Xebia

Xpirit

Showdown of the serverless cloud, orchestrating functions

choose your alliance, go to:
https://presentain.com/showdown

Xebia

Xpirit

Kenny Baas-Schwegler

Software Consultant - Domain-Driven Design

@kenny_baas

Baasie.com

xebia.com/blog/author/kbaas/

Marc Duiker

Azure Consultant - I ❤ Serverless

@marcduiker

blog.marcduiker.nl

Xebia

Xpirit

Why do I like serverless?

Xebia

Xpirit

Why do I like serverless?

Xebia

Xpirit

Xebia

Xpirit

Let's look at the specs!

Xebia

Xpirit

Comparing Cost*

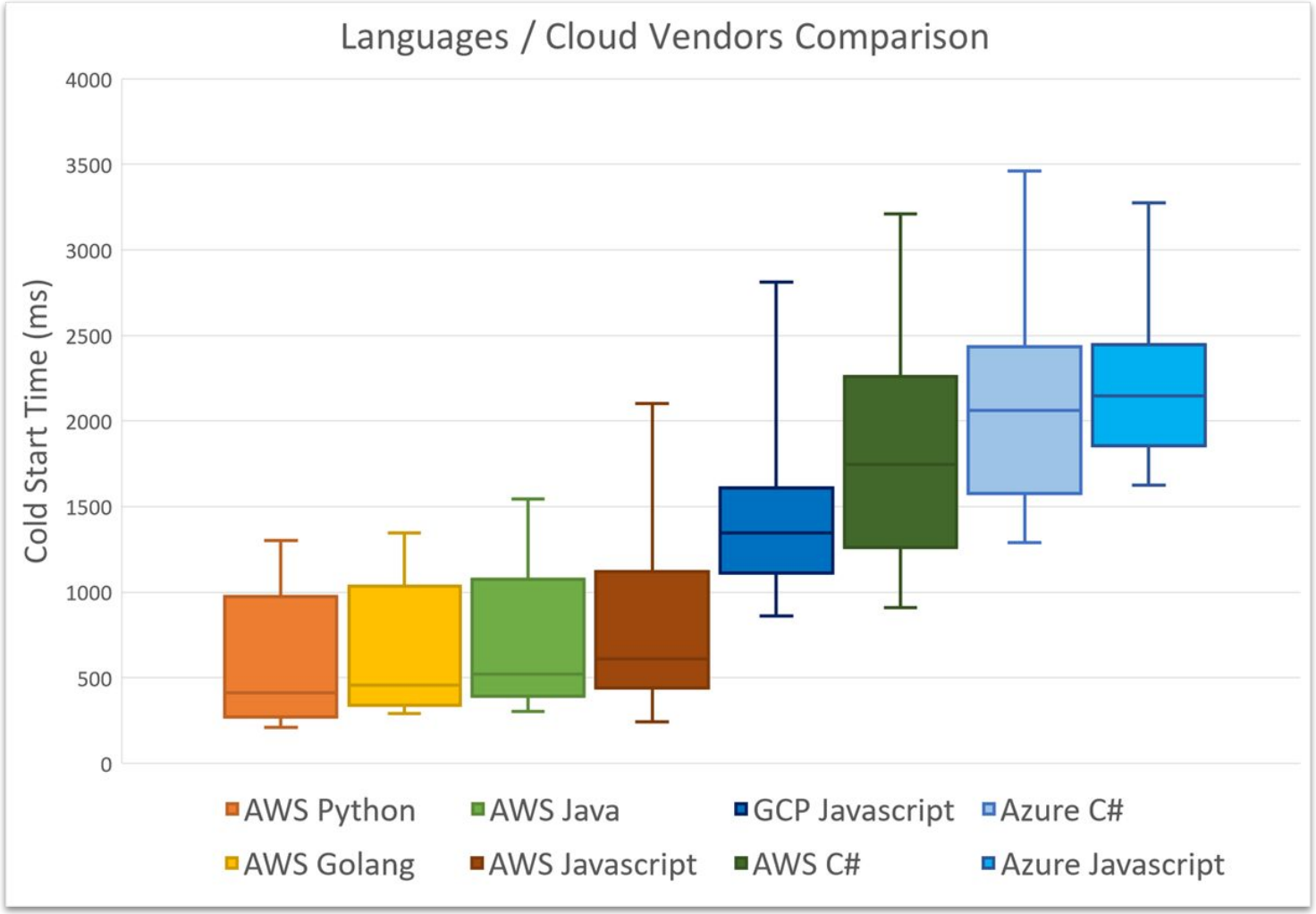
Type	AWS Lambda	Azure Functions	Google Cloud Functions
Price 1M Executions ($)	0.20	0.20	0.40
Price GB-s ($)	0.00001667	0.000016	0.0000025
Price GHz-s ($)	-	-	0.0000100
Free executions/month	1M	1M	2M
Free GB-s/month	400K	400K	400K
Free GHz-s/month	-	-	200K
Total cost/month ($)	**78.48**	**75.40**	**92.70**

* For 10M function executions of 1 sec each using a 512MB / 800MHz machine.

Comparing Runtimes

Runtimes	AWS Lambda	Azure Functions	Google Cloud Functions
Node.js	v6 & v8	v6, v8, v10	v6 & v8 (beta)
.NET Framework	-	v4.7	-
.NET Core	v1, v2.x	v2.x	-
Python	v.2.7, v3.6 & v3.7	v3.6 (preview)	v3.7.1 (beta)
Java	v8	v8 (preview)	-
Ruby	v2.5	-	-
Go	v1.x	-	v1.11(beta)

