Introduction to Cloud Services

UWB Hacks the Cloud
April 17, 2020
Presented by UWB ACM

Agenda

- Introduction
- Computation with AWS Lambda
- Notifications with SNS
- Example Project: FeelGood
- File Storage with S3
- Databases with DynamoDB
- APIs with API Gateway
- Example Project: Crow Facts
- Closing Remarks

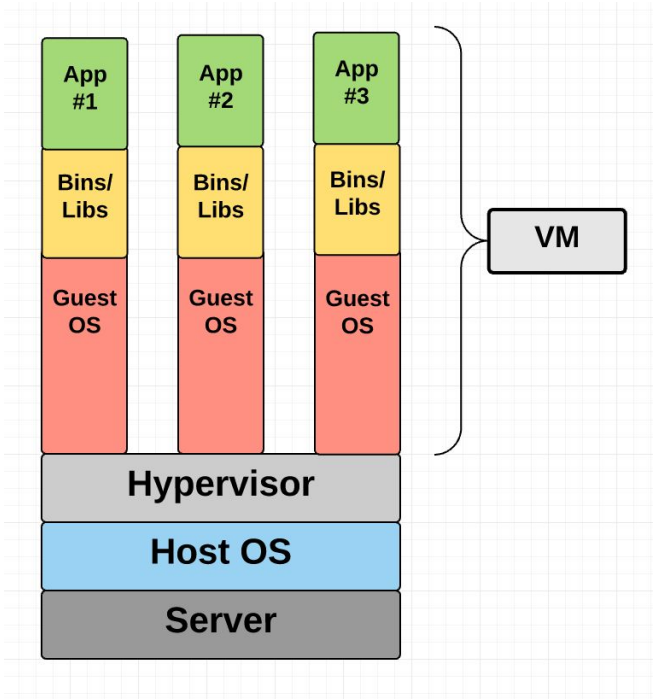
Introduction

What is “The Cloud”?

- Someone else's computer(s)
- Cloud providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) maintain data centers with many servers.
- This approach means developers do not need to worry about server maintenance and security
- As a cloud user, you need to keep track of your usage in order to avoid getting overcharged

Services vs Virtual Machines

- Virtual Machines are quite literally a virtualized computer
  - Each VM has its own OS and software
  - VMs are an Infrastructure-as-a-Service (IaaS) offering
- Services are at a higher level of abstraction than VMs
  - You don't need to worry about the underlying OS or hardware
  - Services are a Platform-as-as-Service (PaaS) offering
  - Examples:
    - AWS Lambda
    - AWS DynamoDB
    - Azure App Services



Serverless Computation

What is Serverless Computation?

- Code that runs by schedule or trigger
- Traditionally, setting up a server for computation means a lot of configuration
- Serverless computing abstracts away all of the low level details
- You simply write your code, configure access settings, and deploy the code.
- Serverless computing offerings are easy to connect to the provider's other services
- Cheap!

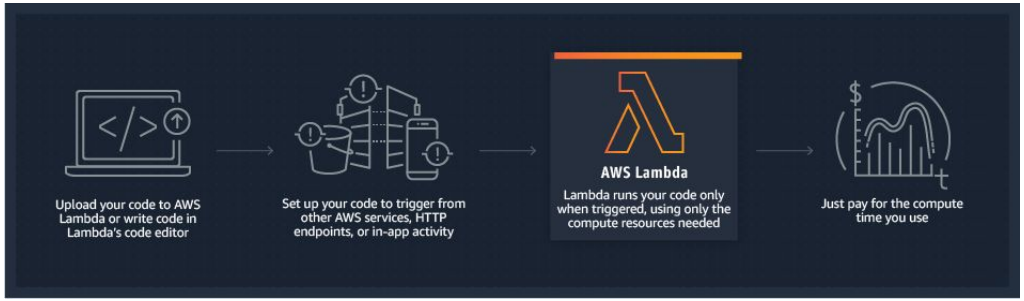


Figure 1: How AWS Lambda Works

How Do I Use It?

- Create a function in the web console
- Configure access settings
  - This is how you will be able to invoke the function and connect it to other cloud services, like databases or analytics
- Write the code. Two ways:
  - Directly in the web console interface (we'll stick to this for now)
  - Locally on your computer, if you’re a power user. You will need to package and upload the code if you use this option
- Deploy the code
- That's it?
  - Not quite!
  - You still need to configure other services in order to invoke your function and handle its output

Cloud Provider Options

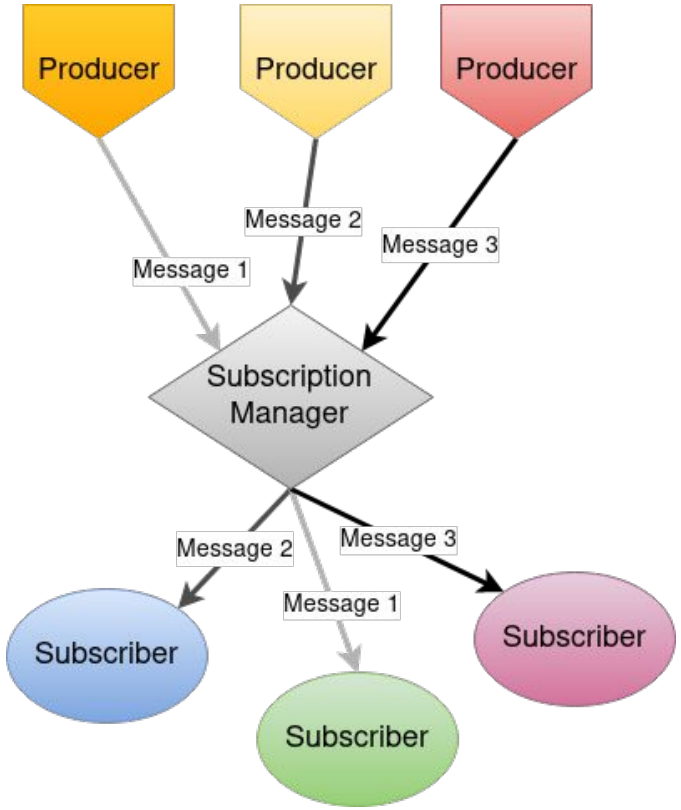
- Each cloud provider's serverless compute offerings differ in their functionality
  - The services each provider will allow you to connect to your serverless functions will differ as well

- The documentation is always a great place to start
  - AWS Lambda
  - Azure Functions
  - GCP Cloud Functions

