



## SEINFELD AS A SERVICE

---

# A SERVICE ABOUT NOTHING

Presented by Julian Naydichev



## ABOUT ME

- ▶ Julian Naydichev
- ▶ Sytac Employee
  - ▶ [julian.naydichev@sytac.io](mailto:julian.naydichev@sytac.io)
- ▶ AWS Developer Certified
- ▶ Worked at Amazon/AWS for 6 years
- ▶ Seinfeld Fan



## GOAL

- ▶ The goal of this workshop is to give you some exposure to a few AWS services.
- ▶ We will interact with the following services:
  - ▶ API Gateway
  - ▶ Lambda
  - ▶ RDS
  - ▶ CloudFormation
  - ▶ Route 53 (Optional)



## ABOUT THE PROJECT

- ▶ I'm a huge fan of the TV show Seinfeld. It was a show about nothing, and as an homage to it, we're going to create a service about nothing.
- ▶ We will create an API that has several endpoints that provide some light information about the show.
  - ▶ A list of seasons
  - ▶ A list of episodes in a season
  - ▶ A random quote from any of the following:
    - ▶ The entire show
    - ▶ A specific season
    - ▶ A specific episode



## PREREQUISITES

- ▶ Your favorite YAML editor
- ▶ Your preferred IDE
  - ▶ Lambda currently supports NodeJS, Python, Ruby, Java, Go and .NET runtimes.
  - ▶ The provided source is written in Java.
- ▶ The AWS CLI
- ▶ The mysql CLI - for inserting data
- ▶ Git - all good projects use git
- ▶ A domain name (optional)



## GETTING STARTED

- ▶ To begin, you can download a copy of these slides, all of the code and references from the following git repository:
  - ▶ <https://github.com/naydichev/SaaS>
- ▶ The code is in the final form.
- ▶ The CloudFormation templates are broken into steps, use the leading numbers to determine order.



# AWS ACCOUNT ACCESS

- ▶ You will need to sign up or have access to an AWS account.
- ▶ Sign-up here: [https://portal.aws.amazon.com/billing/  
signup#/start](https://portal.aws.amazon.com/billing/signup#/start)
- ▶ All of the resources used within this presentation should fit within the free tier.
- ▶ You may still need to enter in a credit card, but as long as you close the account after we're done today, you shouldn't be charged.



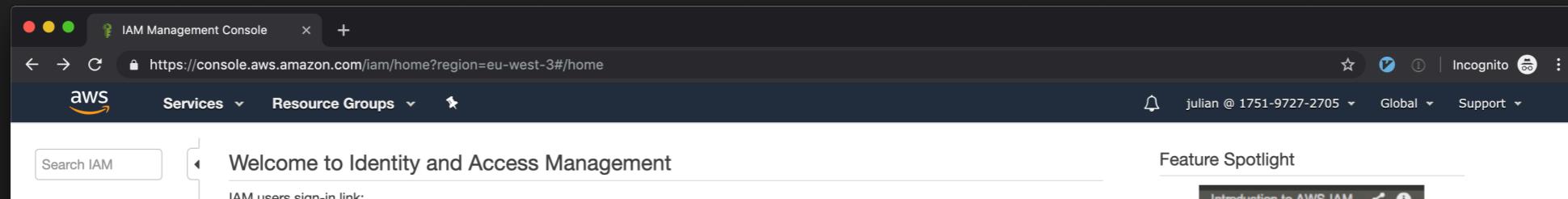
## LOCATION, LOCATION, LOCATION

- ▶ AWS has many regions, and they are continuously expanding.
- ▶ Not all of the regions have all of the services, nor are they all running the same versions as the other regions.
  - ▶ You should check the documentation to make sure that a service you want to use is available in your preferred region.
- ▶ For this presentation, I've chosen to use `eu-west-3`, because who doesn't romanticize Paris?
  - ▶ Unless otherwise indicated, make sure that you're using the same region for all the steps of this workshop.



# IAM - YOUR VERY OWN USER

- ▶ Now that you have access to the AWS Console, you should make yourself a new user.
- ▶ In general, logging in as the "root" user is considered bad practice, and you should instead create a user with Administrator privileges.
- ▶ You'll also want to create access keys so that you can use the AWS CLI (take note of these for later).
- ▶ Let's do that now...





## CLOUDFORMATION

- ▶ CloudFormation is the AWS approach to Infrastructure as a Service, or IaaS.
- ▶ Most AWS resources can be modeled and configured via CloudFormation.
- ▶ This allows you to have a clear picture of the resources your service uses, and you can keep the template in version control to track its changes over time.
- ▶ Templates are written in YAML or JSON. There are some intrinsic functions that are also available which make templating easier. You may have already noticed some of these in `00-start.yaml`.
- ▶ You can find general documentation about the various resources that you can create via CloudFormation here:
  - ▶ <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>
- ▶ There are several other tools available that help you create your CloudFormation templates.
- ▶ We will use vanilla CloudFormation for this workshop, but you can look into these tools if you're interested:
  - ▶ Serverless - <https://serverless.com/>
  - ▶ Terraform - <https://www.terraform.io/>



## CLOUDFORMATION, RDS AND ROUTE53

- ▶ We're going to start by creating two things:
  - ▶ A MySQL database on RDS
  - ▶ A HostedZone in Route53 (optional)
- ▶ To do this, we're going to use the CloudFormation template named `00-start.yaml` in the cloudformation folder.



# START.YAML

- ▶ Let's take a look at what's in `00-start.yaml` before we create our first CloudFormation stack.

```
start.yaml (~/Workspace/SaaS/cloudformation) - VIM
1 AWSTemplateFormatVersion: '2010-09-09'
2 Description: A beginning CloudFormation template to create a MySQL database and a Route53 HostedZone
3 Parameters:
4   DatabaseName:
5     Type: String
6     Description: The name of the database you want to create.
7     Default: devjam-workshop
8     AllowedPattern: '[a-zA-Z][a-zA-Z0-9]*'
9     MaxLength: '64'
10    MinLength: '1'
11    ConstraintDescription: must begin with a letter and contain only alphanumeric characters between 1 and 64 characters long.
12  DatabaseMasterUser:
13    Type: String
14    Description: The master username for your database.
15    Default: devjam-db-user
16    MinLength: '1'
17    MaxLength: '16'
18    AllowedPattern: '[a-zA-Z][a-zA-Z0-9]*'
19    ConstraintDescription: must begin with a letter and contain only alphanumeric characters between 1 and 16 characters long.
20  DatabaseMasterPassword:
21    Type: String
22    NoEcho: true
23    Description: The master password for your database.
24    MinLength: '1'
25    MaxLength: '41'
26    AllowedPattern: '[a-zA-Z0-9]+'
27    ConstraintDescription: must contain only alphanumeric characters.
28  DomainName:
29    Type: String
30    Description: A domain name to use for this DevJam workshop.
31
32 Conditions:
33  UseRoute53: !Not [ !Equals [ "", !Ref DomainName ] ]
```



