Amazon Web Services

An Experience Report

Bryan Nehl

@k0emt

Background

```
$ whoami
```

Bryan Nehl

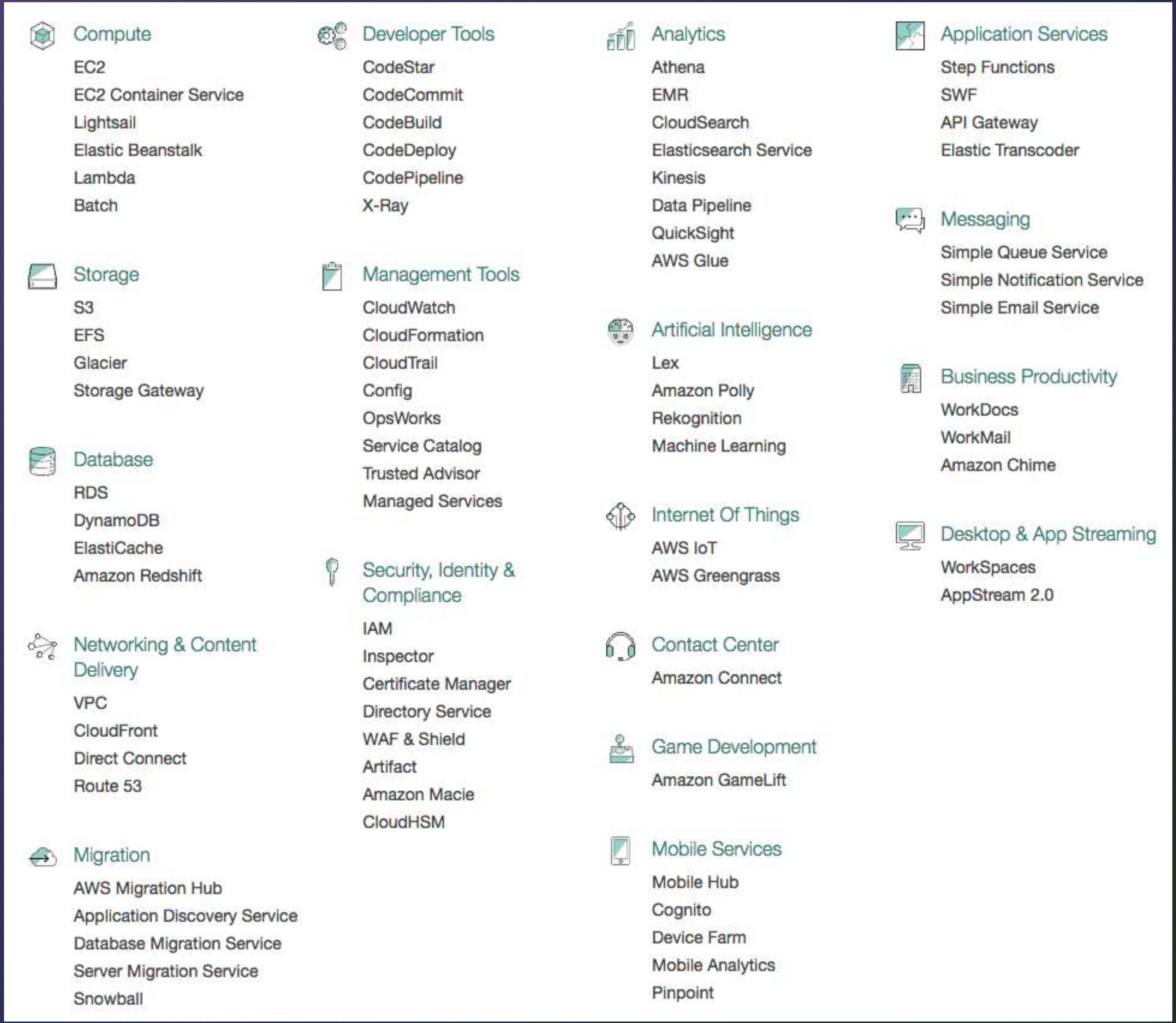
Developer on the myCARFAX Team

dbBear.com

@k0emt

Background

Why have I been delving into AWS stuff?



Compute
- EC2
- EC2 Container Service
- Lightsail
- Elastic Beanstalk
- Lambda
- Batch

Storage
- S3
- EFS
- Glacier
- Storage Gateway

Database
- RDS
- DynamoDB
- ElastiCache
- Amazon Redshift

Networking & Content Delivery
- VPC
- CloudFront
- Direct Connect
- Route 53

Migration
- AWS Migration Hub
- Application Discovery Service
- Database Migration Service
- Server Migration Service
- Snowball

Developer Tools
- CodeStar
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline
- X-Ray

Management Tools
- CloudWatch
- CloudFormation
- CloudTrail
- Config
- OpsWorks
- Service Catalog
- Trusted Advisor
- Managed Services

Security, Identity & Compliance
- IAM
- Inspector
- Certificate Manager
- Directory Service
- WAF & Shield
- Artifact
- Amazon Macie
- CloudHSM

Analytics
- Athena
- EMR
- CloudSearch
- Elasticsearch Service
- Kinesis
- Data Pipeline
- QuickSight
- AWS Glue

Artificial Intelligence
- Lex
- Amazon Polly
- Rekognition
- Machine Learning

Internet Of Things
- AWS IoT
- AWS Greengrass

Contact Center
- Amazon Connect

Game Development
- Amazon GameLift

Mobile Services
- Mobile Hub
- Cognito
- Device Farm
- Mobile Analytics
- Pinpoint

Application Services
- Step Functions
- SWF
- API Gateway
- Elastic Transcoder

Messaging
- Simple Queue Service
- Simple Notification Service
- Simple Email Service

Business Productivity
- WorkDocs
- WorkMail
- Amazon Chime

Desktop & App Streaming
- WorkSpaces
- AppStream 2.0

Identity and Access Management

- Users
- Groups
- Permissions
- Auditing

https://aws.amazon.com/iam/

http://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/getting-your-credentials.html

Static Content Website

S3 - Fundamentals

- Simple Storage Service
- https://aws.amazon.com/sdk-for-node-js/
- http://docs.aws.amazon.com/AWSJavaScriptSDK/latest/
- http://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-examples.html
- http://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-example-photo-album.html
- http://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html
- http://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-example-creating-buckets.html

Setting up a website in AWS

Follow the instructions!

https://docs.aws.amazon.com/AmazonS3/latest/dev/website-hosting-custom-domain-walkthrough.html

Setting up a website in AWS

S3

- Bucket must be publicly readable

Setting up a website in AWS

Amazon Certificate Manager (ACM)

