Email Masking Service on AWS

Project Overview

This project creates an email masking service that forwards messages while protecting your real email address. It combines multiple AWS services to provide anonymous communication.

What You'll Build

- **Email Masking System** - Create disposable email addresses that forward to your real inbox
- **Mail Storage**- Archive all received emails for future reference
- **Email Database** - Store your collection of masked email addresses
- **Serverless Architecture** - Fully automated email processing without servers

Architecture Overview

This solution uses these AWS microservices:

- **Route 53** manages domain registration and DNS settings
- **SES (Simple Email Service)** handles email receiving and forwarding
- **Lambda Functions** process incoming emails and create new masked addresses
- **DynamoDB** stores the mapping between masked and real email addresses
- **S3** archives all received emails

Prerequisites

Software Requirements

- Web browser to access AWS Console
- Email account to receive forwarded messages
- Text editor for DNS configuration

Cost

- Domain registration: ~$3-15/year for .click domains
- SES: Free for receiving emails, $0.10 per 1,000 emails sent
- Lambda: Free tier includes 1M free requests per month
- DynamoDB: Free tier includes 25GB storage
- S3: ~$0.023/GB-month for storage

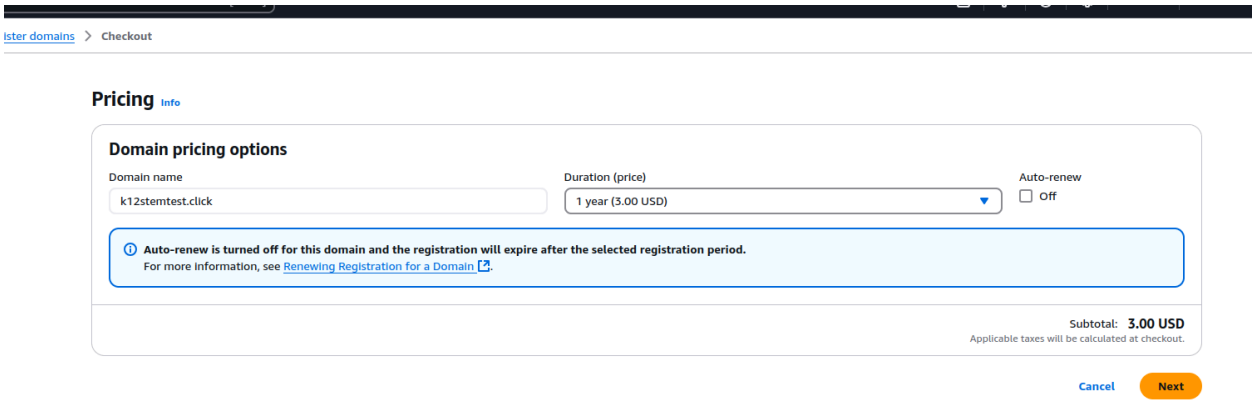
Knowledge

- Basic understanding of AWS services
- Familiarity with email systems and DNS
- Basic understanding of JSON for testing

Project Start

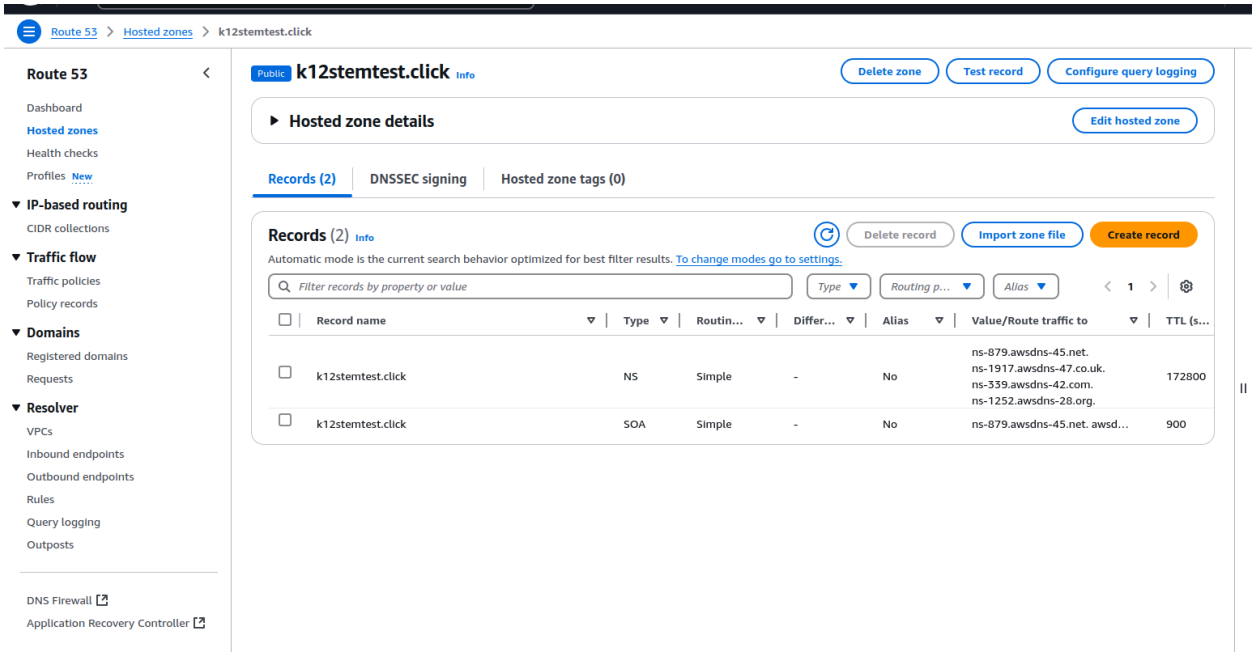
Domain Registration

Use .click domains. Turn **off** auto-renew.

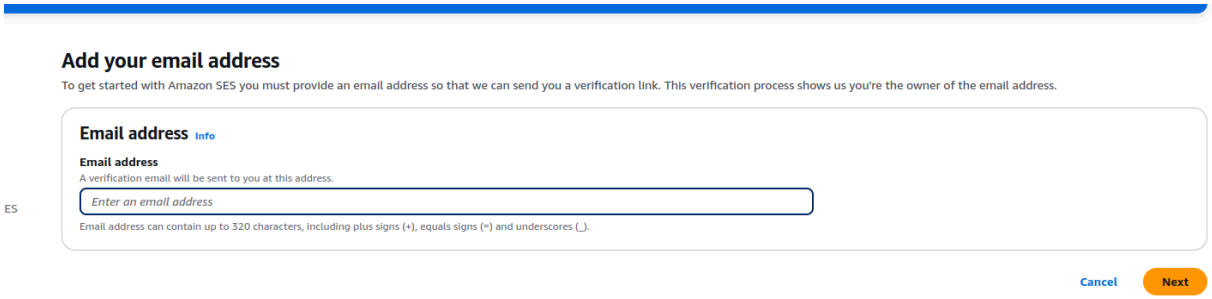


Fill out all the information required for the next page, then click submit.

Now wait a few minutes.

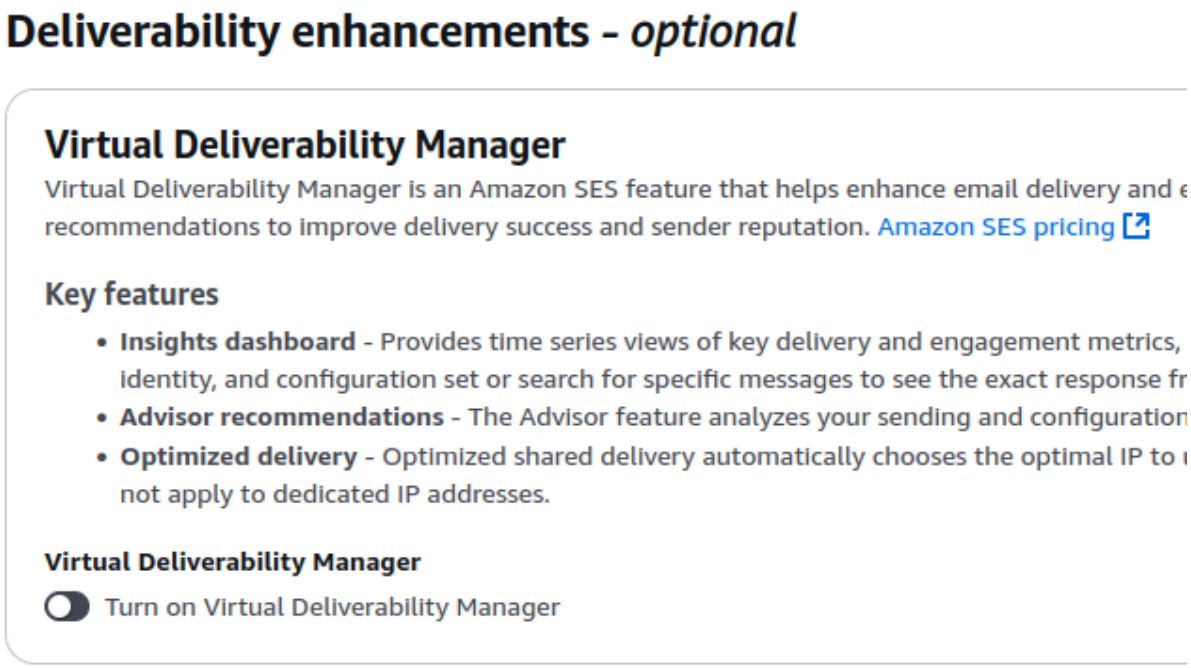


Amazon SES



On the next page, put your domain in the sending domain. Ignore the MAIL FROM domain

Skip this as well

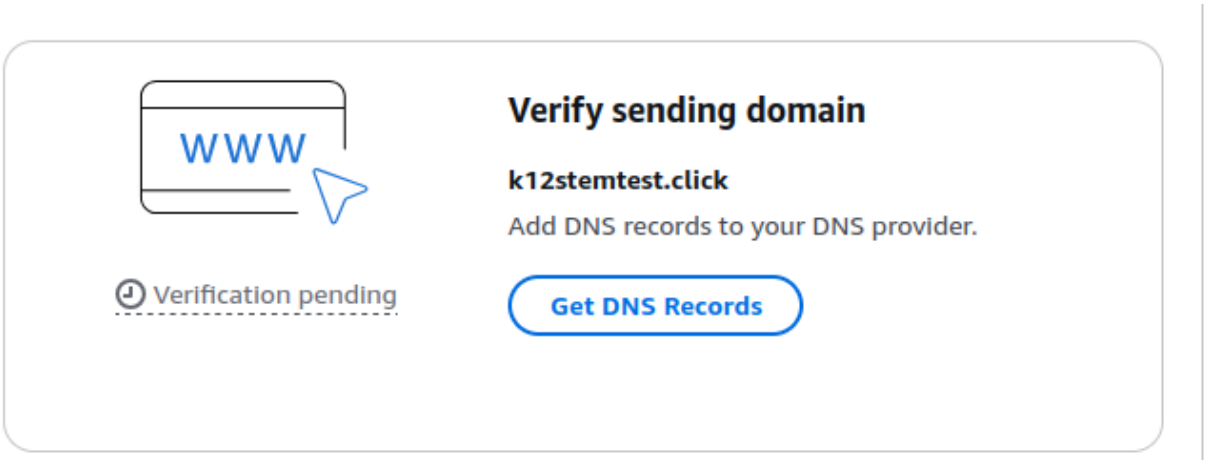


Now hit confirm and create your SES.