Notification Services

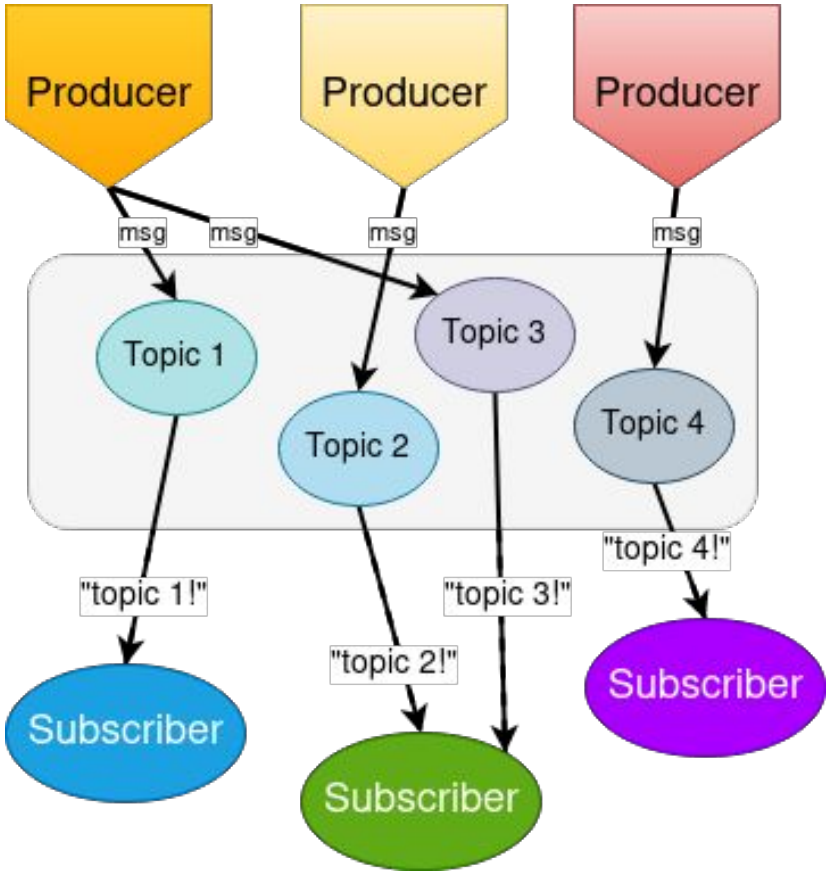
What are Notifications as a Service?

- **Send messages** to entities
  - Other cloud services
  - External applications
  - Push notifications (mobile apps)
  - Text messages
- The **“Pub/Sub” Model**
- Entities **subscribe** to the service to receive future notifications
- Other entities **publish** notifications in text format



The Topic Model

- Key mechanism used by subscription manager to determine recipients for messages
- Topics are defined message pipelines with a specific set of subscribers
- Allows subscribers to remain anonymous to service which published message
- Multiplicity between producers and topics, as well as topics and subscribers



How To Use It (an AWS example)

1. Create a topic; give it a relevant name if desired
    a. The new topic will have an Amazon Resource Name (ARN) created for it
    b. The ARN will allow other services to interact with the topic programmatically
2. Provide subscription mechanism
    a. Services can subscribe to topics to receive messages and trigger events
    b. Users can subscribe to topics; delivery facilitated through push notifications, SMS or email
    c. Subscription can be done in the AWS console (browser-based) or programmatically, via SDKs (Python SDK here)
3. Create a publishing mechanism
    a. Use the SNS SDK to programmatically push messages to a topic
    b. Publish messages to a topic from the AWS console (good for testing)
4. Give it a whirl!

Subscribers and Publishers

Allowed subscribing entities and publishing entities will vary between providers. Customize a publishing pipeline to suit your needs.

For example:

- **Problem:**
  - AWS allows SMS-enabled phone numbers to subscribe to a topic, but GCP and Azure do not.
- **Solution:**
  - All 3 providers allow publishing to a 3rd party HTTP/S enabled API.
  - Use a 3rd party application for SMS with a robust API (such as Twilio).
  - Subscribe the 3rd party endpoint to the SNS topic.

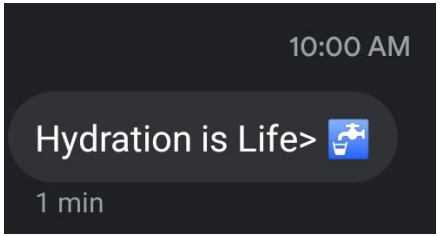
Cloud Provider Options

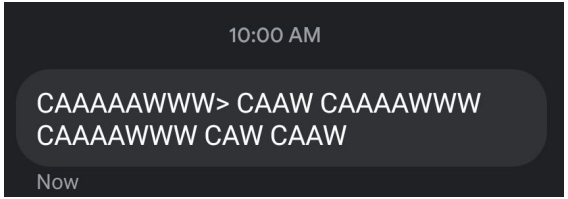
- Each provider will differ in service capabilities
  - Different subscribing entity options and publishing mechanics
  - Different user-facing subscription options
- The documentation is always a great place to start!
  - AWS Simple Notification Service developer guide and API reference
  - Azure event handling service comparison and Event Grid documentation
  - GCP Pub/Sub documentation (including examples, tutorials, and API reference)

Example Project: **FeelGood**

Goals

- Good vibes
  - We wanted to send good vibes to folks who wanted to receive them.
  - Wouldn’t it be nice to get words of encouragement delivered to your phone sometimes?
- Text messages
  - Get positivity delivered directly to your phone via SMS!
- Opt into several different types of vibes
  - Feel Good ➜ good vibes via SMS
  - Feel Bad ➜ silliness via SMS
  - Feel Hydrated ➜ quench via SMS
  - Feel Caww ➜ CAWW CAAWW CAW CAW