domain.com *and* *.domain.com

Setting up a website in AWS

Route 53

S3 bucket redirect for
www.domain.com ➡ domain.com

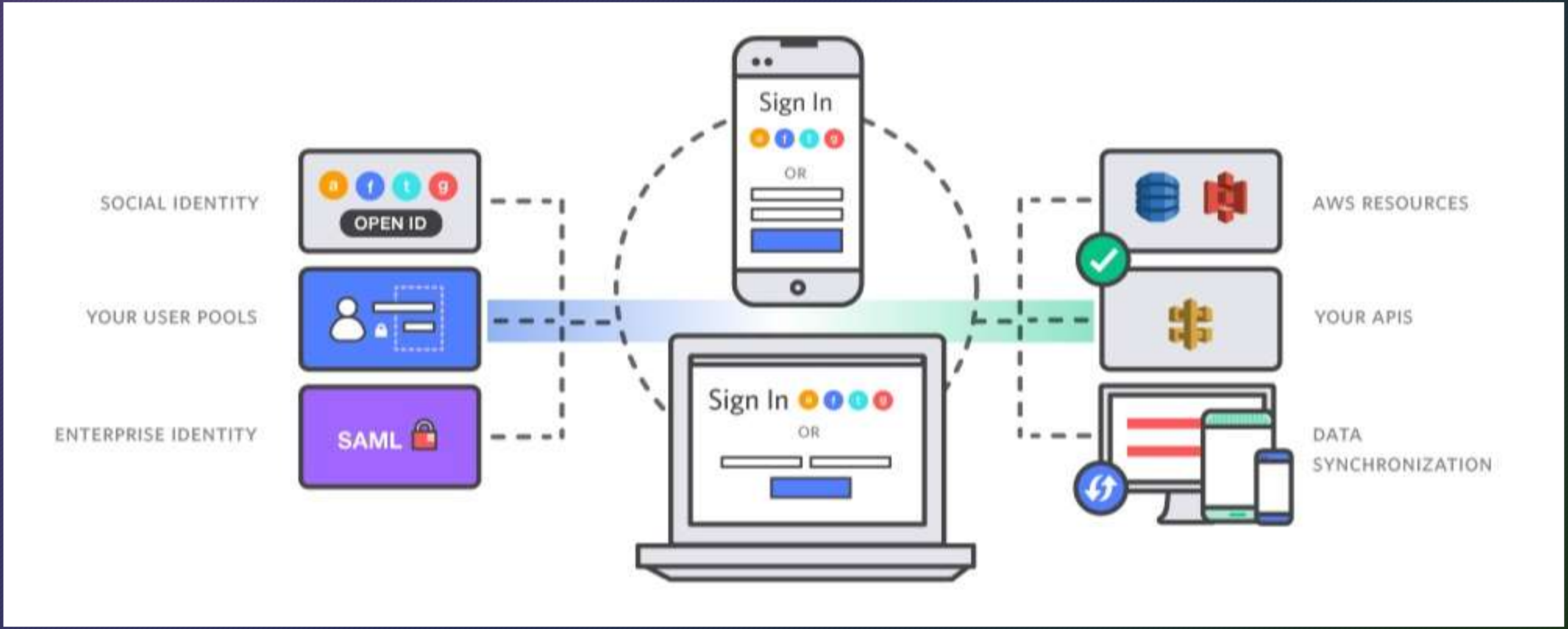
Setting up a website in AWS

CloudFront

- Content Delivery Network

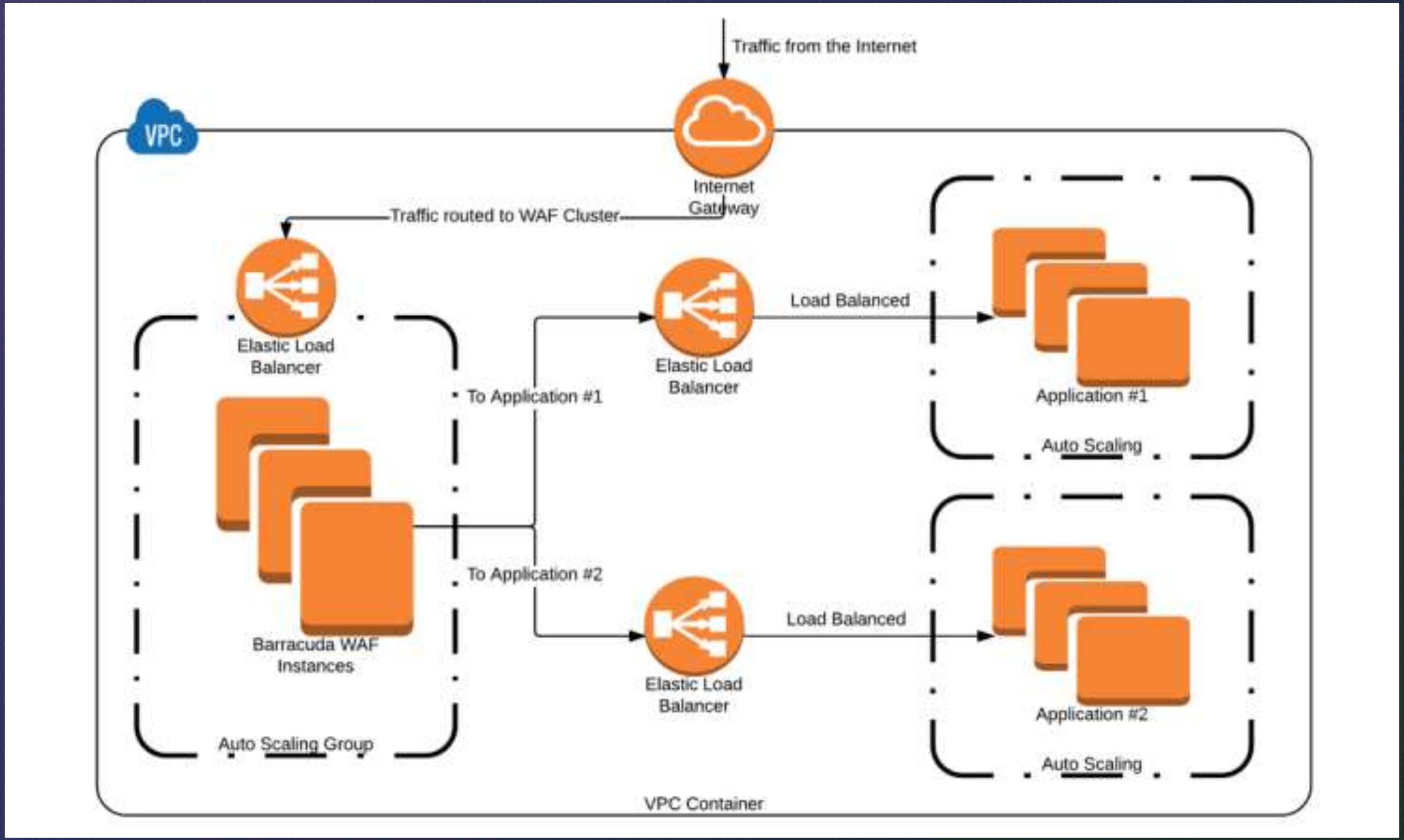
Setting up a website in AWS

Cognito



Setting up a website in AWS

Web Application Firewall



Back of the House

What’s a VPC?

Virtual Private Cloud

https://aws.amazon.com/vpc

Relational Database Service (RDS)

- Read the docs first!
- https://aws.amazon.com/getting-started/tutorials/create-connect-postgresql-db/ *(NOTE: Don't use the sql workbench app)*
- http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.PostgreSQL.html

(NOTE: DO use pgAdmin)

Relational Database Service (RDS)

- For node.js, need the `pg` package and potentially the `pg-cursor` package.
- http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-nodejs.rds.html

- node-postgress (pg) for client
- https://node-postgres.com/
- https://node-postgres.com/features/connecting
- https://node-postgres.com/guides/upgrading
- https://node-postgres.com/api/cursor

Relational Database Service (RDS)

- Follow database best practices: create and use appropriately authorized roles, etc.
  - https://stackoverflow.com/questions/760210/how-do-you-create-a-read-only-user-in-postgresql
  - https://www.postgresql.org/docs/9.1/static/sql-grant.html
- **Encryption**
- http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html
- You do want your db externally accessible, *for you!*

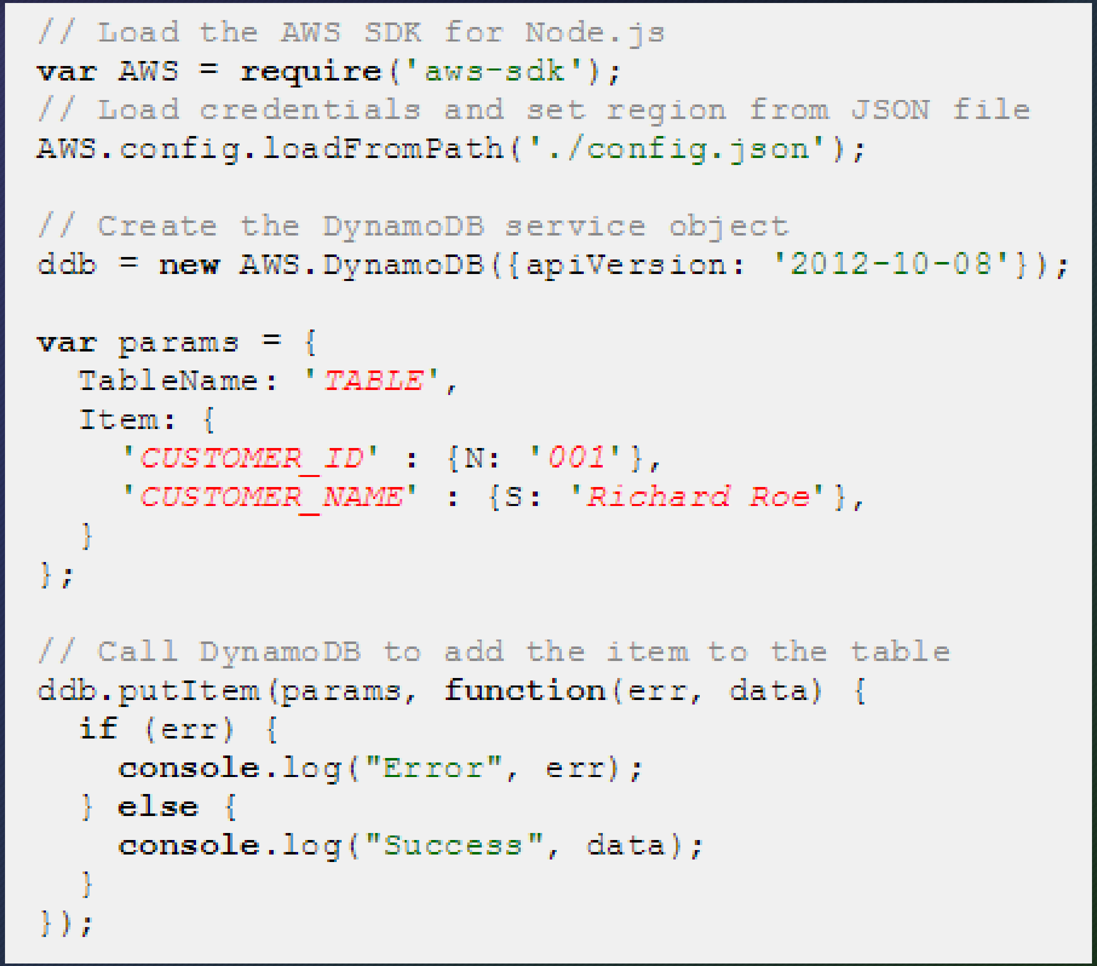
Relational Database Service (RDS)

- Access to RDS from outside
  - Create a security group
  - Modify the security
    - choose the inbound tab and edit it
      - Add PostgreSQL, port 5432 and the correct source
      - Verify that you have all traffic configured for the VPC and source
      - You'll need to have the IP range that you are originating from

DynamoDB

NOSQL?