# CREATE CLOUDFORMATION STACK

- ▶ Fire up your browser and login to the AWS console:
- ▶ <https://console.aws.amazon.com>
- ▶ On the main page, type in "CloudFormation" and click the popup

The screenshot shows the AWS Management Console homepage. The browser address bar displays the URL <https://eu-west-3.console.aws.amazon.com/console/home?region=eu-west-3>. The AWS logo is in the top left, and the user Julian is logged in at the top right. The main content area has a dark header with the text "AWS Management Console". Below the header, there's a sidebar titled "AWS services" with a "Find Services" search bar containing the text "cloudformation". A search result for "CloudFormation" is shown, with a brief description: "Create and Manage Resources with Templates". To the right of the main content area, there are two boxes: "Access resources on the go" (with a link to the AWS Console Mobile App) and "Explore AWS".



# CREATE CLOUDFORMATION STACK

The screenshot shows the AWS CloudFormation 'Create stack' wizard in progress. The current step is 'Step 1 Specify template'. The left sidebar lists steps 1 through 4: Step 1 (Specify template), Step 2 (Specify stack details), Step 3 (Configure stack options), and Step 4 (Review). The main area is titled 'Create stack' and contains two sections: 'Prerequisite - Prepare template' and 'Specify template'.

**Prerequisite - Prepare template**

Prepare template: Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Template is ready    Use a sample template    Create template in Designer

**Specify template**

A template is a JSON or YAML file that describes your stack's resources and properties.

**Template source**: Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL    Upload a template file

**Upload a template file**: Choose file  No file chosen  
JSON or YAML formatted file

S3 URL: Will be generated when template file is uploaded  

At the bottom right are 'Cancel' and 'Next' buttons. A blue arrow points from the 'Choose file' button in the 'Upload a template file' section to the 'Upload a template file' radio button in the 'Template source' section. A blue arrow also points from the 'Upload a template file' radio button down to the 'View in Designer' button.



# CREATE CLOUDFORMATION STACK

Screenshot of the AWS CloudFormation 'Create stack' wizard.

The URL in the browser is <https://eu-west-3.console.aws.amazon.com/cloudformation/home?region=eu-west-3#/stacks/create/parameters>.

The sidebar shows the steps:

- Step 1: Specify template
- Step 2: Specify stack details (current step)
- Step 3: Configure stack options
- Step 4: Review

**Specify stack details**

**Stack name**

**Stack name**: devjam-workshop

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

**Parameters**

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

**DatabaseMasterPassword**  
The master password for your database.  
.....

**DatabaseMasterUser**  
The master username for your database.  
devjam

**DatabaseName**  
The name of the database you want to create.  
devjamWorkshop

**DomainName**  
A domain name to use for this DevJam workshop.  
devjam.naydichev.dev

Buttons at the bottom: Cancel, Previous, Next



## CREATE CLOUDFORMATION STACK

- ▶ It will take some time for the CloudFormation stack to finish creating.
- ▶ For now, I'll show you the various tabs from the CloudFormation console and explain what they are used for.



# CREATE CLOUDFORMATION STACK

Screenshot of the AWS CloudFormation console showing the creation of a stack named "devjam-workshop".

The left sidebar shows the CloudFormation navigation menu with "Stack details" selected. The main content area displays the "Events" tab for the "devjam-workshop" stack.

**Events Table:**

Timestamp	Logical ID	Status	Status reason
2019-06-17 11:02:15 UTC+0200	devjam-workshop	CREATE_COMPLETE	-
2019-06-17 11:02:13 UTC+0200	Database	CREATE_COMPLETE	-
2019-06-17 10:49:31 UTC+0200	HostedZone	CREATE_COMPLETE	-
2019-06-17 10:48:31 UTC+0200	Database	CREATE_IN_PROGRESS	Resource creation Initiated
2019-06-17 10:48:29 UTC+0200	HostedZone	CREATE_IN_PROGRESS	Resource creation Initiated
2019-06-17 10:48:29 UTC+0200	Database	CREATE_IN_PROGRESS	-
2019-06-17 10:48:28 UTC+0200	HostedZone	CREATE_IN_PROGRESS	-
2019-06-17 10:48:24 UTC+0200	devjam-workshop	CREATE_IN_PROGRESS	User Initiated



# CREATE CLOUDFORMATION STACK

Screenshot of the AWS CloudFormation console showing a successful stack creation.

**CloudFormation - Stack devjan**

<https://eu-west-3.console.aws.amazon.com/cloudformation/home?region=eu-west-3#/stacks/resources?stackId=arn%3Aaws%3Acloudformation%3Aeu-west-3%3A175197272705%3Astack...>

**devjam-workshop**

Logical ID	Physical ID	Type	Status	Status reason
Database	devjamworkshop-devjam-db	AWS::RDS::DBInstance	CREATE_COMPLETE	-
HostedZone	Z2WHV3NENBFJMU	AWS::Route53::HostedZone	CREATE_COMPLETE	-



# CREATE CLOUDFORMATION STACK

The screenshot shows the AWS CloudFormation console interface. On the left, a sidebar menu includes 'CloudFormation' (selected), 'Stacks', 'Drifts', 'StackSets', 'Exports', 'Designer', 'Previous console', and 'Feedback'. The main area displays a 'Stacks (1)' section with a 'Create stack' button and a search bar. Below this is a list of stacks, with 'devjam-workshop' highlighted. The stack details show it was created on '2019-06-17 10:48:24 UTC+0200' and is in a 'CREATE\_COMPLETE' state. The main content area is titled 'devjam-workshop' and shows tabs for 'Stack info', 'Events', 'Resources', 'Outputs' (which is selected), 'Parameters', 'Template', and 'Change sets'. Under the 'Outputs' tab, there are two entries:

Key	Value	Description	Export name
DatabasePort	3306	The port of the database.	-
DatabaseUrl	devjamworkshop-devjam-db.c5qsfxxm8k9.eu-west-3.rds.amazonaws.com	The url of the database.	-



# CREATE CLOUDFORMATION STACK

Screenshot of the AWS CloudFormation console showing the creation of a stack named "devjam-workshop".

