

Managed Cloud Services

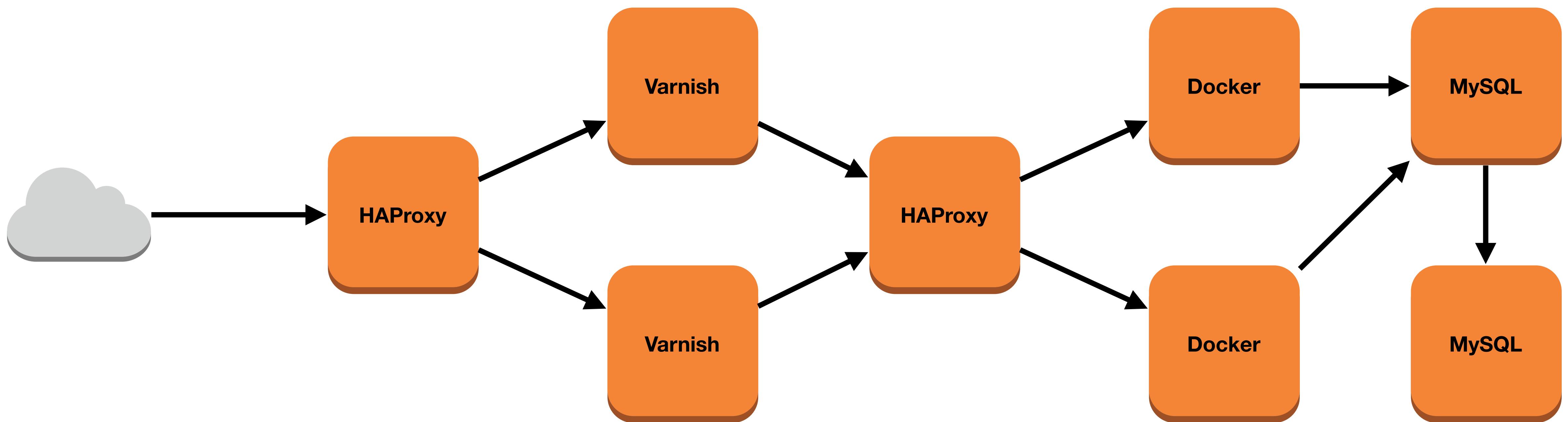
When you don't want to run it yourself

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

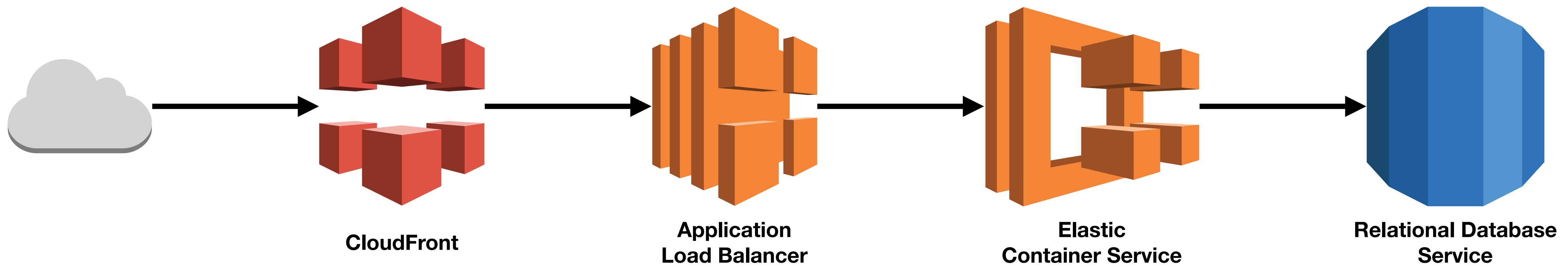
VM Centric Architecture



Managed Cloud Services

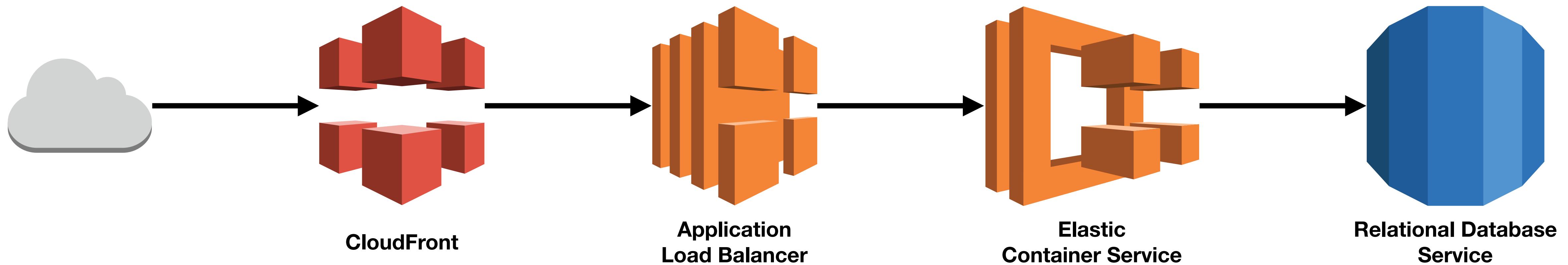
	VM / EC2	AWS Service
Database	MySQL	RDS MySQL
Load Balancer	HAProxy	Elastic Load Balancer Application Load Balancer
Docker	Docker	Elastic Container Service
Caching	Varnish	CloudFront

