS Service Flow



Lab Architectures

What you’ll build today!

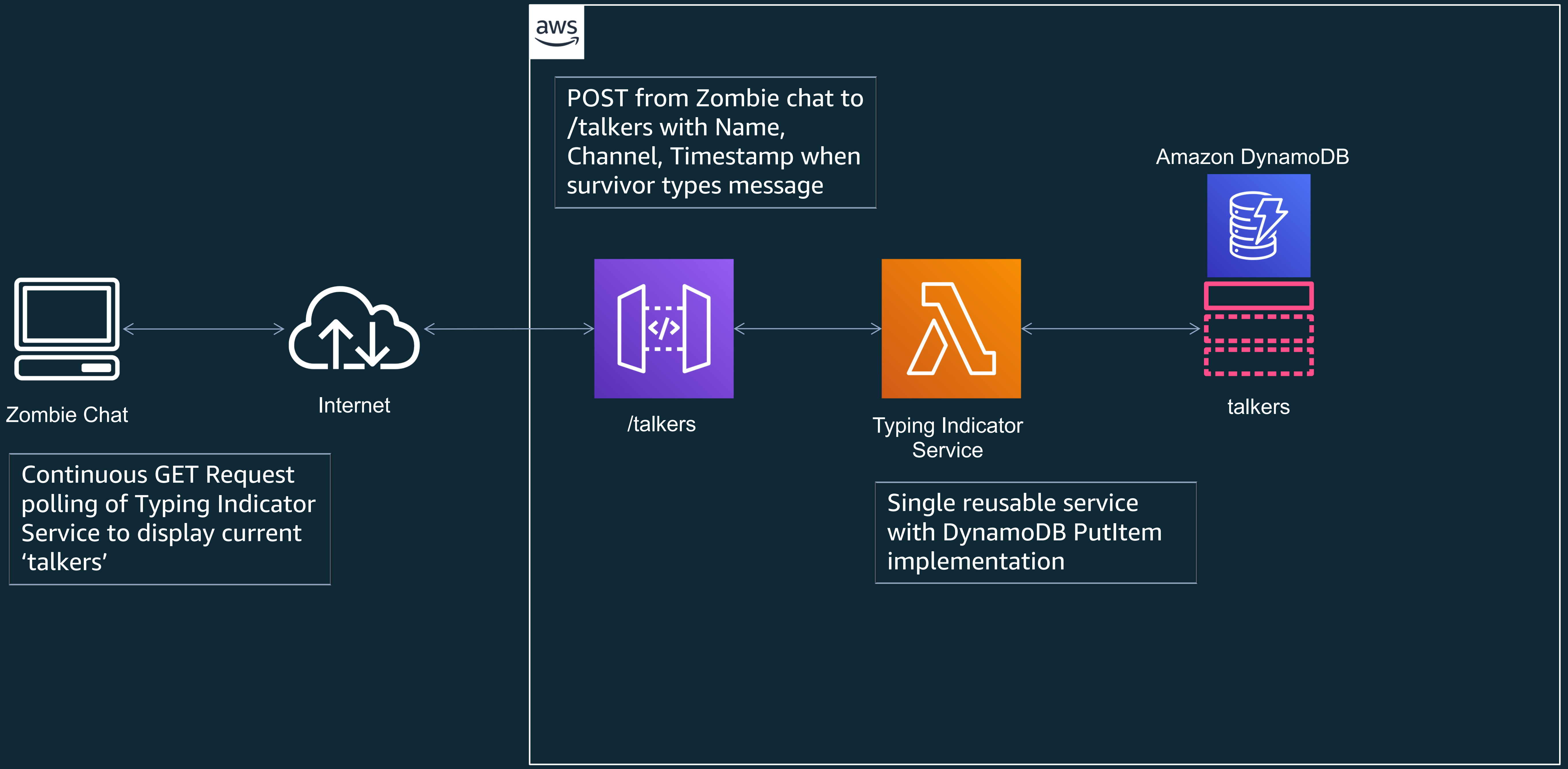
Setup Authentication w/ Cognito User Pools



IAM Authorization enabled on protected endpoint. Requests from client are signed with credentials from Identity Pool