The left sidebar shows the CloudFormation service navigation with the "Stacks" section selected. A list of stacks is shown, with "devjam-workshop" highlighted.

The main content area displays the "Parameters" tab for the "devjam-workshop" stack. It lists four parameters:

Key	Value	Resolved value
DatabaseMasterPassword	****	-
DatabaseMasterUser	devjam	-
DatabaseName	devjamWorkshop	-
DomainName	devjam.naydichev.dev	-



# CREATE CLOUDFORMATION STACK

Screenshot of the AWS CloudFormation console showing the creation of a stack named "devjam-workshop".

The left sidebar shows the CloudFormation service navigation with the "Stack details" tab selected. The main area displays the "devjam-workshop" stack, which was created on 2019-06-17 at 10:48:24 UTC+0200 and is in a "CREATE\_COMPLETE" state.

The "Template" tab is active, showing the CloudFormation template code:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: A beginning CloudFormation template to create a MySQL database and a Route53 HostedZone
Parameters:
  DatabaseName:
    Type: String
    Description: The name of the database you want to create.
    Default: devjamWorkshop
    AllowedPattern: '[a-zA-Z][a-zA-Z0-9]*'
    MaxLength: '64'
    MinLength: '1'
    ConstraintDescription: must begin with a letter and contain only alphanumeric characters between 1 and 64 characters long.
  DatabaseMasterUser:
    Type: String
    Description: The master username for your database.
    Default: devjam
    MinLength: '1'
    MaxLength: '16'
    AllowedPattern: '[a-zA-Z][a-zA-Z0-9]*'
    ConstraintDescription: must begin with a letter and contain only alphanumeric characters between 1 and 16 characters long.
  DatabaseMasterPassword:
    Type: String
    NoEcho: true
    Description: The master password for your database.
    MinLength: '1'
    MaxLength: '41'
    AllowedPattern: '[a-zA-Z0-9]+'
    ConstraintDescription: must contain only alphanumeric characters.
  DomainName:
    Type: String
    Description: A domain name to use for this DevJam workshop.

Conditions:
  UseRoute53: !Not [ !Equals [ "", !Ref DomainName ] ]

Resources:
  HostedZone:
```



# NEW RESOURCES

- ▶ Once CloudFormation is done, we should have some new resources that we can view through the console.
- ▶ Let's take a look at them briefly.



# NEW RESOURCES - MYSQL DATABASE

Screenshot of the AWS RDS console showing the details of a MySQL database named "devjamworkshop-devjam-db".

**Summary**

DB identifier devjamworkshop-devjam-db	CPU 2.50%	Info Available	Class db.t2.micro
Role Instance	Current activity 0 Connections	Engine MySQL	Region & AZ eu-west-3b

**Connectivity & security**

Endpoint & port	Networking	Security
Endpoint devjamworkshop-devjam-db.c5qsjfxm8k9.eu-west-3.rds.amazonaws.com	Availability zone eu-west-3b	VPC security groups <a href="#">default (sg-4a176b21) (active)</a>
Port 3306	VPC <a href="#">vpc-18012071</a>	Public accessibility Yes
	Subnet group default	Certificate authority <a href="#">rds-ca-2015</a>
	Subnets <a href="#">subnet-456ac208</a> <a href="#">subnet-c9b9deb2</a> <a href="#">subnet-5c8fc335</a>	Certificate authority date Mar 5th, 2020



# NEW RESOURCES - ROUTE53 HOSTEDZONE

Screenshot of the AWS Route 53 Management Console showing a list of resource record sets for a hosted zone.

The left sidebar shows navigation links: Dashboard, Hosted zones (selected), Health checks, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, Pending requests, Resolver, VPCs, Inbound endpoints, Outbound endpoints, and Rules.

The main content area displays a table of resource record sets:

Name	Type	Value	Evaluate Target Health	Health Check ID	TTL	Region	Weight
devjam.naydichev.dev.	NS	ns-1607.awsdns-08.co.uk. ns-1247.awsdns-27.org. ns-627.awsdns-14.net. ns-288.awsdns-36.com.	-	-	172800		
devjam.naydichev.dev.	SOA	ns-1607.awsdns-08.co.uk. awsdns-hostmaster.amazon.com.	-	-	900		

Buttons at the top of the table include: Back to Hosted Zones, Create Record Set (highlighted in blue), Import Zone File, Delete Record Set, and Test Record Set.

A message on the right side of the table says: "To get started, click Create Record Set button or click an existing record set."



## SETUP

- ▶ Before we move any further, let's do some prep work for our Seinfeld service.
- ▶ We're going to briefly touch on the following services:
  - ▶ Route53 and Certificate Manager to create an SSL certificate.
  - ▶ RDS to prepare our database and load it with all of these salty pretzels.
  - ▶ I mean quotes.



## ROUTE53 SETUP

- ▶ In order for the HostedZones to work, we need to configure DNS on the parent domain.
- ▶ You'll need to go to the DNS provider for your domain and add the four NS records from the Route53 console.
- ▶ Not all DNS providers allow for subdomain NS records.
- ▶ If yours doesn't, then you'll need to manually create records when needed.



## DATABASE SETUP

- ▶ There is a lot of data to import, so before we get much further, I'd like to have everyone import the data to their database.
- ▶ You'll need access to the mysql command line tool.
  - ▶ If you don't have it, you can either install it or use a Docker container.
- ▶ The command you need to run is the following:
  - ▶ `zcat seinfeld.sql.gz | mysql -h $RDS_HOST -u $USER -p $DB_NAME`
  - ▶ This will take some time to import.



## CERTIFICATE MANAGER (AKA ACM)

- ▶ If you're not using a domain name, you can skip this step.
- ▶ Let's take a moment to setup a certificate. We'll need this later and it helps to get it out of the way now.
- ▶ When using a custom domain name with CloudFront, you must create the SSL certificate in us-east-1.
- ▶ Wait, CloudFront, I thought we were using API Gateway?
  - ▶ CloudFront is used by API Gateway when you create a custom domain.
  - ▶ CloudFront is a CDN service that allows you to specify multiple origins and caching behaviors.



# CERTIFICATE MANAGER (AKA ACM)

- ▶ We're going to use the 01-acm.yaml file to create the certificate for us.