Cloud Centric Architecture

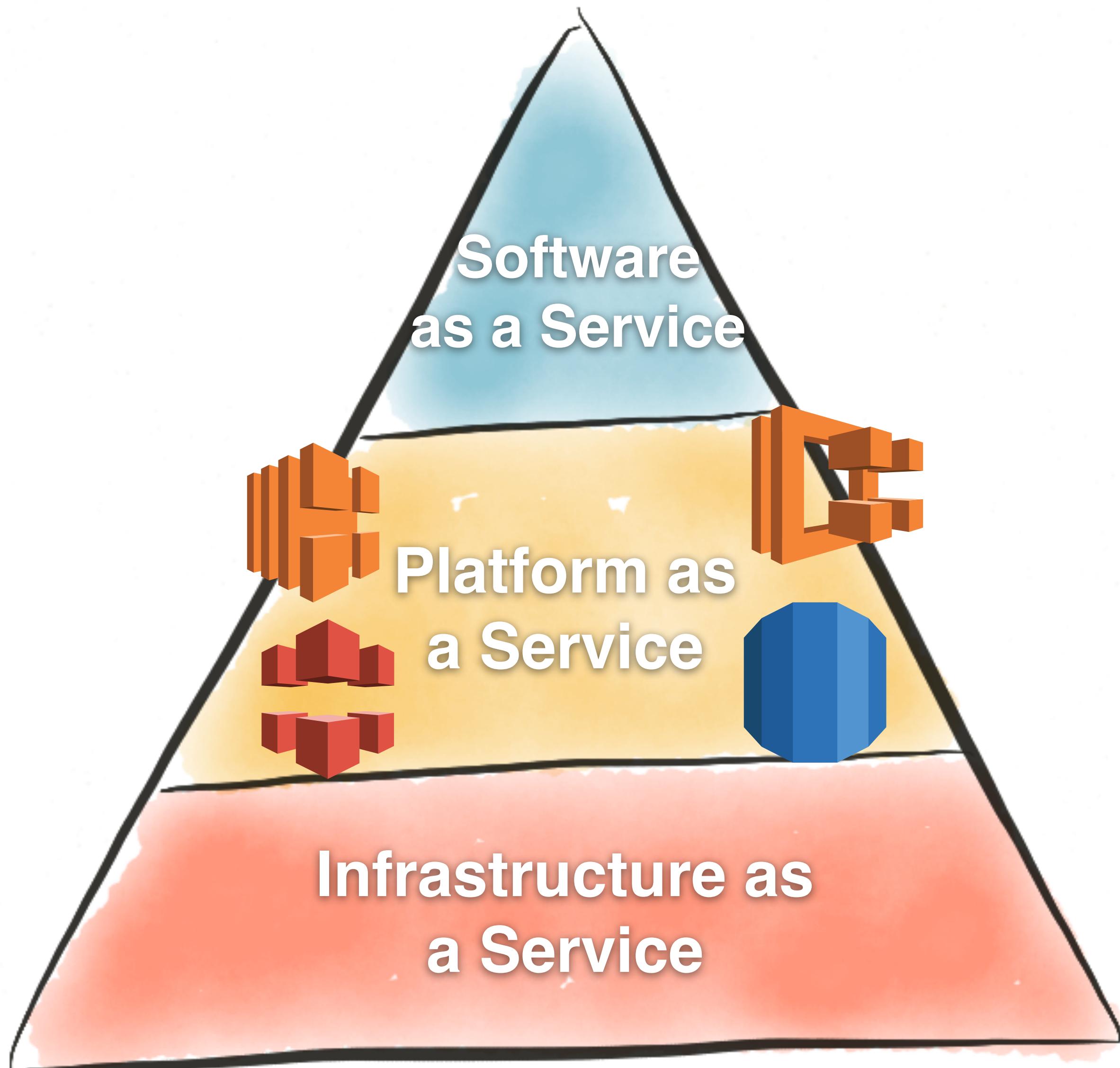


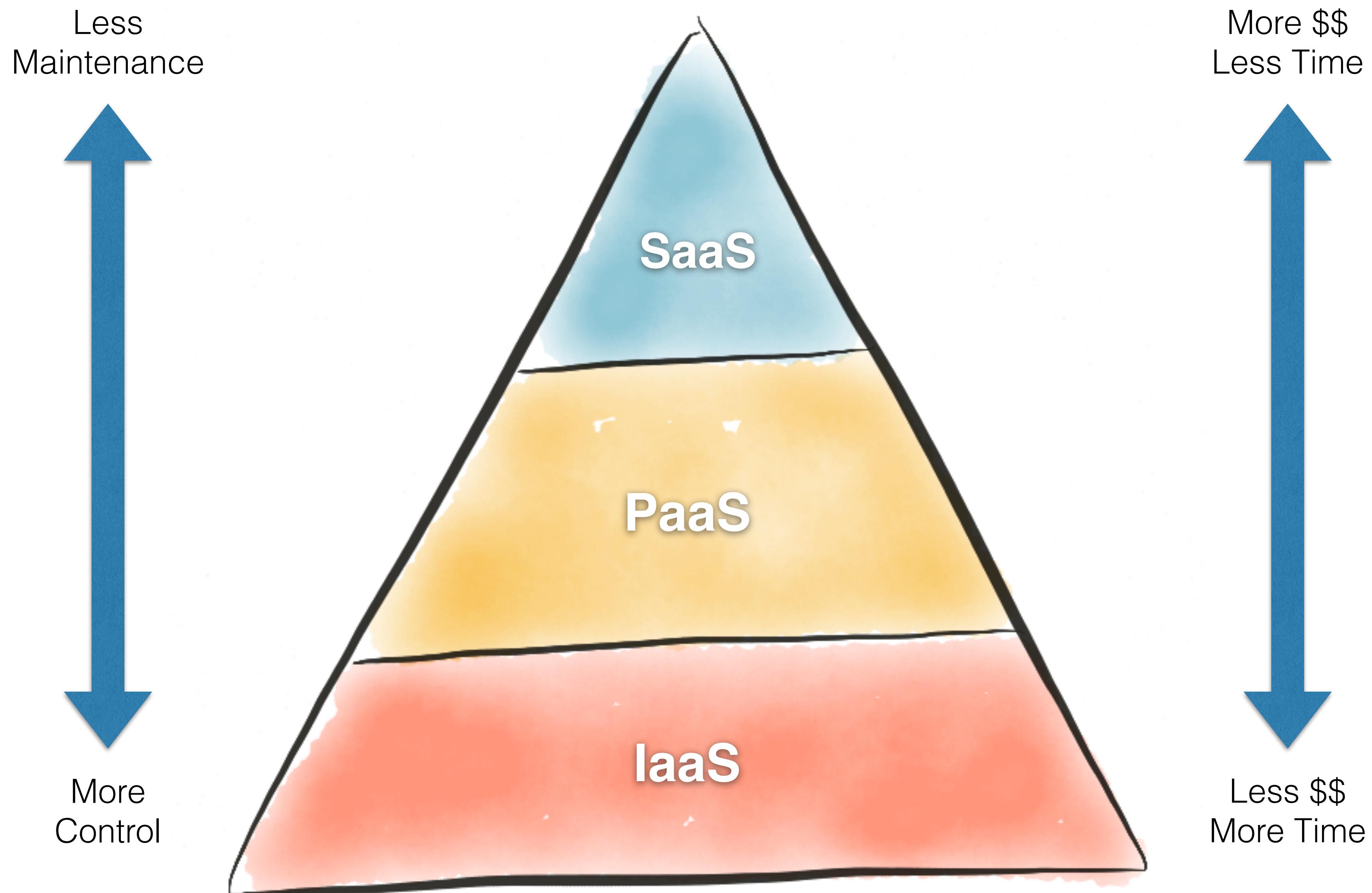
Managed Cloud Services

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





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 top navigation bar includes tabs for 'Learner Lab' and 'CloudFormation - Stack'. The URL in the address bar is <https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks?filteringStatus=active&filteringText=&viewNest...>. The main menu bar has 'Services' selected, along with icons for CloudFormation, VPC, EC2, RDS, S3, and Lambda. The left sidebar has 'CloudFormation' selected, with options for 'Stacks', 'StackSets', and 'Exports'. The main content area is titled 'CloudFormation > Stacks' and shows 'Stacks (2)'. It includes buttons for 'Create stack', 'View nested', and filtering by 'Active'. A table lists the stacks:

Stack name	Status	Description
fischerm-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 interface. The top navigation bar includes tabs for 'Learner Lab' and 'CloudFormation - Stack'. The URL in the address bar is <https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks?filteringStatus=active&filteringText=&viewNest...>. The main menu bar has 'Services' selected, with options like 'Elastic Container Service', 'Elastic Container Registry', 'CloudFormation', 'VPC', 'EC2', 'RDS', 'S3', and 'Lambda'. On the left, a sidebar titled 'CloudFormation' shows 'Stacks' (selected), 'StackSets', and 'Exports'. The main content area is titled 'CloudFormation > Stacks' and shows 'Stacks (2)'. It includes buttons for 'Create stack' and 'With new resources (standard)'. A table lists two stacks:

Stack name	Status	Description
fischerm-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

<https://fischerm-csc346-upload-bucket.s3.amazonaws.com/templates/s3-website.yaml>

Amazon S3 template URL

https://fischerm-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			
Stack info		Events	
Delete		Update	
Stack actions ▾		Create stack ▾	
Events (9)		C	
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. The browser title bar reads "fischerm-csc346-download - S3". The URL in the address bar is "https://s3.console.aws.amazon.com/s3/buckets/fischerm-csc346-download?region=us-east-1&bucket=fischerm-csc346-download". The top navigation bar includes the AWS logo, a "Services" dropdown, a search bar, and various AWS service links: Elastic Container Service, CloudFormation, CloudWatch, EC2, Amazon EventBridge, Systems Manager, Support, and CodeBuild. A "Global" dropdown and a user account link "fdn-SysAdmin/fischerm@arizona.edu" are also present. The main content area shows the "Amazon S3 > Buckets > fischerm-csc346-download" path. The bucket name "fischerm-csc346-download" is displayed with a "Publicly accessible" status indicator. Below the bucket name are tabs for "Objects", "Properties", "Permissions", "Metrics", "Management", and "Access Points", with "Objects" being the active tab. Under the "Objects" tab, there are two items: "examples/" (Folder) and "s3_template.yaml" (yaml). Both items were modified on March 25, 2024, at 08:52:07 (UTC-07:00). The "Actions" menu is visible, showing options like Copy S3 URI, Copy URL, Download, Open, Delete, and Create folder. A large orange "Upload" button is prominently displayed. A note below the objects states: "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](#)". A search bar labeled "Find objects by prefix" is also present.

Name	Type	Last modified	Size	Storage class
examples/	Folder	-	-	-
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

The screenshot shows the AWS S3 console with the URL <https://s3.console.aws.amazon.com/s3/object/fischerm-csc346-download?region=us-east-1&bucket>. The object key is `examples/Old-Main-Milky.jpg`.

Properties tab selected.

Object overview

Owner	S3 URI
ua-uits-sia-nonprod-aws	s3://fischerm-csc346-download/examples/Old-Main-Milky.jpg
AWS Region	Amazon Resource Name (ARN)
US East (N. Virginia) us-east-1	arn:aws:s3:::fischerm-csc346-download/examples/Old-Main-Milky.jpg

Bucket Name: fischerm-csc346-download

Object Key: examples/Old-Main-Milky.jpg

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 top navigation bar includes the AWS logo, a search bar, and various service links like Elastic Container Service, CloudFormation, CloudWatch, EC2, Amazon EventBridge, Systems Manager, Support, and CodeBuild. The user is signed in as fdn-SysAdmin/fischerm@arizona.edu. The current view is under the 'Amazon S3' section, specifically for the bucket 'fischerm-csc346-download' and the key 'examples/ Old-Main-Milky.jpg'. The object name 'Old-Main-Milky.jpg' is displayed with an 'Info' link. Below the object name are three buttons: 'Copy S3 URI', 'Download', and 'Open'. A dropdown menu labeled 'Object actions' is also present.

Properties

Attribute	Value
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://fischerm-csc346-download/examples/Old-Main-Milky.jpg

Amazon Resource Name (ARN)
arn:aws:s3:::fischerm-csc346-download/examples/Old-Main-Milky.jpg

Entity tag (Etag)
aaa3e5256dc5cc6b9758bd36331499f0

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

<https://fischerm-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.

