Wild Rydes : Serverless Workshop

Servers

(AAHHHHHHHHH!!)

How should my app withstand a server **failing**?

How can I tell if a server has been **compromised**?

How can I increase **utilization** of my servers?

Which **OS** should my servers run?

How much remaining **capacity** do my servers have?

How should I implement dynamic **configuration changes** on my servers

When should I decide to **scale up** my servers?

What size servers are right for my budget?

How will I keep my server OS **patched**?

How can I control **access from** my servers?

Which packages should be baked into my **server images**?

How will new code be **deployed** to my servers?

How will the application handle server **hardware failure**?

How many users create **too much load** for my servers?

What size server is right for my **performance**?

How many servers should I budget for?

Which users should have **access to** my servers?

Should I **tune OS settings** to optimize my application?

When should I decide to **scale out** my servers?

Owning servers means dealing with ...

Operations and management

Scaling

Provisioning and utilization

Availability and fault tolerance

What to expect from this workshop

- Serverless Architectures Overview – BRIEF!
    - Overview of
        - AWS Lambda
        - Amazon API Gateway
        - Amazon DynamoDB
        - Amazon Cognito
- Workshop Breakout – Time to build

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

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

AWS Lambda

Benefits of using Lambda

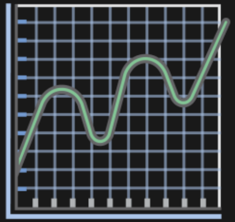
1

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, C#
- 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

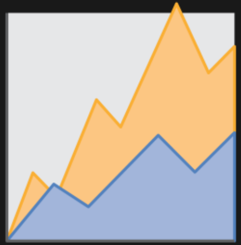
Flexible invocation paths

- 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 party 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?

