

Solution: Uploading a Lambda Function


1. Navigate to the Lambda Console
2. Create a new Lambda Function
3. Use **Author from scratch**, **Python3.8**, and name the function **hello**

Create function [Info](#)

Choose one of the following options to create your function.


Author from scratch

Start with a simple Hello World example.




Use a blueprint

Build a Lambda application from sample code and configuration presets for common use cases.



Browse serverless app repository

Deploy a sample Lambda application from the AWS Serverless Application Repository.



Basic information

Function name
Enter a name that describes the purpose of your function.

hello

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function.

Python 3.8

4. Expand the permissions section
5. Click the **IAM Console** link

Permissions [Info](#)

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an 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


ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

6. Click on **Roles** from the left sidebar
7. Create a new role
8. Set the type of trust identity to Lambda


Create role

1 2 3 4


Select type of trusted entity




AWS service
EC2, Lambda and others



Another AWS account
Belonging to you or 3rd party



Web Identity
Cognito or any OpenID provider



SAML 2.0 federation
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose a use case

Common use cases

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

9. Attach the built-in policy: "AWSLambdaVPCAccessExecutionRole"


Create role 1 2 3 4

▼ **Attach permissions policies**

Choose one or more policies to attach to your new role.

[Create policy](#) [Refresh](#)

Filter policies Showing 1 result

	Policy name ▼	Used as
<input type="checkbox"/>	 AWSLambdaVPCAccessExecutionRole	Permissions policy (1)

10. Name the role **lambda_vpc_role**

Create role 1 2 3 4


Review

Provide the required information below and review this role before you create it.

Role name*
Use alphanumeric and '+,=, @, -, _' characters. Maximum 64 characters.

Role description
Maximum 1000 characters. Use alphanumeric and '+,=, @, -, _' characters.

Trusted entities AWS service: lambda.amazonaws.com

Policies  AWSLambdaVPCAccessExecutionRole [↗](#)

11. Click the **Actions** drop-down menu from the **Function code** section of the Lambda Function Console

12. Select **Upload a .zip file**

hello Throttle Qualifiers ▼ Actions ▼ Select a test event ▼ Test Save

[+ Add trigger](#) [+ Add destination](#)

Function code [Info](#)

File Edit Find View Go Tools Window Save Test ▼

Environment hello / lambda_function.py

```

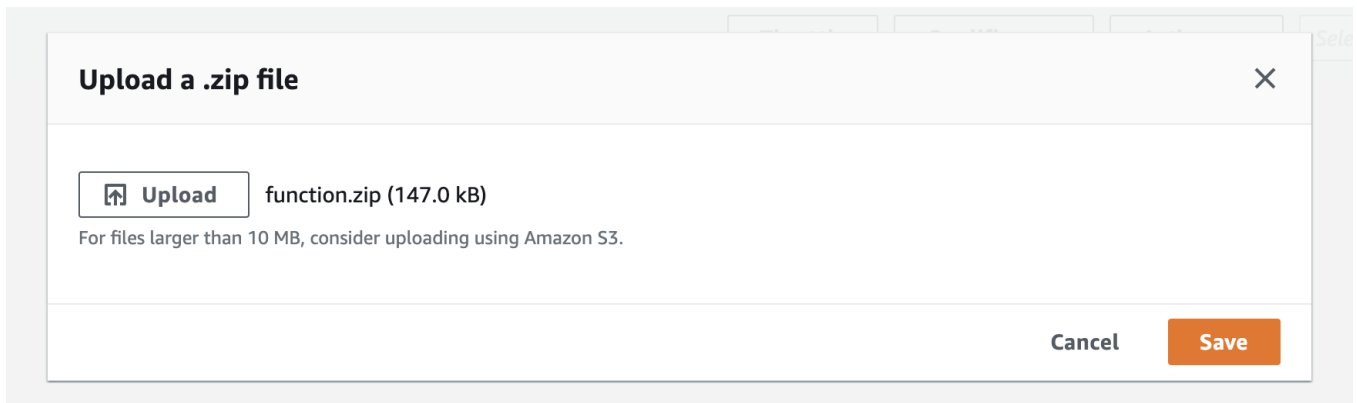
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9

