

The scalability and parallel execution available within Lambda can be used to execute common penetration tests faster.

The following lab will walk through taking a deployment package of nmap and use it as a lambda function.

The simplest Lambda functions consist of only a single file which acts as the lambda handler. If the function only uses libraries that are packaged with the runtime, no code is needed beyond the handler itself.

Non-standard libraries, such as the nmap lib, or executables, such as the nmap binaries, need to be included as part of a "deployment package".

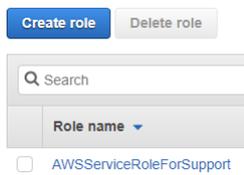
Since Lambda runs in containers on top of the Amazon Linux OS, deployment packages are most likely to be successful if built on an Amazon Linux EC2 Instance.

This function will trigger from an SNS notification, scan the intended host, and store the results to an S3 bucket.

## Roles

In the AWS Console, browse to the "IAM" service and click the "Roles" link.

Then create an IAM Role for the Lambda Function to execute as, by clicking the "Create role" button



Ensure that the Trusted Entity chosen is "Lambda"...

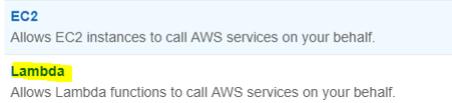
## Create role

Select type of trusted entity



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

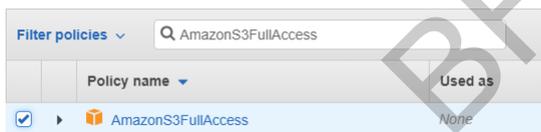
Choose the service that will use this role



Click the "Next: Permissions" button...

Grant the role the following Managed Policies:

- CloudWatchLogsFullAccess
- AmazonSNSReadOnlyAccess
- AmazonS3FullAccess



Click the "Next: Tags" button...

Click the "Next: Review" button...

Name the role: "lambda\_nmap\_role"

You final screen before you click the "Create role" button should look similar to this:

# Create role

## 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**

-  CloudWatchLogsFullAccess [↗](#)
-  AmazonSNSReadOnlyAccess [↗](#)
-  AmazonS3FullAccess [↗](#)

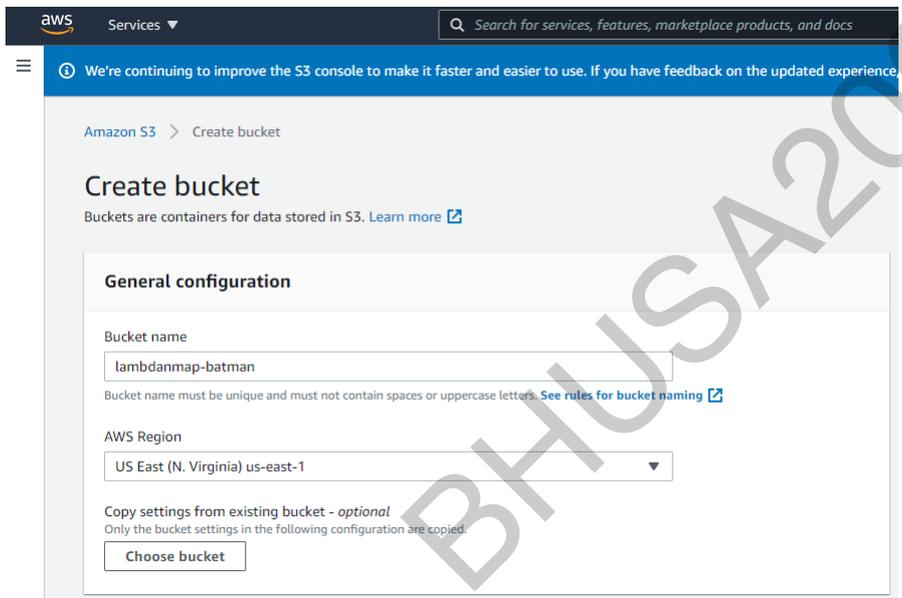
**Permissions boundary** Permissions boundary is not set

## S3 Bucket

Browse to the "S3" Service.

Create an S3 Bucket (e.g. "lambdanmap-batman") in the US East (N. Virginia) us-east-1 region...

Note: S3 Bucket Names are globally unique across all AWS accounts, so replace "batman" with a string that is unique.



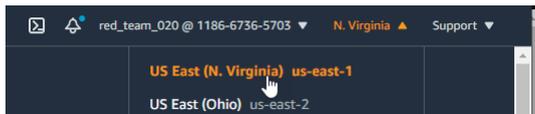
Scroll down to the bottom of the page...