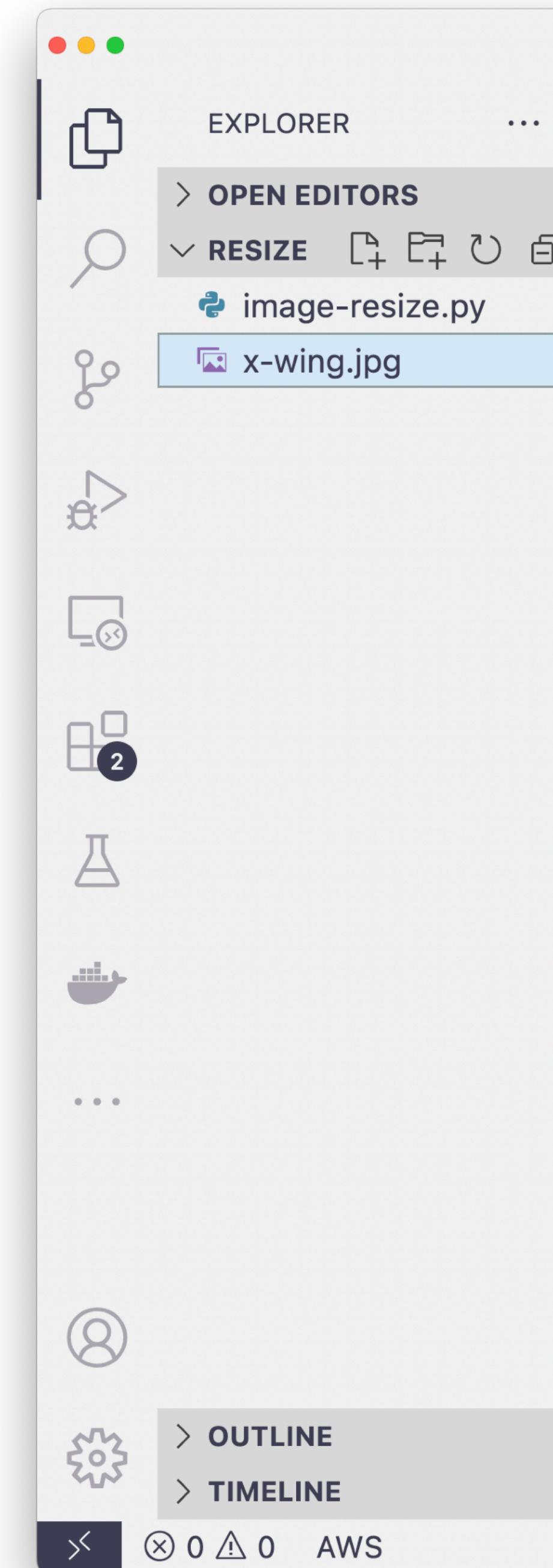
AWS Lambda

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



The screenshot shows the AWS Lambda function configuration interface. The 'RESIZE' tab is selected in the top navigation bar. In the 'HANDLER' section, 'image-resize.py' is listed as the handler. Below it, 'x-wing.jpg' is listed as the input image file. The 'CODEWHISPERER REFERENCE LOG' tab is active, displaying the following terminal output:

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

On the right side of the interface, there is a preview window titled 'x-wing.jpg — resize' showing a Star Wars X-wing fighter in a hangar.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "image-resize.py — resize". The left sidebar (Explorer) lists files: "OPEN EDITORS" (empty), "RESIZE" (selected), "image-resize.py" (highlighted), "x-wing-200.jpg", and "x-wing.jpg". The main editor area contains the following Python code:

```
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 ...

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

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 with placeholder text "[Option+S]", and a "Create function" button. The main content area is titled "Functions (2)" and displays two entries:

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

The left sidebar contains links for Dashboard, Applications, Functions (which is selected), and Additional resources (Code signing configurations, Layers, Replicas). At the bottom, there is a "Related AWS resources" section.

AWS Lambda

Image Resizing in the Cloud

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

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The top navigation bar includes the AWS logo, services like Elastic Container Service, CloudFormation, CloudWatch, EC2, Amazon EventBridge, Systems Manager, Support, and CodeBuild, and a search bar. The URL is https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/create/function.

The main content area is titled 'Create function' with an 'Info' link. It says 'Choose one of the following options to create your function.' Three options are shown:

- 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.
- Container image: Select a container image to deploy for your function.

The 'Basic information' section contains fields for 'Function name' (set to 'resize-image'), 'Runtime' (set to 'Python 3.10'), and 'Architecture' (set to 'x86_64'). Below these are sections for 'Permissions' and a note about default execution role creation.

At the bottom, there are links for 'CloudShell', 'Feedback', and copyright information: © 2024, Amazon Web Services, Inc. or its affiliates. There are also links for 'Privacy', 'Terms', and 'Cookie preferences'. A search bar at the bottom contains the text 'sia-non'.

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”

The screenshot shows the AWS Lambda function creation interface. In the 'Permissions' section, the 'Use an existing role' option is selected. A dropdown menu is open, showing the role 'LabRole' which was chosen earlier. The 'Advanced settings' section is partially visible below.

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

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
 Use an existing role
 Create a new role from AWS policy templates

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.
LabRole
[View the LabRole role on the IAM console](#).

► Advanced settings

Feedback Looking for language selection? Find it in the new Unified Settings [↗](#)

© 2022, Amazon Web Services, Inc. or its affiliates

AWS Lambda

Image Resizing in the Cloud

- Default “Hello World” function

The screenshot shows the AWS Lambda console interface. At the top, the URL is https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/resize-image?newFunction=true&tab=code. The navigation bar includes services like Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, RDS, S3, and Lambda. The Lambda service is selected.

Function Overview: The 'resize-image' function is listed. It has no triggers or destinations added. The Function ARN is arn:aws:lambda:us-east-1:561707296892:function:resize-image. The Function URL is available but not yet used.

Code Source: The 'lambda_function.py' file is displayed in the code editor. The code is as follows:

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

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

The screenshot shows the AWS Lambda function configuration interface. At the top, the 'Runtime settings' section is displayed, showing 'Runtime' set to 'Python 3.10' and 'Handler' set to 'lambda_function.lambda_handler'. Below this, the 'Code source' section is shown in a code editor. A red arrow points from the 'Handler' setting in the top right towards the 'lambda_handler' definition in the code editor. Another red arrow points from the 'lambda_function.py' file in the environment sidebar towards the same 'lambda_handler' definition.

Runtime settings [Info](#)

Runtime
Python 3.10

Handler [Info](#)
lambda_function.lambda_handler

Learner Lab [resize-image - Lambda](#)

https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/resize-image?tab=code

aws Services Search [Option+S] N. Virginia v vocabs/user2177624=Mark_Fischer @ 5617-0729-6892 v

Elastic Container Service Elastic Container Registry CloudFormation VPC EC2 RDS S3 Lambda

Code source [Info](#)

File Edit Find View Go Tools Window Test Deploy

Upload from

Environment

Go to Anything (⌘ P)

lambda_function x Execution results x

```
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 def lambda_handler(event, context):
13     filename = "pillars-of-creation.png"
14     source_url = f"https://fischerm-csc346-upload-bucket.s3.amazonaws.com/input/{filename}"
15
16     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/singed-exc",
    "accept-encoding": "gzip, deflate, br",
    "Host": "70ixmpl4f1.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:AROAYFSC5FB6KLKFWGIOO: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": "fischerm-csc346-upload-bucket",  
          "ownerIdentity": {"principalId": "A3NRT1KH8KAG57"},  
          "arn": "arn:aws:s3:::fischerm-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”

The screenshot shows the AWS Lambda console interface. At the top, there are three tabs: "Learner Lab", "resize-image - Lambda", and "fischerm-csc346-upload-buck". The URL in the address bar is <https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/resize-image?tab=code>. The navigation bar includes "Services" (with "Lambda" selected), "Search", "[Option+S]", and various AWS service icons like ECR, CloudFormation, VPC, EC2, RDS, S3, and Lambda.

The main content area shows the "resize-image" function. The "Function overview" section displays the function name, ARN, and a "Layers" section indicating one layer. Below this are "Add trigger" and "Add destination" buttons, both of which are highlighted with a blue border. The "Code" tab is currently selected, showing the "Code source" section with an "Info" link. At the bottom, there is a navigation bar with links for "File", "Edit", "Find", "View", "Go", "Tools", "Window", "Test", "Deploy", and "Feedback". A note at the bottom says "Looking for language selection? Find it in the new Unified Settings". The right side of the screen has a sidebar with sections for "Description", "Last modified" (3 days ago), "Function ARN" (arn:aws:lambda:2:function:resize-image), and "Function URL".

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 with the "Trigger configuration" step open. The "Event type" dropdown is set to "All object create events". The "Prefix - optional" field contains "input/", and the "Suffix - optional" field contains ".jpg". The "Recursive invocation" note at the bottom is visible.

Trigger configuration [Info](#)

S3

Bucket

Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

s3/fischerm-csc346-upload-bucket X C

Bucket region: us-east-1

Event type

Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events

Prefix - optional

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

input/

Suffix - optional

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

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](#)

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!

The screenshot shows the AWS Lambda console interface. The top navigation bar includes tabs for Learner Lab, resize-image - Lambda, and fischerm-csc346-upload-bucket. The main menu has options like Services, Search, and N. Virginia. The Lambda function 'resize-image' is selected. The Configuration tab is active. On the left, a sidebar lists General configuration, Triggers (which is selected and highlighted in orange), Permissions, Destinations, Function URL, Environment variables, Tags, VPC, Monitoring and operations tools, Concurrency, Asynchronous invocation, and Code signing. The main content area shows the Triggers (1) section with a table. One trigger is listed: S3: fischerm-csc346-upload-bucket. The trigger details pane shows the following information:

- Bucket arn: arn:aws:s3:::fischerm-csc346-upload-bucket
- Event type: s3:ObjectCreated:*
- Notification name: e2733ff1-399f-4645-8778-7e4fef7a7c3a
- Prefix: input/
- Service principal: s3.amazonaws.com
- Source account: 561707296892
- Statement ID: lambda-70cb8c7a-9df0-4c11-a6c4-8d239ce0e8f1

