SEINFELD AS A SERVICE

A SERVICE ABOUT NOTHING

Presented by Julian Naydichev

ABOUT ME

- Julian Naydichev
  - Sytac Employee
    - 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

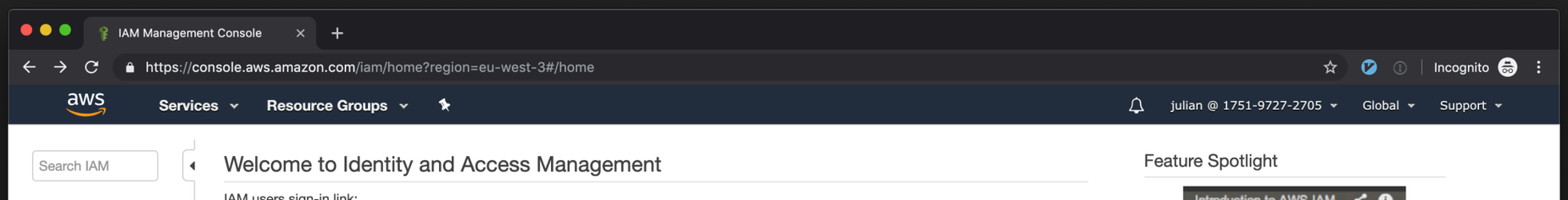
- You will need to sign up or have access to an AWS account.
  - Sign-up here: https://portal.aws.amazon.com/billing/signup#/start
- All of the resources used within this presentation should fit within the free tier.
  - You may still need to enter in a credit card, but as long as you close the account 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

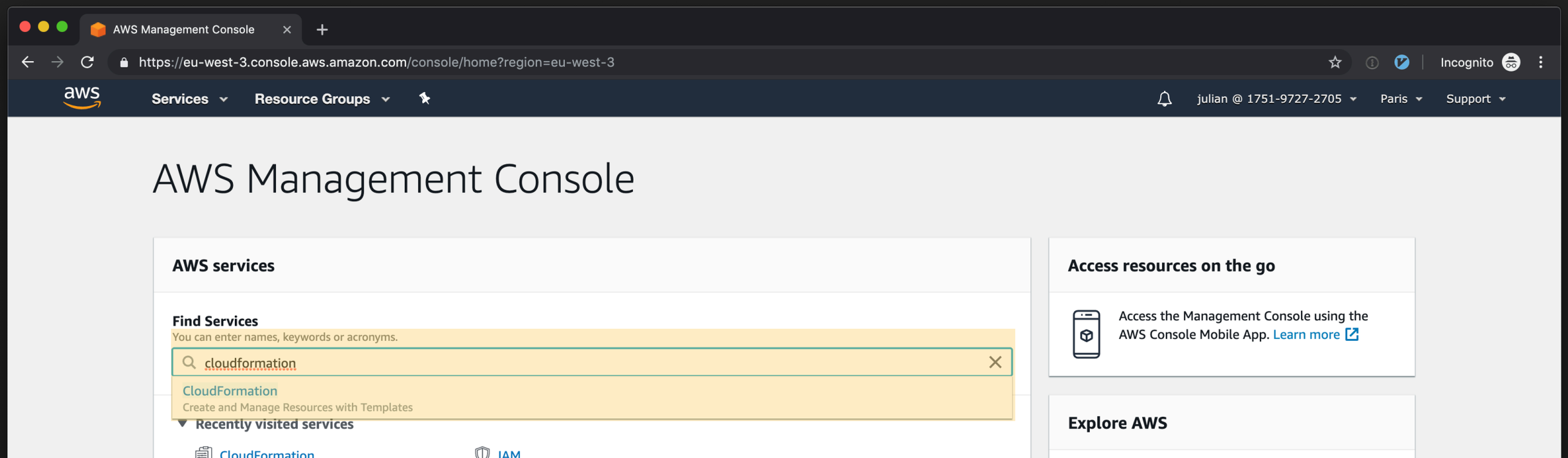
- Let's take a look at what's in `00-start.yaml` before we create our first CloudFormation stack.

start.yaml (~/Workspace/SaaS/cloudformation) - VIM

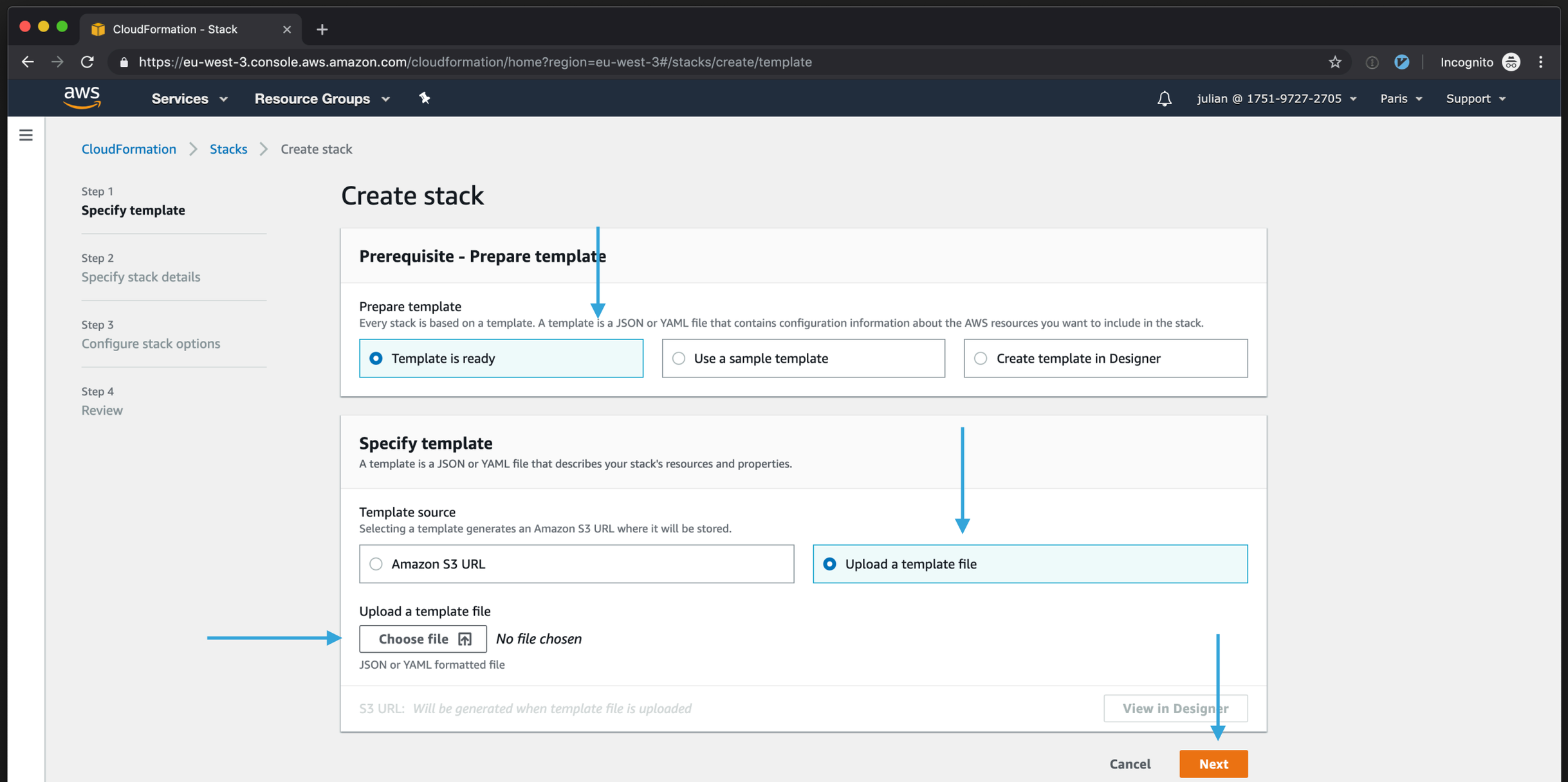
```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: A beginning CloudFormation template to create a MySQL database and a Route53 HostedZone
Parameters:
  DatabaseName:
    Type: String
    Description: The name of the database you want to create.
    Default: devjam-workshop
    AllowedPattern: '[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-db-user
    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 ] ]
```