Click the "Create bucket" button.

## Simple Notification Service (SNS)

Browse to the "Simple Notification Service" service within the AWS management console.

Change your region to be "N. Virginia" (aka us-east-1):



We will create an SNS Topic to trigger the function...

Set the "Topic name" to "nmap" and then click the "Next step" button...

aws Services ▾ Search for services, features, marketplace products, and docs [Alt+S] red\_team\_020 @ 1186-6734

Application Integration

# Amazon Simple Notification Service

## Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

### Create topic

Topic name  
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

  
[Next step](#)  
[Start with an overview](#)

Scroll to the bottom, accepting the default settings...

aws Services ▾ Search for services, features, marketplace products, and docs [Alt+S]

Amazon SNS > Topics > Create topic

## Create topic

### Details

**Type** [Info](#)  
Topic type cannot be modified after topic is created

**FIFO (first-in, first-out)**

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

**Standard**

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

**Name**

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (\_).

**Display name - optional**  
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. [Info](#)

Maximum 100 characters, including hyphens (-) and underscores (\_).

► **Encryption - optional**  
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

► **Access policy - optional**  
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic. [Info](#)

► **Delivery retry policy (HTTP/S) - optional**  
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section. [Info](#)

► **Delivery status logging - optional**  
These settings configure the logging of message delivery status to CloudWatch Logs. [Info](#)

► **Tags - optional**  
A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

Cancel [Create topic](#)

Click the "Create topic" button

We should now see a screen similar to the following...

Make note of the ARN of the newly-created SNS topic in a text editor on your laptop (e.g. "arn:aws:sns:us-east-1:118667365703:nmap"):

Now let's create the Lambda function, by browsing to the "Lambda" service within AWS.

A lambda deployment package includes any runtime dependencies and required binaries for the function to execute. Python Deployment Packages can be built by creating a Python Virtual Environment, installing dependencies to the environment, adding a lambda\_function.py entrypoint to the site-packages directory, and zipping the entire site-packages directory.

One such deployment package has been prepared and is available:

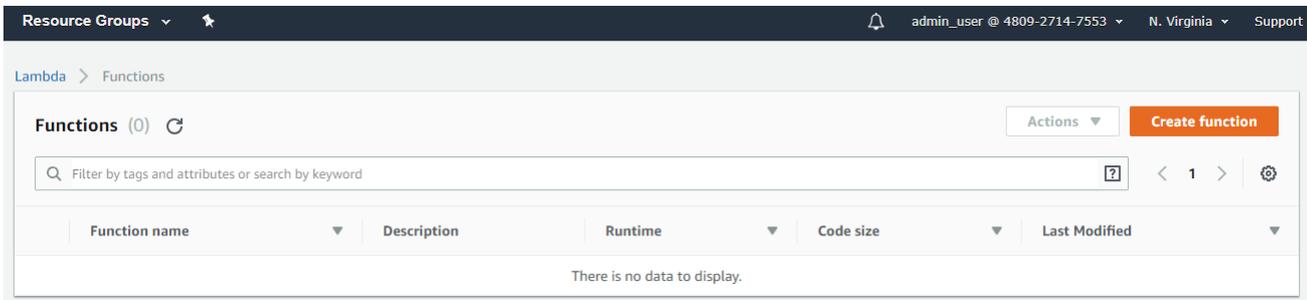
[https://public-astute-cloud-20200813-935672326788.s3.amazonaws.com/nmap\\_deployment-v1\\_1\\_0.zip](https://public-astute-cloud-20200813-935672326788.s3.amazonaws.com/nmap_deployment-v1_1_0.zip)

- Extract the files from this zip folder
- Open the "lambda\_function.py" within a text editor locally on your laptop
- Review the lambda function in the package: nmap\_deployment-v1\_1\_0.zip/lambda\_function.py
- Note that the S3 bucket to store the results into is retrieved from an Environment Variable on line 5.
- Note that nmap binary built into the deployment package at nmap\_deployment-v1\_1\_0.zip/bin/nmap is added to the path on line 20.

