FIREBASE FUNCTIONS, CALLBACKS, CREATING AND TESTING OUR FIRST FUNCTIONS

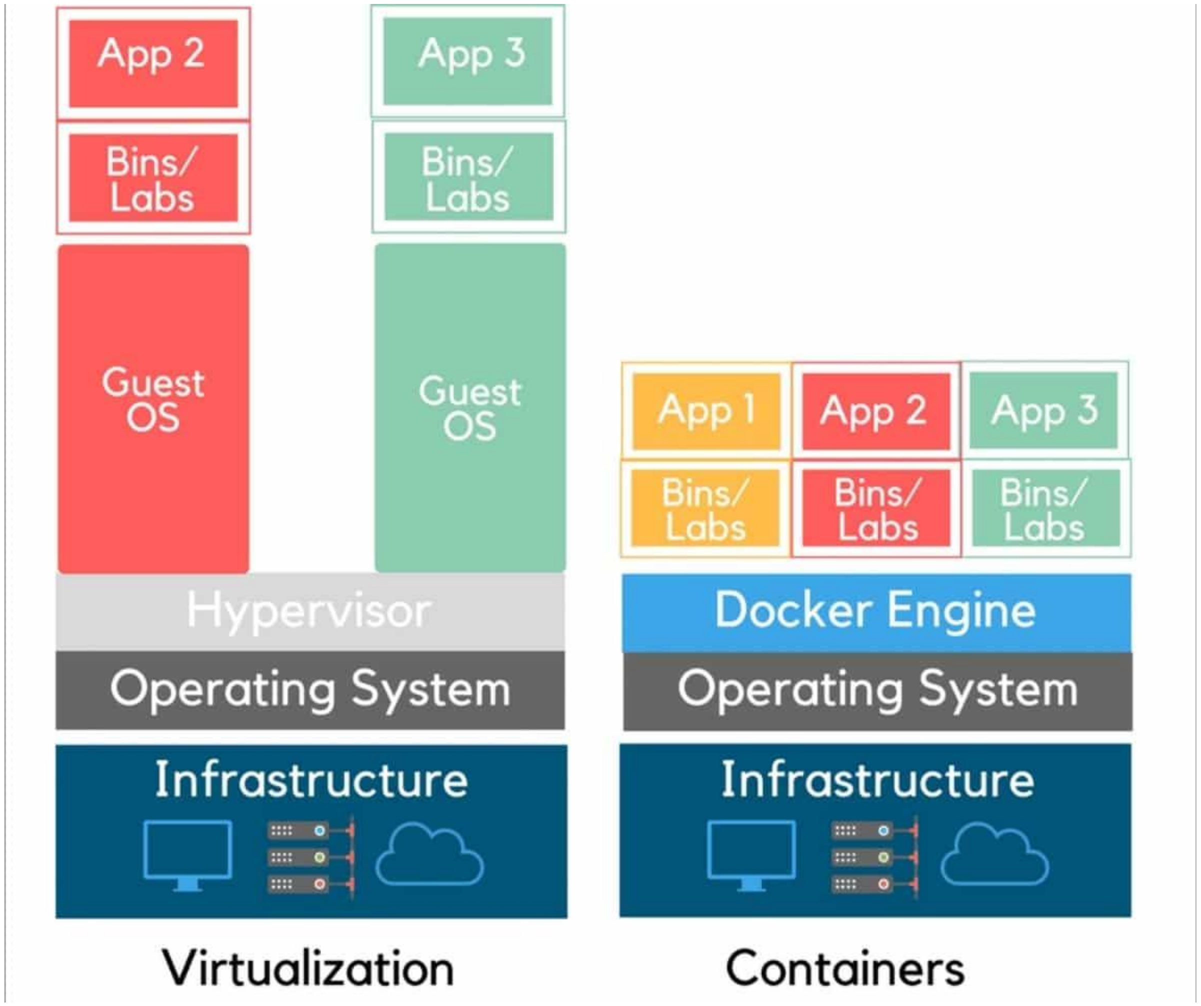
Summary

- Firebase functions
- Callback functions
- HTTP Verbs GET and POST
- Creating a dumb function which receives data and returns it back
- Testing with POSTMAN
- JSON
- Summary