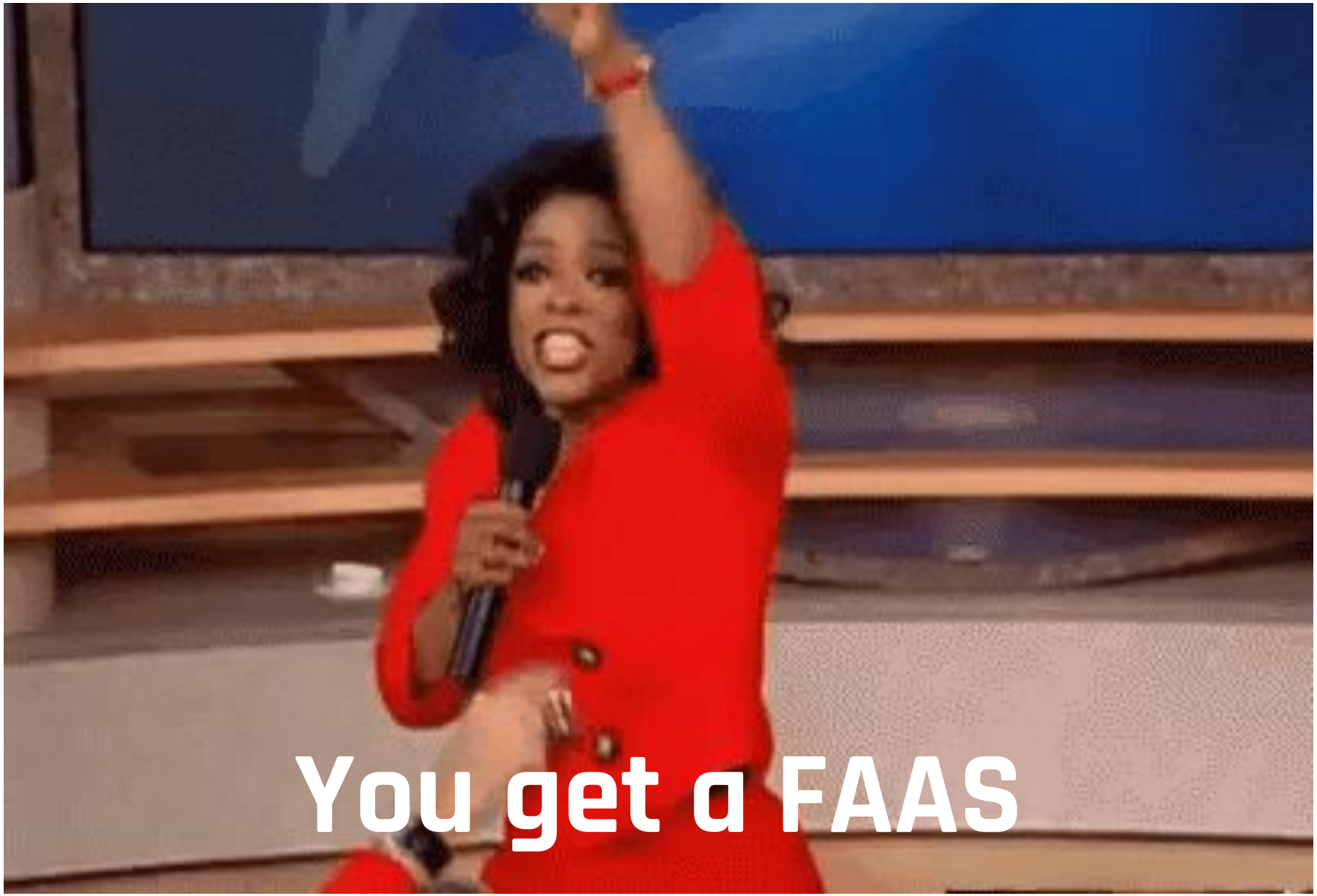
Comparing Cold Start



Source: https://mikhail.io/2018/08/serverless-cold-start-war/

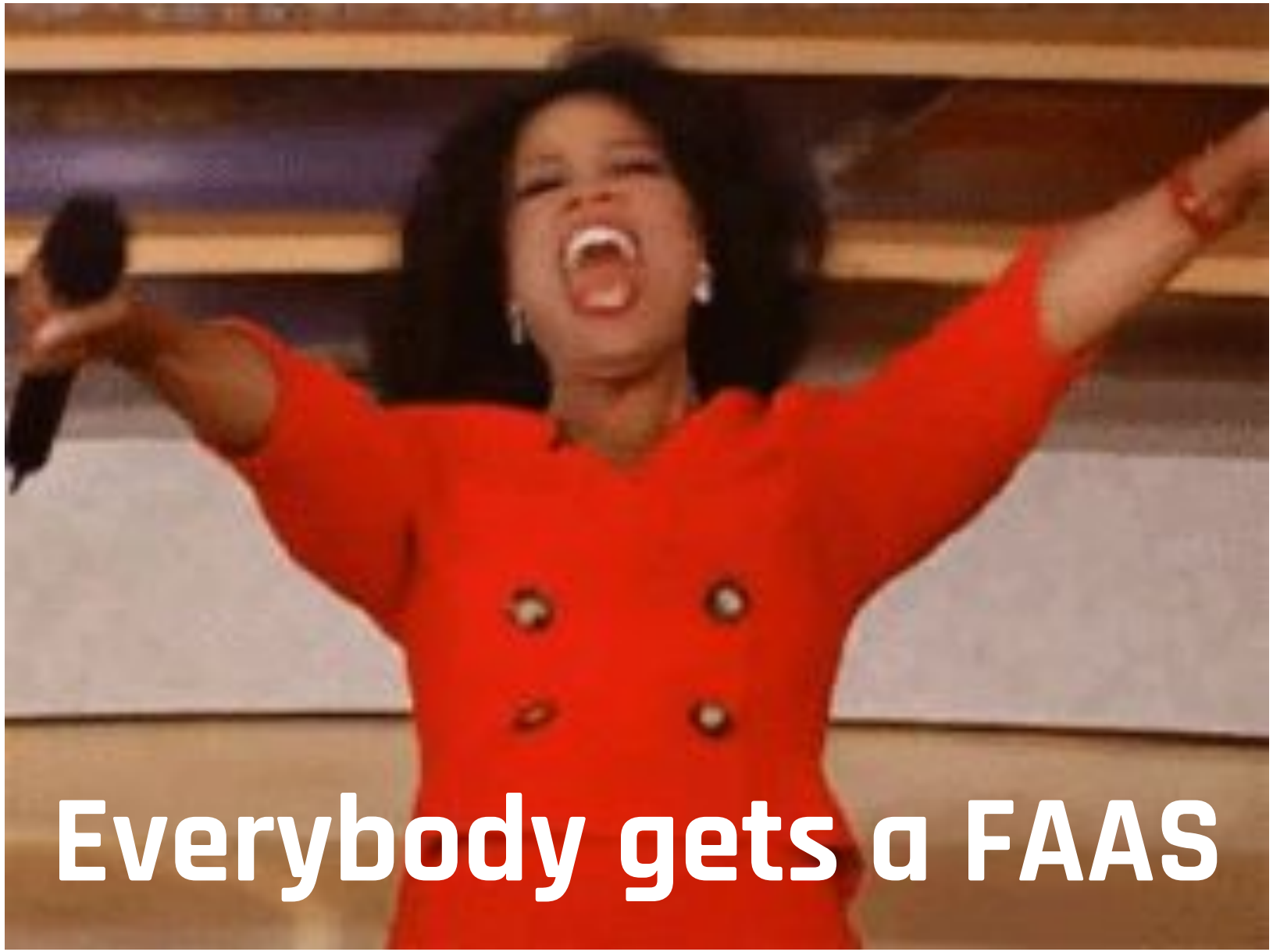

Xebia
Xpirit

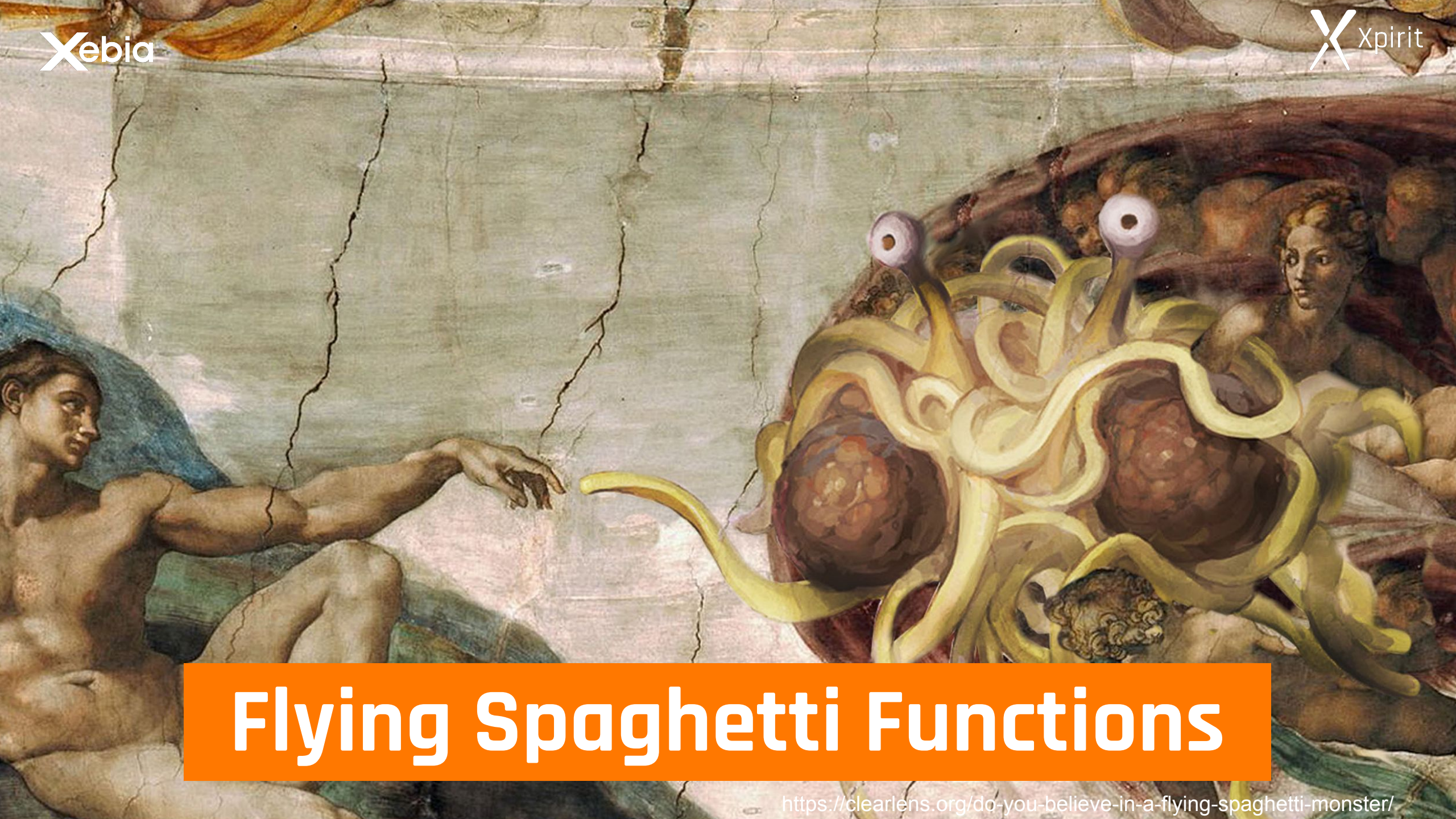

Xebia
Xpirit
UITRIT WIBAUTVESTE
INRIT WIBAUTVESTE
Parking Garage Use Case

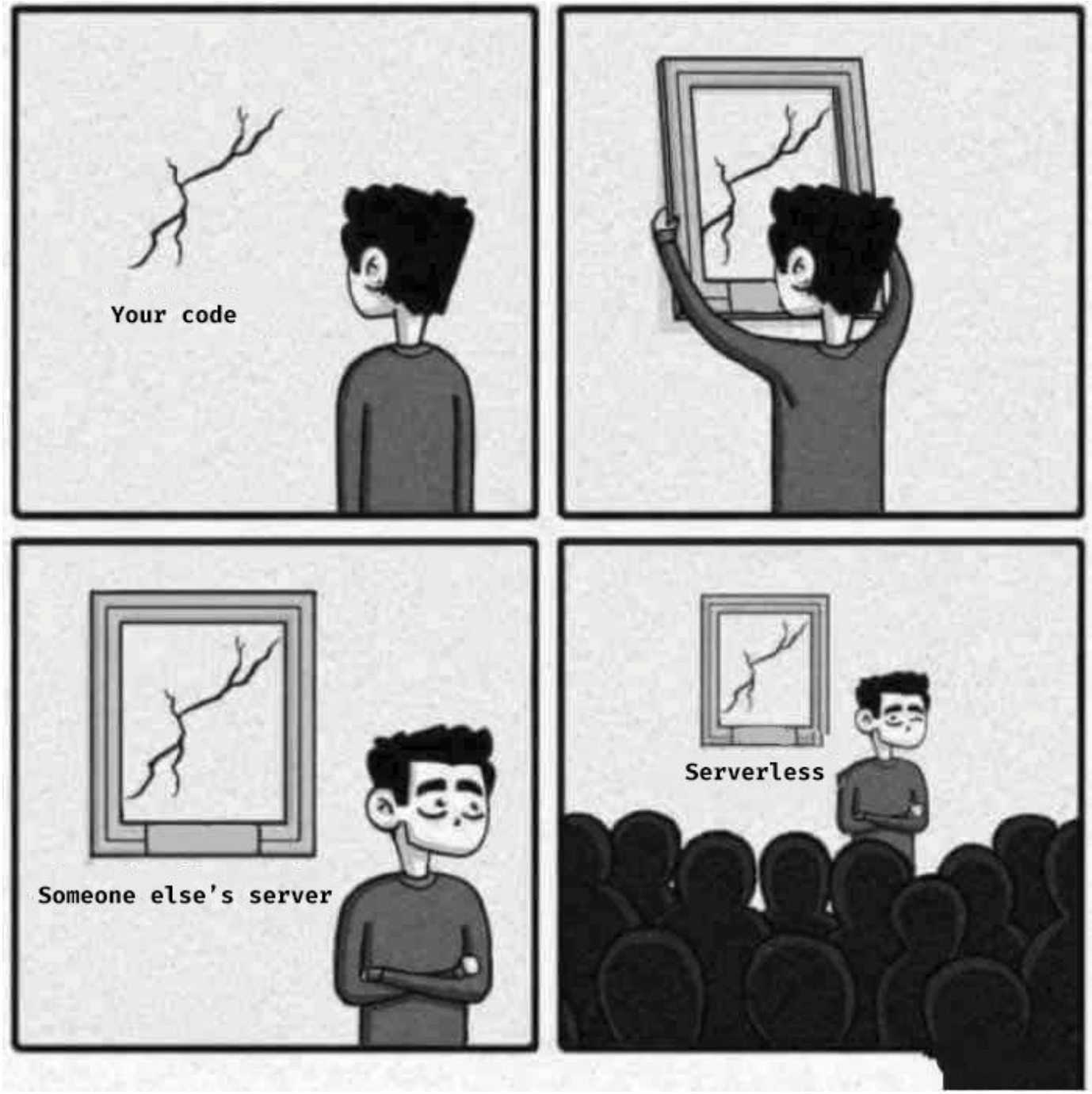

You get a FAAS

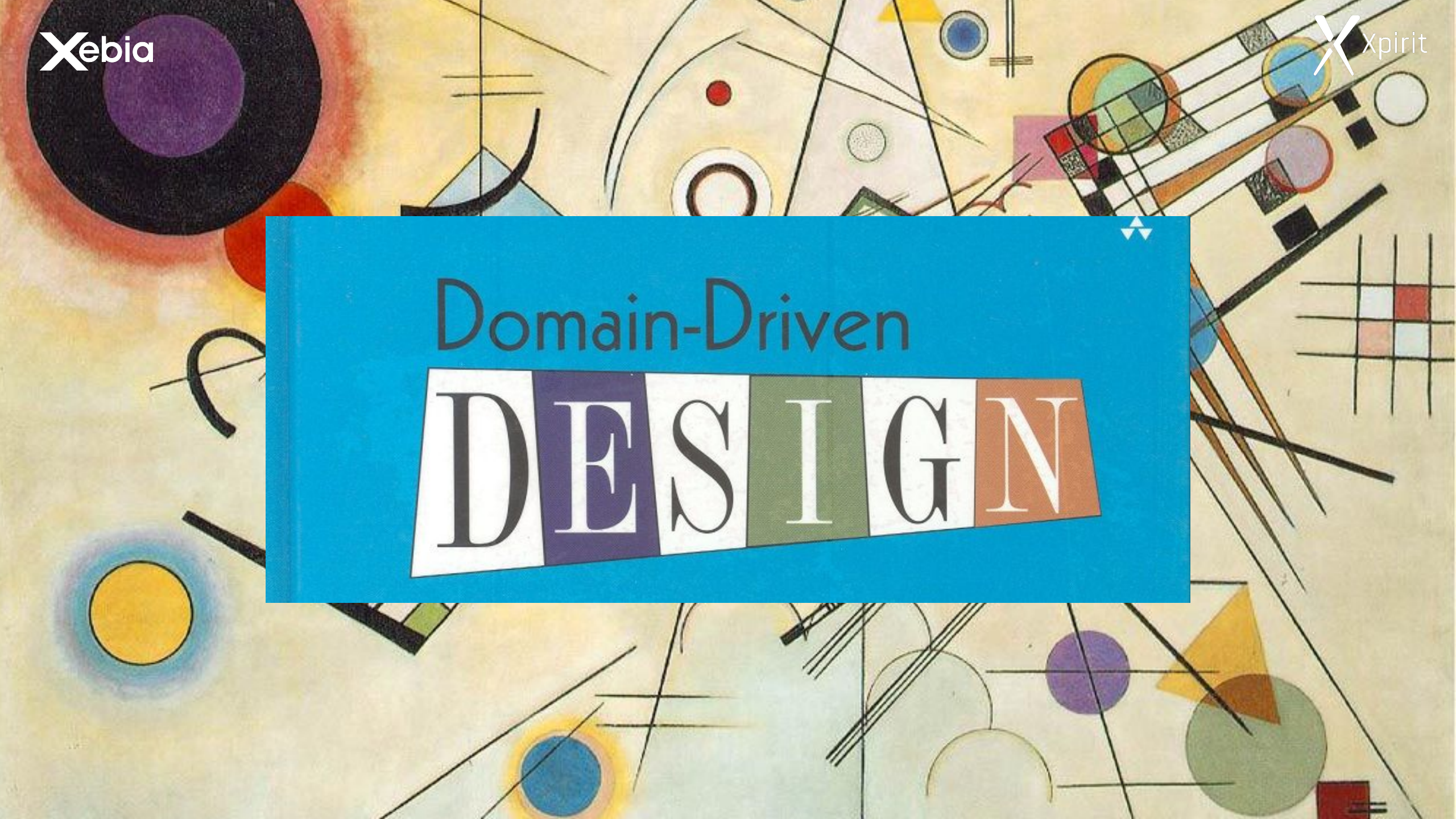
Xebia

Xpirit

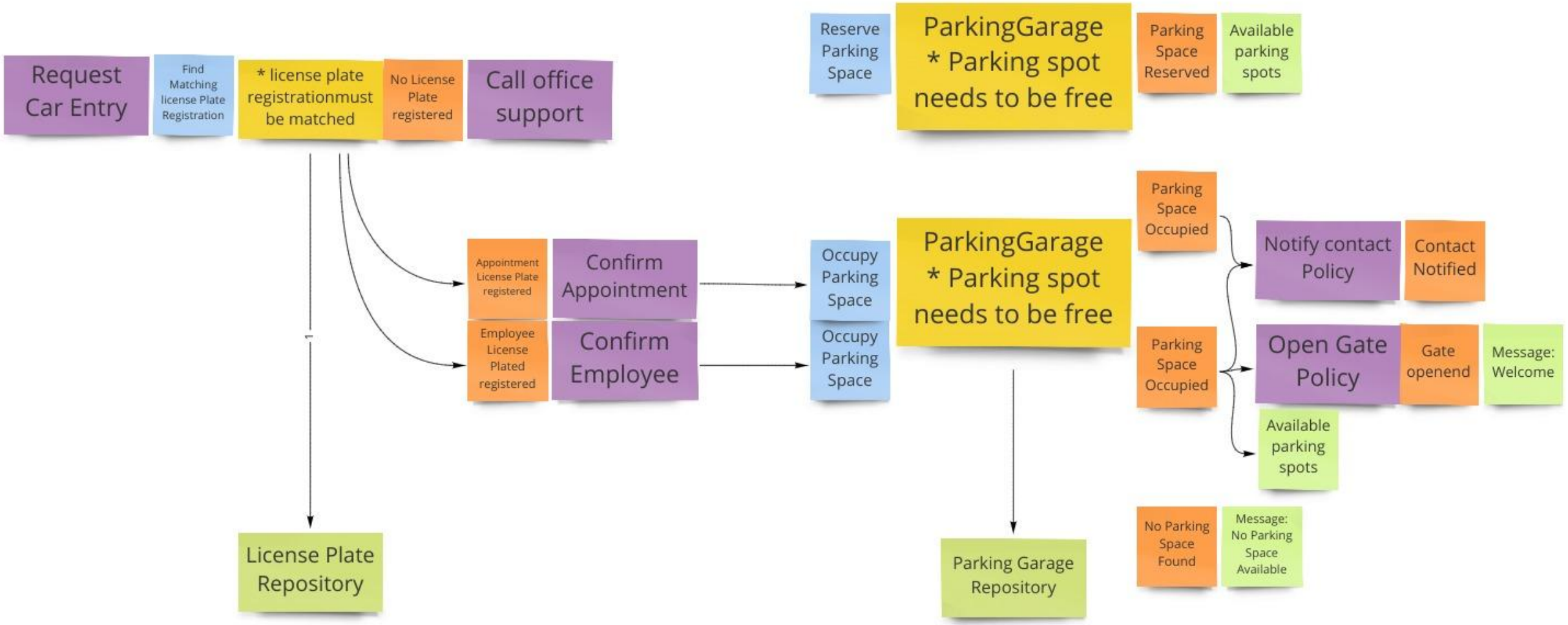



Xebia
Xpirit
Flying Spaghetti Functions
https://clearlens.org/do-you-believe-in-a-flying-spaghetti-monster/


Your code
Someone else's server
Serverless


Xebia
Xpirit
Domain-Driven
DESIGN

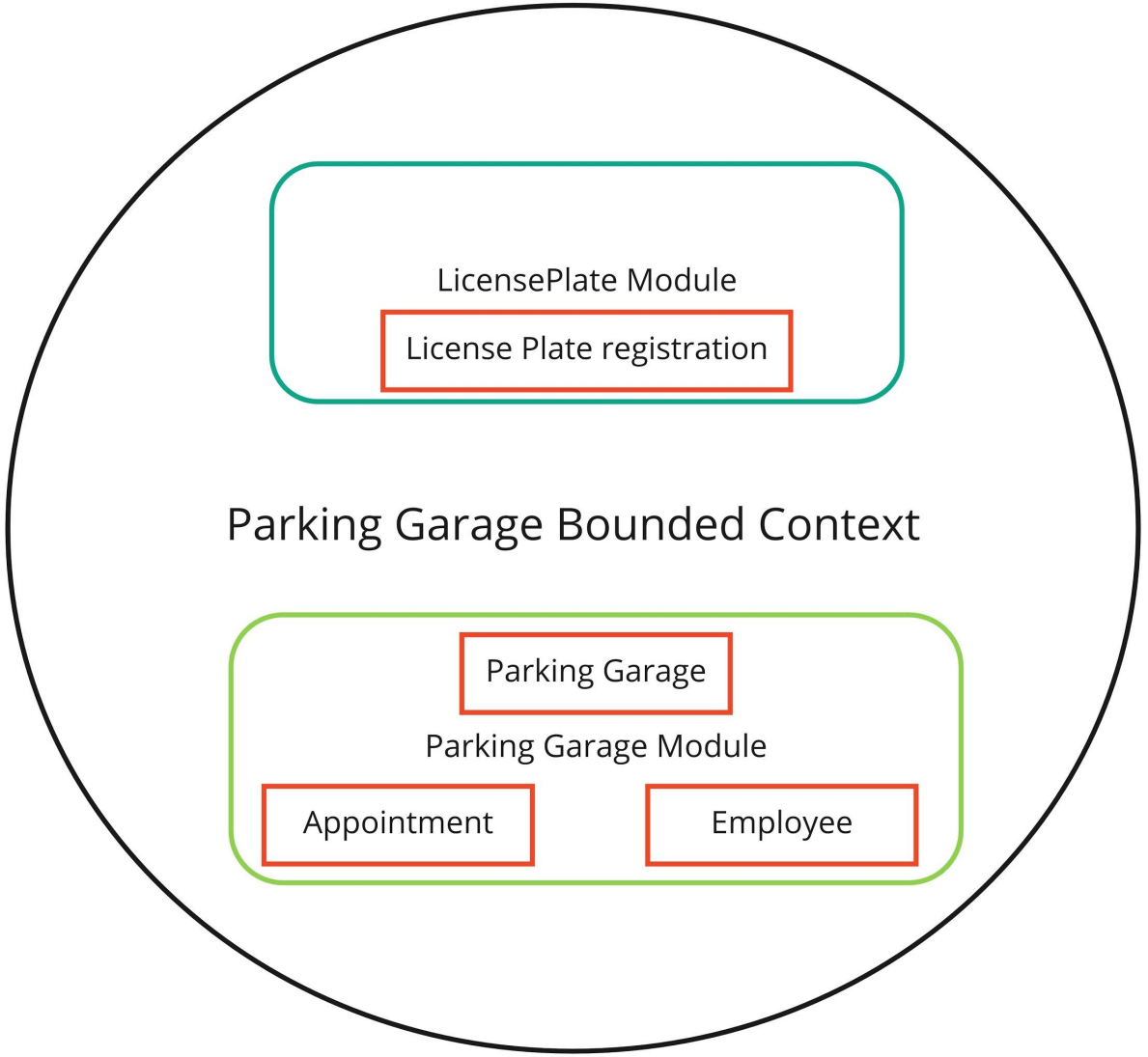

Xebia
Xpirit
EVENT
STORMING


Request Car Entry
Find Matching license Plate Registration
* license plate registrationmust be matched
No License Plate registered
Call office support
Reserve Parking Space
ParkingGarage * Parking spot needs to be free
Parking Space Reserved
Available parking spots
Appointment License Plate registered
Confirm Appointment
Employee License Plated registered
Confirm Employee
Occupy Parking Space
Occupy Parking Space
ParkingGarage * Parking spot needs to be free
Parking Space Occupied
Parking Space Occupied
Notify contact Policy
Contact Notified
Open Gate Policy
Gate openend
Message: Welcome
Available parking spots
1
License Plate Repository
Parking Garage Repository
No Parking Space Found
Message: No Parking Space Available

Question: How do we split systems into functions?

