
AWS
re:Invent

SVS208

Thinking serverless: From business problem to serverless solution

James Beswick

Senior Developer Advocate, AWS Serverless

AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- Where we have come from – servers
- Where to start with serverless
- Good practices of serverless design
- 3 example applications



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

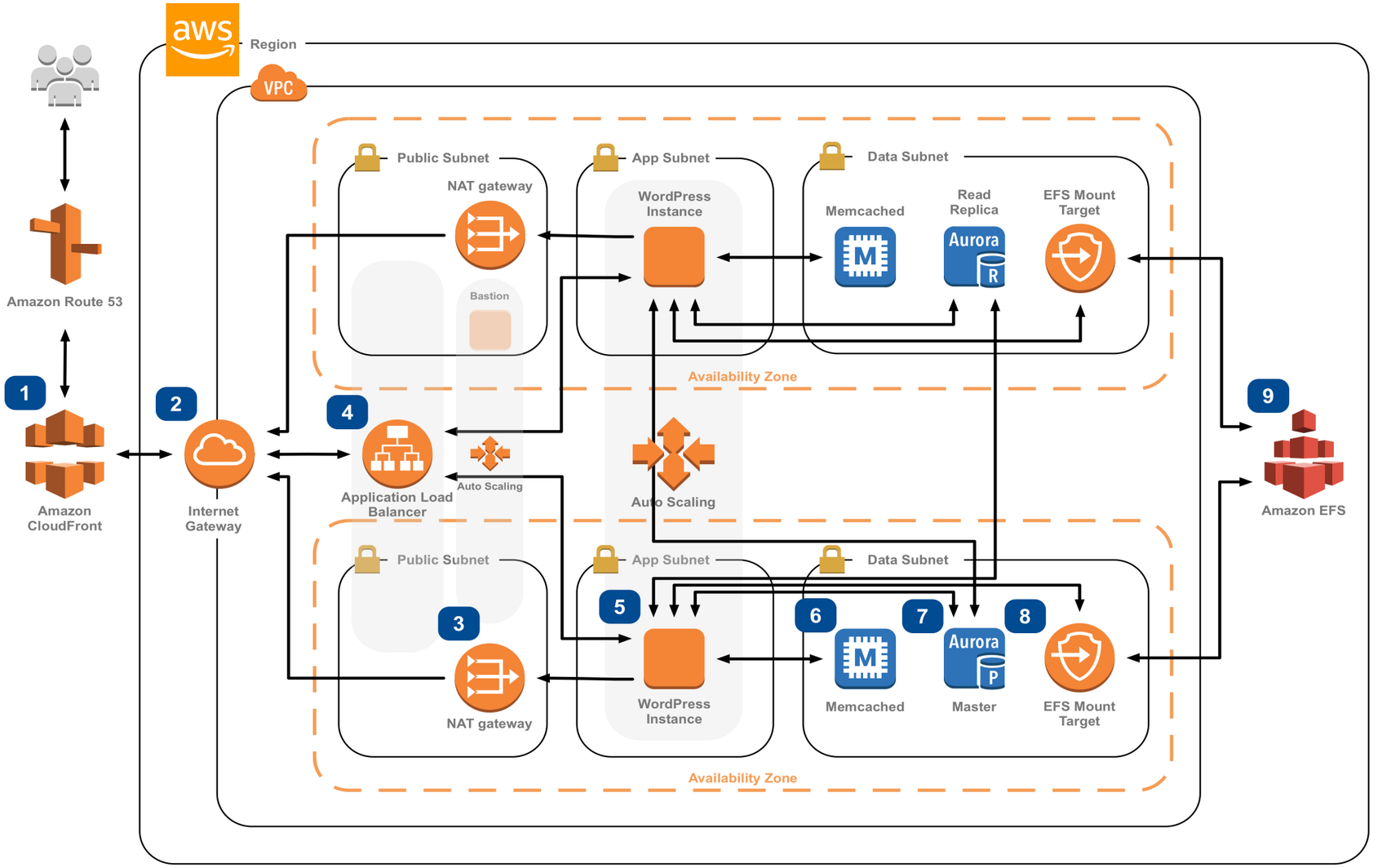
How do we use servers?

- State management
- Monolithic container for functionality
- One version, one server
- Server is an atomic unit of thinking
- The challenges of this model

WordPress Hosting

How to run WordPress on AWS

WordPress is one of the world's most popular web publishing platforms, being used to publish 27% of all websites, from personal blogs to some of the biggest news sites. This reference architecture simplifies the complexity of deploying a scalable and highly available WordPress site on AWS.



1. Static and dynamic content is delivered by **Amazon CloudFront**.
2. An **Internet gateway** allows communication between instances in your VPC and the Internet.
3. **NAT gateways** in each public subnet enable Amazon EC2 instances in private subnets (App & Data) to access the Internet.
4. Use an **Application Load Balancer** to distribute web traffic across an Auto Scaling Group of Amazon EC2 instances in multiple AZs.
5. Run your WordPress site using an **Auto Scaling group** of **Amazon EC2 instances**. Install the latest versions of WordPress, Apache web server, PHP 7, and OPcache and build an Amazon Machine Image that will be used by the Auto Scaling group launch configuration to launch new instances in the Auto Scaling group.
6. If database access patterns are read-heavy, consider using a WordPress plugin that takes advantage of a caching layer like **Amazon ElastiCache** (**Memcached**) in front of the database layer to cache frequently accessed data.
7. Simplify your database administration by running your database layer in **Amazon RDS** using either Aurora or MySQL.
8. Amazon EC2 instances access shared WordPress data in an Amazon EFS file system using **Mount Targets** in each AZ in your VPC.
9. Use **Amazon EFS**, a simple, highly available, and scalable network file system so WordPress instances have access to your shared, unstructured WordPress data, like php files, config, themes, plugins, etc.