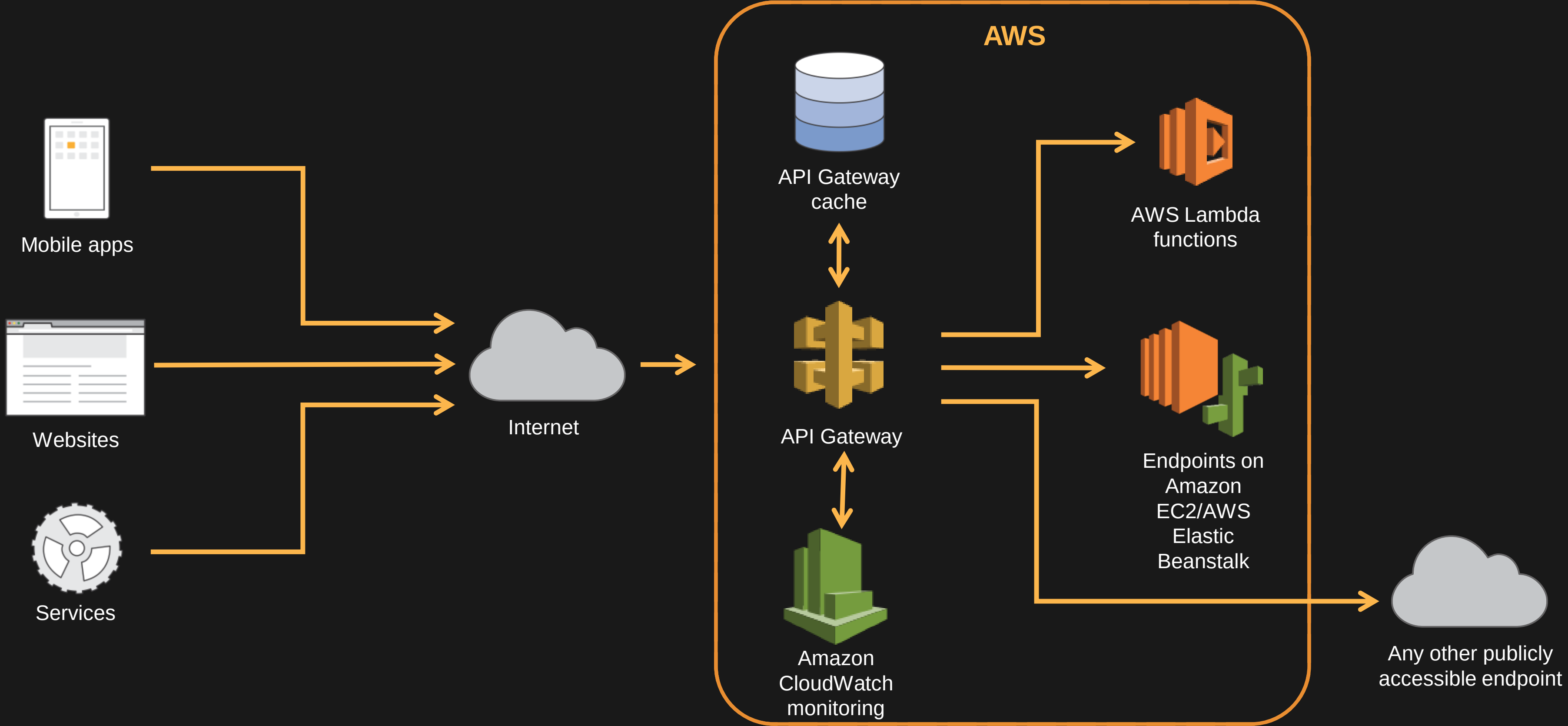
API Gateway - Capabilities

- 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


Mobile apps
Websites
Services
Internet
AWS
API Gateway cache
API Gateway
Amazon CloudWatch monitoring
AWS Lambda functions
Endpoints on Amazon EC2/AWS Elastic Beanstalk
Any other publicly accessible endpoint

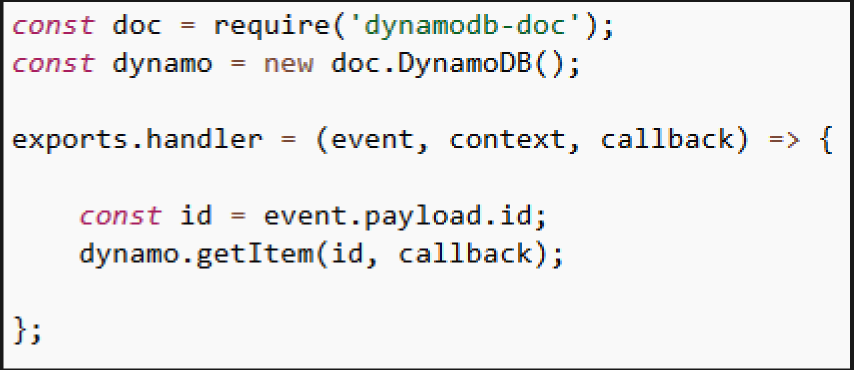
Amazon DynamoDB

Amazon DynamoDB

Fast and flexible NoSQL database service
for any scale

Dead Simple

- GetItem(primaryKey)
- PutItem(item)



```
const doc = require('dynamodb-doc');
const dynamo = new doc.DynamoDB();

exports.handler = (event, context, callback) => {

    const id = event.payload.id;
    dynamo.getItem(id, callback);

};
```

Robust Depth

- Fine-Grained Access Control
- Streams
- Triggers
- Cross-Region Replication
- DynamoDB local
- Free-text search
- Titan Graph Database integration
- Strong consistency option
- Atomic counters

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.