Heuristic: Within a bounded context modules are a perfect candidate for splitting up into functions.

Heuristic: Use one source control repository per bounded context.



Xebia

Xpirit

Portal

Documentation

SDK

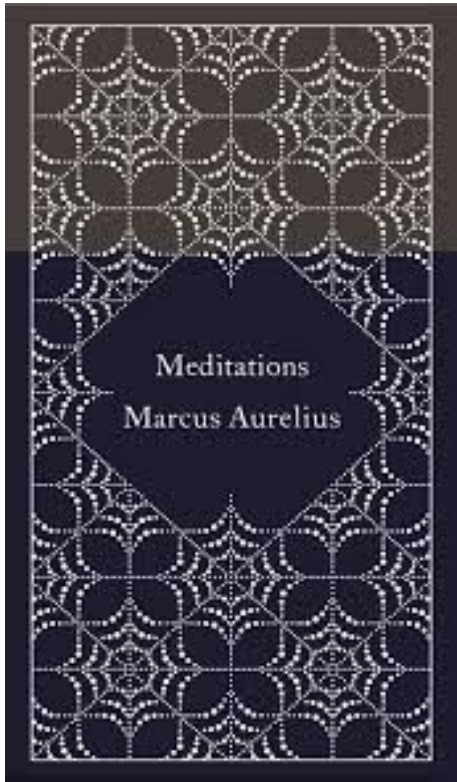
Testing

Orchestration

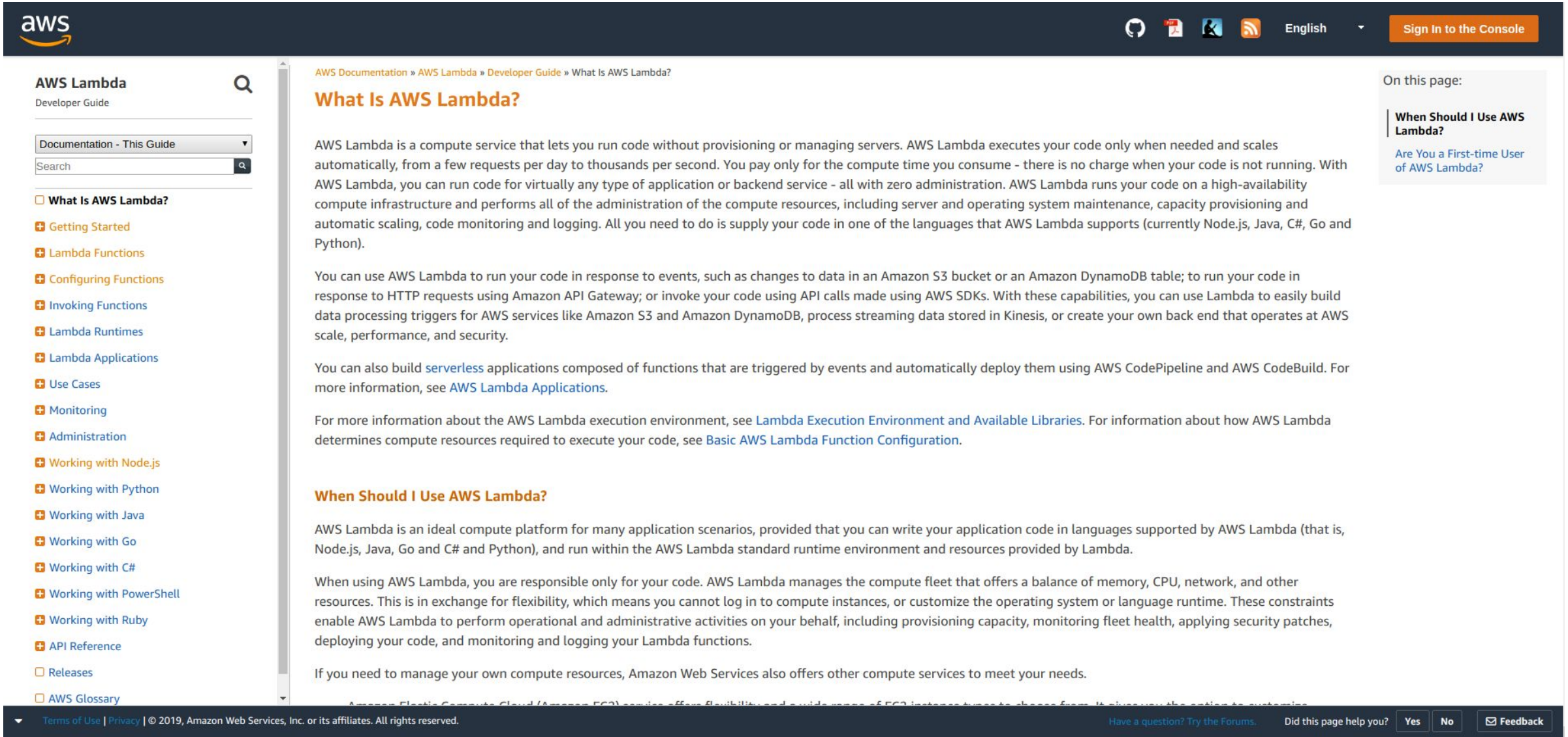
Monitoring

“Everything we hear is an ***opinion***, ***not a fact***.
Everything we see is a ***perspective***, ***not the truth.***”

– Marcus Aurelius , Meditations



Documentation - AWS



aws

English

Sign In to the Console

AWS Lambda

Developer Guide

Documentation - This Guide

Search

- What Is AWS Lambda?
- Getting Started
- Lambda Functions
- Configuring Functions
- Invoking Functions
- Lambda Runtimes
- Lambda Applications
- Use Cases
- Monitoring
- Administration
- Working with Node.js
- Working with Python
- Working with Java
- Working with Go
- Working with C#
- Working with PowerShell
- Working with Ruby
- API Reference
- Releases
- AWS Glossary

AWS Documentation » AWS Lambda » Developer Guide » What Is AWS Lambda?

What Is AWS Lambda?

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes 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. With AWS Lambda, you can run code for virtually any type of application or backend service - all with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. All you need to do is supply your code in one of the languages that AWS Lambda supports (currently Node.js, Java, C#, Go and Python).

You can use AWS Lambda to run your code in response to events, such as changes to data in an Amazon S3 bucket or an Amazon DynamoDB table; to run your code in response to HTTP requests using Amazon API Gateway; or invoke your code using API calls made using AWS SDKs. With these capabilities, you can use Lambda to easily build data processing triggers for AWS services like Amazon S3 and Amazon DynamoDB, process streaming data stored in Kinesis, or create your own back end that operates at AWS scale, performance, and security.

You can also build serverless applications composed of functions that are triggered by events and automatically deploy them using AWS CodePipeline and AWS CodeBuild. For more information, see AWS Lambda Applications.

For more information about the AWS Lambda execution environment, see Lambda Execution Environment and Available Libraries. For information about how AWS Lambda determines compute resources required to execute your code, see Basic AWS Lambda Function Configuration.

When Should I Use AWS Lambda?

AWS Lambda is an ideal compute platform for many application scenarios, provided that you can write your application code in languages supported by AWS Lambda (that is, Node.js, Java, Go and C# and Python), and run within the AWS Lambda standard runtime environment and resources provided by Lambda.

When using AWS Lambda, you are responsible only for your code. AWS Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources. This is in exchange for flexibility, which means you cannot log in to compute instances, or customize the operating system or language runtime. These constraints enable AWS Lambda to perform operational and administrative activities on your behalf, including provisioning capacity, monitoring fleet health, applying security patches, deploying your code, and monitoring and logging your Lambda functions.

If you need to manage your own compute resources, Amazon Web Services also offers other compute services to meet your needs.

On this page:

- When Should I Use AWS Lambda?
- Are You a First-time User of AWS Lambda?

Terms of Use | Privacy | © 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Have a question? Try the Forums. Did this page help you? Yes No Feedback

Documentation - AWS

Documentation - Azure

Create your first function in Az

https://docs.microsoft.com/en-us/azure/azure-functions/funct

Most Visited Getting Started

Microsoft Azure

Contact Sales: 1-800-867-1389 Search Portal

Overview Solutions Products Documentation Pricing Training Marketplace Partners Support Blog More Free account

Azure / Functions

Feedback Edit Share Dark Sign in

Filter by title

- Functions Documentation
- Overview
- Quickstarts
  - Create function - Visual Studio
  - Create function - Visual Studio Code
  - Create function - Java/Maven
  - Create function - Python
  - Create function - Azure CLI
  - Create function - portal
  - Create function - Linux
- Triggers
- Integrate
- Tutorials
- Samples
- Concepts
- How-to guides

Create your first function using Visual Studio

10/17/2018 • 5 minutes to read • Contributors all

Azure Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application.

In this article, you learn how to use the Visual Studio 2017 tools for Azure Functions to locally create and test a "hello world" function. You then publish the function code to Azure. These tools are available as part of the Azure development workload in Visual Studio 2017.

https://mydemofunctionapp2018

https://mydemofunctionapp20181016.azurewebsites.net/api/Function1?name=Bill

```
Hello, Bill
```

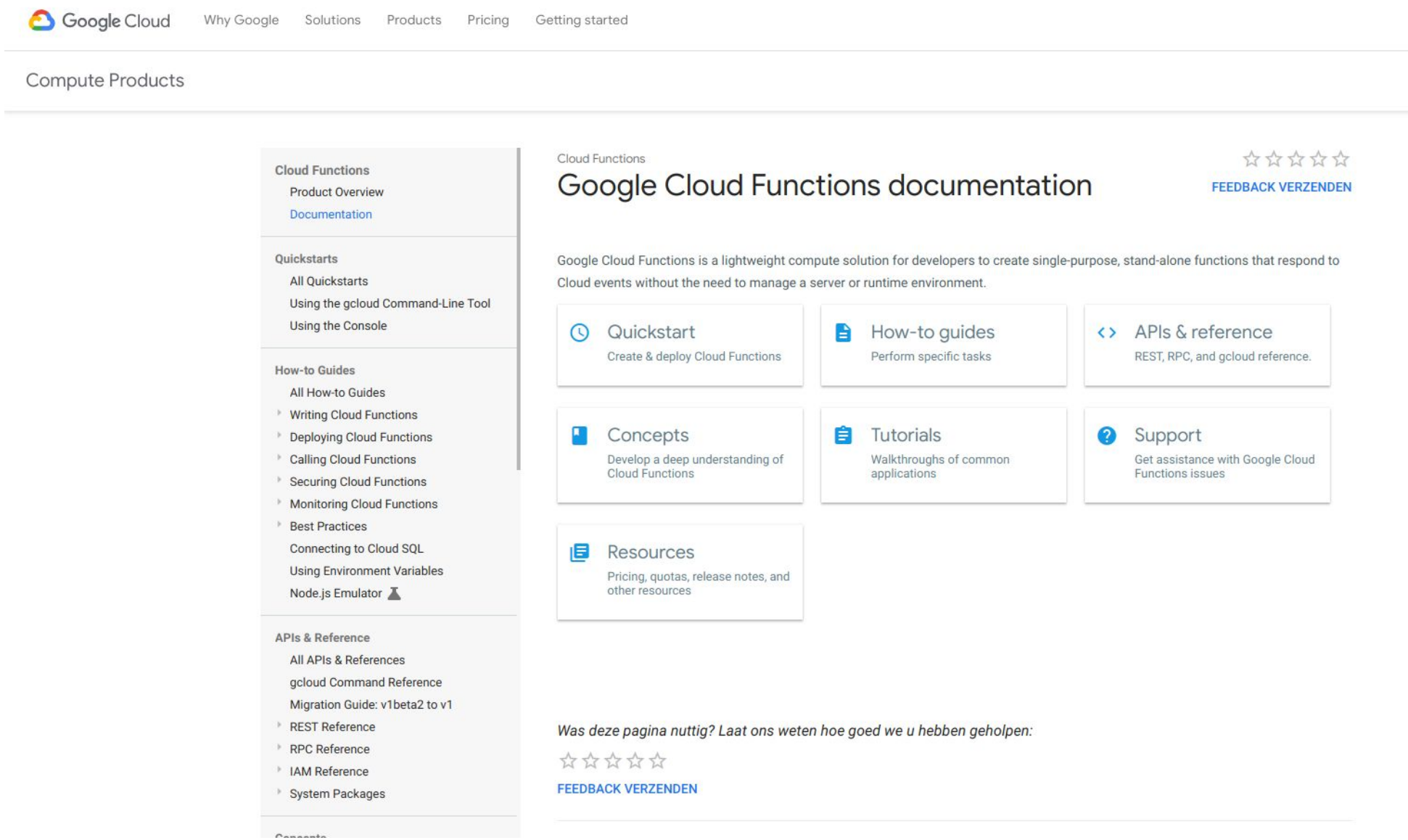
This topic includes a video that demonstrates the same basic steps.