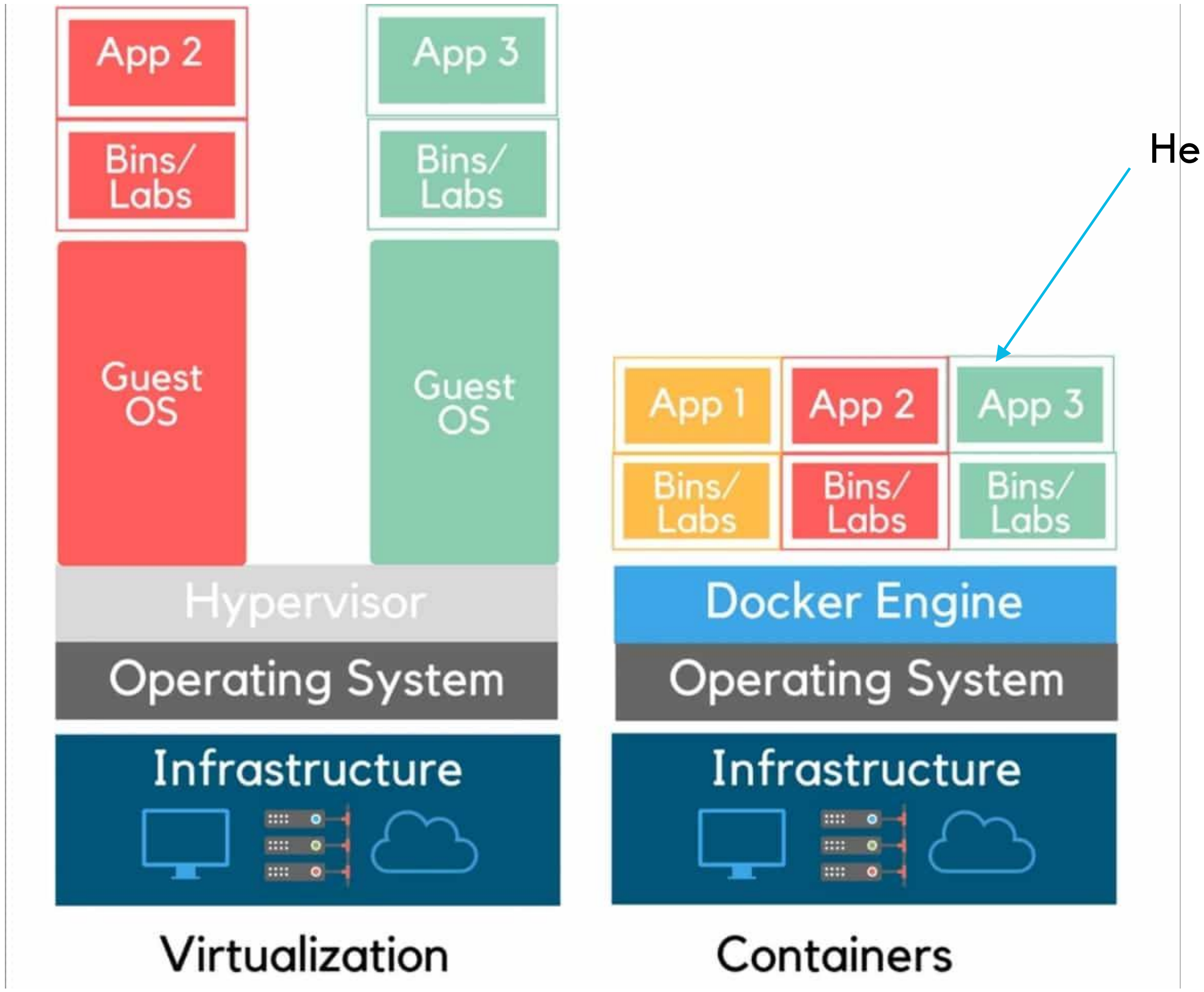
Firebase Functions

- It is a compute service that lets you run code without provisioning or managing servers.
- Similar to AWS Lambda, Azure Functions… It runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- You pay only for the compute time you consume - there is no charge when your code is not running.