User Sign-up/Sign-in
1. Users authenticate to your
Cognito User Pool
2. Receive temporary AWS credentials from Identity Pool

Lab 1: Typing Indicator

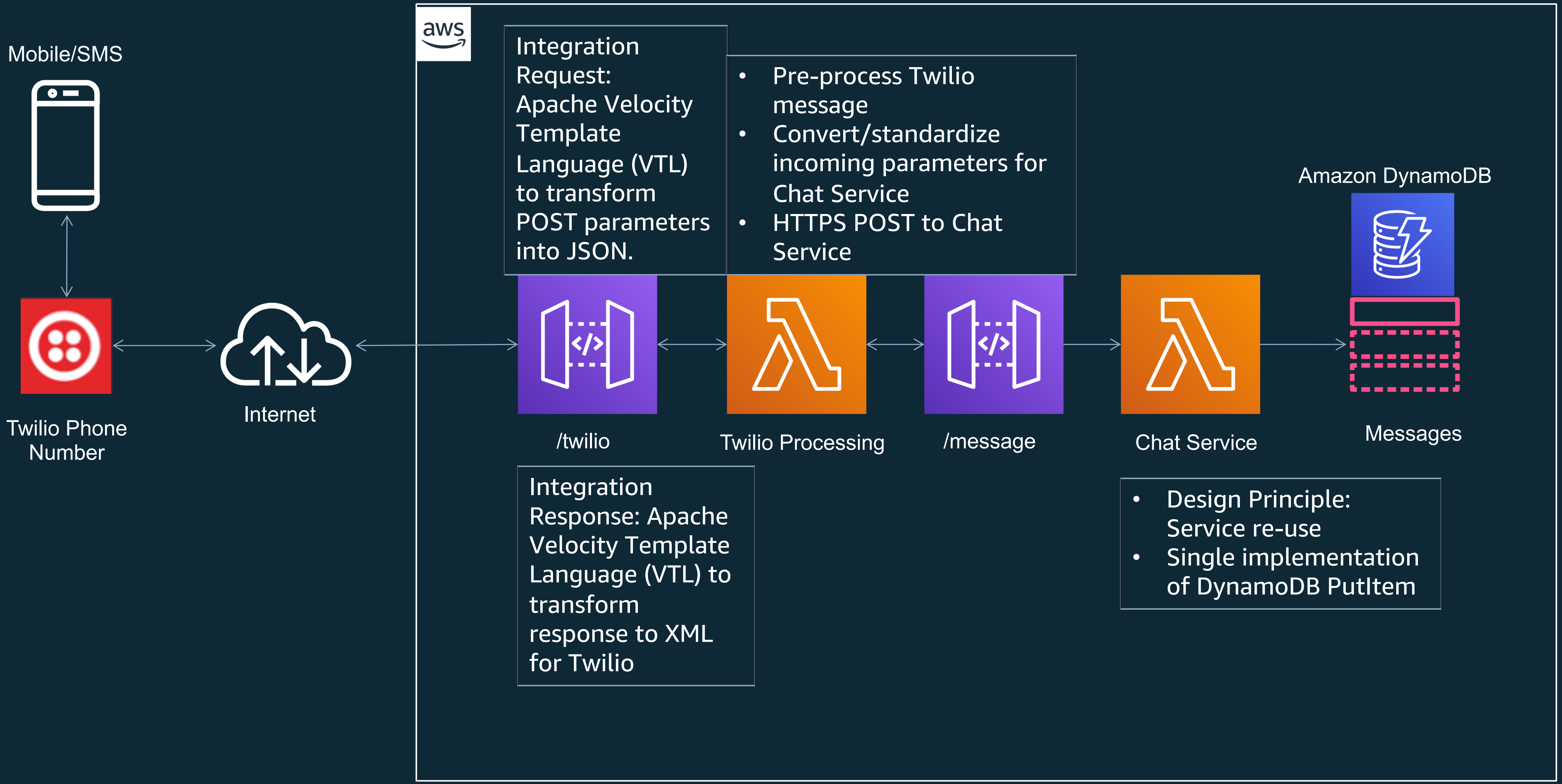


POST from Zombie chat to /talkers with Name, Channel, Timestamp when survivor types message