CREATE CLOUDFORMATION STACK

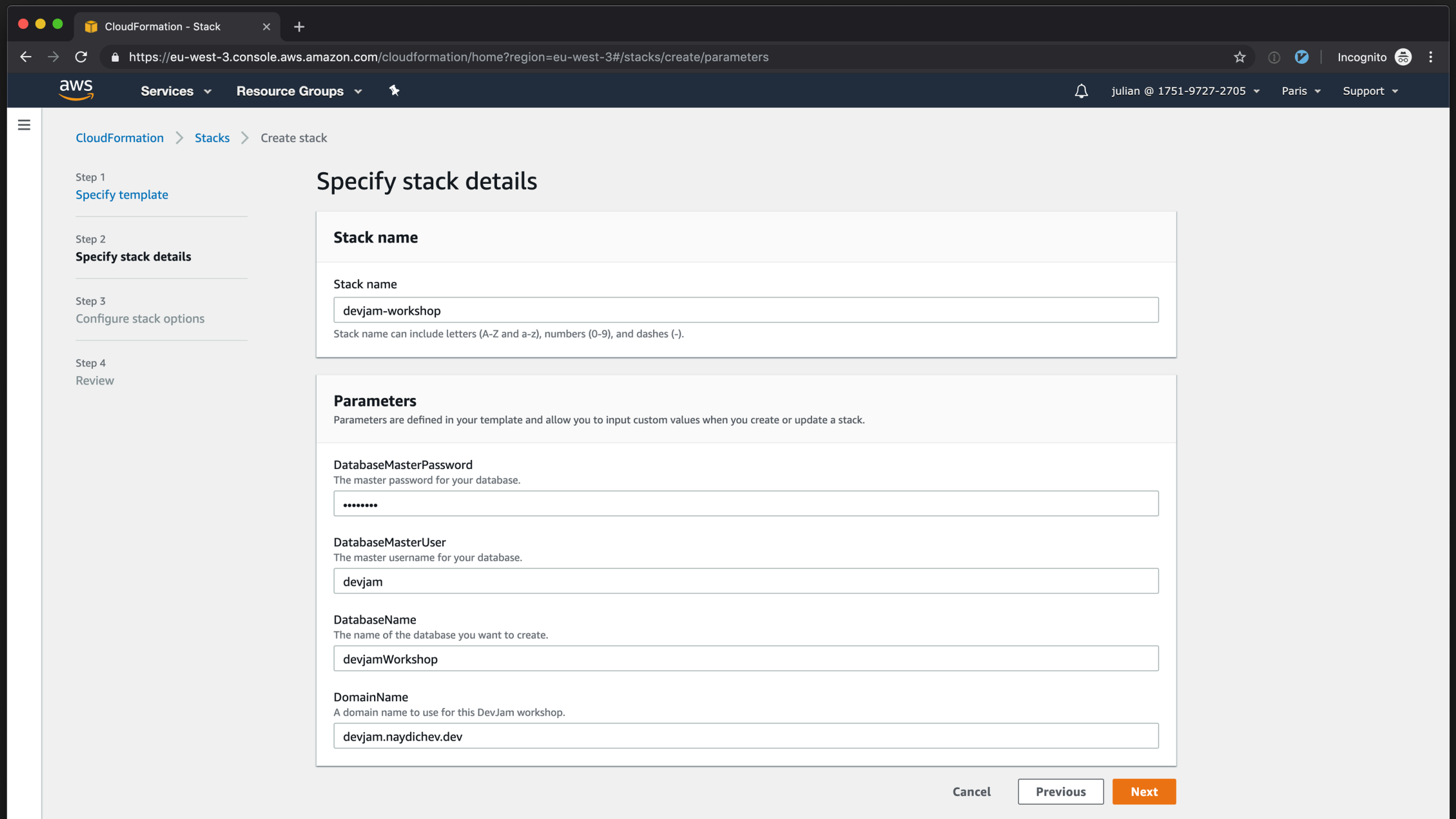
- Fire up your browser and login to the AWS console:
  - https://console.aws.amazon.com
- On the main page, type in "`CloudFormation`" and click the popup



CREATE CLOUDFORMATION STACK



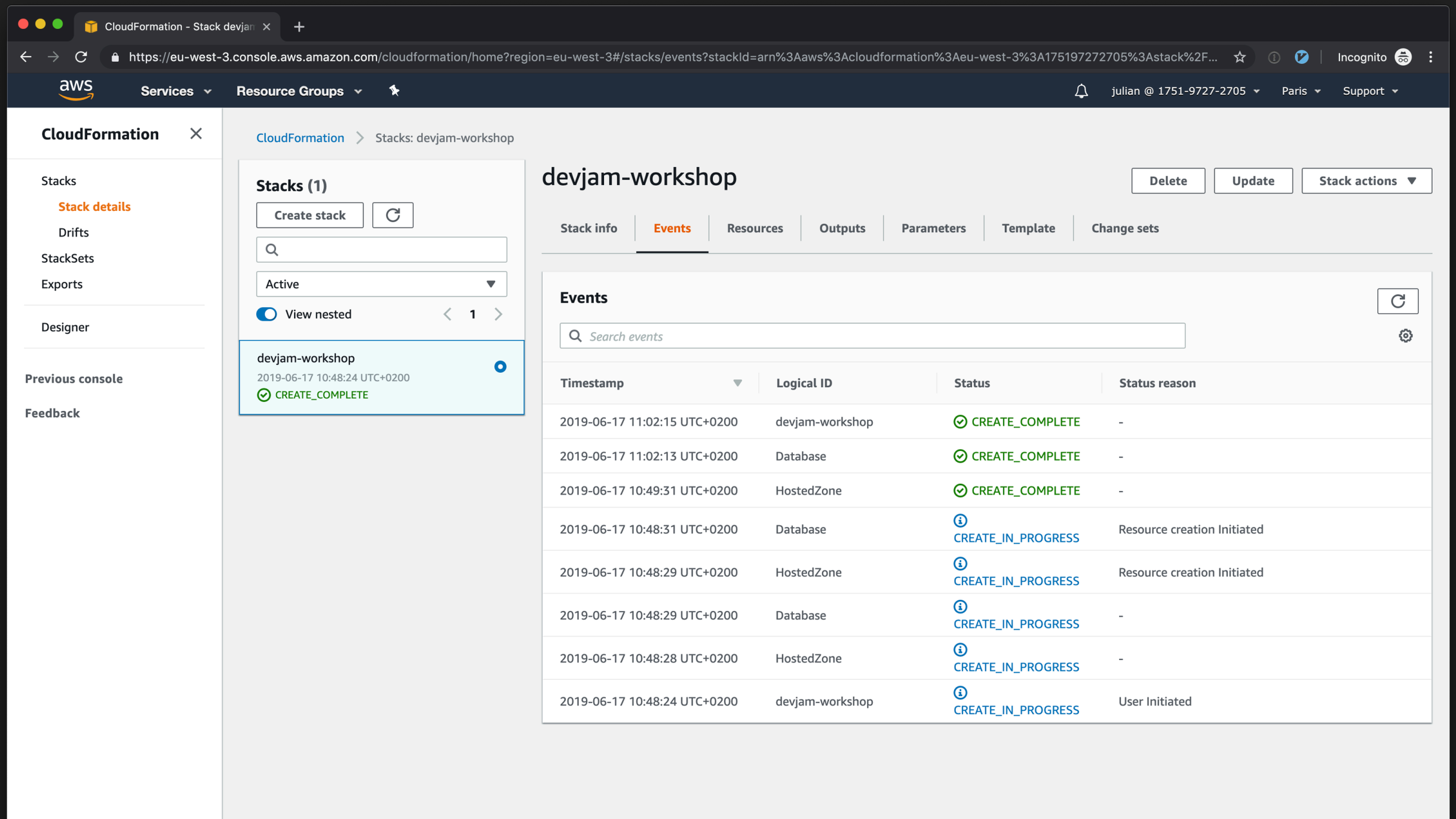
CREATE CLOUDFORMATION STACK



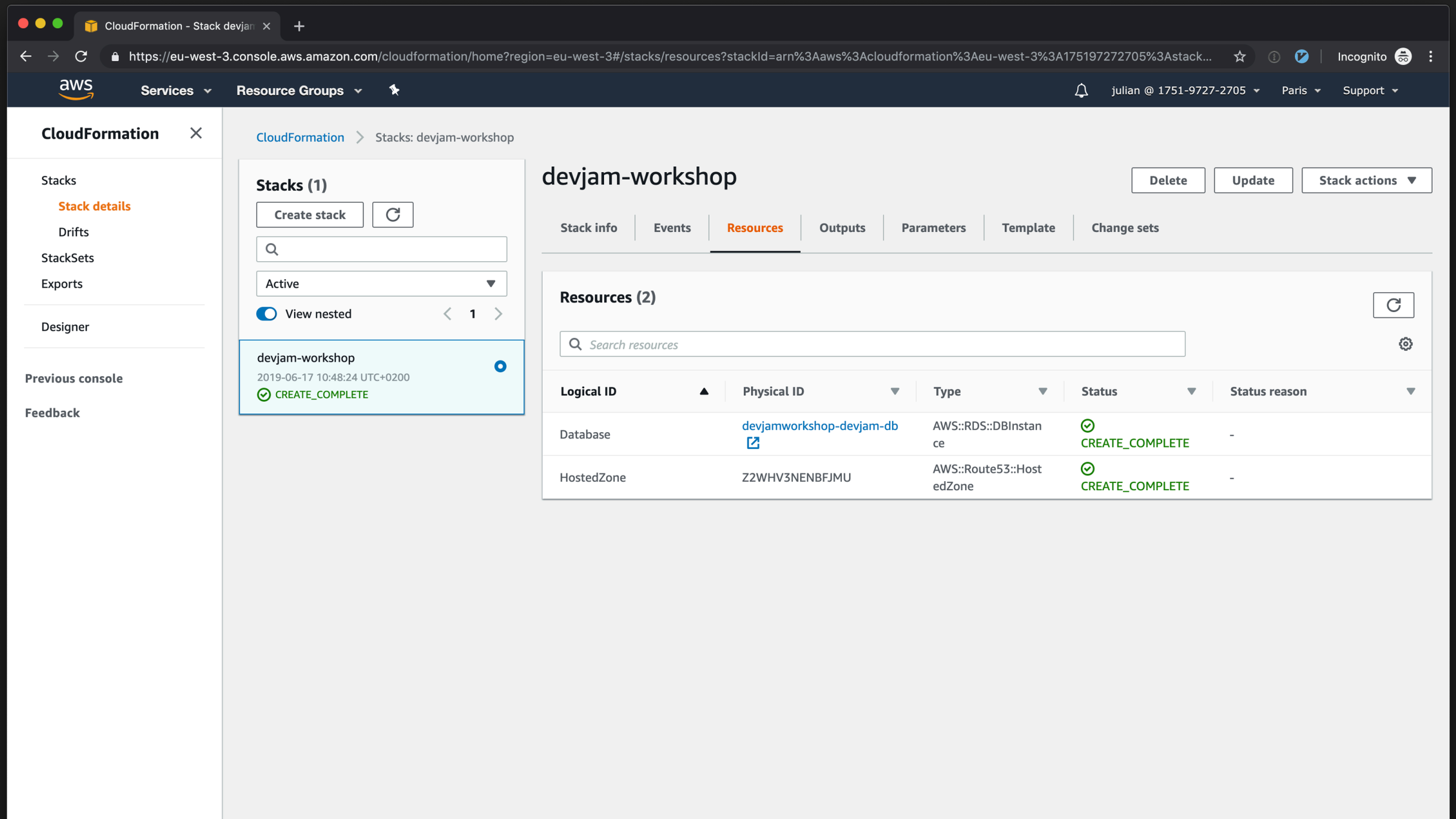
CREATE CLOUDFORMATION STACK

- 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.