AWS Reference Architectures

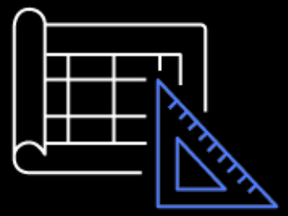
© 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Where to start with serverless

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

General approach to thinking serverlessly



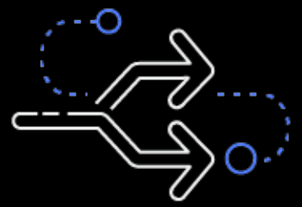
Features first

Avoid monolithic thinking



Focus on events

Events are triggers that cause action



Stateless

The key to scaling effectively



Data flow

Make data decisions early on



Use the services

Don’t reinvent the wheel

What are serverless services?



Amazon EventBridge



Amazon Simple Notification Service (Amazon SNS)



Amazon Simple Queue Service (Amazon SQS)



Amazon API Gateway



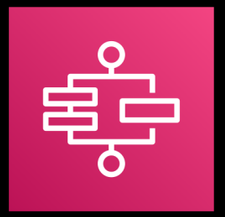
Amazon DynamoDB



AWS Lambda



Amazon S3



AWS Step Functions



Amazon Kinesis



AWS IoT Core



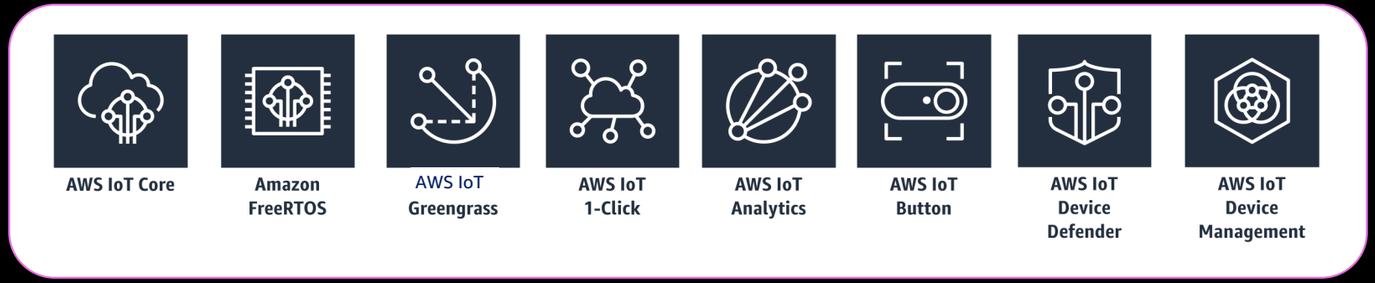
Amazon Elastic Transcoder

Integrating with other AWS services

Machine Learning



Internet of Things



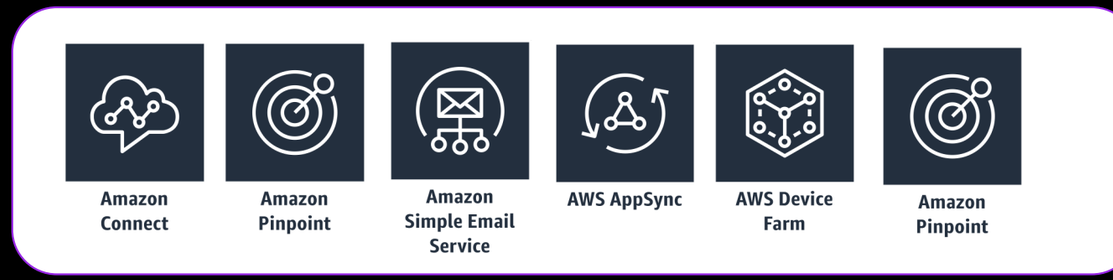
Analytics



Media



Web/Mobile/Digital



Best practices

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Understanding how AWS Lambda fits in

Attributes

- Runs on demand
- Supports many runtimes
- Responds to events
- Stateless
- Automatically scales

Best practices

- Avoid lift-and-shift
- One function per purpose
- Keep functions small
- Choose the right runtime for the job
- Use functions for business logic and plumbing between services

Good serverless practices

- Infrastructure is disposable
- Asynchronous versus synchronous processing
- You can mix and match runtimes
- Security still matters: build in from the start
- Automation
  - AWS Serverless Application Model (AWS SAM)
  - Serverless Framework