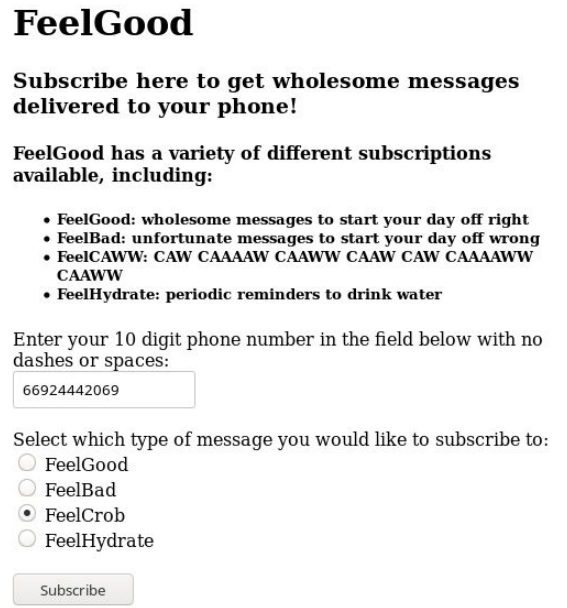


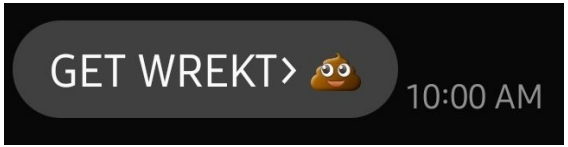


Design

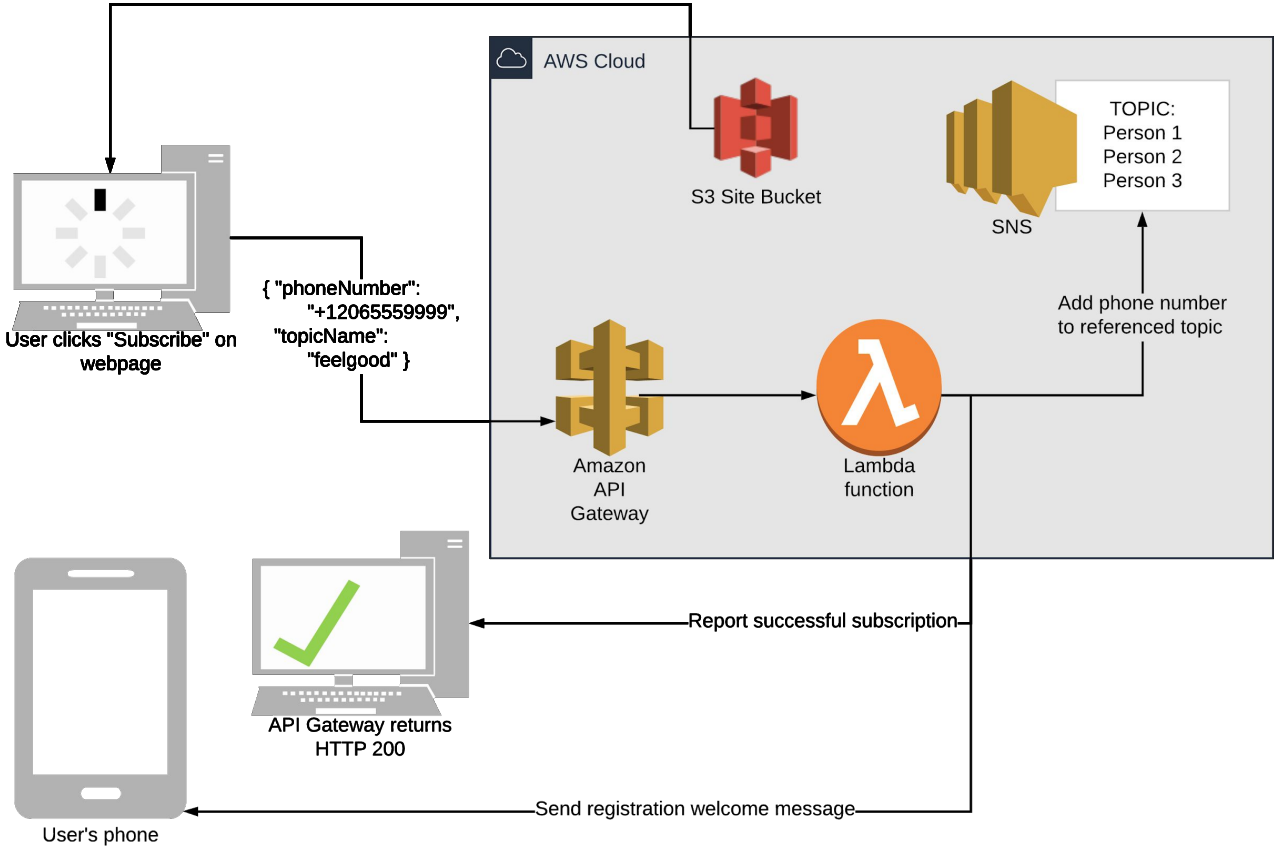
- SMS
  - Delivered to subscribers via Amazon SNS
- How to get users registered?
  - Let users register via a website hosted on S3
- How to send data from site to SNS?
  - Set up API call via Amazon API Gateway

- When to send SMS?
  - Publishing messages to SNS is triggered via CloudWatch at specific time intervals
- How to choose messages to send?
  - Lambda function picks all messages to publish

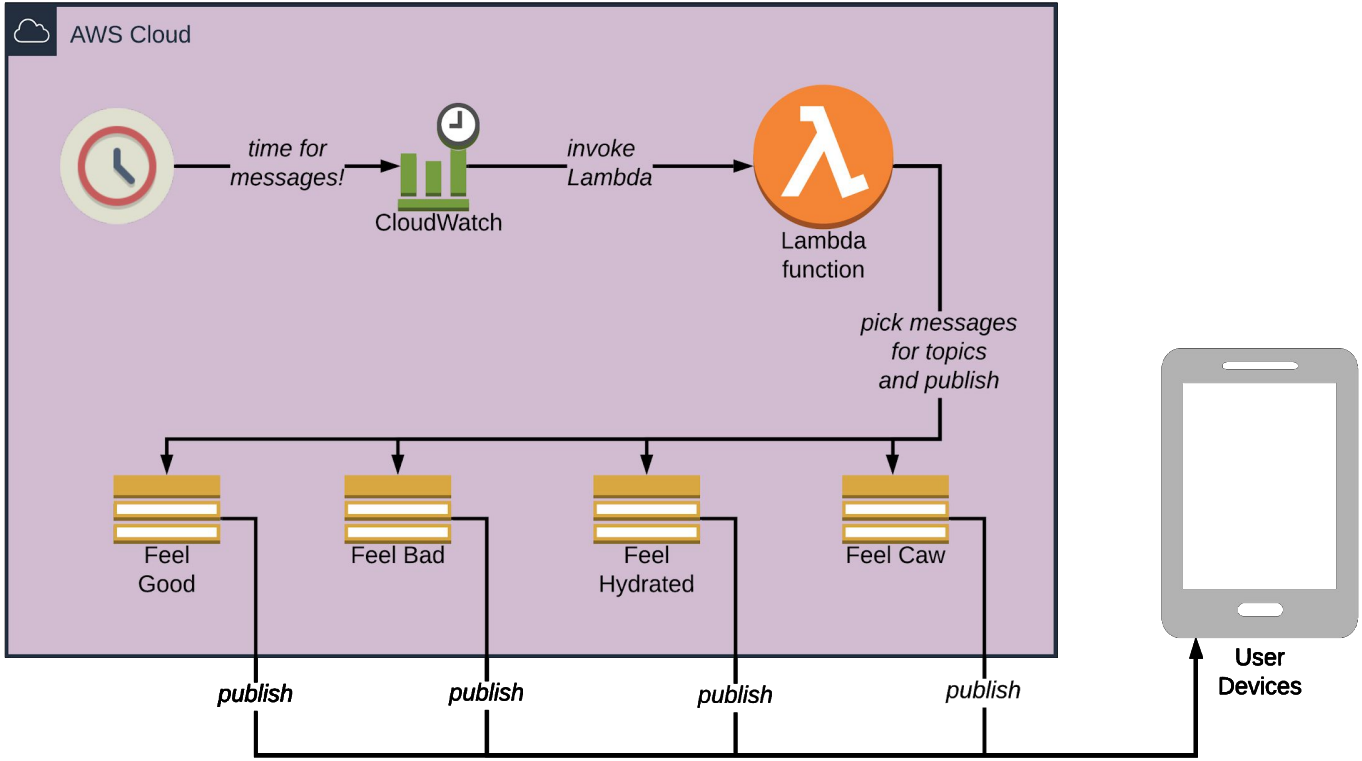




Implementation: Signups



Implementation: Sending Messages



Try it Out!