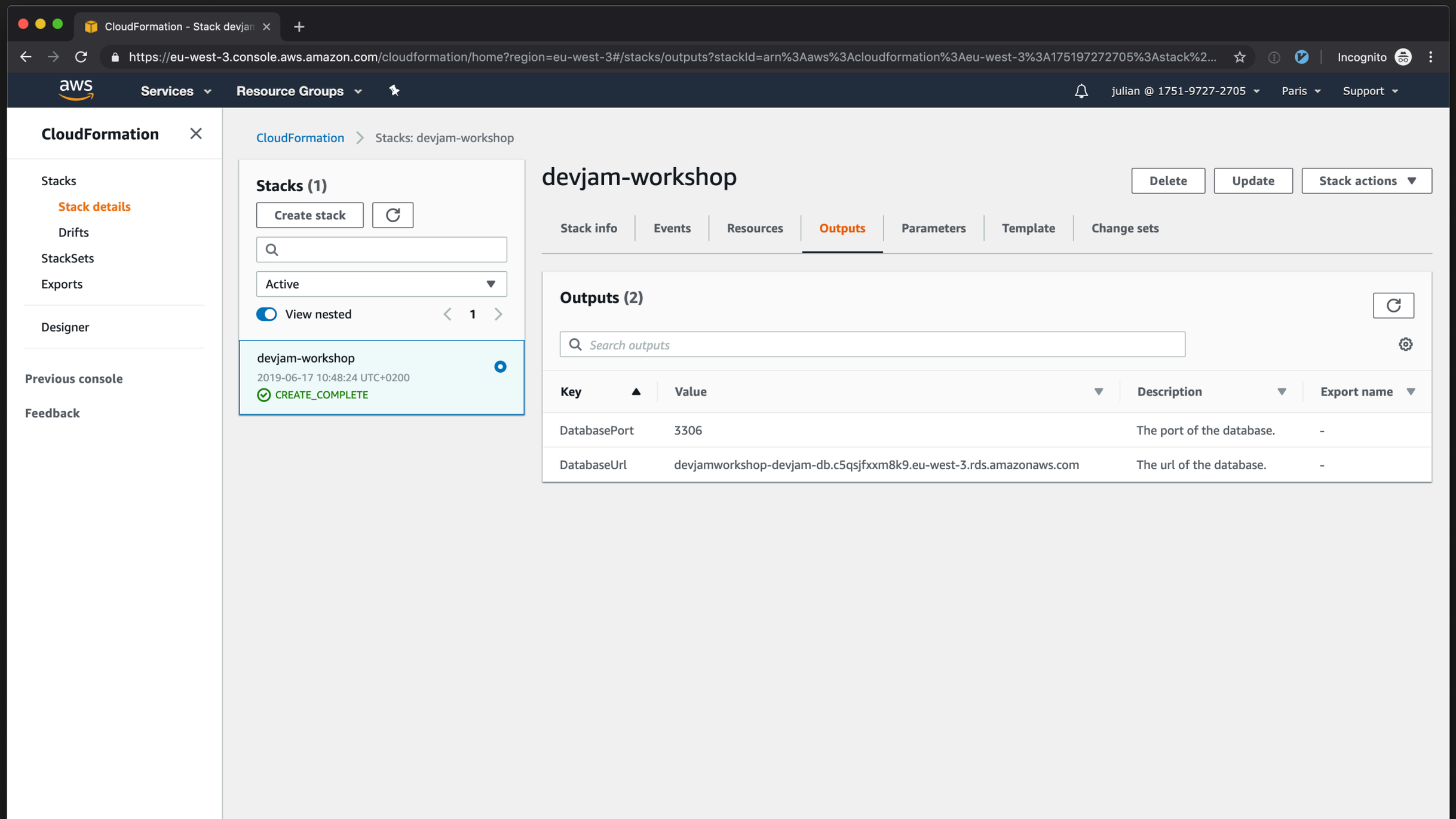
CREATE CLOUDFORMATION STACK



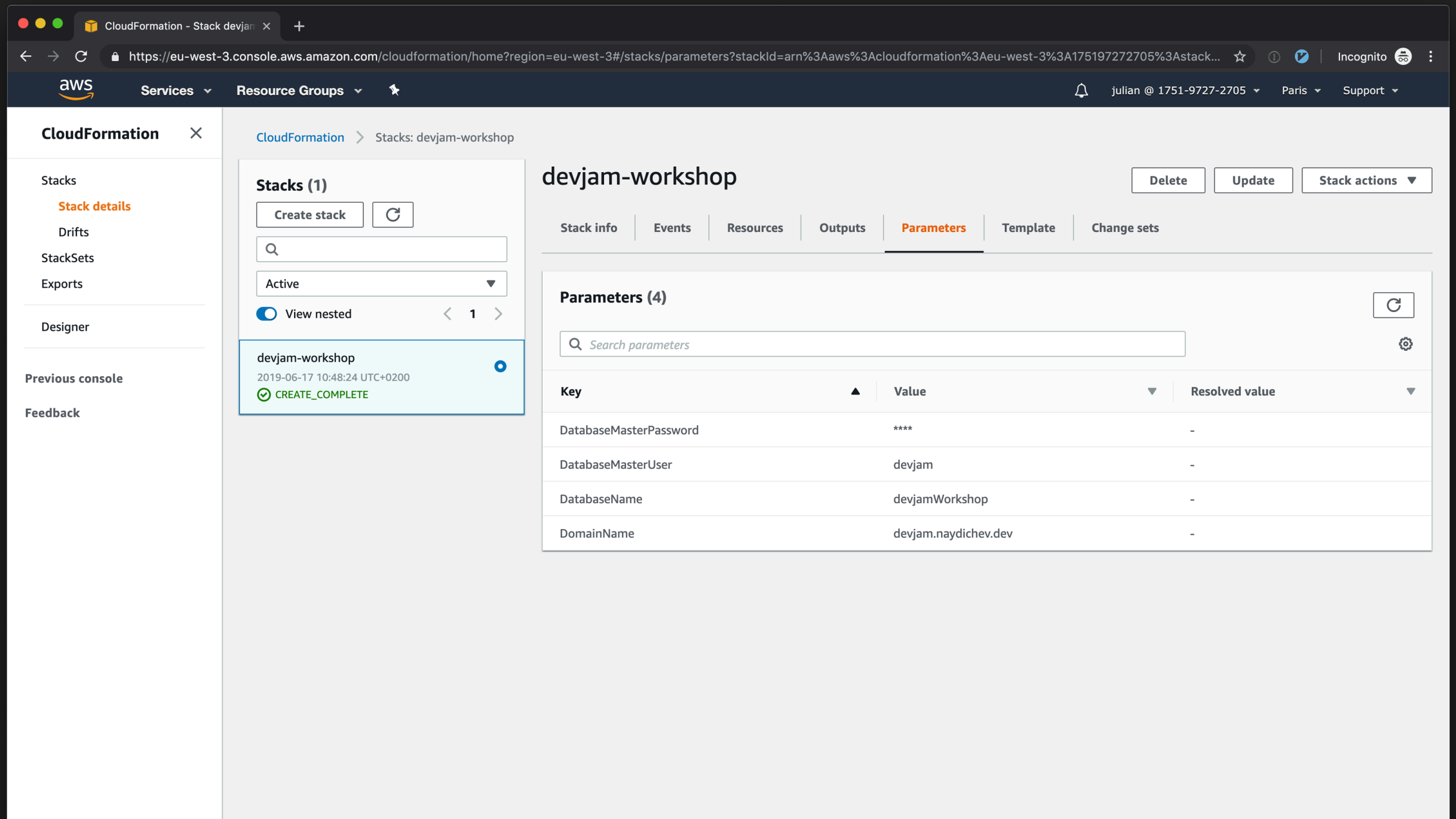
CREATE CLOUDFORMATION STACK



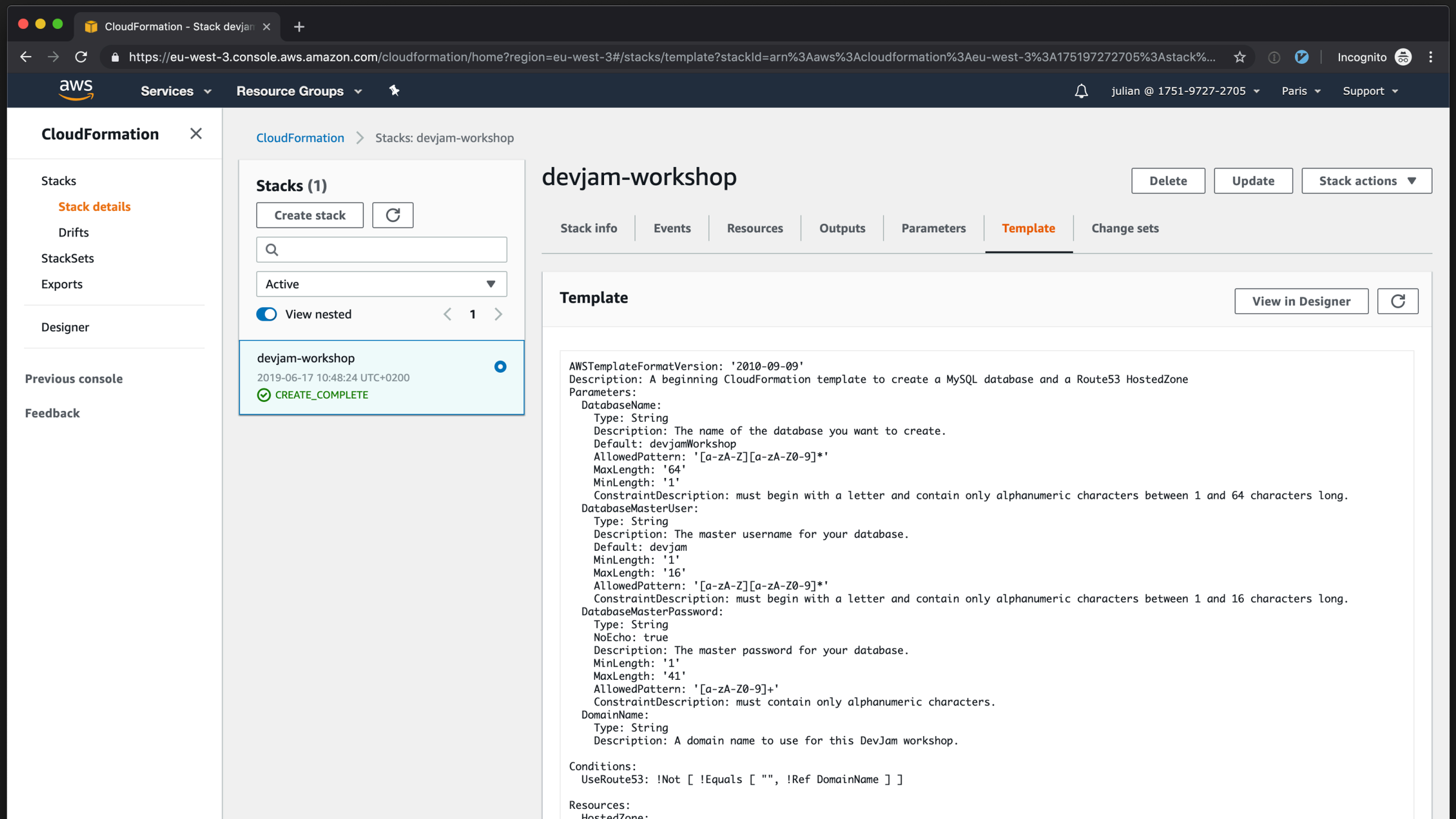
CREATE CLOUDFORMATION STACK



CREATE CLOUDFORMATION STACK



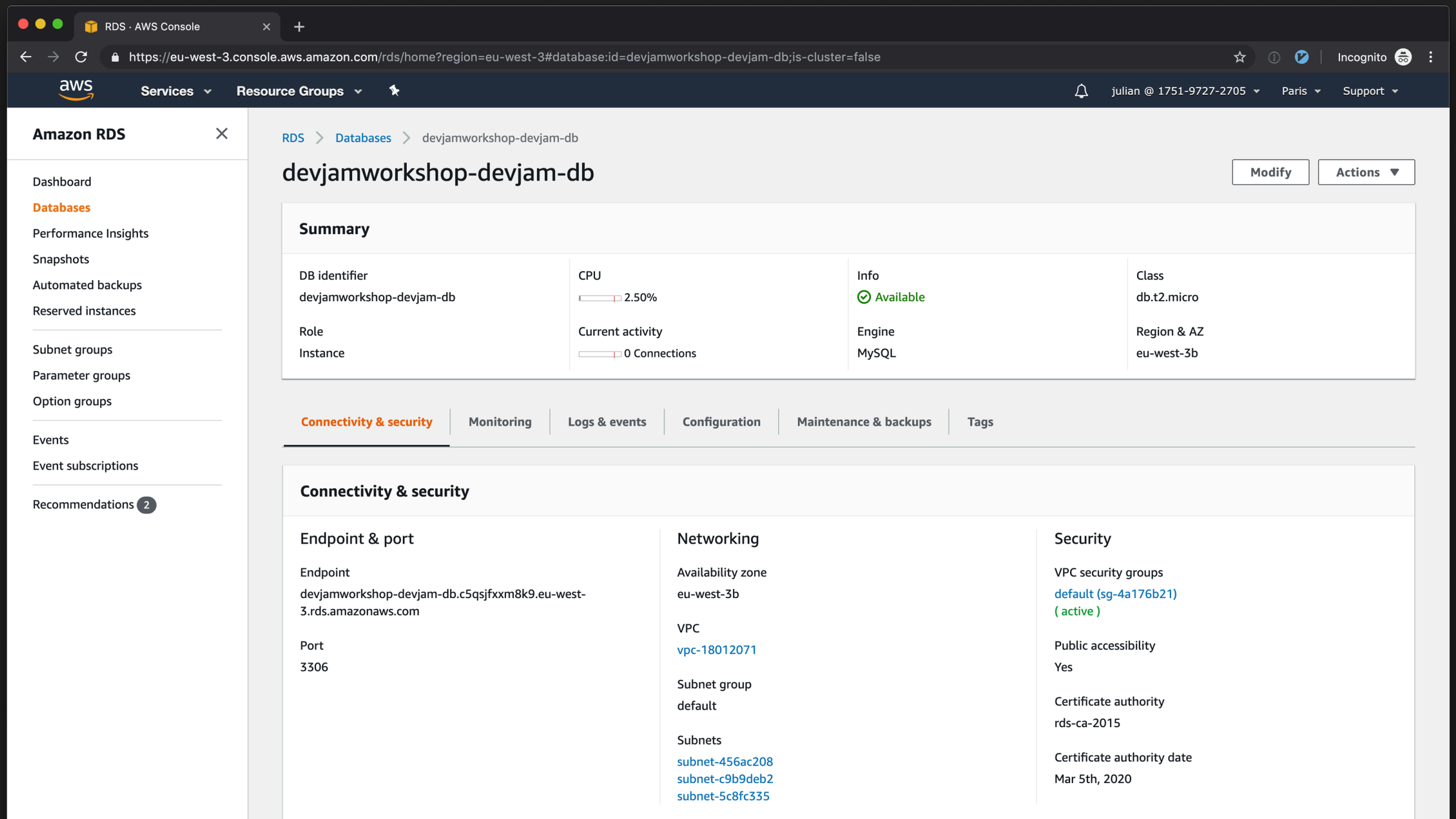
CREATE CLOUDFORMATION STACK



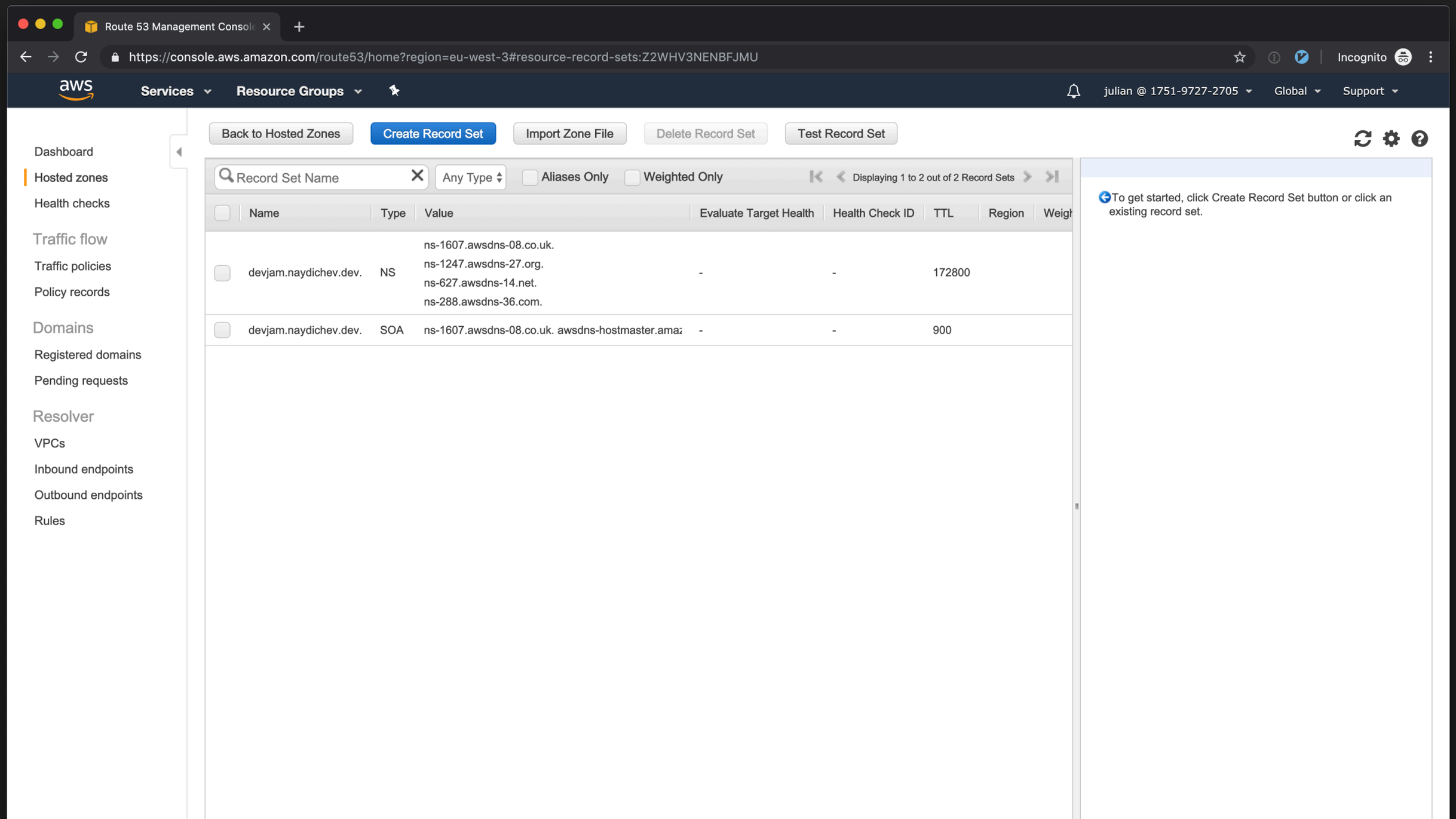
NEW RESOURCES

- 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



NEW RESOURCES – ROUTE53 HOSTEDZONE



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

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: A CloudFormation template to create a certificate.
Parameters:
  DomainName:
    Type: String
    Description: A domain name to use for this DevJam workshop.

Resources:
  Certificate:
    Type: AWS::CertificateManager::Certificate
    Properties:
      DomainName: !Ref DomainName
      ValidationMethod: DNS

Outputs:
  CertificateArn:
    Description: The ARN of the Certificate
    Value: !Ref Certificate
    Export:
      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)

AWS Certificate Manager

https://console.aws.amazon.com/acm/home?region=us-east-1#/

Incognito

Viewing certificates 1 to 2

Name	Domain name	Additional names	Status	Type	In use?	Renewal eligibility
	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.

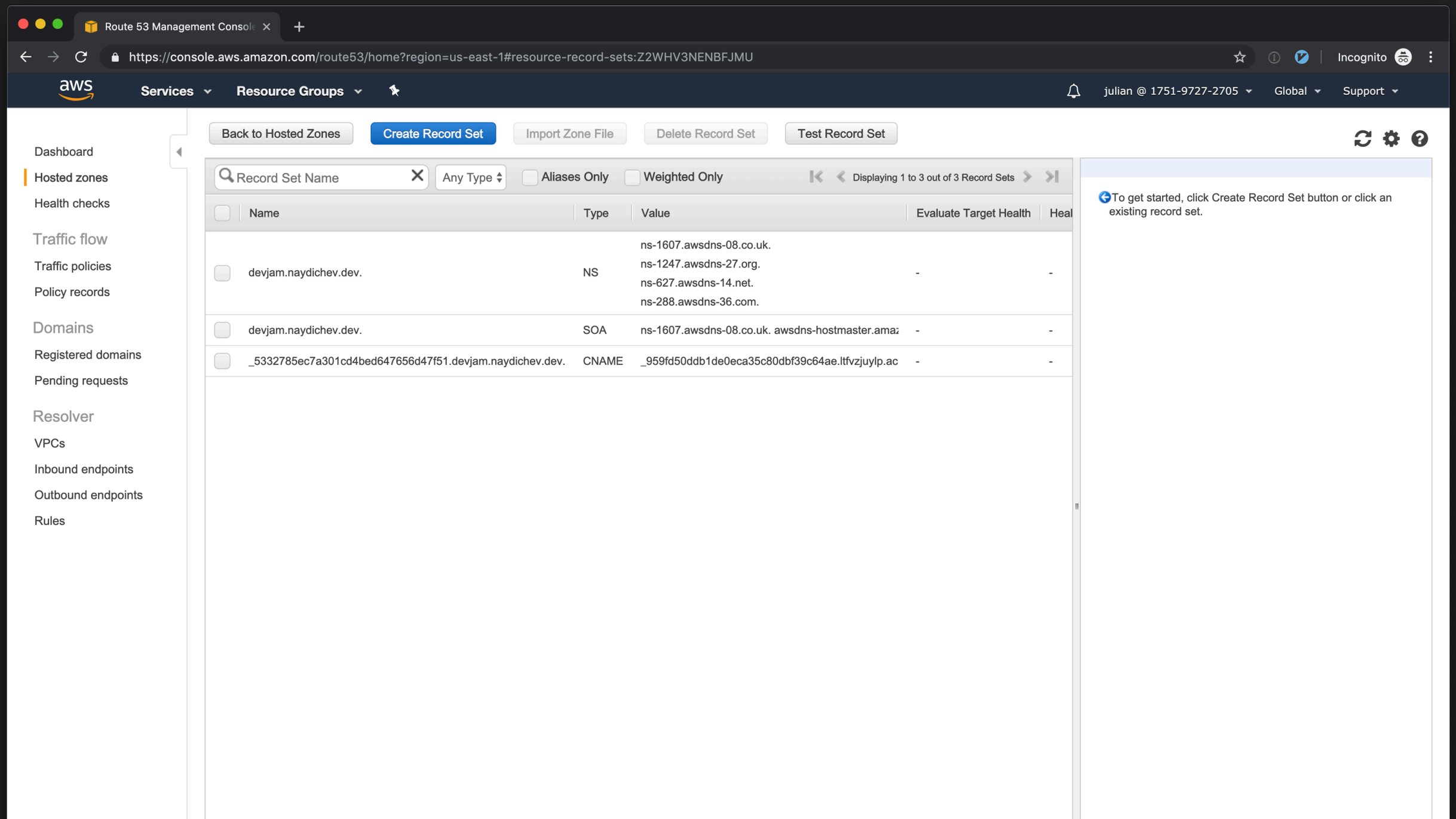
Create record in Route 53 **Amazon Route 53 DNS Customers** ACM can update your DNS configuration for you. Learn more.

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)