```
acm.yaml (~/Workspace/SaaS/cloudformation) - VIM

1 AWSTemplateFormatVersion: '2010-09-09'
2 Description: A CloudFormation template to create a certificate.
3 Parameters:
4   DomainName:
5     Type: String
6     Description: A domain name to use for this DevJam workshop.
7
8 Resources:
9   Certificate:
10    Type: AWS::CertificateManager::Certificate
11    Properties:
12      DomainName: !Ref DomainName
13      ValidationMethod: DNS
14
15 Outputs:
16   CertificateArn:
17     Description: The ARN of the Certificate
18     Value: !Ref Certificate
19     Export:
20       Name: !Join [":", [ !Ref "AWS::StackName", "CertificateArn" ] ]
```



# CERTIFICATE MANAGER (AKA ACM)

- ▶ We can create this stack in the same fashion as we did the first one.
- ▶ This stack **must** be created in us-east-1.
- ▶ Since the certificate is using DNS validation, we will have to also create some DNS records.



# CERTIFICATE MANAGER (AKA ACM)

Screenshot of the AWS Certificate Manager console showing a pending validation certificate for devjam.naydichev.dev.

The screenshot shows the AWS Certificate Manager interface with the URL <https://console.aws.amazon.com/acm/home?region=us-east-1#/>. The certificate listed is:

Name	Domain name	Additional names	Status	Type	In use?	Renewal eligibility
devjam.naydichev.dev	devjam.naydichev.dev		Pending validation	Amazon Issued	No	Ineligible

**Status**

**Validation not complete**  
The status of this certificate request is "Pending validation". Further action is needed to validate and approve the certificate. [Learn more](#).

Domain	Validation status
devjam.naydichev.dev	Pending validation

Add the following CNAME record to the DNS configuration for your domain. The procedure for adding CNAME records depends on your DNS service Provider. [Learn more](#).

Name	Type	Value
_5332785ec7a301cd4bed647656d47f51.devjam.naydichev.dev.	CNAME	_959fd50ddb1de0eca35c80dbf39c64ae.ltfvzjuylp.acm-validations.aws.

**Note:** Changing the DNS configuration allows ACM to issue certificates for this domain name for as long as the DNS record exists. You can revoke permission at any time by removing the record. [Learn more](#).

**Create record in Route 53**    Amazon Route 53 DNS Customers ACM can update your DNS configuration for you. [Learn more](#).

**Export DNS configuration to a file**    You can export all of the CNAME records to a file

**Details**

Type	Amazon Issued	Requested at	2019-06-17T10:50:02UTC
In use?	No	Public key info	RSA 2048-bit
Domain name	devjam.naydichev.dev	Signature algorithm	SHA256WITHRSA
Number of additional names	0	ARN	arn:aws:acm:us-east-1:175197272705:certificate/175dcb87-4216-471c-9a25-d0c06b045549
Identifier	175dcb87-4216-471c-9a25-d0c06b045549	Validation state	Pending
Serial number	N/A		



# CERTIFICATE MANAGER (AKA ACM)

The screenshot shows the AWS Route 53 Management Console interface. The left sidebar contains navigation links for various services like Dashboard, Hosted zones, and Domains. The main content area displays a table of DNS record sets. The table has columns for Name, Type, Value, and other details. Three records are listed:

Name	Type	Value
devjam.naydichev.dev.	NS	ns-1607.awsdns-08.co.uk. ns-1247.awsdns-27.org. ns-627.awsdns-14.net. ns-288.awsdns-36.com.
devjam.naydichev.dev.	SOA	ns-1607.awsdns-08.co.uk. awsdns-hostmaster.amazon.com.
_5332785ec7a301cd4bed647656d47f51.devjam.naydichev.dev.	CNAME	_959fd50ddb1de0eca35c80dbf39c64ae.ltvzjuylp.ac

A message on the right side of the screen says: "To get started, click Create Record Set button or click an existing record set."



# CERTIFICATE MANAGER (AKA ACM)

Screenshot of the AWS Certificate Manager console showing a certificate for devjam.naydichev.dev.

**Certificates**

AWS Certificate Manager logs domain names from your certificates into public certificate transparency (CT) logs when renewing certificates. You can opt out of CT logging. [Learn more](#)

**Actions**

**Status**

**Details**

Type	Amazon Issued	Requested at	2019-06-17T10:50:02UTC
In use?	No	Issued at	2019-06-17T10:57:16UTC
Domain name	devjam.naydichev.dev	Not before	2019-06-17T00:00:00UTC
Number of additional names	0	Not after	2020-07-17T12:00:00UTC
Identifier	175dcb87-4216-471c-9a25-d0c06b045549	Public key info	RSA 2048-bit
Serial number	06:34:49:e0:7f:d2:5d:6f:44:10:7c:4e:aa:46:5b:1b	Signature algorithm	SHA256WITHRSA
		ARN	arn:aws:acm:us-east-1:1751972705:certificate/175dcb87-4216-471c-9a25-d0c06b045549



# CERTIFICATE MANAGER (AKA ACM)

Screenshot of the AWS CloudFormation console showing the Outputs tab for the stack "devjam-workshop-certificate".

The Outputs section displays one output:

Key	Value	Description	Export name
CertificateArn	arn:aws:acm:us-east-1:175197272705:certificate/175dc87-4216-471c-9a25-d0c06b045549	The ARN of the Certificate	devjam-workshop-certificate:CertificateArn



## WHAT IS AWS LAMBDA?

- ▶ AWS Lambda is the hot new craze that allows you to run code without needing any infrastructure (mostly).
- ▶ There are several runtimes available including Java, Ruby, Javascript and Python, just to name a few.
- ▶ AWS Lambda integrates with a number of other services which allows you to easily respond to various events.
- ▶ Some examples include SNS topics, DynamoDB Streams, S3 Events, and many many more.



# AWS LAMBDA

- ▶ Let's take a few minutes to play with AWS Lambda.
- ▶ Fire up your AWS Console, and navigate to the Lambda control plane, and click on create function.

The screenshot shows the AWS Lambda Management Console in a web browser. The URL in the address bar is <https://eu-west-3.console.aws.amazon.com/lambda/home?region=eu-west-3#/create>. The page title is "Create function". The top navigation bar includes links for Services, Resource Groups, and a user profile for "julian @ 1751-9727-2705".

The main content area displays three options for creating a new Lambda function:

- Author from scratch**: Start with a simple Hello World example. This option is currently selected, indicated by a blue border around its card.
- Use a blueprint**: Build a Lambda application from sample code and configuration presets for common use cases.
- Browse serverless app repository**: Deploy a sample Lambda application from the AWS Serverless Application Repository.

Below these options, there is a section titled "Basic information" with a "Function name" field containing the placeholder "myFunctionName".



# AWS LAMBDA

Lambda Management Console <https://eu-west-3.console.aws.amazon.com/lambda/home?region=eu-west-3#/create?f0=a3c%3D%3AaGVsbG8td29ybGQ%3D&tab=blueprints>

Services Resource Groups Julian @ 1751-9727-2705 Paris Support

Lambda > Functions > Create function

## Create function Info

Choose one of the following options to create your function.

- Author from scratch**  
Start with a simple Hello World example.
- Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.
- Browse serverless app repository**  
Deploy a sample Lambda application from the AWS Serverless Application Repository.

**Blueprints Info** Export

Add filter < 1 >

Keyword : hello-world X

<b>greengrass-hello-world</b> Deploy this lambda to a Greengrass core where it will send a hello world message to a topic python · greengrass · iot · hello world	<b>hello-world</b> A starter AWS Lambda function. nodejs	<b>hello-world-python</b> A starter AWS Lambda function. python3.7	<b>greengrass-hello-world-counter-python</b> Deploy this lambda to a Greengrass core where it will count how many times the function has been invoked python · greengrass · iot · hello world
---	--	--	---

Cancel Configure



# AWS LAMBDA

Lambda Management Console <https://eu-west-3.console.aws.amazon.com/lambda/home?region=eu-west-3#/create/new?bp=hello-world-python>

Services Resource Groups

Lambda Functions Create function Using blueprint hello-world-python

**Basic information** Info

Function name: hello-world

Execution role: Create a new role with basic Lambda permissions

**Info** Role creation might take a few minutes. The new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.

Lambda will create an execution role named hello-world-role-834a4fti, with permission to upload logs to Amazon CloudWatch Logs.

**Lambda function code**

Code is preconfigured by the chosen blueprint. You can configure it after you create the function.

Runtime: Python 3.7