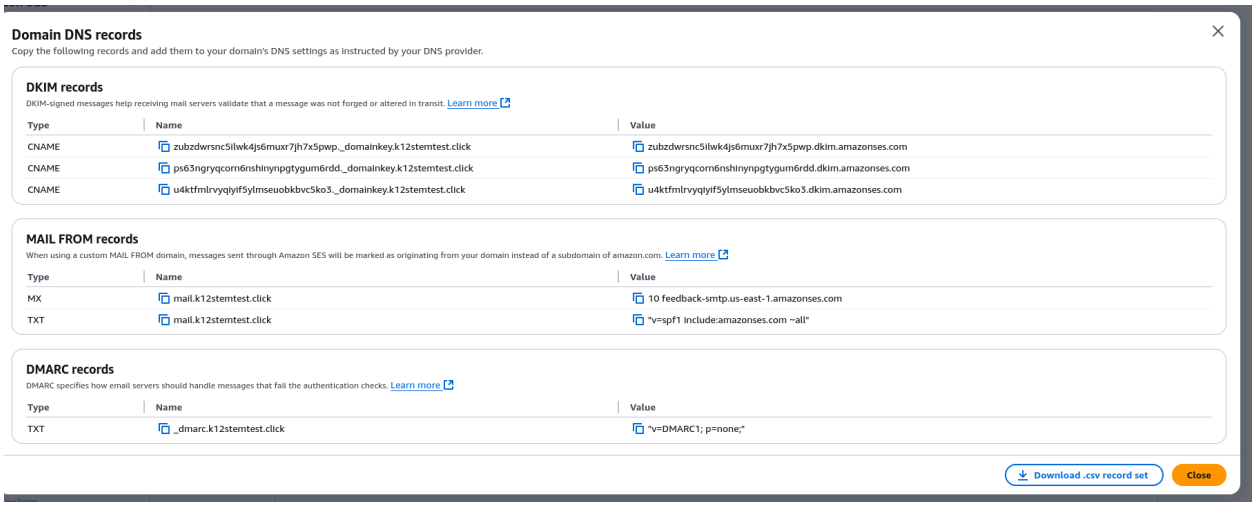
Now go press the confirmation email link in your personal email that you put in to sign up for the SES.

Verify Domain with SES

Click on the get DNS Records on your SES page



Stay on this page and open up a notepad (IGNORE THE MAIL FROM records)



Domain DNS records

Copy the following records and add them to your domain's DNS settings as instructed by your DNS provider.

DKIM records

DKIM-signed messages help receiving mail servers validate that a message was not forged or altered in transit. Learn more

Type	Name	Value
CNAME	zubzdwrsnc5ilwk4js6muxr7jh7x5pwp._domainkey.k12stemtest.click	zubzdwrsnc5ilwk4js6muxr7jh7x5pwp.dkim.amazonses.com
CNAME	ps63ngryqcorn6nshinynpgtygum6rdd._domainkey.k12stemtest.click	ps63ngryqcorn6nshinynpgtygum6rdd.dkim.amazonses.com
CNAME	u4ktfmlrvyqiyif5ylmseuobkbvc5ko3._domainkey.k12stemtest.click	u4ktfmlrvyqiyif5ylmseuobkbvc5ko3.dkim.amazonses.com

MAIL FROM records

When using a custom MAIL FROM domain, messages sent through Amazon SES will be marked as originating from your domain instead of a subdomain of amazon.com. Learn more

Type	Name	Value
MX	mail.k12stemtest.click	10 feedback-smtp.us-east-1.amazonses.com
TXT	mail.k12stemtest.click	"v=spf1 include:amazonses.com ~all"

DMARC records

DMARC specifies how email servers should handle messages that fail the authentication checks. Learn more

Type	Name	Value
TXT	_dmarc.k12stemtest.click	"v=DMARC1; p=none;"

Download .csv record set Close

Now replace the values marked with [ ] in here with your own on a notepad. Do not include the brackets [ ] in the final result.

; Zone file for k12stemtest.click

$TTL 3600 ; Default TTL of 1 hour

$ORIGIN [domain name].

; DKIM records for Amazon SES

[cname name] IN CNAME [cname value].

[cname name] IN CNAME [cname value].

[cname name] IN CNAME [cname value].

; MAIL FROM domain settings

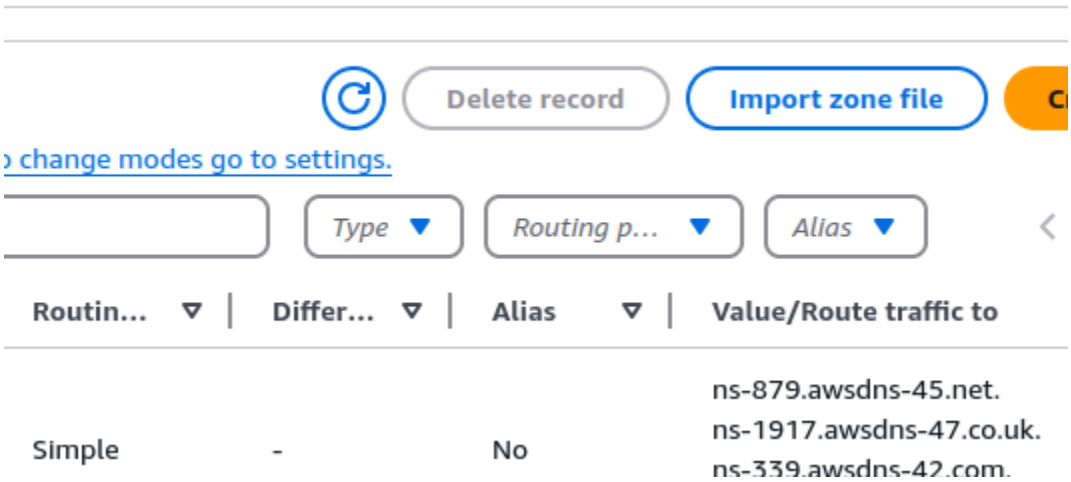
IN MX 10 inbound-smtp.us-east-1.amazonaws.com.

IN TXT "v=spf1 include:amazonses.com ~all"

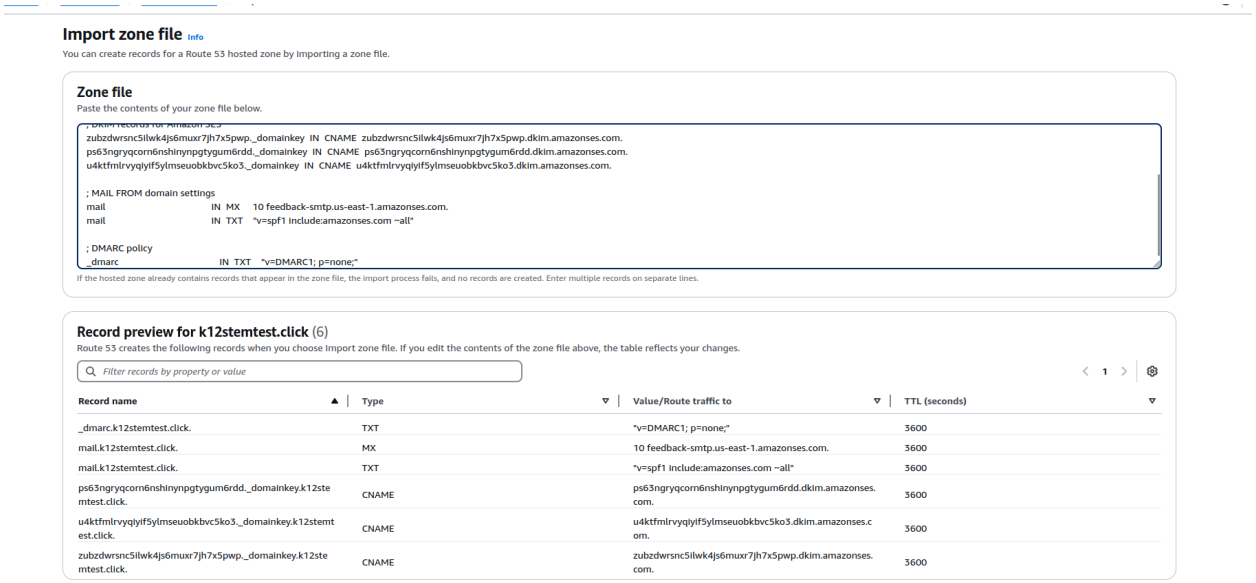
; DMARC policy

_dmarc IN TXT "v=DMARC1; p=none;"