Examples

AWS re:Invent

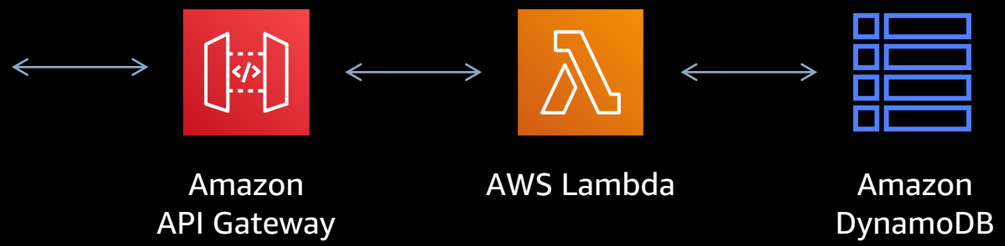
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

1. Form upload

Create a serverless application to support a customer review form submitted from a webpage

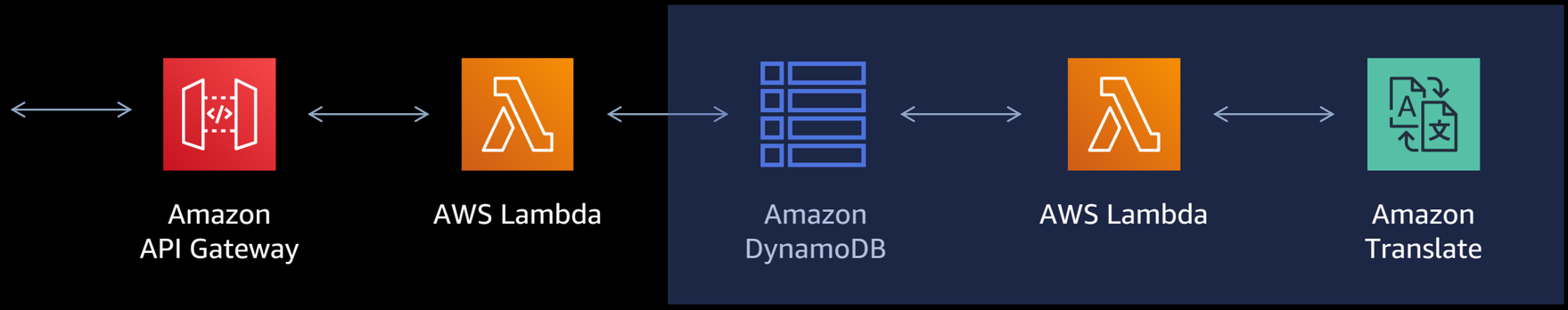
1. Form upload

Create a serverless application to support a customer review form submitted from a webpage



1. Form upload

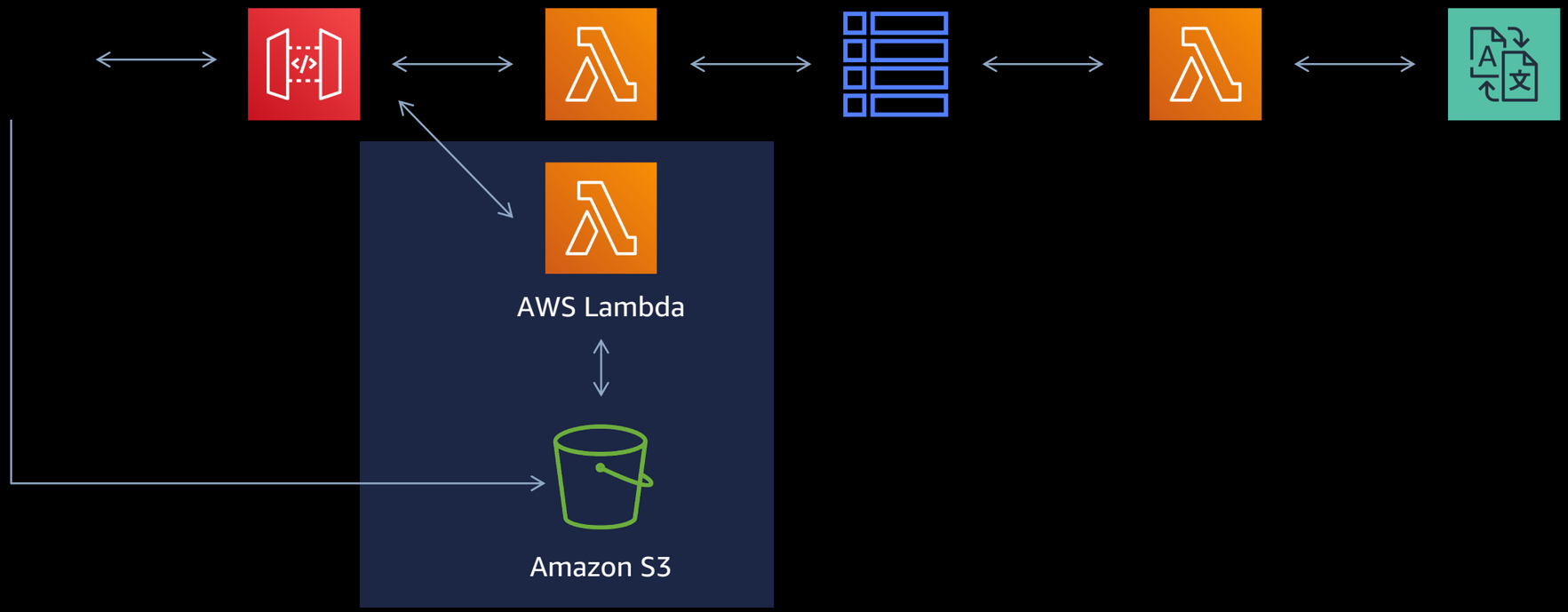
Create a serverless application to support a customer review form submitted from a webpage