Prerequisites

In this article

- Prerequisites
- Create a function app project
- Test the function locally
- Publish the project to Azure
- Test your function in Azure
- Watch the video
- Next steps

Documentation - GCloud



Documentation

	+	-
AWS Lambda	• Extensive, detailed documentation. • Everything you need!	• Extensive wall of text crits you
Azure Functions	• Quickstarts for novices • Detailed reference material for the more experienced.	• Not 100% complete across all languages.
GCP Functions	• Quickstarts for novices.	• Hard to find specific information

SDK - AWS

```
export function handle(event: APIGatewayEvent, context: Context, callback: Callback) {

    let licensePlate: LicensePlate;
    try {
        licensePlate = LicensePlate.fromJSON(JSON.parse(event.body as string));
    } catch (error) {
        console.log("error: "+error);
        return callback( error: null, Response.BAD_REQUEST(error));
    }

    return callback( error: null, Response.OK(licensePlate.number));
}
```

https://www.npmjs.com/package/@types/aws-lambda

Xebia

Xpirit

SDK - AWS

```
export class Response {

    public static OK = (body: string) => new Response( statusCode: 200, body);
    public static BAD_REQUEST = (body: string) => new Response( statusCode: 400, body);
    public static CONFLICT = (body: string) => new Response( statusCode: 409, body);
    public static INTERNAL_SERVER_ERROR = (body: string) => new Response( statusCode: 500, body);

    private headers: { [name: string]: string } = {};

    constructor(public readonly statusCode: number, public readonly body: string) {

    }
}
```

Xebia

Xpirit

SDK - AWS

```
public static async saveParkingGarage(parkingGarage: ParkingGarage): Promise<ParkingGarage> {
    const tableName = await resolveTableName();
    const params: PutItemInput = {
        TableName: tableName,
        Item: parkingGarage.toJSON() as PutItemInputAttributeMap,
    };
    return documentClient.put(params).promise().then(
        onfulfilled: data => Promise.resolve(parkingGarage),
        onrejected: error => Promise.reject(error)
    );

}
```

SDK - AWS

```
public static async findParkingGarage(id: string): Promise<ParkingGarage> {
    const tableName = await resolveTableName();

    const params = {
        TableName: tableName,
        KeyConditionExpression: '#id = :id',
        ExpressionAttributeNames: {
            '#id': 'id',
        },
        ExpressionAttributeValues: {
            ':id': id,
        }
    };

    let data = await documentClient.query(params).promise();
    if (data.Items) {
        return Promise.resolve(ParkingGarage.fromJSON(data.Items.pop() as any));
    }
    return Promise.reject( reason: 'no parking garage found');

}
```

SDK - Azure

```
0 references
public static class Function1
{
    [FunctionName("Function1")]
    0 references
    public static async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,
        ILogger log)
    {
        log.LogInformation("C# HTTP trigger function processed a request.");

        string name = req.Query["name"];

        string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
        dynamic data = JsonConvert.DeserializeObject(requestBody);
        name = name ?? data?.name;

        return name != null
            ? (ActionResult)new OkObjectResult($"Hello, {name}")
            : new BadRequestObjectResult("Please pass a name on the query string or in the request body");
    }
}
```

SDK - Azure

```
1 reference | 0 changes | 0 authors, 0 changes
public async Task<ParkingGarage> GetByIdAsync(string id)
{
    ParkingGarage result;
    var documentUri = UriFactory.CreateDocumentUri(DatabaseId, CollectionId, id);
    var requestOptions = new RequestOptions { PartitionKey = new PartitionKey(PartitionKeyValue) };
    try
    {
        var document = await _documentClient.ReadDocumentAsync<ParkingGarage>(documentUri, requestOptions);
        result = document.Document;
    }
    catch (Exception e)
    {
        throw RepositoryExceptionBuilder.CreateExceptionForDocumentRead(documentUri, e);
    }

    return result;
}
```

SDK - GCP

```
import {LicensePlate} from "./entities/license-plate";
import {Request, Response} from "express";

export function handle(request: Request, response: Response) {
    let licensePlate: LicensePlate;
    try {
        licensePlate = LicensePlate.fromJSON(request.body);
    } catch (error) {
        console.log("error: " + error);
        return response.status( code: 400).send(error);
    }

    return response.send(licensePlate.number);
}
```

Xebia

Xpirit

SDK - GCP

```
export class LicensePlateRepository {

    public static async findLicensePlate(license: string): Promise<LicensePlateRegistration> {
        const query = datastore.createQuery( kind: 'License').filter( property: "license", operator: "=", license);

        const [licensePlateRegistrations] = await datastore.runQuery(query);
        if (licensePlateRegistrations.length > 0) {
            return Promise.resolve(LicensePlateRegistration.fromJSON(licensePlateRegistrations.pop() as any));
        }
        return Promise.resolve(new LicensePlateRegistration(license, LicensePlateType.UNKNOWN));

    }

}
```

https://www.npmjs.com/package/@types/google-cloud__datastore

SDK - GCP

Request

The `req` object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on. In this documentation and by convention, the object is always referred to as `req` (and the HTTP response is `res`) but its actual name is determined by the parameters to the callback function in which you're working.

For example:

```
app.get('/user/:id', function(req, res) {
  res.send('user ' + req.params.id);
});
```

But you could just as well have:

```
app.get('/user/:id', function(request, response) {
  response.send('user ' + request.params.id);
});
```

The `req` object is an enhanced version of Node's own request object and supports all built-in fields and methods.

Properties

In Express 4, `req.files` is no longer available on the `req` object by default. To access uploaded files on the `req.files` object, use multipart-handling middleware like busboy, multer, formidable, multiparty, connect-multiparty, or pez.

req.app

This property holds a reference to the instance of the Express application that is using the middleware.

If you follow the pattern in which you create a module that just exports a middleware function and `require()` it in your main file, then the middleware can access the Express instance via `req.app`

For example:

```
//index.js
app.get('/viewdirectory', require('./mymiddleware.js'))
```

SDK

	+	-
AWS Lambda	• Flexible SDK, extensive. • Can also create your own runtime.	• Quirks in the gateway body as string. Need to do validation yourself, also typed.
Azure Functions	• Easy to use SDK for C#.	• Limited usage of certain NuGet package versions the framework is using (e.g. NewtonSoft.Json).
GCP Functions	• Using standard 3th party libraries.	

Orchestration - AWS

```
Type: "AWS::StepFunctions::StateMachine"
Properties:
  StateMachineName: serverless-showdown-state-machine
  DefinitionString: !Sub |
    {
      "Comment": "Car Request",
      "StartAt": "FindMatchingLicensePlate",
      "Version": "1.0",
      "States": {
        "FindMatchingLicensePlate": {
          "Type": "Task",
          "Resource": "${FindMatchingLicensePlate.Arn}",
          "Next": "HandleParking"
        },
        "HandleParking": {
          "Type": "Choice",
          "Choices": [
            {
              "Variable": "$.type",
              "StringEquals": "NoLicensePlateRegistered",
              "Next": "CallOfficeSupport"
```

```
RoleParkingAccessStateMachine:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: !Sub 'states.${AWS::Region}.amazonaws.com'
          Action: 'sts:AssumeRole'
    Policies:
      - PolicyName: lambda
        PolicyDocument:
          Statement:
            - Effect: Allow
              Action: 'lambda:InvokeFunction'
              Resource:
                - !GetAtt 'FindMatchingLicensePlate.Arn'
                - !GetAtt 'ConfirmAppointment.Arn'
                - !GetAtt 'ConfirmEmployee.Arn'
```

https://github.com/binxio/aws-cfn-update

Orchestration - AWS

Execution ARN

arn:aws:states:eu-west-1:761563646002:execution:serverless-showdown-state-machine:7642ab7a-7b92-6464-638c-b1be2e273f5b

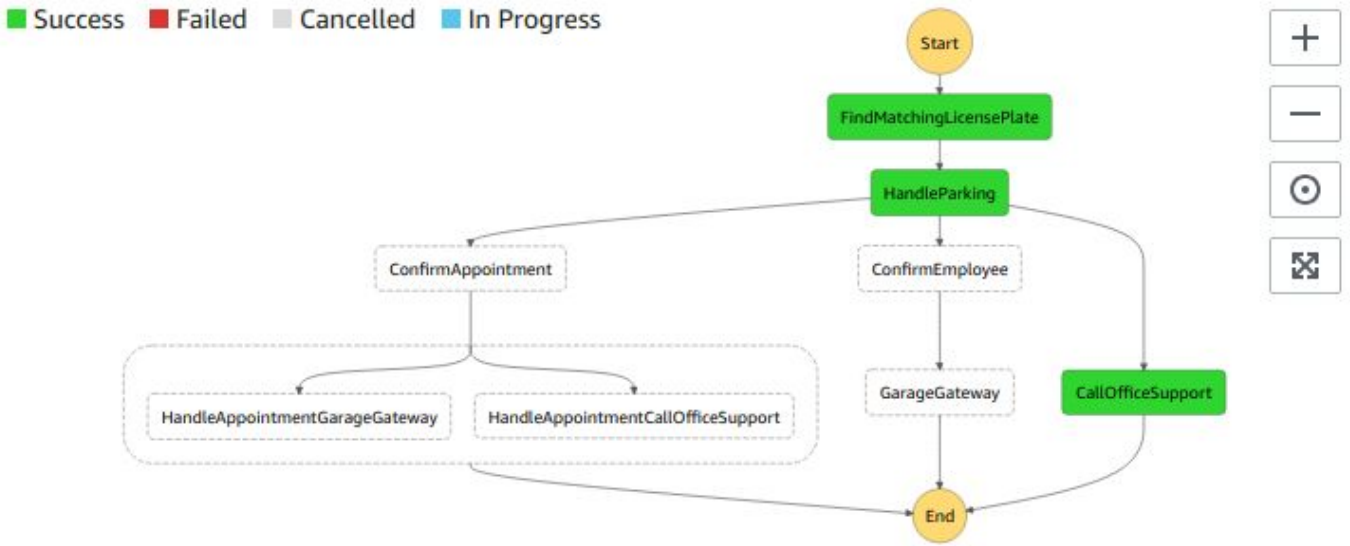
▶ Input

End Time

Feb 5, 2019 06:03:05.219 AM

▶ Output

Visual workflow | Code



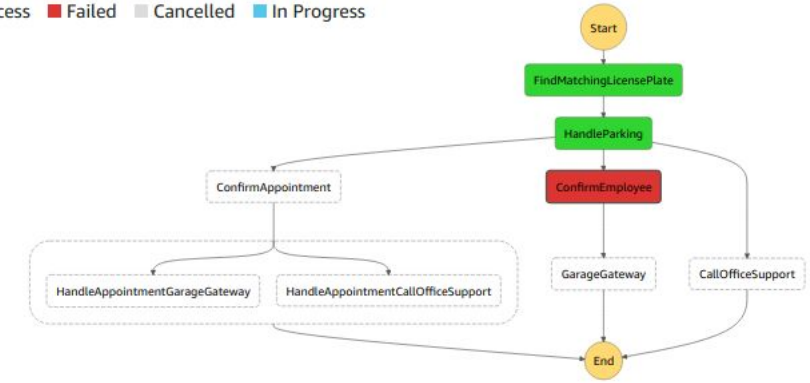
Step details

Select a step to view its details.

Orchestration - AWS

Visual workflow | Code

Success Failed Cancelled In Progress



Step details (ConfirmEmployee)

Status

Failed

Resource

arn:aws:lambda:eu-west-1:761563646002:function:pg-confirmemployee-kb-local | CloudWatch logs

▼ Input

```
{
  "number": "170",
  "type": "EmployeeLicensePlateMatched"
}
```

▼ Output

▼ Exception

Error

```
Lambda.Unknown
```

Cause

```
The cause could not be determined because Lambda did not return an error type.
```

Orchestration - Azure

```
2 references | Marc Duiker, 1 day ago | 1 author, 1 change
public static class ParkingGarageCarEntryOrchestration
{
    [FunctionName(nameof(ParkingGarageCarEntryOrchestration))]
    0 references | Marc Duiker, 1 day ago | 1 author, 1 change
    public static async Task<ParkingOrchestrationResponse> Run(
        [OrchestrationTrigger] DurableOrchestrationContextBase context,
        ILogger logger)
    {
        if (!context.IsReplaying)
        {
            logger.LogInformation($"Started {nameof(ParkingGarageCarEntryOrchestration)} with InstanceId: {context.InstanceId}.");
        }

        var request = context.GetInput<ParkingOrchestrationRequest>();

        var licensePlateResult = await context.CallActivityAsync<LicensePlateRegistration>(
            nameof(GetLicensePlateRegistration),
            request.LicensePlateNumber);


        var confirmParkingRequest = ConfirmParkingRequestBuilder.Build(request.ParkingGarageName, licensePlateResult);
        var confirmParkingResponse = await ConfirmParking(confirmParkingRequest, licensePlateResult, context);

        if (confirmParkingResponse.IsSuccess)
        {
            await context.CallActivityAsync(
                nameof(OpenGate),
                confirmParkingResponse.ParkingGarageName);
        }
        else
        {
```

SDK - Azure

```
3 references | Marc Duiker, 1 day ago | 1 author, 1 change
public static class GetLicensePlateRegistration
{
    private static readonly ILicensePlateRegistrationService Service = new LicensePlateRegistrationService();

    [FunctionName(nameof(GetLicensePlateRegistration))]
    0 references | Marc Duiker, 1 day ago | 1 author, 1 change
    public static async Task<Domain.LicensePlateRegistration> Run(
        [ActivityTrigger] string licensePlateNumber,
        ILogger logger)
    {
        logger.LogInformation($"Started {nameof(GetLicensePlateRegistration)} with {licensePlateNumber}.");

        var licenseplate = await Service.GetAsync(licensePlateNumber);

        return licenseplate;
    }
}
```

Orchestration - GCP

Not supported OOTB :(

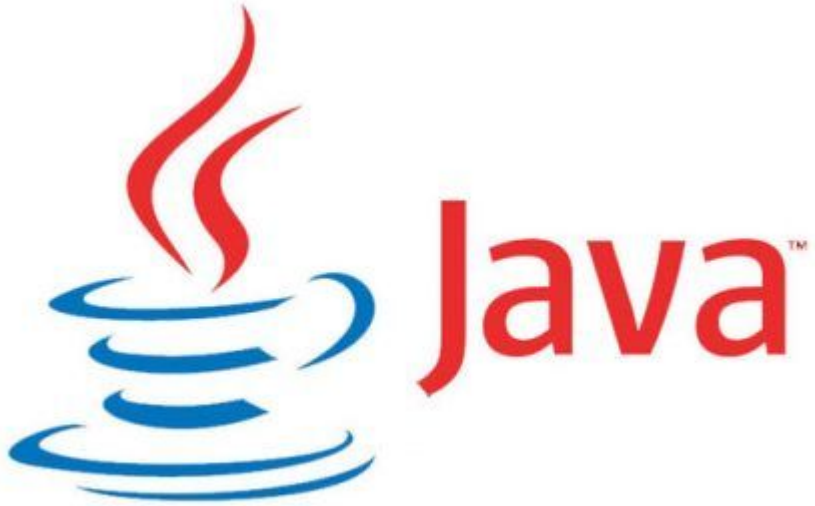
Orchestration

	+	-
AWS Lambda	• Visualisation in the portal	• Uses JSON/YAML configuration.
Azure Functions	• Flexibility because the orchestration is in code.	• The orchestration needs to be deterministic, so don't use code which is not (e.g. new GUIDs, DateTime, new threads).
GCP Functions	• N/A	• N/A