Containers vs VMs



Deploy a function what happens



Callback functions

Too busy to take a call, please leave your name and number (**function**) and I'll call you back when I'm **finished** my work (**invoke your function**).

Function callbacks

- Examining our first REST API (cloud function)

```
const functions = require('firebase-functions');

// // Create and Deploy Your First Cloud Functions
// // https://firebase.google.com/docs/functions/write-firebase-functions
// Callback function -- please run this code for me Firebase when a request is made

exports.helloWorld = functions.https.onRequest((request, response) => {
  functions.logger.info("Hello logs!", {structuredData: true});
  response.send("Hello from Firebase!");
});
```

myCoolApp/functions/index.js

Exercise: Deploying a function

- Deploy a function onto Firebase and return a string when it is invoked which says “Welcome to my cool new backend function”

- Add a second function below the first one and this time return a message saying “Not logged In!”

Passing Data - URL Query String

- We can encode parameters in the URL, we generally refer to this as the query string

- E.g. If you run a google search query, look at the URL after you search…

- http://www.mywebsite.com?data=hello

Simple function to mirror data

- Assume you want to create a function which parses out data submitted per request and mirrors it back to the requester

```
const functions = require('firebase-functions');

// Accept comment and return the same comment to the user

exports.echofunction = functions.https.onRequest((request, response) => {
    response.send(request.query.data);
});
```

myCoolApp/functions/index.js

Exercise: Parsing params from QS

- Write a function that assumes the data passed in the via the Query String is a number, take the value that is submitted per request, double it and then return it via the response.

- If the user submits a value that isn't a number then return a response that says "Please Enter a Number"

HTTP Verbs (GET and POST)

- GET and POST are HTTP request methods to transfer data from client to server.
- So far we have been making GET requests, GET is designed to request data from a specified resource
- POST is designed to submit data to the specified resource
- Both can be used to send requests and receive responses

Request Structure

- All HTTP requests have a three main parts
  - Request line
    - HTTP Method (GET, POST, etc.)
    - URL – address of the resource that is being requested
    - HTTP version
  - Headers
    - Additional information passed between the browser and the server, i.e. cookies, browser version, OS version, auth tokens, content-type
  - Message body
    - Client and server use the message body to transmit data back and forth between each other. POST request method will usually have data in the body. GET requests leave the message data empty

GET Method

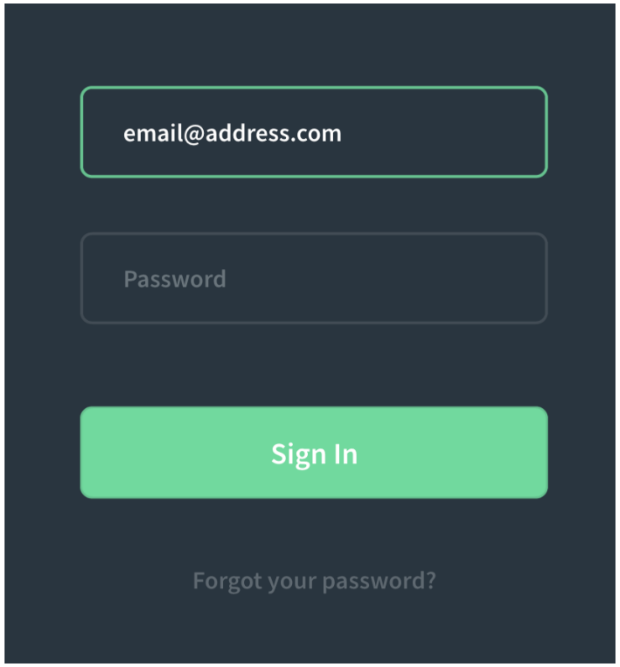
- GET requests can be cached
- GET requests remain in the browser history (you can go back!)
- GET can't be used to send binary data, like images or word documents to the server
- GET requests can be bookmarked
- GET requests have length restrictions
- GET requests should only be used to retrieve data
- Using GET data can be sent to the server by adding name=value pairs at end of the URL, i.e. Querystring
  - `mysite.app.web…/page?id=101&name=John`

POST Method

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length
- The POST method can be used to send ASCII as well as binary data

POST Request Form Data

- Form data is often sent to the server via a POST request



Alternative option is to send username and password to the server via the QueryString – uid=email@address.com pwd=password

But is this a good idea?