In the AWS Console open the Lambda service

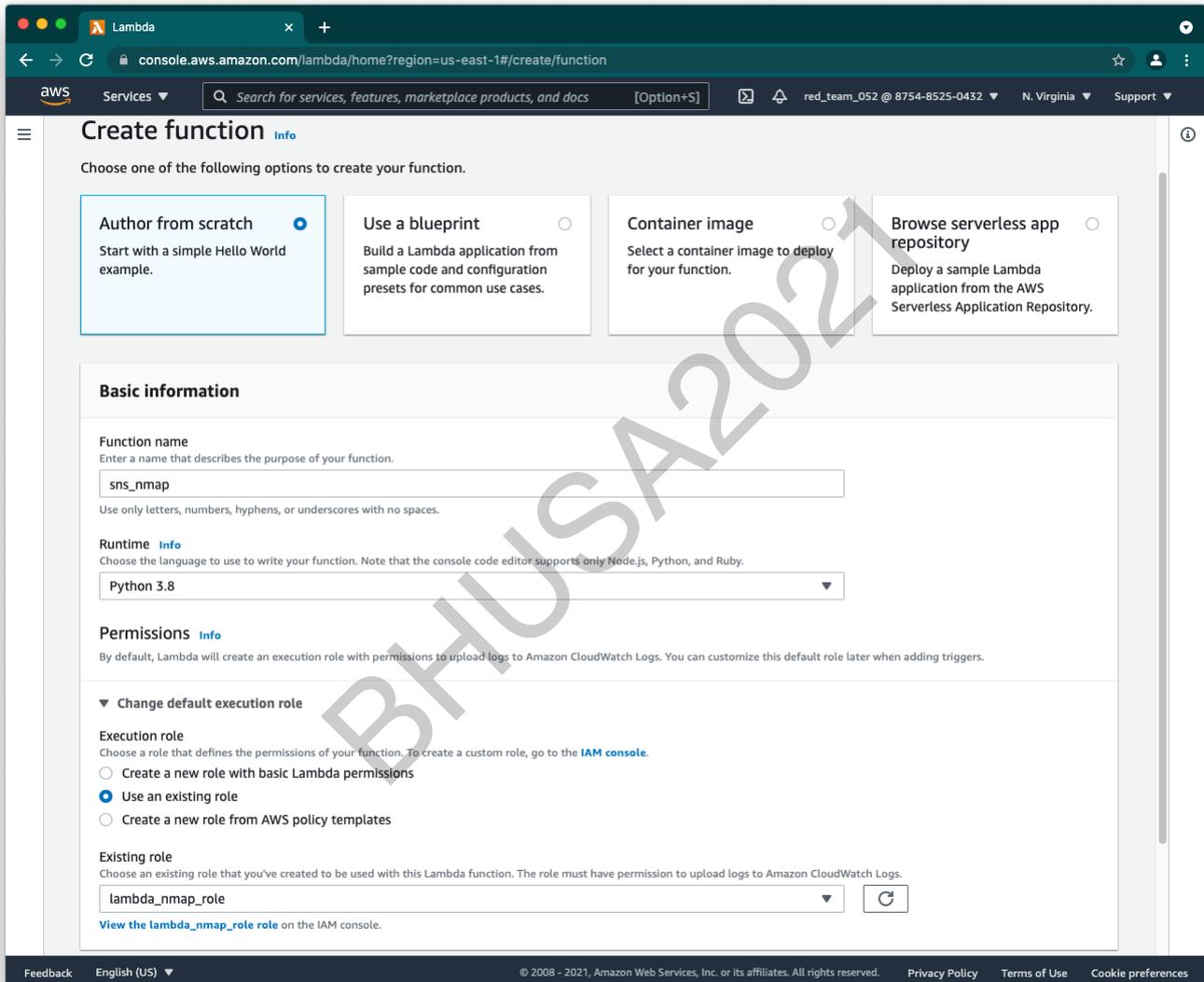
Ensure you are in the US East (N. Virginia) us-east-1 region:

Click the "Create function" button to create a new Lambda Function from Scratch...

