

Managed Cloud Services

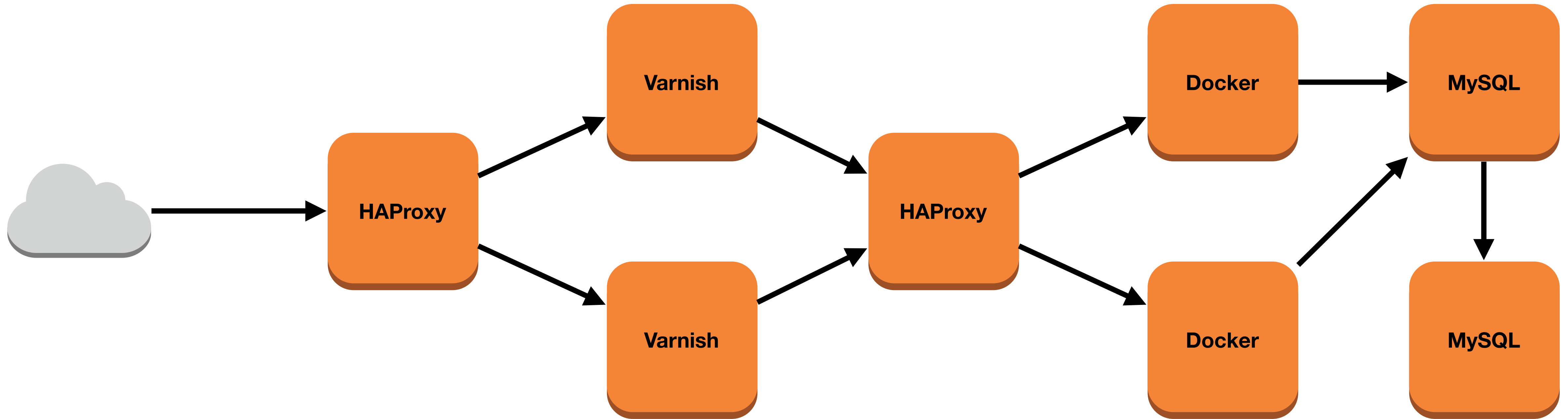
When you don't want to run it yourself

Managed Cloud Services

Virtual Servers vs Cloud Services

- All the pieces of internet applications began as discrete software run on a server you managed
- Everyone had to be at least an intermediate level sysadmin
- Managed Cloud Services aim to take away the “undifferentiated heavy lifting” from your application stack

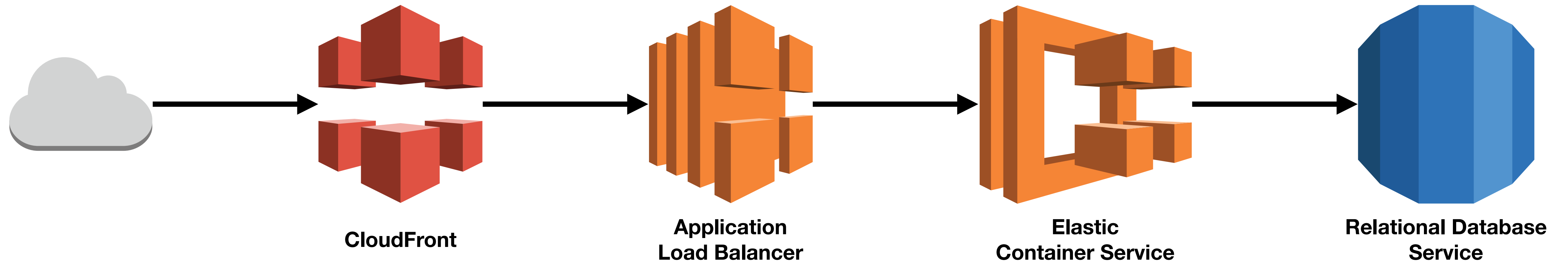
VM Centric Architecture



Managed Cloud Services

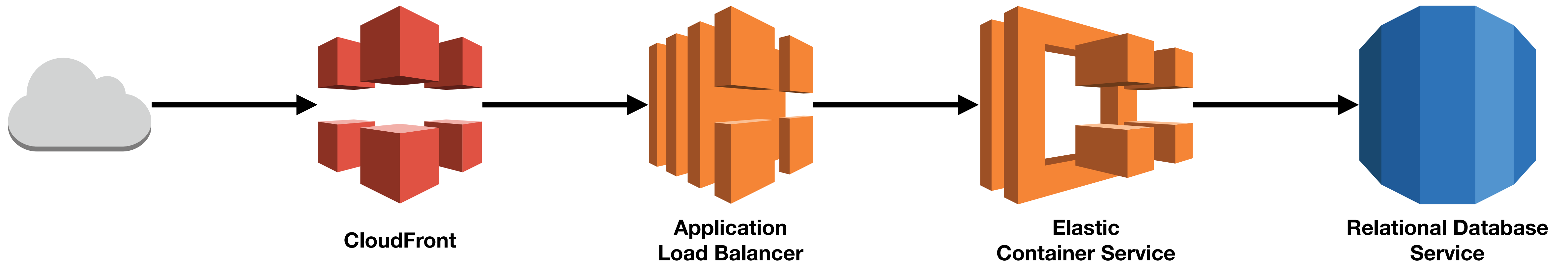
	VM / EC2	AWS Service
Database	MySQL	RDS MySQL
Load Balancer	HAProxy	Elastic Load Balancer Application Load Balancer
Docker	Docker	Elastic Container Service
Cacheing	Varnish	CloudFront

Cloud Centric Architecture

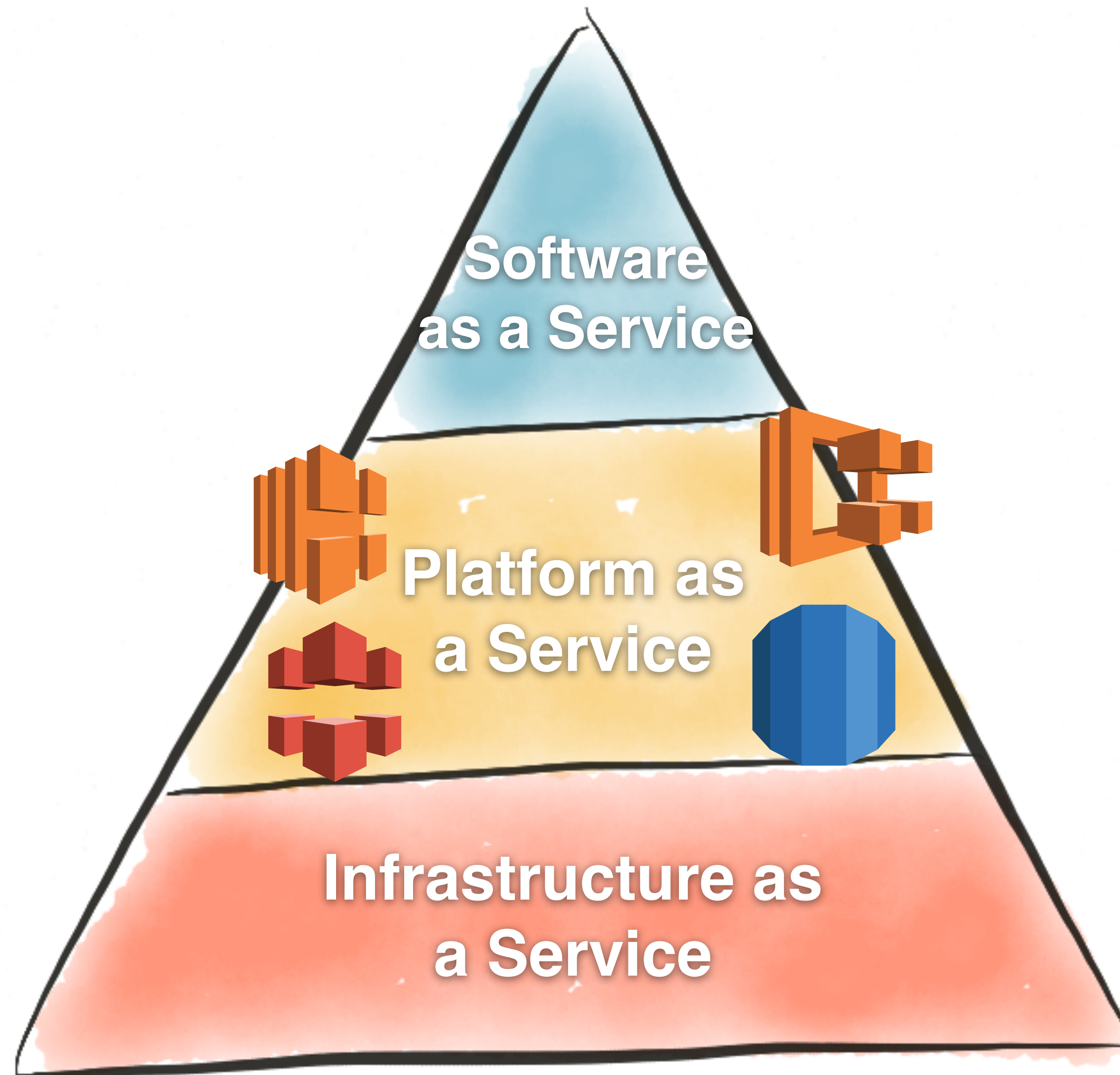


Managed Cloud Services

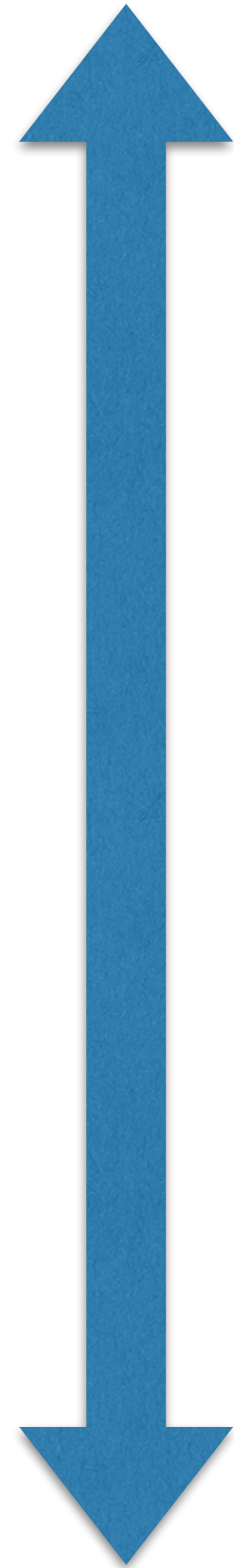
Virtual Servers vs Cloud Services



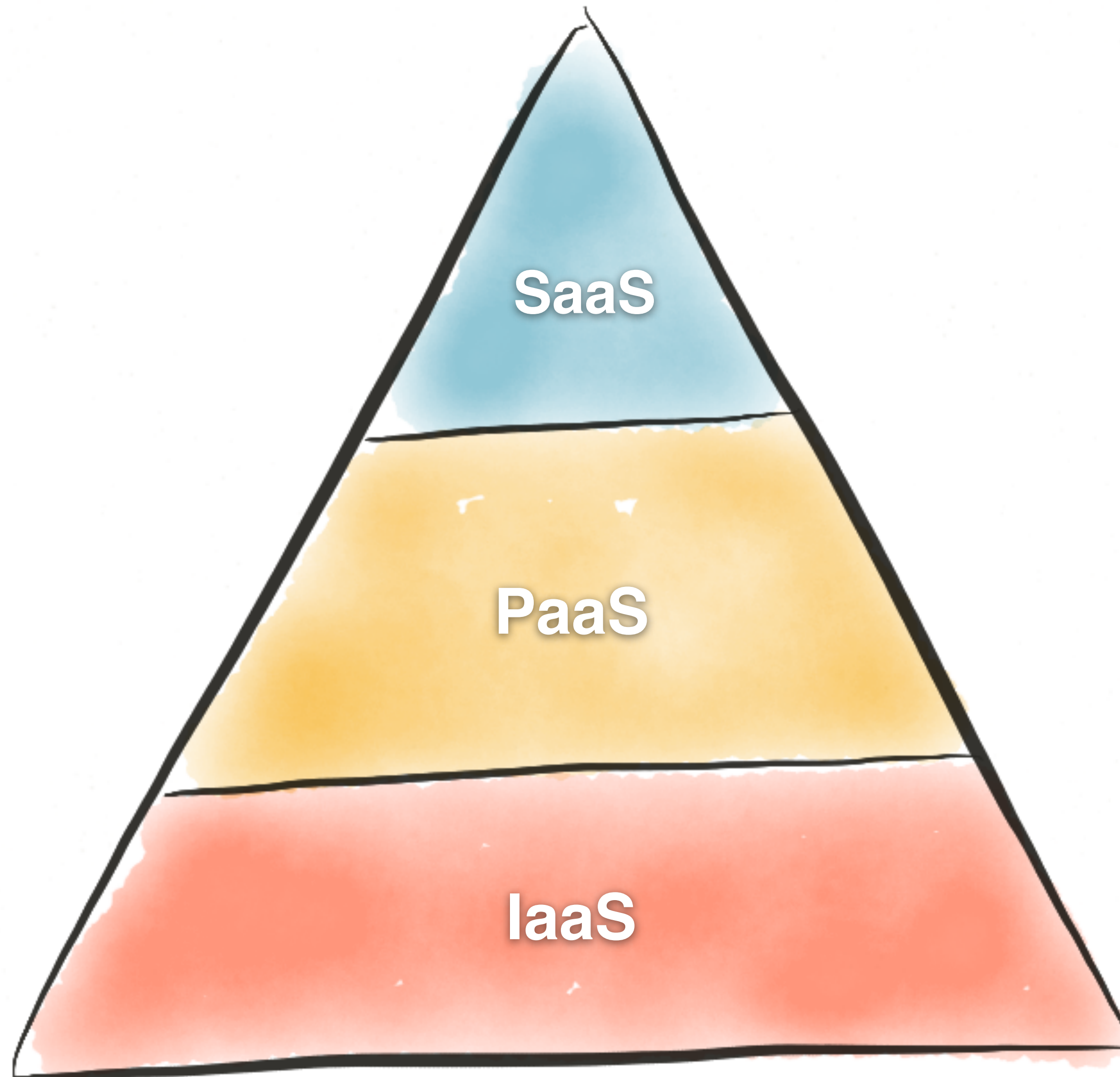
- All these AWS services are highly available, fault tolerant, and can be automatically deployed and backed up
- Only the RDS instance needs to be updated, and ~80% of that is automatic



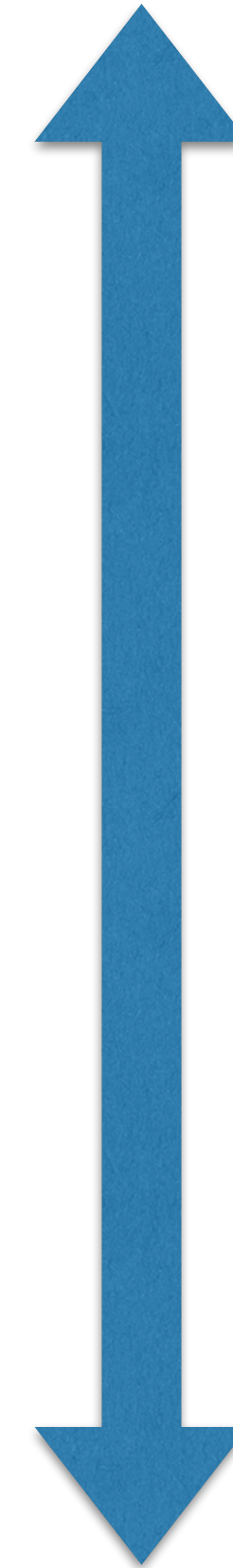
Less
Maintenance



More
Control



More \$\$
Less Time



Less \$\$
More Time

AWS S3

Simple Storage Service

AWS S3

Cloud Object Storage

- Amazon S3 is an object storage service that stores data as objects within buckets.
- An object is a file and any metadata that describes the file.
- A bucket is a container for objects.
- Not a File System
- Read/Write object data through AWS API

AWS S3

Cloud Object Storage

- Bucket names must be globally unique
- No size limits
- Objects can be public or private
- Public objects can have URLs for direct access
 - This makes S3 ideal for storing data on the internet you want other people to access.

AWS S3

S3 Public Website Bucket

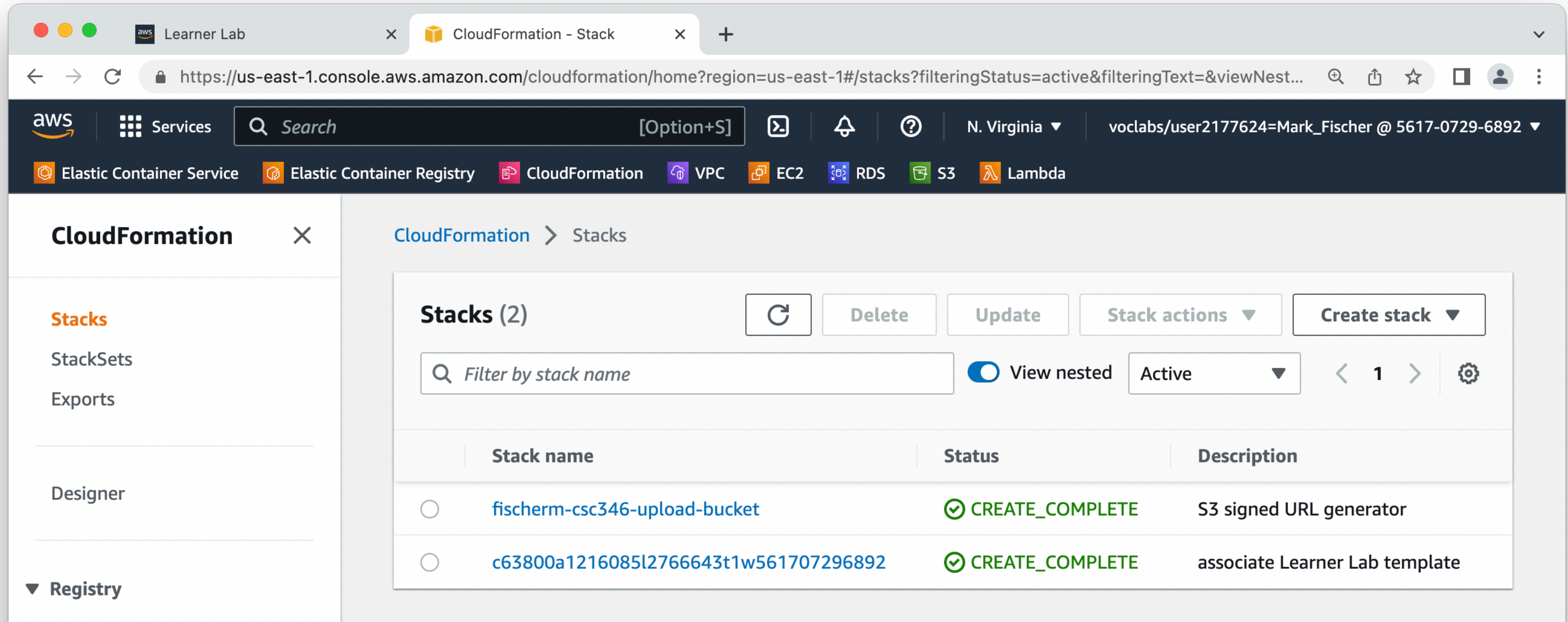
- There are enough little things that need to be configured on an S3 bucket to allow for public web access that I built a CloudFormation template to codify it.
- AWS has a full tutorial for this:

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/HostingWebsiteOnS3Setup.html>

AWS S3

S3 Public Website Bucket

- To deploy the template, go to the CloudFormation console in the web UI.