Continuous GET Request polling of Typing Indicator Service to display current 'talkers'

Single reusable service with DynamoDB PutItem implementation

Lab 2: SMS Integration with Twilio



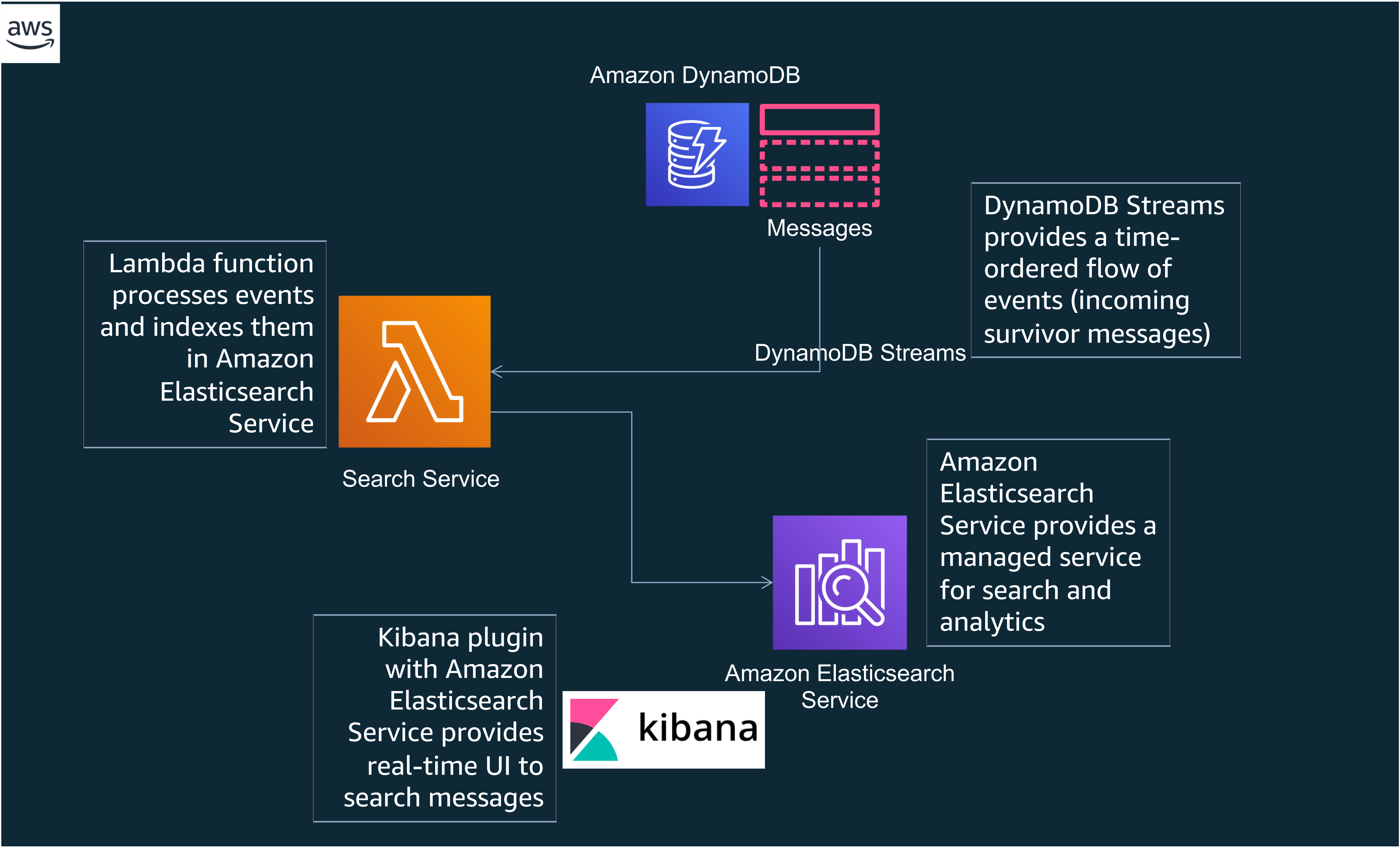
Integration Request: Apache Velocity Template Language (VTL) to transform POST parameters into JSON.