Go to...

https://feelgood.support/

to register for one or more topics of your choice! SMS will be sent at the top of each hour.

This project has a detailed write-up at *https://docs.uwbhacks.com/feelgood.html*

Code examples and configuration details are available in the GitHub repository: *https://github.com/UWB-ACM/feelgood*

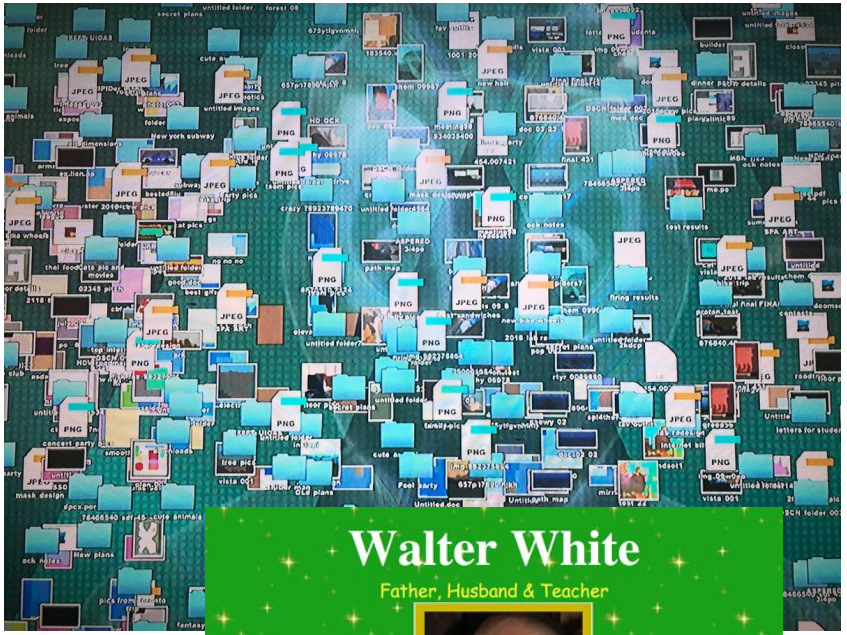
File Storage

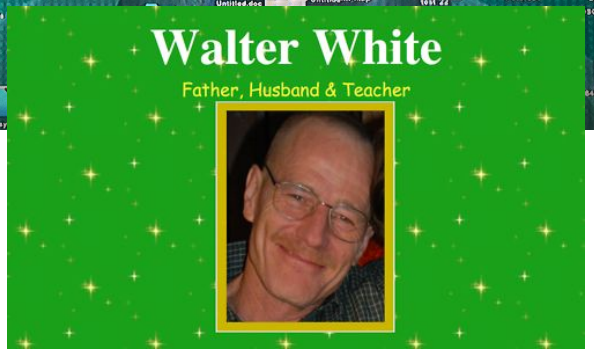
What is File Storage in the Cloud?

- Cloud storage is scalable, reliable, secure, and cost-effective
- Cloud service providers typically offer three types of cloud storage:
  - **File storage:** Get a storage block which can be mounted to many VM hosts.
  - **Block Storage:** Get a storage block which can be mounted as an external drive to a single VM host.
  - **Object storage:** Group relevant files and directories into a relevant bucket. These storage entities are decoupled from host machines but can be accessed through cloud provider APIs.
- How do I know which storage type I should choose?
  - AWS has a good explanation of what storage solution to choose based on use cases.

How Do I Use It?

- Programmatically back files up to the cloud via APIs
  - Source code
  - Images
  - Log files
  - Or anything else!
- Create filesystems for one or more applications
  - Network File Storage (NFS) volumes
- Serve a static website
  - AWS Usage
  - Azure Usage
  - GCP Usage





Make your own fundraising site?
http://www.savewalterwhite.com/

Cloud Provider Options

- Filesystem shared between hosts: AWS Elastic File System
- Filesystem for single host: AWS Elastic Block Storage
- Object storage: AWS S3

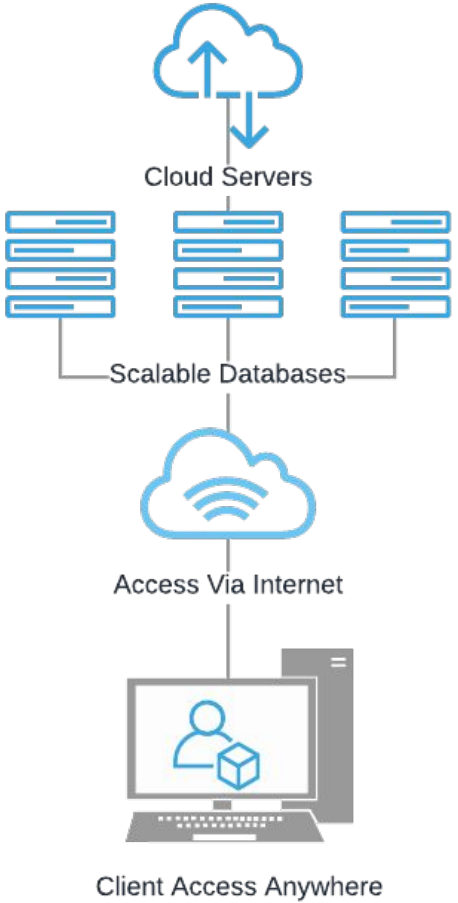
- Filesystem shared between hosts: Azure Files
- Filesystem for single host: Azure Disk Storage
- Object storage: Azure blob storage

- Filesystem shared between hosts: GCP Filestore
- Filesystem for single host: GCP Persistent Disk
- Object storage: GCP Cloud Storage

Databases

Databases in the Cloud

- Store information on **hosted servers**
  - No database server management on user end
  - Accessible from anywhere
- **Scalable** for project growth
  - Fast response rates
  - As big or as small as needed
- Additional **functionality provided** by service
  - Multiple database types
    - SQL, NoSQL, etc.
  - Automatic performance tuning
  - Security, disaster recovery, etc



Database Types

SQL

- Relational
  - Defined schema
  - Normalization
- Vertical scaling, supports sharding
- Structured Query Language

```
SELECT COUNT(1)
FROM CrowFacts
WHERE habitat = 'Eastern Indonesia';
```

When to use?

- Defined dataset
- Data integrity needed
- Experienced developers

NoSQL

- Non-relational
  - Key-Value Store
  - Denormalization
- Horizontal scaling
- .JSON, .CSV, XML, etc.