The screenshot shows the AWS CloudFormation console interface. The browser address bar indicates the URL: `https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks?filteringStatus=active&filteringText=&viewNest...`. The console header includes the AWS logo, a search bar, and navigation icons. The main content area displays the 'Stacks (2)' section with a table of active stacks.

Stack name	Status	Description
fischem-csc346-upload-bucket	✔ CREATE_COMPLETE	S3 signed URL generator
c63800a1216085l2766643t1w561707296892	✔ CREATE_COMPLETE	associate Learner Lab template

AWS S3

S3 Public Website Bucket

- Create a new stack with new resources

The screenshot shows the AWS CloudFormation console. The top navigation bar includes the AWS logo, a search bar, and the current region (N. Virginia). The main content area is titled 'CloudFormation > Stacks'. There are two stacks listed in a table, both with a status of 'CREATE_COMPLETE'. A blue box highlights the 'Create stack' button and its dropdown menu, which is open to show two options: 'With new resources (standard)' and 'With existing resources (import resources)'. The 'With new resources (standard)' option is selected.

Stack name	Status	Description
fischem-csc346-upload-bucket	CREATE_COMPLETE	S3 signed URL generator
c63800a1216085l2766643t1w561707296892	CREATE_COMPLETE	associate Learner Lab template

Create stack

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

Amazon S3 URL

Amazon S3 template URL

`s3 https://fischem-csc346-download.s3.amazonaws.com/s3_template.yaml`

Cancel

Next

AWS S3

S3 Public Website Bucket

- You can use my template directly from my class bucket.

Specify stack details

Stack name

Stack name

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.

BucketName

The name of the S3 bucket.

Cancel

Previous

Next

AWS S3

S3 Public Website Bucket

- You need to specify a Stack name
- There's one parameter for this template, the bucket name
- I often have the stack name and bucket name be the same. Makes things simple
- Create a unique bucket name!

fischerm-csc346-upload-bucket

Delete

Update

Stack actions ▼

Create stack ▼

Stack info

Events

Resources

Outputs

Parameters

Template

Change sets

Events (9)

Search events

Timestamp	Logical ID	Status	Status reason
2022-10-30 20:37:58 UTC-0700	fischerm-csc346-upload-bucket	✔ CREATE_COMPLETE	-
2022-10-30 20:37:57 UTC-0700	S3BucketPublicPolicy	✔ CREATE_COMPLETE	-
2022-10-30 20:37:57 UTC-0700	S3BucketPublicPolicy	ⓘ CREATE_IN_PROGRESS	Resource creation Initiated
2022-10-30 20:37:56 UTC-0700	S3BucketPublicPolicy	ⓘ CREATE_IN_PROGRESS	-
2022-10-30 20:37:54 UTC-0700	S3UploadBucket	✔ CREATE_COMPLETE	-
2022-10-30 20:37:33 UTC-0700	S3UploadBucket	ⓘ CREATE_IN_PROGRESS	Resource creation Initiated
2022-10-30 20:37:32 UTC-0700	S3UploadBucket	ⓘ CREATE_IN_PROGRESS	-

AWS S3

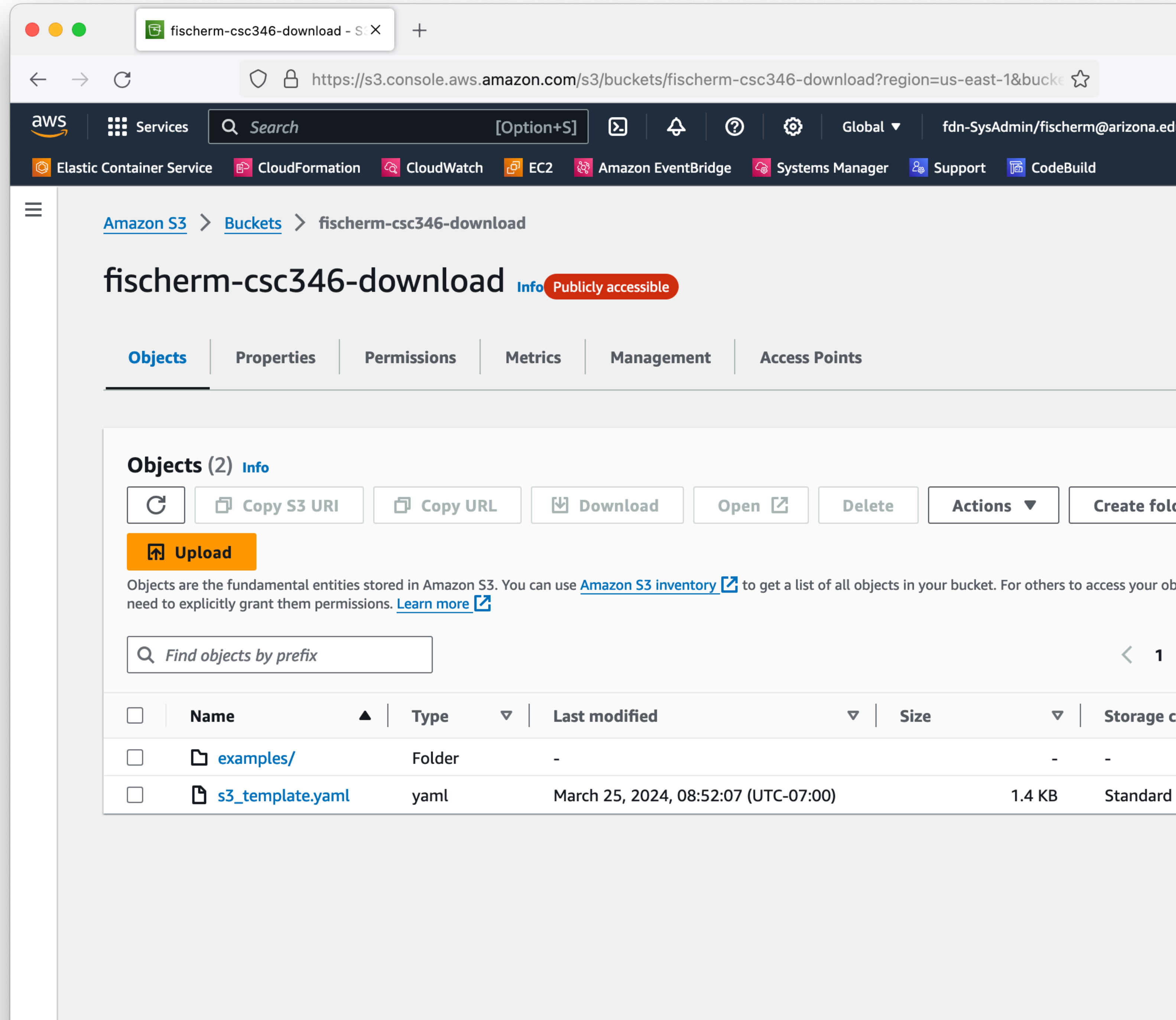
S3 Public Website Bucket

- Click through to deploy the stack
- Once the stack reaches **CREATE_COMPLETE** your S3 bucket should be created and configured correctly to host files able to be accessed publicly.
- We will use this in an upcoming homework to store images for our chat app.

AWS S3

Cloud Object Storage

- Clicking on a bucket shows its contents
- Can create “folders” and upload objects directly in the web UI



The screenshot shows the AWS S3 console interface for a bucket named 'fischem-csc346-download'. The bucket is publicly accessible. The 'Objects' tab is selected, showing a list of two objects: 'examples/' (Folder) and 's3_template.yaml' (yaml file). The console includes navigation menus, search bars, and various action buttons like 'Upload', 'Download', and 'Delete'.

Amazon S3 > Buckets > fischem-csc346-download

fischem-csc346-download Info Publicly accessible

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

Objects (2) Info

[Refresh](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#)

[Upload](#)

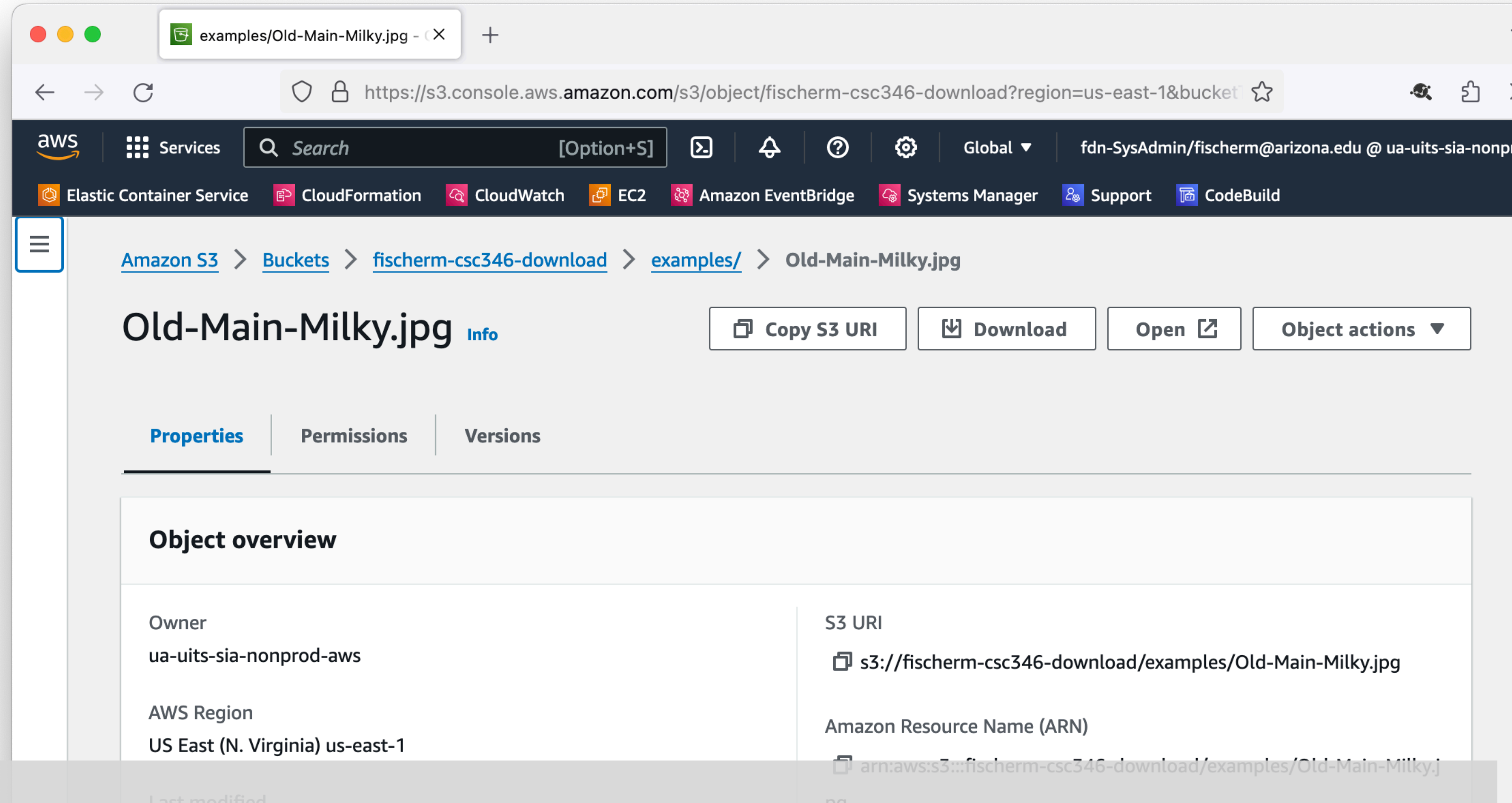
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you need to explicitly grant them permissions. [Learn more](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	examples/	Folder	-	-	-
<input type="checkbox"/>	s3_template.yaml	yaml	March 25, 2024, 08:52:07 (UTC-07:00)	1.4 KB	Standard

AWS S3

Cloud Object Storage

- “Folders” are just part of the object key
- It’s not a File System



```
arn:aws:s3::fischer-csc346-download/examples/Old-Main-Milky.jpg
```

Bucket Name

Object Key

AWS S3

Cloud Object Storage

- If configured as a public website, objects have publicly available URLs
- You can download this image from the URL

The screenshot shows the AWS S3 console interface. The breadcrumb navigation is: Amazon S3 > Buckets > fischer-csc346-download > examples/ > Old-Main-Milky.jpg. The object name 'Old-Main-Milky.jpg' is displayed with an 'Info' link. Action buttons include 'Copy S3 URI', 'Download', 'Open', and 'Object actions'. The 'Properties' tab is selected, showing the following details:

Object overview	
Owner	ua-uits-sia-nonprod-aws
AWS Region	US East (N. Virginia) us-east-1
Last modified	March 25, 2024, 08:55:17 (UTC-07:00)
Size	5.5 MB
Type	jpg
Key	examples/Old-Main-Milky.jpg
S3 URI	s3://fischer-csc346-download/examples/Old-Main-Milky.jpg
Amazon Resource Name (ARN)	arn:aws:s3:::fischer-csc346-download/examples/Old-Main-Milky.jpg
Entity tag (Etag)	aaa3e5256dc5cc6b9758bd36331499f0
Object URL	https://fischer-csc346-download.s3.amazonaws.com/examples/Old-Main-Milky.jpg

The Object URL is highlighted with a blue box. At the bottom of the console, there is a search bar with the text 'sia-non' and search options: Highlight All, Match Case, Match Diacritics, Whole Words, and 1 of 1 match.

<https://fischer-csc346-download.s3.amazonaws.com/examples/Old-Main-Milky.jpg>

AWS S3

Cloud Object Storage

- S3 underpins much of AWS
- Docker images in ECR are stored in S3 under the hood
- All CloudFormation templates you upload are stored in an S3 bucket
- All EC2 AMI images are stored in S3
- It is a really important service!

AWS S3

Too many features to go over in class

- Storage tiers - save money if you accept more risk
- Lifecycle Policies - Delete stuff after a while, or transition it to archive storage
- Integrates with many other Services - Event Based Triggers
- Cross-account access - Host files that others can use
- Requestor-pays - Host files that others have to pay to download (they don't pay you, they pay the AWS S3 network costs)
- Yes, you have to pay to read data out of S3, that's where they getcha!

AWS Lambda

Function as a Service?

AWS Lambda

Managed Code Execution

- Up to this point, if we had code we needed to execute, it had to run on a machine we managed.
 - Laptop
 - EC2
- AWS Lambda introduces another model

AWS Lambda

Managed Code Execution

“Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging.”

<https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>

AWS Lambda

Advantages

- Serverless - No infrastructure to manage
- Event-Driven - Nothing is “always running” (this can be good and bad)
- Pricing based only on what you use
- Scales automatically (can have limits placed)
- Can be massively parallelized
- Lets you focus on just your core application logic

AWS Lambda

Disadvantages

- Not for long-running processes. A given Lambda invocation cannot last longer than 15 minutes.
- Requires a different mental model for how you build an application.
 - Micro-services vs monolithic services.
- Vendor lock-in. Can't really take your AWS Lambda functions to Google App Engine.
- Memory and CPU limits are not as high as dedicated EC2 instances.
- Access to persistent file systems is not simple.

AWS Lambda

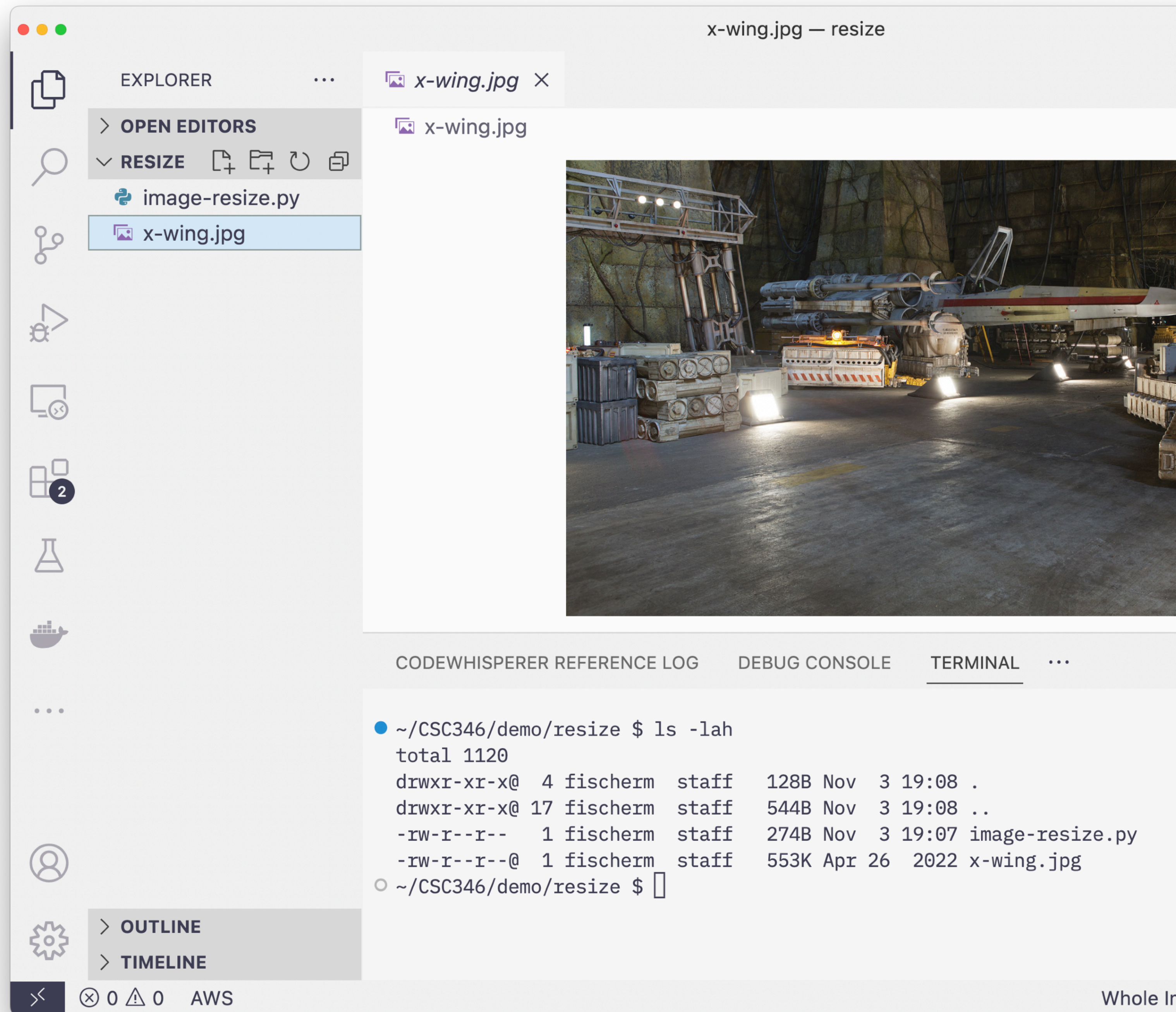
Image Resizing

- Let's add images to our app.
- Images are uploaded of all sorts of various sizes.
- In the posts list, we want the images to all be a uniform size.
- We want to normalize any uploaded image to be a set of standard sizes, a square thumbnail and a larger view, but still possibly smaller than the original image.