Deployment Cycle - AWS

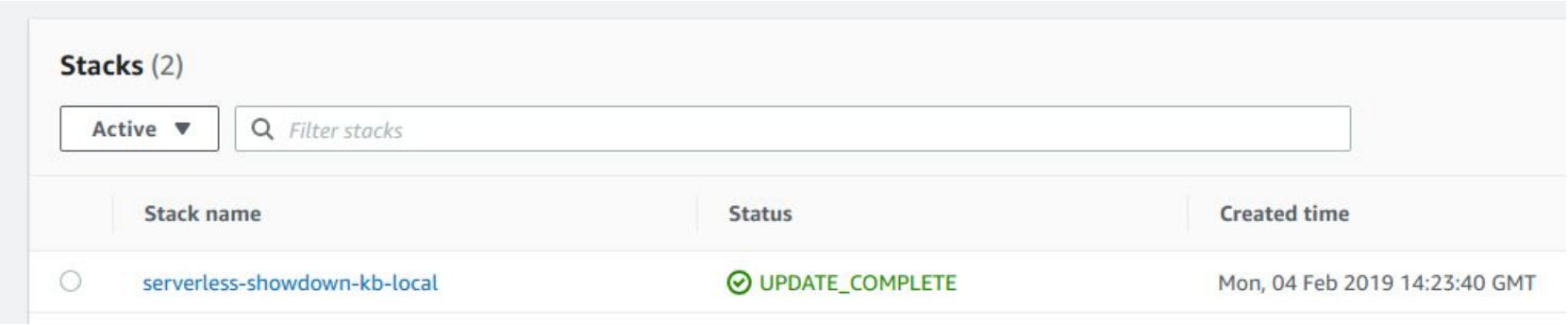
```
FindMatchingLicensePlate:
  Type: 'AWS::Serverless::Function'
  Properties:
    FunctionName: !Sub "pg-findmatchinglicenseplate-${Stage}"
    Handler: find-matching-license-plate.handle
    CodeUri: dist/
    Description: FaaS handler for requesting a car entry
    MemorySize: 128
    ReservedConcurrentExecutions: 20
    Environment:
      Variables:
        LICENSE_PLATE_TABLE_NAME: !Ref 'LicensePlateRepository'
    Policies:
      - Version: "2012-10-17"
        Statement:
          - Effect: "Allow"
            Action:
              - "dynamoDb:Query"
            Resource:
              !Sub 'arn:aws:dynamodb:${AWS::Region}:${AWS::AccountId}:table/LicensePlateRepository*'
```

Deployment Cycle - AWS



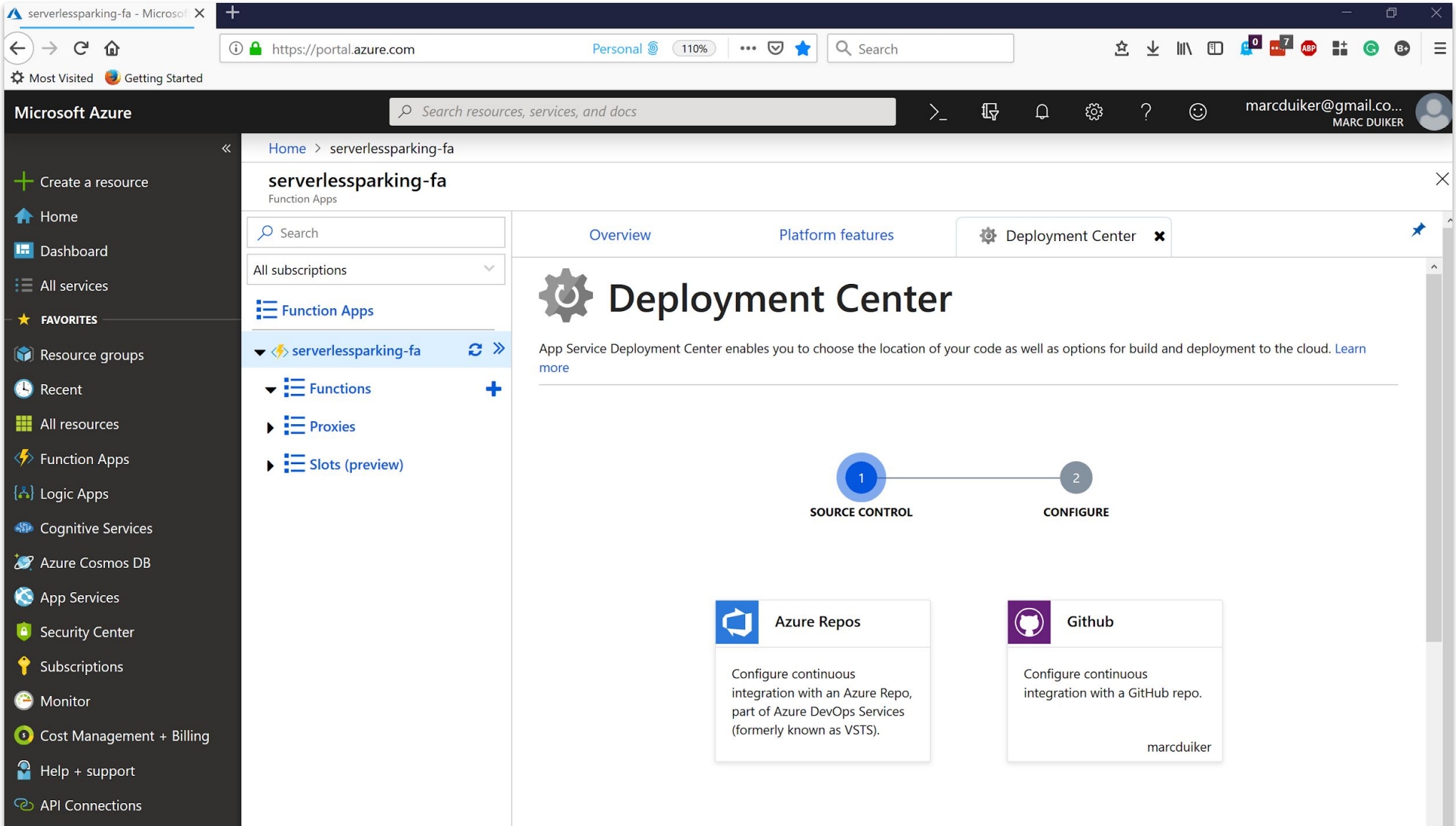
https://xebia.com/blog/building-an-elixir-runtime-for-aws-lambda/

Deployment Cycle - AWS

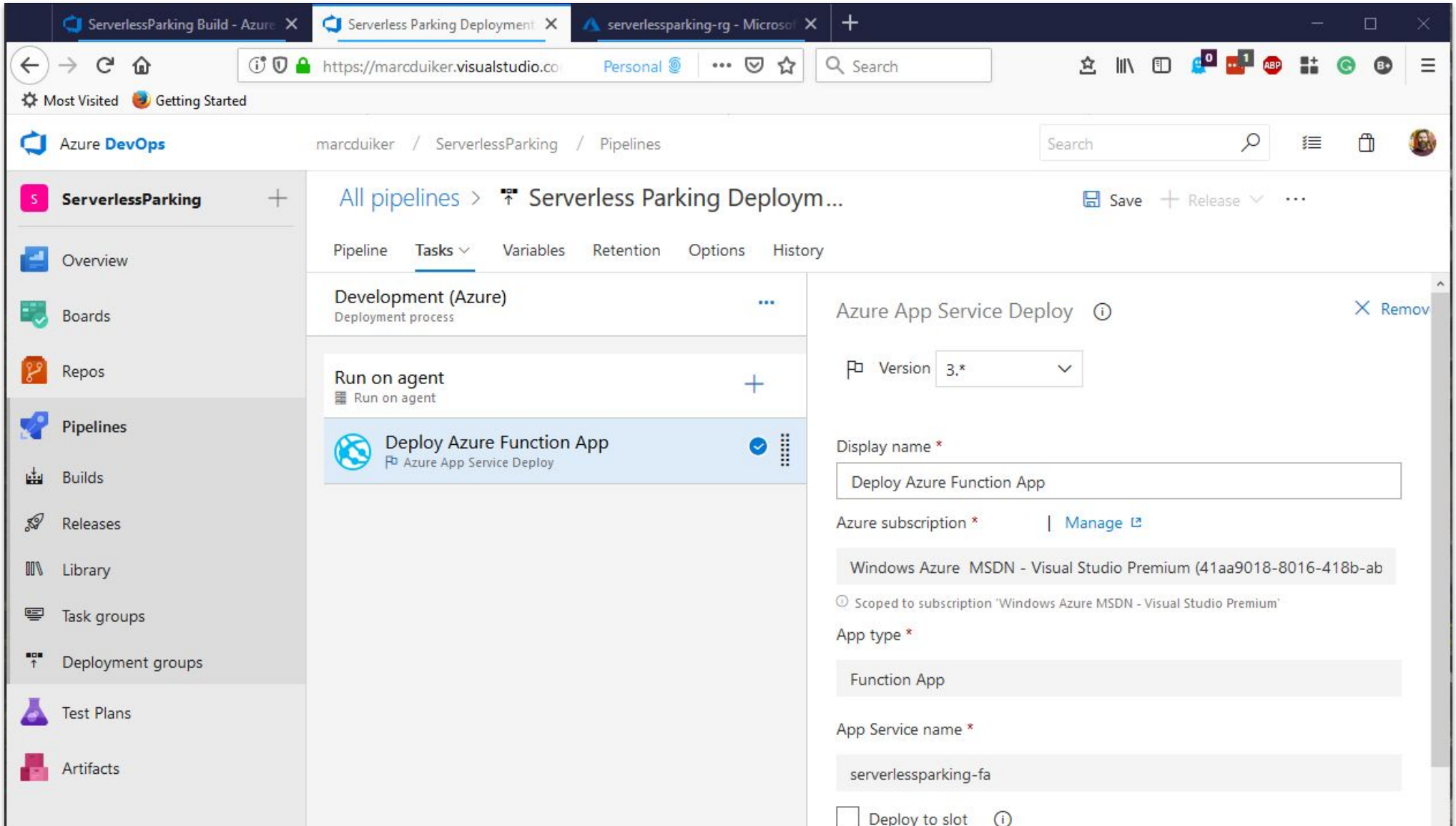


16:46:51 UTC-0500	UPDATE_ROLLBACK_COMPLETE	AWS::CloudFormation::Stack	serverless-showdown-kb-local	
16:46:49 UTC-0500	UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS	AWS::CloudFormation::Stack	serverless-showdown-kb-local	
16:46:48 UTC-0500	UPDATE_COMPLETE	AWS::StepFunctions::StateMachine	ParkingAccessStateMachine	
16:46:46 UTC-0500	UPDATE_COMPLETE	AWS::Lambda::Function	GarageGateway	
16:46:46 UTC-0500	UPDATE_COMPLETE	AWS::Lambda::Function	ConfirmEmployee	
16:46:46 UTC-0500	UPDATE_COMPLETE	AWS::Lambda::Function	FindMatchingLicensePlate	
16:46:46 UTC-0500	UPDATE_COMPLETE	AWS::Lambda::Function	ConfirmAppointment	
16:46:46 UTC-0500	UPDATE_COMPLETE	AWS::Lambda::Function	ContactNotification	
16:46:46 UTC-0500	UPDATE_COMPLETE	AWS::Lambda::Function	RequestCarEntry	
16:46:46 UTC-0500	UPDATE_IN_PROGRESS	AWS::Lambda::Function	GarageGateway	
16:46:46 UTC-0500	UPDATE_IN_PROGRESS	AWS::Lambda::Function	ConfirmAppointment	
16:46:46 UTC-0500	UPDATE_IN_PROGRESS	AWS::Lambda::Function	ConfirmEmployee	
16:46:46 UTC-0500	UPDATE_IN_PROGRESS	AWS::Lambda::Function	FindMatchingLicensePlate	
16:46:46 UTC-0500	UPDATE_IN_PROGRESS	AWS::Lambda::Function	ContactNotification	
16:46:46 UTC-0500	UPDATE_IN_PROGRESS	AWS::Lambda::Function	RequestCarEntry	
16:46:13 UTC-0500	UPDATE_ROLLBACK_IN_PROGRESS	AWS::CloudFormation::Stack	serverless-showdown-kb-local	The following resource(s) failed to update: [ParkingAccessStateMachine].
16:46:12 UTC-0500	UPDATE_FAILED	AWS::StepFunctions::StateMachine	ParkingAccessStateMachine	Invalid State Machine Definition: 'MISSING_TRANSITION_TARGET: Missing 'Next' target: Choice at /States/FindMatchingLicensePlate/Next, MISSING_TRANSITION_TARGET: State "HandleParking" is not reachable. at /States/HandleParking' (Service: AWSStepFunctions; Status Code: 400; Error Code: InvalidDefinition; Request ID: 49e6359e-28c6-11e9-83bf-e983c9efb683)
16:46:10 UTC-0500	UPDATE_IN_PROGRESS	AWS::StepFunctions::StateMachine	ParkingAccessStateMachine	

Deployment Cycle - Azure



Deployment Cycle - Azure



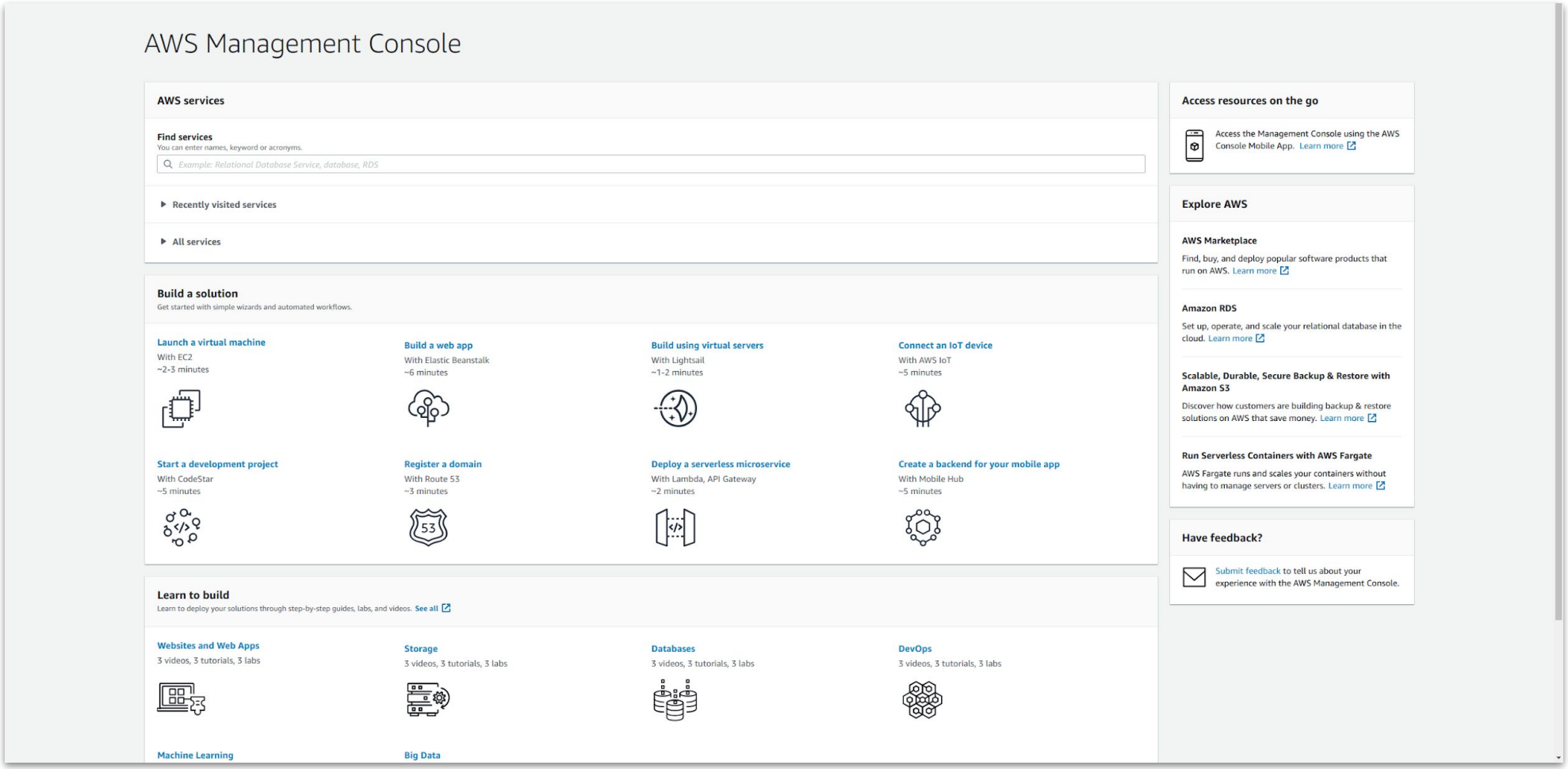
Deployment Cycle - GCP

```
-/+ google_storage_bucket_object.archive (new resource required)
      id:             "showdown-serverless-deploy-http_trigger.zip" => <computed> (forces new resource)
      bucket:         "showdown-serverless-deploy" => "showdown-serverless-deploy"
      content_type:   "application/zip" => <computed>
      crc32c:         "sDzRbA==" => <computed>
      detect_md5hash: "hVC+76p+XCpeDt6ZfpFEiA==" => "different hash" (forces new resource)
      md5hash:        "hVC+76p+XCpeDt6ZfpFEiA==" => <computed>
      name:           "http_trigger.zip" => "http_trigger.zip"
      source:         "./../dist/index.zip" => "./../dist/index.zip"
      storage_class:  "STANDARD" => <computed>


Plan: 1 to add, 0 to change, 1 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

google_storage_bucket_object.archive: Destroying... (ID: showdown-serverless-deploy-http_trigger.zip)
google_storage_bucket_object.archive: Destruction complete after 1s
google_storage_bucket_object.archive: Creating...
  bucket:         "" => "showdown-serverless-deploy"
  content_type:   "" => "<computed>"
  crc32c:         "" => "<computed>"
  detect_md5hash: "" => "different hash"
  md5hash:        "" => "<computed>"
  name:           "" => "http_trigger.zip"
  source:         "" => "./../dist/index.zip"
  storage_class:  "" => "<computed>"
google_storage_bucket_object.archive: Creation complete after 0s (ID: showdown-serverless-deploy-http_tr
igger.zip)

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

The state of your infrastructure has been saved to the path
below. This state is required to modify and destroy your
infrastructure, so keep it safe. To inspect the complete state
use the `terraform show` command.

State path: ./state/terraform.tfstate
baasie@baasie-Precision-5520:~/gitrepos/prive/serverless-showdown/gcloud/infra$ terraform destroy
```