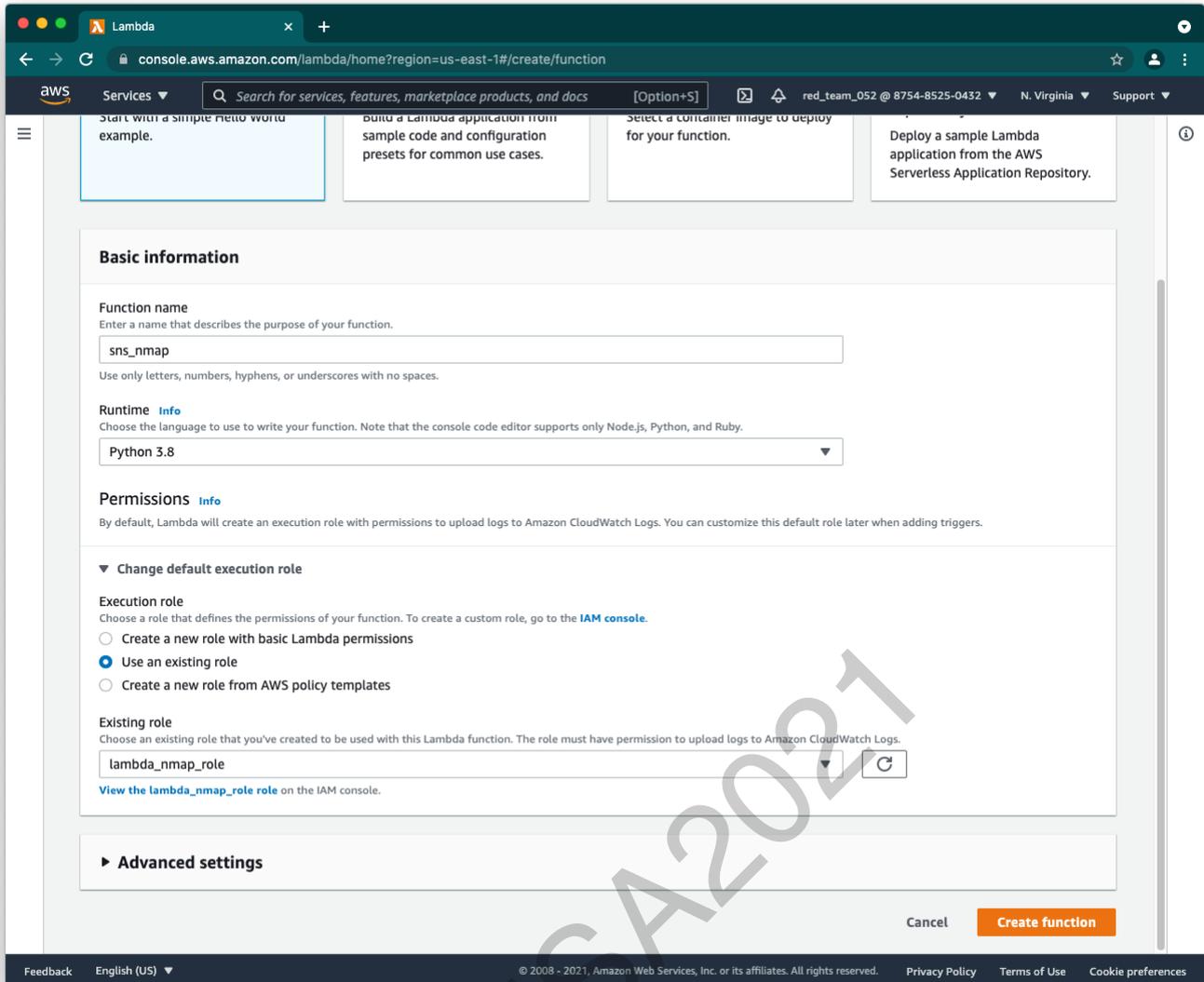


Use the following options for Name, Runtime, and Role.