AWS Lambda

Image Resizing in Python

- How do we resize an image in Python?
- Use the Pillow / PIL module



EXPLORER

> OPEN EDITORS

RESIZE

image-resize.py

x-wing-200.jpg

x-wing.jpg

image-resize.py ×

image-resize.py > ...

```
1  from PIL import Image
2
3
4  def resize_image(image_path, resized_path, size):
5      with Image.open(image_path) as image:
6          image.thumbnail((size, size))
7          image.save(resized_path)
8          print(f"Resized {image_path} to {size}px")
9
10
11  source = "x-wing.jpg"
12  resized = "x-wing-200.jpg"
13
14  resize_image(source, resized, 200)
15
```

CODEWHISPERER REFERENCE LOG

DEBUG CONSOLE

TERMINAL

bash - resize + -

```
● ~/CSC346/demo/resize $ ls -lah
total 1120
drwxr-xr-x@ 4 fischer staff 128B Nov  3 19:08 .
drwxr-xr-x@ 17 fischer staff 544B Nov  3 19:08 ..
-rw-r--r--  1 fischer staff 274B Nov  3 19:07 image-resize.py
-rw-r--r--@ 1 fischer staff 553K Apr 26 2022 x-wing.jpg
● ~/CSC346/demo/resize $ python image-resize.py
Resized x-wing.jpg to 200px
○ ~/CSC346/demo/resize $
```

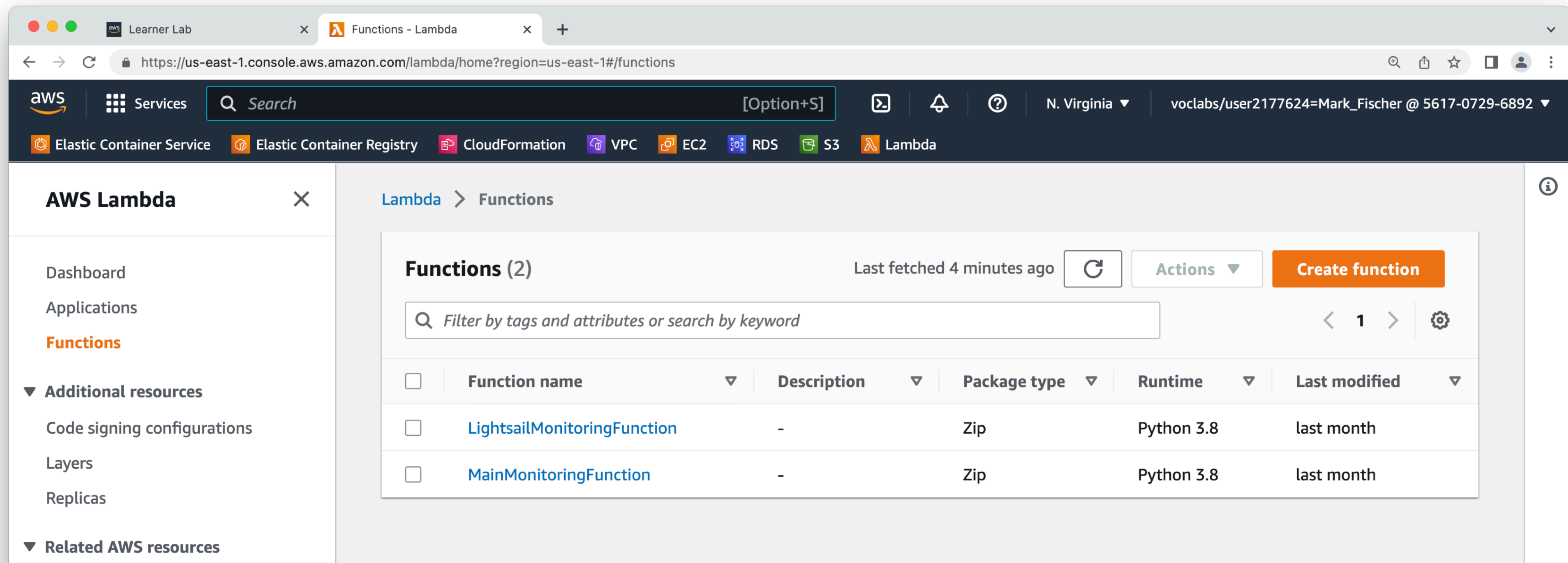
> OUTLINE

> TIMELINE

AWS Lambda

Image Resizing in the Cloud

- That's all fine for a laptop, how do we do this in the cloud?
- AWS Lambda Console - Search for Lambda



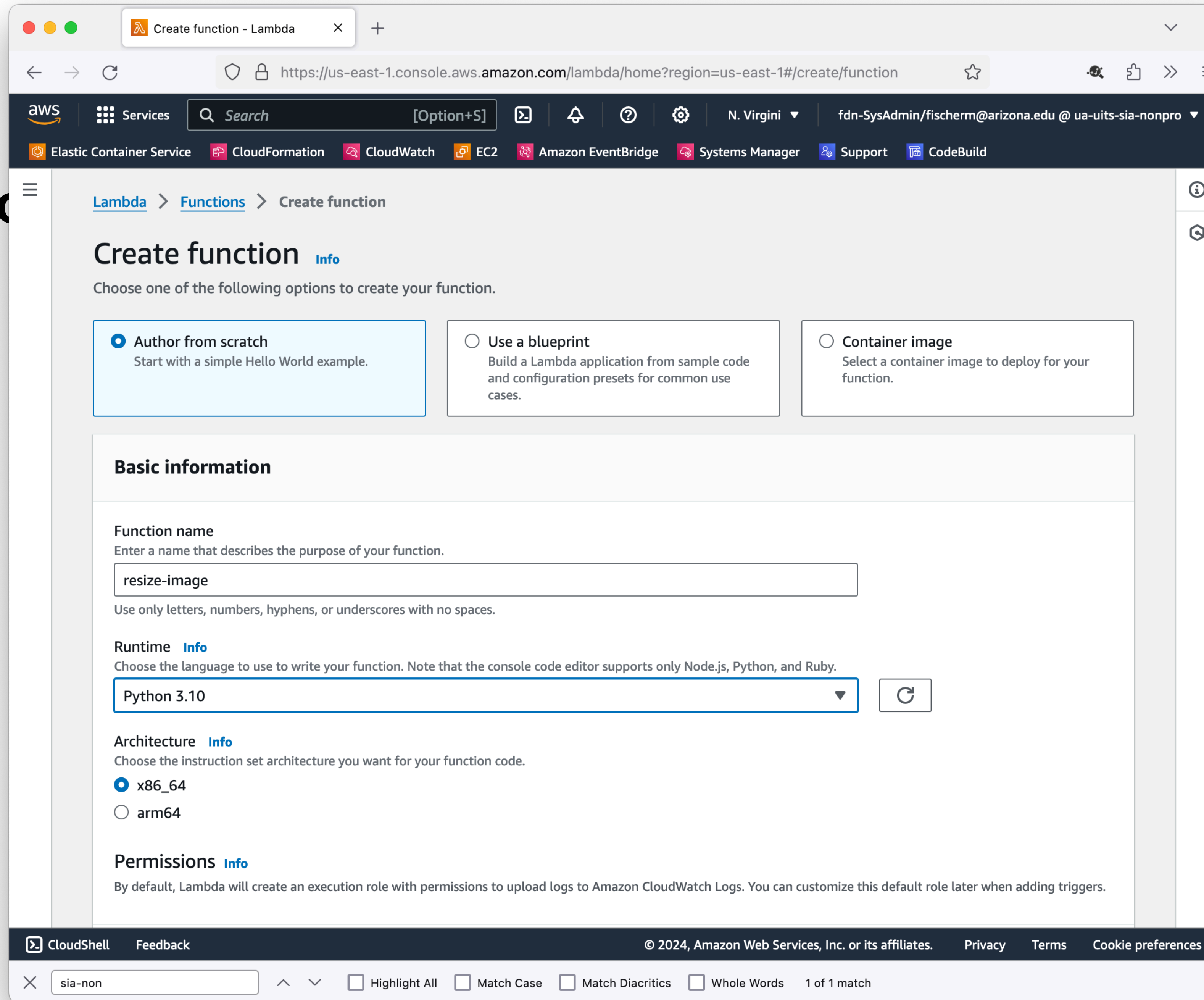
The screenshot shows the AWS Lambda console interface. The top navigation bar includes the AWS logo, a search bar, and the current region (N. Virginia) and user profile (voclabs/user2177624=Mark_Fischer @ 5617-0729-6892). Below the navigation bar, there are several service icons including Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, RDS, S3, and Lambda. The main content area is titled "AWS Lambda" and "Functions (2)". It displays a table of functions with columns for Function name, Description, Package type, Runtime, and Last modified. Two functions are listed: LightsailMonitoringFunction and MainMonitoringFunction, both using Zip package type and Python 3.8 runtime.

Function name	Description	Package type	Runtime	Last modified
LightsailMonitoringFunction	-	Zip	Python 3.8	last month
MainMonitoringFunction	-	Zip	Python 3.8	last month

AWS Lambda

Image Resizing in the Cloud

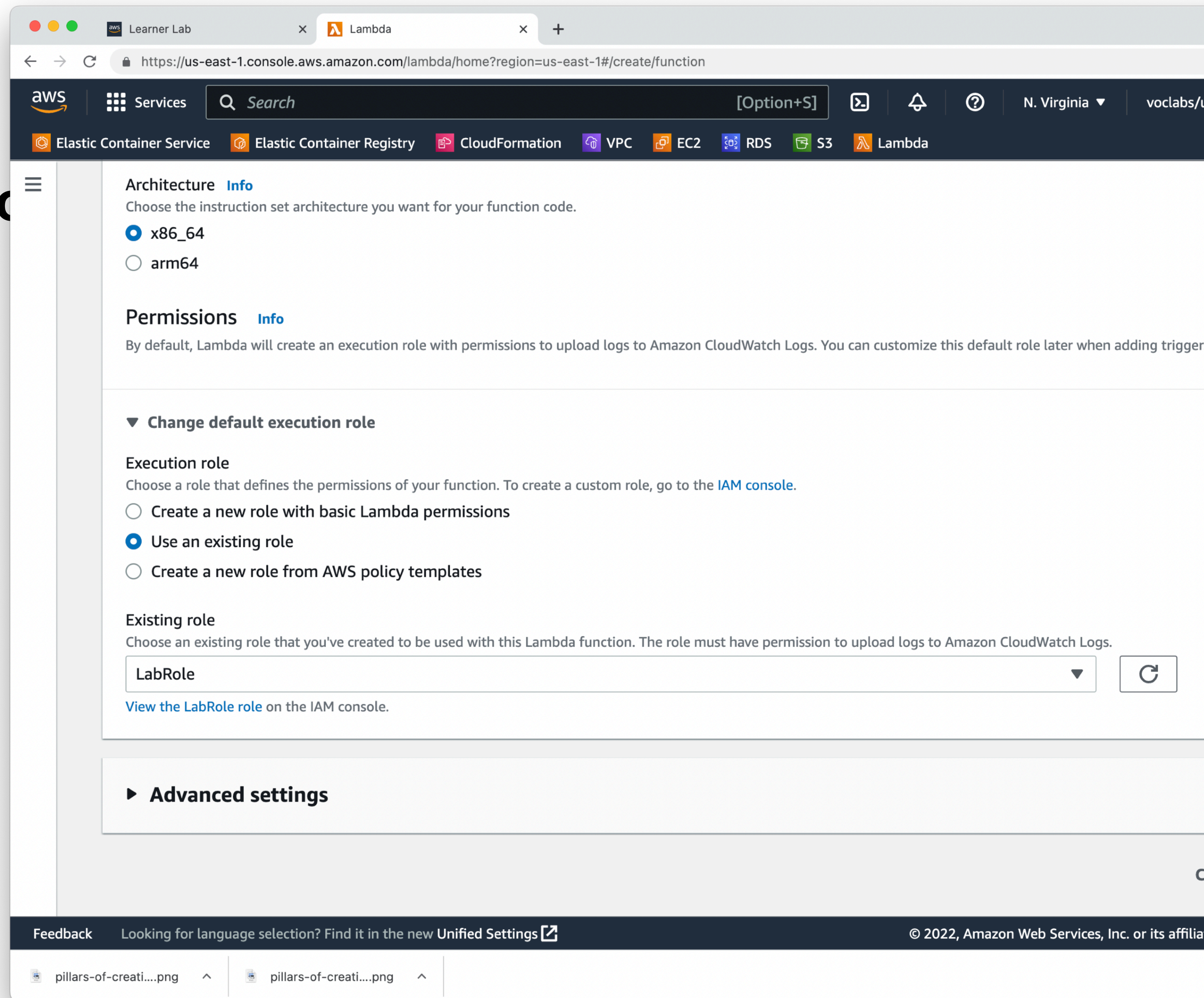
- Create a new function
- Give it a name
- Use python 3.10 for the runtime
- x86_64



AWS Lambda

Image Resizing in the Cloud

- Change the default execution role
- We can't make new IAM roles in the Academy account
- Use the existing "LabRole"



AWS Lambda

Image Resizing in the Cloud

- Default “Hello World” function

The screenshot displays the AWS Lambda console for a function named 'resize-image'. The top navigation bar includes the AWS logo, a search bar, and various service icons like Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, RDS, S3, and Lambda. The function name 'resize-image' is prominently displayed at the top of the console, along with buttons for 'Throttle', 'Copy ARN', and 'Actions'. Below this, the 'Function overview' section shows the function icon, name, and 'Layers (0)'. There are buttons for '+ Add trigger' and '+ Add destination'. On the right side, a metadata panel lists 'Description' (empty), 'Last modified' (1 minute ago), 'Function ARN' (arn:aws:lambda:us-east-1:561707296892:function:resize-image), and 'Function URL' (empty). Below the overview, a horizontal menu allows switching between 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code source' section is active, showing a code editor with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for 'Test' and 'Deploy'. The code editor displays a Python lambda handler function:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')}
8
9
```

The environment pane on the left shows the file structure for the function, including 'lambda_function.py'.

AWS Lambda

Event Handler

- We mentioned that Lambda is event driven
- Your code runs inside of the Lambda Runtime
- The Lambda Runtime handles receipt of events, then calls your code and passes the event to it
- The entry point to your code is your event handler function

AWS Lambda Event Handler

Runtime settings [Info](#)

Runtime: Python 3.10

Handler: [Info](#)
lambda_function.lambda_handler

Code source [Info](#) Upload from ▾

File Edit Find View Go Tools Window **Test** Deploy

Go to Anything (⌘ P)

Environment

- resize-image - /
- lambda_function.py**

```
1 import boto3
2 import requests
3 from PIL import Image
4
5
6 def resize_image(image_path, resized_path, size):
7     with Image.open(image_path) as image:
8         image.thumbnail((size, size))
9         image.save(resized_path)
10    print(f"Resized {image_path} to {size}px")
11
12
13 def lambda_handler(event, context):
14     filename = "pillars-of-creation.png"
15     source_url = f"https://fischerm-csc346-upload-bucket.s3.amazonaws.com/input/{filename}"
16
17     tmpkey = filename.replace("/", "")
```

AWS Lambda

Event Triggers

- So what is in an event?
- It's largely dependent on what is triggering your Lambda Function
- So what can trigger Lambda?
 - In short, a lot of things!
- Most basic trigger is direct invocation. Either in the web console, or with the API

```
aws lambda invoke --function-name resize-image --payload '{ "file": "x-wing.jpg" }'
```

AWS Lambda

Event Triggers

- Lambda integrates with more than 140 AWS services via direct integration and the Amazon EventBridge event bus.
- Commonly used Lambda event sources:
 - API Gateway
 - SNS
 - SQS
 - S3
 - CloudWatch Logs
 - CloudWatch Events
 - DynamoDB
 - EventBridge
 - Kinesis Data Streams
 - Step Functions

AWS Lambda

Event Triggers

- Each event source will send different bits of data in the incoming event object.
- Here is a sample event coming from API Gateway
- Data relevant to an incoming HTTP REST call