Incoming responses must be translated to English

1. Form upload

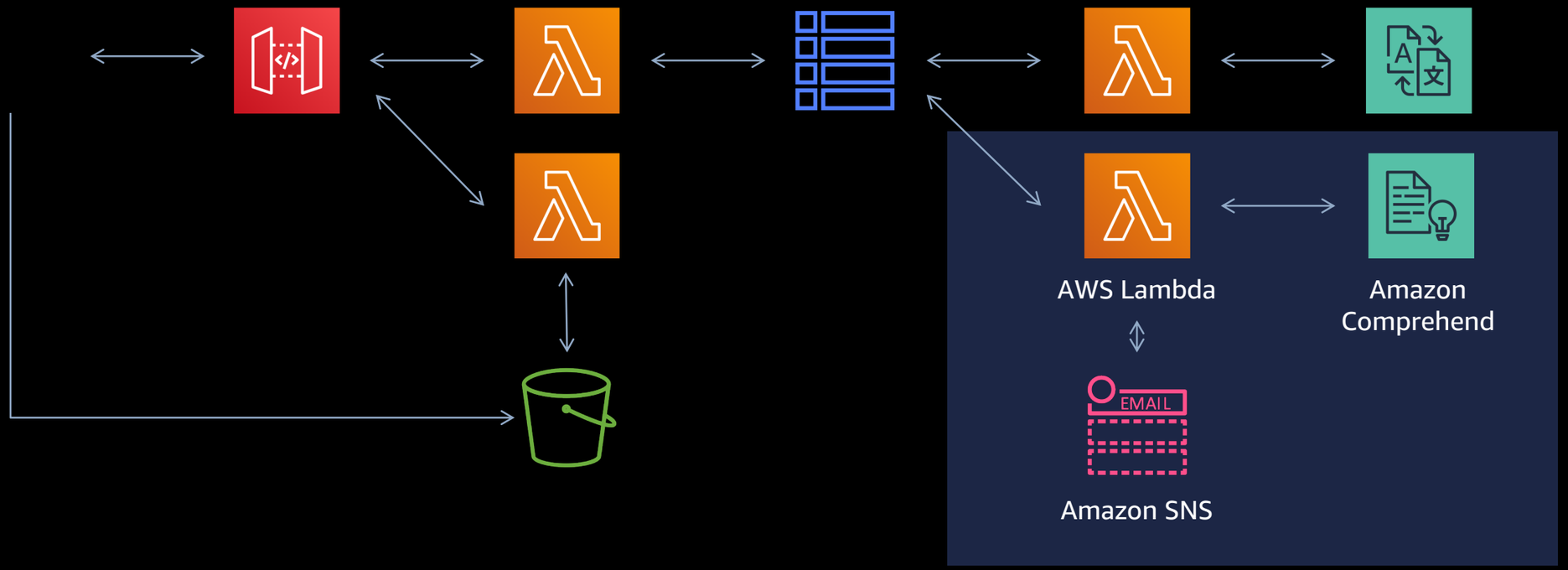
Create a serverless application to support a customer review form submitted from a webpage