CERTIFICATE MANAGER (AKA ACM)

AWS Certificate Manager

https://console.aws.amazon.com/acm/home?region=us-east-1#/

Incognito

aws | Services | Resource Groups | julian @ 1751-9727-2705 | N. Virginia | Support

Certificates
Certificate Manager

Private certificate authority
Private CAs

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

Request a certificate | Import a certificate | Actions

Viewing certificates 1 to 2

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

Status

Status Issued
Detailed status The certificate was issued at 2019-06-17T10:57:16UTC

Domain	Validation status
devjam.naydichev.dev	Success

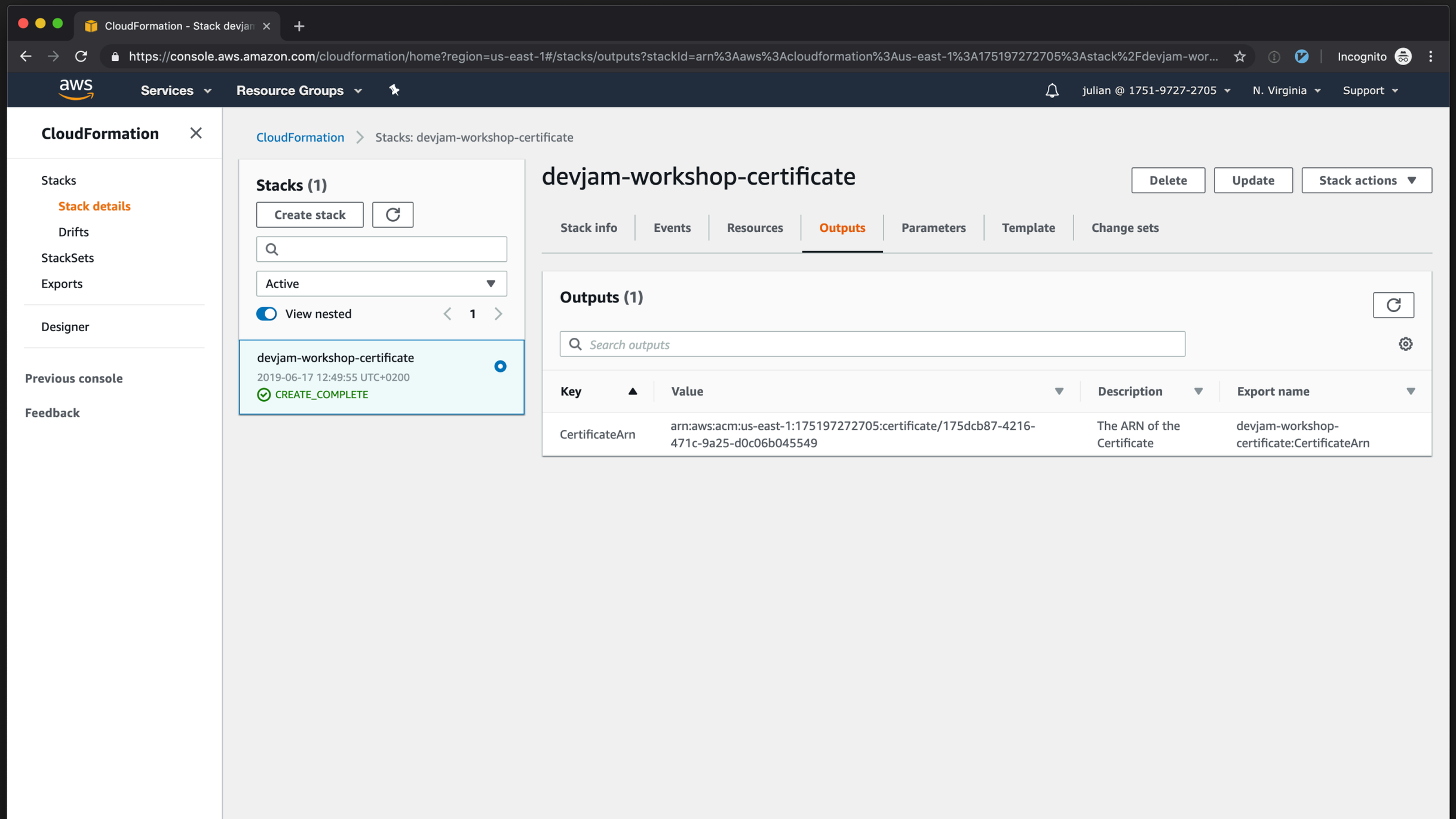
Export DNS configuration to a file — You can export all of the CNAME records to a file