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:fischerm-csc346-imagelayer:2
```

AWS Lambda Layers

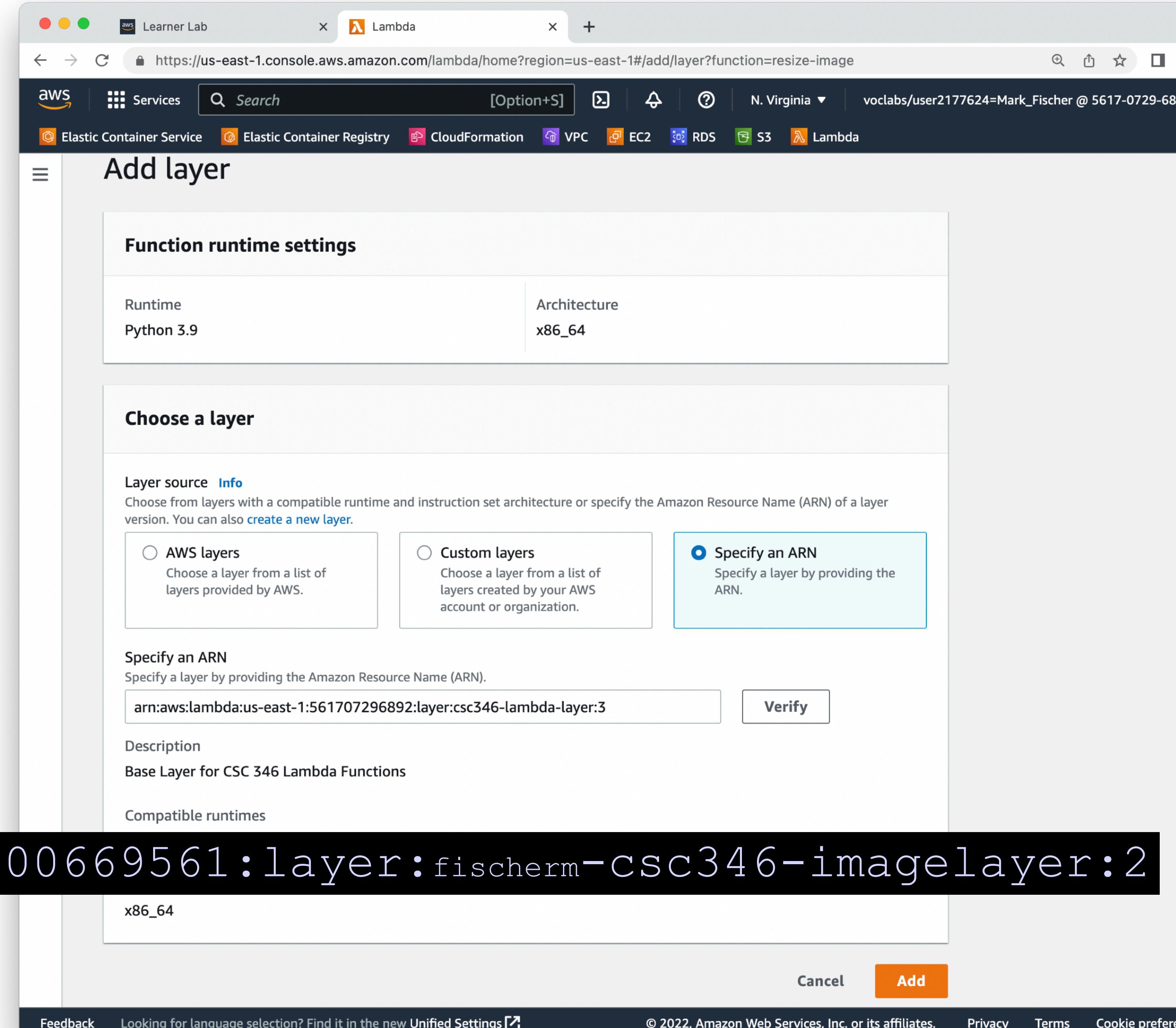
- Scroll down
- Click the “Add a layer” button

The screenshot shows the AWS Lambda function configuration interface for a function named "resize-image". The "Code properties" section displays the package size (299.0 byte), SHA256 hash (f106ZlRH/KN6Ra3twvdRllUYaxv182Tjx0qNWNlKhl=), and last modified time (November 3, 2022 at 07:30 PM MST). The "Runtime settings" section shows the runtime (Python 3.9), handler (lambda_function.lambda_handler), and architecture (x86_64). In the "Layers" section, there is a table with columns: Merge order, Name, Layer version, Compatible runtimes, Compatible architectures, and Version ARN. A blue box highlights the "Add a layer" button.

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
There is no data to display.					

AWS Lambda Layers

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



```
arn:aws:lambda:us-east-1:269800669561:layer:fischerm-csc346-imagelayer: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 function editor interface. At the top, the title bar reads "Learner Lab" and "resize-image - Lambda". The URL is "https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/resize-image?tab=code". The navigation bar includes "aws", "Services", "Search [Option+S]", "N. Virginia", and "voclabs/user2177624=Mark_Fischer @ 5617-0729-68". Below the navigation bar are tabs for "Code", "Test", "Monitor", "Configuration", "Aliases", and "Versions", with "Code" being the active tab. On the left, there's a sidebar with "Environment" and a file tree showing a folder named "resize-image - /" containing a file named "lambda_function.py". The main area is titled "Code source" and "Info". It features a toolbar with "File", "Edit", "Find", "View", "Go", "Tools", "Window", "Test" (which is highlighted), and "Deploy". A search bar says "Go to Anything (% P)". Below the toolbar is a preview pane titled "lambda_function" with "Execution results" and a green plus sign icon. The code editor displays the following Python code:

```
import json

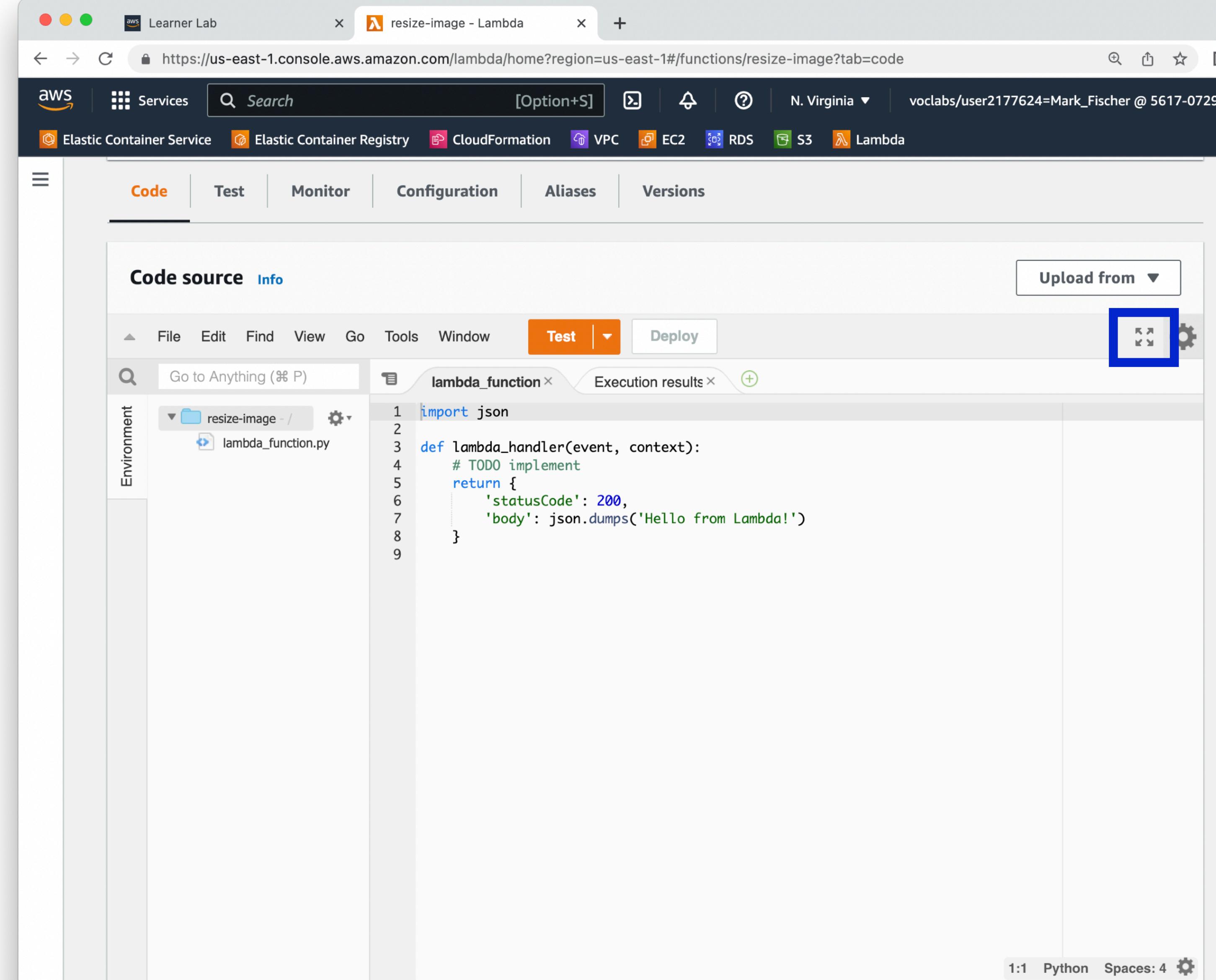
def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

At the bottom right of the code editor, it says "1:1 Python Spaces: 4".

AWS Lambda

Layers

- Can make the code editor fill the browser window



The screenshot shows the AWS Lambda console interface with the 'Code' tab selected. The main area displays a code editor for a Python function named 'lambda_function'. The code is as follows:

```
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 interface includes a sidebar for the environment, a toolbar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test' (which is currently selected), and 'Deploy' buttons. There is also a search bar and a file navigation panel.

Code properties

Feedback

Looking for language selection? Find it in the new Unified Settings [⚙️](#)

© 2022, Amazon Web Services, Inc. or its affiliates.