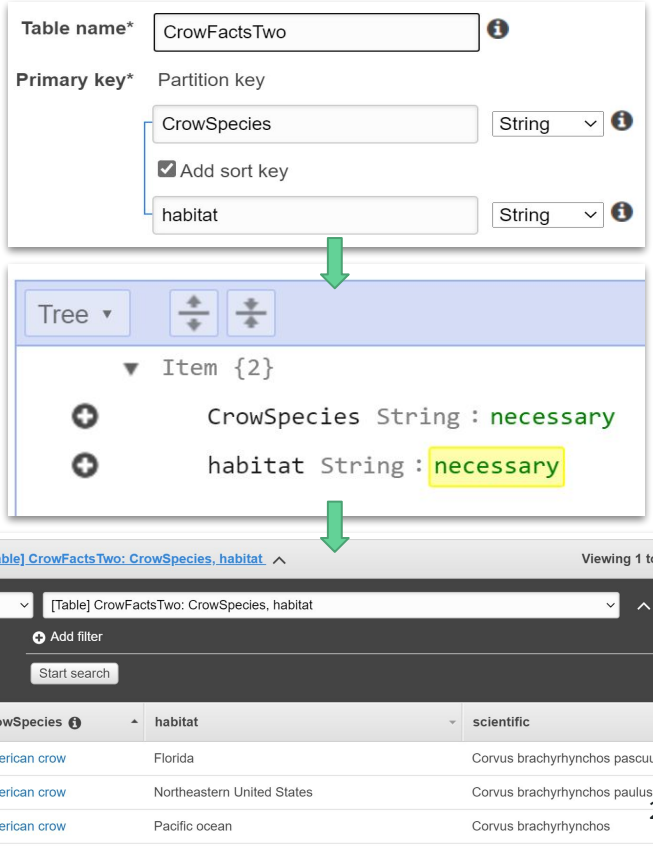
```
db.book.count({
  "habitat": "Eastern Indonesia"
});
```

When to use?

- Undefined or changing data sets
- Less experience with DB's
- Smaller projects (like these!)

How To Use (AWS DynamoDB)

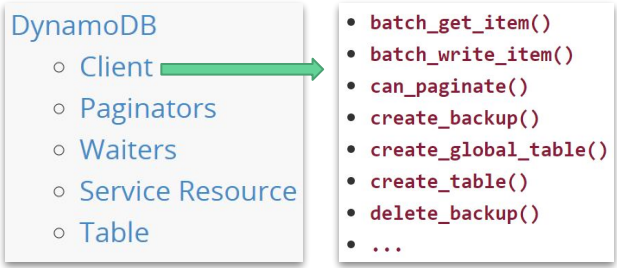
- Create a Table
  - Choose **primary key**
    - PK = partition key + sort (optional!)
  - Choose table **settings**
    - **Default** is good!
- Populate with Data
  - Data must **contain** primary key
  - Create **new item** from console
    - **Individual** entries only
  - Can import **.json or .csv** with Lambda
- Search and Sort
  - DynamoDB **Console**
    - **Scan**
    - **Query**
  - Lambda functions



Important Tidbits

SDK's

- Provides toolset
  - Libraries
  - API's
- Languages
  - Python
  - Java
  - Etc.
- Cloud services provide
- Example: Boto3



Primary Keys

- Necessary to retrieve info
- Must be unique
  - Single

```
PRIMARY KEY (CrowSpecies)
```

  - Combination

```
PRIMARY KEY (CrowSpecies, habitat)
```

- Reserved Words
  - E.g.; location, add, etc.
  - AWS
    - Found Here
  - Azure
    - Found Here
  - Google Cloud
    - Found Here

External Interaction

- Permissions
  - Different for providers
  - Necessary to access DB
- Cloud Provider Console
  - Different for service
  - Similar functionality
  - Direct coding!
- Personal Machine
  - Extra downloading
  - SDK's
  - Use own IDE

Cloud Provider Options

AWS Databases

- Aurora
- Redshift
- DynamoDB
- DocumentDB

https://aws.amazon.com/products/databases/

Azure Databases

- Azure Cosmos DB
- Azure SQL Database
- SQL Server on VM's
- Azure Database for MySQL

https://azure.microsoft.com/en-us/product-categories/databases/

Google Cloud Databases

- Cloud SQL
- Cloud Spanner
- Cloud Bigtable
- Cloud Firestore

https://cloud.google.com/products/databases/

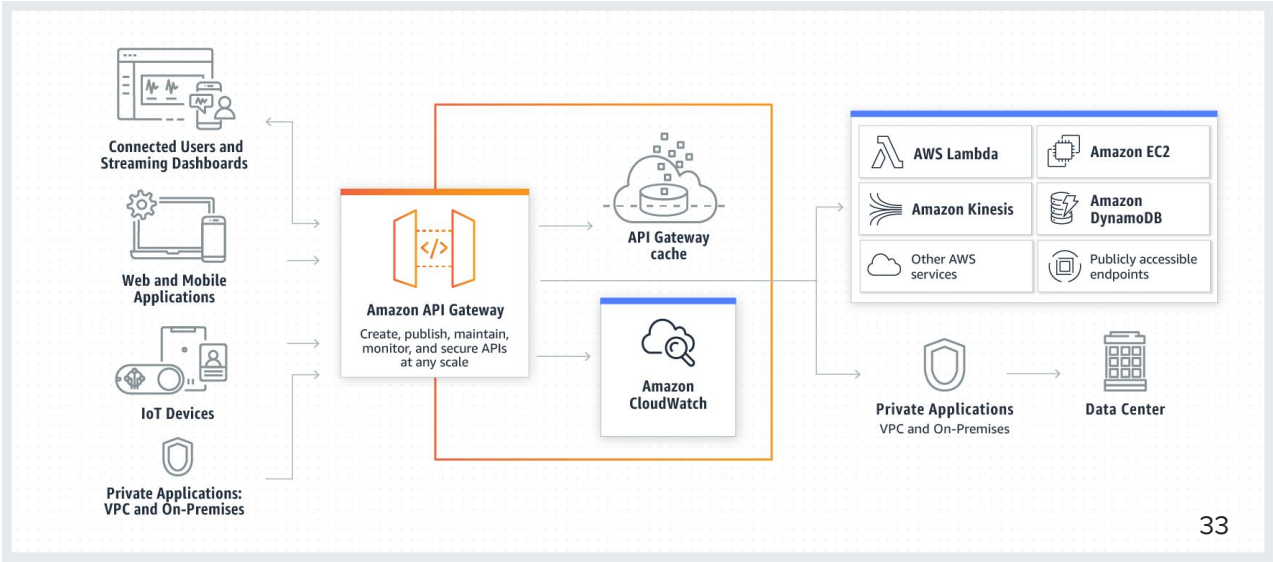
REST APIs in the Cloud

What is an API?