```
{
  "resource": "/",
  "path": "/",
  "httpMethod": "GET",
  "requestContext": {
    "resourcePath": "/",
    "httpMethod": "GET",
    "path": "/Prod/",
    ...
  },
  "headers": {
    "accept": "text/html,application/signed-exe",
    "accept-encoding": "gzip, deflate, br",
    "Host": "70ixmpl4fl.execute-api.us-east-2.a",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0",
    ...
  },
  "queryStringParameters": null,
  "multiValueQueryStringParameters": null,
  "pathParameters": null,
  "stageVariables": null,
  "body": null,
  "isBase64Encoded": false
}
```

AWS Lambda

Event Triggers

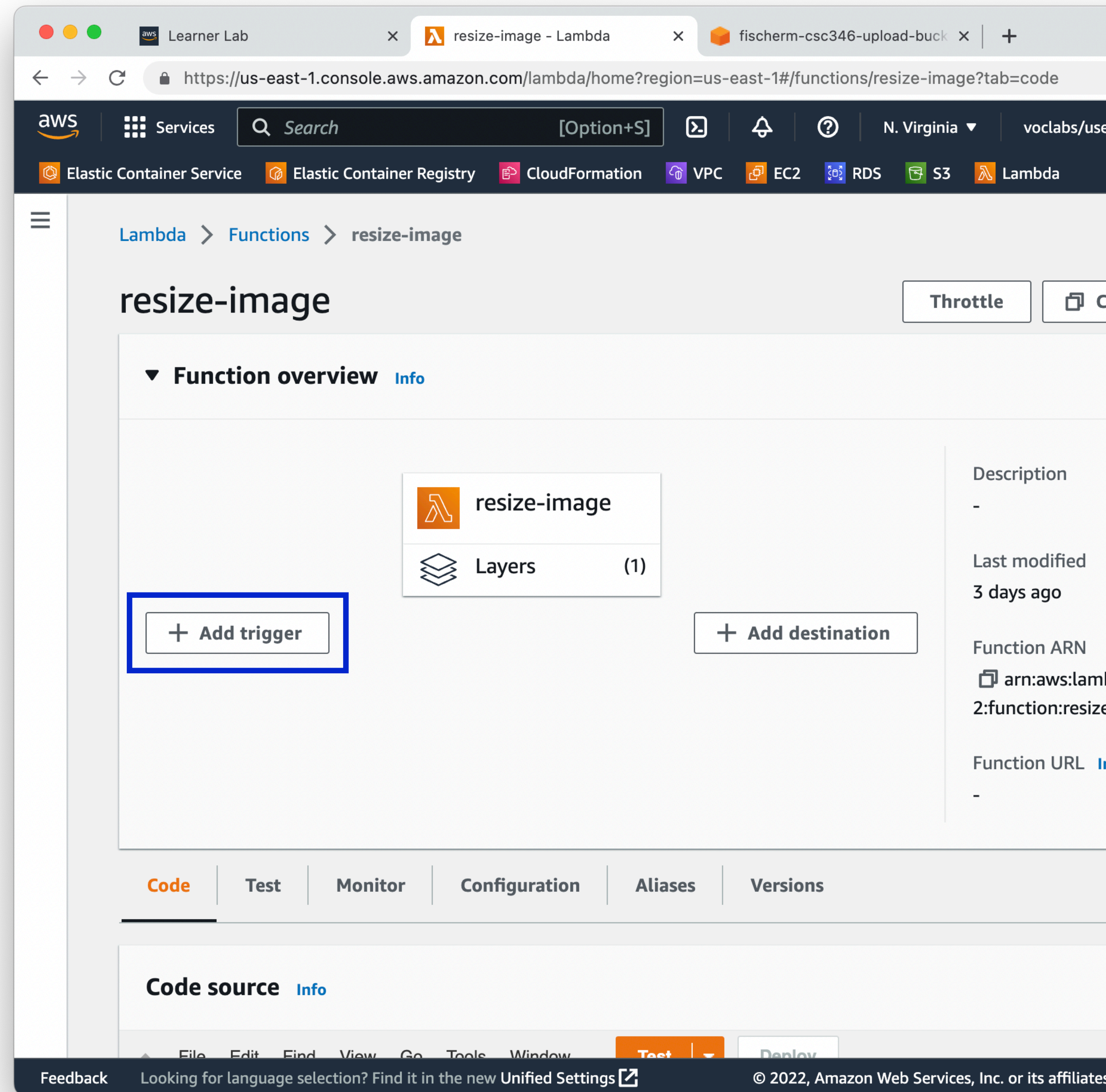
- Here's an example of an S3 ObjectCreated:Put event
- Information about which bucket the object was created in as well as the object itself
- Note that the Records key in the top level dictionary is an array. This event may contain multiple objects

```
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2022-11-06T20:17:18.352Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "AWS:AROAYFSC5FB6KLKFWGIO0:user2177624=Ma
      },
      "requestParameters": {"sourceIPAddress": "67.1.196.37"},
      "responseElements": {
        "x-amz-request-id": "VV31VSKAKPTP7R4C",
        "x-amz-id-2": "v+A+vGX30SW08cb8JhbAj7wRPmtDLn1dgYtZQOf9Z
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "e2733ff1-399f-4645-8778-7e4fef7a7c3a
        "bucket": {
          "name": "fischer-csc346-upload-bucket",
          "ownerIdentity": {"principalId": "A3NRT1KH8KAG57"},
          "arn": "arn:aws:s3:::fischer-csc346-upload-bucket",
        },
        "object": {
          "key": "input/x-wing.jpg",
          "size": 566695,
          "eTag": "09a9b11f91823dd69fefc3ecbd9f7e9c",
          "sequencer": "006368164E4491ED05",
        },
      },
    },
  ]
}
```


AWS Lambda

Event Triggers

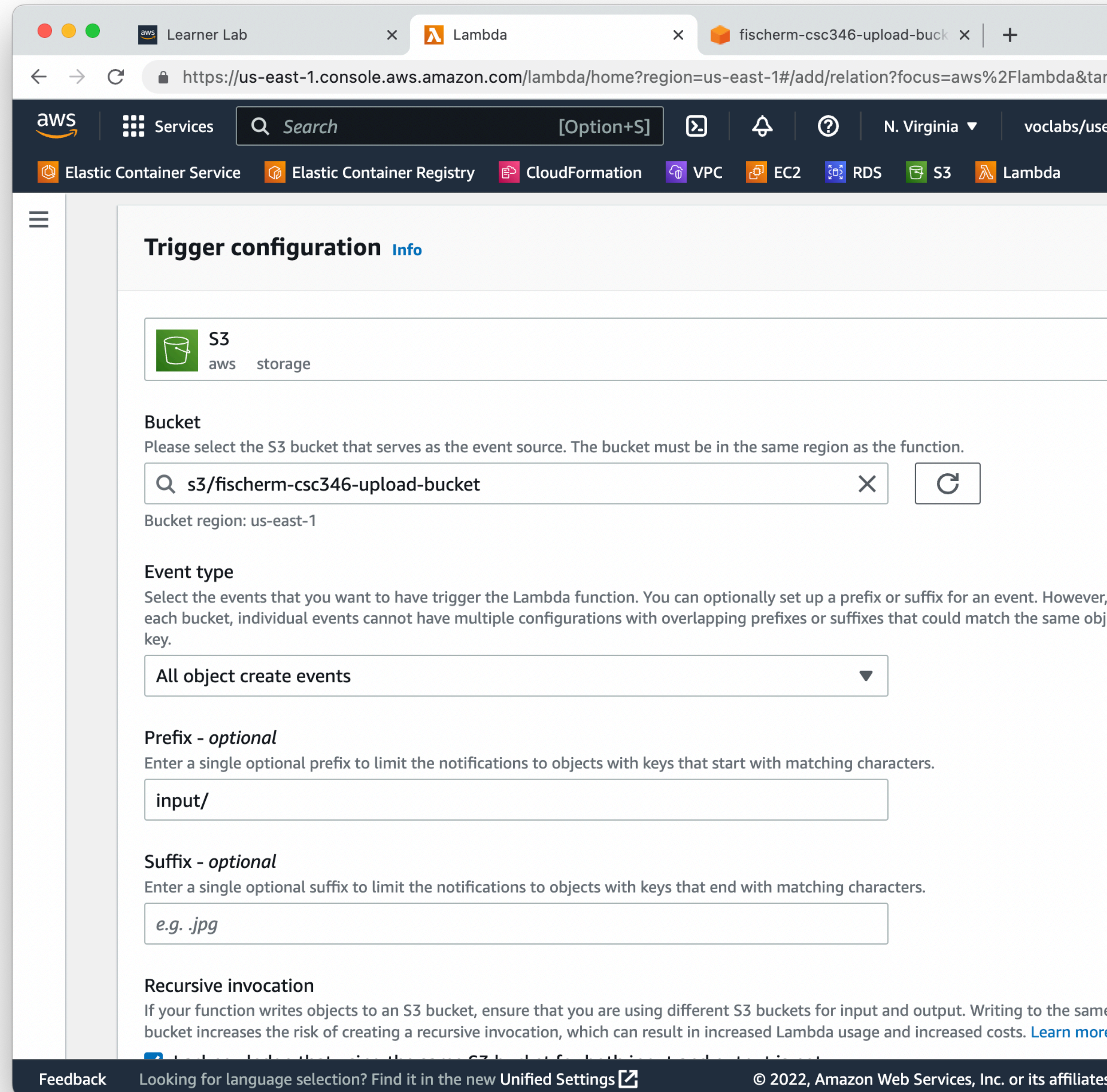
- In the Lambda console, click “Add trigger”



AWS Lambda

Event Triggers

- Choose S3 as the event source
- Select the S3 bucket you want
- We'll trigger on all the “CreateObject” events
- I only want to trigger on objects with keys beginning with “input/”
- Be careful about recursive triggering!!



The screenshot shows the AWS Lambda console interface. The browser tabs include "Learner Lab", "Lambda", and "fischer-csc346-upload-buck". The URL is "https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/add/relation?focus=aws%2Flambda&tar". The navigation bar shows "Services" and a search bar. The main content area is titled "Trigger configuration Info" and shows the following configuration:

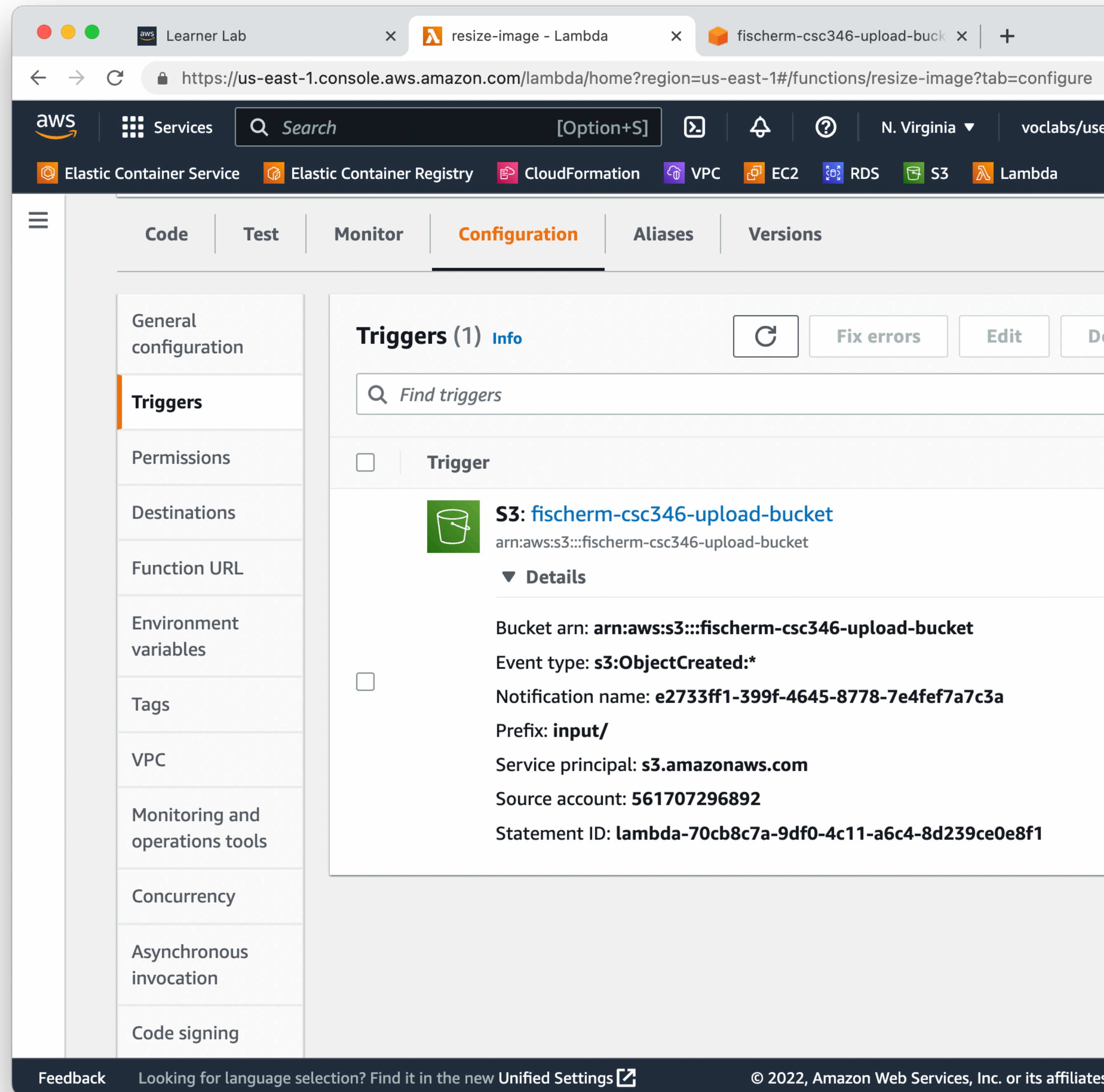
- S3** (aws storage)
- Bucket:** s3/fischer-csc346-upload-bucket (Bucket region: us-east-1)
- Event type:** All object create events
- Prefix - optional:** input/
- Suffix - optional:** e.g. .jpg
- Recursive invocation:** If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

At the bottom, there is a "Feedback" link and a note: "Looking for language selection? Find it in the new Unified Settings". The footer text reads: "© 2022, Amazon Web Services, Inc. or its affiliates".

AWS Lambda

Event Triggers

- Once saved, you can see the trigger configuration in the “Configuration” tab of your function
- Now every time a new object is created in the input folder of that bucket, our Lambda function will run!



AWS Lambda

Layers

- How do we import all the various python modules, such as the Pillow/PIL module?
- Lambda supports the idea of shared layers.
- I've created a layer which has all the dependencies built in.
- Layers aren't too hard to create, but we don't have enough time to go into that in class unfortunately.
- Only available in the same region, so use `us-east-1`

```
arn:aws:lambda:us-east-1:269800669561:layer:fischem-csc346-image:2
```

AWS Lambda Layers

- Scroll down
- Click the “Add a layer” button

The screenshot shows the AWS Lambda console for a function named 'resize-image'. The interface includes a navigation bar with the AWS logo, a search bar, and various service icons. The main content area is divided into several sections:

- Code properties:** Displays package size (299.0 byte), SHA256 hash (f106ZIRH/KN6Ra3twvdRllUYaxv182Tjx0qNWNlKihl=), and last modified date (November 3, 2022 at 07:30 PM MST).
- Runtime settings:** Shows runtime (Python 3.9), handler (lambda_function.lambda_handler), and architecture (x86_64).
- Layers:** This section is highlighted with a blue box. It contains an 'Edit' button and an 'Add a layer' button, which is also highlighted with a blue box. Below the buttons is a table with columns for Merge order, Name, Layer version, Compatible runtimes, Compatible architectures, and Version ARN. The table currently displays 'There is no data to display.'

The footer of the console shows the URL: <https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/add/layer?function=resize-image>, copyright information for Amazon Web Services, and links for Privacy, Terms, and Cookie preferences.

AWS Lambda Layers

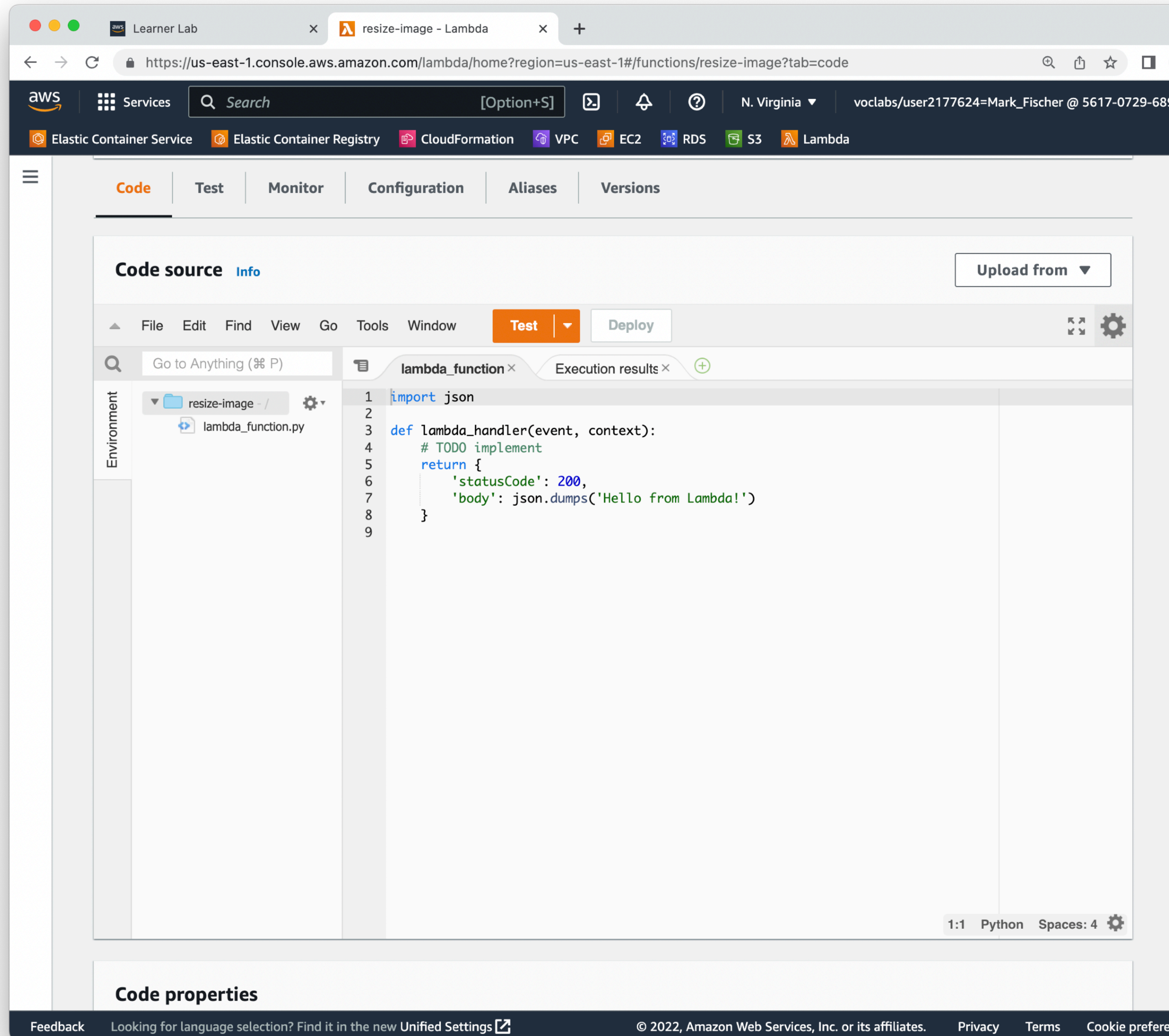
- Specify an ARN
- Use my layer ARN
- Click the Verify button to make sure things are working
- Click the Add button

The screenshot shows the AWS Lambda console interface for adding a layer. The page title is "Add layer". Under "Function runtime settings", the runtime is "Python 3.9" and the architecture is "x86_64". The "Choose a layer" section has three options: "AWS layers", "Custom layers", and "Specify an ARN". The "Specify an ARN" option is selected. Below this, there is a text input field containing the ARN "arn:aws:lambda:us-east-1:561707296892:layer:csc346-lambda-layer:3" and a "Verify" button. The description of the layer is "Base Layer for CSC 346 Lambda Functions" and the compatible runtime is "x86_64". At the bottom right, there are "Cancel" and "Add" buttons.