Privacy

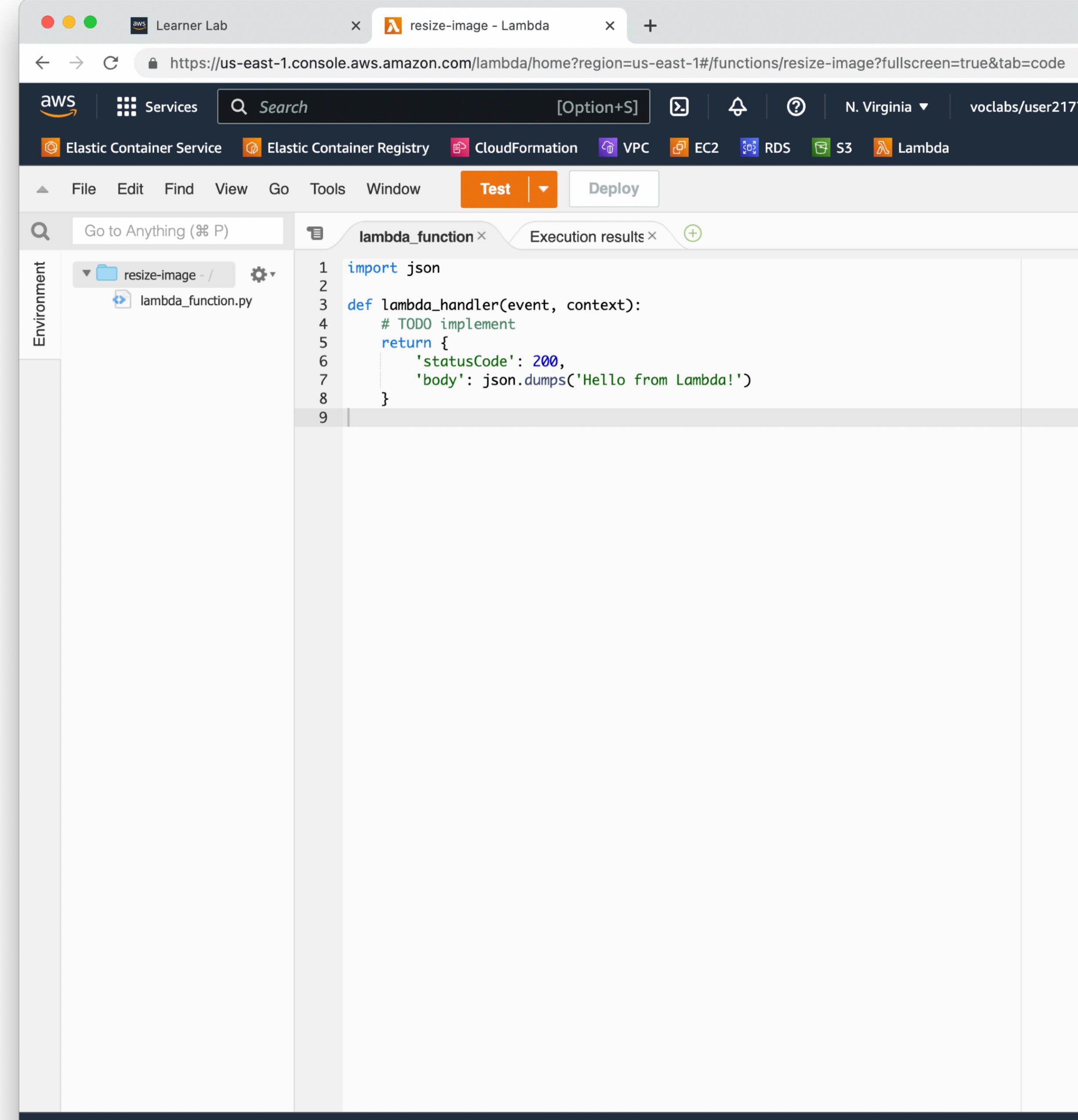
Terms

Cookie preferences

AWS Lambda

Layers

- Can make the code editor fill the browser window



The screenshot shows the AWS Lambda console interface. The browser tab is titled "resize-image - Lambda". The URL is <https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/resize-image?fullscreen=true&tab=code>. The top navigation bar includes "Learner Lab", "Services", a search bar, and account information for "N. Virginia" and "voclabs/user2177624=Mark_Fischer @ 5617-0729-". Below the navigation is a menu bar with "File", "Edit", "Find", "View", "Go", "Tools", "Window", "Test" (which is highlighted in orange), and "Deploy". On the left, there's a sidebar labeled "Environment" with a "Go to Anything" search bar and a file tree showing a folder "resize-image - /" containing "lambda_function.py". The main content area is titled "lambda_function" and "Execution results". It displays the following Python code:

```
import json
def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

At the bottom right of the code editor, it says "9:1 Python Space".

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' dropdown, a search bar containing 'cloudwatch', and account information for 'N. Virginia' and 'voclabs/user2177624'. Below the navigation bar, a secondary navigation bar includes links for 'Elastic Container Service', 'Elastic Container Registry', 'CloudFormation', 'VPC', 'EC2', 'RDS', 'S3', and 'Lambda'. The main area is a code editor titled 'lambda_function'. On the left, there's a sidebar labeled 'Environment' with a 'resize-image - /' folder containing a file named 'lambda_function.py'. The code editor itself displays the following Python script:

```
1 import boto3
2 import os
3 import mimetypes
4 from PIL import Image
5
6 resize_bucket = "fischerm-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.

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, a search bar containing 'cloudwatch', and account information for 'N. Virginia' and 'voclabs/user2177624'. Below the navigation bar, a row of service links includes 'Elastic Container Service', 'Elastic Container Registry', 'CloudFormation', 'VPC', 'EC2', 'RDS', 'S3', and 'Lambda'. The main area has a toolbar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test' (which is currently selected), and 'Deploy' buttons. A blue box highlights the 'Deploy' button. On the left, there's an 'Environment' sidebar with a 'resize-image - /' folder icon and a 'lambda_function.py' file icon. The main content area displays the Python code for the Lambda function:

```
1 import boto3
2 import os
3 import mimetypes
4 from PIL import Image
5
6 resize_bucket = "fischerm-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

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, search bar ('cloudwatch'), and account information ('N. Virginia' and 'voclabs/user2177624'). Below the navigation bar, a row of service links includes 'Elastic Container Service', 'Elastic Container Registry', 'CloudFormation', 'VPC', 'EC2', 'RDS', 'S3', and 'Lambda'. The main area has a toolbar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test' (which is highlighted with a blue border), and 'Deploy' buttons. A 'Go to Anything' search bar is above a file browser window. The file browser shows a folder named 'resize-image-' containing a file named 'lambda_function.py'. The code editor displays the following Python script:

```
1 import boto3
2 import os
3 import mimetypes
4 from PIL import Image
5
6 resize_bucket = "fischerm-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 'Configure test event' dialog in the AWS Lambda console. The 'Event name' field is highlighted with a blue border and contains the value 'test'. Below the event name, a note states: 'Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.' Under 'Event sharing settings', the 'Private' option is selected. In the 'Template - optional' section, 'hello-world' is listed. At the bottom, the 'Event JSON' section displays the following JSON code:

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

There is a 'Format JSON' button to the right of the JSON code.

AWS Lambda

Image Resizing in the Cloud

- 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 top navigation bar includes tabs for Learner Lab, Services, and Lambda. The main content area displays a Lambda function named "resize-image". A search bar at the top right contains the placeholder "[Option+S]". Below the search bar are links to other AWS services: Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, RDS, S3, and Lambda. The Lambda menu bar includes File, Edit, Find, View, Go, Tools, Window, Test, Deploy, and a status message "Changes not deployed". A "Test" button is highlighted in orange. The left sidebar shows the "Environment" section with a folder named "resize-image - /" containing "lambda_function.py". The main panel displays the "Execution results" for a test event named "test". The response object contains an error message: {"errorMessage": "2022-11-04T03:30:42.791Z d1f34153-8462-4956-bb40-74458cd40aa0 Task timed out after 3.05 seconds"}. The "Function Logs" section shows the start, end, and report logs for the same request ID. The "Request ID" is listed as d1f34153-8462-4956-bb40-74458cd40aa0. The status bar at the bottom right indicates "Status: Failed" and "Max memory".