Go to your Route 53
Now import zone file



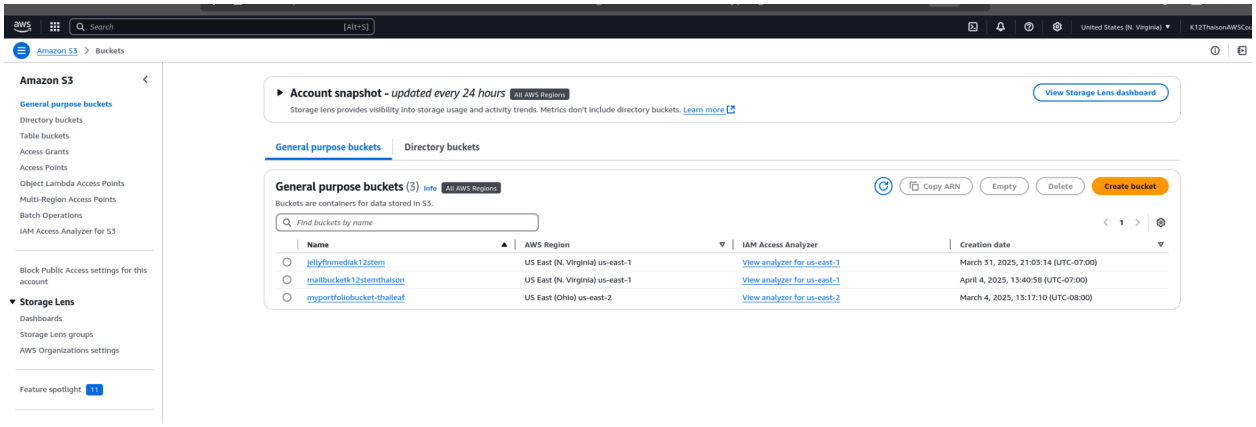
Put in your zone file that you edited

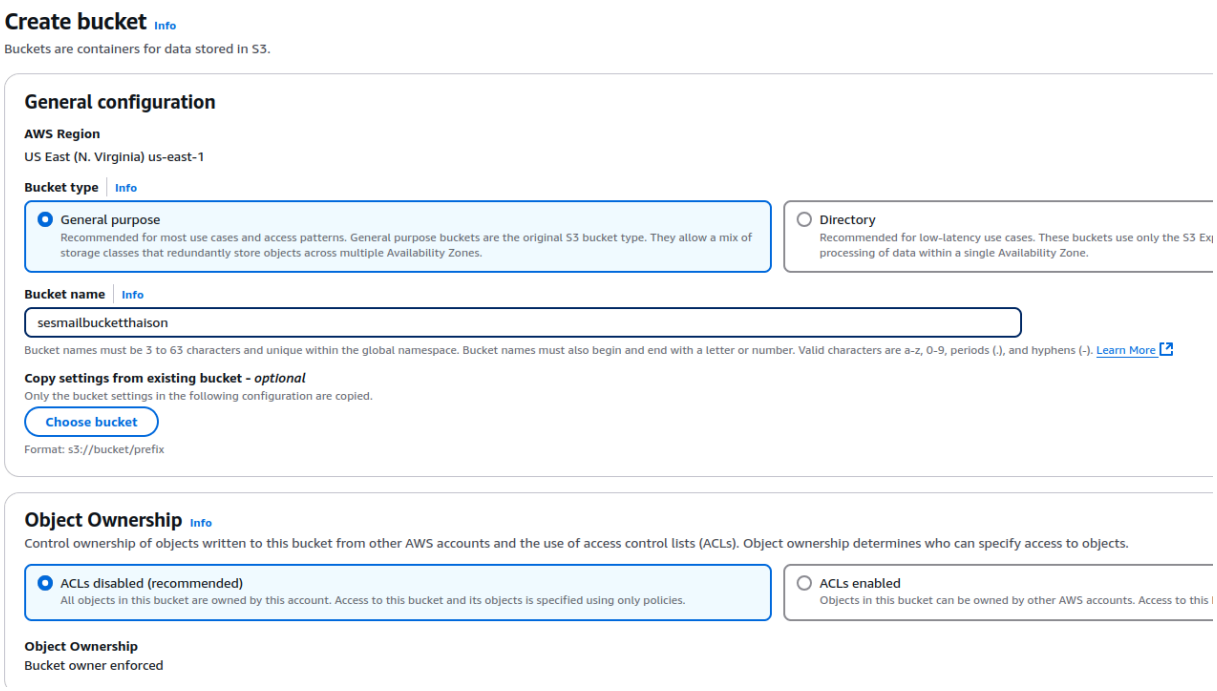


Your domain is now configured to send emails to amazon SES

Creating an S3 Bucket to store emails

Go to your main page and create an S3 bucket. Use default settings, and name it something that makes sense.

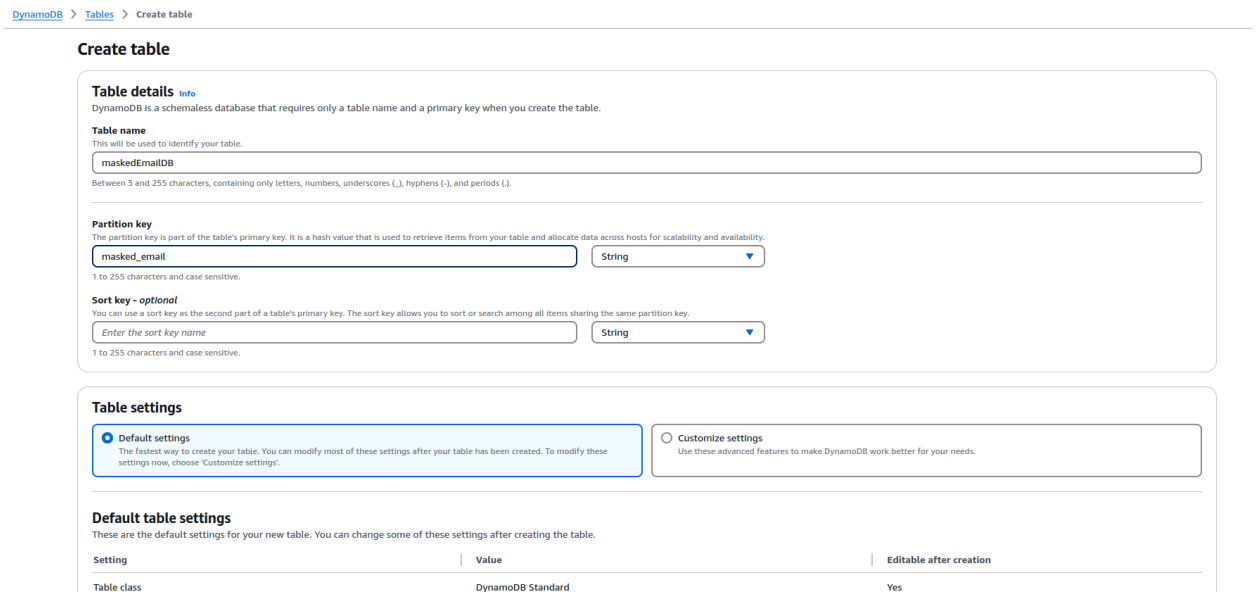




Using DynamoDB

We need to create the table that stores the translations of masked emails to real emails.

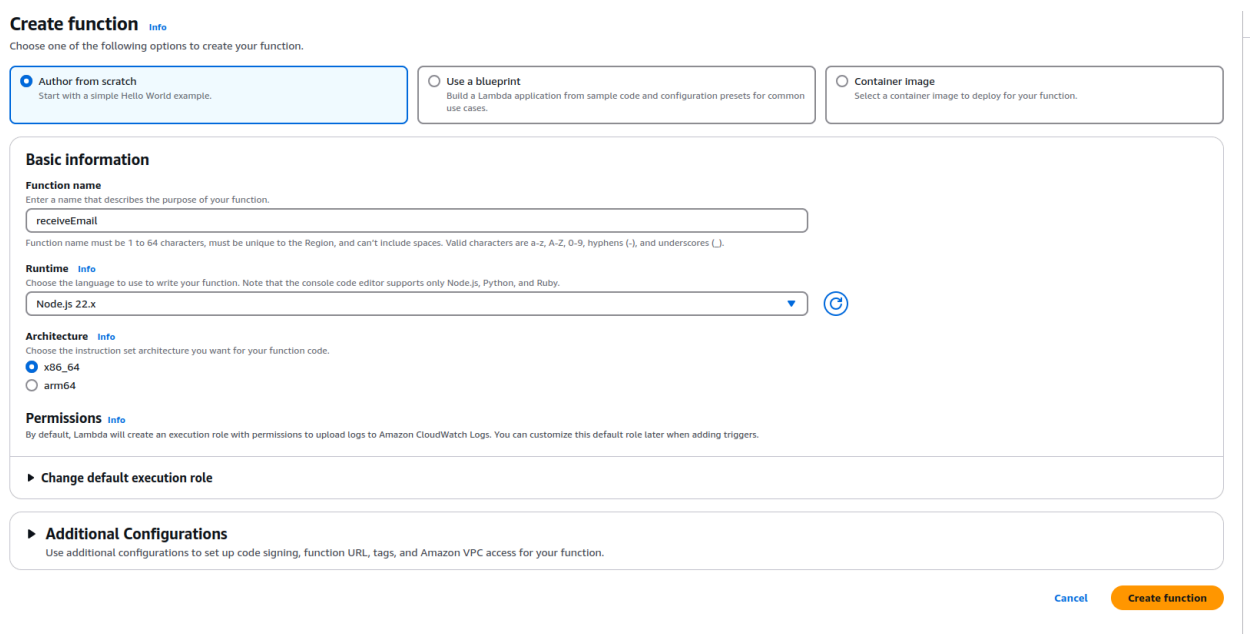
Go to the dynamo db homepage by using the search bar.
Create a table. Name it something and keep it all the defaults. The partition key will be masked_email, and it will be of type string.



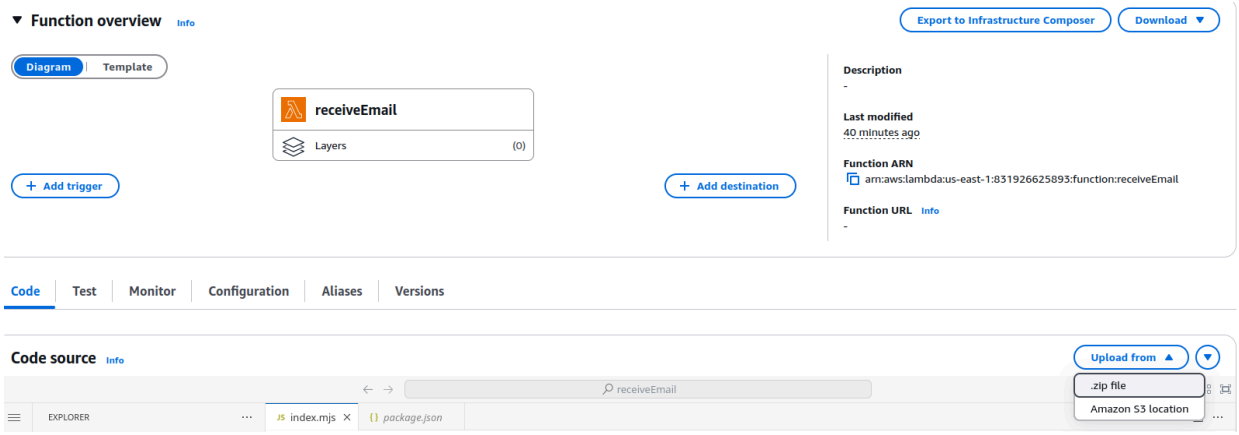
Set up lambda functions

Creating the functions

Create The Receive email function



Now upload the provided zip file to the function



Configuring the receiveEmail function

Add these environment variables

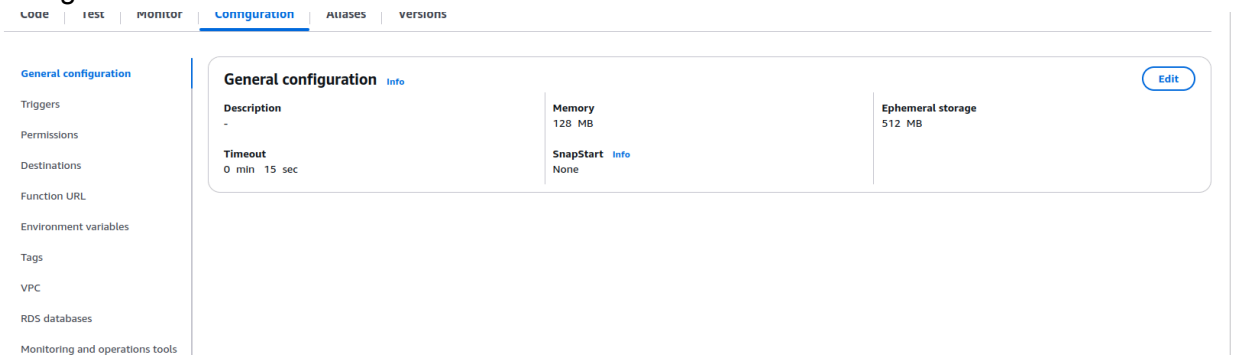
`FORWARDING_EMAIL` = [no-reply@domain.click]

`DOMAIN_NAME` = [domain.click]