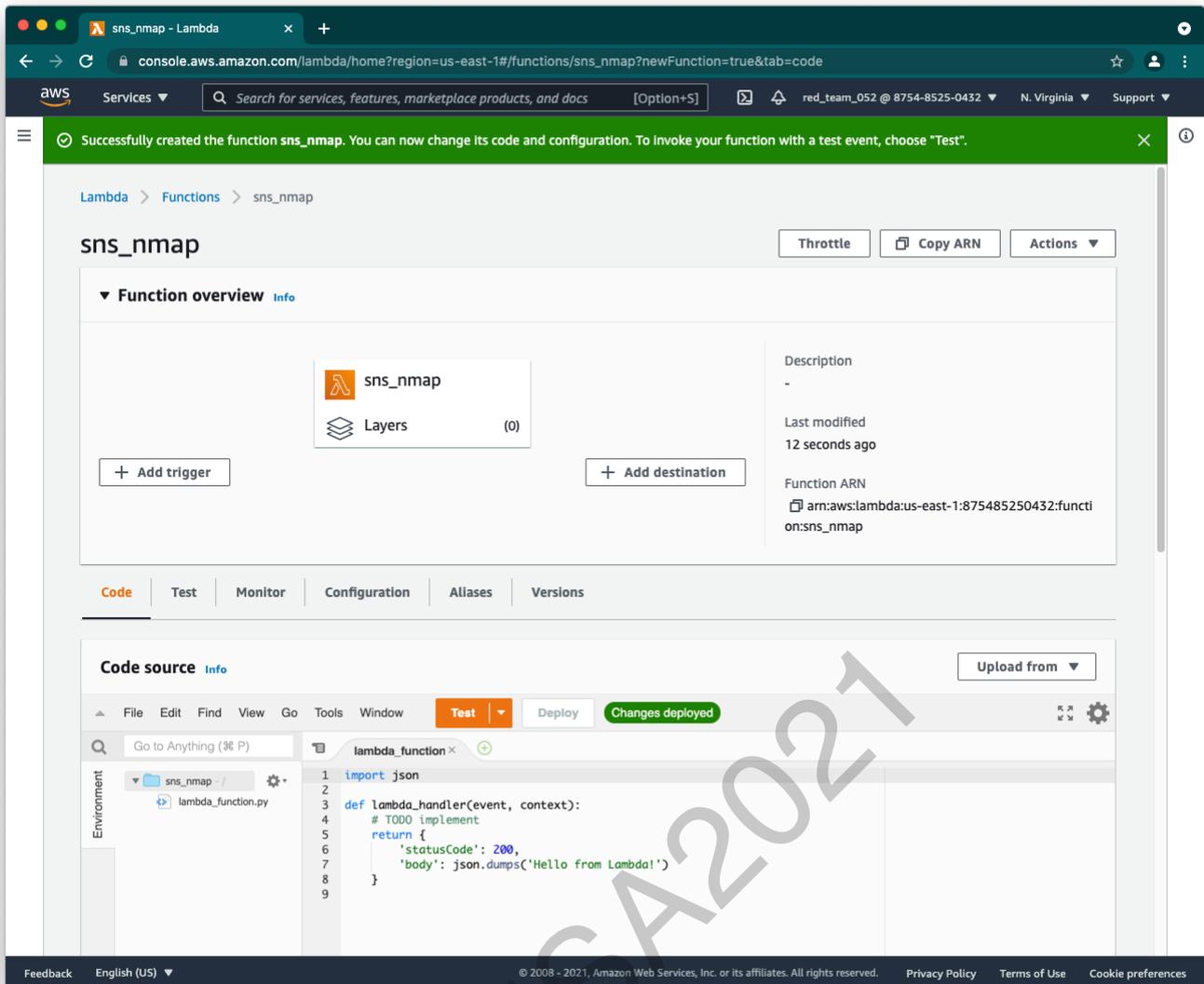
- Function name: sns\_nmap
- Runtime: Python 3.8
- Permissions
  - Execution role: Choose an existing role
    - Existing role: lambda\_nmap\_role



Click the "Create function" button at the bottom of the page...

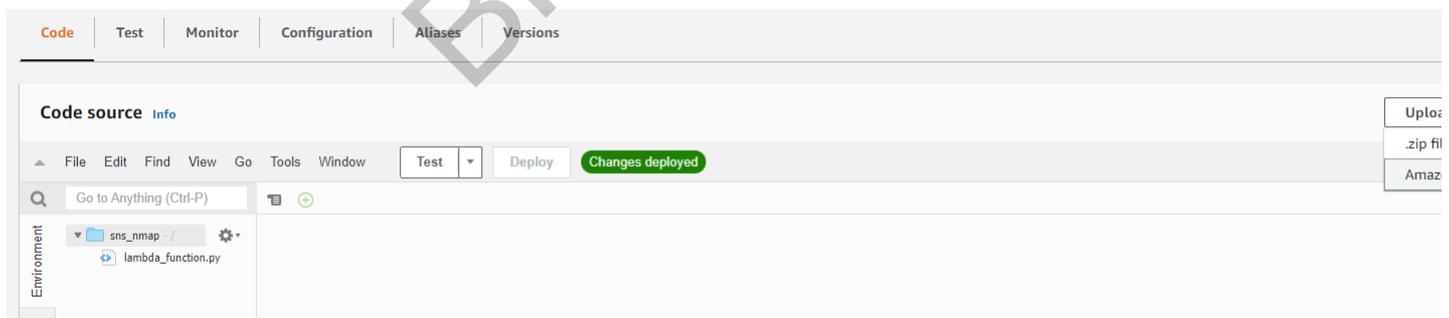


We should now see a screen similar to this...



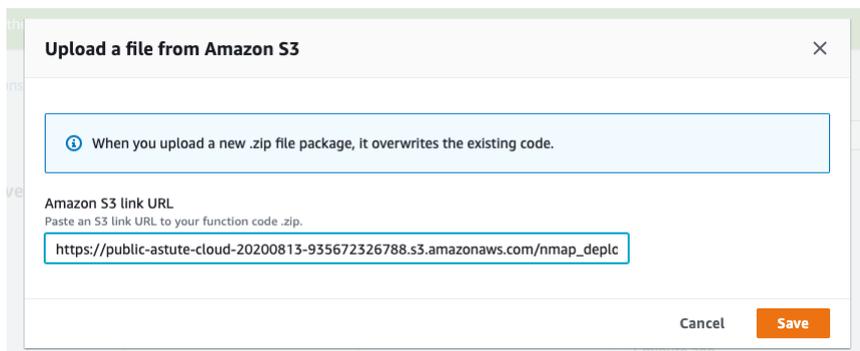
Click the "Upload from" button...