```
1 import json
2
3 print('Loading function')
4
5
6 def lambda_handler(event, context):
7     #print("Received event: " + json.dumps(event, indent=2))
8     print("value1 = " + event['key1'])
9     print("value2 = " + event['key2'])
10    print("value3 = " + event['key3'])
```



# AWS LAMBDA

Lambda Management Console < https://eu-west-3.console.aws.amazon.com/lambda/home?region=eu-west-3#/functions/hello-world?newFunction=true&tab=graph Incognito

aws Services Resource Groups Julian @ 1751-9727-2705 Paris Support

Congratulations! Your Lambda function "hello-world" has been successfully created. You can now change its code and configuration. Choose Test to input a test event when you want to test your function.

ARN - arn:aws:lambda:eu-west-3:175197272705:function:hello-world

hello-world Throttle Qualifiers Actions Select a test event Test Save

Configuration Monitoring

Designer

Add triggers Choose a trigger from the list below to add it to your function.

- API Gateway
- Application Load Balancer
- CloudWatch Events
- CloudWatch Logs
- CodeCommit
- DynamoDB

hello-world Layers (0)

Add triggers from the list on the left

Amazon CloudWatch Logs

Resources that the function's role has access to appear here

Function code Info

Code entry type Edit code inline

Runtime Python 3.7

Handler Info lambda\_function.lambda\_handler



# AWS LAMBDA

The screenshot shows the AWS Lambda Management Console interface. The top navigation bar includes the AWS logo, Services dropdown, Resource Groups dropdown, and user information (julian @ 1751-9727-2705, Paris, Support). The main title is "hello-world". The "Code entry type" is set to "Edit code inline", "Runtime" is "Python 3.7", and the "Handler" is "lambda\_function.lambda\_handler". The code editor displays the following Python code:

```
import json

print('Loading function')

def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))
    print("value1 = " + event['key1'])
    print("value2 = " + event['key2'])
    print("value3 = " + event['key3'])
    return event['key1'] # Echo back the first key value
    #raise Exception('Something went wrong')
```

The bottom right corner of the code editor shows "1:1 Python Spaces: 4" and a gear icon.



# AWS LAMBDA

Lambda Management Console <https://eu-west-3.console.aws.amazon.com/lambda/home?region=eu-west-3#/functions/hello-world?tab=graph>

Services [Resource Groups](#) Julian @ 1751-9727-2705 Paris Support

## hello-world

Throttle Qualifiers Actions DevJamHelloWorld Test Save

### Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

Key Value Remove

▶ Encryption configuration

### Tags

You can use tags to group and filter your functions. A tag consists of a case-sensitive key-value pair. [Learn more](#)

lambda-console:blueprint hello-world-python Remove

Key Value Remove

### Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Use an existing role

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/hello-world-role-834a4fti

View the [hello-world-role-834a4fti](#) role on the IAM console.

### Basic settings

Description

A starter AWS Lambda function.

Memory (MB) [Info](#)

Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout [Info](#)

0 min 3 sec



# AWS LAMBDA

Lambda Management Console <https://eu-west-3.console.aws.amazon.com/lambda/home?region=eu-west-3#/functions/hello-world?newFunction=true&tab=graph>

Services Resource Groups Julian @ 1751-9727-2705 Paris Support

Congratulations! Your Lambda function "hello-world" has been created.

hello-world

Configuration Monitoring

Add triggers API Gateway Application Load Balancer CloudWatch Events CloudWatch Logs CodeCommit DynamoDB

Add triggers

Choose a trigger from the list below to add it to your function.

API Gateway Application Load Balancer CloudWatch Events CloudWatch Logs CodeCommit DynamoDB

Add triggers

Function code Info

Code entry type Edit code inline

Configure test event

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

Create new test event  Edit saved test events

Event template Hello World

Event name DevJamHelloWorld

```
1 {  
2   "key1": "Hello",  
3   "key2": "DevJam",  
4   "key3": "From Lambda"  
5 }
```

Select a test event Test Save

Logs

has access to appear here

lambda\_handler



# AWS LAMBDA

Lambda Management Console < https://eu-west-3.console.aws.amazon.com/lambda/home?region=eu-west-3#/functions/hello-world?newFunction=true&tab=graph Incognito

aws Services Resource Groups Julian @ 1751-9727-2705 Paris Support

Congratulations! Your Lambda function "hello-world" has been successfully created. You can now change its code and configuration. Choose Test to input a test event when you want to test your function.

Lambda > Functions > hello-world ARN - arn:aws:lambda:eu-west-3:175197272705:function:hello-world

hello-world Throttle Qualifiers Actions DevJamHelloWorld Test Save

Execution result: succeeded (logs)

Details

The section below shows the result returned by your function execution.