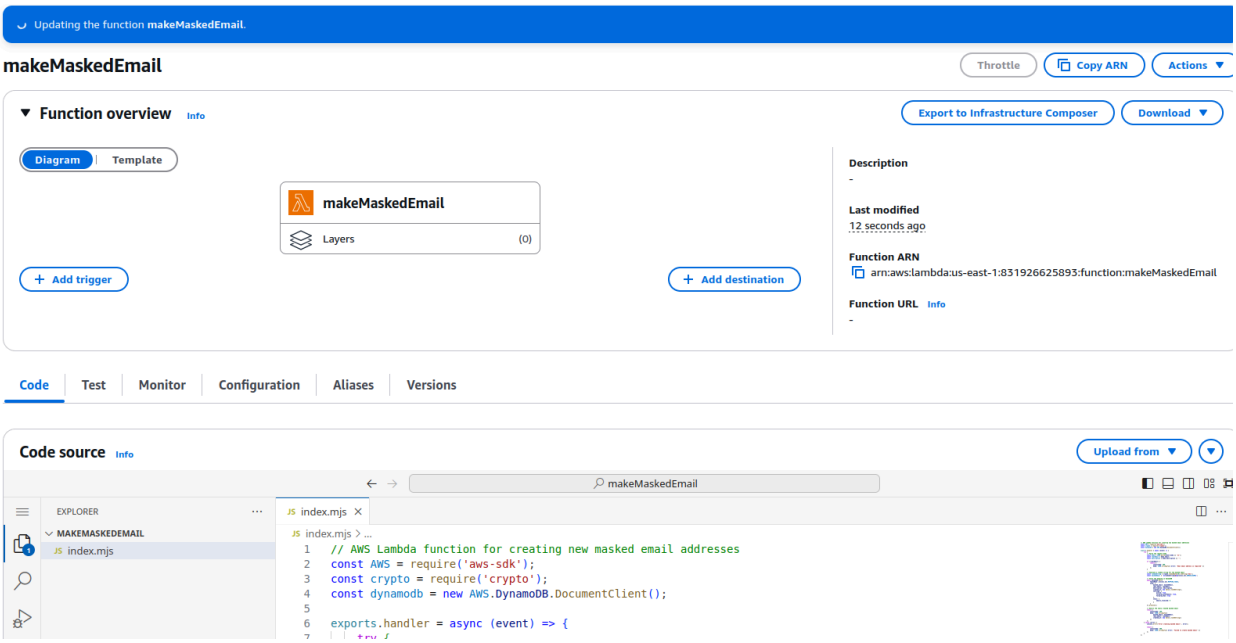
`MAPPING_TABLE` = [DynamoDB Mapping Table that you created]

S3_BUCKET = [Your S3 name bucket you created]

Change timeout to 15s



Do the same for the creating masked email function




```
import { DynamoDB } from '@aws-sdk/client-dynamodb';
import { DynamoDBDocument } from '@aws-sdk/lib-dynamodb';
import crypto from 'crypto';

const dynamoClient = new DynamoDB();
const dynamodb = DynamoDBDocument.from(dynamoClient);

export const handler = async (event) => {
   try {
       // Parse the request body
       const body = event.body || '{}';
       const realEmail = body.email;
       const description = body.description || '';

       console.log("Parsed the body")

       if (!realEmail) {
           return {
               statusCode: 400,
               body: JSON.stringify({ error: 'Real email address is
required' })
           };
       }
```

```
        // Generate a random string for the masked email
        const randomString = crypto.randomBytes(8).toString('hex');
        const maskedEmail = `${randomString}@${process.env.DOMAIN_NAME}`;

        console.log("Generated strings")

        // Store the mapping in DynamoDB
        await dynamodb.put({
            TableName: process.env.MAPPING_TABLE,
            Item: {
                masked_email: maskedEmail,
                real_email: realEmail,
                description: description,
                created_at: new Date().toISOString(),
                settings: {
                    active: true,
                    forward_attachments: true,
                    forward_html: true
                },
                stats: {
                    emails_received: 0
                }
            }
        });

        console.log("DynamoDB item created")

        // Return the newly created masked email
        return {
            statusCode: 200,
            body: JSON.stringify({
                masked_email: maskedEmail,
                real_email: realEmail,
                created_at: new Date().toISOString()
            })
        };
    } catch (error) {
        console.error('Error creating masked email:', error);

        return {
            statusCode: 500,
            body: JSON.stringify({ error: 'Failed to create masked email' })
        };
```

```
    }
};
```

Change the timeout to 15s.

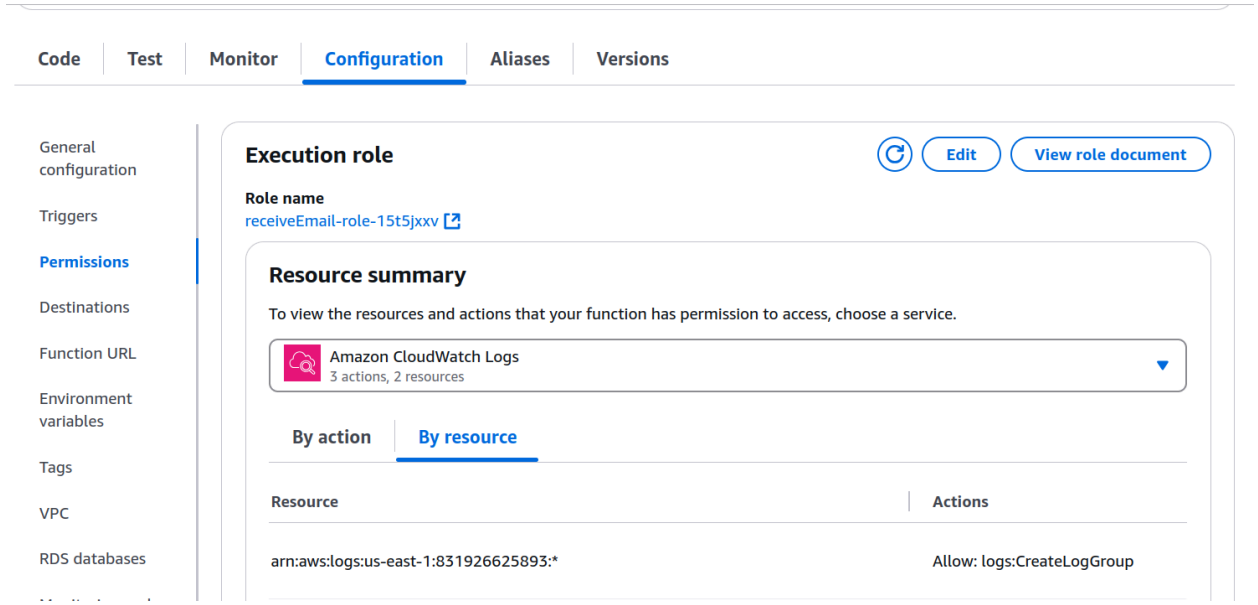
Environment variables to add:

`DOMAIN_NAME`

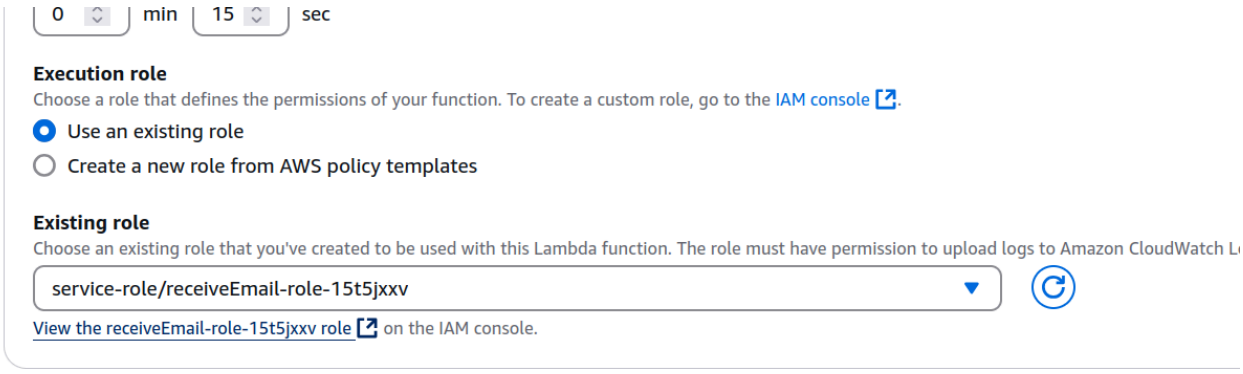
`MAPPING_TABLE`

Set up Lambda IAM Role

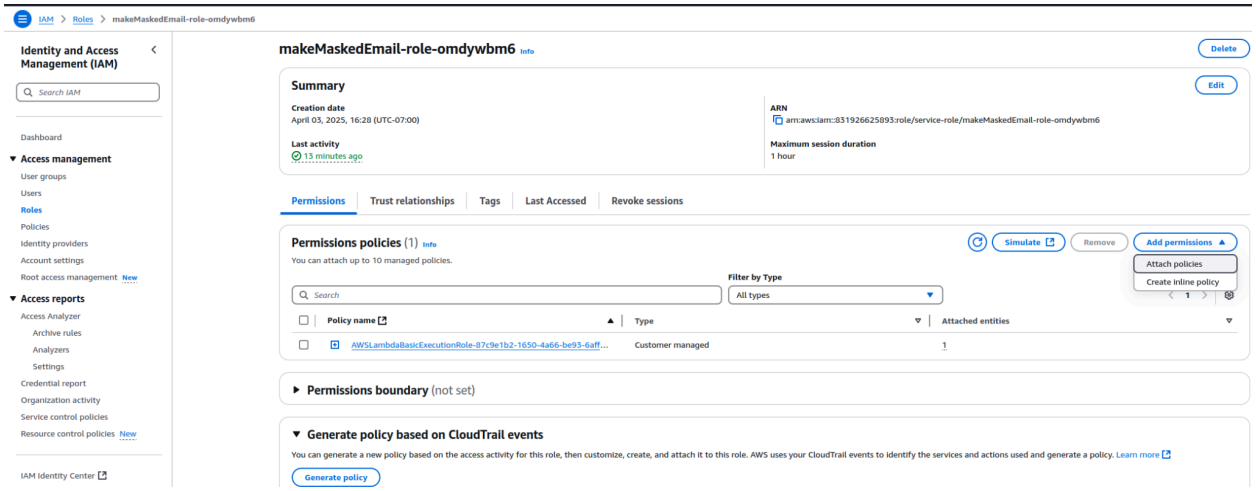
This allows the lambda functions to access the resources needed.
Go to the **makeEmail** lambda function and click on the permissions tab in the configuration menu.



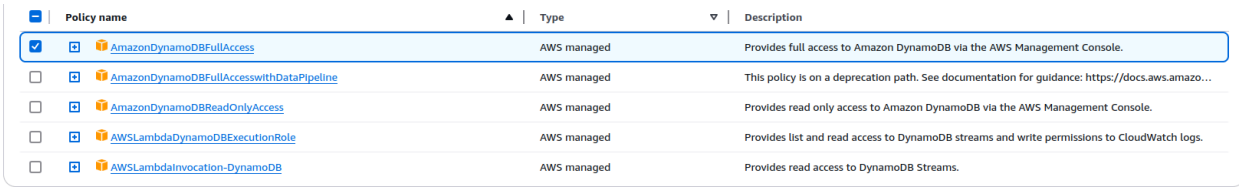
Click on the role name link.