```
arn:aws:lambda:us-east-1:269800669561:layer:fischerm-csc346-image:2
```

AWS Lambda Layers

- We can edit the code directly in the browser to start.
- Works for simple functions.
- OK for testing.
- You'll want to have more Infrastructure as Code scaffolding around any real project.



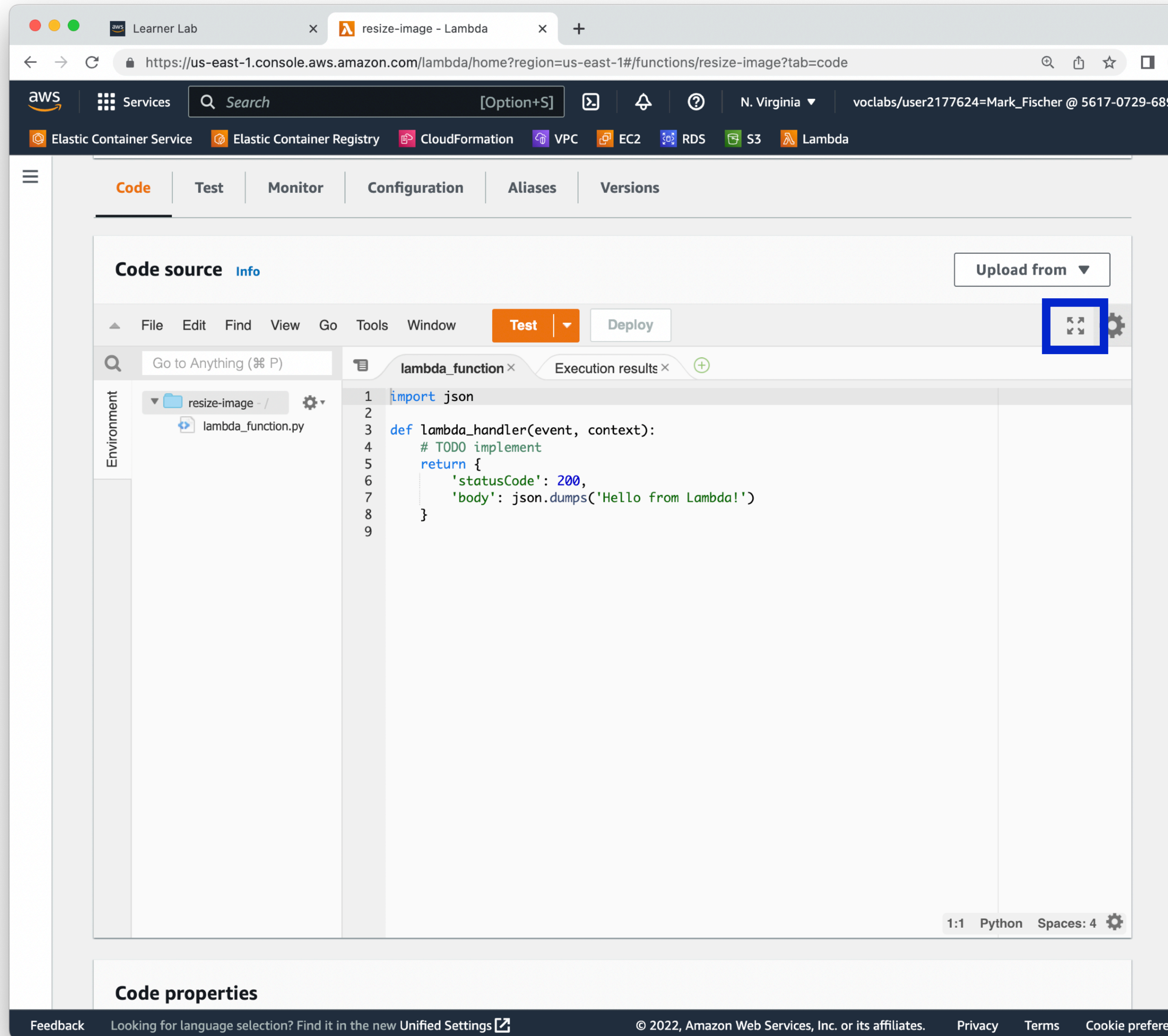
The screenshot shows the AWS Lambda console interface for a function named 'resize-image'. The browser address bar shows the URL: `https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/resize-image?tab=code`. The console header includes the AWS logo, 'Services', a search bar, and navigation icons for Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, RDS, S3, and Lambda. The main navigation bar has tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is active, showing a code editor with the following Python code:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
```

The editor includes a file explorer on the left showing the 'resize-image' folder with a file named 'lambda_function.py'. The top of the editor has a menu bar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', and 'Window', along with 'Test' and 'Deploy' buttons. The bottom of the editor shows '1:1 Python Spaces: 4'. Below the code editor is a 'Code properties' section. The footer of the console includes 'Feedback', a link for 'Looking for language selection? Find it in the new Unified Settings', and copyright information: '© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

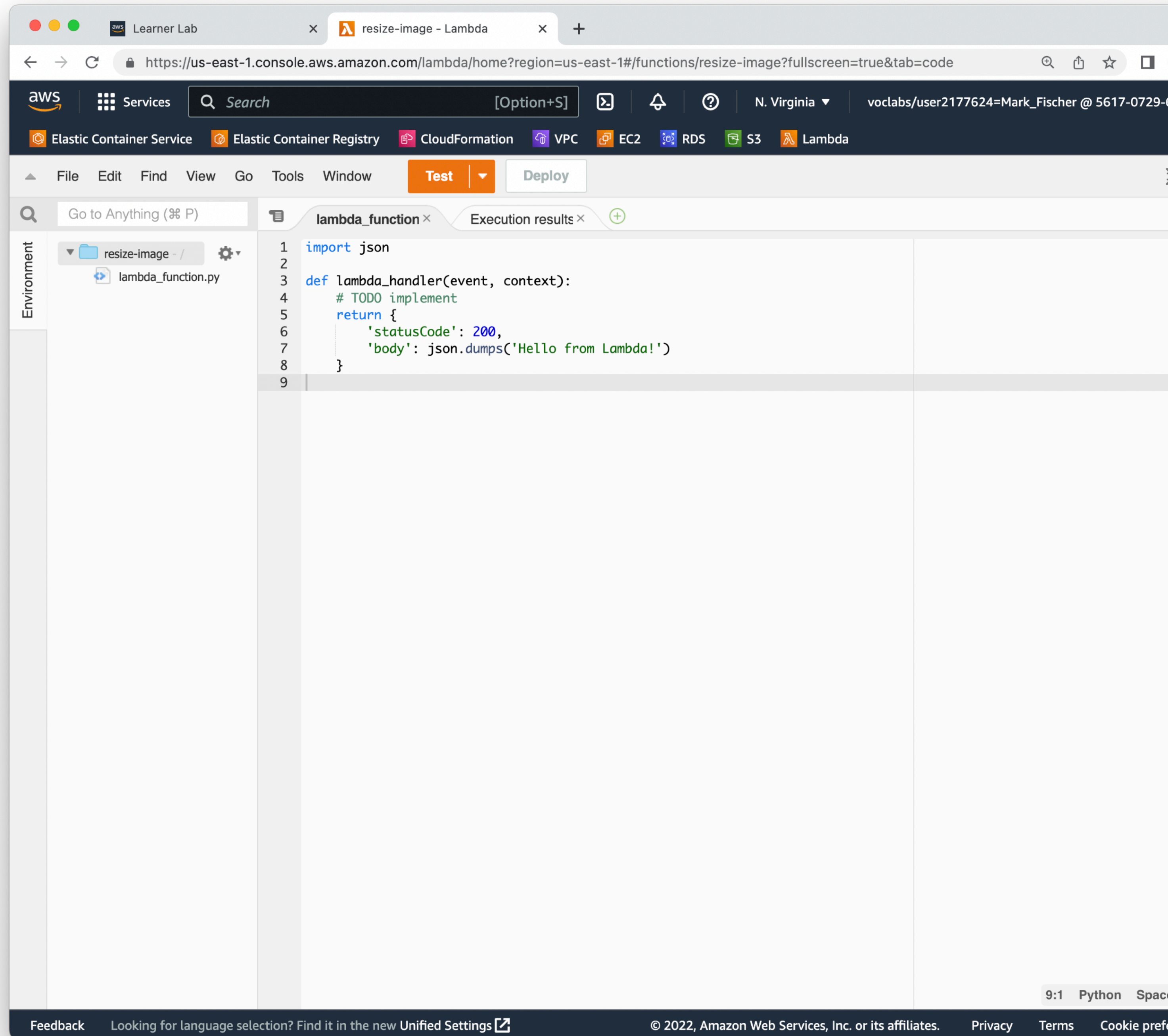
AWS Lambda Layers

- Can make the code editor fill the browser window



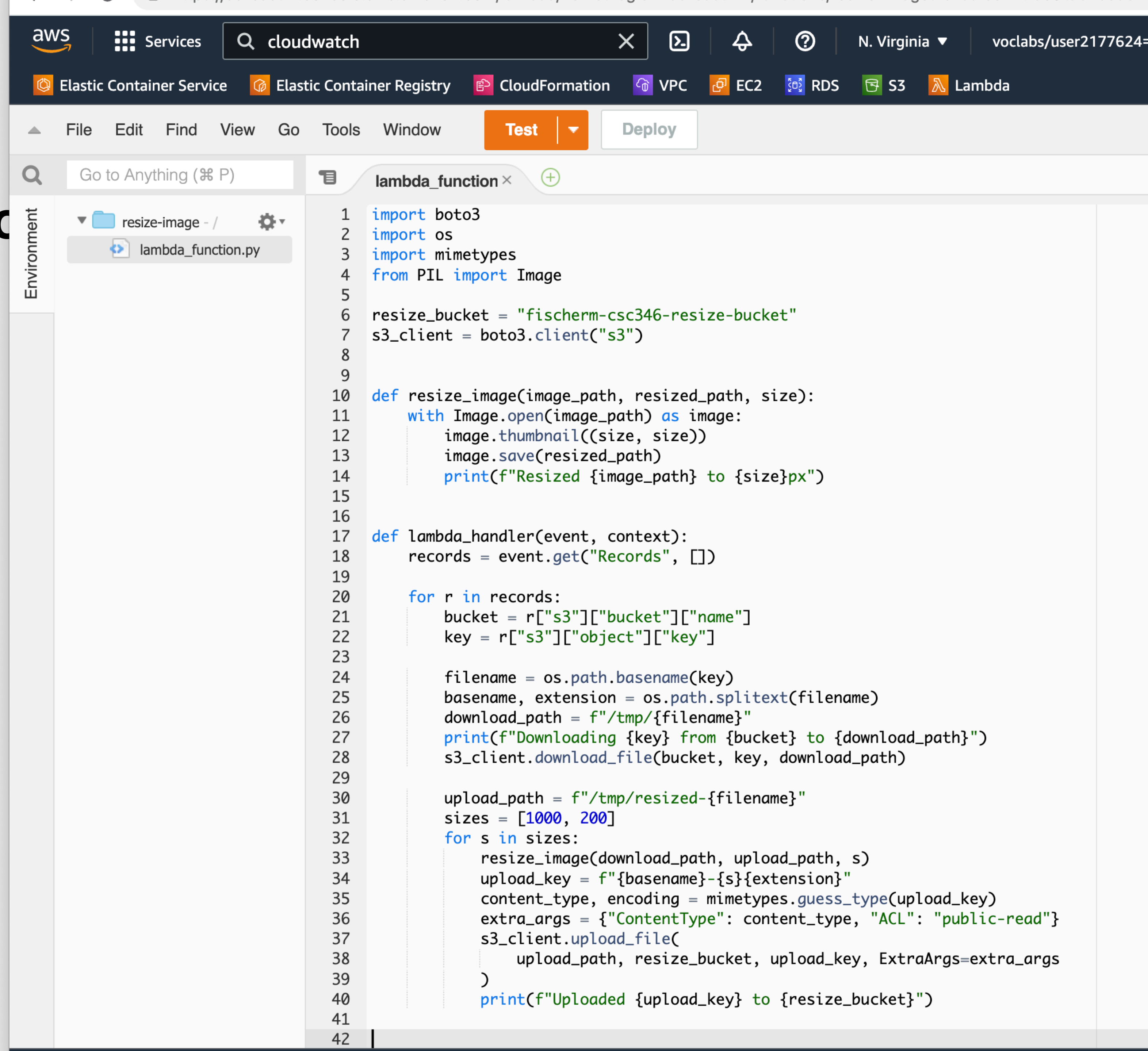
AWS Lambda Layers

- Can make the code editor fill the browser window



AWS Lambda Image Resizing in the Cloud

- Where are our files?
- The Lambda runtime has access to some temporary local storage
- We need to get the file to resize from the event when a new object is added to the bucket

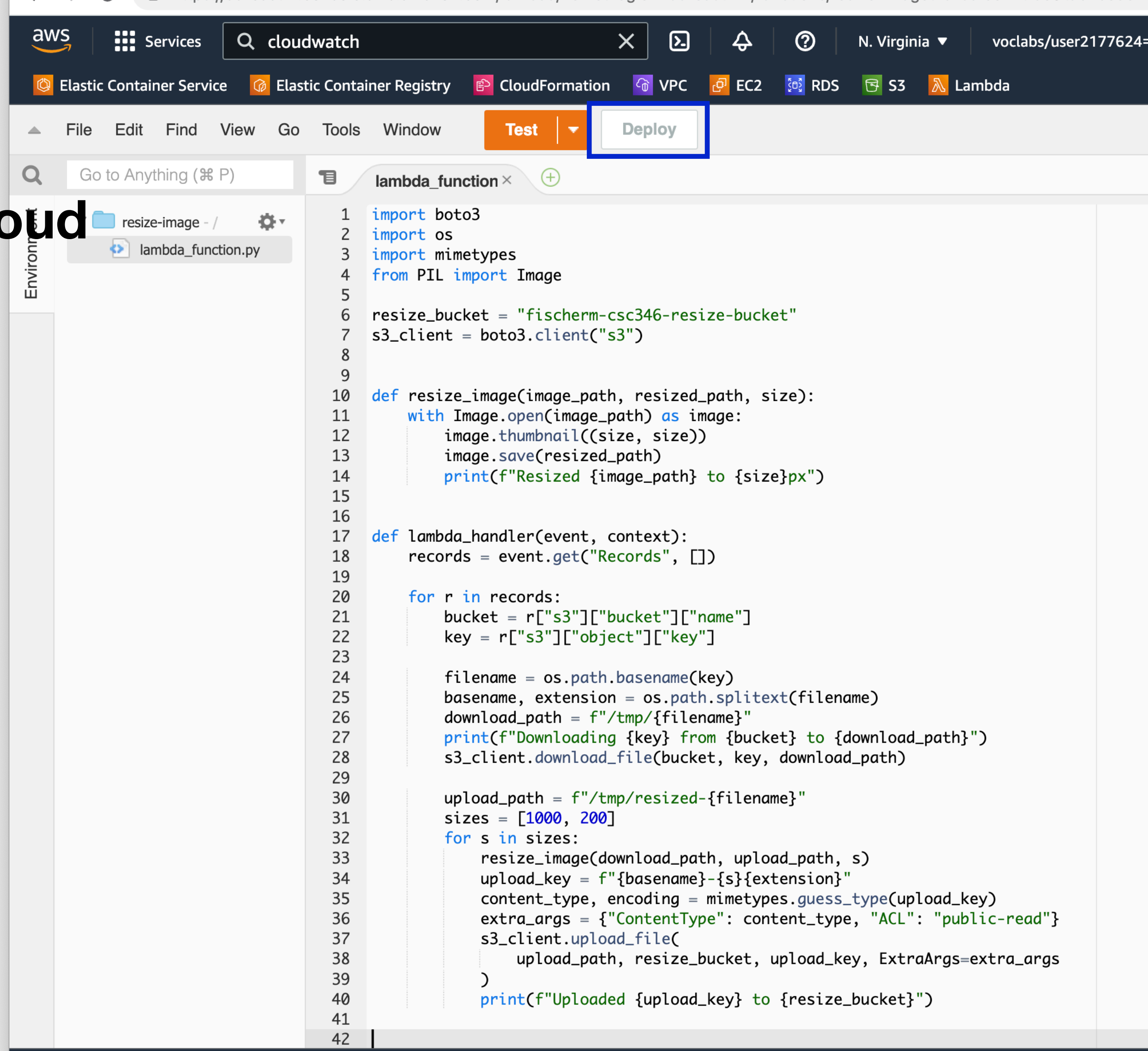


The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, 'Services' menu, a search bar containing 'cloudwatch', and various service icons like Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, RDS, S3, and Lambda. Below the navigation bar is a menu with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', and buttons for 'Test' and 'Deploy'. The main area is split into two panes. The left pane, titled 'Environment', shows a file explorer view with a folder 'resize-image - /' containing a file 'lambda_function.py'. The right pane shows the Python code for the lambda function, with line numbers 1 through 42. The code imports boto3, os, and mimetypes, and uses PIL for image processing. It defines a 'resize_image' function and a 'lambda_handler' function that processes records from an S3 bucket, downloading images, resizing them, and uploading the results back to the bucket.

```
1 import boto3
2 import os
3 import mimetypes
4 from PIL import Image
5
6 resize_bucket = "fischem-csc346-resize-bucket"
7 s3_client = boto3.client("s3")
8
9
10 def resize_image(image_path, resized_path, size):
11     with Image.open(image_path) as image:
12         image.thumbnail((size, size))
13         image.save(resized_path)
14         print(f"Resized {image_path} to {size}px")
15
16
17 def lambda_handler(event, context):
18     records = event.get("Records", [])
19
20     for r in records:
21         bucket = r["s3"]["bucket"]["name"]
22         key = r["s3"]["object"]["key"]
23
24         filename = os.path.basename(key)
25         basename, extension = os.path.splitext(filename)
26         download_path = f"/tmp/{filename}"
27         print(f"Downloading {key} from {bucket} to {download_path}")
28         s3_client.download_file(bucket, key, download_path)
29
30         upload_path = f"/tmp/resized-{filename}"
31         sizes = [1000, 200]
32         for s in sizes:
33             resize_image(download_path, upload_path, s)
34             upload_key = f"{basename}-{s}{extension}"
35             content_type, encoding = mimetypes.guess_type(upload_key)
36             extra_args = {"ContentType": content_type, "ACL": "public-read"}
37             s3_client.upload_file(
38                 upload_path, resize_bucket, upload_key, ExtraArgs=extra_args
39             )
40             print(f"Uploaded {upload_key} to {resize_bucket}")
41
42
```

AWS Lambda Image Resizing in the Cloud

- Function needs to be Deployed before testing.



```
1 import boto3
2 import os
3 import mimetypes
4 from PIL import Image
5
6 resize_bucket = "fischem-csc346-resize-bucket"
7 s3_client = boto3.client("s3")
8
9
10 def resize_image(image_path, resized_path, size):
11     with Image.open(image_path) as image:
12         image.thumbnail((size, size))
13         image.save(resized_path)
14         print(f"Resized {image_path} to {size}px")
15
16
17 def lambda_handler(event, context):
18     records = event.get("Records", [])
19
20     for r in records:
21         bucket = r["s3"]["bucket"]["name"]
22         key = r["s3"]["object"]["key"]
23
24         filename = os.path.basename(key)
25         basename, extension = os.path.splitext(filename)
26         download_path = f"/tmp/{filename}"
27         print(f"Downloading {key} from {bucket} to {download_path}")
28         s3_client.download_file(bucket, key, download_path)
29
30         upload_path = f"/tmp/resized-{filename}"
31         sizes = [1000, 200]
32         for s in sizes:
33             resize_image(download_path, upload_path, s)
34             upload_key = f"{basename}-{s}{extension}"
35             content_type, encoding = mimetypes.guess_type(upload_key)
36             extra_args = {"ContentType": content_type, "ACL": "public-read"}
37             s3_client.upload_file(
38                 upload_path, resize_bucket, upload_key, ExtraArgs=extra_args
39             )
40             print(f"Uploaded {upload_key} to {resize_bucket}")
41
42
```