SAML
Your own auth
Guest

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

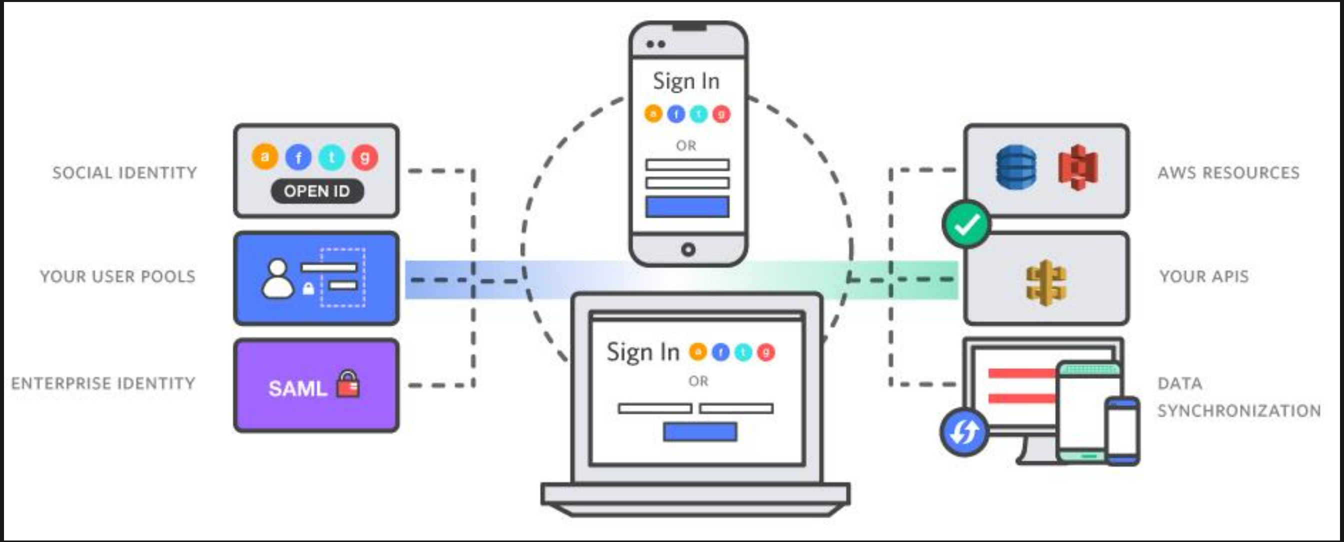
A simple, secure, low-cost, and fully managed service to create and maintain a user directory that scales to 100s of millions of users

3

Enhanced Security Features

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

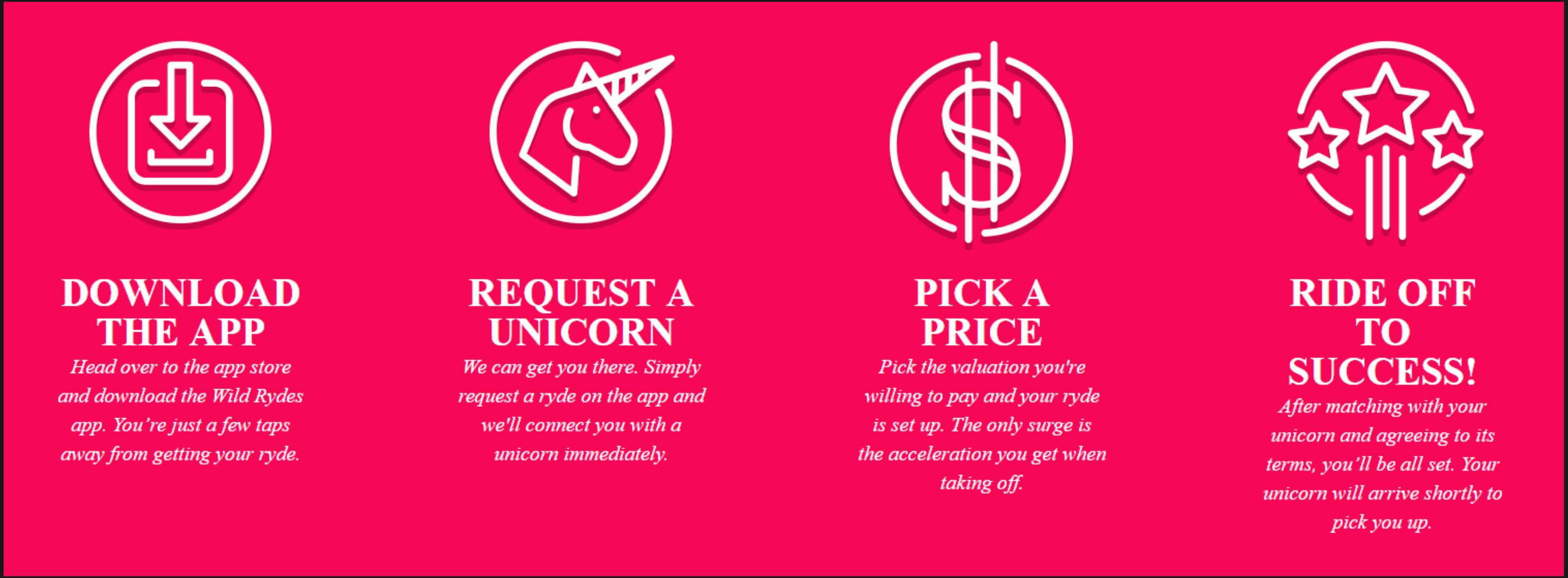
Comprehensive Support for Identity Use Cases