Attach a policy

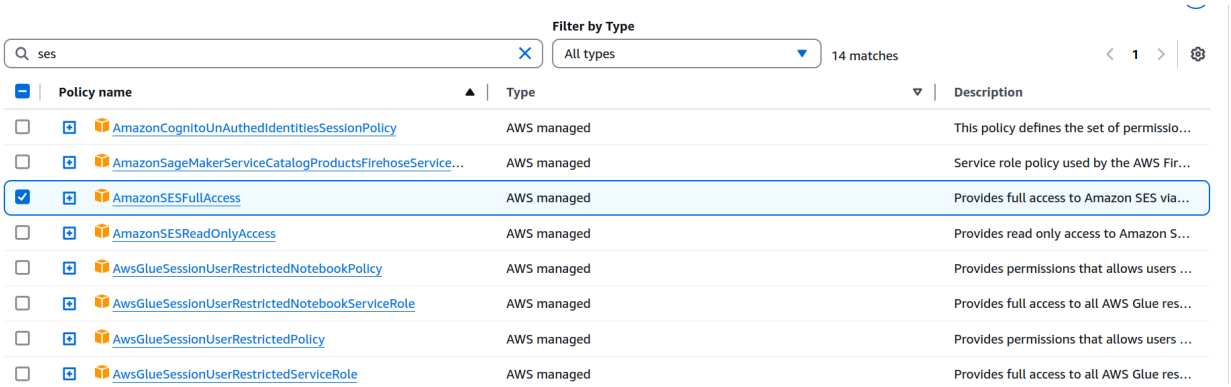


Add the dynamoDBFullAccess



Policy name	Type	Description
AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon DynamoDB via the AWS Management Console.
AmazonDynamoDBFullAccesswithDataPipeline	AWS managed	This policy is on a deprecation path. See documentation for guidance: https://docs.aws.amazo...
AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only access to Amazon DynamoDB via the AWS Management Console.
AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read access to DynamoDB streams and write permissions to CloudWatch logs.
AWSLambdaInvocation-DynamoDB	AWS managed	Provides read access to DynamoDB Streams.

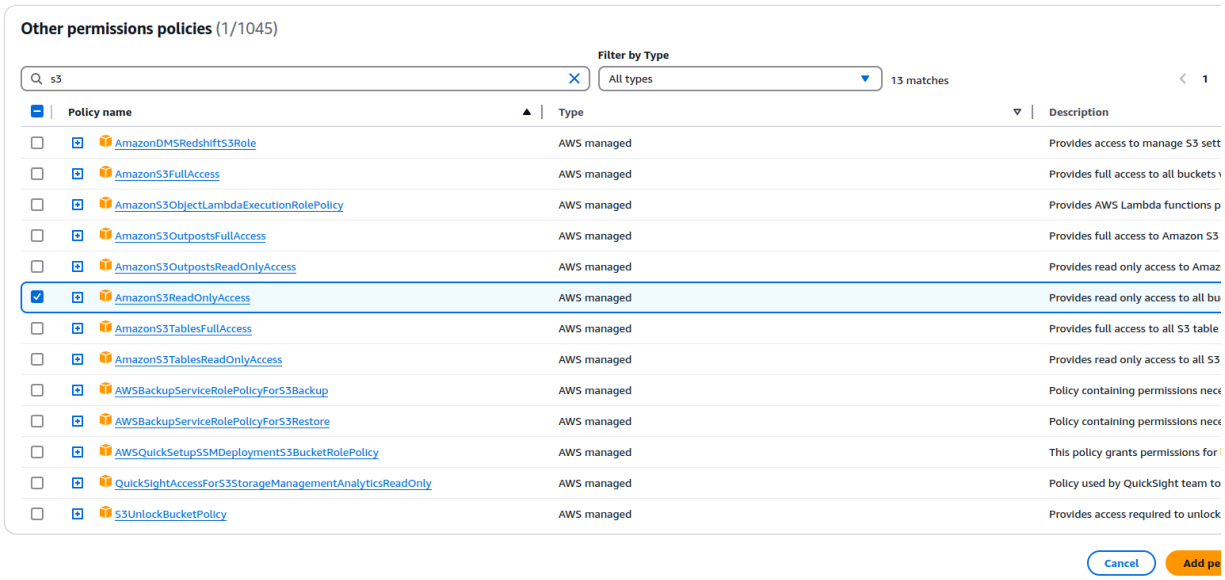
Now do the same for the **receivingEmail** function, but also add the SES full access and S3 Read only access as well.



Filter by Type

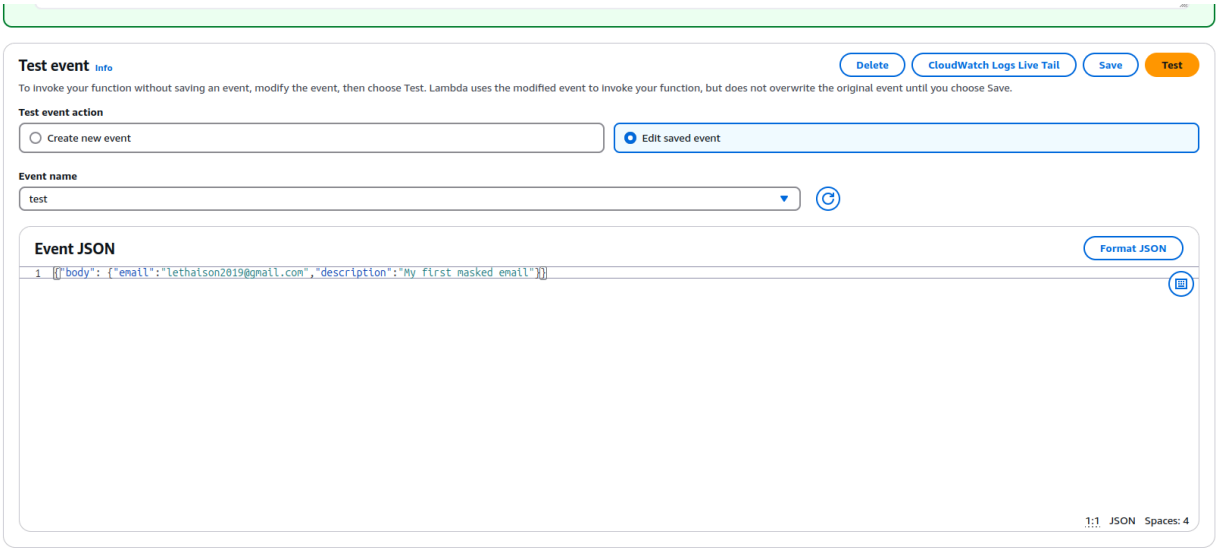
ses | All types | 14 matches | 1

Policy name	Type	Description
AmazonCognitoUnAuthedIdentitiesSessionPolicy	AWS managed	This policy defines the set of permissio...
AmazonSageMakerServiceCatalogProductsFirehoseService...	AWS managed	Service role policy used by the AWS Fir...
AmazonSESFullAccess	AWS managed	Provides full access to Amazon SES via...
AmazonSESReadOnlyAccess	AWS managed	Provides read only access to Amazon S...
AwsGlueSessionUserRestrictedNotebookPolicy	AWS managed	Provides permissions that allows users ...
AwsGlueSessionUserRestrictedNotebookServiceRole	AWS managed	Provides full access to all AWS Glue res...
AwsGlueSessionUserRestrictedPolicy	AWS managed	Provides permissions that allows users ...
AwsGlueSessionUserRestrictedServiceRole	AWS managed	Provides full access to all AWS Glue res...



Testing/Creating a new masked email

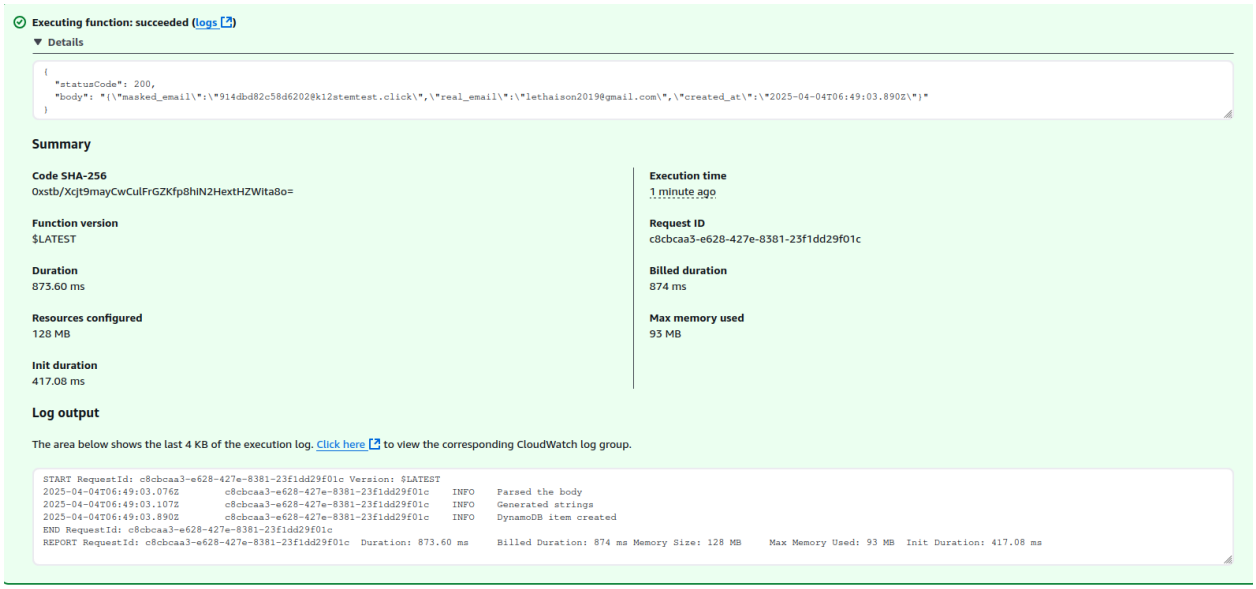
Now run a test of the code with a lambda test. This will be the makeEmail test.