Allow user to upload image with a response

1. Form upload

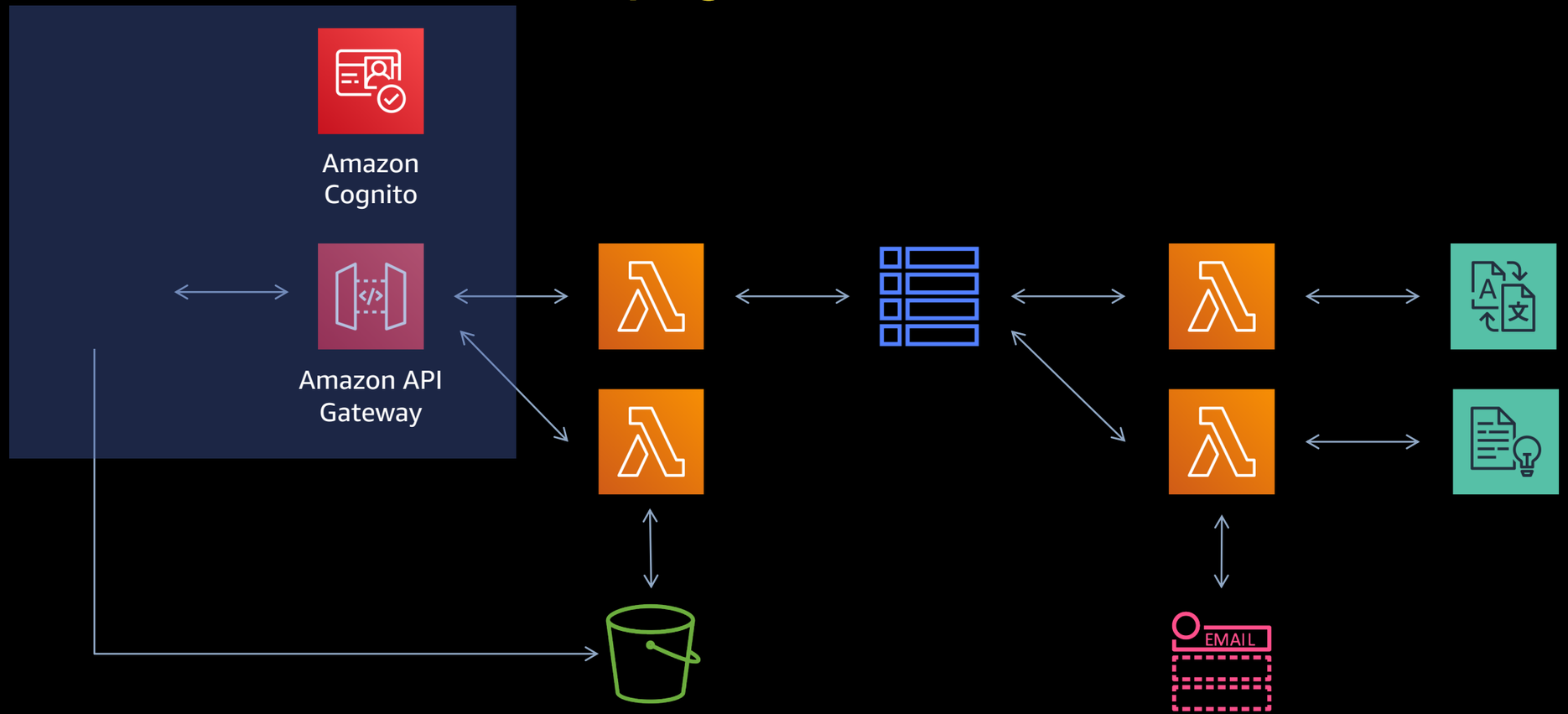
Create a serverless application to support a customer review form submitted from a webpage



Email any negative comments immediately

1. Form upload

Create a serverless application to support a customer review form submitted from a webpage



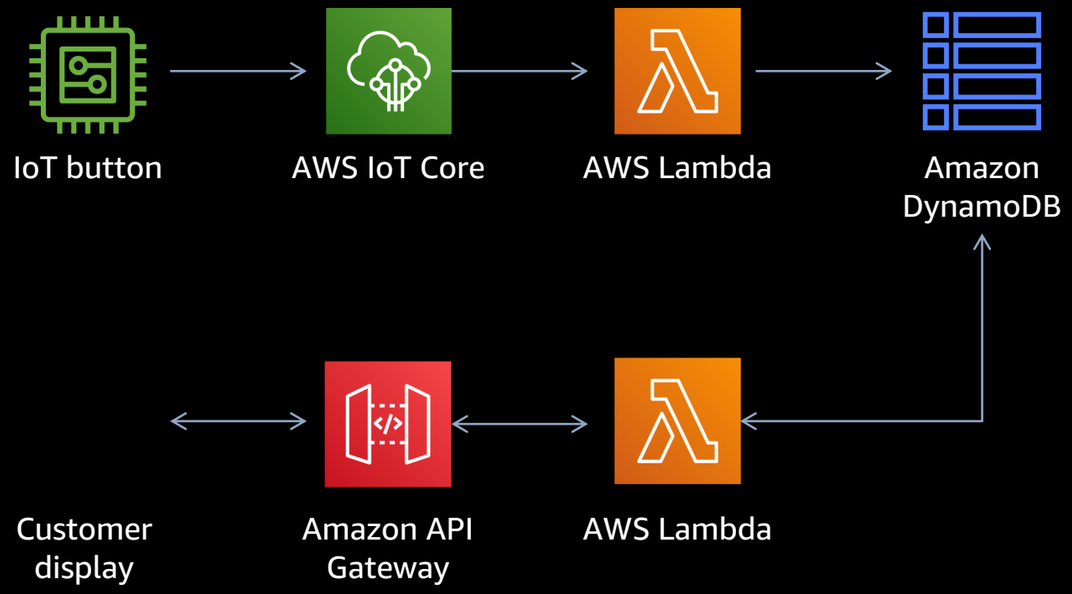
Only allow authenticated users

2. Create a virtual queue

A global retailer wants to implement a virtual queue for customers. Use an IoT button to manage a “now serving” counter.