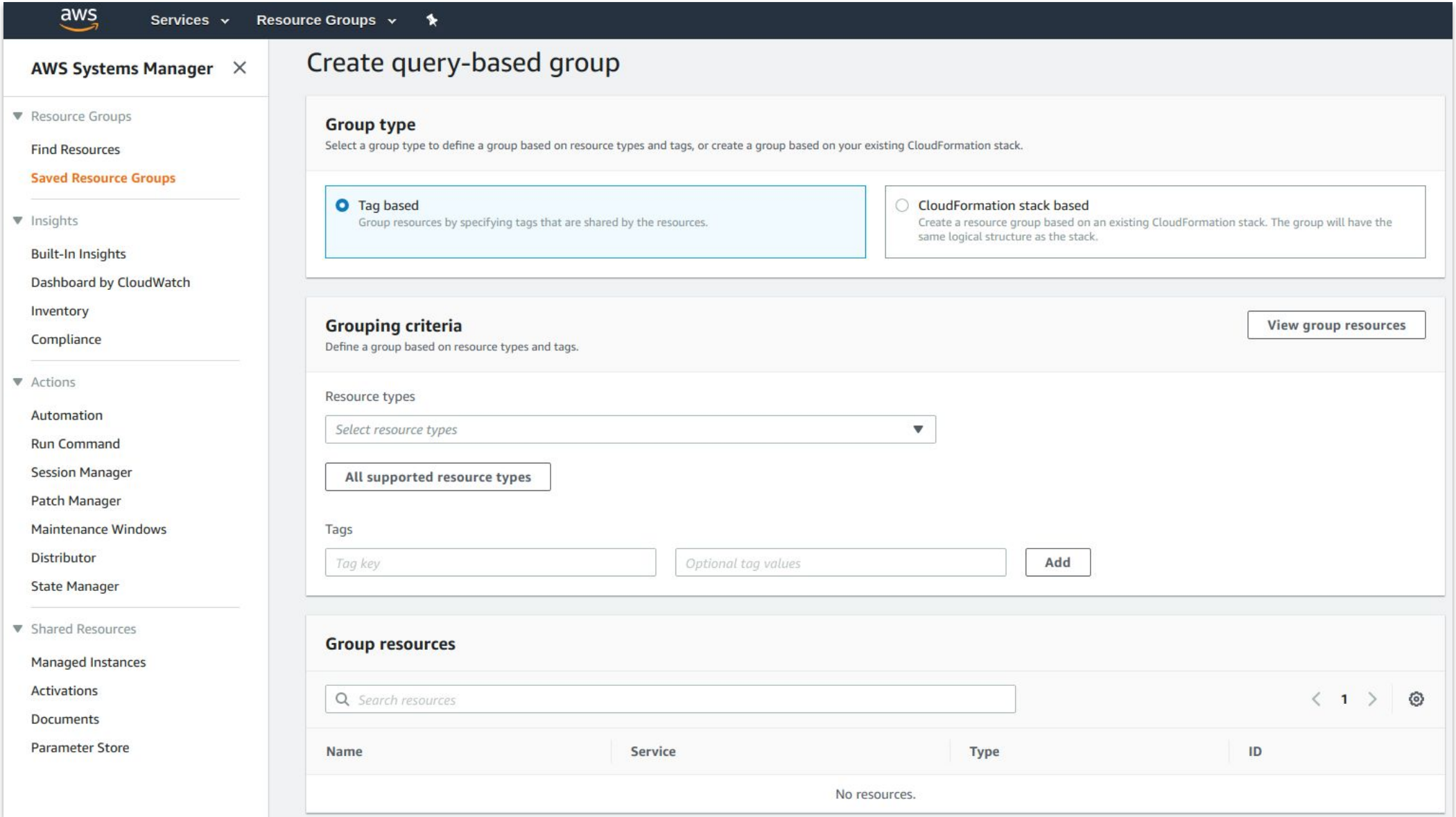
Deployment Cycle

	+	-
AWS Lambda	• Stateful Cloudformation.	• Sometimes manual interference when config is false.
Azure Functions	• Multiple ways to deploy your functions to the cloud. • Prefer CLI over ARM.	• Watch out for breaking changes in orchestrations.
GCP Functions	• Terraform to the rescue!	• Google Cloud deployment configuration counter intuitive.

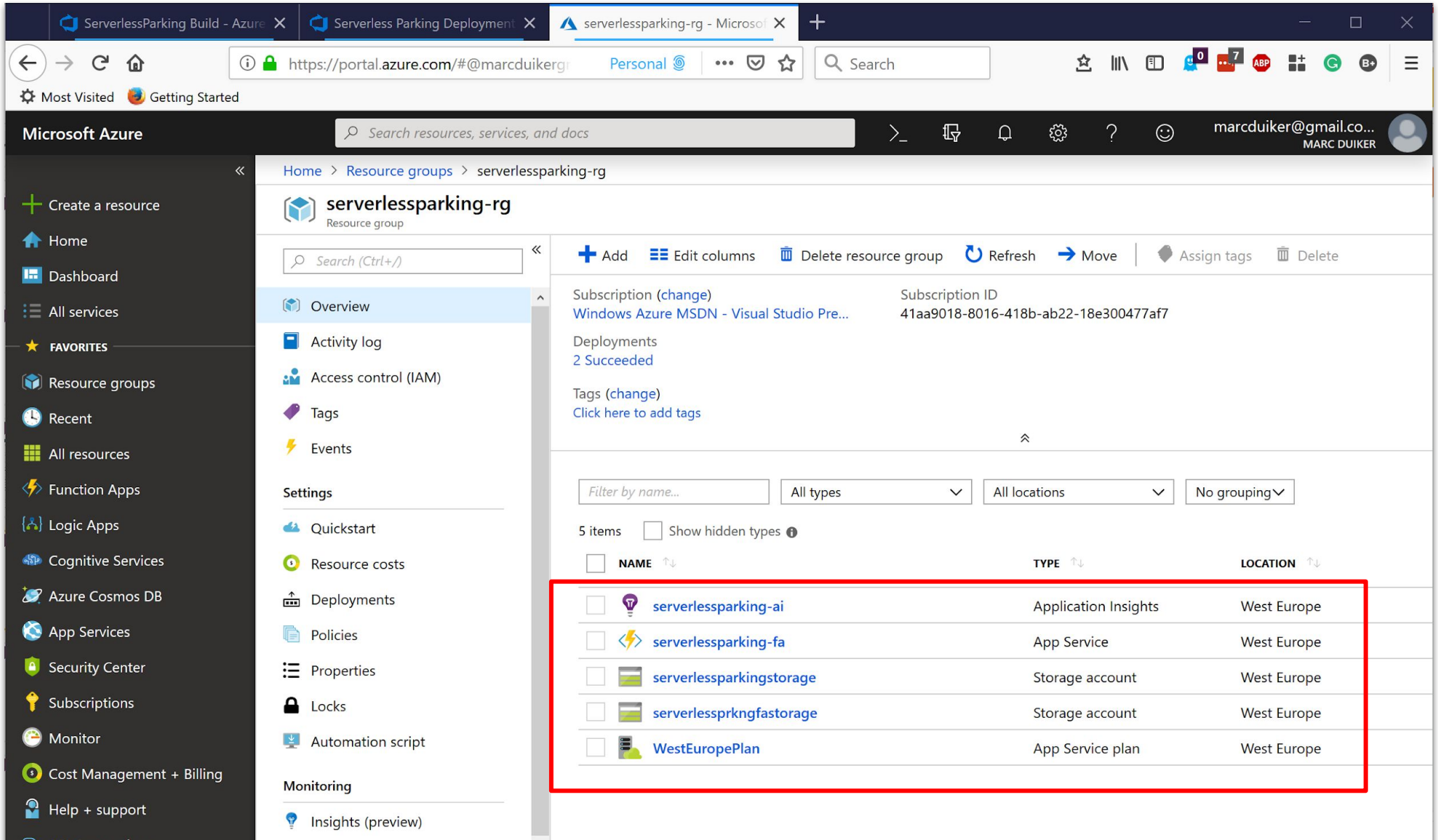
Portal - AWS


AWS Management Console
AWS services
Find services
You can enter names, keyword or acronyms.
Example: Relational Database Service, database, RDS
Recently visited services
All services
Build a solution
Get started with simple wizards and automated workflows.
Launch a virtual machine
With EC2
~2-3 minutes
Build a web app
With Elastic Beanstalk
~6 minutes
Build using virtual servers
With Lightsail
~1-2 minutes
Connect an IoT device
With AWS IoT
~5 minutes
Start a development project
With CodeStar
~5 minutes
Register a domain
With Route 53
~3 minutes
Deploy a serverless microservice
With Lambda, API Gateway
~2 minutes
Create a backend for your mobile app
With Mobile Hub
~5 minutes
Learn to build
Learn to deploy your solutions through step-by-step guides, labs, and videos. See all
Websites and Web Apps
3 videos, 3 tutorials, 3 labs
Storage
3 videos, 3 tutorials, 3 labs
Databases
3 videos, 3 tutorials, 3 labs
DevOps
3 videos, 3 tutorials, 3 labs
Machine Learning
Big Data
Access resources on the go
Access the Management Console using the AWS Console Mobile App. Learn more
Explore AWS
AWS Marketplace
Find, buy, and deploy popular software products that run on AWS. Learn more
Amazon RDS
Set up, operate, and scale your relational database in the cloud. Learn more
Scalable, Durable, Secure Backup & Restore with Amazon S3
Discover how customers are building backup & restore solutions on AWS that save money. Learn more
Run Serverless Containers with AWS Fargate
AWS Fargate runs and scales your containers without having to manage servers or clusters. Learn more
Have feedback?
Submit feedback to tell us about your experience with the AWS Management Console.

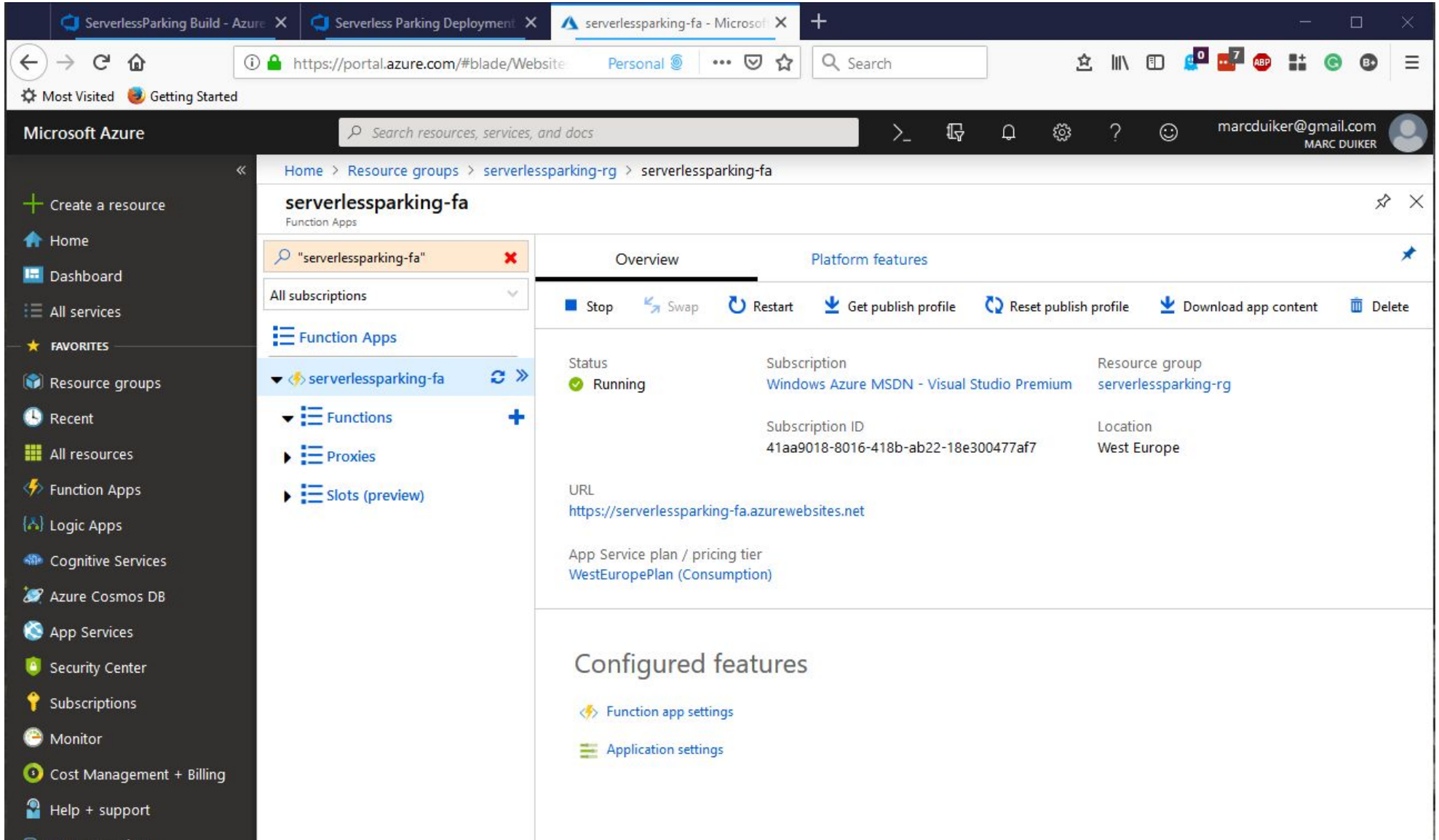
Portal - AWS


aws
Services
Resource Groups
AWS Systems Manager
Resource Groups
Find Resources
Saved Resource Groups
Insights
Built-In Insights
Dashboard by CloudWatch
Inventory
Compliance
Actions
Automation
Run Command
Session Manager
Patch Manager
Maintenance Windows
Distributor
State Manager
Shared Resources
Managed Instances
Activations
Documents
Parameter Store
Create query-based group
Group type
Select a group type to define a group based on resource types and tags, or create a group based on your existing CloudFormation stack.
Tag based
Group resources by specifying tags that are shared by the resources.
CloudFormation stack based
Create a resource group based on an existing CloudFormation stack. The group will have the same logical structure as the stack.
Grouping criteria
Define a group based on resource types and tags.
View group resources
Resource types
Select resource types
All supported resource types
Tags
Tag key
Optional tag values
Add
Group resources
Search resources
1
Name
Service
Type
ID
No resources.