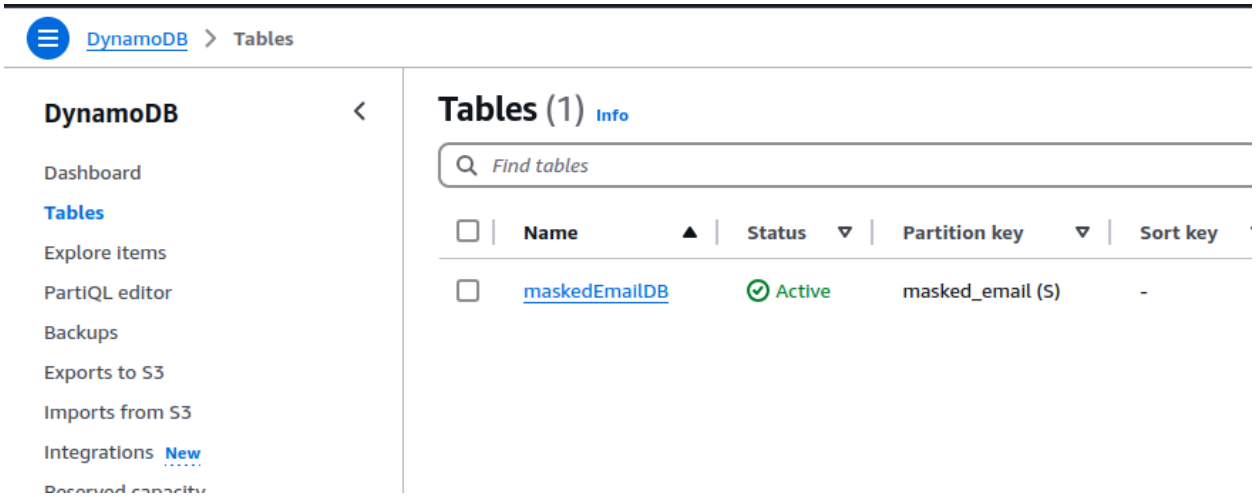
Use the email you would like to mask here
{"body": {"email":"[email]","description":"My first masked email"}}

Hit test.

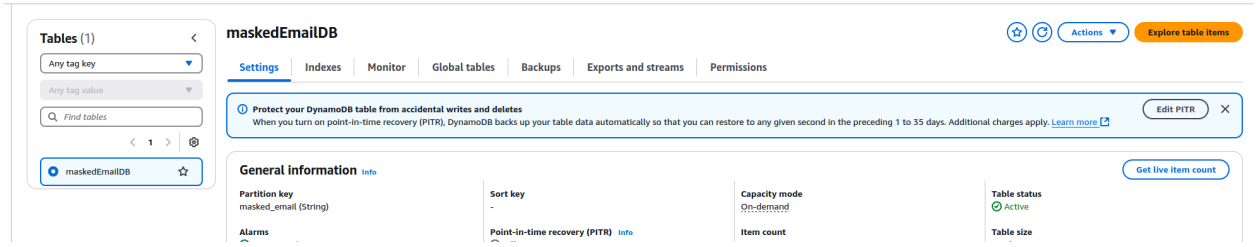
You should get this type of execution



Check the DynamoDB table as well to see your masked email.



Explore table items



It should have something like this.

Items returned (1)

masked_email *(String)*	created_at	description	real_email	settings	stats
914dbd82c58d6202@k12...	2025-04-04...	My first mask...	lethaison20...	{ "active" : {...	{ "emails_received" : { "N" : "1" } }

You can also test the receive email function to see if it runs.
Below is the json AWS SES will send to the lambda. You can create a new test with this example.


```
{
      "Records": [
      {
            "eventSource": "aws:ses",
            "eventVersion": "1.0",
            "ses": {
                  "mail": {
                  "timestamp": "2025-04-04T21:29:50.765Z",
                  "source": "lethaison2019@gmail.com",
                  "messageId": "a0fv0egdj005b2gmcocvn2u9idpdgl9bu0c8ld01",
                  "destination": [
                        "914dbd82c58d6202@k12stemtest.click"
                  ],
                  "headersTruncated": false,
                  "headers": [
                        {
                              "name": "Return-Path",
                              "value": "<lethaison2019@gmail.com>"
                        },
                        {
                              "name": "Received",
                              "value": "from mail-pj1-f49.google.com
(mail-pj1-f49.google.com [209.85.216.49]) by
inbound-smtp.us-east-1.amazonaws.com with SMTP id
a0fv0egdj005b2gmcocvn2u9idpdgl9bu0c8ld01 for
914dbd82c58d6202@k12stemtest.click; Fri, 04 Apr 2025 21:29:50 +0000 (UTC)"
                        },
                        {
                              "name": "X-SES-Spam-Verdict",
                              "value": "PASS"
                        },
```

```
                        {
                              "name": "X-SES-Virus-Verdict",
                              "value": "PASS"
                        },
                        {
                              "name": "Received-SPF",
                              "value": "pass (spfCheck: domain of
_spf.google.com designates 209.85.216.49 as permitted sender)
client-ip=209.85.216.49; envelope-from=lethaison2019@gmail.com;
helo=mail-pj1-f49.google.com;"
                        },
                        {
                              "name": "Authentication-Results",
                              "value": "amazonses.com; spf=pass (spfCheck:
domain of _spf.google.com designates 209.85.216.49 as permitted sender)
client-ip=209.85.216.49; envelope-from=lethaison2019@gmail.com;
helo=mail-pj1-f49.google.com; dkim=pass header.i=@gmail.com; dmarc=pass
header.from=gmail.com;"
                        },
                        {
                              "name": "X-SES-RECEIPT",
                              "value":
"AEFBQUFBQUFBQUFFTVFNVWUwZ0wwTEU4ZEs4d0YxS3I4NnBBSnFndE9mNTk3UE81WWNIVXk5NV
N0SHlldlFHTDA0WHYvQzJ6d245T2ZtVHNWb0lOUlJFRk10YlBva0g5TDdwMWw1UU5WczlnNVFCN
nlndGlXMmZUU1paeVVTN2lxN3dZSFdzNFJDTExHYVViZ3EzTnJSNlZ0ZG1welZFbzM2eXAzR0xK
M1JxdDN0bFl2c0gvYlZyUHpldCtkU25MKzRhbEMzN09seEM1NTBra3U4U0xvRXR5WENHbURmaXF
3eHhicXQ4ajhBTlNyNDNTUlk4UXVLN0VKWU9wY0pNbUdJUVgrRzNBYXA4eFlaSEhYSnJmUVQ2K3
BvRDROrysrR0s3bkFzWjIwMnVMM24yeldoaS81K0E9PQ=="
                        },
                        {
                              "name": "X-SES-DKIM-SIGNATURE",
                              "value": "a=rsa-sha256; q=dns/txt;
b=tg8DXavp8P+rdC/cuIUSHgUBKwOPSTnI4Sjh8Ym1sXqlj/4lvHPU6g0P56cHJFuusYYWCymtb
o3jPfUqIOs0KFhecDxjhWiM4kIENjHich/wbmX8RxWrNswKaDHKc8CIgFPejmANiBOk3gqNzMc6
Tk0jUtrSo7hNx28wu7MXD2w=; c=relaxed/simple;
s=6gbrjpgwjskckoa6a5zn6fwqkn67xbtw; d=amazonses.com; t=1743802191; v=1;
bh=ma4bidNDdb+4SkjU3n/LCli3dRfgCI2cXwmezxt+wVU=;
h=From:To:Cc:Bcc:Subject:Date:Message-ID:MIME-Version:Content-Type:X-SES-RE
CEIPT;"
                        },
                        {
                              "name": "Received",
                              "value": "by mail-pj1-f49.google.com with
```