```
"Hello"
```

Summary

Code SHA-256	Request ID
y/S7LbY1skLhDA7qezbDa1zvPnoGmoae3hmZMHOTVnE=	57d2bea8-92b0-4f35-8bc8-ba0b7364f86c
Duration	Billed duration
1.21 ms	100 ms
Resources configured	Max memory used
128 MB	53 MB

Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: 57d2bea8-92b0-4f35-8bc8-ba0b7364f86c Version: $LATEST
value1 = Hello
value2 = DevJam
value3 = From Lambda
END RequestId: 57d2bea8-92b0-4f35-8bc8-ba0b7364f86c
REPORT RequestId: 57d2bea8-92b0-4f35-8bc8-ba0b7364f86c Duration: 1.21 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 53 MB
```

Configuration Monitoring



## AWS LAMBDA

- ▶ You can also create and update functions via CloudFormation (more on that in a minute) and via the AWS CLI.
- ▶ Check out the help guide in the CLI to learn how to use it:
  - ▶ `aws lambda help`
  - ▶ `aws lambda create-function help`
  - ▶ `aws lambda update-function-code help`



## AWS LAMBDA HANDLER

- ▶ The last, but most important bit, is the Lambda handler.
- ▶ When you configure a function, you have to tell Lambda what part of your code to execute.
- ▶ Here are a few examples:
  - ▶ Python: `lambda_function.lambda_handler`
    - ▶ A file '`lambda_function.py`' with a method named '`lambda_handler`'
  - ▶ Java: `dev.naydichev.devjam.saas.LambdaHandler::handleRequest`
    - ▶ A class named '`LambdaHandler`' with a method named '`handleRequest`'
  - ▶ JavaScript: `functions/export.handler`
    - ▶ A file in the '`functions`' directory named '`export.js`' that contains a module export called '`handler`'



# AWS LAMBDA HANDLER

- ▶ The input to these defined methods will depend on the language you choose and the type of event you're responding to.
- ▶ In this case, all of the events are from API Gateway, so I can give you some guidance.
- ▶ **Python:** `def handler(event, context)`
- ▶ **Java:** `handleRequest(InputStream inputStream, OutputStream outputStream, Context context)`
- ▶ **JavaScript:** `const handleRequest = (event, context, callback) => { . . . }`
- ▶ You can see a sample full request here:
  - ▶ <https://docs.aws.amazon.com/lambda/latest/dg/with-on-demand-https.html>



# AWS LAMBDA DEPENDENCIES

- ▶ So, let's say you write a function, but you're relying on some third party code (outside of the AWS libraries, which are available by default).
- ▶ You'll need to bundle your dependencies with your function code.
- ▶ Most of the time you just need to include the appropriate directories or files in the root of a zip file.
- ▶ You can check the packaging requirements for your language of choice in the documentation:
  - ▶ <https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>
  - ▶ See "Working with \${YOUR\_LANGUAGE}" and then check out the "Deployment Package" section.



## SAM - SERVERLESS APPLICATION MODEL

- ▶ SAM (or Serverless Application Model) is essentially a CloudFormation template with some special tools and utilities to help you write and use Lambda functions in conjunction with API Gateway or other event sources.
- ▶ The templates are *transformed* by the power of grayskull!
  - ▶ More likely, it's just some CloudFormation magic, utilizing *Change Sets* (which we saw on the stack page).
- ▶ The nice thing about SAM is that there is a command line utility that also uploads your function code (with its dependencies) to S3 and places a reference directly into the template.
- ▶ Using SAM is outside the scope of this workshop, but I encourage you to check it out along with the other tools mentioned at the beginning.
  - ▶ Instead, we'll be dealing with the nitty-gritty work ourselves.
- ▶ Let's get our hands dirty!



## INPUT FROM API GATEWAY

- ▶ Before we begin writing code, we need to talk about the input to Lambda from API Gateway.
- ▶ When looking at a function, you can choose to create a test event from the template "Amazon API Gateway AWS Proxy"
  - ▶ It contains all of the juicy details you'd expect from an HTTP request.
  - ▶ The part that we're primarily concerned with is the path.
  - ▶ It's a top level element in the event object.



# INPUT FROM API GATEWAY

Lambda Management Console <https://eu-west-3.console.aws.amazon.com/lambda/home?region=eu-west-3#/functions/saas-function?tab=graph>

Services Resource Groups

Lambda > Functions > saas-function

### saas-function

Execution result: succeeded (logs)

Details

The area below shows the result returned by your function.

```
{  
  "statusCode": 200,  
  "body": "{\n    \"speaker\": \"ELAINE\"\n  }"}  
}
```

Summary

Code SHA-256  
weFacyxJxExnSRHQXZMULFqmVJa/QeIHs6bOk

Duration  
221.79 ms

Resources configured  
512 MB

Log output

The section below shows the logging calls in your function.

```
START RequestId: 1a68bdfd-7187-49b1  
Connecting to 'jdbc:mysql://devjam:  
quotes=false'][response={  
  "speaker": "ELAINE",  
  "utterance": "Let's face it, you  
}]END RequestId: 1a68bdfd-7187-49b1  
REPORT RequestId: 1a68bdfd-7187-49b1
```

Configure test event

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

Create new test event  
 Edit saved test events

Event template  
Amazon API Gateway AWS Proxy

Event name  
MyEventName

```
1 [ {  
2   "body": "eyJ0ZXN0IjoiYm9keSJ9",  
3   "resource": "/{proxy+}",  
4   "path": "/path/to/resource",  
5   "httpMethod": "POST",  
6   "isBase64Encoded": true,  
7   "queryStringParameters": {  
8     "foo": "bar"  
9   },  
10  "pathParameters": {  
11    "proxy": "/path/to/resource"  
12  },  
13  "stageVariables": {  
14    "baz": "qux"  
15  },  
16  "headers": {  
17    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",  
18    "Accept-Encoding": "gzip, deflate, sdch",  
19    "Accept-Language": "en-US,en;q=0.8",  
20    "Cache-Control": "max-age=0",  
21    "CloudFront-Forwarded-Proto": "https",  
22    "CloudFront-Is-Desktop-Viewer": "true",  
23    "CloudFront-Is-Mobile-Viewer": "false",  
24    "CloudFront-Is-SmartTV-Viewer": "false",  
25    "CloudFront-Is-Tablet-Viewer": "false",  
26  }  
}
```

Root Test Save

to the ear guy. \\"\n}"

view the CloudWatch log group.

vjam'[season=null, episode=null,

Max Memory Used: 119 MB



# API OVERVIEW

## Route

## Result

/

A random quote from the entire show.