Scenario: Wild Rydes (www.wildrydes.com)

Help Wild Rydes Disrupt Transportation!

So how does this magic work?



DOWNLOAD THE APP

Head over to the app store and download the Wild Rydes app. You're just a few taps away from getting your ryde.

REQUEST A UNICORN

We can get you there. Simply request a ryde on the app and we'll connect you with a unicorn immediately.

PICK A PRICE

Pick the valuation you're willing to pay and your ryde is set up. The only surge is the acceleration you get when taking off.

RIDE OFF TO SUCCESS!

After matching with your unicorn and agreeing to its terms, you'll be all set. Your unicorn will arrive shortly to pick you up.

Wild Rydes is Backed by Leading Investors

THE BARN ACCELERATOR

TENDERLOIN CAPITAL

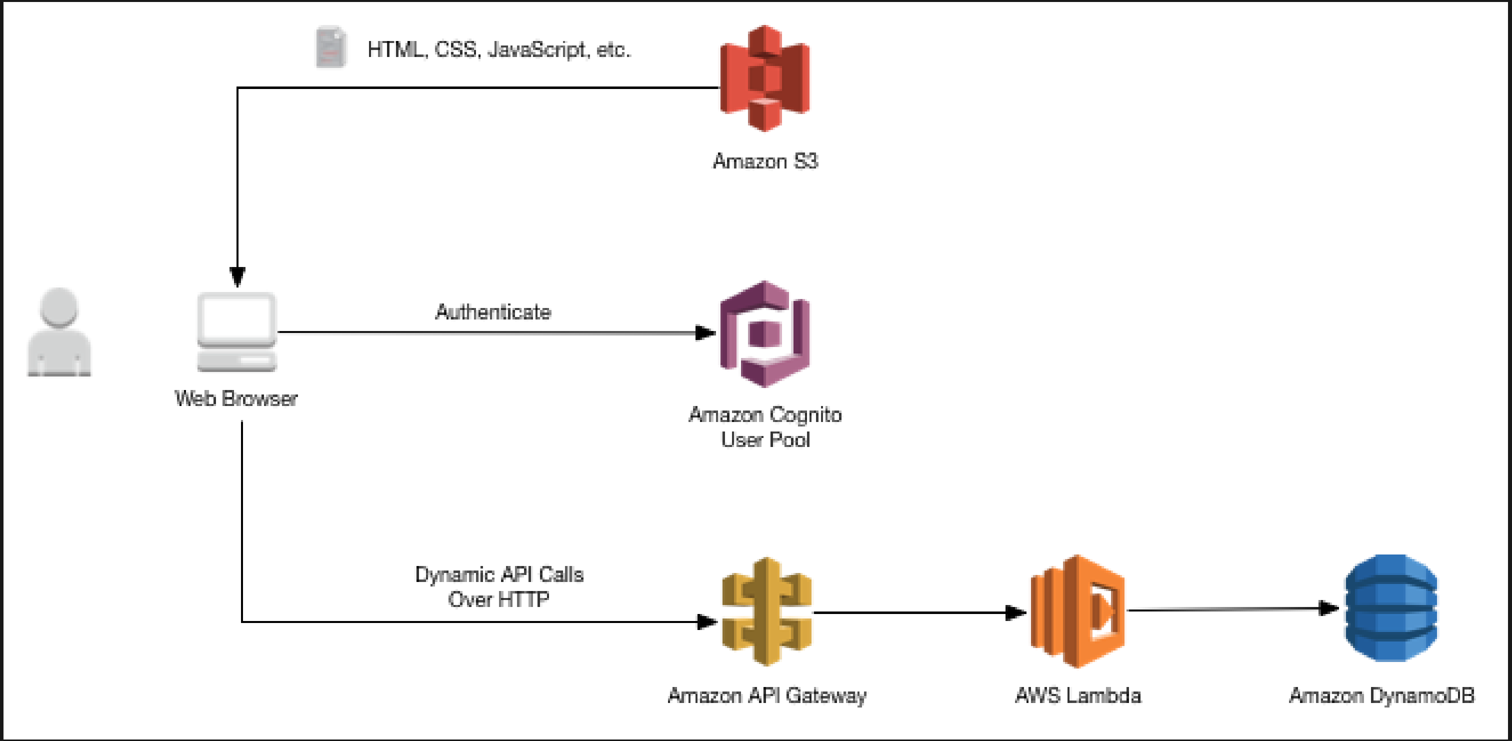
PENGLAI COMMUNICATIONS AND POST NEW CENTURY TECHNOLOGY CORP LIMITED

Your Task: Build the Wild Rydes Website

Welcome to Wild Rydes Inc.,
Employee #3!

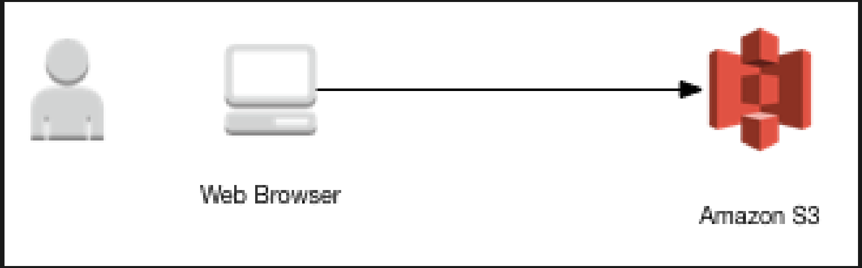
Scenario: Wild Rydes

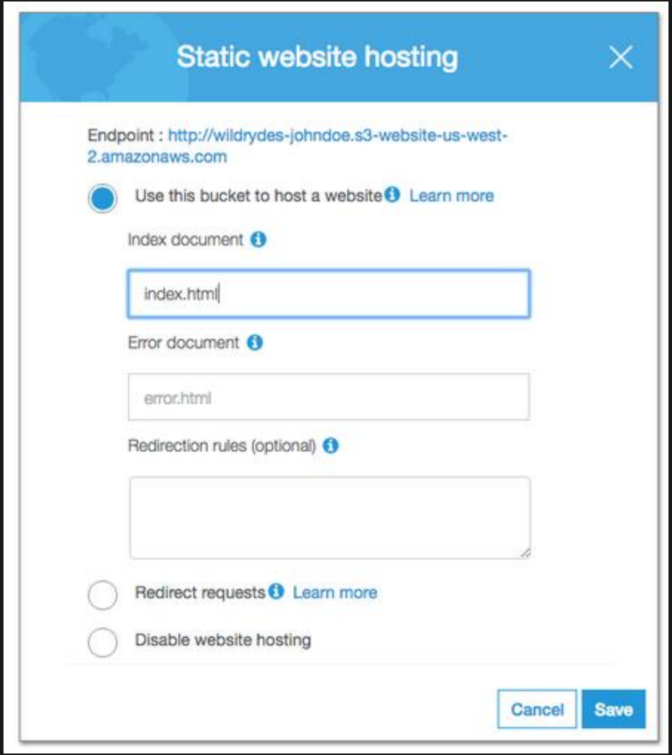
The Wild Rydes Serverless Web Application Workshop introduces the basics of building web applications using serverless infrastructure.