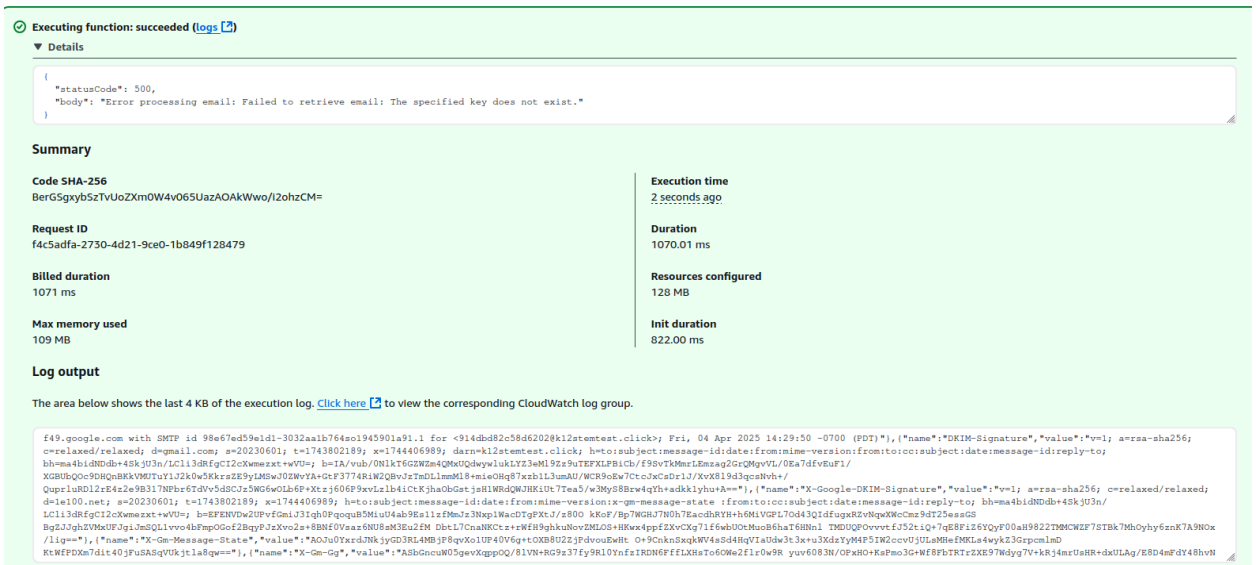
SMTP id 98e67ed59e1d1-3032aa1b764so1945901a91.1 for
<914dbd82c58d6202@k12stemtest.click>; Fri, 04 Apr 2025 14:29:50 -0700
(PDT)"
                        },
                        {
                              "name": "DKIM-Signature",
                              "value": "v=1; a=rsa-sha256;
c=relaxed/relaxed; d=gmail.com; s=20230601; t=1743802189; x=1744406989;
darn=k12stemtest.click;
h=to:subject:message-id:date:from:mime-version:from:to:cc:subject:date:mess
age-id:reply-to; bh=ma4bidNDdb+4SkjU3n/LCli3dRfgCI2cXwmezxt+wVU=;
b=IA/vub/0NlkT6GZWZm4QMxUQdwywlukLYZ3eMl9Zz9uTEFXLPBiCb/f9SvTkMmrLEmzag2GrQ
MgvVL/0Ea7dfvEuF1/XGBUbQOc9DHQnBKkVMUTuY1J2k0w5KkrsZE9yLMSwJ0ZWvYA+GtF3774R
iW2QBvJzTmDLlmmMl8+mieOHq87xzb1L3umAU/WCR9oEw7CtcJxCsDr1J/XvX8l9d3qcsNvh+/Q
uprluRDl2rE4z2e9B317NPbr6TdVv5dSCJz5WG6wOLb6P+Xtzj606P9xvLzlb4iCtKjhaObGstj
sH1WRdQWJHKiUt7Tea5/w3MyS8Brw4qYh+adkk1yhu+A=="
                        },
                        {
                              "name": "X-Google-DKIM-Signature",
                              "value": "v=1; a=rsa-sha256;
c=relaxed/relaxed; d=1e100.net; s=20230601; t=1743802189; x=1744406989;
h=to:subject:message-id:date:from:mime-version:x-gm-message-state
:from:to:cc:subject:date:message-id:reply-to;
bh=ma4bidNDdb+4SkjU3n/LCli3dRfgCI2cXwmezxt+wVU=;
b=EFENVDw2UPvfGmiJ3Iqh0PqoquB5MiuU4ab9Es11zfMmJz3Nxp1WacDTgPXtJ/z80O
kKoF/Bp7WGHJ7N0h7EacdhRYH+h6MiVGPL7Od43QIdfugxRZvNqwXWcCmz9dT25essGS
BgZJJghZVMxUFJgiJmSQL1vvo4bFmpOGof2BqyPJzXvo2s+8BNf0Vsaz6NU8sM3Eu2fM
DbtL7CnaNKCtz+rWfH9ghkuNovZMLOS+HKwx4ppfZXvCXg71f6wbUOtMuoB6haT6HNnl
TMDUQPOvvvtfJ52tiQ+7qE8FiZ6YQyF00aH9822TMMCWZF7STBk7MhOyhy6znK7A9NOx
/lig=="
                        },
                        {
                              "name": "X-Gm-Message-State",
                              "value":
"AOJu0YxrdJNkjyGD3RL4MBjP8qvXo1UP40V6g+tOXB8U2ZjPdvouEwHt
O+9CnknSxqkWV4sSd4HqVIaUdw3t3x+u3XdzYyM4P5IW2ccvUjULsMHefMKLs4wykZ3GrpcmlmD
KtWfPDXm7dit40jFuSASqVUkjtla8qw=="
                        },
                        {
                              "name": "X-Gm-Gg",
                              "value":
"ASbGncuW05gevXqppOQ/8lVN+RG9z37fy9Rl0YnfzIRDN6FffLXHsTo6OWe2flr0w9R
yuv6083N/OPxHO+KsPmo3G+Wf8FbTRTrZXE97Wdyg7V+kRj4mrUsHR+dxULAg/E8D4mFdY48hvN

```
nbckqWvGKBhWsFiZGjoO3oCFKB8vw8Bd0AvlwN"
                        },
                        {
                              "name": "X-Google-Smtp-Source",
                              "value":
"AGHT+IGvc3Ai5NahZzaWgVd7QHERCpM1b6OgC6B0opsgAo4hQdf45+z0HFtyVTVBLB3UQCEDXV
cKQnlMUev3A2FSXgs="
                        },
                        {
                              "name": "X-Received",
                              "value": "by
2002:a17:90a:d64c:b0:305:5f28:2d5c with SMTP id
98e67ed59e1d1-306a6173b2fmr5759506a91.15.1743802189468; Fri, 04 Apr 2025
14:29:49 -0700 (PDT)"
                        },
                        {
                              "name": "MIME-Version",
                              "value": "1.0"
                        },
                        {
                              "name": "From",
                              "value": "Thaison Le
<lethaison2019@gmail.com>"
                        },
                        {
                              "name": "Date",
                              "value": "Fri, 4 Apr 2025 14:29:38 -0700"
                        },
                        {
                              "name": "X-Gm-Features",
                              "value":
"ATxdqUHRdsBwRbKGF8FkPRo2d_ULPz5Okgmimkjf6E2MzY9n4gonMsdIrppAVd4"
                        },
                        {
                              "name": "Message-ID",
                              "value":
"<CAHmH_jdV1thzR65oYP6=1_Q4jDtVA0JGCwXo3xd8zSH8hue7kQ@mail.gmail.com>"
                        },
                        {
                              "name": "Subject",
                              "value": "my love"
                        },
                        {
```

```
                            "name": "To",
                            "value": "914dbd82c58d6202@k12stemtest.click"
                      },
                      {
                            "name": "Content-Type",
                            "value": "multipart/alternative;
boundary=\"0000000000000fdd180631fa96ba\""
                      }
                ],
                "commonHeaders": {
                      "returnPath": "lethaison2019@gmail.com",
                      "from": [
                            "Thaison Le <lethaison2019@gmail.com>"
                      ],
                      "date": "Fri, 4 Apr 2025 14:29:38 -0700",
                      "to": [
                            "914dbd82c58d6202@k12stemtest.click"
                      ],
                      "messageId":
"<CAHmH_jdV1thzR65oYP6=1_Q4jDtVA0JGCwXo3xd8zSH8hue7kQ@mail.gmail.com>",
                      "subject": "my love"
                }
                },
                "receipt": {
                "timestamp": "2025-04-04T21:29:50.765Z",
                "processingTimeMillis": 804,
                "recipients": [
                      "914dbd82c58d6202@k12stemtest.click"
                ],
                "spamVerdict": {
                      "status": "PASS"
                },
                "virusVerdict": {
                      "status": "PASS"
                },
                "spfVerdict": {
                      "status": "PASS"
                },
                "dkimVerdict": {
                      "status": "PASS"
                },
                "dmarcVerdict": {
                      "status": "PASS"
```

```
                    },
                    "action": {
                          "type": "Lambda",
                          "functionArn":
"arn:aws:lambda:us-east-1:831926625893:function:receiveEmail",
                          "invocationType": "Event"
                    }
                    }
              }
        }
        ]
}
```

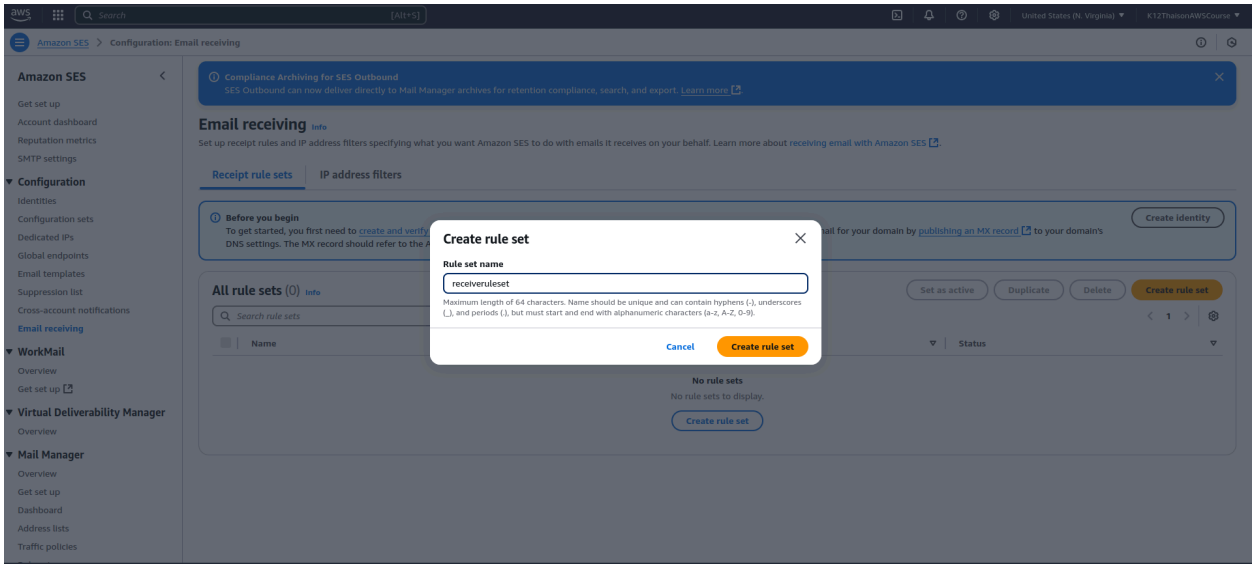
You should get a result that looks like this



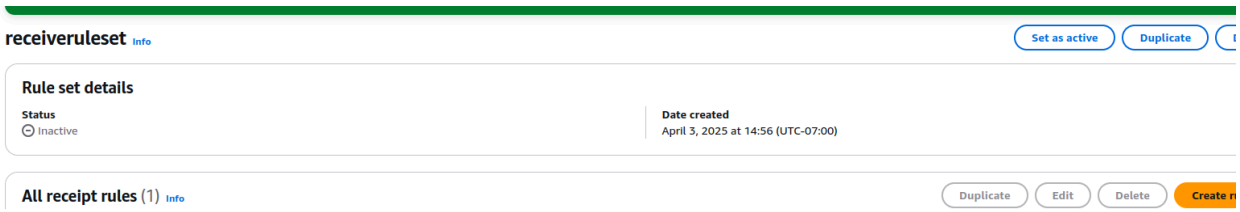
AWS SES Rule set

We need a ruleset to make sure the emails trigger the lambda function.

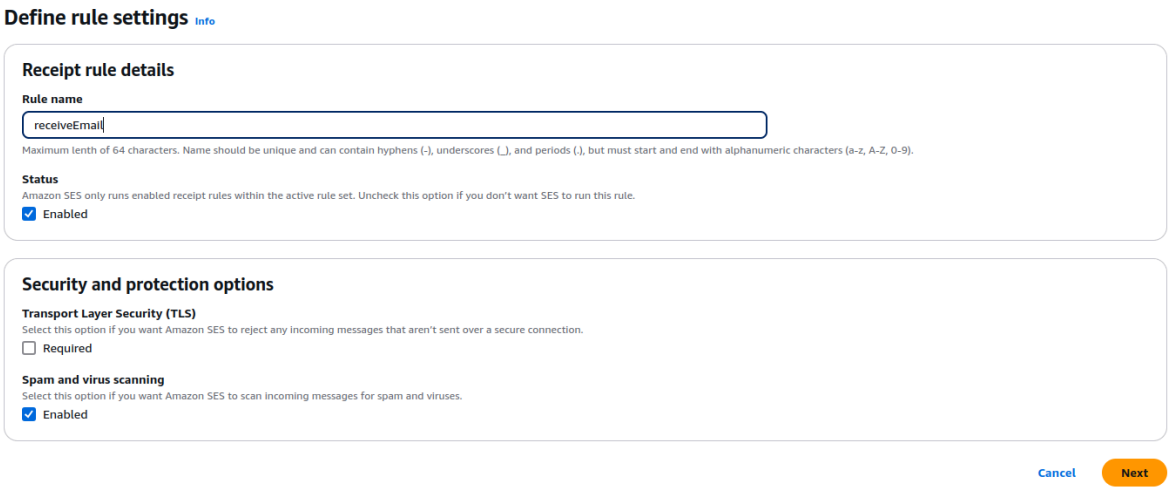
Create a new ruleset. Name it something that makes sense.



Make sure you activate the rule set by clicking set as active.



Create a new rule. Also name it something that makes sense.



Add a recipient condition. This is basically how to match the email. Put your domain there to catch all emails sent to AWS SES.

When the recipient of an incoming message matches the recipient conditions of a receipt rule, Amazon SES performs an ordered list of actions associated with that rule.

▼ **Guidelines**

If you want to...	Then specify the following recipient condition...
Match a specific email address.	*user@example.com*
Match all addresses within a domain, but not those within its subdomains.	*example.com*
Match all addresses within a specific subdomain, but not those within the parent domain.	*subdomain.example.com*
Match all addresses within all subdomains, but not those within the parent domain.	*.example.com*
Match all addresses within a domain, and all addresses within all of its subdomains.	*example.com and .example.com*
Match all recipients in all verified domains.	None

Recipient conditions Info

Amazon SES can only receive mail on your behalf for domains that you own. Any email address that you specify as a recipient condition must belong to a verified domain identity.