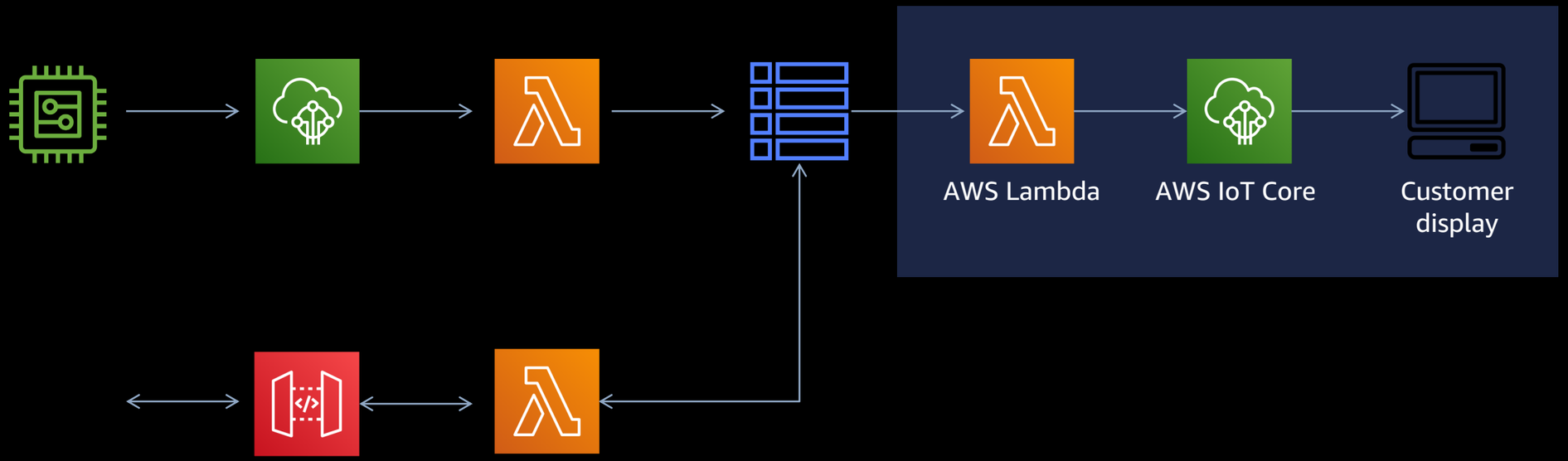
2. Create a virtual queue

A global retailer wants to implement a virtual queue for customers. Use an IoT button to manage a “now serving” counter.



2. Create a virtual queue

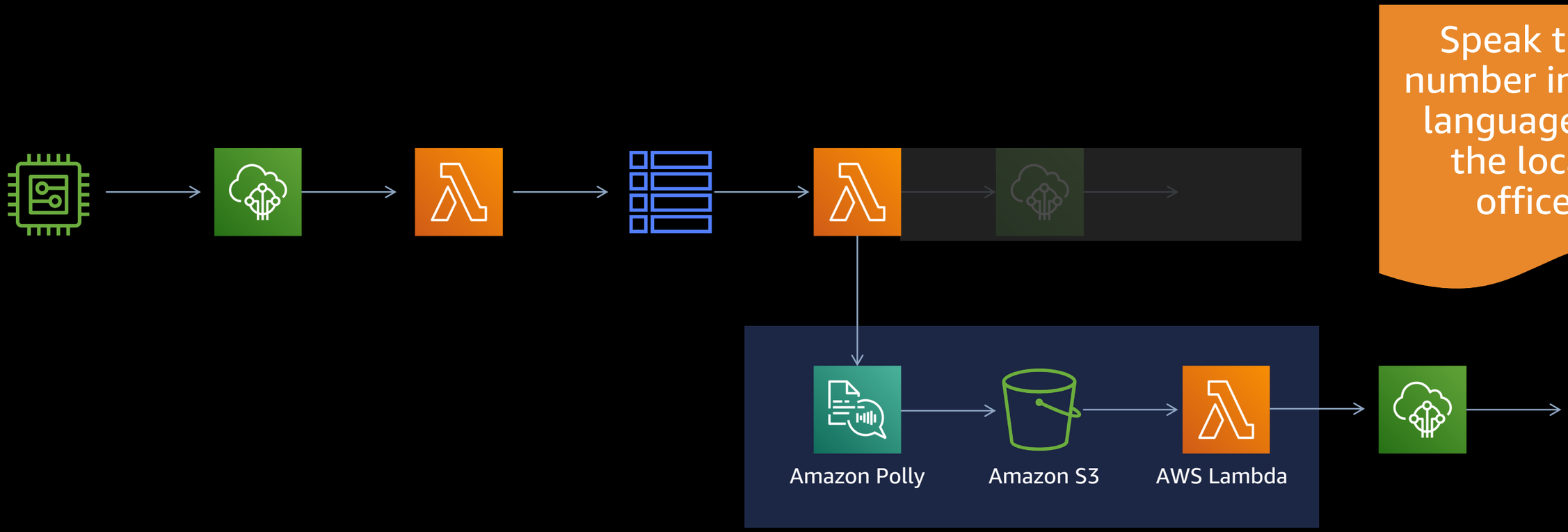
A global retailer wants to implement a virtual queue for customers. Use an IoT button to manage a “now serving” counter.