REST APIs takes **requests from a client and processes the request according to the developer's specifications.**

Think of it like **the front-door for clients to access back-end application services.**

All major cloud providers support API creation.

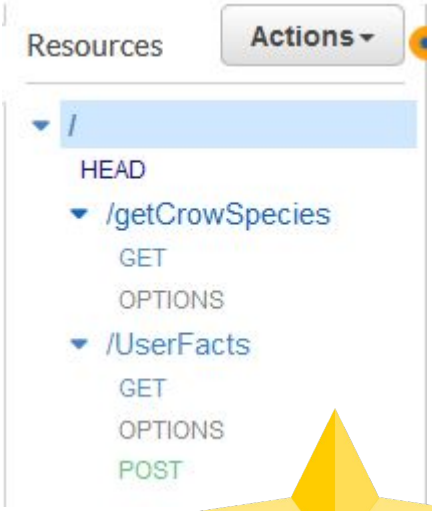


What are API Resources?

A Resource is a hierarchy of endpoints in your API. Best practice design ensures **resource hierarchies of APIs represent data models**.

There is a 1:1 correlation between a Resource and a URL for an API endpoint.

For each Resource, we can associate 1 or more methods.



Q: What's wrong with this resource schema?

If we want to get to our *GET* method for *CrowSpecies* we would send a GET request to:

`StageURL.serverLocation.com/StageName/`**`getCrowSpecies`**

What are API Methods?

RESTful APIs assign one or more HTTP methods for each resource.

The HTTP methods (also called HTTP verbs) let the client interact with the data represented by that resource following standard CRUD models.

For example, on Linux, we can GET (retrieve) user facts and POST (add) a new user fact with these method calls:

```
curl UR.L/stage/UserFacts -X GET
curl UR.L/stage/UserFacts -X POST -d '{"data":"hmm"}'
```

This `curl` syntax is incomplete; for illustrative purposes only

GET

Send an HTTP request to retrieve information from the endpoint.

The request can include parameters specifying which records the client needs. For example, the GitHub API allows the “username” parameter to get details about the user.

POST

Send new data to create a new entry. May create duplicate entries. The new data is specified as a method parameter.

PUT

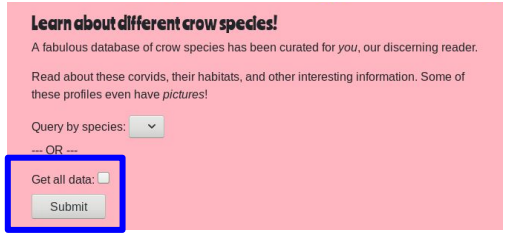
Same as the **POST,** except calling **PUT** multiple times will not produce duplicate entries. This method’s protocol allows existing entries to be overwritten or updated.

https://restfulapi.net/http-methods/

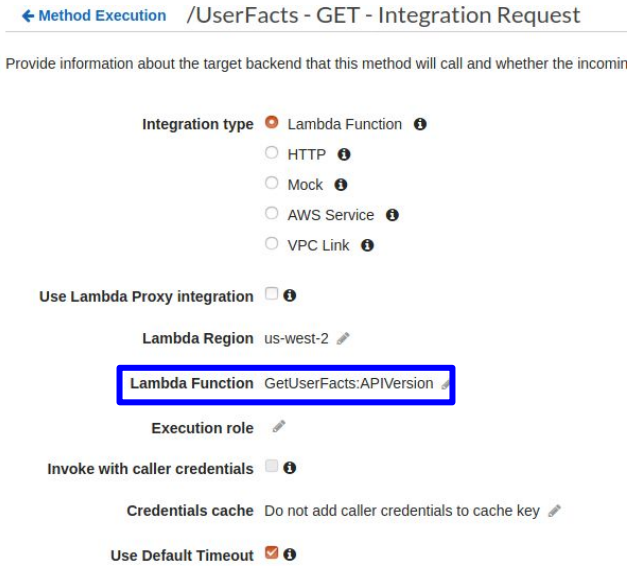
Connecting an API Method to a Service

A common way to use API services is to have a front-end application access a serverless function like AWS Lambda.

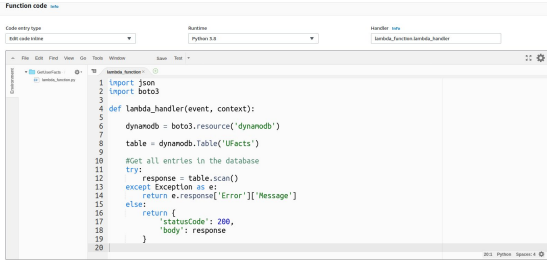
Front-End Sends a request to API Gateway



API Gateway Sends Request to Lambda Function



Lambda does computation and returns



CORS (Cross-Origin Resource Sharing)

CORS is an HTTP protocol which specifies whether specific domains can interact with external resources.

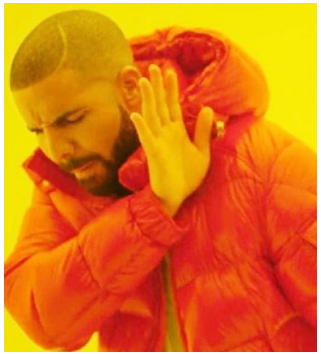
By default, most browsers require the external resource to "whitelist" the site you're visiting (aka the `Origin`) using relevant HTTP headers. This includes API calls to API Gateway.

To make matters simple, you may enable CORS for all origins with the `*` wildcard.

Why is this a bad idea?

Read more about CORS:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
- https://serverless.com/blog/cors-api-gateway-survival-guide/



Allow all domains

*Allow only the domains which should have access***

***Do as I say, not as I do...*

API Gateway & The Front-End

Let’s say our website wants all the data in the database.

We would make a **GET** request to our Lambda function through API Gateway using any standard request library.

A popular and easy-to-use HTTP request library for JavaScript is the jQuery AJAX library.

Method Request

API Gateway URL

CORS

```
function getAllUserFacts() {
    // The call which retrieves the DynamoDB data for crow species.
    $.ajax({
        type: "GET",
        url: "https://i        h.execute-api.us-west-2.amazonaws.com/test/UserFacts",
        dataType: "json",
        crossDomain: true,
        contentType: "application/json; charset=utf-8",
        success: function (data) {
            // show a success message
            alert("Success!");
            // add content to page
            displayUserFacts(data);
        },
        error: function () {
            alert("Unsuccessful");
        }
    });
}
```

How Do I Use It? (Putting It All Together in AWS)

1. Create an API in API Gateway
2. Add some resources
3. For each resource, add 1 or more methods
4. For each method, choose your integration point

   https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-method-settings.html

   a. Choose “Mock” if you only want to experiment
   b. Choose Lambda to select an existing lambda function to send the request to
   c. Choose other options if you’re a power user
5. Enable CORS for your API

   https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-cors.html

   a. This is optional if you’re a power user and know exactly what you’re doing
6. Deploy your API
   a. Create a new deployment stage if you don’t already have one
7. Call your API via a website (JavaScript), command line (curl), Postman

Cloud Provider Options

- AWS: API Gateway
  - https://aws.amazon.com/api-gateway/
- GCP: Cloud Endpoints/API Management
  - https://cloud.google.com/apigee/
- Azure: API Management
  - https://azure.microsoft.com/en-us/services/api-management/

Essentially all these API services aim to solve similar problems but may use different syntax and processes.