Details

Type Amazon Issued
In use? No
Domain name devjam.naydichev.dev
Number of additional names 0
Identifier 175dcb87-4216-471c-9a25-d0c06b045549
Serial number 06:34:49:e0:7f:d2:5d:6f:44:10:7c:4e:aa:46:5b:1b

Requested at 2019-06-17T10:50:02UTC
Issued at 2019-06-17T10:57:16UTC
Not before 2019-06-17T00:00:00UTC
Not after 2020-07-17T12:00:00UTC
Public key info RSA 2048-bit
Signature algorithm SHA256WITHRSA
ARN arn:aws:acm:us-east-1:175197272705:certificate/175dcb87-4216-471c-9a25-d0c06b045549

CERTIFICATE MANAGER (AKA ACM)

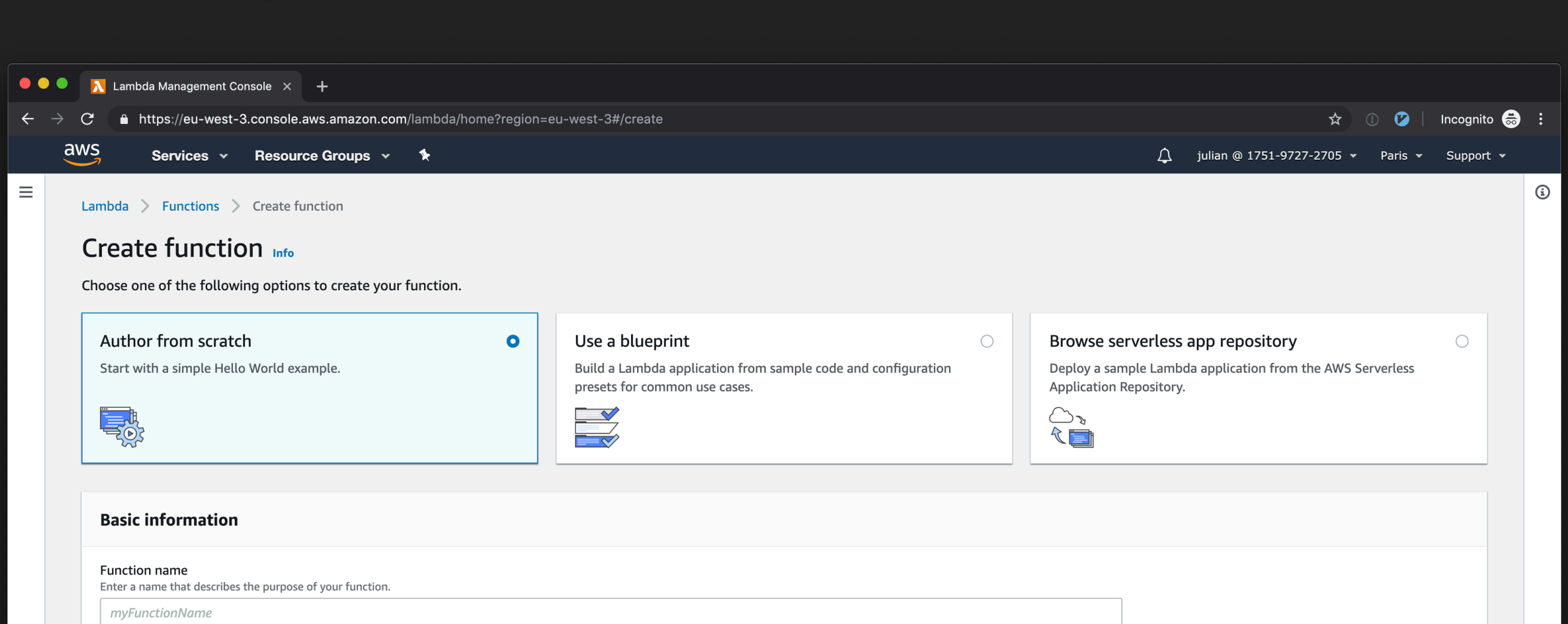


WHAT IS AWS LAMBDA?

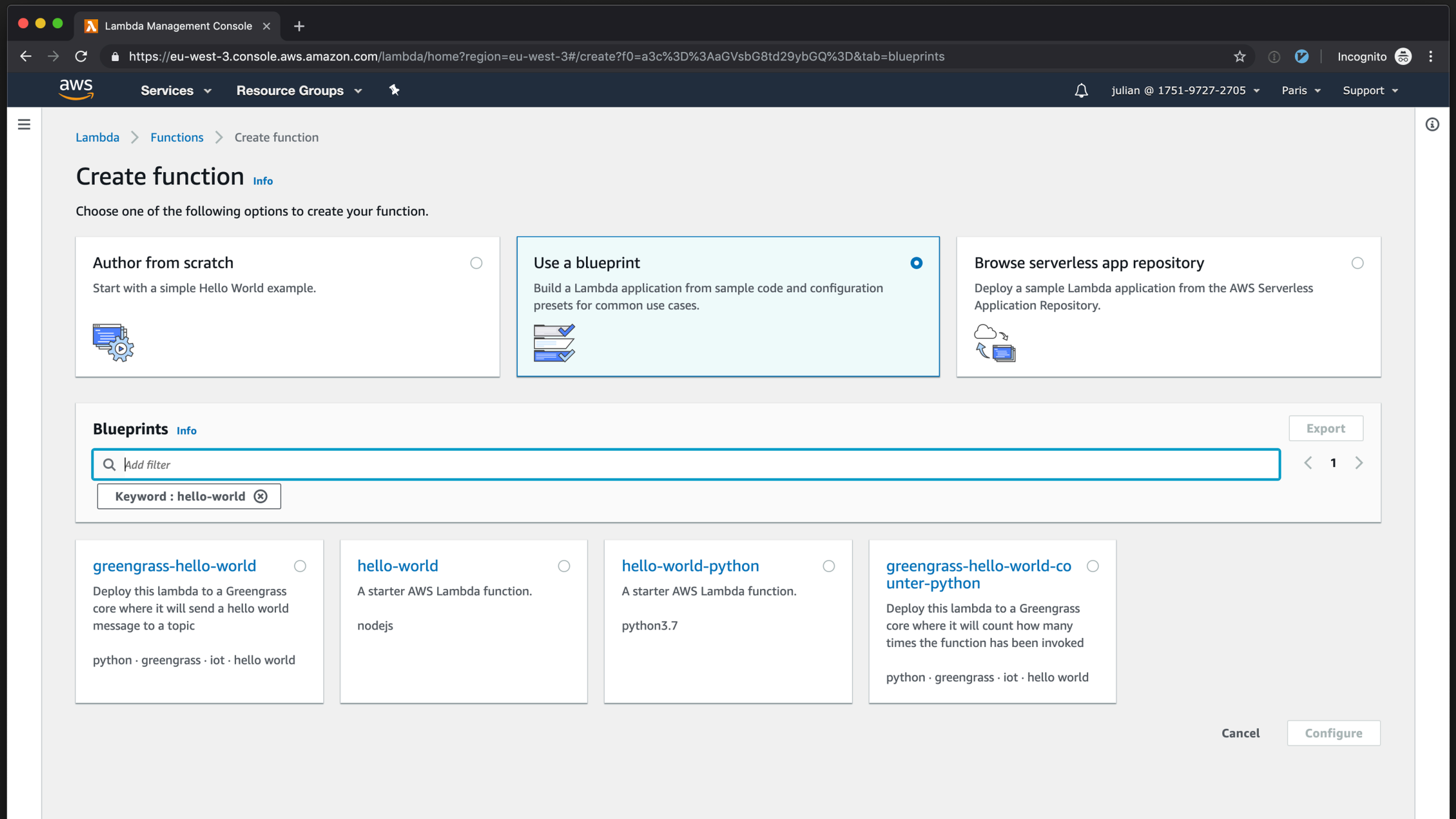
- AWS Lambda is the hot new craze that allows you to run code without needing any infrastructure (mostly).
- There are several runtimes available including Java, Ruby, Javascript and Python, just to name a few.
- AWS Lambda integrates with a number of other services which allows you to easily respond 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.



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
Incognito
aws
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
greengrass-hello-world
Deploy this lambda to a Greengrass core where it will send a hello world message to a topic
python · greengrass · iot · hello world
hello-world
A starter AWS Lambda function.
nodejs
hello-world-python
A starter AWS Lambda function.
python3.7
greengrass-hello-world-counter-python
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

Incognito

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

Lambda > Functions > Create function > Using blueprint hello-world-python

Basic information Info

Function name

hello-world

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the **IAM console**.