Create a real-time display showing the current count

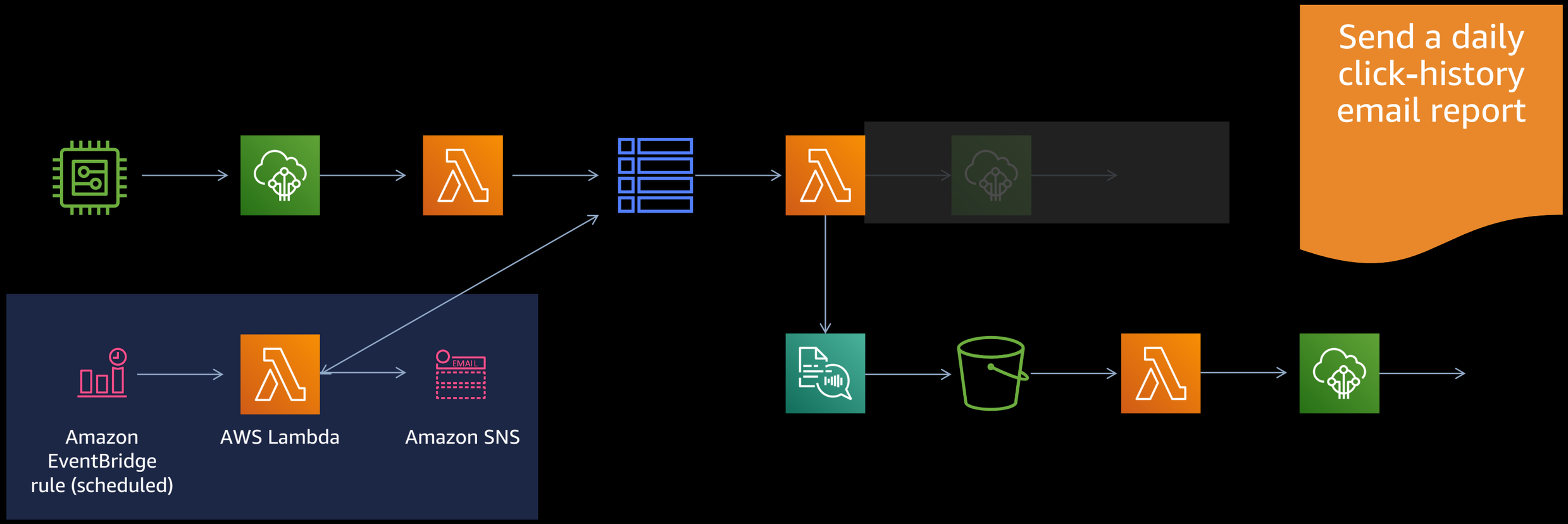
2. Create a virtual queue

A global retailer wants to implement a virtual queue for customers. Use an IoT button to manage a “now serving” counter.



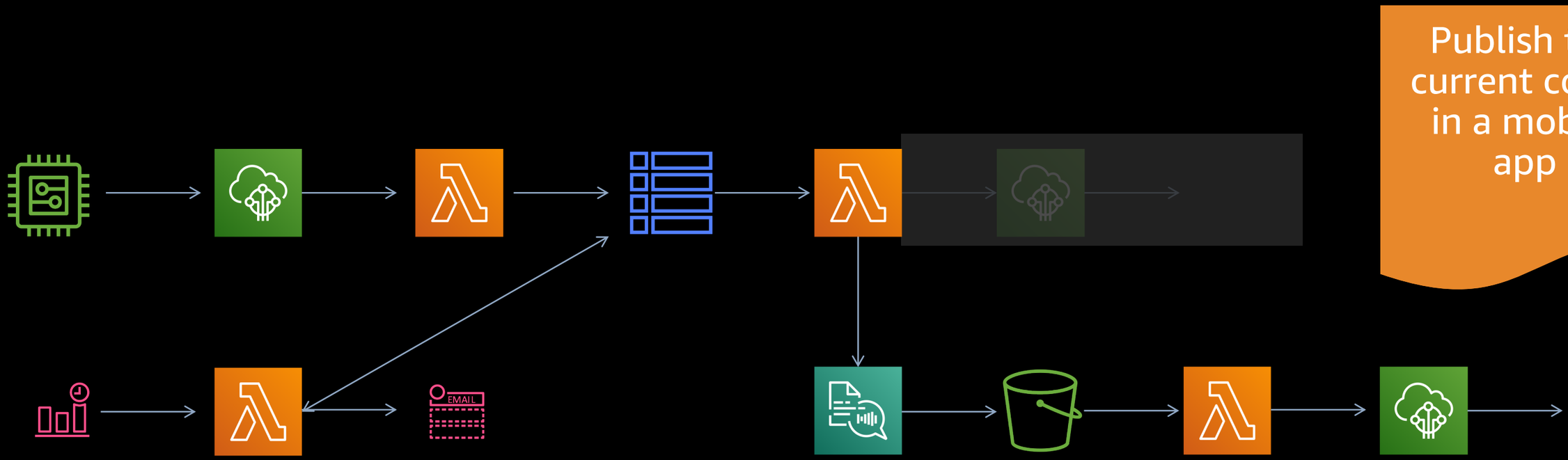
2. Create a virtual queue

A global retailer wants to implement a virtual queue for customers. Use an IoT button to manage a “now serving” counter.



2. Create a virtual queue

A global retailer wants to implement a virtual queue for customers. Use an IoT button to manage a “now serving” counter.