- Cost
- Management
- Consider the data



```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Load credentials and set region from JSON file
AWS.config.loadFromPath('./config.json');

// Create the DynamoDB service object
ddb = new AWS.DynamoDB({apiVersion: '2012-10-08'});

var params = {
  TableName: 'TABLE',
  Item: {
    'CUSTOMER_ID' : {N: '001'},
    'CUSTOMER_NAME' : {S: 'Richard Roe'},
  }
};

// Call DynamoDB to add the item to the table
ddb.putItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Example from http://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/dynamodb-example-table-read-write.html

Cloud Watch

- Log all the things!
- Can use to trigger events

Lambda Functions

- Serverless Cloud Computing
- Comprehensive Developer Guide PDF *Great resource!*
  - http://docs.aws.amazon.com/lambda/latest/dg/lambda-dg.pdf
- http://docs.aws.amazon.com/lambda/latest/dg/welcome.html
- http://docs.aws.amazon.com/lambda/latest/dg/programming-model.html
- http://docs.aws.amazon.com/lambda/latest/dg/tutorial-env_console.html

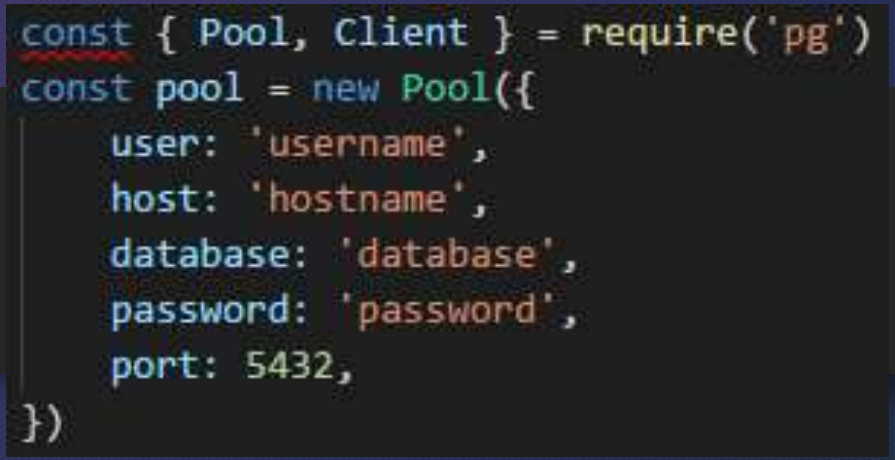
Lambda Functions

- IAM role needs execution access
- Complete the lambda
  - context.done()
  - context.succeed()
  - context.fail()

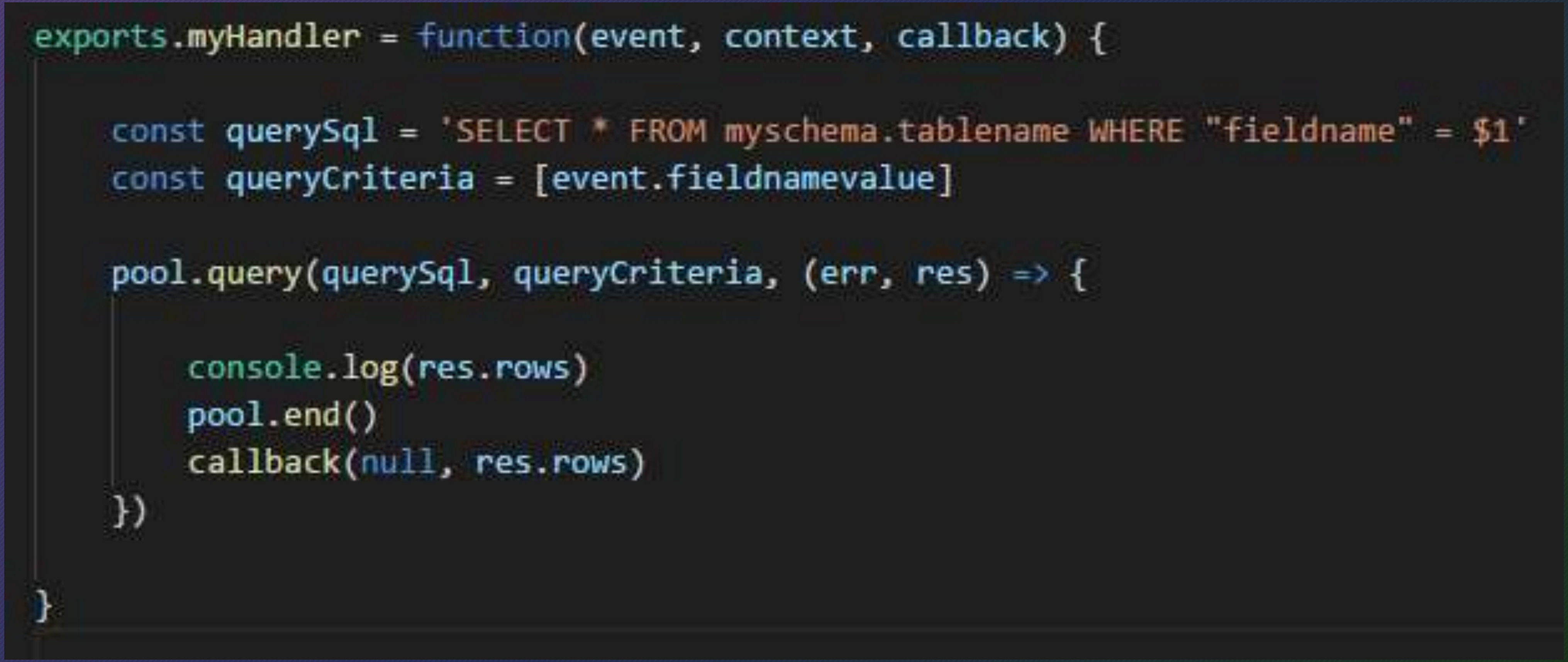
Lambda Functions

- Deploy
  - Via the AWS web console
  - Uploaded zip
    - don’t include the host directory
    - zip -r -X ../deploy.zip *
  - S3 Hosted zip
    - private bucket

Lambda Functions



```
const { Pool, Client } = require('pg')
const pool = new Pool({
    user: 'username',
    host: 'hostname',
    database: 'database',
    password: 'password',
    port: 5432,
})
```



```
exports.myHandler = function(event, context, callback) {

    const querySql = 'SELECT * FROM myschema.tablename WHERE "fieldname" = $1'
    const queryCriteria = [event.fieldnamevalue]

    pool.query(querySql, queryCriteria, (err, res) => {

        console.log(res.rows)
        pool.end()
        callback(null, res.rows)
    })