AWS Lambda Image Resizing in the Cloud

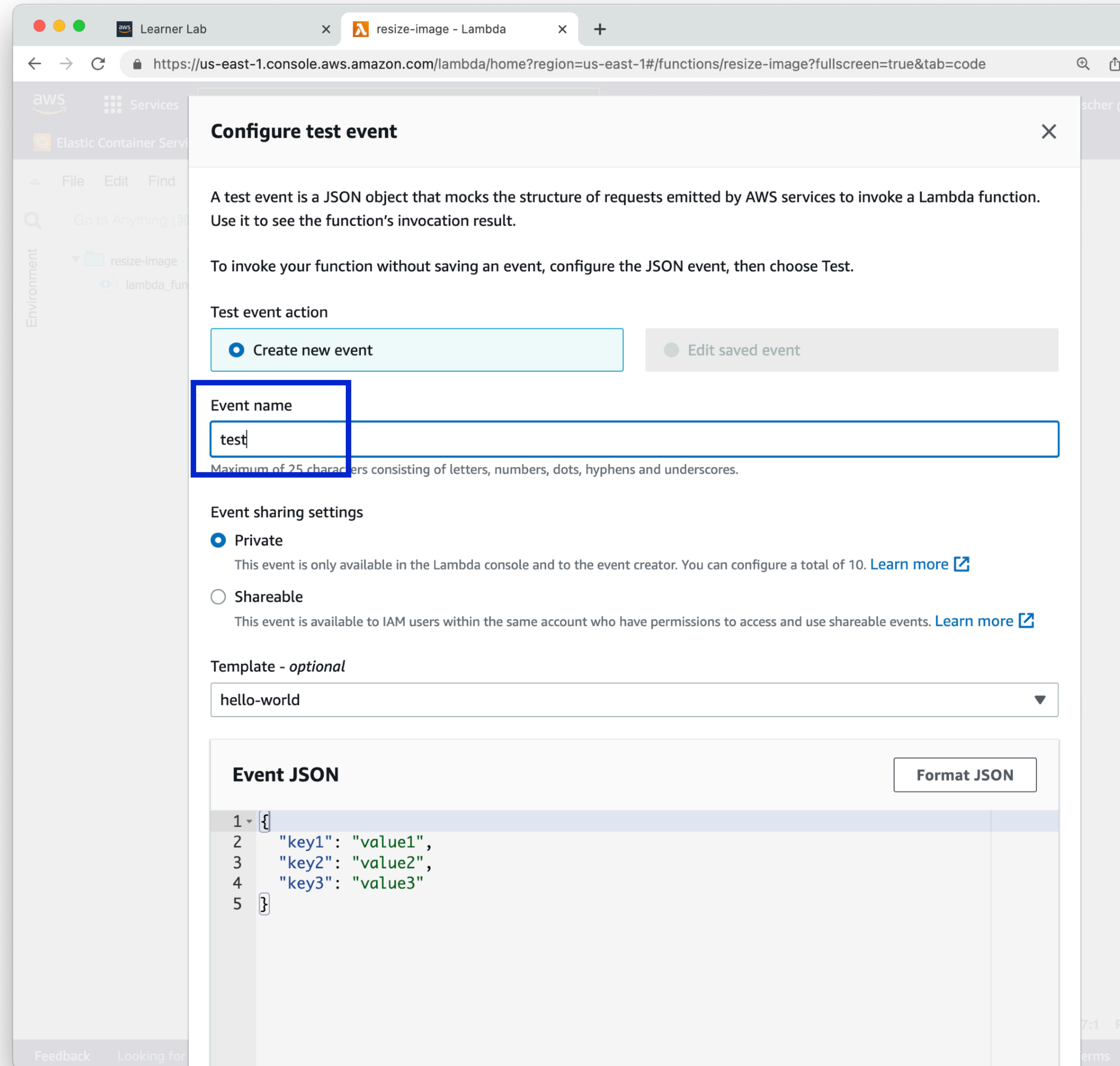
- Once deployed, we can Test

```
1 import boto3
2 import os
3 import mimetypes
4 from PIL import Image
5
6 resize_bucket = "fischem-csc346-resize-bucket"
7 s3_client = boto3.client("s3")
8
9
10 def resize_image(image_path, resized_path, size):
11     with Image.open(image_path) as image:
12         image.thumbnail((size, size))
13         image.save(resized_path)
14         print(f"Resized {image_path} to {size}px")
15
16
17 def lambda_handler(event, context):
18     records = event.get("Records", [])
19
20     for r in records:
21         bucket = r["s3"]["bucket"]["name"]
22         key = r["s3"]["object"]["key"]
23
24         filename = os.path.basename(key)
25         basename, extension = os.path.splitext(filename)
26         download_path = f"/tmp/{filename}"
27         print(f"Downloading {key} from {bucket} to {download_path}")
28         s3_client.download_file(bucket, key, download_path)
29
30         upload_path = f"/tmp/resized-{filename}"
31         sizes = [1000, 200]
32         for s in sizes:
33             resize_image(download_path, upload_path, s)
34             upload_key = f"{basename}-{s}{extension}"
35             content_type, encoding = mimetypes.guess_type(upload_key)
36             extra_args = {"ContentType": content_type, "ACL": "public-read"}
37             s3_client.upload_file(
38                 upload_path, resize_bucket, upload_key, ExtraArgs=extra_args
39             )
40             print(f"Uploaded {upload_key} to {resize_bucket}")
41
42
```

AWS Lambda

Image Resizing in the Cloud

- The first time we hit Test, we're prompted to define a Test Event
- Lambda is Event Driven
- Our function currently doesn't use the event at all, so the default "hello-world" event is fine
- Give it an Event name
- Scroll down and Save



The screenshot shows the AWS Lambda console interface for configuring a test event. The browser address bar indicates the URL: `https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/resize-image?fullscreen=true&tab=code`. The main heading is "Configure test event". Below the heading, there is a descriptive text: "A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result." and a note: "To invoke your function without saving an event, configure the JSON event, then choose Test." Under "Test event action", there are two buttons: "Create new event" (selected) and "Edit saved event". The "Event name" field is highlighted with a blue box and contains the text "test". Below this field, a note states: "Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores." Under "Event sharing settings", there are two radio buttons: "Private" (selected) and "Shareable". Below "Private", there is a note: "This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)". Below "Shareable", there is a note: "This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)". Under "Template - optional", there is a dropdown menu with "hello-world" selected. At the bottom, there is a section titled "Event JSON" with a "Format JSON" button. The JSON content is as follows:

```
1 - {
2   "key1": "value1",
3   "key2": "value2",
4   "key3": "value3"
5 }
```

AWS Lambda Image Resizing in the C

- Try testing again
- Error!
- Task timed out after 3 seconds?
- Lambda functions can last *up to 15 minutes*, but default to 3 seconds.

The screenshot shows the AWS Lambda console interface. The browser address bar displays the URL: `https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/resize-image?fullscreen=true&tab=code`. The console header includes the AWS logo, a search bar, and navigation links for various services like Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, RDS, S3, and Lambda. The main content area shows the 'Execution result' for a test event. The status is 'Failed'. The response is a JSON object: `{ "errorMessage": "2022-11-04T03:30:42.791Z d1f34153-8462-4956-bb40-74458cd40aa0 Task timed out after" }`. The function logs show: `START RequestId: d1f34153-8462-4956-bb40-74458cd40aa0 Version: $LATEST 2022-11-04T03:30:42.791Z d1f34153-8462-4956-bb40-74458cd40aa0 Task timed out after 3.05 seconds` and `END RequestId: d1f34153-8462-4956-bb40-74458cd40aa0`. The report shows: `REPORT RequestId: d1f34153-8462-4956-bb40-74458cd40aa0 Duration: 3047.66 ms Billed Duration: 3000`. The request ID is `d1f34153-8462-4956-bb40-74458cd40aa0`.

AWS Lambda

Image Resizing in the Cloud

- Memory size is also tied to CPU allocation. Let's raise the memory limit to 1024, that gives us more CPU and our function will run faster
- Change the Timeout to 1 minute.
- Save

The screenshot shows the AWS Lambda console interface. The browser address bar displays the URL: `https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/resize-image/edit/basic-settings?tab=configure`. The page title is "Edit basic settings".

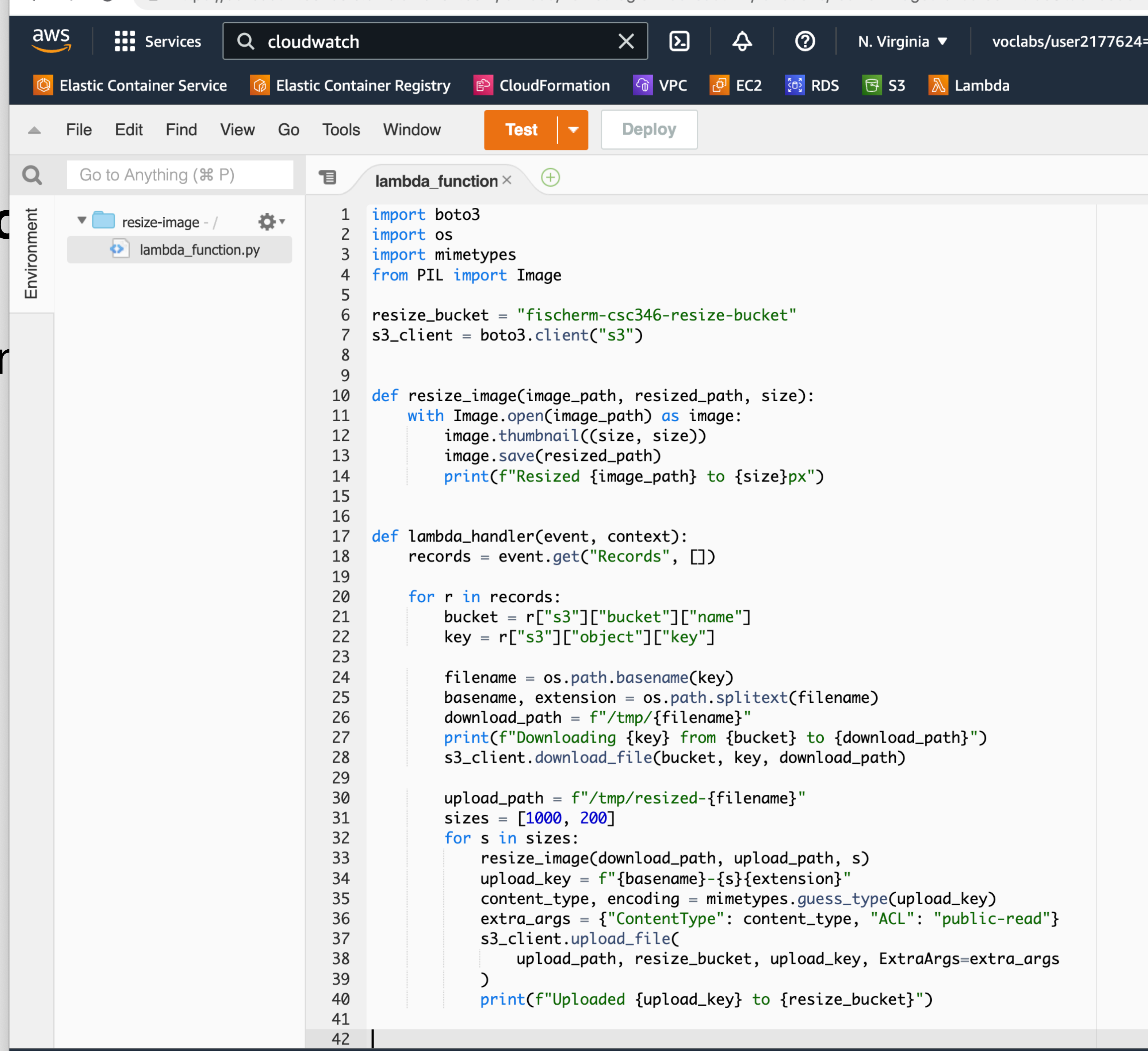
The "Basic settings" section includes the following fields:

- Description - optional:** An empty text input field.
- Memory:** A text input field containing "1024" followed by "MB". Below it, a note states: "Your function is allocated CPU proportional to the memory configured." and "Set memory to between 128 MB and 10240 MB".
- Ephemeral storage:** A text input field containing "512" followed by "MB". Below it, a note states: "You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)" and "Set ephemeral storage (/tmp) to between 512 MB and 10240 MB".
- Timeout:** Two input fields: "1" for minutes and "0" for seconds.
- Execution role:** Radio buttons for "Use an existing role" (selected) and "Create a new role from AWS policy templates".
- Existing role:** A dropdown menu showing "LabRole" and a refresh button. Below it, a link says "View the LabRole role on the IAM console."

At the bottom right of the page, there are "Cancel" and "Save" buttons.

AWS Lambda Image Resizing in the Cloud

- Go back to the code view and let's try our test again

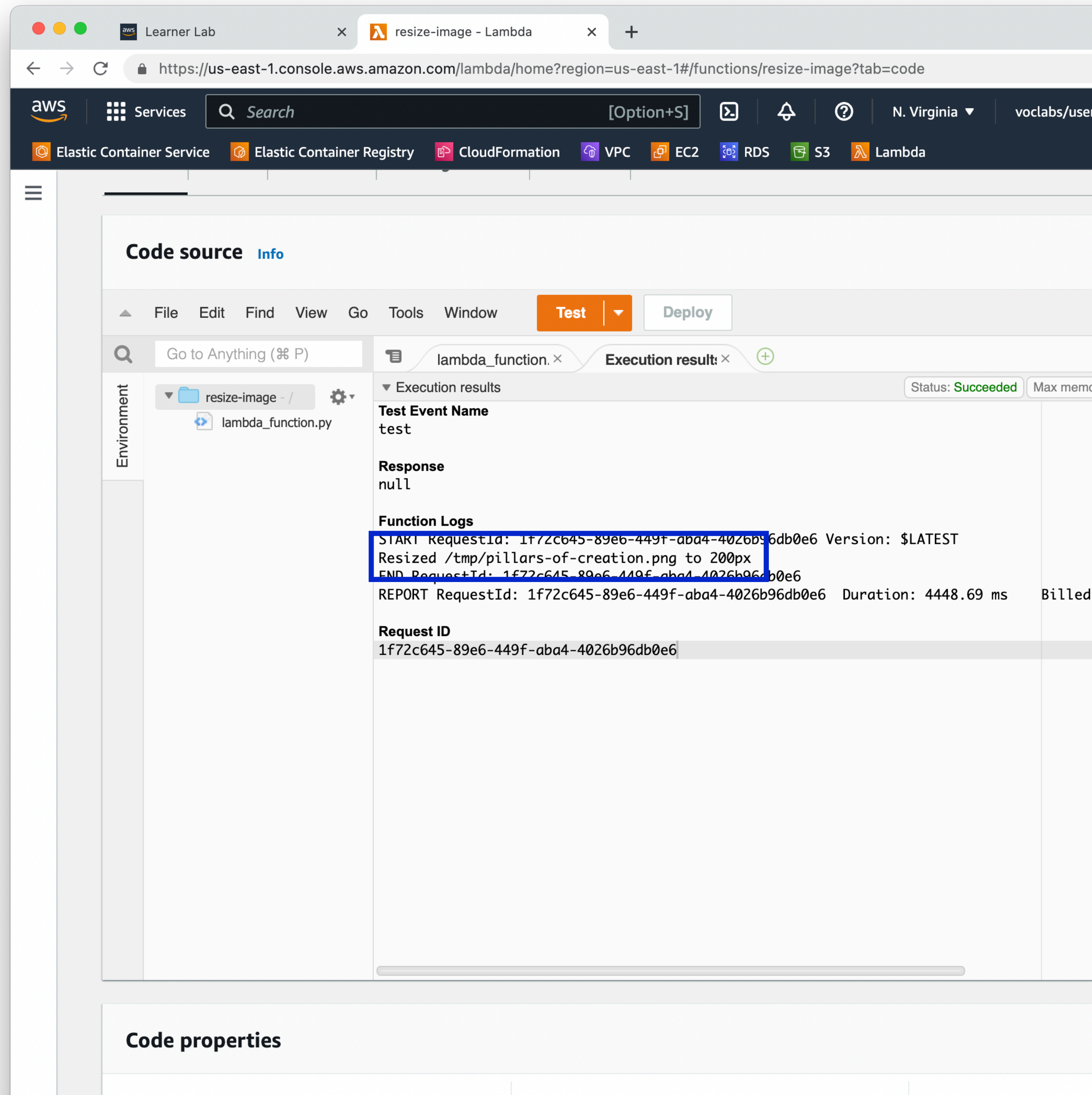


```
aws Services cloudwatch N. Virginia voclabs/user2177624=
Elastic Container Service Elastic Container Registry CloudFormation VPC EC2 RDS S3 Lambda
File Edit Find View Go Tools Window Test Deploy
Go to Anything (⌘ P)
Environment
  resize-image - /
  lambda_function.py
1 import boto3
2 import os
3 import mimetypes
4 from PIL import Image
5
6 resize_bucket = "fischem-csc346-resize-bucket"
7 s3_client = boto3.client("s3")
8
9
10 def resize_image(image_path, resized_path, size):
11     with Image.open(image_path) as image:
12         image.thumbnail((size, size))
13         image.save(resized_path)
14         print(f"Resized {image_path} to {size}px")
15
16
17 def lambda_handler(event, context):
18     records = event.get("Records", [])
19
20     for r in records:
21         bucket = r["s3"]["bucket"]["name"]
22         key = r["s3"]["object"]["key"]
23
24         filename = os.path.basename(key)
25         basename, extension = os.path.splitext(filename)
26         download_path = f"/tmp/{filename}"
27         print(f"Downloading {key} from {bucket} to {download_path}")
28         s3_client.download_file(bucket, key, download_path)
29
30         upload_path = f"/tmp/resized-{filename}"
31         sizes = [1000, 200]
32         for s in sizes:
33             resize_image(download_path, upload_path, s)
34             upload_key = f"{basename}-{s}{extension}"
35             content_type, encoding = mimetypes.guess_type(upload_key)
36             extra_args = {"ContentType": content_type, "ACL": "public-read"}
37             s3_client.upload_file(
38                 upload_path, resize_bucket, upload_key, ExtraArgs=extra_args
39             )
40             print(f"Uploaded {upload_key} to {resize_bucket}")
41
42
```


AWS Lambda

Image Resizing in the Cloud

- No errors!
- We see our resize message.
- We have to copy our resized image somewhere
- Let's put it into an S3 bucket!
- Recommended to use different buckets for input and output to protect against recursive triggering of your function



The screenshot displays the AWS Lambda console interface for a function named 'resize-image'. The top navigation bar includes the AWS logo, 'Services', a search bar, and the region 'N. Virginia'. The main content area shows the 'Code source' tab, which includes a file explorer with 'lambda_function.py' and a 'Test' button. The 'Execution results' section shows a successful test event with a null response. The function logs are visible, showing the start and end of the execution, the request ID, and the message 'Resized /tmp/pillars-of-creation.png to 200px'. The request ID is highlighted in blue. The bottom section shows 'Code properties'.