Then Click the "Amazon S3 location" link...



Input the following URL...

- Amazon S3 link URL: [https://public-astute-cloud-20200813-935672326788.s3.amazonaws.com/nmap\\_deployment-v1\\_1\\_0.zip](https://public-astute-cloud-20200813-935672326788.s3.amazonaws.com/nmap_deployment-v1_1_0.zip)



Click the "Save" button.

Click the "Configuration" link

Click the "Environment variables" link

The screenshot shows the AWS Lambda console Configuration tab. The left sidebar has a menu with options: General configuration, Triggers, Permissions, Destinations, Environment variables (highlighted), and Tags. The main content area is titled "Environment variables (0)" and contains a table with columns "Key" and "Value". Below the table, it says "No environment variables" and "No environment variables associated with this function." There is an "Edit" button at the bottom right.

Create an environment variable named "s3bucket" with a value of the exact name of the S3 bucket you created earlier.

Click the "Edit" button

Click the "Add environment variable" button

- Key: s3bucket
- Value: lambdanmap-your\_unique\_string (e.g. lambdanmap-batman)

The screenshot shows the "Edit environment variables" dialog in the AWS Lambda console. It has a title bar with "aws Services" and a search bar. The breadcrumb is "Lambda > Functions > sns\_nmap > Edit environment variables". The main heading is "Edit environment variables". Below it, there is a section "Environment variables" with a description: "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 is a table with one row: Key: s3bucket, Value: lambdanmap-batman, and a "Remove" button. Below the table is an "Add environment variable" button. At the bottom, there is an "Encryption configuration" section with a right-pointing arrow. At the bottom right, there are "Cancel" and "Save" buttons.

Click the "Save" button

Increase the memory allotment and maximum execution time...

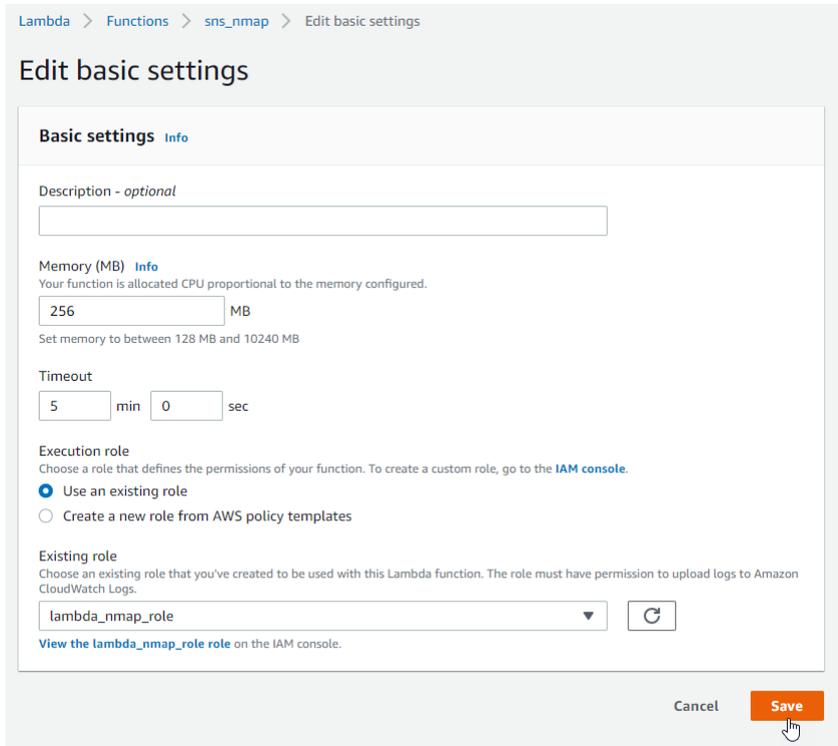
Click the "Configuration" link

Then, click the "General Configuration" link

The screenshot shows the AWS Lambda console Configuration tab. The left sidebar has a menu with options: General configuration (highlighted), Triggers, Permissions, Destinations, Environment variables, and Tags. The main content area is titled "General configuration info". It contains a table with columns: Description, Memory (MB), and Timeout. The Description is "-", Memory (MB) is 128, and Timeout is 0 min 3 sec. Below the table, there is a blue box with an information icon and the text "AWS Compute Optimizer" and "Opt in to see memory recommendations for your Lambda functions. [View details](#)".