```

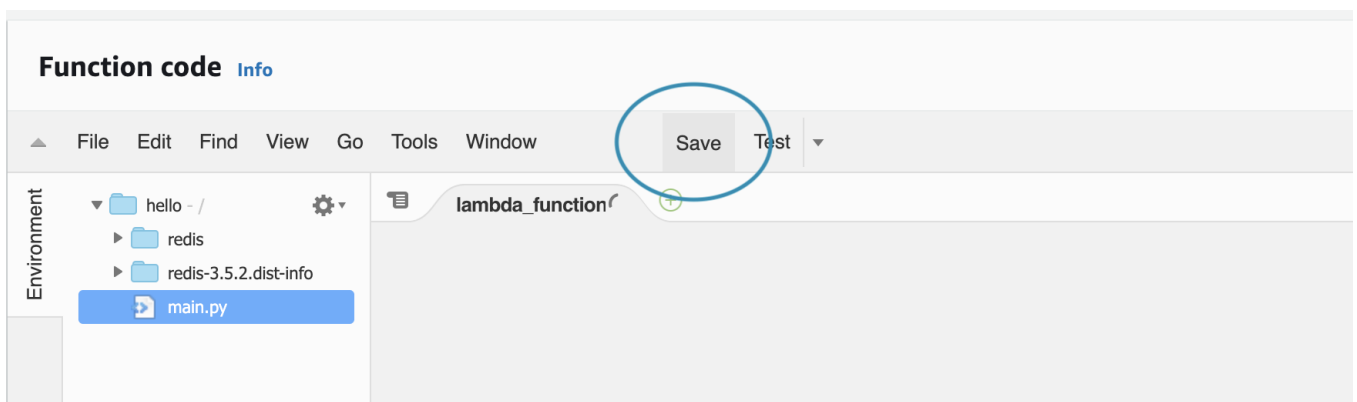
Actions ▲

- Upload a .zip file
- Upload a file from Amazon S3

13. Select the **function.zip** package created in the previous exercise

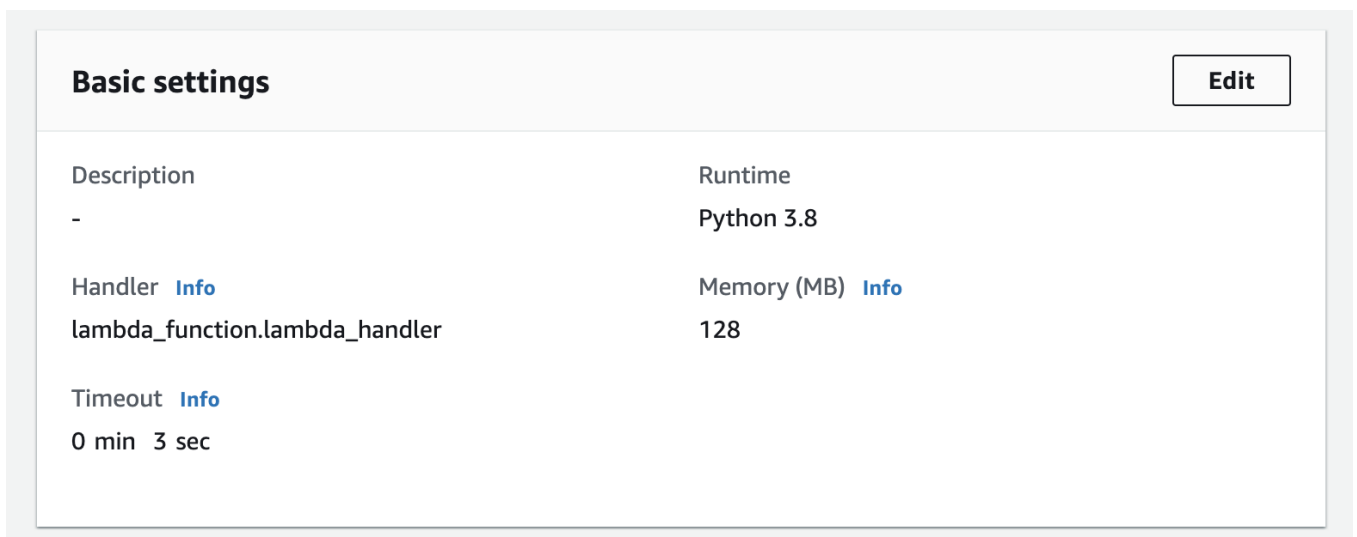


Once selected, you **MUST** click **Save**



14. Scroll down to the **Basic settings** section

15. Click the **Edit** button



16. Set the **Handler** field to **main.handler**

Edit basic settings

Basic settings

Description - *optional*

Runtime

Python 3.8 ▼

Handler [Info](#)

main.handler

Memory (MB) [Info](#)

Your function is allocated CPU proportional to the memory configured.