AWS Lambda S3 Demo

AWS Lambda

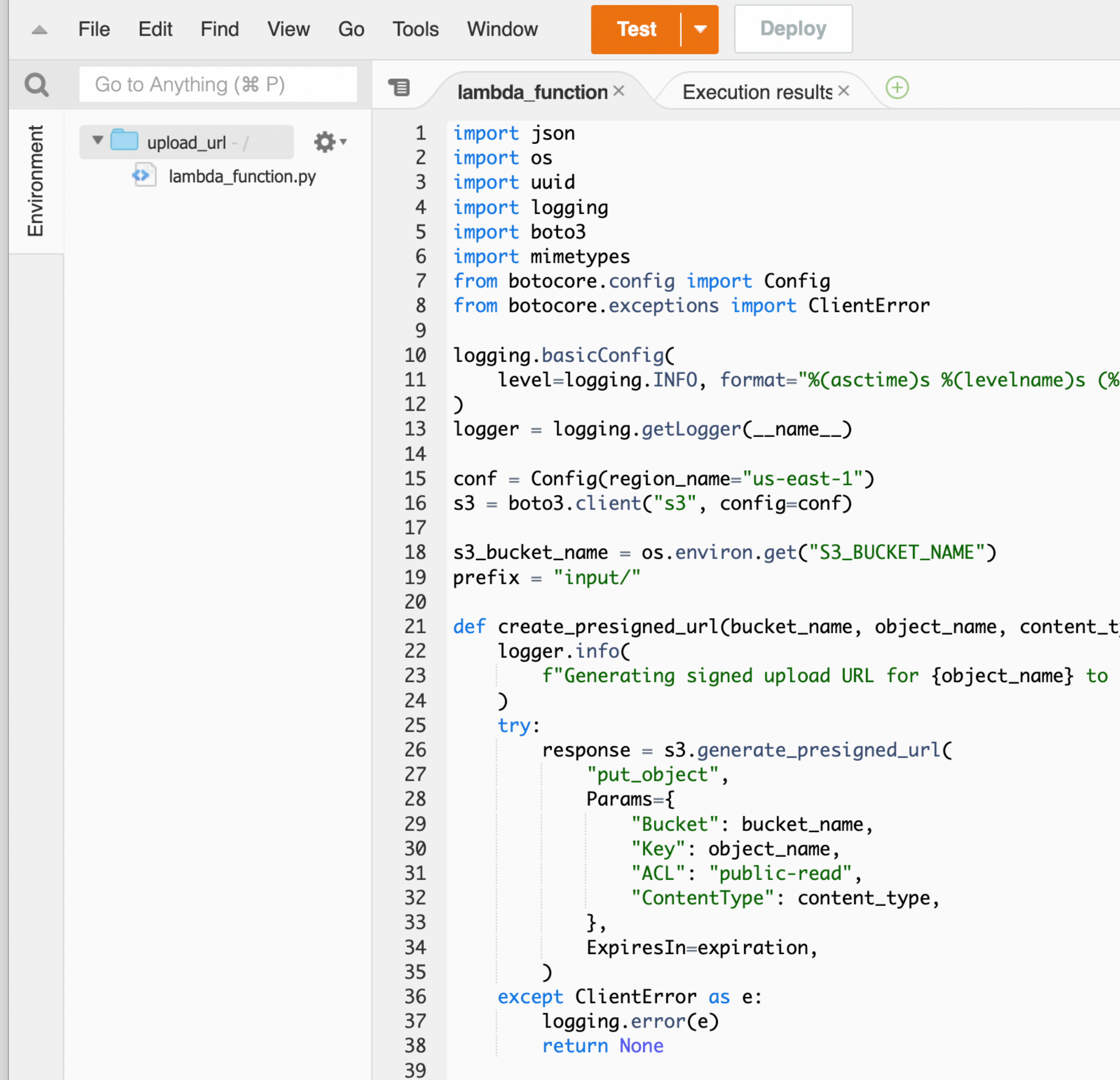
Upload Images

- How do we get our Chat client app to upload an image to our S3 bucket?
- AWS API calls!
- AWS S3 API provides a way to craft a ‘signed’ URL which we can use as the basis for a `PUT` or `POST` HTTP call to upload data directly to a bucket

AWS Lambda

Upload Images

- Using the `boto3` SDK we can create an `s3_client` object and use the `generate_presigned_url` method
- Get the bucket name from an Environment Variable



The screenshot shows an IDE window with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for 'Test' and 'Deploy'. The search bar contains 'Go to Anything (⌘ P)'. The environment sidebar shows a folder 'upload_url - /' and a file 'lambda_function.py'. The main editor displays the following Python code:

```
1 import json
2 import os
3 import uuid
4 import logging
5 import boto3
6 import mimetypes
7 from botocore.config import Config
8 from botocore.exceptions import ClientError
9
10 logging.basicConfig(
11     level=logging.INFO, format="%(asctime)s %(levelname)s (%
12 )
13 logger = logging.getLogger(__name__)
14
15 conf = Config(region_name="us-east-1")
16 s3 = boto3.client("s3", config=conf)
17
18 s3_bucket_name = os.environ.get("S3_BUCKET_NAME")
19 prefix = "input/"
20
21 def create_presigned_url(bucket_name, object_name, content_t
22     logger.info(
23         f"Generating signed upload URL for {object_name} to
24     )
25     try:
26         response = s3.generate_presigned_url(
27             "put_object",
28             Params={
29                 "Bucket": bucket_name,
30                 "Key": object_name,
31                 "ACL": "public-read",
32                 "ContentType": content_type,
33             },
34             ExpiresIn=expiration,
35         )
36     except ClientError as e:
37         logging.error(e)
38     return None
39
```

AWS Lambda

Environment Variables

- Just like almost every other code execution method, Lambda provides a way to define Environment Variables

The screenshot shows the AWS Lambda console interface for a function named 'generate-url'. The 'Configuration' tab is selected, and the 'Environment variables' section is expanded. The console displays the following details:

- Function Name:** generate-url
- Layers:** (1)
- Related functions:** Select a function
- Function ARN:** arn:aws:lambda:us-east-1:fdn-SysAdmin/fischer-csc346-GenerateURL
- Application:** fischer-csc346
- Function URL:** Info

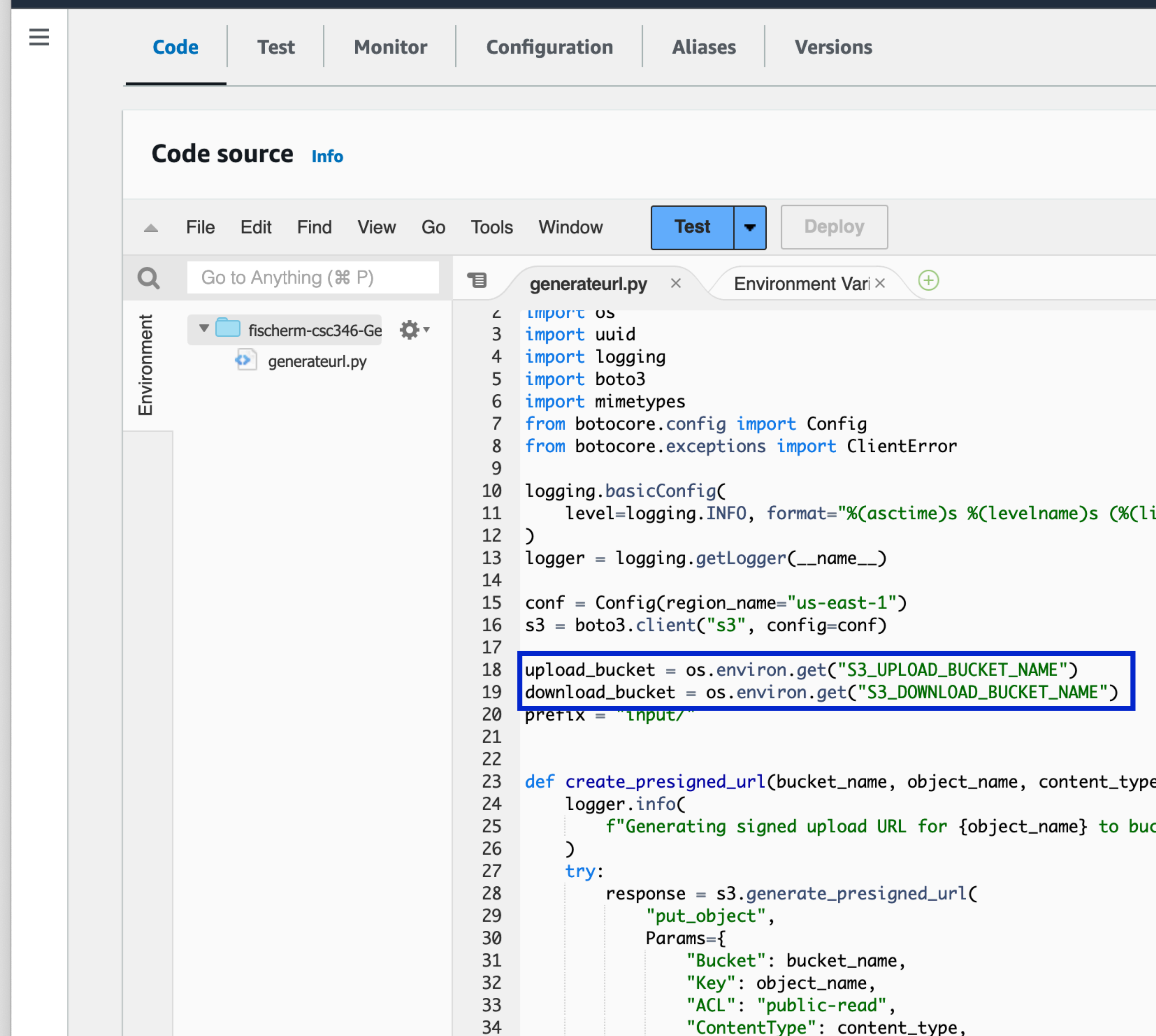
The 'Environment variables (2)' section shows the following variables:

Key	Value
S3_DOWNLOAD_BUCKET_NAME	fischer-csc346-download
S3_UPLOAD_BUCKET_NAME	fischer-csc346-upload

AWS Lambda

Environment Variables

- These are accessible from your code using standard language functions for accessing environment variables
- Code can be used in multiple runtime environments without having to know the specifics of the runtime



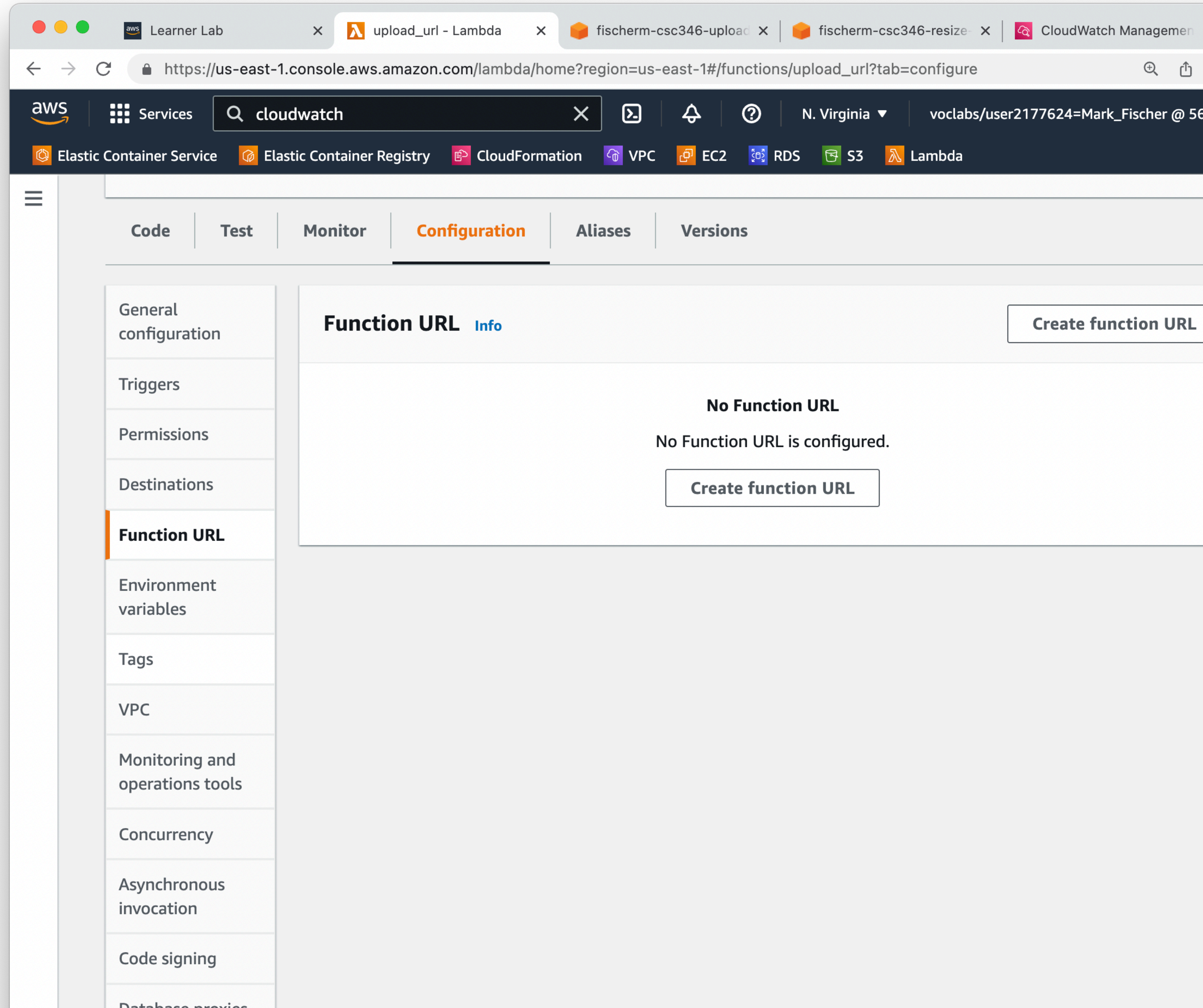
The screenshot shows the AWS Lambda console interface. At the top, there are tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The 'Code source' tab is active, displaying a code editor for a file named 'generateurl.py'. The editor has a menu bar with File, Edit, Find, View, Go, Tools, and Window. Below the menu bar, there are buttons for 'Test' and 'Deploy'. The code editor shows the following Python code:

```
1 import os
2 import uuid
3 import logging
4 import boto3
5 import mimetypes
6 from botocore.config import Config
7 from botocore.exceptions import ClientError
8
9
10 logging.basicConfig(
11     level=logging.INFO, format="%(asctime)s %(levelname)s %(li
12 )
13 logger = logging.getLogger(__name__)
14
15 conf = Config(region_name="us-east-1")
16 s3 = boto3.client("s3", config=conf)
17
18 upload_bucket = os.environ.get("S3_UPLOAD_BUCKET_NAME")
19 download_bucket = os.environ.get("S3_DOWNLOAD_BUCKET_NAME")
20 prefix = "input/"
21
22
23 def create_presigned_url(bucket_name, object_name, content_type
24     logger.info(
25         f"Generating signed upload URL for {object_name} to buc
26     )
27     try:
28         response = s3.generate_presigned_url(
29             "put_object",
30             Params={
31                 "Bucket": bucket_name,
32                 "Key": object_name,
33                 "ACL": "public-read",
34                 "ContentType": content_type,
```

AWS Lambda

Function URLs

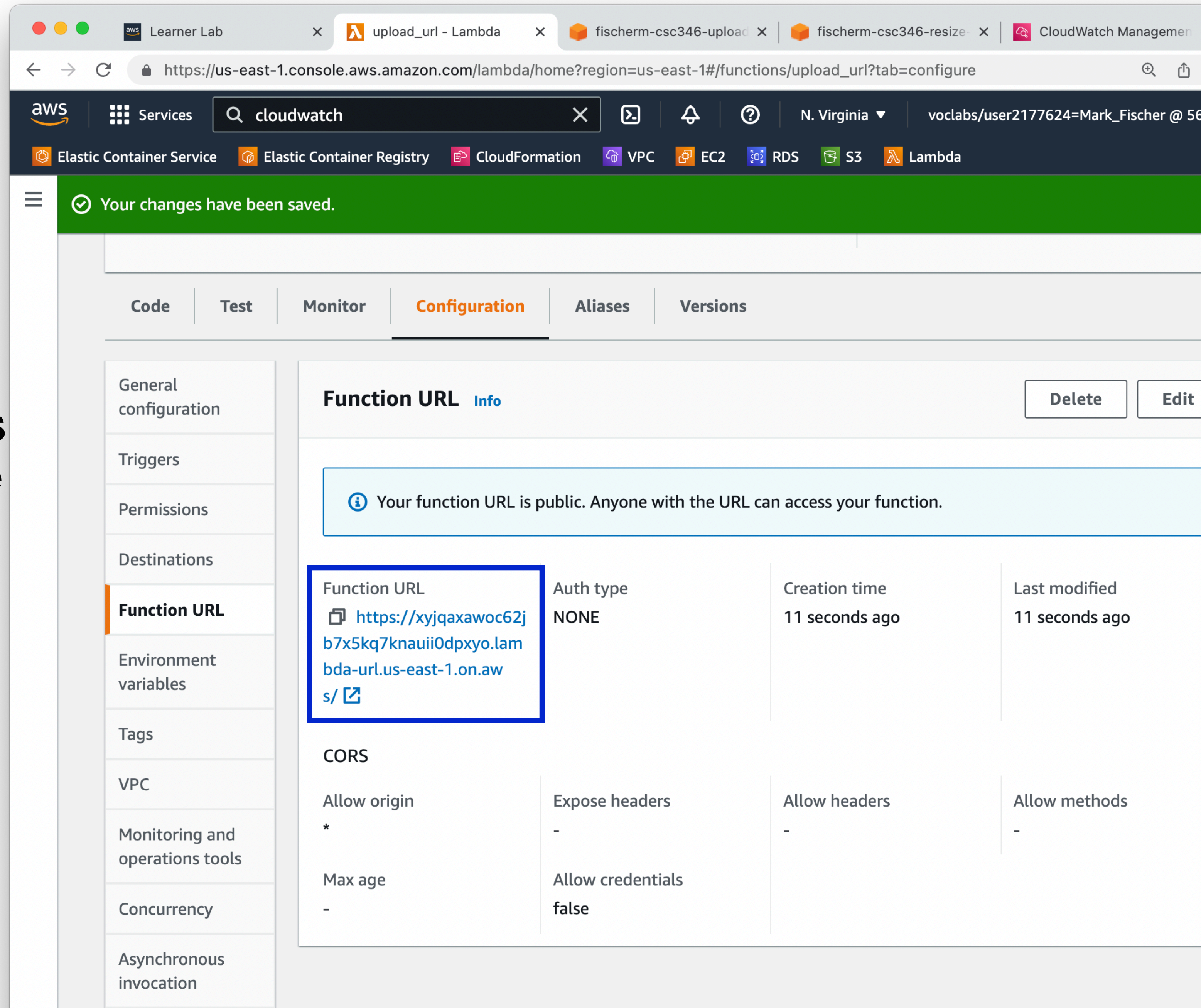
- For simple use cases, Lambda now provides a direct way to invoke the function through a URL
- Basic functionality
- API Gateway is a more robust and featured service for more production projects