Lab 1: Static Website Hosting

OBJECTIVE: Create a bucket in Amazon S3 and configure it for static website hosting. The static HTML, JS, and CSS will be served directly to user browsers from Amazon S3.





Endpoint : http://wildrydes-johndoe.s3-website-us-west-2.amazonaws.com

[✓ Use this bucket to host a website] Learn more

Index document: index.html

Error document: error.html

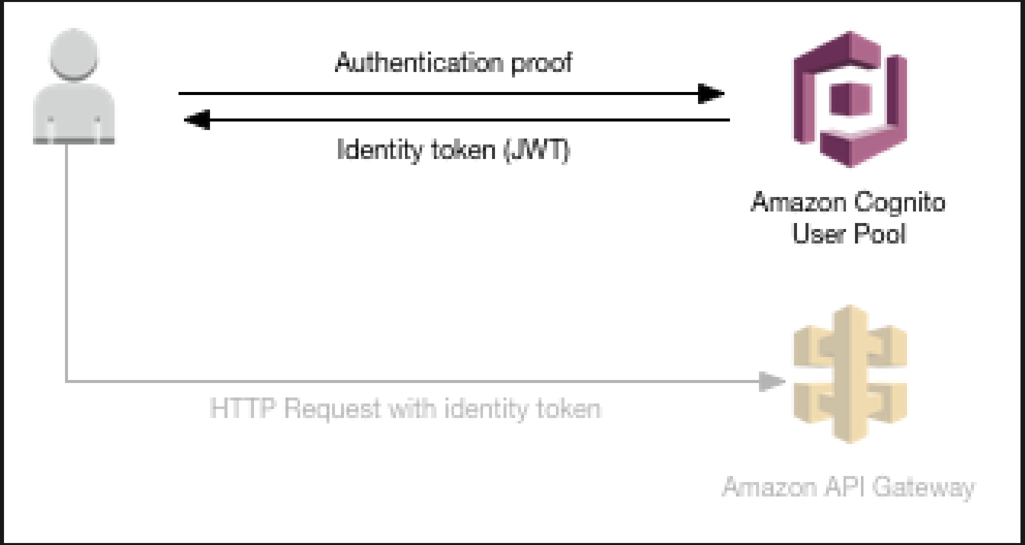
Redirection rules (optional):

Redirect requests Learn more

Disable website hosting

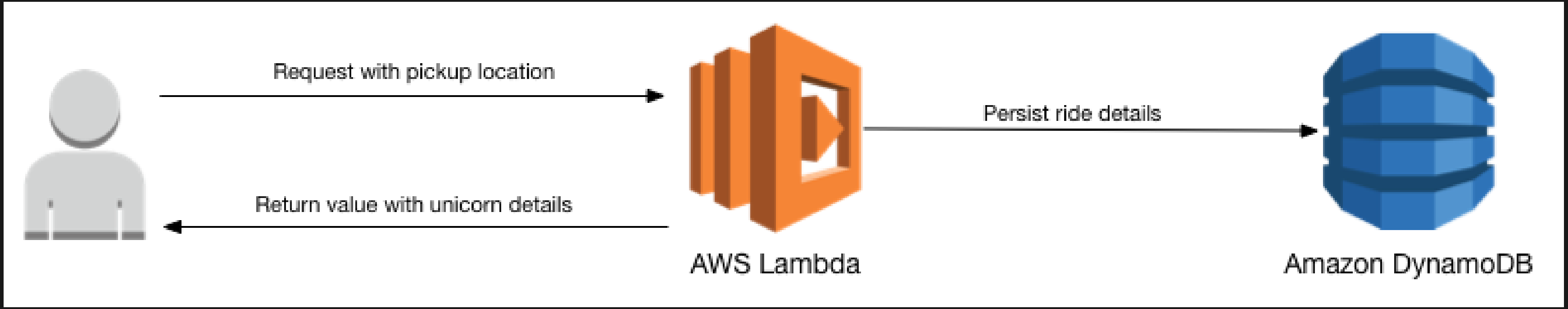
Lab 2: User Management

OBJECTIVE: Allow visitors to register as a new user on Wild Rydes, by providing and validating their email address. Amazon Cognito will be used to manage the User Pool for Wild Rydes.