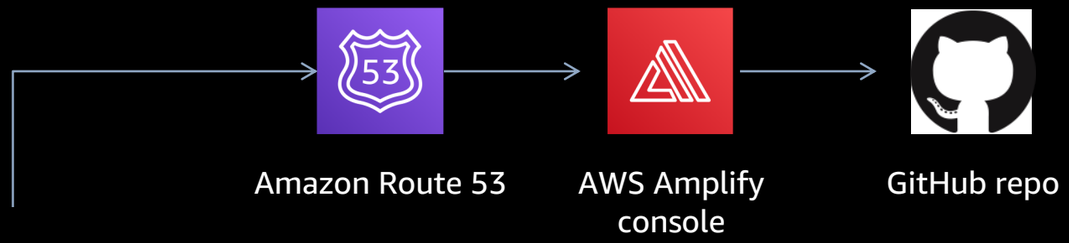


3. Web application

Create a serverless web application to support a national e-commerce business

3. Web application

Create a serverless web application to support a national e-commerce business



Ensure fast performance for visitors in multiple regions

3. Web application

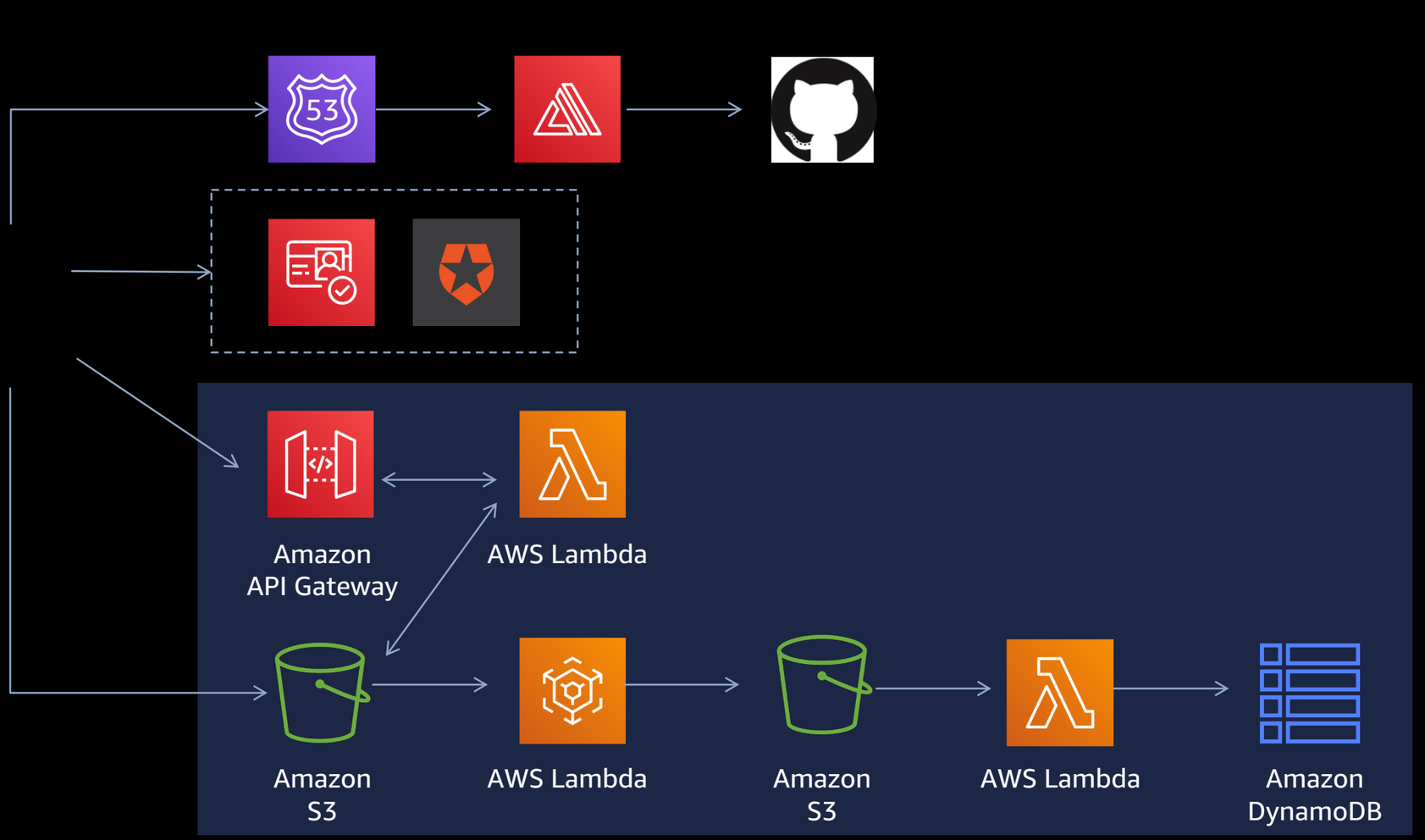
Create a serverless web application to support a national e-commerce business



Allow users to create accounts (including social login)

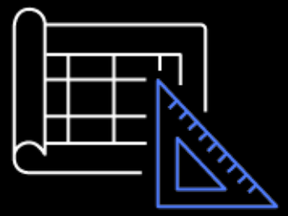
3. Web application

Create a serverless web application to support a national e-commerce business



Support uploading and serving user videos

Conclusion



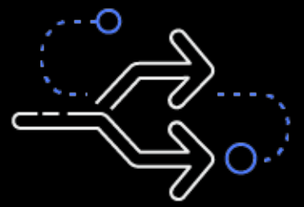
Features first

Avoid monolithic thinking



Focus on events

Events are triggers that cause action



Stateless

The key to scaling effectively



Data flow

Make data decisions early



Use the services

Don’t reinvent the wheel

Thank you!

James Beswick

Senior Developer Advocate, AWS Serverless
AWS
Email: jbeswick@amazon.com
Twitter: @jbesw

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.