}
```

Lambda Functions

- Keep your zip file as lean as possible.
- The lambda will also take longer to execute if it's running from a cold start.
- The body of a request response needs to be a string; so, if you're returning a JSON object, call JSON.stringify on it.
- *NOTE*: An S3 PUT overwrites existing content and triggers associated events.

Lambda Functions

Lambda Triggers and Retries

Events that set off lambda triggers such as an S3 put event may be fired **multiple** times. However, the same context.awsRequestId will come through with it. It is up to the developer to either create idempotent functions or deal with it. If the function fails to complete successfully, it will be automatically retried twice for asynchronous events. Synchronous invoked events throw a 429 error that the client has to handle.

Lambda Functions

Lambda Function to access VPC resources

http://docs.aws.amazon.com/lambda/latest/dg/vpc.html

In order to hit S3 from a lambda in a VPC, a VPC endpoint must be configured.

https://aws.amazon.com/blogs/aws/new-vpc-endpoint-for-amazon-s3/

http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-endpoints.html

http://docs.aws.amazon.com/lambda/latest/dg/best-practices.html

API Gateway

- http://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html
- https://us-east-2.console.aws.amazon.com/apigateway/home?region=us-east-2#/apis/q6xlco0s52/stages/test

- If you stumble across a “Missing Authentication Token” when querying the API, you’re most likely not using the correct URL.

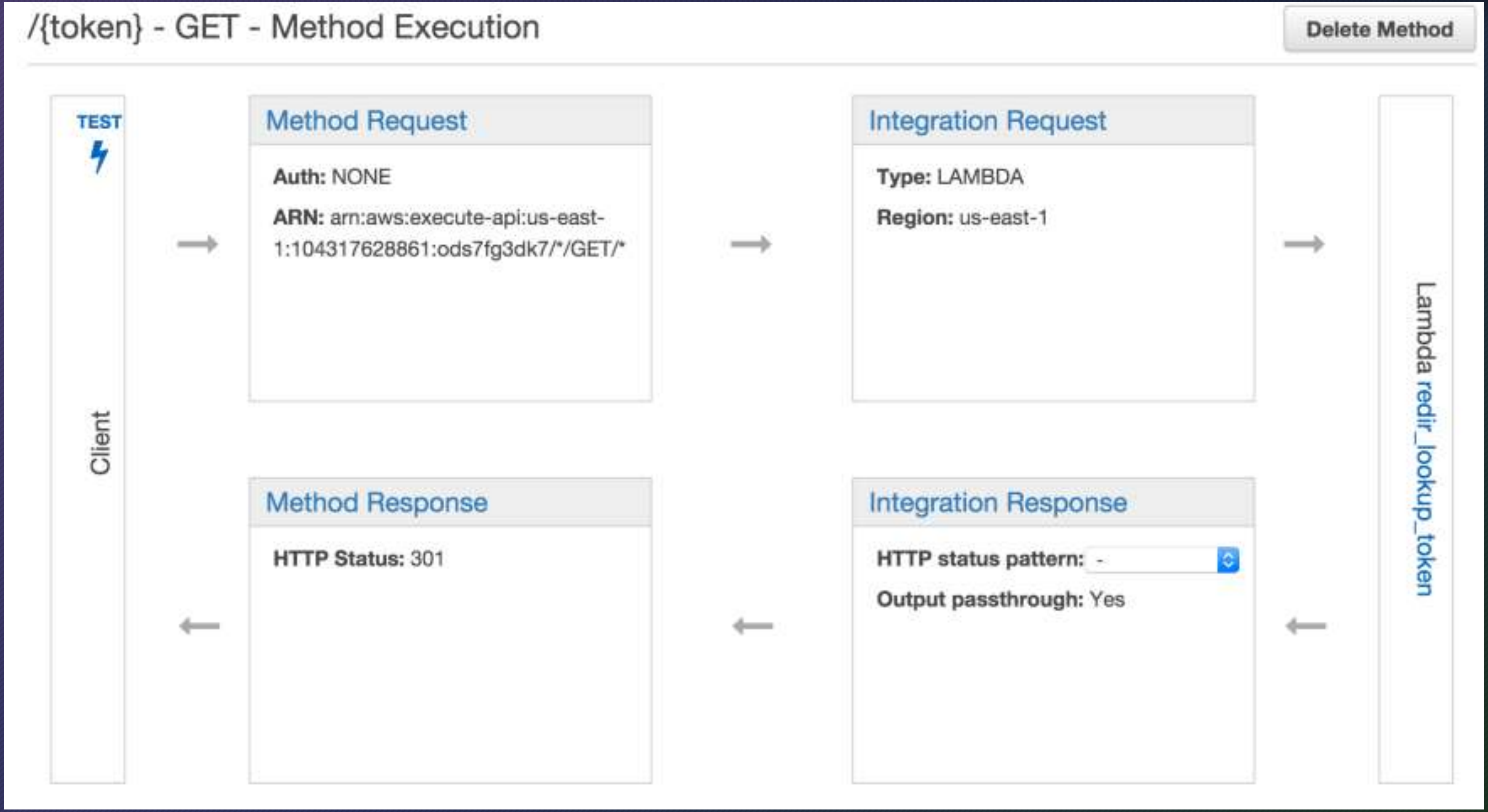
API Gateway

- You can use a custom authorizer with the Gateway, implemented with a lambda function of course!
  http://docs.aws.amazon.com/apigateway/latest/developerguide/use-custom-authorizer.html

API Gateway - Parameters

- http://docs.aws.amazon.com/apigateway/latest/developerguide/integrating-api-with-aws-services-lambda.html
- http://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-mapping-template-reference.html
- https://medium.com/simple-thoughts-amplified/passing-variables-from-aws-api-gateway-to-lambda-3c5d8602081b
- https://stackoverflow.com/questions/31329958/how-to-pass-a-querystring-or-route-parameter-to-aws-lambda-from-amazon-api-gatew

API Gateway



/{token} - GET - Method Execution

Method Request

Auth: NONE

ARN: arn:aws:execute-api:us-east-1:104317628861:ods7fg3dk7/*/GET/*

Integration Request

Type: LAMBDA

Region: us-east-1

Lambda redir_lookup_token

Method Response

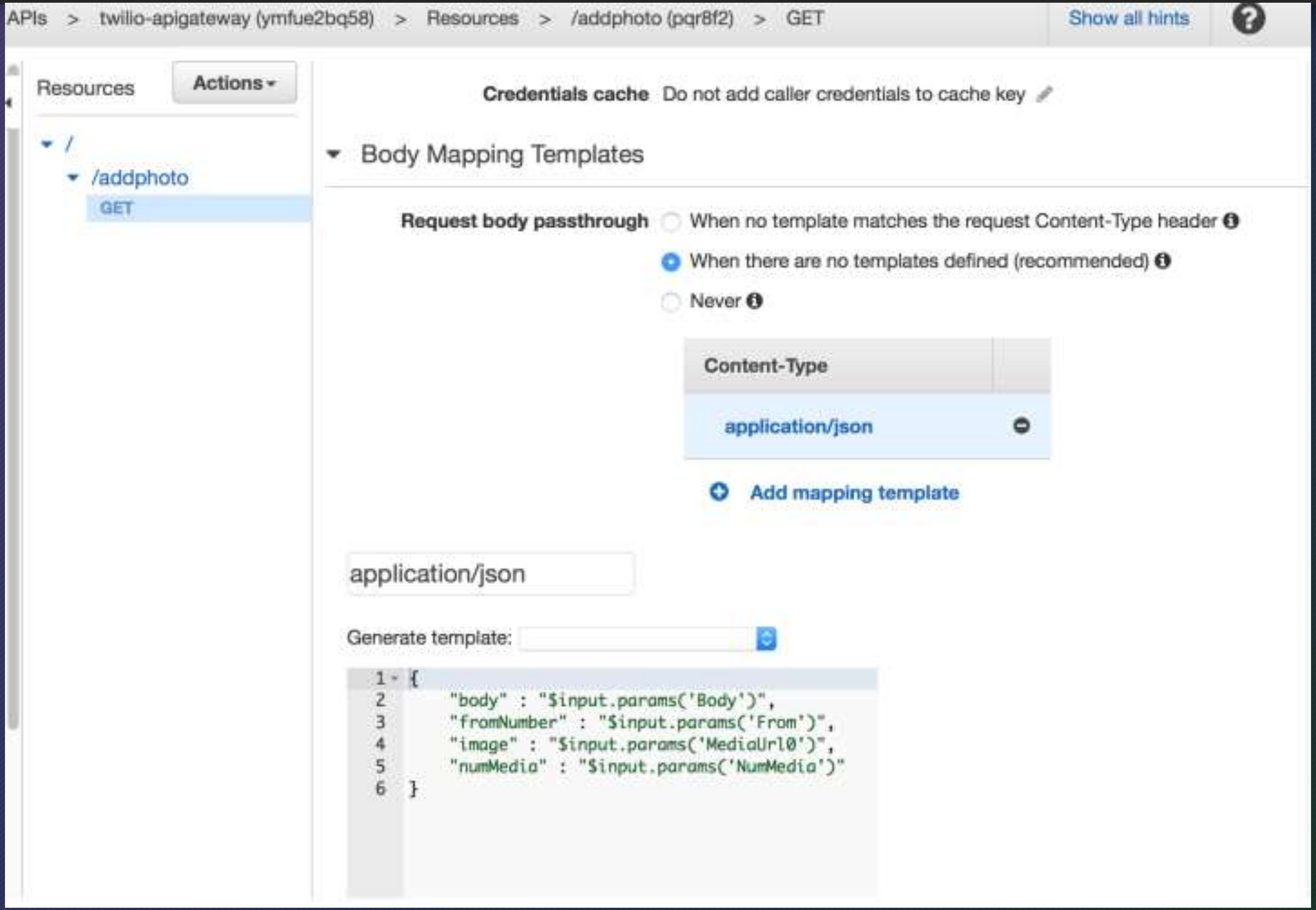
HTTP Status: 301

Integration Response

HTTP status pattern: -

Output passthrough: Yes

API Gateway


APIs > twilio-apigateway (ymfue2bq58) > Resources > /addphoto (pqr8f2) > GET

Credentials cache Do not add caller credentials to cache key

Body Mapping Templates

Request body passthrough
- When no template matches the request Content-Type header
- [✓ When there are no templates defined (recommended)]
- Never

Content-Type
application/json

application/json

Generate template:

```
{
    "body" : "$input.params('Body')",
    "fromNumber" : "$input.params('From')",
    "image" : "$input.params('MediaUrl0')",
    "numMedia" : "$input.params('NumMedia')"
}
```

API Gateway

- DEPLOY when you've finished testing
- You may not always get a successful execution
  - { "errorMessage": "RequestId: 09097e53-899c-11e7-a682-e9b4420d4070 Process exited before completing request" }
- Note: the above error response is still a status 200! why?
- Handle the failure (including retries)
- Identify any potential issues in the lambda function code and correct/optimize
- Adjust settings on the backing lambda function

API Gateway

GOTCHA!

If when you test externally, like with postman, you get a:

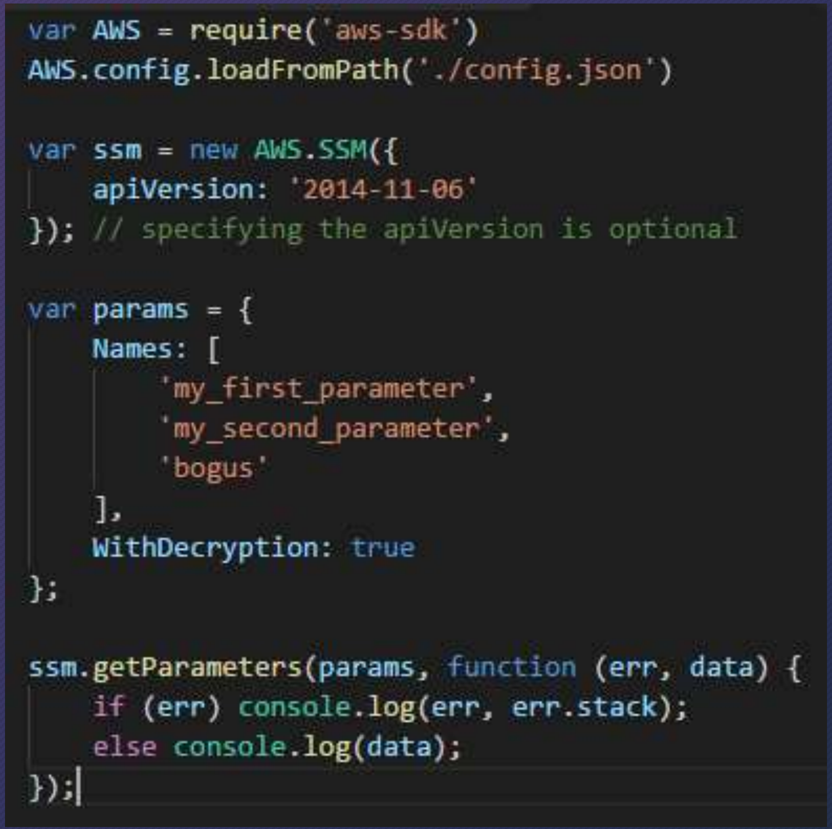
{ "message" : "Missing Authentication Token" }

It is likely that you either:

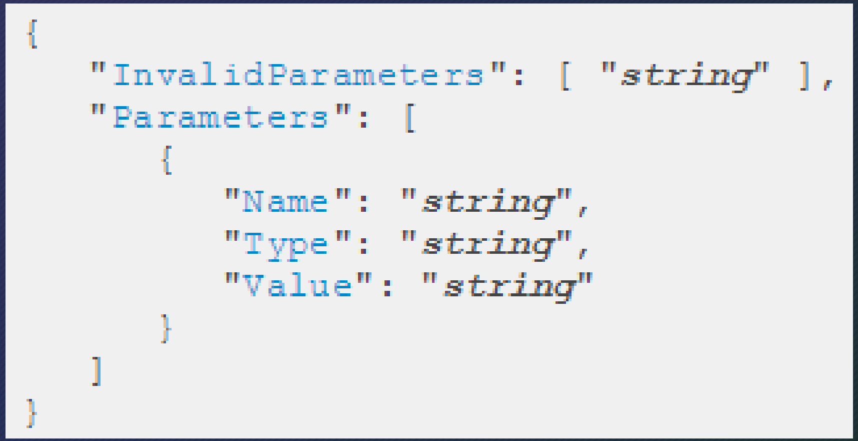
- haven't deployed the API
- you have an incorrectly spelled URL/method

EC2 - Simple System Parameters

- https://aws.amazon.com/ec2/systems-manager/parameter-store/



```javascript
var AWS = require('aws-sdk')
AWS.config.loadFromPath('./config.json')

var ssm = new AWS.SSM({
    apiVersion: '2014-11-06'
}); // specifying the apiVersion is optional

var params = {
    Names: [
        'my_first_parameter',
        'my_second_parameter',
        'bogus'
    ],
    WithDecryption: true
};

ssm.getParameters(params, function (err, data) {
    if (err) console.log(err, err.stack);
    else console.log(data);
});
```



```json
{
   "InvalidParameters": [ "string" ],
   "Parameters": [
      {
         "Name": "string",
         "Type": "string",
         "Value": "string"
      }
   ]
}
```

Construction

Building the Environment

Command Line Interface (CLI)

http://docs.aws.amazon.com/cli/latest/userguide/installing.html

http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html)

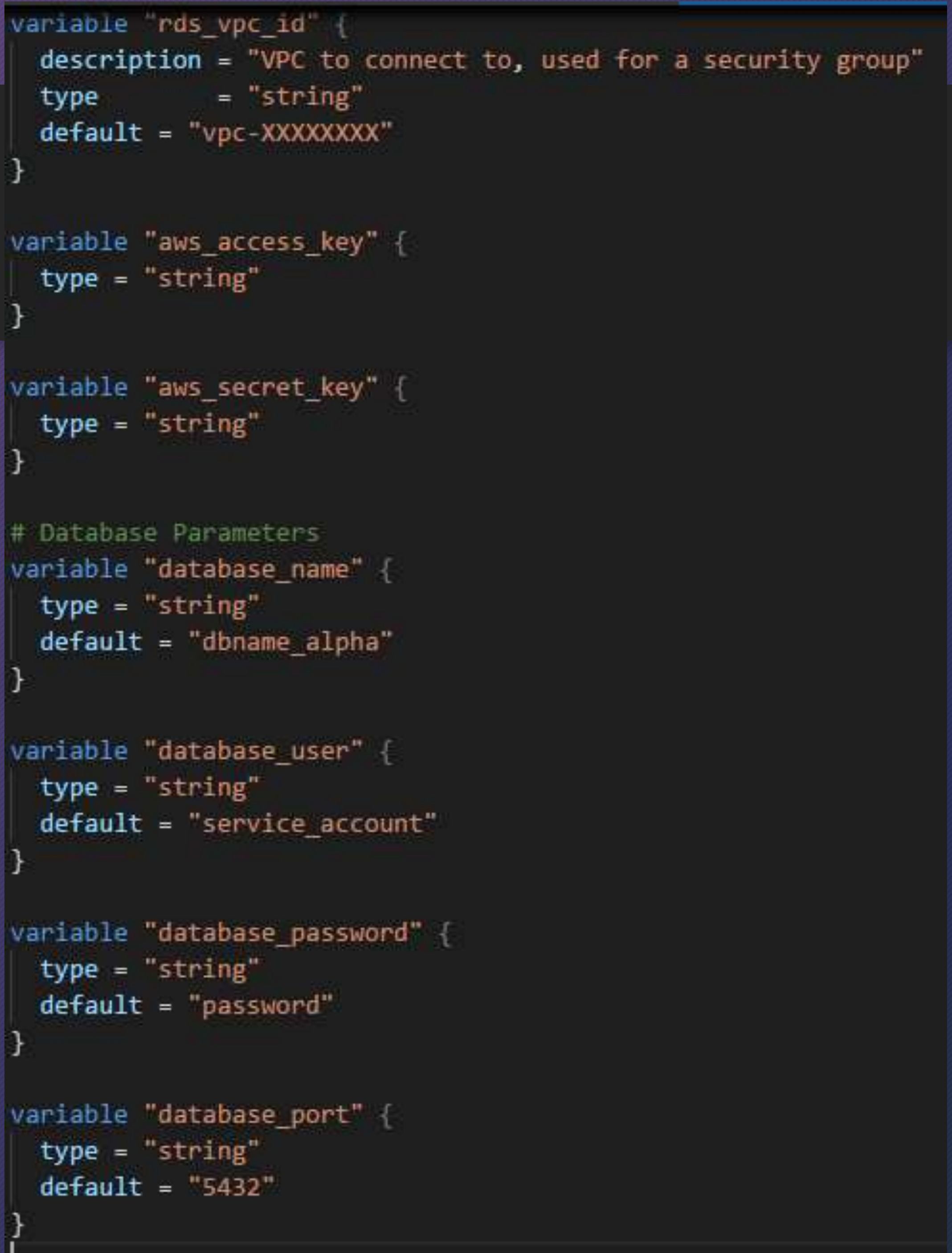
- aws

```
aws ssm put-parameter --name a_name --value "a value" --type SecureString
```