AWS Lambda

Function URLs

- For now we will not use any Authentication
- Potentially a security risk as this would allow anyone to generate upload URLs for our buckets and upload files
- Acceptable risk for now
- Could implement your own Basic Auth in the lambda function



AWS Lambda Function URLs

- Use our function URL in Postman
- GET
- Pass the file name in through query string parameters

The screenshot shows a Postman interface for a GET request to the following URL: `https://xyjqaxawoc62jb7x5kq7knauui0dpxyo.lambda-url.us-east-1.on.aws/?filename=x-wing.jpg`. The request is part of a collection named "CSC346 Chat API" and is titled "Generate Upload URL". The "Params" tab is active, showing a single query parameter: "filename" with the value "x-wing.jpg". The response is a JSON object with the following structure:

```
1 {
2   "status": "OK",
3   "upload_url": "https://fischer-csc346-upload-bucket.s3.amazonaws.com/input/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0.jpg?AWSAccessKeyId=ASIAYFSC5FB6C2YAPH7Q&Signature=BM4vxR0JNZ%2BJv7YbYb6MEDk2UE8%3D&x-amz-acl=public-read&content-type=image%2Fjpeg&x-amz-security-token=FwoGZXIvYXZlZEDkaDM6K6tFzj8Ym27tzhSL%2FAWrG1ZyHhojUwg3WxHiDZ%2B1fvhWJdNW4SwDa5bYgYt88EWKTQhUjx9Zk9pjc%2Bt%2B5xHRHwWg4SAfwRRwLV3a1hSkW8cwT%2B5woknvBv12jFeIJyY7GQeBCsn1t802cc9td5GiqSbMuxGnoQyeJx1qekPrz%2BfSXIVoqj99r6zEuhK5mqDZviy45xJxEDRkyHGoH%2F04XZMpvj9p4bs4MlqcevjkSvX%2F0pVLElzSZBq8qblg2ZYkhoxzUxVi2myy6%2F5YMnGTubkHXdhFEbjwmix42iFqZzB9kp%2B0btDRMiD3oDVA9SMws%2FbT7iYyaG6MFIQyDqCYmz2zJ5C54rCIyhSLjSjohKGbBjIusDVfSseUDYHJ%2BZbH6C1uUkEfDm%2F0tTLA2irr6zZI pohkNqa806iB%2F28iy7hKzgz%3D%3D&Expires=1667778013",
4   "full_url": "https://fischer-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-1000.jpg",
5   "thumbnail_url": "https://fischer-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-200.jpg"
6 }
```

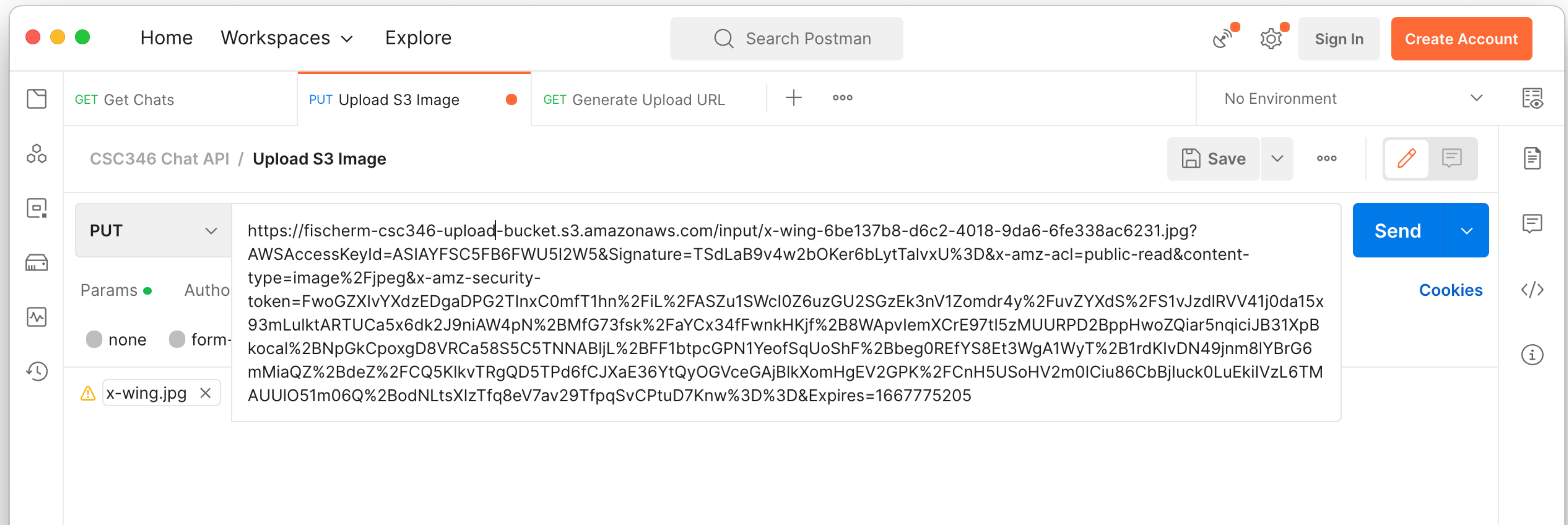
The response status is 200 OK, with a time of 604 ms and a size of 1.28 KB. The response is displayed in the "Body" tab, formatted as JSON.

`https://xyjqaxawoc62jb7x5kq7knauui0dpxyo.lambda-url.us-east-1.on.aws/?filename=x-wing.jpg`

AWS Lambda

Function URLs

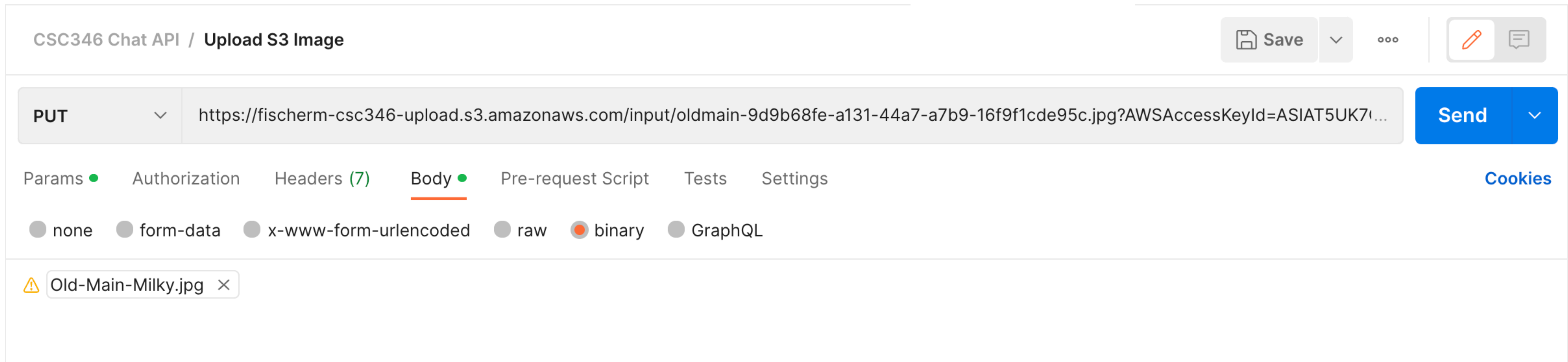
- Use the returned URL as the destination for a PUT HTTP request that passes a file



AWS Lambda

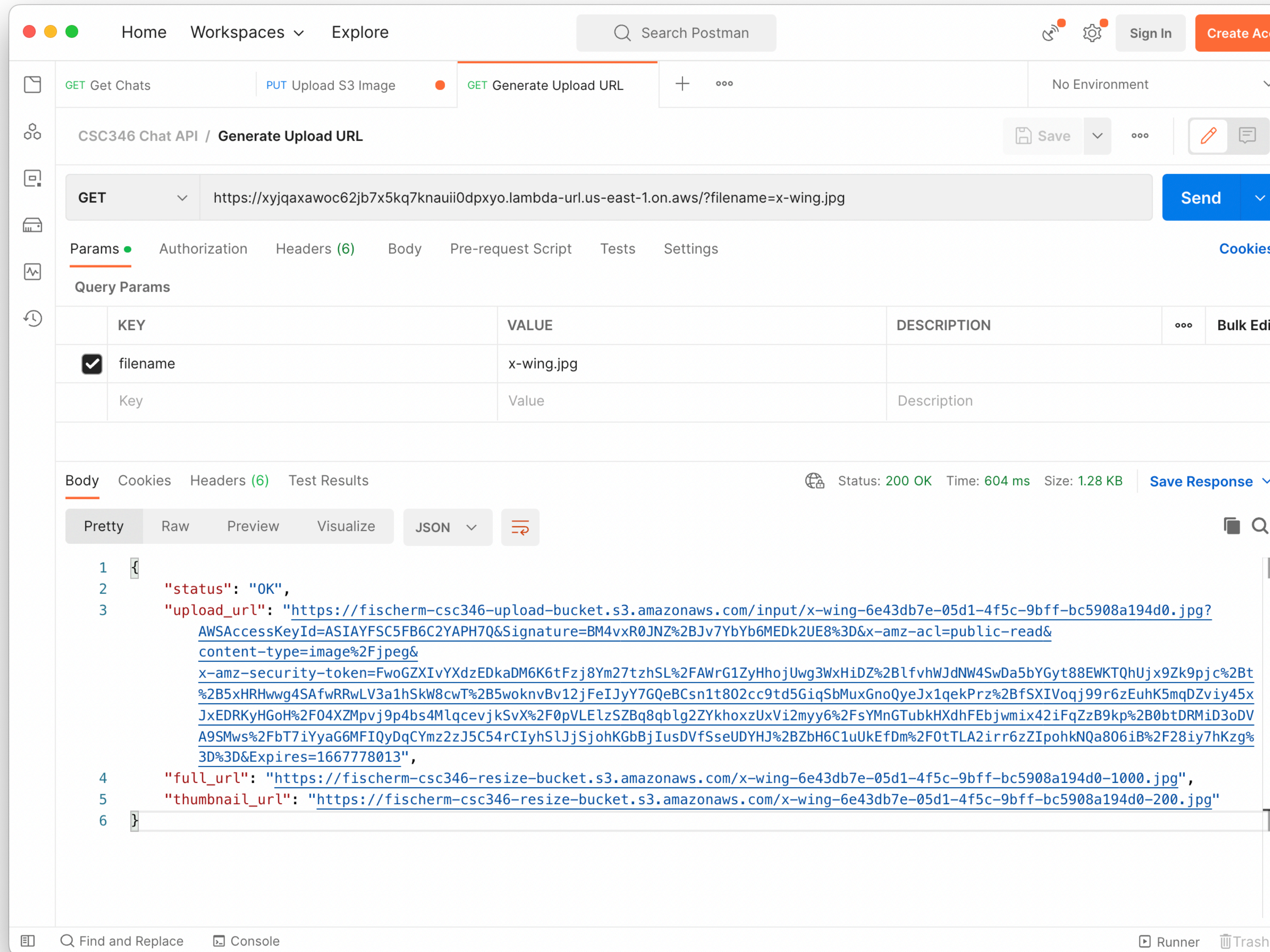
Function URLs

- In Postman, select the image in the Body tab, and choose “binary”



AWS Lambda Function URLs

- Since we uploaded the image to the S3 bucket configured as the trigger for our resize function, the image should be resized automatically
- Our create upload URL call also returns the URLs of the resized objects
- We can view them directly

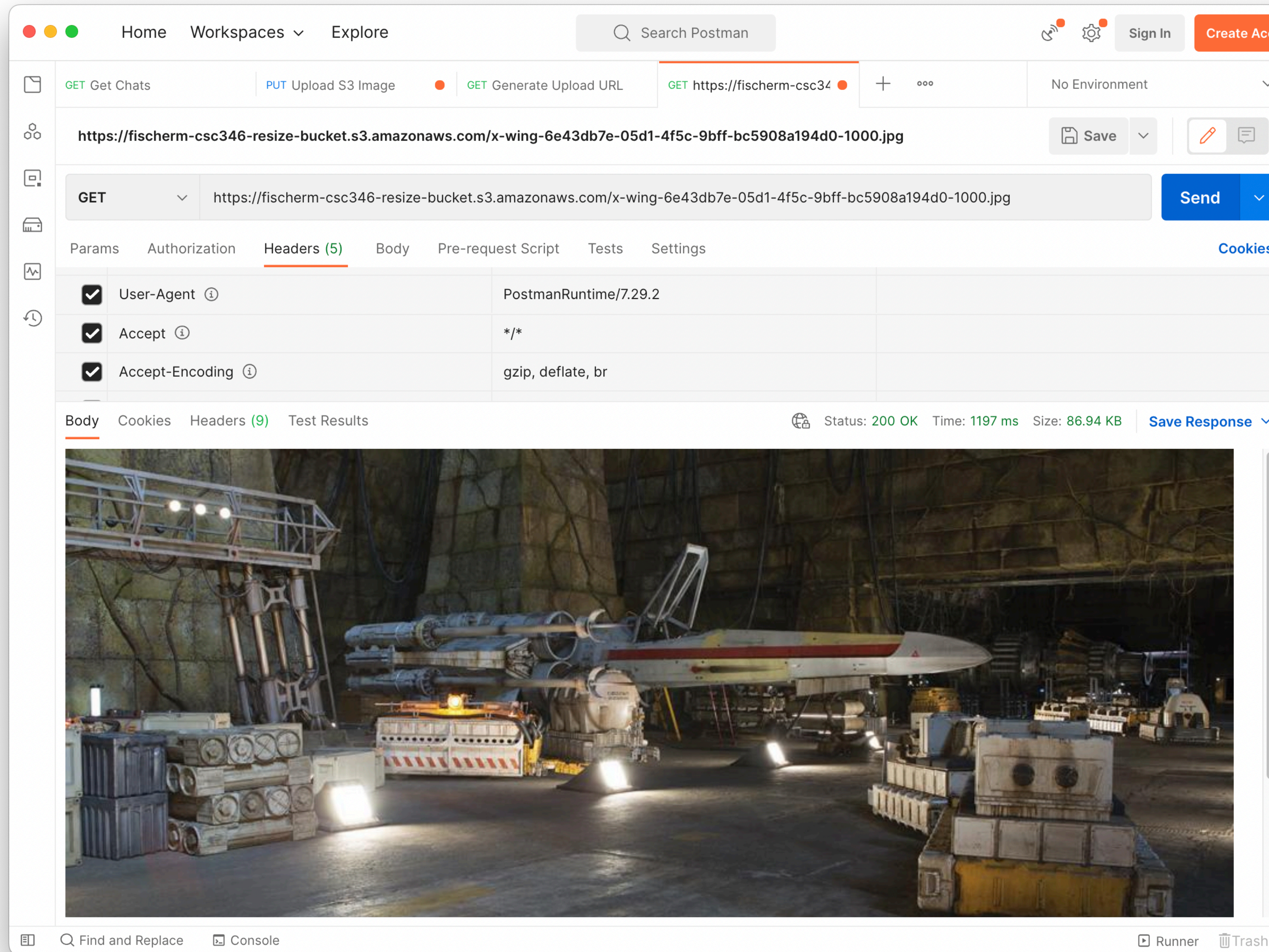


The screenshot shows a Postman interface for a REST client. The active request is a GET request to the endpoint `https://xyjqaxawoc62jb7x5kq7knauui0dpxyo.lambda-url.us-east-1.on.aws/?filename=x-wing.jpg`. The response is a JSON object with the following structure:

```
1 {
2   "status": "OK",
3   "upload_url": "https://fischer-csc346-upload-bucket.s3.amazonaws.com/input/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0.jpg?AWSAccessKeyId=ASIAYFSC5FB6C2YAPH7Q&Signature=BM4vxR0JNZ%2BJv7YbYb6MEDk2UE8%3D&x-amz-acl=public-read&content-type=image%2Fjpeg&x-amz-security-token=FwoGZXIvYXZlZDkaDM6K6tFzj8Ym27tzhSL%2FAWrG1ZyHhojUwg3WxHiDZ%2B1fvhWJdNW4SwDa5bYgYt88EWKTQhUjx9Zk9pjc%2Bt%2B5xHRHwWg4SAfwRRwLV3a1hSkW8cwT%2B5woknvBv12jFeIJyY7GQeBCsn1t802cc9td5GiqSbMuxGnoQyeJx1qekPrz%2BfSXIVoqj99r6zEuhK5mqDZvivy45xJxEDRkyHGoH%2F04XZMpvj9p4bs4MlqcevjkSvX%2F0pVLElzSZBq8qblg2ZYkhoxzUxVi2myy6%2F5YMnGTubkHXdhFEbjwmix42iFqZzB9kp%2B0btDRMiD3oDVA9SMws%2FbT7iYyaG6MFIQyDqCYmz2zJ5C54rCIyhSLjSjohKGbBjIusDVfSseUDYHJ%2BZbH6C1uUkEfDm%2F0tTLA2irr6zZIpohkNqa806iB%2F28iy7hKzgz3D%3D&Expires=1667778013",
4   "full_url": "https://fischer-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-1000.jpg",
5   "thumbnail_url": "https://fischer-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-200.jpg"
6 }
```

AWS Lambda Function URLs

- Since we uploaded the image to the S3 bucket configured as the trigger for our resize function, the image should be resized automatically
- Our create upload URL call also returns the URLs of the resized objects
- We can view them directly



The screenshot shows the Postman interface for a GET request to the URL `https://fischer-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-1000.jpg`. The request headers are expanded, showing:

Header	Value
User-Agent	PostmanRuntime/7.29.2
Accept	*/*
Accept-Encoding	gzip, deflate, br

The response body shows a large image of a Starfighter in a hangar. The status bar indicates a 200 OK response with a time of 1197 ms and a size of 86.94 KB.