Example Project: 

Goals

Crow Facts for Users

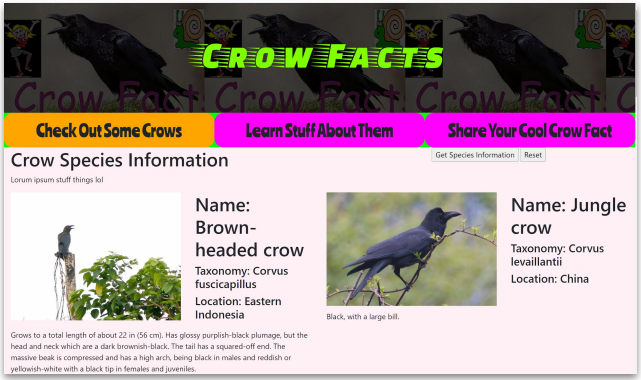
- Give users some fun facts about crows.
- Provide info on different species of crows.
- Access and input your own information.

Front End

- Website to interact with data
  - Render crow species information retrieved from database! Get all crows we got!
  - Render fun facts retrieved from the database! Get all the facts!

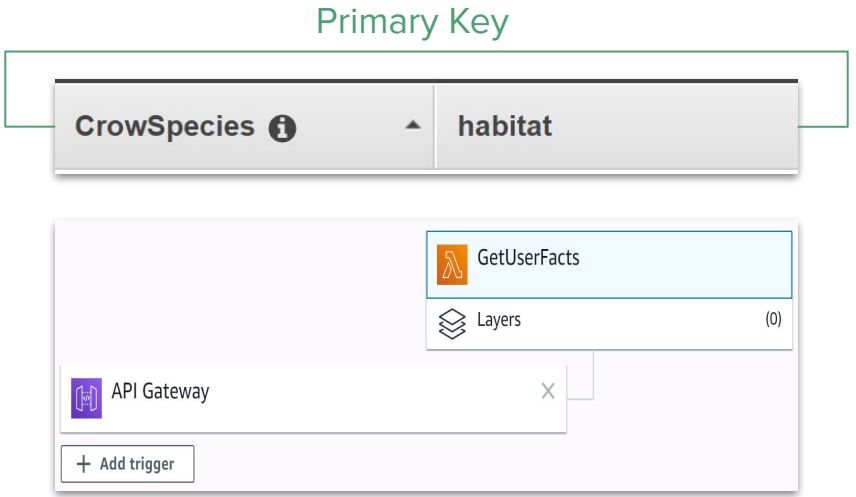
Add your own data

- User ability to make and share Crow Facts!
  - Input made-up fact.
  - Retrieve all entries (including new ones) in site.
  - Easter eggs :)