 128 MB

Timeout [Info](#)

0 min 3 sec

17. Select **Use an existing role**

18. Select **lambda-vpc-role** from the drop-down menu of existing roles

Edit basic settings

Basic settings

Description - *optional*

Runtime

Python 3.8 ▼

Handler [Info](#)

main.handler

Memory (MB) [Info](#)

Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout [Info](#)

0 min 3 sec

Execution role

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

- ☒ Use an existing role
- ☐ 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.

lambda-vpc-role ▼



[View the lambda-vpc-role role](#) on the IAM console.

Setting the REDIS_HOST Environment Variable

1. In another tab, open the [ElastiCache Console](#)
2. Expand the "redis" node
3. Copy the node's endpoint

The screenshot shows the AWS ElastiCache console. On the left is a navigation menu with options like 'Global Datastore', 'Service Updates', 'Reserved Nodes', 'Backups', 'Parameter Groups', 'Subnet Groups', 'Events', and 'ElastiCache Cluster Client'. The main area displays a table of Redis clusters. The table has columns for Cluster Name, Mode, Shards, Nodes, Node Type, Status, Update Action Status, Encryption in-transit, Encryption at-rest, Global Datastore, and Global Endpoint. One cluster is visible with the name 'redis', Mode 'Redis', 0 Shards, 1 node, cache.t2.micro Node Type, and Status 'available'. A mouse cursor is hovering over the 'redis' cluster name.

Cluster Name	Mode	Shards	Nodes	Node Type	Status	Update Action Status	Encryption in-transit	Encryption at-rest	Global Datastore	Global Endpoint
redis	Redis	0	1 node	cache.t2.micro	available	up to date	No	No	-	-

The screenshot shows the AWS ElastiCache console. On the left is a navigation menu with options like Memcached, Redis, Global Datastore, Service Updates, Reserved Nodes, Backups, Parameter Groups, Subnet Groups, Events, and ElastiCache Cluster Client. The main panel is titled '< Name: redis' and has tabs for 'Description' and 'Nodes'. The 'Nodes' tab is active, showing a table with one node:

Node Name	Status	Port	Endpoint	Parameter Group Status	Zone	Created on
redis	available	6379	redis.2ulaxe.0001.usw1.cache.amazonaws.com	in-sync	us-west-1a	May 27, 2020 at 9:44:19 PM UTC-7

4. Click the **Edit** button in the **Environment Variables** section
5. Set the **key** to **REDIS_HOST** and paste the node endpoint as the **value**

The screenshot shows the AWS Lambda console 'Edit environment variables' page. The breadcrumb trail is 'Lambda > Functions > hello > Edit environment variables'. The main heading is 'Edit environment variables'. Below it, a text box explains: 'You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)'. There are two input fields: 'Key' with the value 'REDIS_HOST' and 'Value' with the value 'ilaxe.0001.usw1.cache.amazonaws.com'. A 'Remove' button is next to the value field. Below these fields is an 'Add environment variable' button. At the bottom, there is an 'Encryption configuration' section with a right-pointing arrow. At the very bottom right are 'Cancel' and 'Save' buttons.


Connect The Lambda to the ElastiCache Redis Network

1. Click on the Edit button on the **VPC** section
2. Select **Custom VPC** and select the **default vpc**
3. Select all the subnets under **Subnets**
4. Under the **Security groups** select the **(default)** security group - this enables the Lambda to access the ElastiCache for Redis
5. Click **Save**

VPC [Info](#)**No VPC configuration**

You do not have a VPC configuration for this function

[Edit](#)**VPC**

 When you connect a function to a VPC in your account, it does not have access to the internet unless your VPC provides access. To give your function access to the internet, route outbound traffic to a NAT gateway in a public subnet. [Learn more](#)

VPC connection [Info](#)

Connect to a virtual private cloud (VPC) to access network resources without exposing them to the internet.

- ☐ None
- ☒ Custom VPC

VPC

Choose a VPC for your function to access.

vpc-3754a051 (172.31.0.0/16)

**Subnets**


Select the VPC subnets for Lambda to use to set up your VPC configuration.

		▲
<input type="text" value="Q "/>		
subnet-d3c462b5 (172.31.16.0/20)		us-west-1a
subnet-675aa93d (172.31.0.0/20)		us-west-1c

**Security groups**

Choose the VPC security groups for Lambda to use to set up your VPC configuration. The table below shows the inbound and outbound rules for the security groups that you choose.

		▲
<input type="text" value="Q"/>		
sg-b05075c3 (default)		
default VPC security group		

 **Successfully updated the function hello.**