GET vs POST

- Use GET if you are requesting a resource
  - You may need to send some data to get the correct response back, but in general the idea is to GET a resource

- Use POST if you want to send data to the server
- Other methods
  - PUT //Update/Replace
  - DELETE //Delete
  - PATCH //Partial update/modify

POST data to server

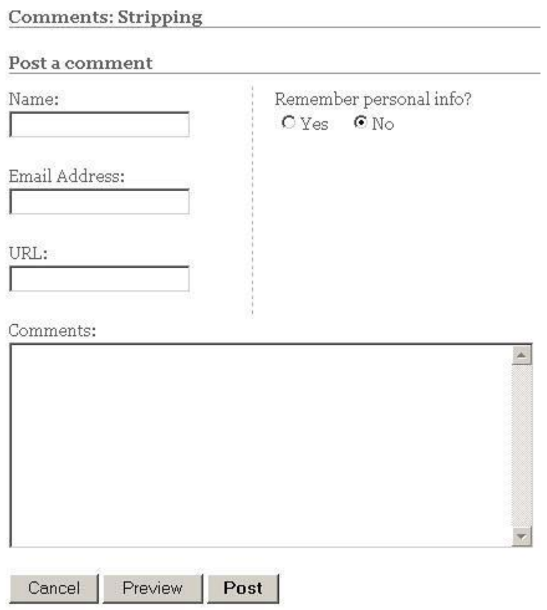
- Assume you are building a form which posts comments from your blog page to the server
- First step is to write a function to accept the comment

```
const functions = require('firebase-functions');

// Accept comment and return the same comment to the user

exports.postcomment = functions.https.onRequest((request, response) => {
    response.send(request.body);
});
```

myCoolApp/functions/index.js

How to test a POST request?

- We can create a form on a web page, then write JavaScript to send the data via POST to the server.
- However it would be nice if there was a way to test it first without having to go back to the frontend



Comments: Stripping

Post a comment

Name:

Remember personal info?
Yes [✓ No]

Email Address:

URL:

Comments:

Postman client

- When writing backend APIs such as the one we have just completed, it’s often necessary to test it quickly.
- You don’t want to have to write a client side request to test each API. Sometimes you may even want to pass in values which would take even longer to code up.
- Postman can help!
- https://www.postman.com/downloads/

POSTMAN

- It's brilliant for letting us test our APIs without having to write client side code to make the requests.
- It will work for all request methods, i.e. GET, POST, PUT etc.

- You can code the backend independent of the frontend!
- How else could we test to see if postcomments is working!

Sending data, what format?

- Now that we have an API available to receive data, and we have a client (postman) willing to send the data, we need to decide on a data format…

- Enter JavaScript Object Notation or JSON

JSON

- JavaScript Object Notation (JSON)
- It is an open, human and machine-readable standard that facilitates data interchange
- Along with XML it is the main data interchange format on the web
- Data types
  - Numbers, Strings, Booleans, Arrays, Objects
  - ISODate() returns a date object
- Firestore uses JSON documents to store records of information

JSON cont.

- Arrays
  - [“a”, “b”, “c”, “d”, “e”, “f”]
  - [“apple”, 3, null, true]
- Objects
  - {“Enda” : 45, “John” : 33, “Sam” : “Smith”}
- Array of Objects
  - [{}]
- Use double quotes, no comma last value