/quotes

A random quote from the entire show.

/seasons

A list of seasons.

/seasons/quotes

A random quote from the entire show.

/seasons/{season}/

A random quote from a particular seasons.

/seasons/{season}/quotes

A list of episodes.

/seasons/{season}/episodes/

A random quote from a particular seasons.

/seasons/{season}/episodes/{episode}

The entire text of a specific episode.

/seasons/{season}/episodes/{episode}/quotes

A random quote from a specific episode.



## READY.... SET... GO!

- ▶ Feel free to look at the code sample included in the repo (which is written in Java) as inspiration.
- ▶ Go ahead and start writing your Lambda function.
  - ▶ You can assume that all of your database information will be passed as environment variables.
  - ▶ You can use the database information from the RDS page.
- ▶ To test, you should create an entry point that sends predefined requests.
- ▶ Let me know if you need help!



# READY.... SET... GO!

- ▶ Once you have something that you think will work, then we need to create your Lambda function.
- ▶ Create a zip file following the packaging rules for your language of choice.
- ▶ If you're using Java, a jar file will work, just make sure that all your dependencies are included in that jar.



## CREATING THE LAMBDA FUNCTION

- ▶ Once you have a deployable Lambda function, you can create a sample function via the AWS Console.
- ▶ Make sure everything works before proceeding.
- ▶ Next, we'll create an S3 bucket to use for storing our Lambda function, and then deploy it with CloudFormation.
- ▶ Use `02-deployment-bucket.yaml` and update your CloudFormation stack.



# CREATING THE LAMBDA FUNCTION

```
 02-deployment-bucket.yaml (~/Workspace/SaaS/cloudformation) - VIM1

35 Resources:
36   HostedZone:
37     Condition: UseRoute53
38     Type: AWS::Route53::HostedZone
39     Properties:
40       Name: !Ref DomainName
41   DatabaseSecurityGroup:
42     Type: AWS::EC2::SecurityGroup
43     Properties:
44       GroupDescription: "An all access pass to data"
45       SecurityGroupIngress:
46         - CidrIp: 0.0.0.0/0
47           FromPort: 3306
48           ToPort: 3306
49           IpProtocol: tcp
50   Database:
51     Type: AWS::RDS::DBInstance
52     Properties:
53       AllocatedStorage: "5"
54       DBInstanceClass: db.t2.micro
55       DBInstanceIdentifier: !Sub
56         - "${DBName}-devjam-db"
57         - { DBName: !Ref DatabaseName }
58       DatabaseName: !Ref DatabaseName
59       Engine: MySQL
60       MasterUserPassword: !Ref DatabaseMasterPassword
61       MasterUsername: !Ref DatabaseMasterUser
62       MultiAZ: true
63       VPCSecurityGroups:
64         - !GetAtt [ DatabaseSecurityGroup, GroupId ]
65
66 DeploymentBucket:
67   Type: AWS::S3::Bucket
68
69 Outputs:
70   DatabaseUrl:
71     Description: The url of the database.
72     Value: !GetAtt Database.Endpoint.Address
73   DatabasePort:
74     Description: The port of the database.
75     Value: !GetAtt Database.Endpoint.Port
```



## CREATING THE LAMBDA FUNCTION

- ▶ Once we have an S3 Bucket, we can upload our artifact to S3.
- ▶ Now's a good time to configure the AWS CLI (if you haven't already).
- ▶ Using the AccessKeyId and SecretAccessKey that you got earlier when creating your own user, you can execute the following to setup the CLI:
  - ▶ aws configure
  - ▶ It will prompt you for relevant values.
  - ▶ If you already have this configured, instead of overwriting your default profile, you can use a custom profile:
    - ▶ aws configure --profile devjam-workshop
    - ▶ You'll need to add `--profile devjam-workshop` to your commands in the following slides.



# CREATING THE LAMBDA FUNCTION

- ▶ Let's upload the Artifact to S3.
  - ▶ `aws s3 upload /path/to/artifact s3://$BUCKET_NAME/artifact.zip`
  - ▶ Replace `$BUCKET_NAME` with the name from the CloudFormation resources tab.
  - ▶ You're welcome to alter the destination path to be anywhere in the S3 Bucket, just make note of the Key.



## CREATING THE LAMBDA FUNCTION

- ▶ Now that our artifact is in S3, we can deploy the function via CloudFormation.
- ▶ Let's take a look at `03-add-lambda-function.yaml` on the next slide.



# CREATING THE LAMBDA FUNCTION

```
86 DeploymentBucket:
87   Type: AWS::S3::Bucket
88
89 LambdaExecutionRole:
90   Type: AWS::IAM::Role
91   Properties:
92     AssumeRolePolicyDocument:
93       Version: '2012-10-17'
94       Statement:
95         - Effect: Allow
96           Principal:
97             Service:
98               - lambda.amazonaws.com
99             Action:
100               - sts:AssumeRole
101             Path: "/"
102             Policies:
103               - PolicyName: root
104                 PolicyDocument:
105                   Version: '2012-10-17'
106                   Statement:
107                     - Effect: Allow
108                       Action:
109                         - logs:*
110                         Resource: arn:aws:logs:***:**
111
112 LambdaFunction:
113   Type: AWS::Lambda::Function
114   Properties:
115     Code:
116       S3Bucket: !Ref DeploymentBucket
117       S3Key: !Ref ArtifactKey
118     Handler: !Ref LambdaHandler
119     Runtime: !Ref LambdaRuntime
120     Role: !GetAtt LambdaExecutionRole.Arn
121     Timeout: 75
122     Memory: 256
123     Environment:
124       Variables:
125         DB_USER: !Ref DatabaseMasterUser
126         DB_PASS: !Ref DatabaseMasterPassword
127         JDBC_URL: !Sub "jdbc:mysql://${Database.Endpoint.Address}:3306/${DatabaseName}"
```



# HOME STRETCH: API GATEWAY

- ▶ Let's take a brief moment to review what we've accomplished so far.
- ▶ We've created and populated an RDS database.
- ▶ We've created and updated a CloudFormation stack.
- ▶ We've created a Lambda function.
- ▶ We've created an SSL certificate.
- ▶ We've created a Route53 zone.