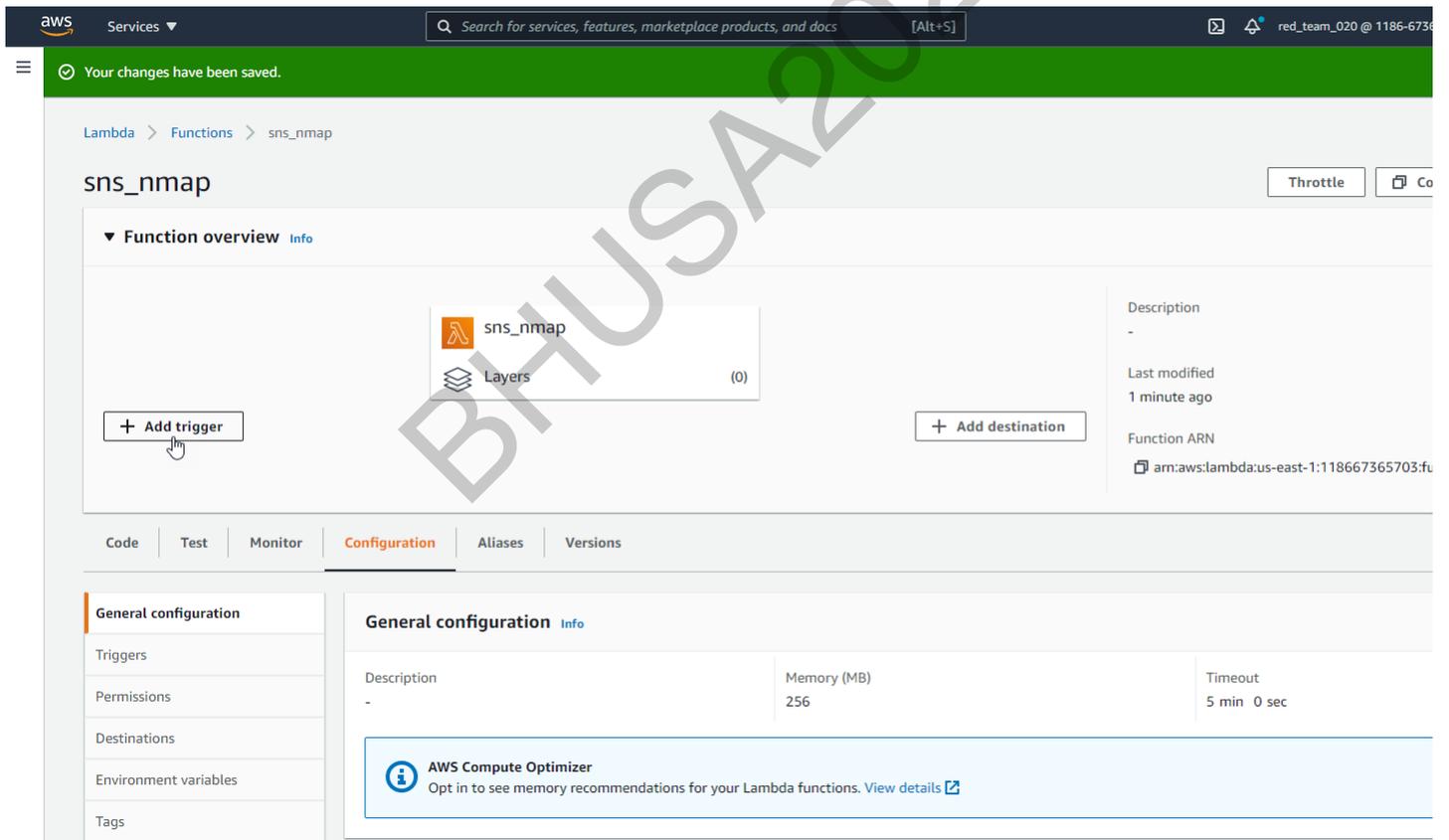
Click the "Edit" button...

- Memory: 256 MB
- Timeout: 5 min 0 sec



Click the "Save" button

We should now see a screen similar to the following...



Click the "+ Add trigger" button

And Select "SNS"...