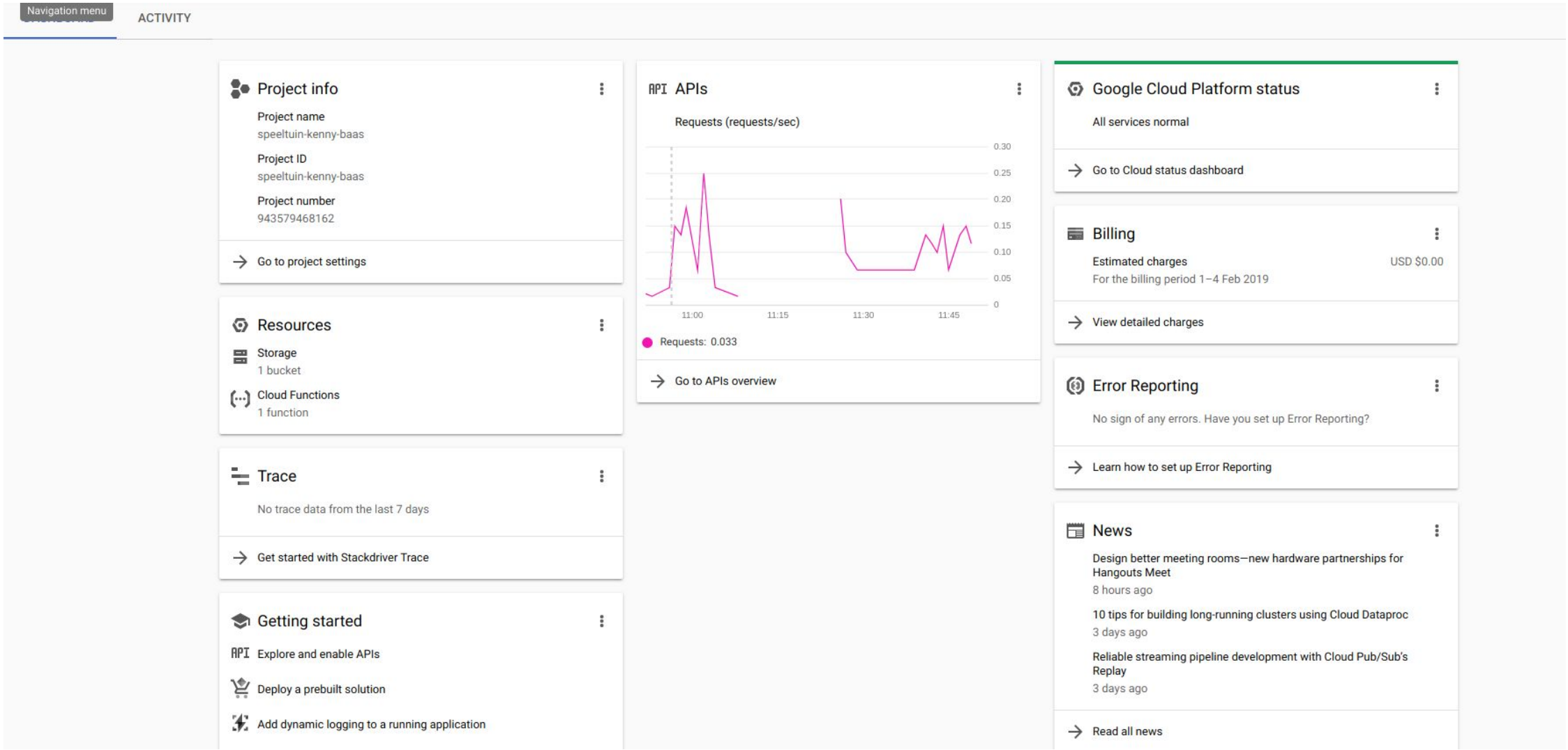
Portal - Azure



Portal - Azure



Portal - GCloud



Portal

	+	-
AWS Lambda	• Most parts are simple and easy to discover.	• No real boundaries of projects. • Some parts are hard to navigate.
Azure Functions	• Resource Groups are useful containers for multiple services.	• It is very slow to navigate & use.
GCP Functions	• Projects is good concept for grouping services. • Portal is quick to use.	

Testing - AWS


```
import {doesJournalEventExist, LocalDate, putBMICalculatedEvent} from '../../src/port.adapter.dynamodb/document-client';
import {BASIC_BMICALCULATEDEVENT} from "../domain/bmi-meter-events.fixtures";
import {PersonId} from "../../src/domain/plastic-categories";

const mockResolveTableName = jest.fn( implementation: () => Promise.resolve( value: 'table'));
jest.mock( moduleName: '../../src/support/properties',  factory: () => ({
    resolvePlasticDayEventsTableName: () => mockResolveTableName(),
}));

let mockCount = 1;
const mockQueryResult = jest.fn( implementation: () => Promise.resolve( value: {
    Count: mockCount
}));
const mockPutResult = jest.fn( implementation: () => Promise.resolve( value: {}));
jest.mock( moduleName: 'aws-sdk',  factory: () => ({
    DynamoDB: {
        DocumentClient: class {
            query() {
                return {
                    promise: mockQueryResult
                }
            };

            put() {
                return {
                    promise: mockPutResult
                }
            };
        }
    }
}));
```

Testing - AWS

- [x] Python Versions support
  - [x] Python 2.7
  - [x] Python 3.6
  - [x] Python 3.7
- [ ] Supported AWS Lambda Runtimes
  - [x] `nodejs`
  - [x] `nodejs4.3`
  - [x] `nodejs6.10`
  - [x] `nodejs8.10`
  - [x] `java8`
  - [x] `python2.7`
  - [x] `python3.6`
  - [x] `python3.7`
  - [x] `go1.x`
  - [ ] `dotnetcore1.0`
  - [x] `dotnetcore2.0`
  - [x] `dotnetcore2.1`
  - [x] `ruby2.5`
  - [x] `Provided`
- [x] AWS credential support
- [x] Debugging support
- [x] Inline Swagger support within SAM templates
- [x] Validating SAM templates locally
- [x] Generating boilerplate templates
  - [x] `nodejs`
  - [x] `nodejs4.3`
  - [x] `nodejs6.10`
  - [x] `nodejs8.10`
  - [x] `java8`
  - [x] `python2.7`
  - [x] `python3.6`
  - [x] `python3.7`
  - [x] `go1.x`
  - [x] `dotnetcore1.0`
  - [x] `dotnetcore2.0`
  - [x] `ruby2.5`
  - [ ] `Provided`

AWS SAM LOCAL

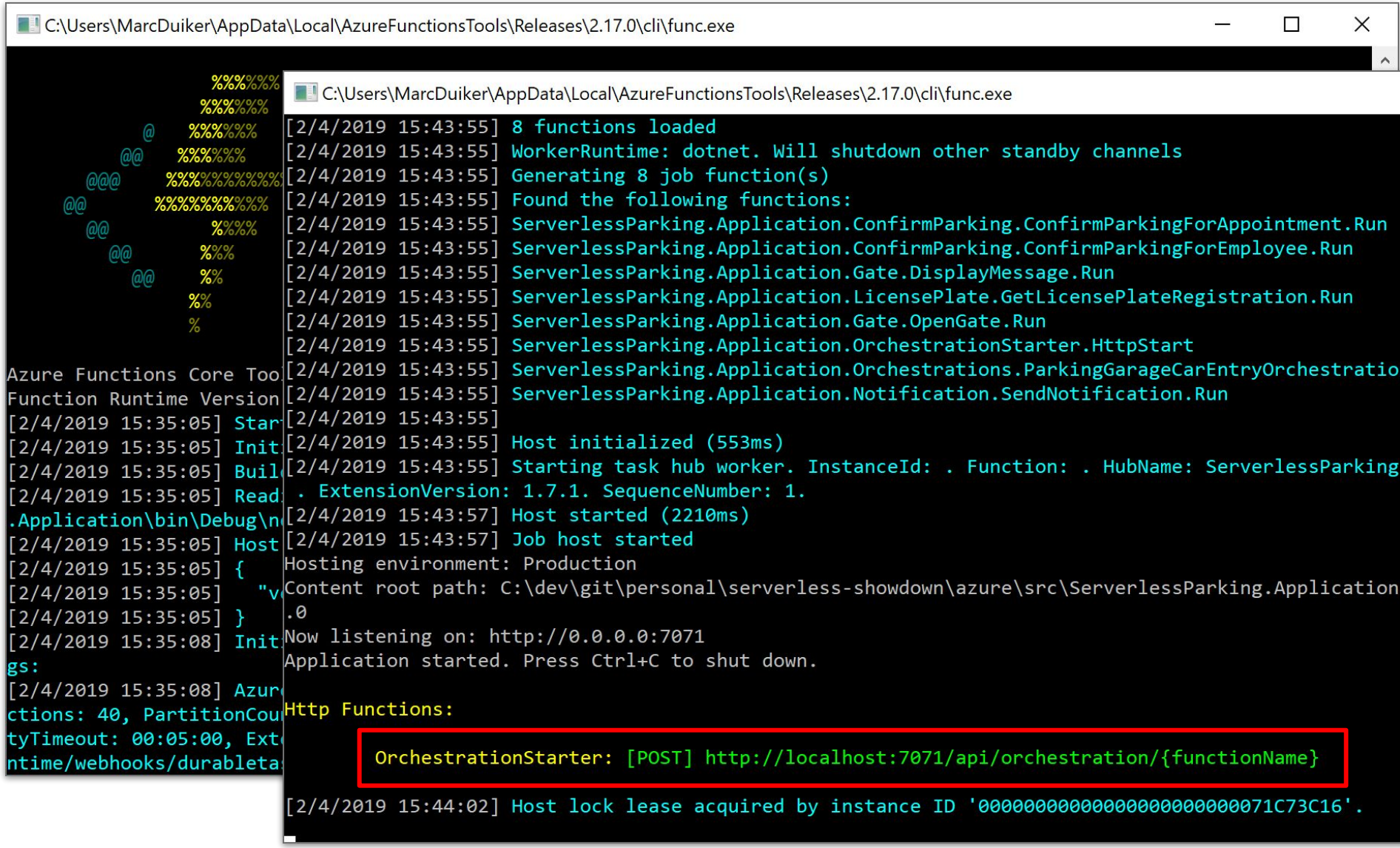
Testing - Azure

```
1 reference | Marc Duiker, 4 days ago | 1 author, 1 change
private DurableOrchestrationContextBase CreateFakeContextForUnkownLicensePlate()
{
    var context = A.Fake<DurableOrchestrationContextBase>();
    // Configure input
    A.CallTo(() => context.GetInput<ParkingOrchestrationRequest>())
        .Returns(new ParkingOrchestrationRequest
        {
            ParkingGarageName = "Parking Garage 1",
            LicensePlateNumber = "ABC-123"
        });

    // Configure GetLicensePlateRegistration activity
    A.CallTo(() => context.CallActivityAsync<LicensePlateRegistration>(
            nameof(GetLicensePlateRegistration),
            A<string>._))
        .Returns(Task.FromResult(new LicensePlateRegistration {Type = LicensePlateType.Unknown}));

    // Configure DisplayMessage activity
    A.CallTo(() => context.CallActivityAsync(
            nameof(DisplayMessage),
            A<DisplayMessageRequest>._))
        .Returns(Task.CompletedTask);

    return context;
}
```

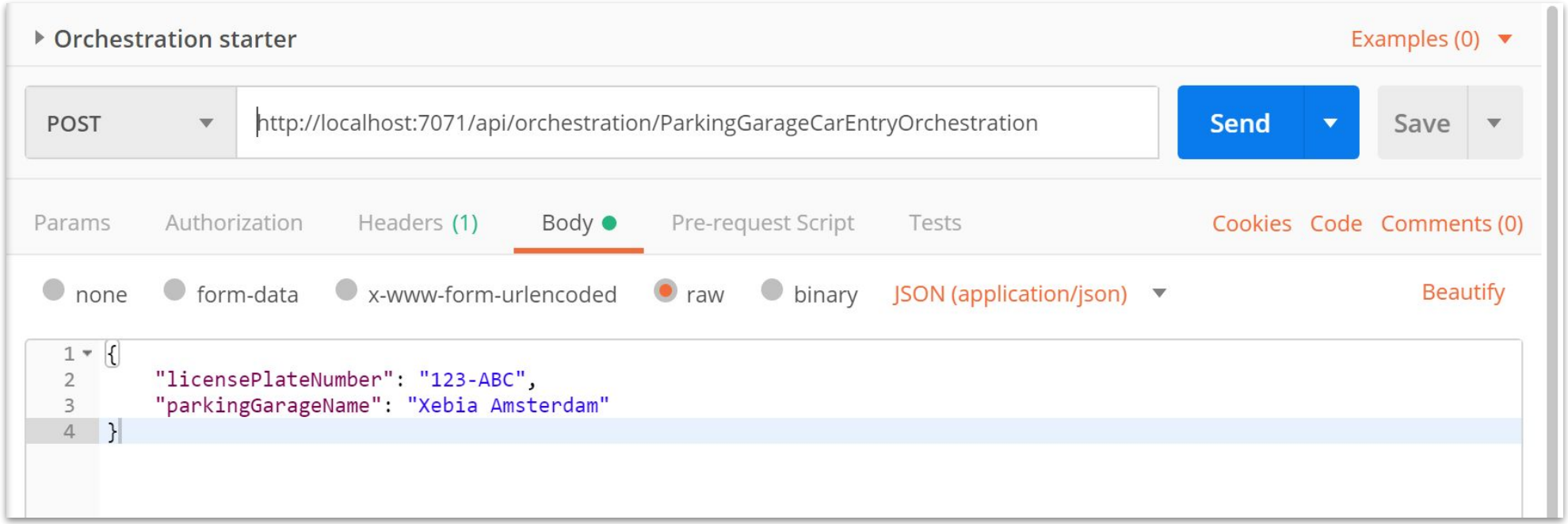
Testing - Azure

```
[Fact]
0 references | Marc Duiker, 4 days ago | 1 author, 1 change
public async Task GivenLicensePlateIsUnknown_WhenOrchestrationIsStarted_ThenGateOpenedShouldBeFalse()
{
    // Arrange
    var context = CreateFakeContextForUnkownLicensePlate();
    var logger = A.Fake<ILogger>();

    // Act
    var result  = await ParkingGarageCarEntryOrchestration.Run(context, logger);

    // Assert
    result.GateOpened.Should().BeFalse();
}
```