```
aws s3api create-bucket --bucket my-bucket --region us-east-1
```

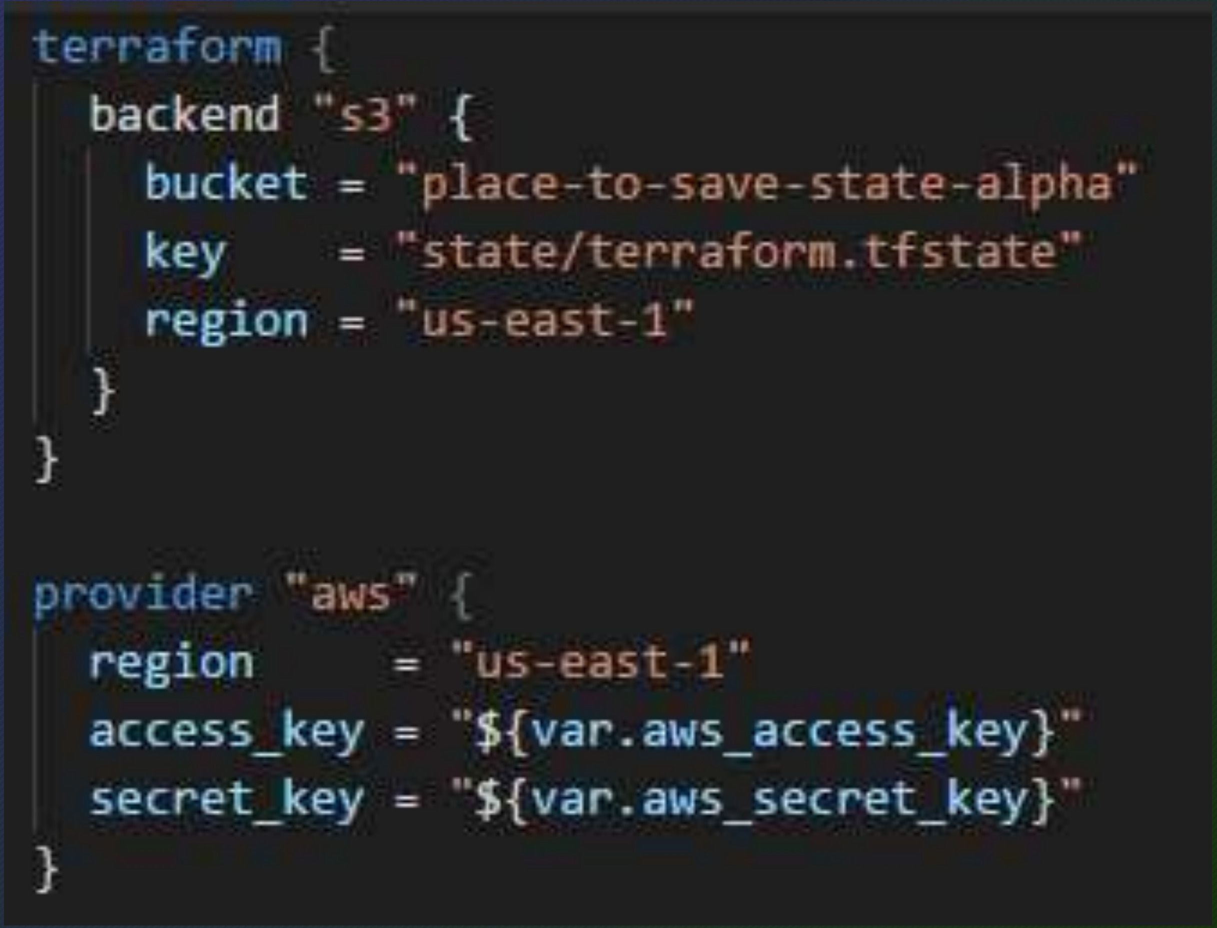
Building the Environment

- Amazon’s solution is Cloud Formation
  - https://aws.amazon.com/cloudformation/
- Terraform
  - https://www.terraform.io/
  - https://www.terraform.io/intro/getting-started/build.html
  - https://medium.com/build-acl/aws-lambda-deployment-with-terraform-24d36cc86533

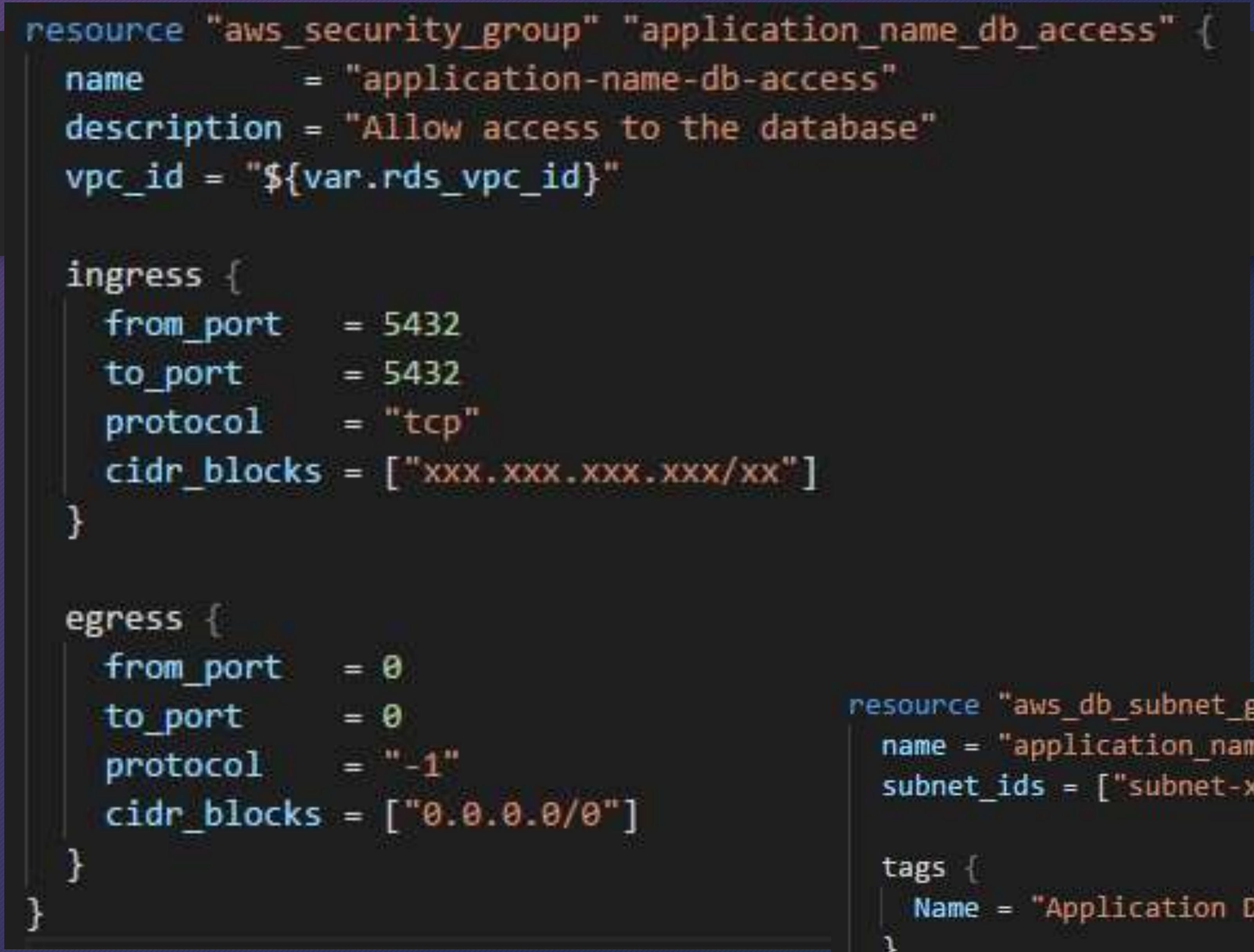


```
variable "rds_vpc_id" {
  description = "VPC to connect to, used for a security group"
  type        = "string"
  default = "vpc-XXXXXXXX"
}

variable "aws_access_key" {
  type = "string"
}

variable "aws_secret_key" {
  type = "string"
}

# Database Parameters
variable "database_name" {
  type = "string"
  default = "dbname_alpha"
}

variable "database_user" {
  type = "string"
  default = "service_account"
}

variable "database_password" {
  type = "string"
  default = "password"
}

variable "database_port" {
  type = "string"
  default = "5432"
}
```

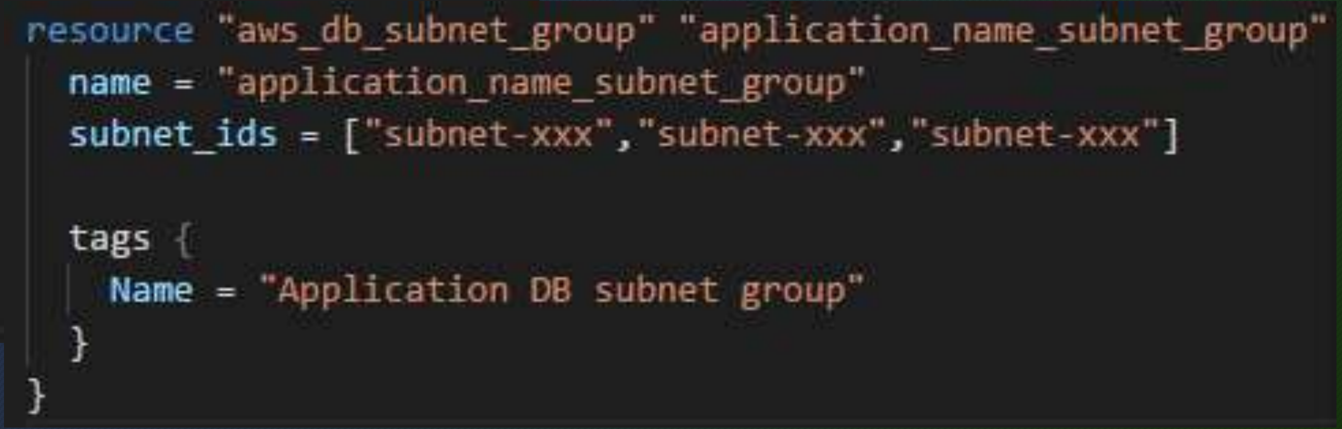
HashiCorp Terraform



```
terraform {
  backend "s3" {
    bucket = "place-to-save-state-alpha"
    key    = "state/terraform.tfstate"
    region = "us-east-1"
  }
}