## HOME STRETCH: API GATEWAY

- ▶ What's left?
  - ▶ API Gateway!
- ▶ We need to setup API Gateway to process requests and direct them to our Lambda function.
- ▶ Rather than doing this by hand, and then again via CloudFormation, let's just dive right into the CloudFormation yaml.



# HOME STRETCH: API GATEWAY

```
04-add-api-gateway.yaml (~/Workspace/SaaS/cloudformation) - VIM
+ 04-add-api-gateway.yaml 03-add-lambda-function.yaml +
```

```
129
130 APIGateway:
131   Type: AWS::ApiGateway::RestApi
132   Properties:
133     Name: Seinfeld as a Service
134     Description: A service about nothing.
135
136 APIGatewayWildcardResource:
137   Type: AWS::ApiGateway::Resource
138   Properties:
139     RestApiId: !Ref APIGateway
140     ParentId: !GetAtt APIGateway.RootResourceId
141     PathPart: "{proxy+}"
142
143 APIGatewayRootMethod:
144   Type: AWS::ApiGateway::Method
145   Properties:
146     AuthorizationType: NONE
147     HttpMethod: GET
148     Integration:
149       IntegrationHttpMethod: POST
150       Type: AWS_PROXY
151       Uri: !Sub "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${LambdaFunction.Arn}/invocations"
152     ResourceId: !GetAtt APIGateway.RootResourceId
153     RestApiId: !Ref APIGateway
154
155 APIGatewayWildcardMethod:
156   Type: AWS::ApiGateway::Method
157   Properties:
158     AuthorizationType: NONE
159     HttpMethod: GET
160     Integration:
161       IntegrationHttpMethod: GET
162       Type: AWS_PROXY
163       Uri: !Sub "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${LambdaFunction.Arn}/invocations"
164     ResourceId: !Ref APIGatewayWildcardResource
165     RestApiId: !Ref APIGateway
166
# NOTE: change the name of this resource everytime you want to deploy a change to API Gateway.
167 APIGatewayDeployment:
168   DependsOn:
169     - APIGatewayRootMethod
```



## HOME STRETCH: API GATEWAY

- ▶ If you look on the outputs tab of CloudFormation, you'll see the API Gateway URL.
- ▶ You can click on this URL, and try the various routes that you've configured.



## BONUS: CONNECTING YOUR DOMAIN NAME

- ▶ This last step will take a few moments, so let's go ahead and update the stack one last time.
- ▶ Use file `05-add-custom-domain.yaml` to update your stack.
- ▶ You'll need the Certificate Arn of the Certificate Manager certificate we created earlier. (\*cough\* certificate \*cough\*)



# BONUS: CONNECTING YOUR DOMAIN NAME

```
 05-add-custom-domain.yaml (~/Workspace/SaaS/cloudformation) - VIM
+ 05-add-custom-domain.yaml 03-add-lambda-function.yaml +
```

```
185  FunctionName: !GetAtt LambdaFunction.Arn
186  Principal: apigateway.amazonaws.com
187  SourceArn: !Sub "arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:${APIGateway}/*/GET/"
188
189 APIGatewayDomain:
190   Condition: UseRoute53
191   Type: AWS::ApiGateway::DomainName
192   Properties:
193     CertificateArn: !Ref CertificateArn
194     DomainName: !Ref DomainName
195
196 APIGatewayBasePathMapping:
197   Condition: UseRoute53
198   Type: AWS::ApiGateway::BasePathMapping
199   Properties:
200     basePath: "/"
201     DomainName: !Ref APIGatewayDomain
202     RestApiId: !Ref APIGateway
203
204 CustomZone:
205   Condition: UseRoute53
206   Type: AWS::Route53::RecordSet
207   Properties:
208     AliasTarget:
209       DNSName: !GetAtt APIGatewayDomain.DistributionDomainName
210       HostedZoneId: Z2FDTNDATAQYW2 # fixed value for CloudFront
211       HostedZoneId: !Ref HostedZone
212       Name: !Sub "${DomainName}."
213       Type: CNAME
214
215 Outputs:
216   DatabaseUrl:
217     Description: The url of the database.
218     Value: !GetAtt Database.Endpoint.Address
219   DatabasePort:
220     Description: The port of the database.
221     Value: !GetAtt Database.Endpoint.Port
222   APIGatewayURL:
223     Description: The API Gateway URL.
224     Value: !Sub "https://${APIGateway}.execute-api.${AWS::Region}.amazonaws.com/kramer/"
```



## API GATEWAY : CONSOLE

- ▶ While we wait for the custom domain name change to propagate, let's talk a little more about API Gateway.
- ▶ API Gateway has a lot of options for configuration.
- ▶ It might be nice to walk you through the AWS Console, and discuss what the various options are and what they do.



## API GATEWAY : CONSOLE

- ▶ This slide serves as a reminder to the presenter not to rely solely on slides.
- ▶ If he's fast enough, you won't even see this.
- ▶ If he's not, then please take this opportunity to heckle him while he finds and loads his web browser.



## WRAP UP

- ▶ That's it! You've created a Lambda function and configured API Gateway to serve traffic based on its responses.
- ▶ Where to go from here?
  - ▶ I'd recommend looking into some of the available deployment tools to help make this process easier.
    - ▶ Serverless or SAM being the top choices among the plethora of options.
    - ▶ You can look into creating an API Gateway site using multiple Lambda functions, or even configuring Authentication/Authorization with Cognito.
    - ▶ Lambda has many, many, many options for use, including configuring it to respond to objects in S3, DynamoDB streams, SNS, SQS, CloudWatch Events, SES emails. The list goes on.



## WRAP UP

- ▶ Finally, a reminder that these slides are available on my Github:
  - ▶ <https://github.com/naydichev/SaaS>
- ▶ If you have any questions, comments or feedback, please let me know.
  - ▶ You can chat to me after I'm done yammering, or send me an email:
    - ▶ [julian.naydichev@sytac.io](mailto:julian.naydichev@sytac.io)