Testing - Azure



Testing - Azure



Testing - Azure

```
2 references | Marc Duiker, 3 days ago | 1 author, 2 changes
18  public static class ParkingGarageCarEntryOrchestration
19  {
20      [FunctionName(nameof(ParkingGarageCarEntryOrchestration))]
        0 references | Marc Duiker, 3 days ago | 1 author, 2 changes
21      public static async Task<ParkingOrchestrationResponse> Run(
22          [OrchestrationTrigger] DurableOrchestrationContextBase context,  context = {DurableOrchestrationContext}
23          ILogger logger)  logger = {Logger}
24      {
25          if (!context.IsReplaying)
26          {
27              logger.LogInformation($"Started {nameof(ParkingGarageCarEntryOrchestration)} with InstanceId: {context.InstanceId}.");
28          }
29
30          var request = context.GetInput<ParkingOrchestrationRequest>();  request = {ParkingOrchestrationRequest}
31
32          var licenseP                                    ctivityAsync<LicensePlateRegistration>(  context = {DurableOrchestrationContext}
33              nameof(G
34              request.LicensePlateNumber);
35
36          var confirmParkingRequest = ConfirmParkingRequestBuilder.Build(request.ParkingGarageName, licensePlateResult);
37          var confirmParkingResponse = await ConfirmParking(confirmParkingRequest, licensePlateResult, context);
38
39          if (confirmParkingResponse.IsSuccess)
40          {

request {ServerlessParking.Application.Orchestrations.Models.ParkingOrchestrationRequest}
LicensePlateNumber  "123-ABC"
ParkingGarageName  "Xebia Amsterdam"
```

Testing - GCP

```
baasie@baasie-Precision-5520:~/gitrepos/prive/serverless-showdown/gcloud$ functions start
Starting Google Cloud Functions Emulator...
Google Cloud Functions Emulator STARTED
```

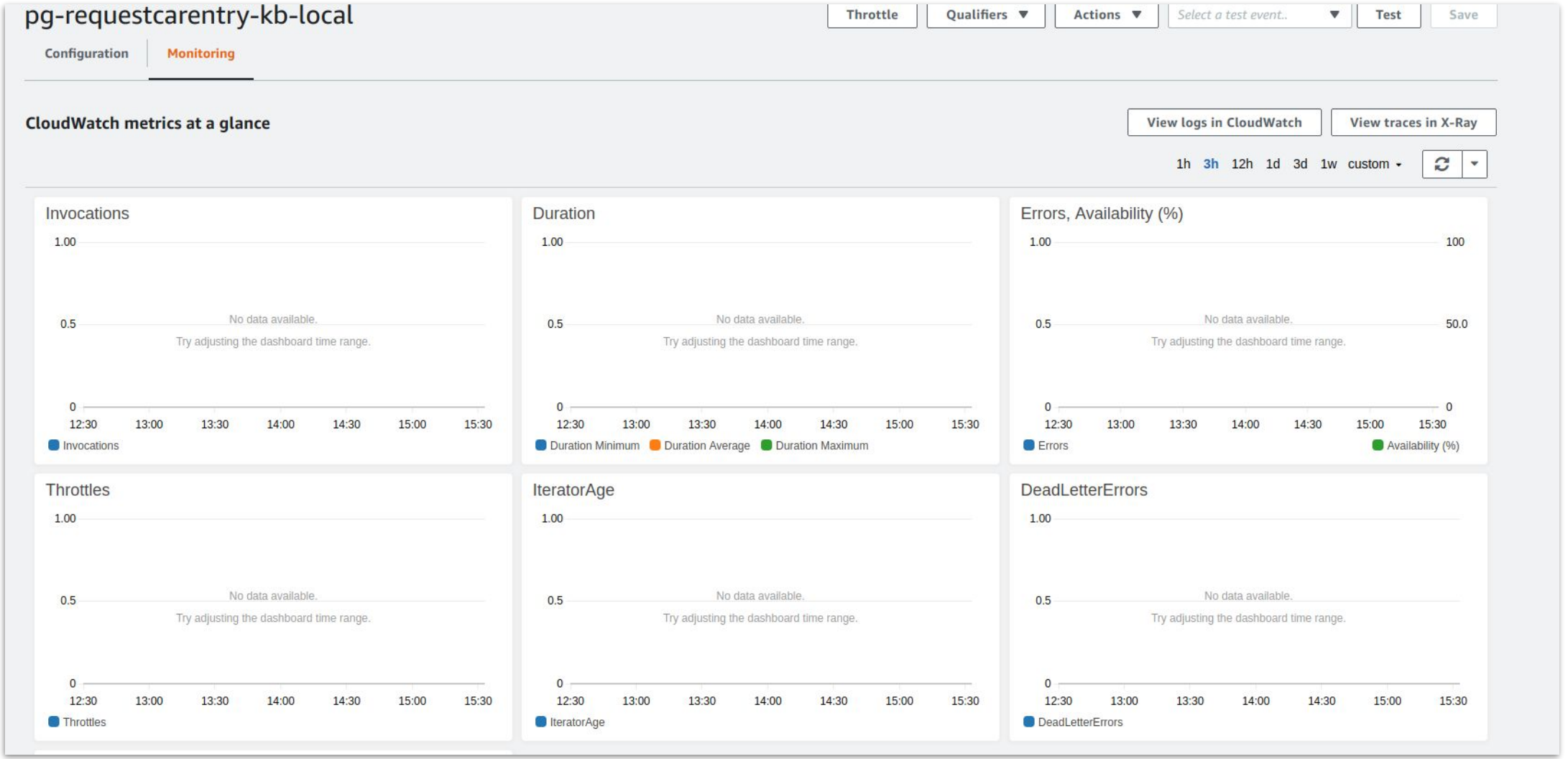
Status	Name	Trigger	Resource
READY	handle	HTTP	http://localhost:8010/speeltuin-kenny-baas/us-central1/handle

```
baasie@baasie-Precision-5520:~/gitrepos/prive/serverless-showdown/gcloud$ functions call handle --data='{"number":"test"}'
ExecutionId: 63abd6a3-4e7c-4eb5-97c2-84eebf8e648e
Result: test
baasie@baasie-Precision-5520:~/gitrepos/prive/serverless-showdown/gcloud$
```

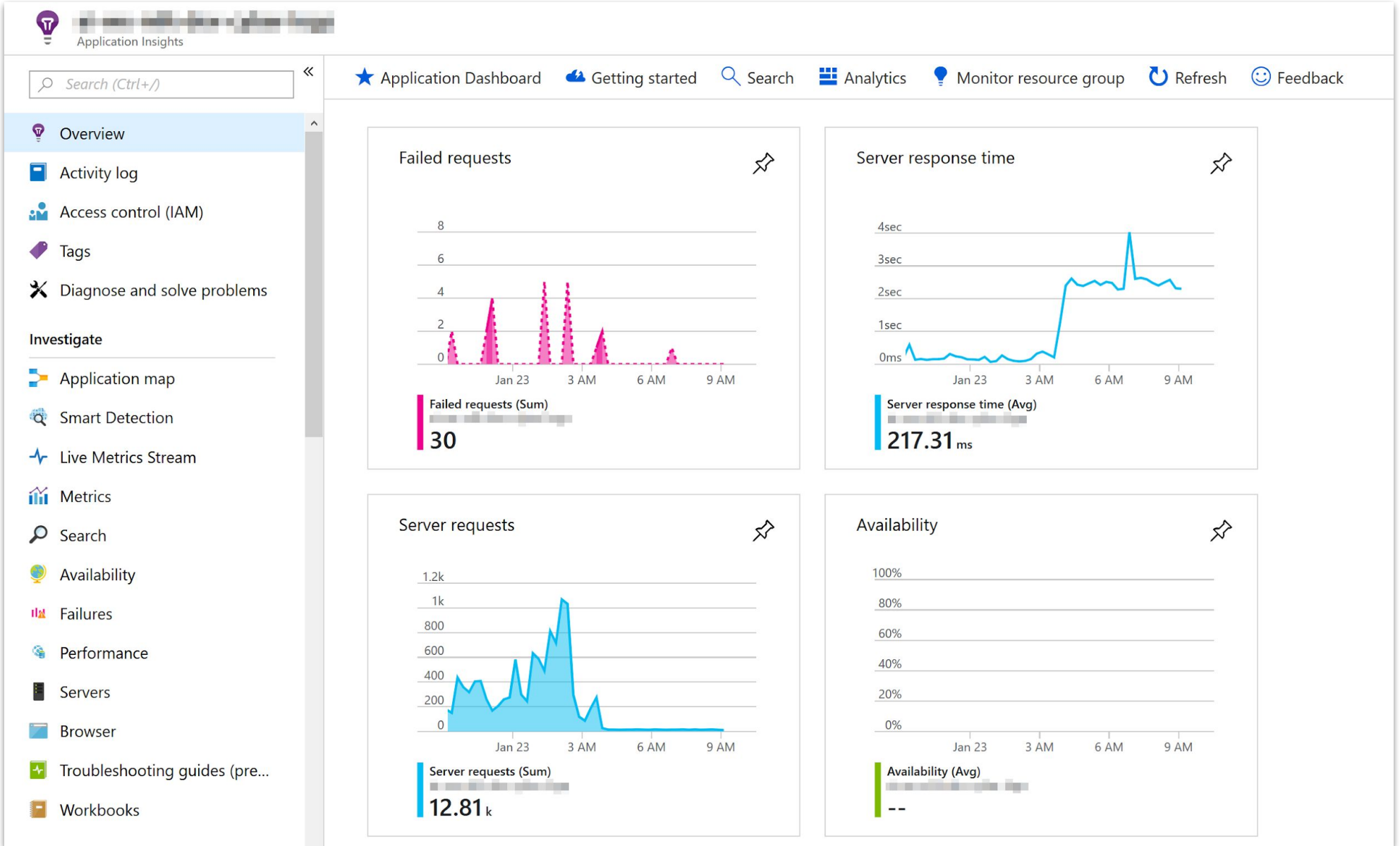
Testing

	+	-
AWS Lambda	• Unit testing is easy • SAM Local seems promising	• SAM Local still in beta • Mocking dynamo takes some time to understand
Azure Functions	• Unit testing is easy • Local runtime is available for running without cloud connection.	• Local runtime depends on a separate storage emulator (not all versions are compatible).
GCP Functions	• Node Emulator available for local testing.	• Emulator is in alpha version • Supports Node v6 only

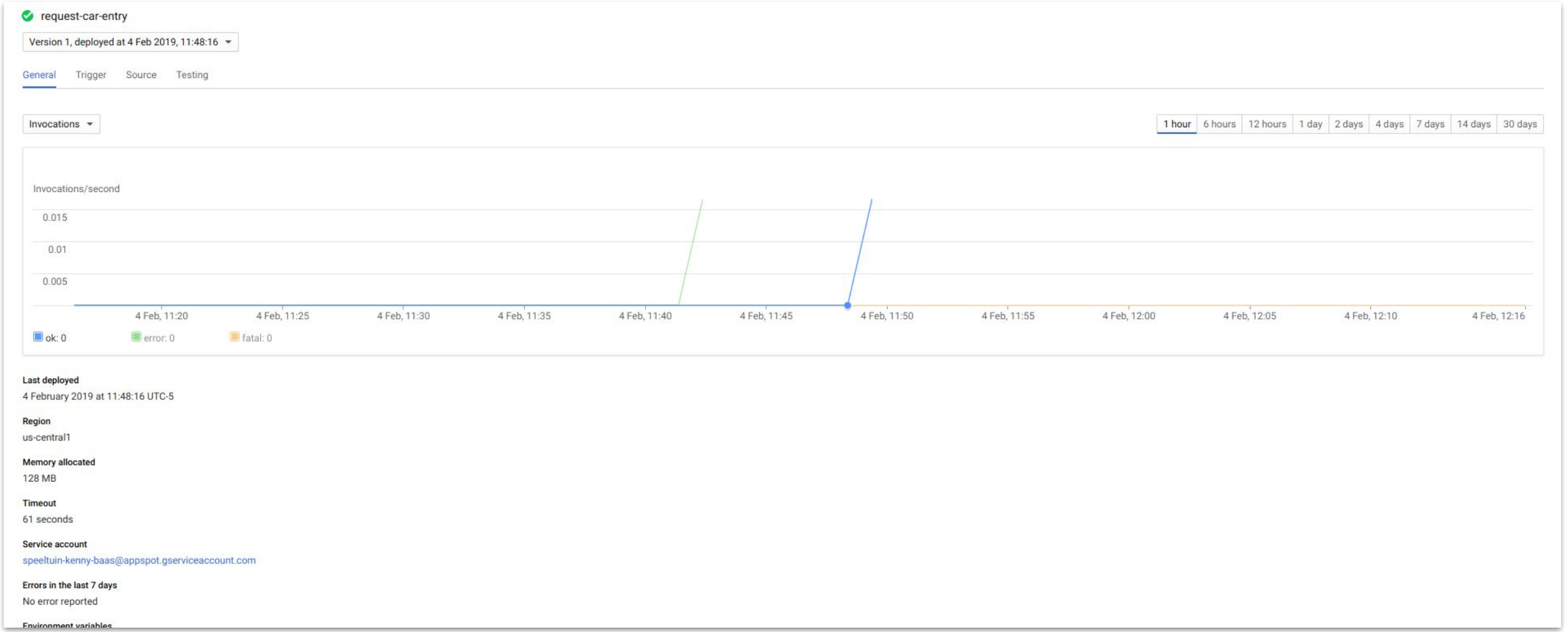
Monitoring - AWS



Monitoring - Azure



Monitoring - GCP



Monitoring

	+	-
AWS Lambda	• CloudWatch is mature, gives the right amount of stats	
Azure Functions	• Basic logging OOTB and easy integration with Application Insights.	
GCP Functions	• Basic monitoring OOTB	

Key takeaways

Question: When do we use **AWS**?

Heuristic: When we want to use many different function runtimes.

Heuristic: When we want to use the latest runtimes.

Key takeaways

Question: When do we use **Azure**?

Heuristic: When we're familiar with the Microsoft ecosystem.

Heuristic: When we want to orchestrate functions.

Key takeaways

Question: When do we use **Google Cloud**?

Heuristic: When we want the best functions portal experience.

Key takeaways

Question: When should we use **Orchestrations**?

Heuristic: Use orchestrations within a bounded context.

Heuristic: Use orchestrations when having processes over multiple modules in your bounded context.

Heuristic: Use orchestrations for long running processes.

Sources

Pricing

https://aws.amazon.com/lambda/pricing/
https://azure.microsoft.com/en-us/pricing/details/functions/
https://cloud.google.com/functions/pricing
Comparison:
https://docs.google.com/spreadsheets/d/1Q6vllVYe1CfHAK6MP6Uz6mTYPiU4DljwL8XN4E5FQdw/edit?usp=sharing

Runtimes & languages

https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html
https://docs.microsoft.com/en-us/azure/azure-functions/supported-languages
https://cloud.google.com/functions/docs/writing/

Cold Start

https://mikhail.io/2018/08/serverless-cold-start-war/

Github Repository with our demo code

https://github.com/Baasie/serverless-showdown