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 'Edit basic settings' page for a Lambda function named 'resize-image'. The page is part of the 'Learner Lab' interface, specifically the 'Configure' tab. The 'Basic settings' section is active. Key configuration options shown include:

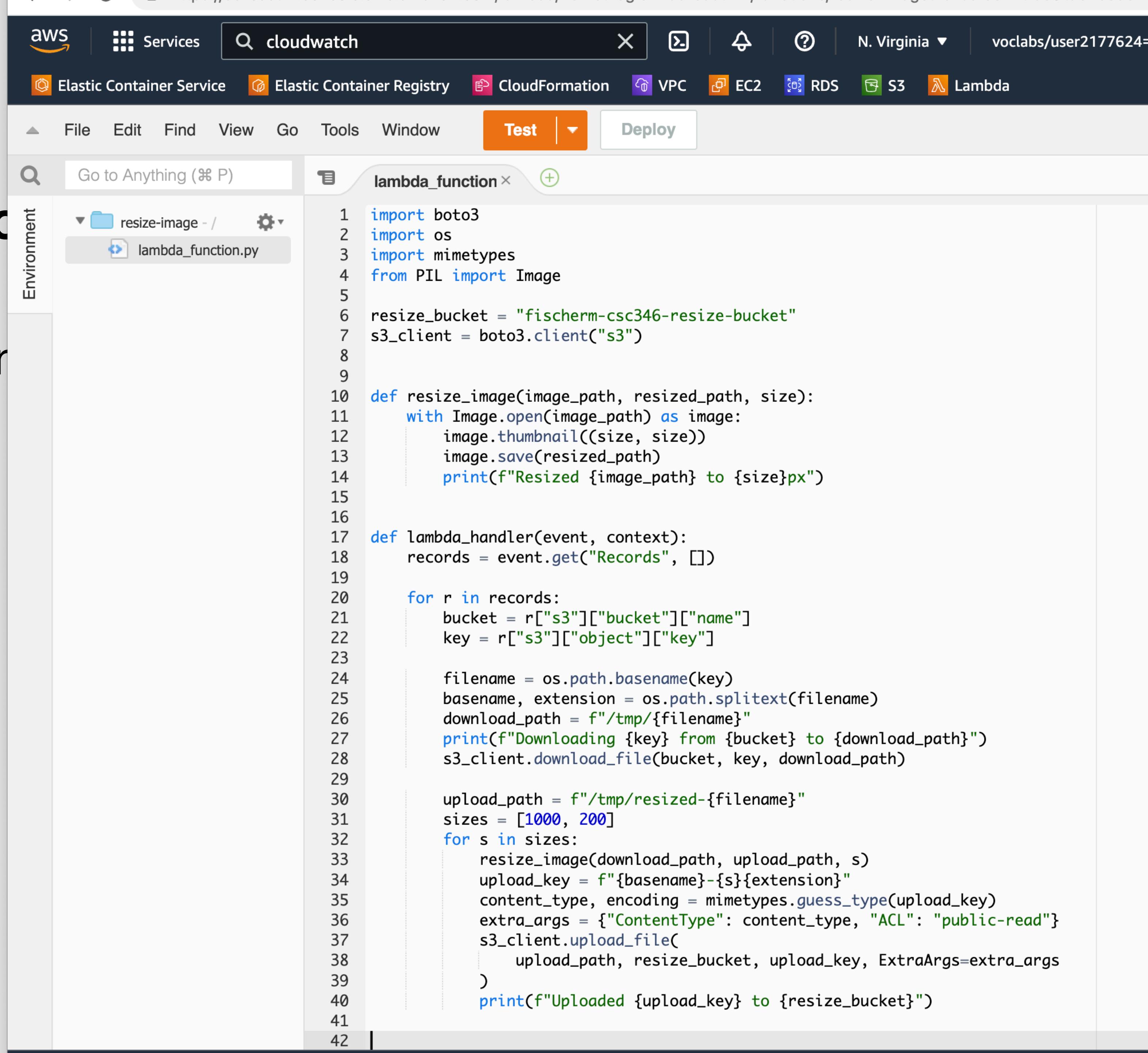
- Description - optional:** A text input field for a function description.
- Memory:** Set to 1024 MB. A note states: "Your function is allocated CPU proportional to the memory configured."
- Ephemeral storage:** Set to 512 MB. A note states: "You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)"
- Timeout:** Set to 1 min 0 sec.
- Execution role:** A radio button group where "Use an existing role" is selected. A note says: "Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#)."
- Existing role:** A dropdown menu showing "LabRole". A note says: "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."

At the bottom right 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



The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, a 'Services' dropdown, a search bar containing 'cloudwatch', and account information for 'N. Virginia' and 'voclabs/user2177624'. Below the navigation bar, a row of service links includes Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, RDS, S3, and Lambda. The main area has a toolbar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test' (which is highlighted in orange), and 'Deploy' buttons. A sidebar on the left is titled 'Environment' and shows a folder structure for 'resize-image' containing a file named 'lambda_function.py'. The main content area displays the Python code for the Lambda function:

```
1 import boto3
2 import os
3 import mimetypes
4 from PIL import Image
5
6 resize_bucket = "fischerm-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 shows the AWS Lambda console interface. The top navigation bar includes tabs for Learner Lab, Services (selected), and Lambda. The main area displays the 'Code source' tab for a function named 'resize-image'. The 'Test' button is highlighted in orange. The 'Execution result' section shows a successful test event named 'test' with a response of 'null'. The 'Function Logs' section contains the following log entries:
START RequestId: 1f72c645-89e6-449f-aba4-4026b96db0e6 Version: \$LATEST
Resized /tmp/pillars-of-creation.png to 200px
END RequestId: 1f72c645-89e6-449f-aba4-4026b96db0e6
REPORT RequestId: 1f72c645-89e6-449f-aba4-4026b96db0e6 Duration: 4448.69 ms Billed

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 the AWS Lambda function editor interface. At the top, there's a navigation bar with File, Edit, Find, View, Go, Tools, Window, a Test button, and a Deploy button. Below the navigation bar is a search bar labeled "Go to Anything (⌘ P)". To the right of the search bar is a tab labeled "lambda_function" with a close button. Further right is a tab labeled "Execution results" with a plus sign icon. On the left side, there's a sidebar titled "Environment" with a "upload_url - /" folder expanded, showing a file named "lambda_function.py". The main area contains 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 (%(name)s)%(message)s")
12 logger = logging.getLogger(__name__)
13 conf = Config(region_name="us-east-1")
14 s3 = boto3.client("s3", config=conf)
15
16 s3_bucket_name = os.environ.get("S3_BUCKET_NAME")
17 prefix = "input/"
18
19 def create_presigned_url(bucket_name, object_name, content_type):
20     logger.info(
21         f"Generating signed upload URL for {object_name} to {bucket_name}")
22     try:
23         response = s3.generate_presigned_url(
24             "put_object",
25             Params={
26                 "Bucket": bucket_name,
27                 "Key": object_name,
28                 "ACL": "public-read",
29                 "ContentType": content_type,
30             },
31             ExpiresIn=expiration,
32         )
33     except ClientError as e:
34         logger.error(e)
35     return response
```

AWS Lambda Environment Variables

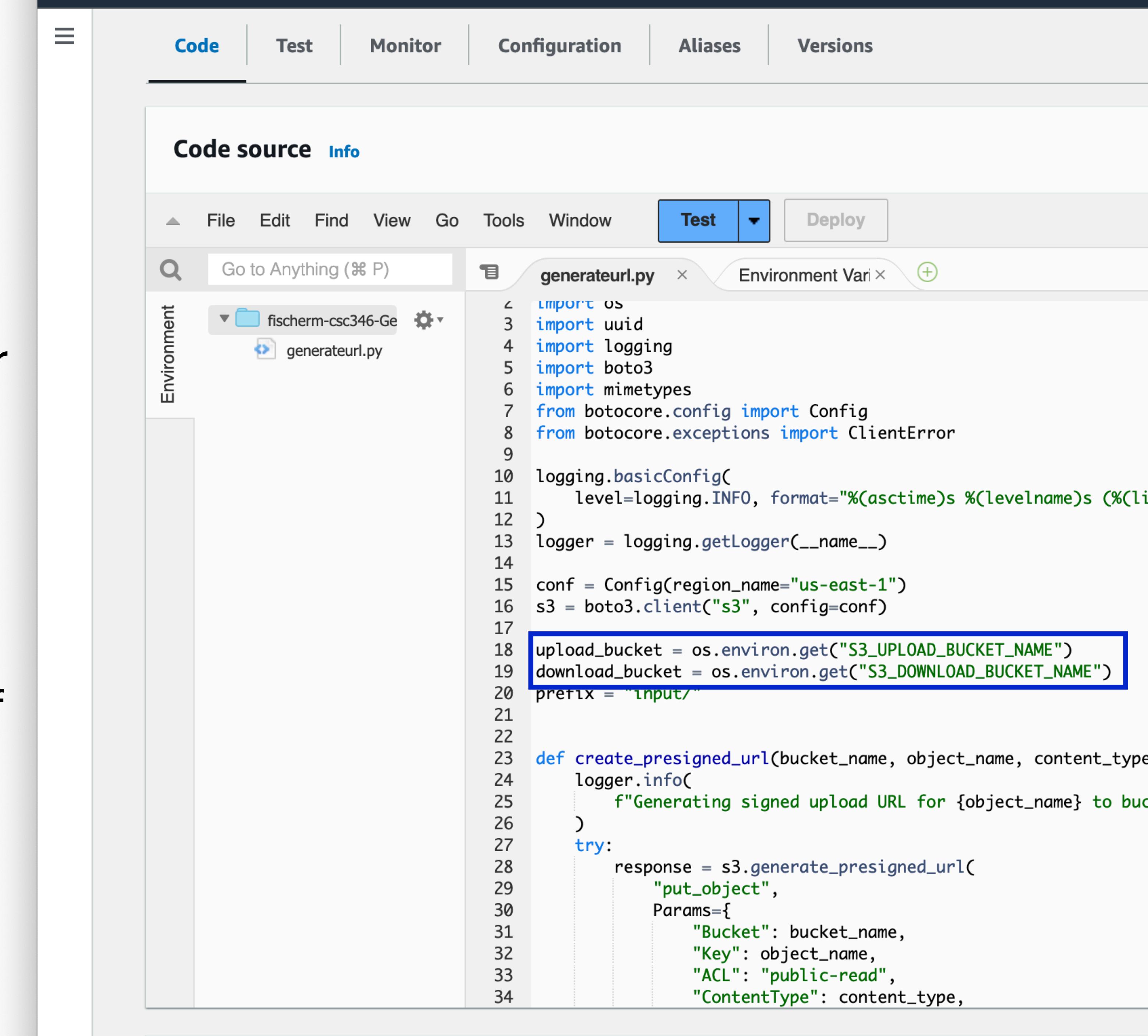
- Just like almost every other code execution method, Lambda provides a way to define Environment Variables

The screenshot shows the AWS Lambda console interface. At the top, the title bar reads "Lambda | us-east-1". The navigation bar includes links to Elastic Container Service, CloudFormation, CloudWatch, EC2, Amazon EventBridge, Systems Manager, Support, and CodeBuild. The main content area displays a Lambda function named "generate-url". Below the function name is a "Layers" section indicating "(1) Layer". A "Related functions" dropdown is set to "Select a function". Two buttons are visible: "+ Add trigger" and "+ Add destination". The "Configuration" tab is selected, showing the "Environment variables" section. This section lists two environment variables: "S3_DOWNLOAD_BUCKET_NAME" with value "fischerm-csc346-download" and "S3_UPLOAD_BUCKET_NAME" with value "fischerm-csc346-upload". Other tabs in the configuration menu include Code, Test, Monitor, Configuration (selected), Aliases, and Versions.