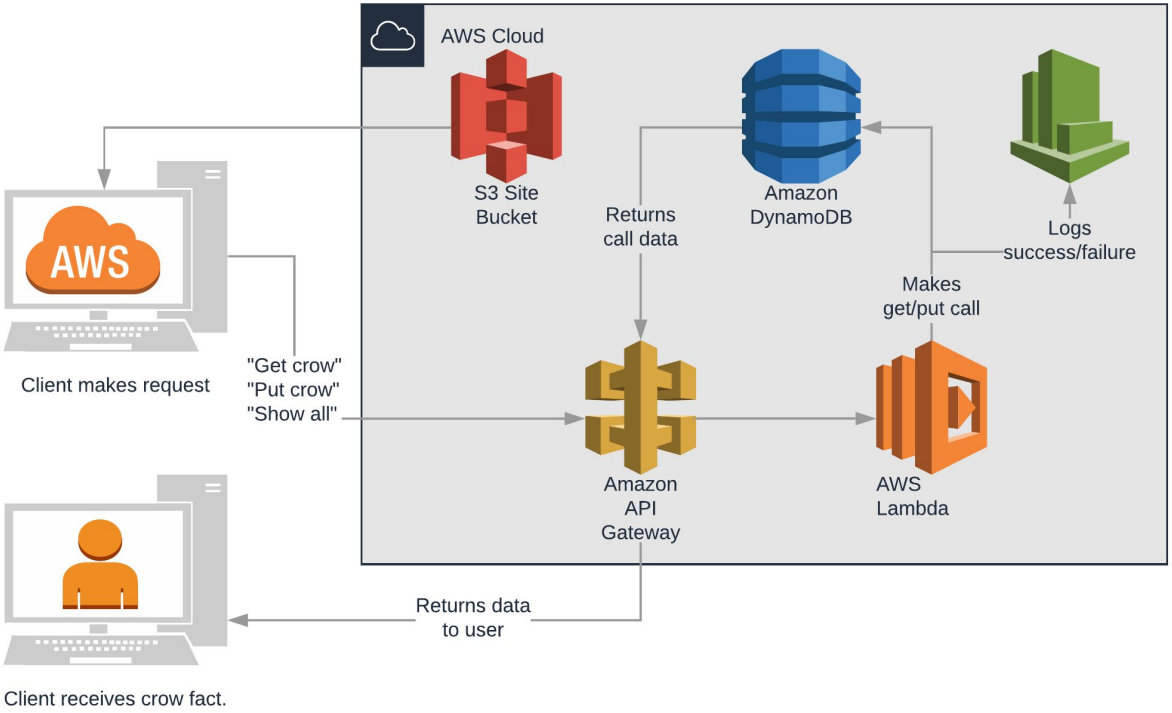


Design

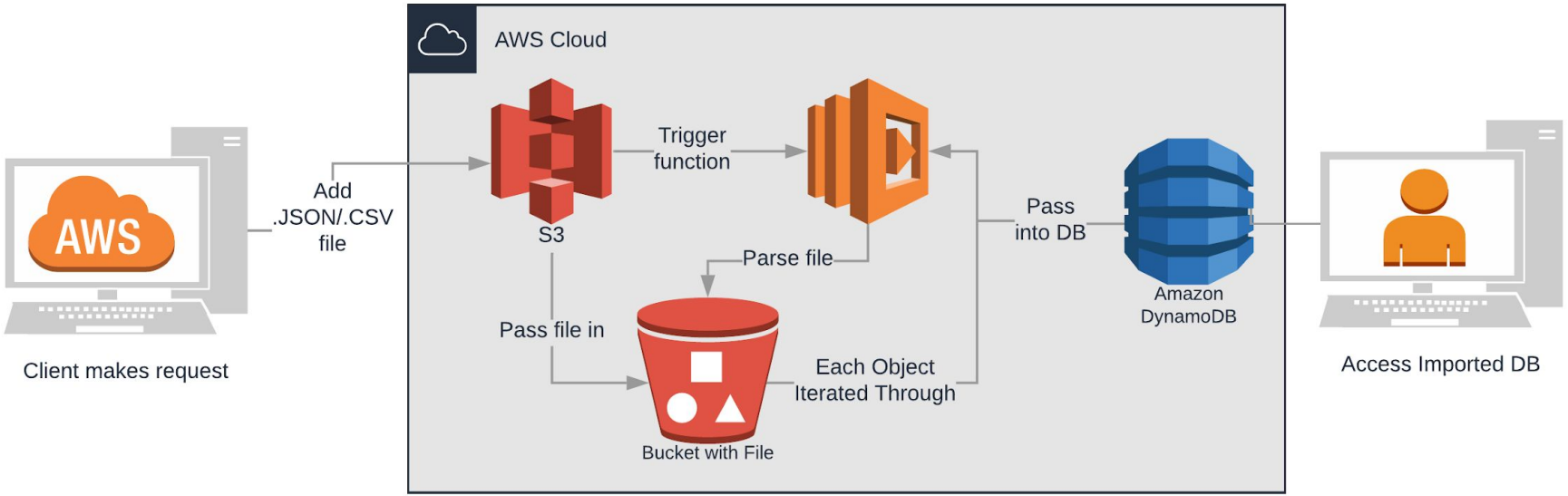
- Database - DynamoDB
  - Stores and sorts on species and habitat
- How to interact?
  - Let users interact via a website hosted on S3
  - Front-end can give species facts, or fun facts!
- How to send data from site to DB table?
  - Set up API call via Amazon API Gateway
  - Import and export data from S3 buckets
- How to have front-end and DB interact?
  - S3 buckets linked to DynamoDB table through API Gateway
  - Users create their crows from the webpage
  - Website handles sanitization of user inputs



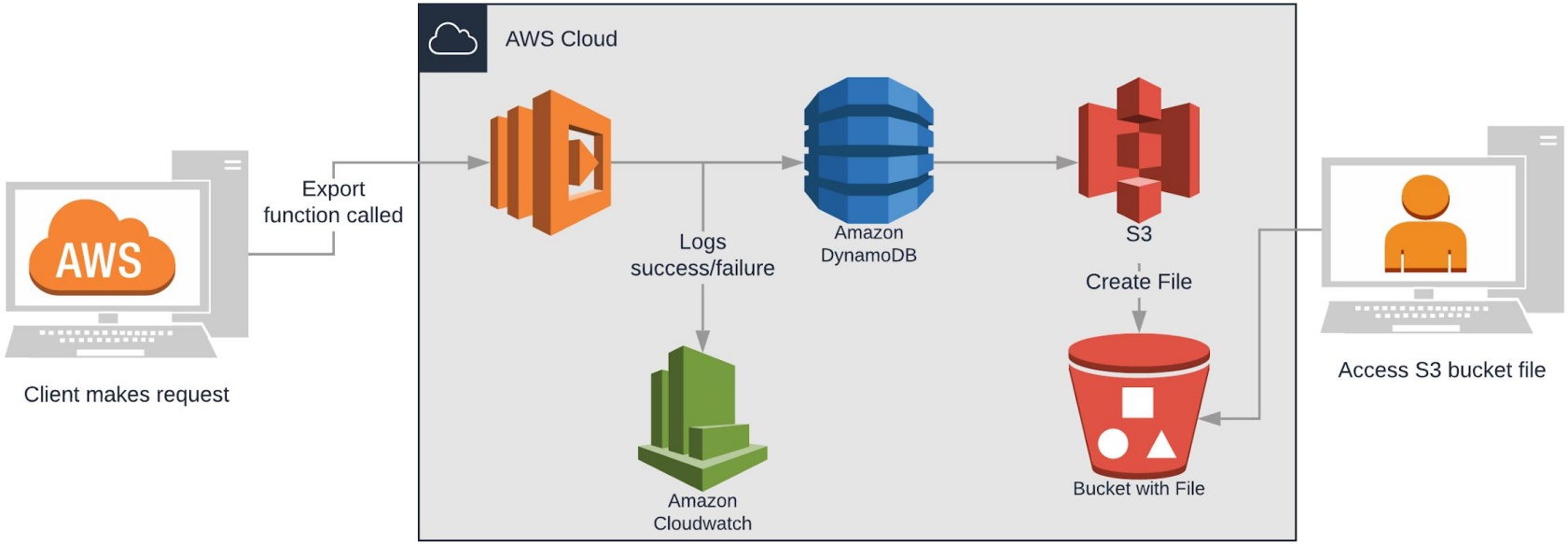
Implementation: Website



Implementation: Import



Implementation: Export



Try It Out!

Go to:

https://crowfacts.uwbhacks.com/

Try retrieving some facts, or putting in your own crow “facts” from the website!

This project has a detailed write-up at *https://docs.uwbhacks.com/crowfacts.html*

Code examples and configuration details are available in the GitHub repository: *https://github.com/UWB-ACM/crowfacts*

Closing Remarks

Other Cloud Services

- IAM & equivalents
  - Manage permissions for additional console users and service-to-service communication
- AWS IoT & equivalents
  - IoT device integration with cloud products
- Machine Learning
  - AWS Product Suite
  - GCP Product Suite
  - Azure Product Suite
- And so much more!

Where Do I Find Out More?

When in doubt, check out the documentation

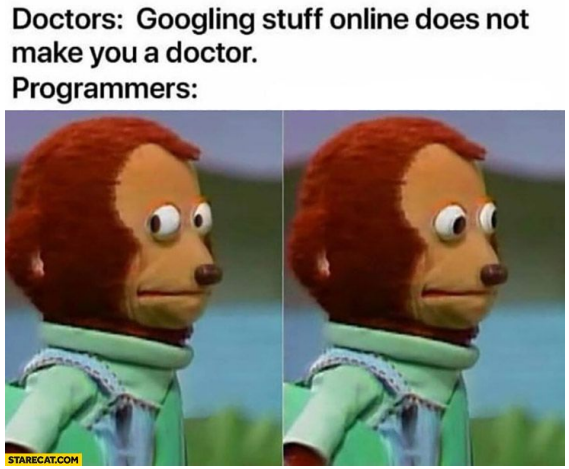
- AWS
- GCP
- Azure

Check out our documentation pages for this hackathon!

- https://docs.uwbhacks.com/
- This page includes this slide deck and other writeups.

Ask hackathon mentors on our Discord server!

Google, StackOverflow, tech blogs



Questions?

We wil answer all questions posted in the event’s Discord server.

Check out the #intro-to-cloud channel and post your questions there!

Thanks For Listening!