- Pre-process Twilio message
- Convert/standardize incoming parameters for Chat Service
- HTTPS POST to Chat Service

Integration Response: Apache Velocity Template Language (VTL) to transform response to XML for Twilio

- Design Principle: Service re-use
- Single implementation of DynamoDB PutItem

Lab 3: Search with ElasticSearch Service



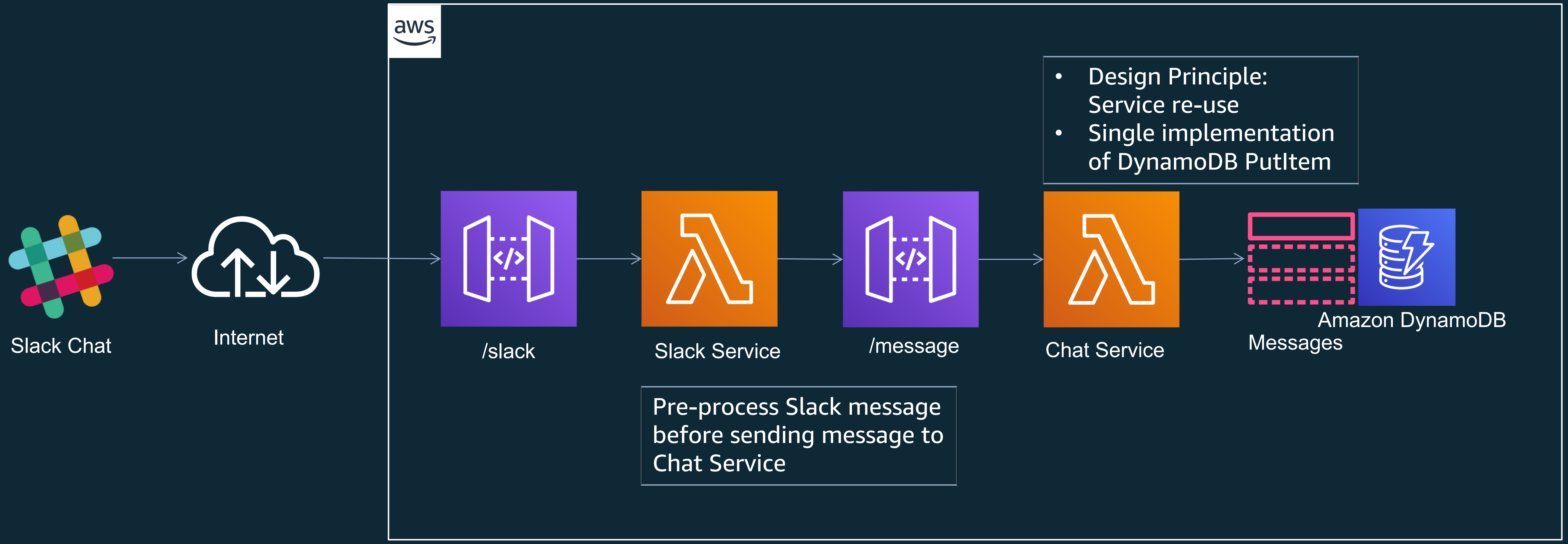
Lambda function processes events and indexes them in Amazon Elasticsearch Service

DynamoDB Streams provides a time-ordered flow of events (incoming survivor messages)

Amazon Elasticsearch Service provides a managed service for search and analytics

Kibana plugin with Amazon Elasticsearch Service provides real-time UI to search messages

Lab 4: Send Messages from Slack



- Design Principle: Service re-use
- Single implementation of DynamoDB PutItem

Pre-process Slack message before sending message to Chat Service

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 chatrooms/private chats
2. Datastore for weapons/food caches & notify survivors of cache levels
3. Build your own challenges and share your design with us!

Steps to get started

- work solo!
- Every participant shall deploy it’s own CloudFormation Stack.
- Complete add-ons from Lab Guide.
- Decide on other challenges you’ll build!
- Share your designs with fellow survivors!
- Cleanup after labs not required → accounts will be wiped automatically

Workshop available at:

https://github.com/shchodro/serverless-workshop

Thank You!