Key	Value
S3_DOWNLOAD_BUCKET_NAME	fischerm-csc346-download
S3_UPLOAD_BUCKET_NAME	fischerm-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

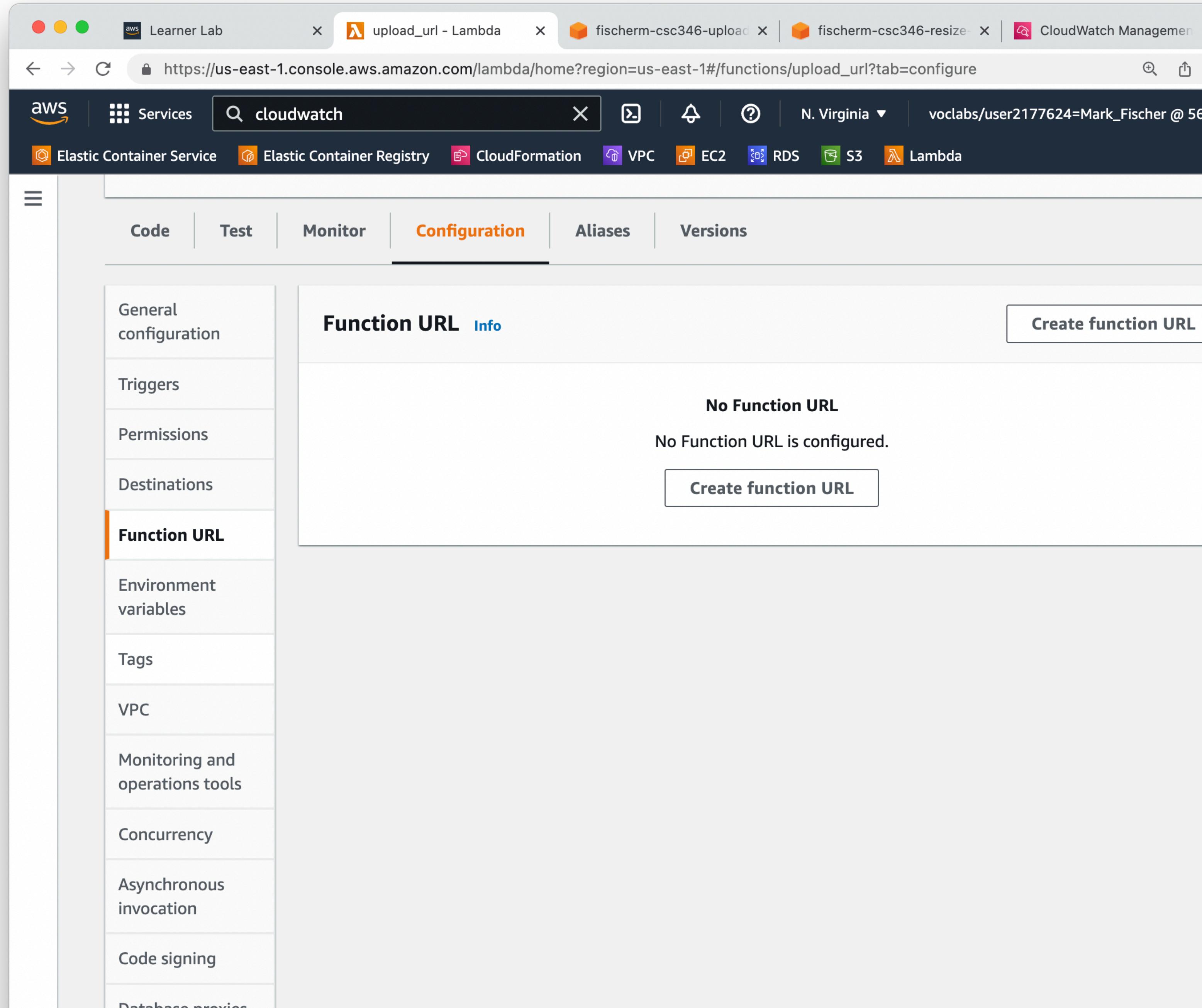


The screenshot shows the AWS Lambda Code source editor interface. The top navigation bar includes tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The Code tab is selected. Below the tabs is a toolbar with File, Edit, Find, View, Go, Tools, Window, a Test dropdown set to 'Test', and Deploy buttons. A search bar says 'Go to Anything (⌘ P)'. On the left, there's a sidebar labeled 'Environment' with a dropdown menu showing 'fischerm-csc346-Ge' and a file icon for 'generateurl.py'. The main area displays a Python script named 'generateurl.py'. The code imports os, uuid, logging, boto3, and mimetypes. It uses botocore config and exceptions. It sets up basic logging with INFO level and a specific format. It creates a logger and configures boto3 with region 'us-east-1'. It retrieves S3 upload and download bucket names from environment variables. A blue box highlights the line 'upload_bucket = os.environ.get("S3_UPLOAD_BUCKET_NAME")'. The code then defines a function 'create_presigned_url' which generates a signed URL for an object in the specified bucket.

```
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 logging.basicConfig(
10     level=logging.INFO, format="%(asctime)s %(levelname)s (%(client_name)s:%(filename)s:%(lineno)d)")
11
12 logger = logging.getLogger(__name__)
13
14 conf = Config(region_name="us-east-1")
15 s3 = boto3.client("s3", config=conf)
16
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 bucket {bucket_name}")
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,
```