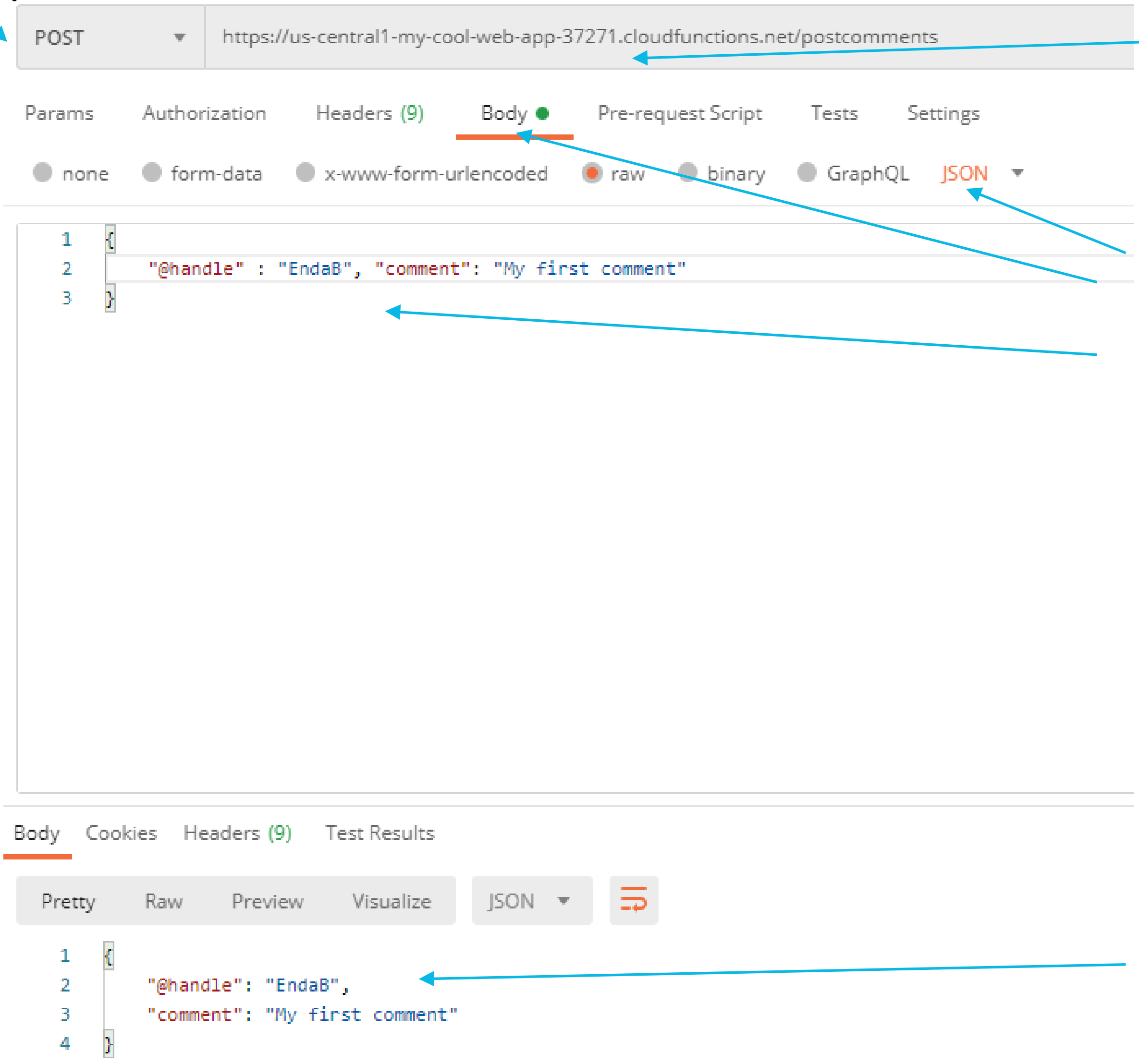
POST JSON to Server

Set POST

Request to our API

Pass JSON data in the request body

Response



https://us-central1-my-cool-web-app-37271.cloudfunctions.net/postcomments

```
{
    "@handle" : "EndaB", "comment": "My first comment"
}
```

```
{
    "@handle": "EndaB",
    "comment": "My first comment"
}
```

Exercise 3

- Create a function which accepts comment information (JSON formatted) in the body of the request i.e. {“Comment”: “This is my comment”}
- Make a request via Postman to send the data via a POST request
- Respond with a message saying “I received your comment, thank you”.

Summary

- Firebase functions
- Callback functions
- HTTP Verbs GET and POST
- Creating a dumb function which receives data and returns it back
- Testing with POSTMAN
- JSON
- Summary