Create a new role with basic Lambda permissions

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

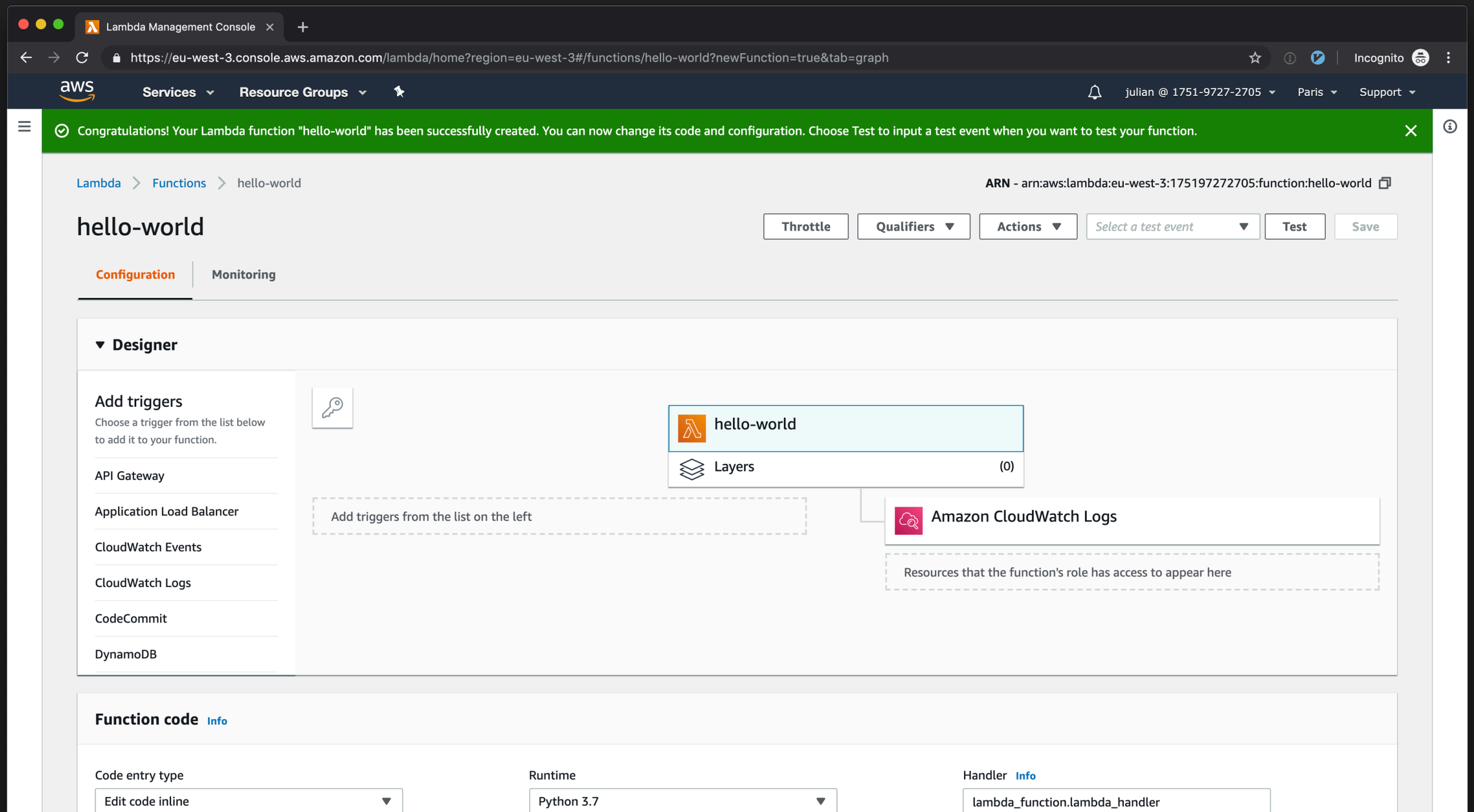
Code is preconfigured by the chosen blueprint. You can configure it after you create the function.

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.
Lambda > Functions > hello-world
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

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

hello-world

Throttle Qualifiers Actions Select a test event Test Save

Code entry type: Edit code inline

Runtime: Python 3.7

Handler Info: lambda_function.lambda_handler

File Edit Find View Go Tools Window

Environment

hello-world

lambda_function.py

lambda_function

```python
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')

```

1:1 Python Spaces: 4

AWS LAMBDA

Lambda Management Console

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

Incognito

aws | 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

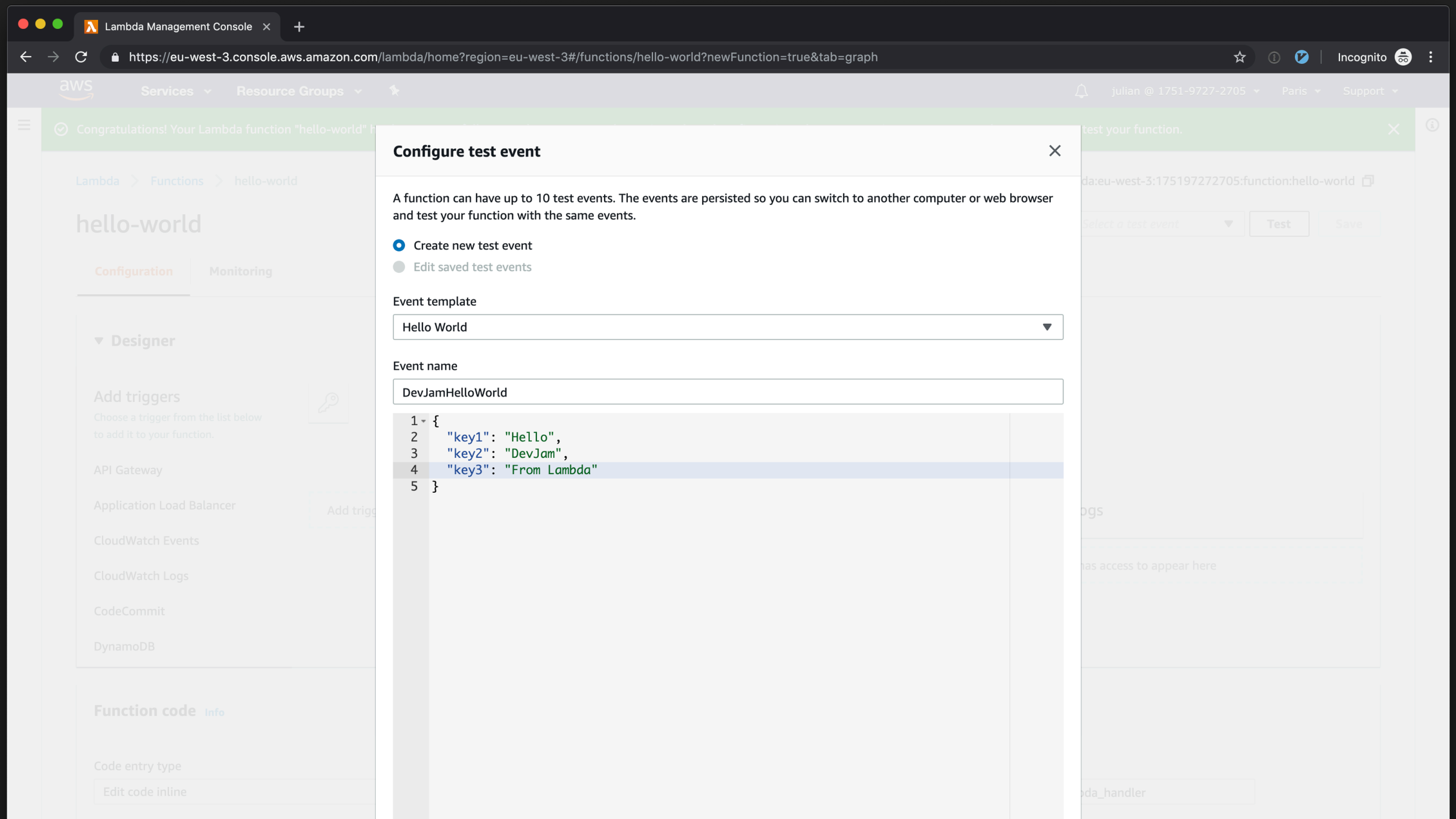
Your function is allocated CPU proportional to the memory configured.