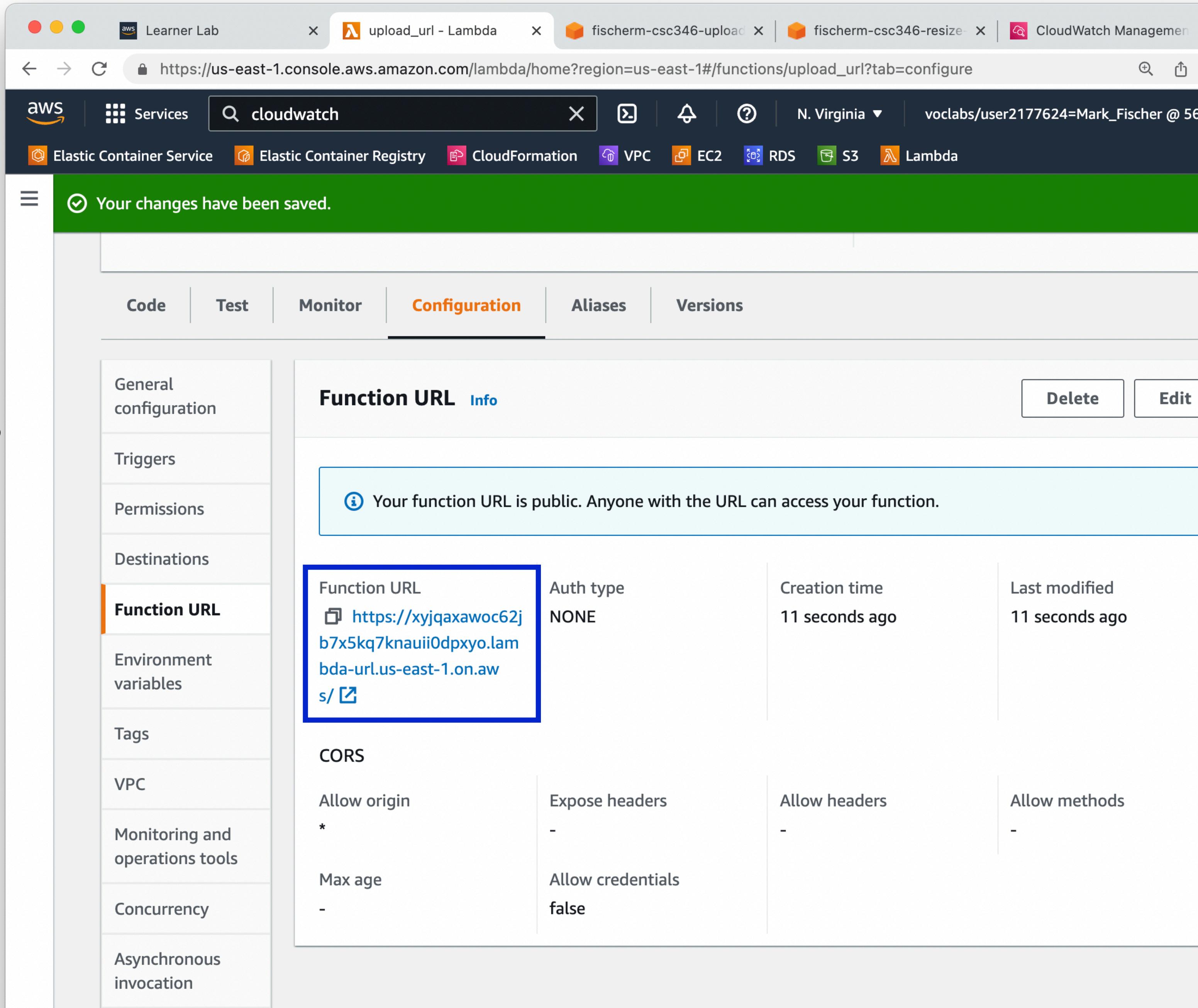
AWS Lambda Function URLs

- 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 the Postman application interface. At the top, there are tabs for Home, Workspaces, and Explore, along with a search bar and user authentication buttons. The main workspace displays a collection named "CSC346 Chat API / Generate Upload URL". A specific request is selected, showing a GET method and the URL <https://xyjqaxawoc62jb7x5kq7knauii0dpxyo.lambda-url.us-east-1.on.aws/?filename=x-wing.jpg>. The "Params" tab is active, showing a single parameter "filename" with the value "x-wing.jpg". Below the URL, the response status is 200 OK, time 604 ms, and size 1.28 KB. The response body is displayed in JSON format:

```
1  {
2      "status": "OK",
3      "upload_url": "https://fischerm-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=FwoGZXIvYXdzEDkaDM6K6tFzj8Ym27tzhSL%2FAWrG1ZyHhojUwg3WxHiDZ%2BlfvhWJdNW4SwDa5bYGyt88EWKTQhUjx9Zk9pj
c%2Bt
%2B5xHRHwwg4SAfwRRwLV3a1hSkW8cwT%2B5woknvBv12jFeIJyY7GQeBCsn1t802cc9td5GiqSbMuxGnoQyeJx1qekPrz%2BfSXIVoqj99r6zEuhK5mqDZviy45x
JxEDRKyHGoH%2F04XZMpvj9p4bs4MlqcevjksVx%2F0pVLElzSZBq8qb1g2ZYkhoxzUxVi2myy6%2FsYMnGTubkHXdhFEbjwmix42iFqZzB9kp%2B0btDRMiD3oDV
A9SMws%2FbT7iYyaG6MFIQyDqCYmz2zJ5C54rCIyhSljjSjohKgbBjIusDvfSseUDYHJ%2BZbH6C1uUkEfDm%2F0tTLA2irr6zZIpohkNQa806iB%2F28iy7hKzg%
3D%3D&Expires=1667778013",
4      "full_url": "https://fischerm-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-1000.jpg",
5      "thumbnail_url": "https://fischerm-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-200.jpg"
6  }
```

At the bottom of the screen, the URL <https://xyjqaxawoc62jb7x5kq7knauii0dpxyo.lambda-url.us-east-1.on.aws/?filename=x-wing.jpg> is displayed again.

AWS Lambda Function URLs

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

The screenshot shows the Postman application interface. At the top, there are tabs for Home, Workspaces (with a dropdown arrow), Explore, a search bar labeled "Search Postman", and account-related buttons for Sign In and Create Account. Below the header, there's a list of collections: "GET Get Chats", "PUT Upload S3 Image" (which is currently selected, indicated by a red border), and "GET Generate Upload URL". To the right of these is a note about "No Environment" and a "Save" button with a dropdown arrow. The main workspace displays a "CSC346 Chat API / Upload S3 Image" collection. A "PUT" request is configured with the URL: `https://fischerm-csc346-upload.s3.amazonaws.com/input/x-wing-6be137b8-d6c2-4018-9da6-6fe338ac6231.jpg?`. The "Params" tab is active, showing parameters: `AWSAccessKeyId=ASIAYFSC5FB6FWU5I2W5&Signature=TSdLaB9v4w2bOKer6bLytTalvxU%3D&x-amz-acl=public-read&content-type=image%2Fjpeg&x-amz-security-token=FwoGXlVYXdzEDgaDPG2TlnxC0mfT1hn%2FiL%2FASZu1Swcl0Z6uzGU2SGzEk3nV1Zomdr4y%2FuvZYXdS%2FS1vJzdIRVV41j0da15x93mLuktARTUCa5x6dk2J9niAW4pN%2BMfG73fsk%2FaYCx34fFwnkHKjf%2B8WApvlemXCrE97tl5zMUURPD2BppHwoZQiar5nqiciJB31XpBkocal%2BNpGkCpoxgD8VRCa58S5C5TNNABljL%2BFF1btpcGPN1YeofSqUoShF%2Bbeg0REfYS8Et3WgA1WyT%2B1rdKlvDN49jnm8IYBrG6mMiaQZ%2BdeZ%2FCQ5KIkvTRgQD5TPd6fCJXaE36YtQyOGVceGAjBlkXomHgEV2GPK%2FCnH5USoHV2m0ICiu86CbBjluck0LuEkiIVzL6TMAUUIO51m06Q%2BodNLtsXlzTfq8eV7av29TfpqSvCPtuD7Knw%3D%3D&Expires=1667775205`. The "Auth" tab is also visible. On the left, there are icons for collections, environments, and other Postman features. On the right, there are buttons for "Send" (blue) and "Cookies" (grey). A status bar at the bottom shows "66" on the left and "100%" on the right.

AWS Lambda

Function URLs

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

The screenshot shows the Postman application interface. At the top, it displays the title "CSC346 Chat API / Upload S3 Image". Below the title, the method is set to "PUT" and the URL is "https://fischerm-csc346-upload.s3.amazonaws.com/input/oldmain-9d9b68fe-a131-44a7-a7b9-16f9f1cde95c.jpg?AWSAccessKeyId=ASIAT5UK7(...". On the right side, there are "Save", "Send", and other toolbar icons. The main area shows tabs for "Params", "Authorization", "Headers (7)", "Body", "Pre-request Script", "Tests", and "Settings". The "Body" tab is currently selected and has a red underline. Below the tabs, there are several radio button options: "none", "form-data", "x-www-form-urlencoded", "raw", "binary" (which is selected and highlighted in orange), and "GraphQL". At the bottom left, there is a file input field containing the file "Old-Main-Milky.jpg" with a delete "X" icon next to it.

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 application interface. At the top, there are tabs for 'Home', 'Workspaces', and 'Explore'. A search bar says 'Search Postman' and there are 'Sign In' and 'Create Account' buttons. Below the tabs, there are three items in a list: 'GET Get Chats', 'PUT Upload S3 Image', and 'GET Generate Upload URL'. The 'Generate Upload URL' item is selected. The main area shows a GET request to 'https://xyjqaxawoc62jb7x5kq7knauii0dpxyo.lambda-url.us-east-1.on.aws/?filename=x-wing.jpg'. Under 'Params', there is a 'filename' parameter set to 'x-wing.jpg'. The 'Body' tab is selected, showing a JSON response:

```
1 {
2     "status": "OK",
3     "upload_url": "https://fischerm-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=FwoGZXIvYXdzEDkaDM6K6tFzj8Ym27tzhSL%2FAWrG1ZyHhojUwg3WxHiDZ%2BlfvhWJdNW4SwDa5bYGyt88EWKTQhUjx9Zk9pj
c%2Bt
%2B5xHRHwwg4SAfwRRwLV3a1hSkW8cwT%2B5woknvBv12jFeIJy7GQeBCsn1t802cc9td5GiqSbMuxGnoQyeJx1qekPrz%2BfSXIVoqj99r6zEuhK5mqDZviy45x
JxEDRKyHGoH%2F04XZMpVj9p4bs4MlqcevjkSvX%2F0pVLElzSZBq8qb1g2ZYkhoxzUxVi2myy%2FsYMnGTubkHXdhFEbjwmix42iFqZzB9kp%2B0btDRMiD3oDV
A9SMws%2FbT7iYyaG6MFIQyDqCYmz2zJ5C54rCIyhSljjSjohKgbBjIusDvfSseUDYHJ%2BZbH6C1uUkEfDm%2F0tTLA2irr6zZIpohkNQa806iB%2F28iy7hKzg%
3D%3D&Expires=1667778013",
4     "full_url": "https://fischerm-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-1000.jpg",
5     "thumbnail_url": "https://fischerm-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-200.jpg"
6 }
```

The status bar at the bottom shows 'Status: 200 OK' and 'Time: 604 ms'. There are also 'Save Response' and 'Runner' buttons.

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 application interface. At the top, there are tabs for 'Home', 'Workspaces', and 'Explore'. A search bar is located at the top right. On the far right, there are buttons for 'Sign In' and 'Create Ac...'. Below the tabs, there are several cards representing API endpoints:

- 'GET Get Chats'
- 'PUT Upload S3 Image'
- 'GET Generate Upload URL'
- 'GET https://fischerm-csc34'

The fourth card, 'GET https://fischerm-csc34', is selected. The URL in the address bar is 'https://fischerm-csc346-resize-bucket.s3.amazonaws.com/x-wing-6e43db7e-05d1-4f5c-9bff-bc5908a194d0-1000.jpg'. The main content area shows a configuration for a 'GET' request to the same URL. The 'Headers' tab is active, showing three checked headers: 'User-Agent' (PostmanRuntime/7.29.2), 'Accept' (*/*), and 'Accept-Encoding' (gzip, deflate, br). Below the headers, the 'Body' tab is active, displaying a large image of a white and red X-wing starfighter fighter jet in a hangar or workshop setting. The image is a high-resolution, slightly blurred photograph. At the bottom of the interface, there are navigation links for 'Find and Replace' and 'Console'.