provider "aws" {
  region     = "us-east-1"
  access_key = "${var.aws_access_key}"
  secret_key = "${var.aws_secret_key}"
}
```

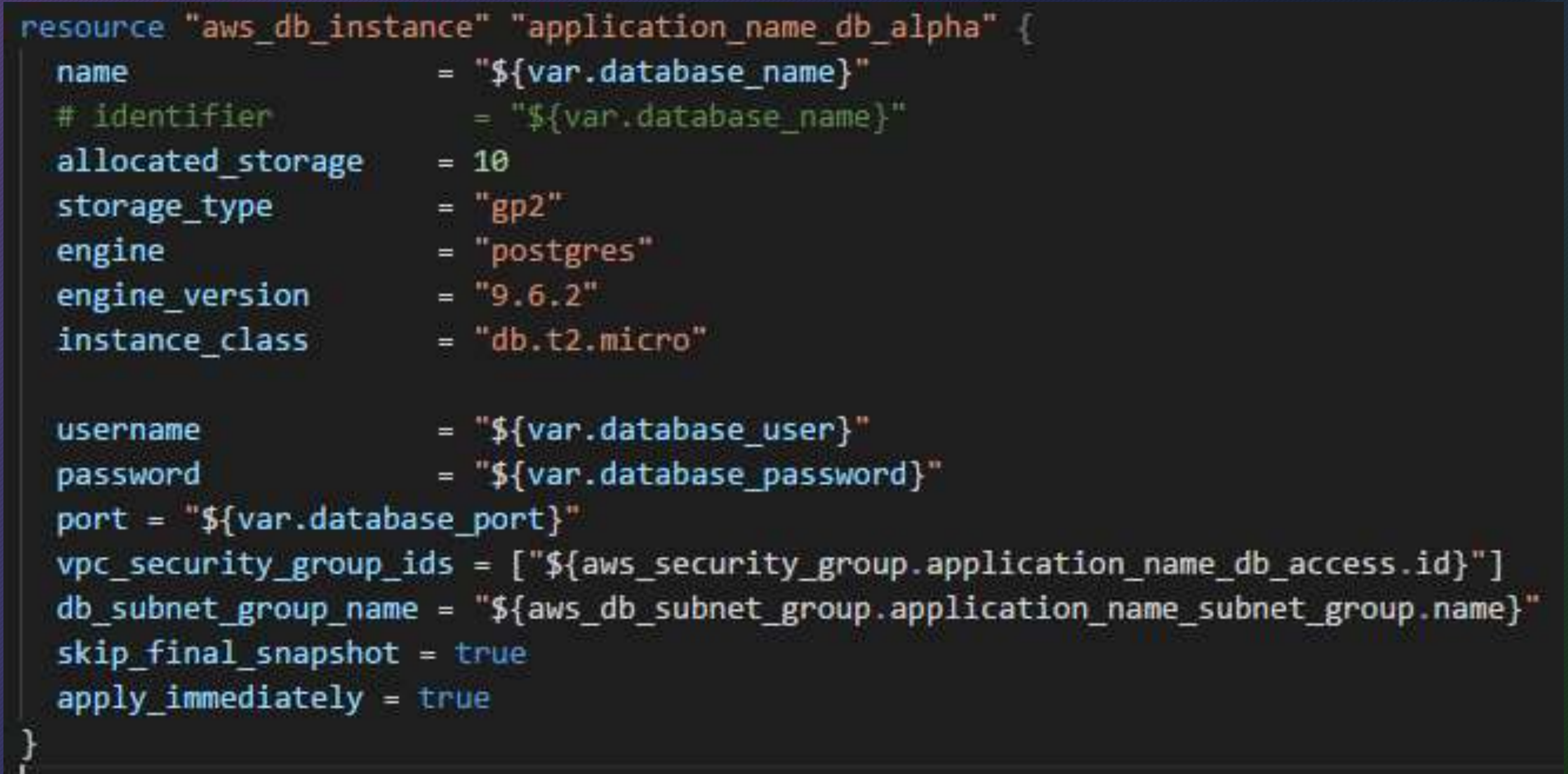


```
resource "aws_security_group" "application_name_db_access" {
  name        = "application-name-db-access"
  description = "Allow access to the database"
  vpc_id = "${var.rds_vpc_id}"

  ingress {
    from_port   = 5432
    to_port     = 5432
    protocol    = "tcp"
    cidr_blocks = ["xxx.xxx.xxx.xxx/xx"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

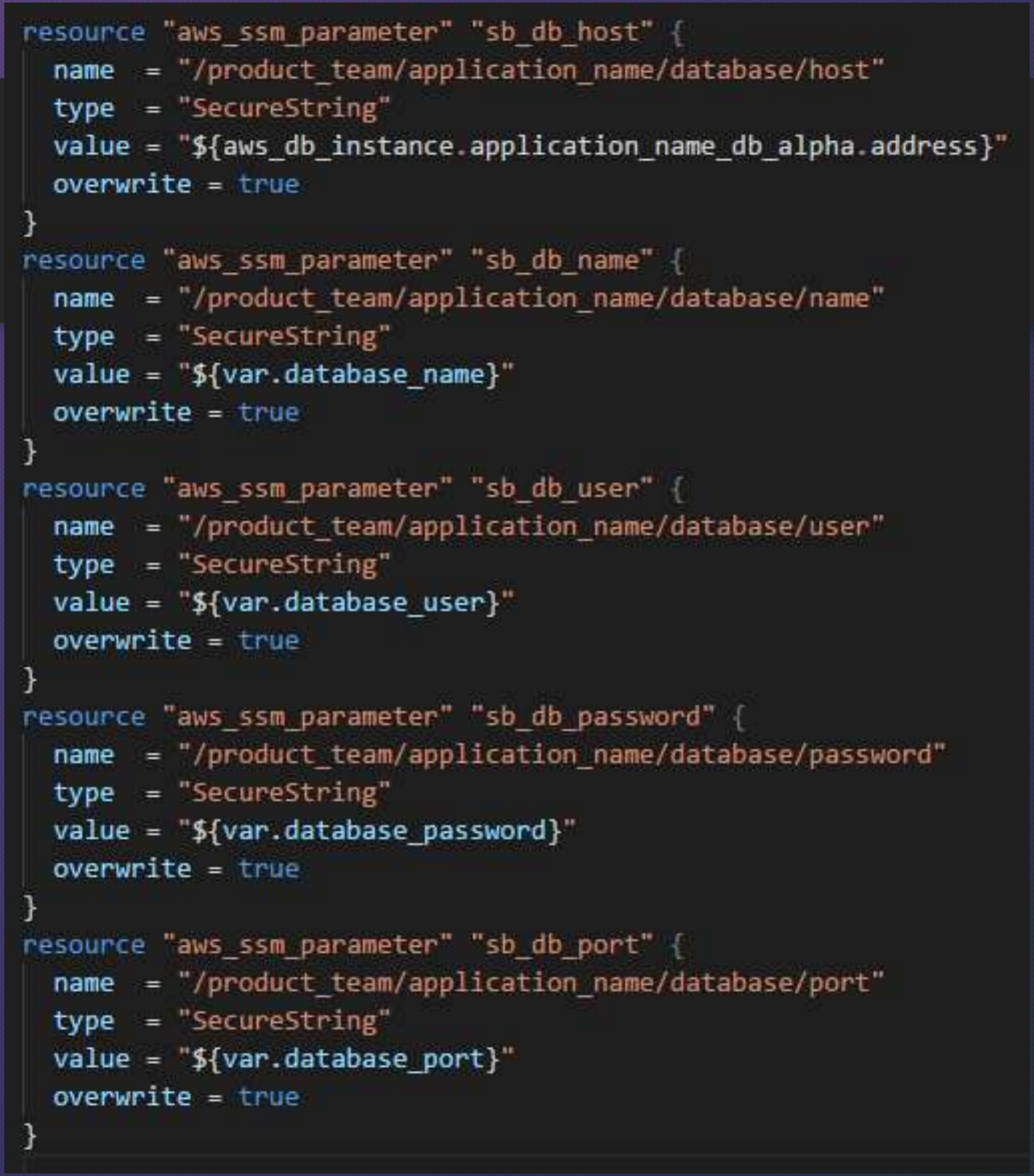


```
resource "aws_db_subnet_group" "application_name_subnet_group"
  name = "application_name_subnet_group"
  subnet_ids = ["subnet-xxx","subnet-xxx","subnet-xxx"]

  tags {
    Name = "Application DB subnet group"
  }
}
```

HashiCorp Terraform



```
resource "aws_db_instance" "application_name_db_alpha" {
  name                    = "${var.database_name}"
  # identifier              = "${var.database_name}"
  allocated_storage     = 10
  storage_type            = "gp2"
  engine                  = "postgres"
  engine_version          = "9.6.2"
  instance_class          = "db.t2.micro"

  username                = "${var.database_user}"
  password                = "${var.database_password}"
  port = "${var.database_port}"
  vpc_security_group_ids = ["${aws_security_group.application_name_db_access.id}"]
  db_subnet_group_name = "${aws_db_subnet_group.application_name_subnet_group.name}"
  skip_final_snapshot = true
  apply_immediately = true
}
```



```
resource "aws_ssm_parameter" "sb_db_host" {
  name  = "/product_team/application_name/database/host"
  type  = "SecureString"
  value = "${aws_db_instance.application_name_db_alpha.address}"
  overwrite = true
}
resource "aws_ssm_parameter" "sb_db_name" {
  name  = "/product_team/application_name/database/name"
  type  = "SecureString"
  value = "${var.database_name}"
  overwrite = true
}
resource "aws_ssm_parameter" "sb_db_user" {
  name  = "/product_team/application_name/database/user"
  type  = "SecureString"
  value = "${var.database_user}"
  overwrite = true
}
resource "aws_ssm_parameter" "sb_db_password" {
  name  = "/product_team/application_name/database/password"
  type  = "SecureString"
  value = "${var.database_password}"
  overwrite = true
}
resource "aws_ssm_parameter" "sb_db_port" {
  name  = "/product_team/application_name/database/port"
  type  = "SecureString"
  value = "${var.database_port}"
  overwrite = true
}
```