128 MB

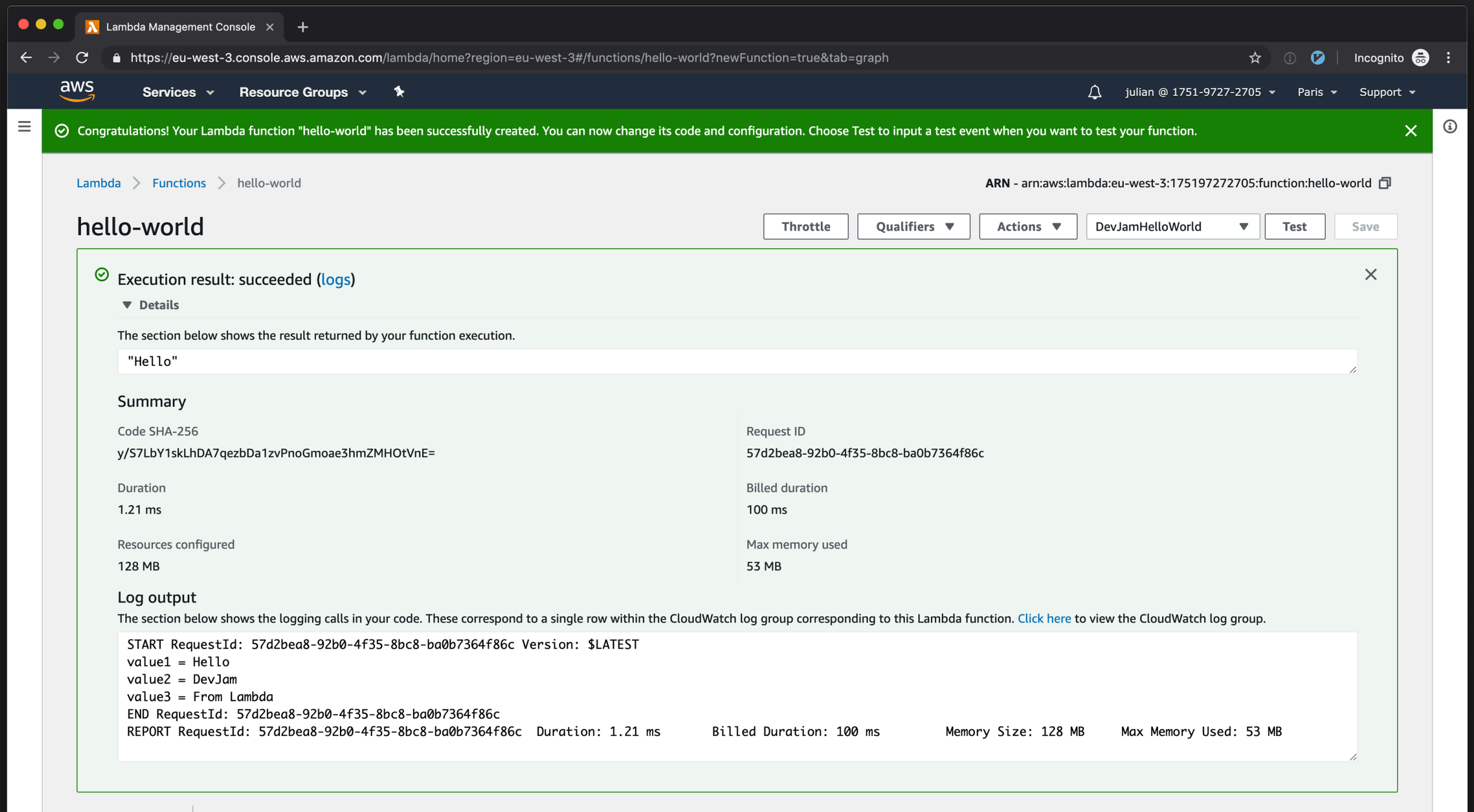
Timeout Info

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
Incognito
Services
Resource Groups
julian @ 1751-9727-2705
Paris
Support
Congratulations! Your Lambda function "hello-world"
test your function.
Lambda > Functions > hello-world
hello-world
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
Function code Info
Code entry type
Edit code inline
Test
Save
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
{
"key1": "Hello",
"key2": "DevJam",
"key3": "From Lambda"
}

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
y/S7LbY1skLhDA7qezbDa1zvPnoGmoae3hmZMHOtVnE=
Request ID
57d2bea8-92b0-4f35-8bc8-ba0b7364f86c
Duration
1.21 ms
Billed duration
100 ms
Resources configured
128 MB
Max memory used
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.

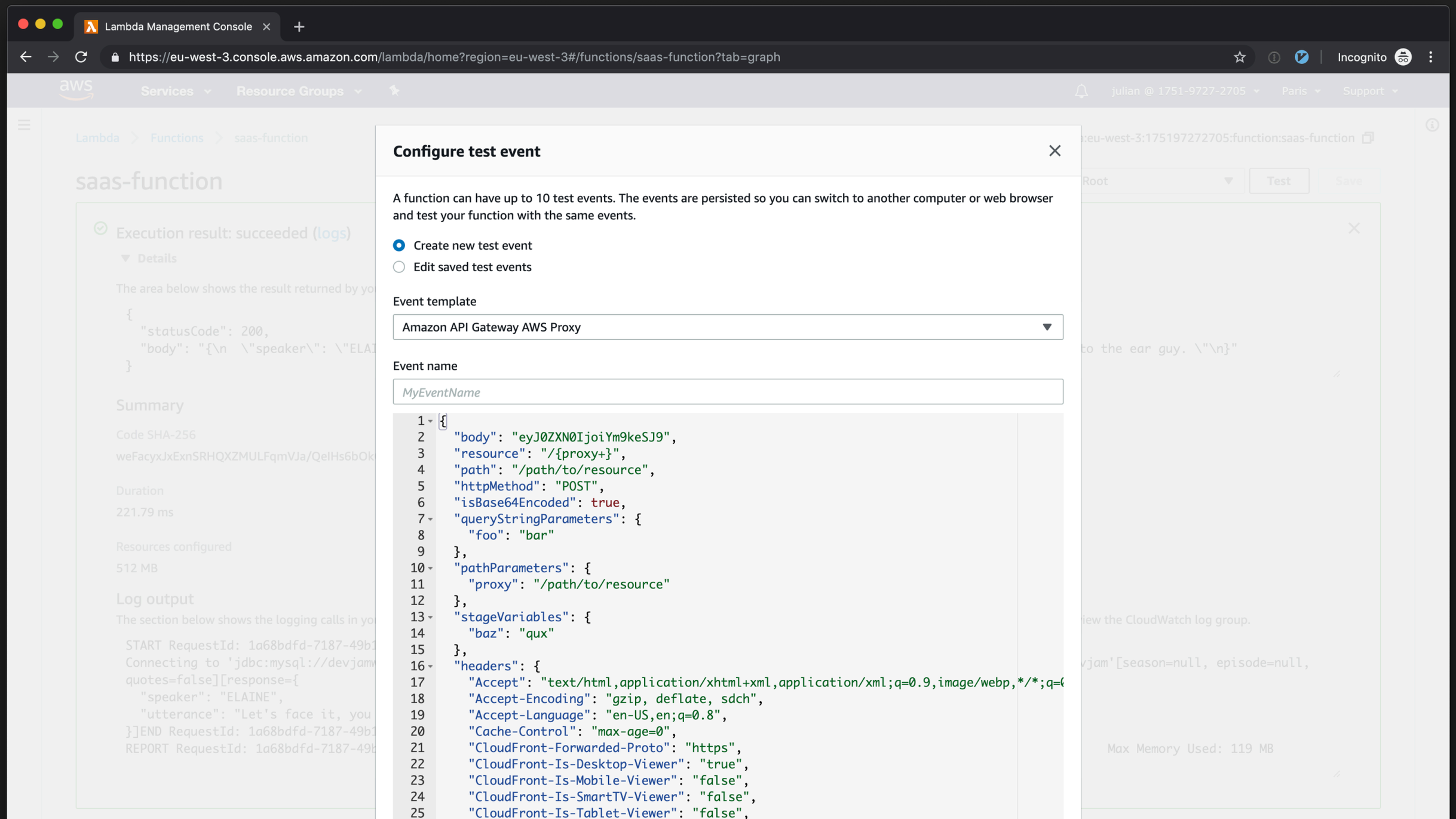
SAM – SERVERLESS APPLICATION MODEL

- 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



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

```yaml
Resources:
  HostedZone:
    Condition: UseRoute53
    Type: AWS::Route53::HostedZone
    Properties:
      Name: !Ref DomainName
  DatabaseSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: "An all access pass to data"
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          FromPort: 3306
          ToPort: 3306
          IpProtocol: tcp
  Database:
    Type: AWS::RDS::DBInstance
    Properties:
      AllocatedStorage: "5"
      DBInstanceClass: db.t2.micro
      DBInstanceIdentifier: !Sub
        - "${DBName}-devjam-db"
        - { DBName: !Ref DatabaseName }
      DBName: !Ref DatabaseName
      Engine: MySQL
      MasterUserPassword: !Ref DatabaseMasterPassword
      MasterUsername: !Ref DatabaseMasterUser
      MultiAZ: true
      VPCSecurityGroups:
        - !GetAtt [ DatabaseSecurityGroup, GroupId ]

  DeploymentBucket:
    Type: AWS::S3::Bucket

Outputs:
  DatabaseUrl:
    Description: The url of the database.
    Value: !GetAtt Database.Endpoint.Address
  DatabasePort:
    Description: The port of the database.
    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

03-add-lambda-function.yaml (~/Workspace/SaaS/cloudformation) - VIM1

```yaml
86
87  DeploymentBucket:
88    Type: AWS::S3::Bucket
89
90  LambdaExecutionRole:
91    Type: AWS::IAM::Role
92    Properties:
93      AssumeRolePolicyDocument:
94        Version: '2012-10-17'
95        Statement:
96        - Effect: Allow
97          Principal:
98            Service:
99            - lambda.amazonaws.com
100         Action:
101         - sts:AssumeRole
102     Path: "/"
103     Policies:
104     - PolicyName: root
105       PolicyDocument:
106         Version: '2012-10-17'
107         Statement:
108         - Effect: Allow
109           Action:
110           - logs:*
111           Resource: arn:aws:logs:*:*:*
112
113 LambdaFunction:
114   Type: AWS::Lambda::Function
115   Properties:
116     Code:
117       S3Bucket: !Ref DeploymentBucket
118       S3Key: !Ref ArtifactKey
119     Handler: !Ref LambdaHandler
120     Runtime: !Ref LambdaRuntime
121     Role: !GetAtt LambdaExecutionRole.Arn
122     Timeout: 75
123     Memory: 256
124     Environment:
125       Variables:
126         DB_USER: !Ref DatabaseMasterUser
127         DB_PASS: !Ref DatabaseMasterPassword
128         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
167  # NOTE: change the name of this resource everytime you want to deploy a change to API Gateway.
168  APIGatewayDeployment:
169    DependsOn:
```

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 server 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