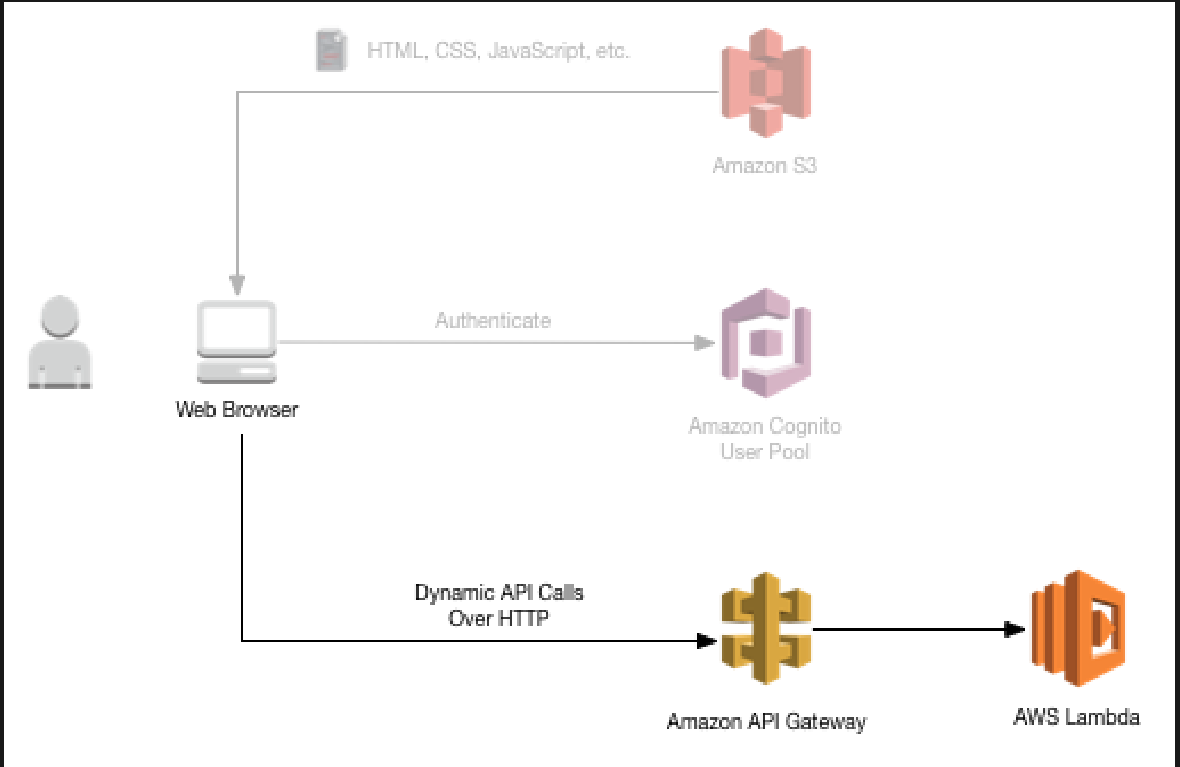
Lab 3: Serverless Service Backend

OBJECTIVE: Create a service backend using AWS Lambda and Amazon DynamoDB to handle requests from your frontend static website content.


Request with pickup location
Return value with unicorn details
AWS Lambda
Persist ride details
Amazon DynamoDB

Lab 4: Create RESTful API

OBJECTIVE: Use API Gateway to expose the Lambda function you built in the previous module as a RESTful API



https://github.com/roryp/wildrydes