HashiCorp Terraform

Building the Environment - Serverless

Serverless is cloud platform agnostic

https://serverless.com/

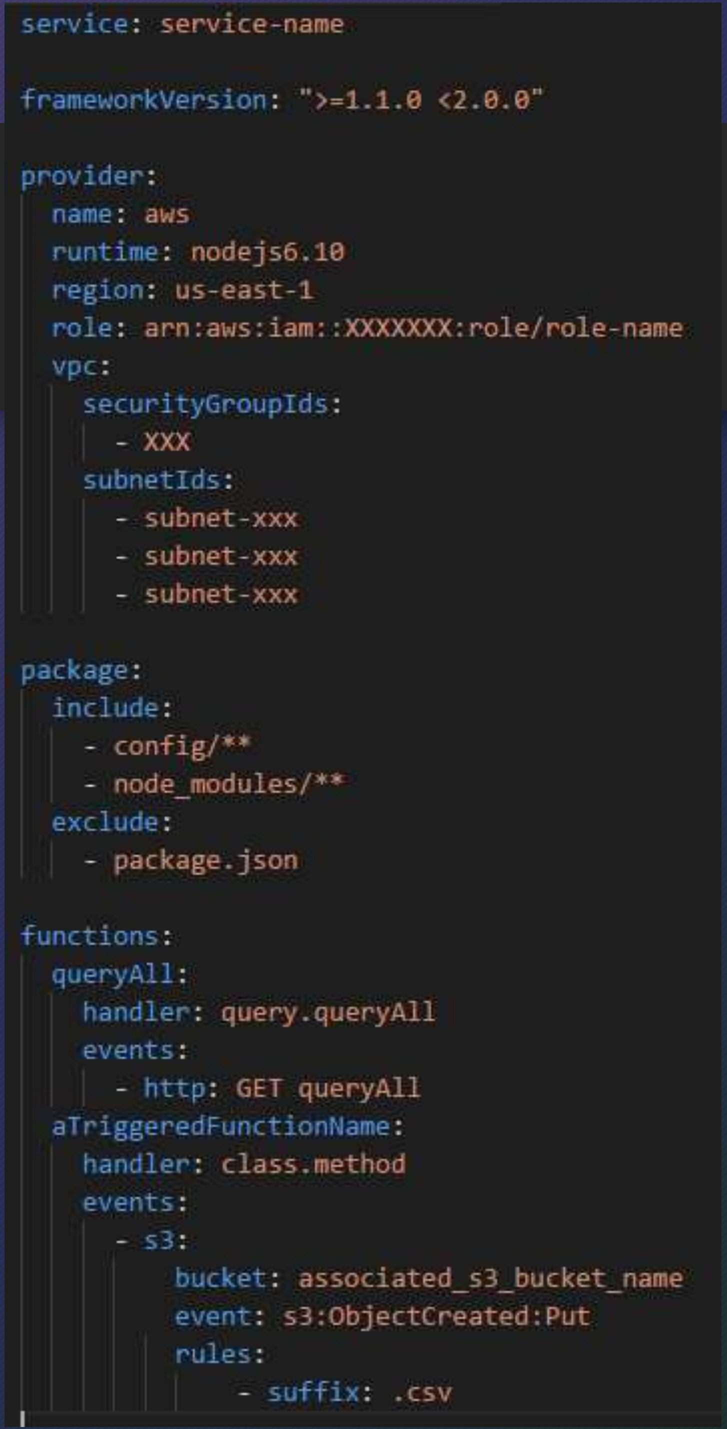
https://serverless.com/framework/docs/providers/aws/guide/functions/

https://serverless.com/framework/docs/providers/aws/guide/deploying/

Serverless Setup also see the AWS Profile Manager

https://serverless.com/framework/docs/providers/aws/guide/quick-start/

https://github.com/DavidWells/aws-profile-manager



```
service: service-name

frameworkVersion: ">=1.1.0 <2.0.0"

provider:
  name: aws
  runtime: nodejs6.10
  region: us-east-1
  role: arn:aws:iam::XXXXXXX:role/role-name
  vpc:
    securityGroupIds:
      - XXX
    subnetIds:
      - subnet-xxx
      - subnet-xxx
      - subnet-xxx

package:
  include:
    - config/**
    - node_modules/**
  exclude:
    - package.json

functions:
  queryAll:
    handler: query.queryAll
    events:
      - http: GET queryAll
  aTriggeredFunctionName:
    handler: class.method
    events:
      - s3:
          bucket: associated_s3_bucket_name
          event: s3:ObjectCreated:Put
          rules:
            - suffix: .csv
```

Building the Environment – Serverless Stack

Serverless Stack is AWS and React focused, uses Serverless
https://serverless-stack.com/

make sure the indentation for your serverless.yml is right, or it might fail some configuration items silently

Videos:

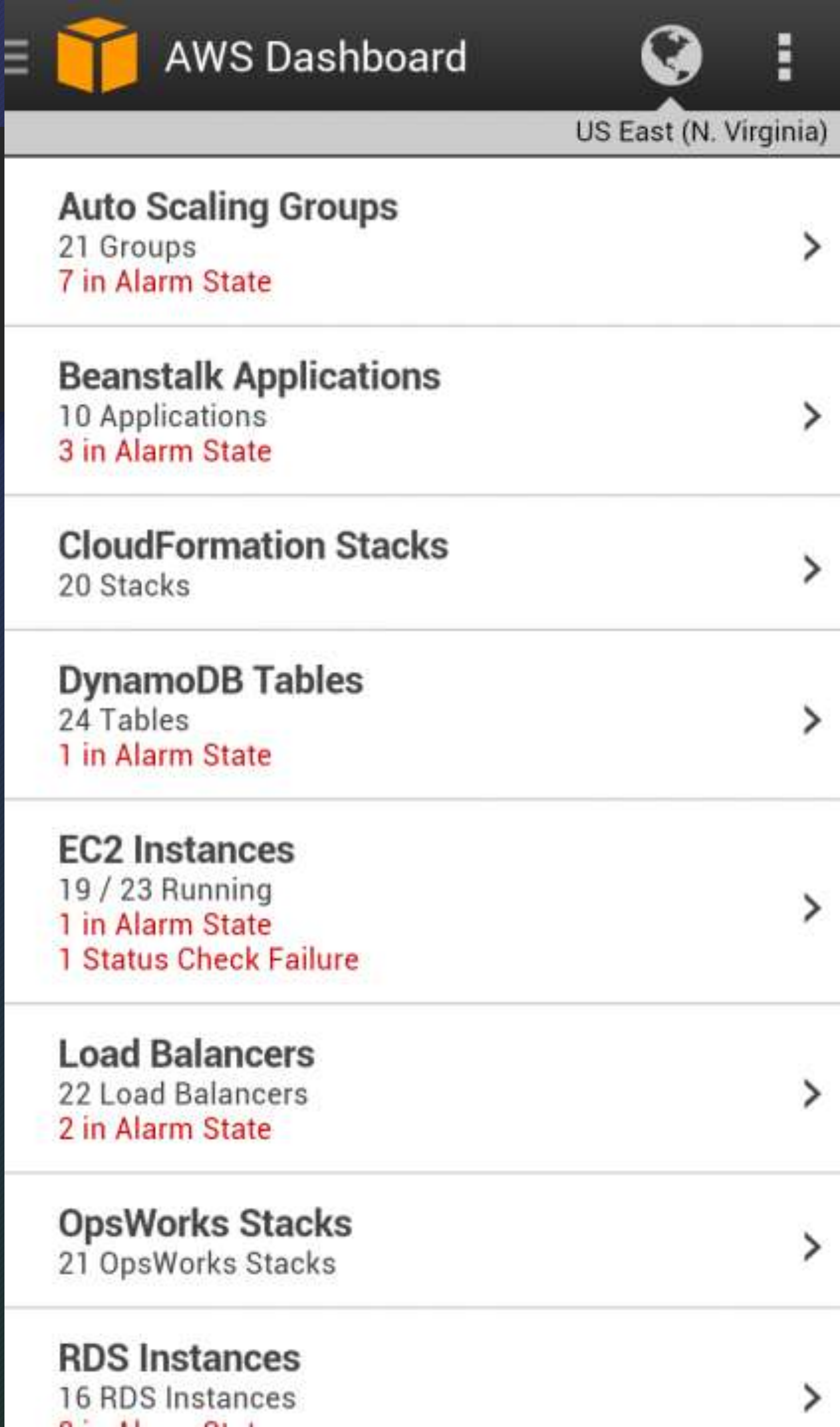
- Credentials setup
- Creating and deploying lambda
- Exposing API Gateway endpoint

Final Thoughts

We’re almost done!

Mobile Apps

- Monitor your resources
- Reboot them
- Check your billing balances
- https://aws.amazon.com/console/mobile/



Auto Scaling Groups
21 Groups
7 in Alarm State

Beanstalk Applications
10 Applications
3 in Alarm State

CloudFormation Stacks
20 Stacks

DynamoDB Tables
24 Tables
1 in Alarm State

EC2 Instances
19 / 23 Running
1 in Alarm State
1 Status Check Failure

Load Balancers
22 Load Balancers
2 in Alarm State

OpsWorks Stacks
21 OpsWorks Stacks

RDS Instances
16 RDS Instances

Gotchas

- Not all AWS zones have all features
- S3 is case sensitive
- Need to include index/error html files at every level for hosted static content website
- Finding good JSON examples for AWS event types can be challenging
- Credentials

Documentation / Resources

- AWS Documentation on Kindle
- AWS podcast
- AWS well architected framework

Other Stuff to Check out

- Elastic Bean Stalk
- Key Store
- Market Place
  - Software as a Service
  - By Usage
  - By Subscription
- Continuous Integration / Deployment Tools
- X-Ray

Next Steps

Get an account!?

Baby Steps

Purpose Driven, time boxed spikes

https://github.com/k0emt/Presentations

Thank you!

Bryan Nehl

dbBear.com