## Add trigger

**Trigger configuration**

Select a trigger

Q

- API Gateway  
api application-services aws serverless
- AWS IoT  
aws devices iot
- Alexa Skills Kit  
alexa iot
- Alexa Smart Home  
alexa iot
- Apache Kafka  
aws stream
- Application Load Balancer  
aws load-balancing
- CloudFront  
aws cdn edge
- CloudWatch Logs  
aws logging management-tools
- CodeCommit  
aws developer-tools git
- Cognito Sync Trigger  
authentication aws identity mobile-services sync
- DynamoDB  
aws database nosql
- EventBridge (CloudWatch Events)  
aws events management-tools
- Kinesis  
analytics aws streaming
- MQ  
aws messaging multi-protocol
- MSK  
aws cluster
- S3  
aws storage
- SNS**  
aws messaging notifications pub-sub push
- SQS  
aws queue

Select the "SNS topic" of "nmap"...

Lambda > Add trigger

## Add trigger

**Trigger configuration**

SNS  
aws messaging notifications pub-sub push

SNS topic  
Select the SNS topic to subscribe to.

Q

nmap

Lambda will add the necessary permissions for Amazon SNS to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Enable trigger  
Enable the trigger now, or create it in a disabled state for testing (recommended).

Cancel Add

And then it should auto-populate the associated ARN for the selected SNS topic...

Lambda > Add trigger

## Add trigger

**Trigger configuration**

SNS  
aws messaging notifications pub-sub push

SNS topic  
Select the SNS topic to subscribe to.

arn:aws:sns:us-east-1:118667365703:nmap

Lambda will add the necessary permissions for Amazon SNS to invoke your Lambda function from this trigger. [Learn more about the Lambda permissions model.](#)

Enable trigger  
Enable the trigger now, or create it in a disabled state for testing (recommended).

Cancel **Add**

Click the "Add" button to add the trigger.

We should now see a banner at the top of the screen that looks similar to the following...

The screenshot shows the AWS Lambda console for the function 'sns\_nmap'. A green banner at the top of the function page reads: 'The trigger nmap was successfully added to function sns\_nmap. The function is now receiving events from the trigger.'

Let's scan the IP address associated with the website "opback.com" using this new lambda function.

```
root@ip-10-0-1-188:/shared# ping -c3 opback.com
PING opback.com (13.82.46.24) 56(84) bytes of data.

--- opback.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 1999ms
```

NOTE: The current IP address of the opback.com domain may be different for you (e.g. 13.82.46.24).

Now use the AWS CLI on your machine to push a message to SNS, triggering the lambda functions scan (replacing the string "ip.ip.ip" with the IP address of the opback.com website)

```
root@ip-10-0-1-215:/shared# cd /shared
root@ip-10-0-1-188:/shared# aws sns list-topics --region us-east-1
{
  "Topics": [
    {
      "TopicArn": "arn:aws:sns:us-east-1:480927147553:nmap"
    }
  ]
}
root@ip-10-0-1-215:/shared# aws sns publish --topic-arn arn:aws:sns:us-east-1:480927147553:nmap --region us-east-1 --message ip.ip.ip
{
  "MessageId": "1842feba-86e4-5e96-8058-ad614c244a86"
}
```

After the time it takes for lambda to invoke and run the scan, approximately 5 minutes, you will find the results written to your specified S3 bucket (e.g. "lambdanmap-batman").

The screenshot shows the AWS S3 console interface. The 'Overview' tab is selected. A search bar contains the text 'Type a prefix and press Enter to search. Press ESC to clear.' Below the search bar are buttons for 'Upload', 'Create folder', 'Download', and 'Actions'. A table lists the contents of the bucket:

Name	Last modified	Size	Storage class
13.82.46.24-1601159457	Sep 26, 2020 4:30:58 PM GMT-0600	385.0 B	Standard

The contents of the file should be similar to the following:

```
1 host;hostname;hostname_type;protocol;port;name;state;product;extrainfo;reason;version;
2 13.82.46.24;;;tcp;22;ssh;open;OpenSSH;"Ubuntu Linux; protocol 2.0";syn-ack;7.2p2 Ubunt
3 13.82.46.24;;;tcp;80;http;open;Apache httpd;(Ubuntu);syn-ack;2.4.18;10;cpe:/a:apache:h
4 13.82.46.24;;;tcp;443;https;closed;;;conn-refused;;3;
```

Thousands of these functions can be invoked simultaneously, significantly speeding up nmap, DNS discovery, directory busting, and similar tasks.

## References

Check out the following references for more information:

- Run a Serverless "Hello, World!" - <https://aws.amazon.com/getting-started/tutorials/run-serverless-code/>
- Create a Simple Lambda Function - <https://docs.aws.amazon.com/lambda/latest/dg/get-started-create-function.html>
- Getting Started - AWS Lambda - <https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html>

BHUSA2021