Recipient condition

k12stemtest.click

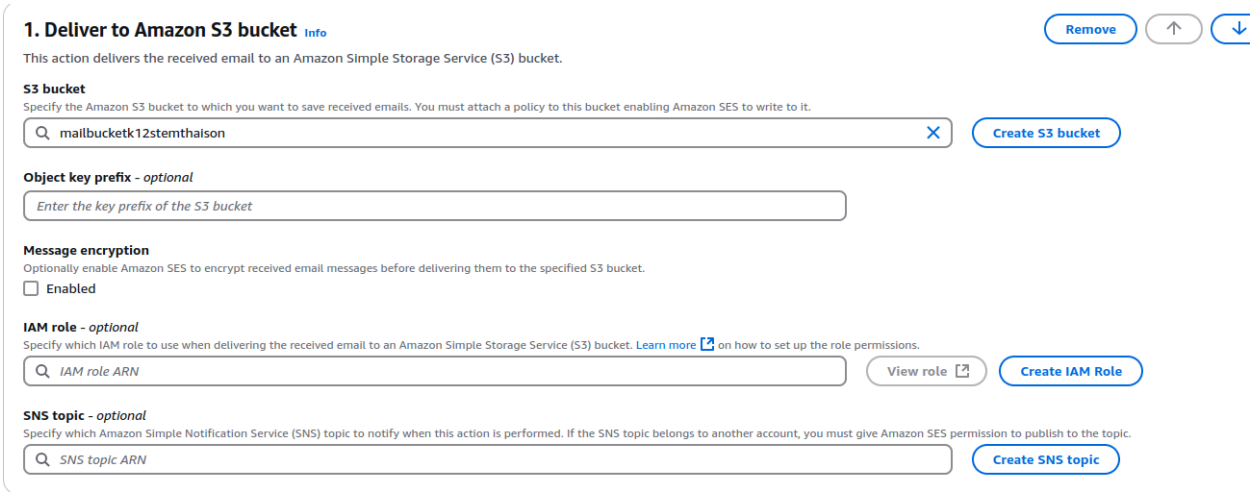
Remove

Add new recipient condition

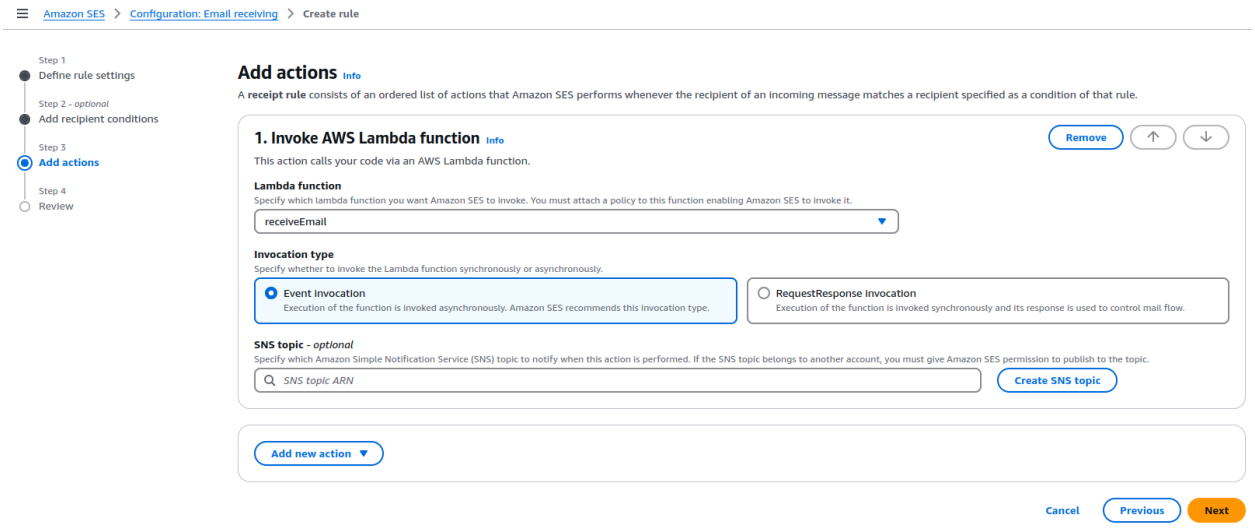
You can add 499 more recipient conditions.

Cancel Previous Next

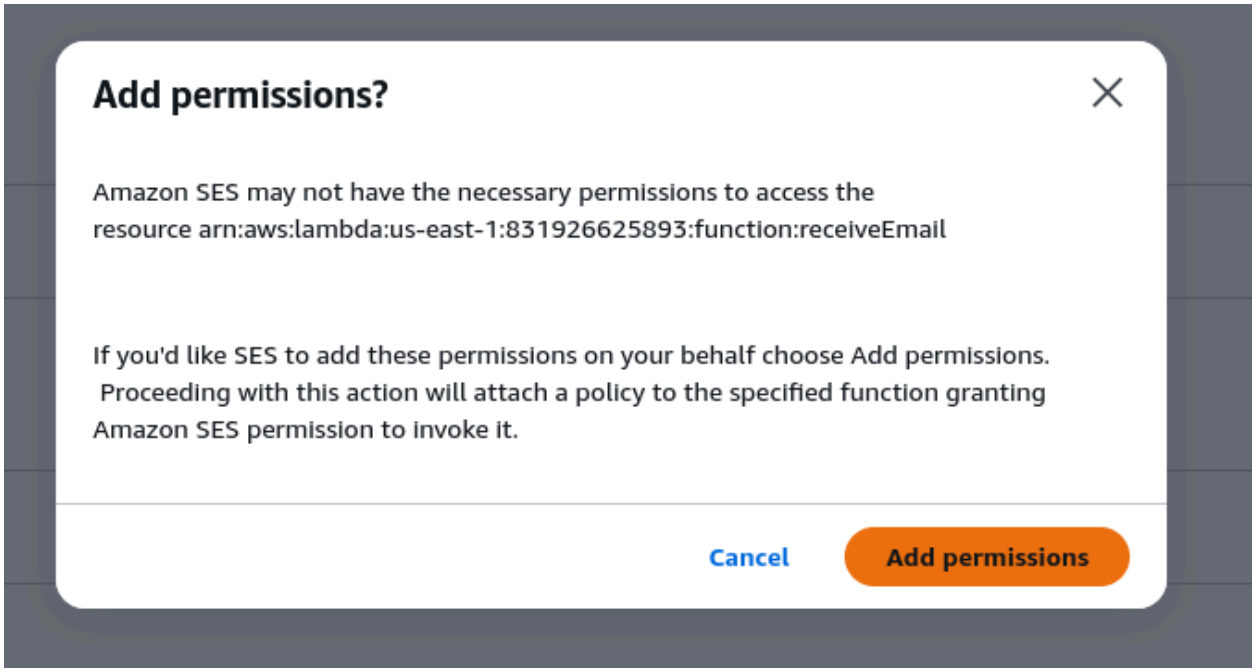
Now we create actions to perform when it catches an email.
For the first action, have the email delivered to the s3 bucket.



Then have it invoke the receive email lambda function. Make sure the s3 bucket action is first though. Use the arrows to move the s3 bucket to the top if it isn't.

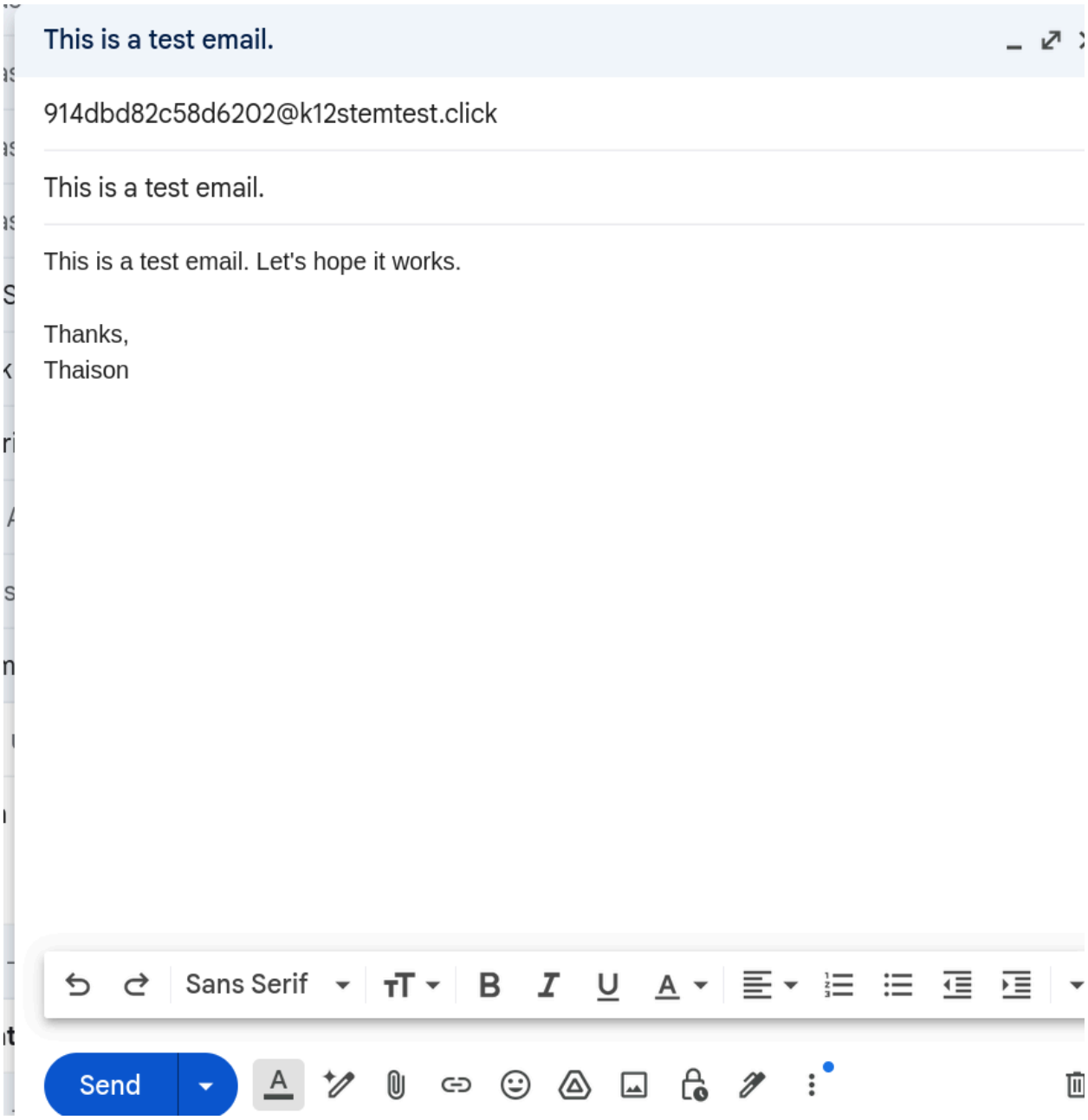


SES will ask for permissions to call the bucket. Give it the permissions.



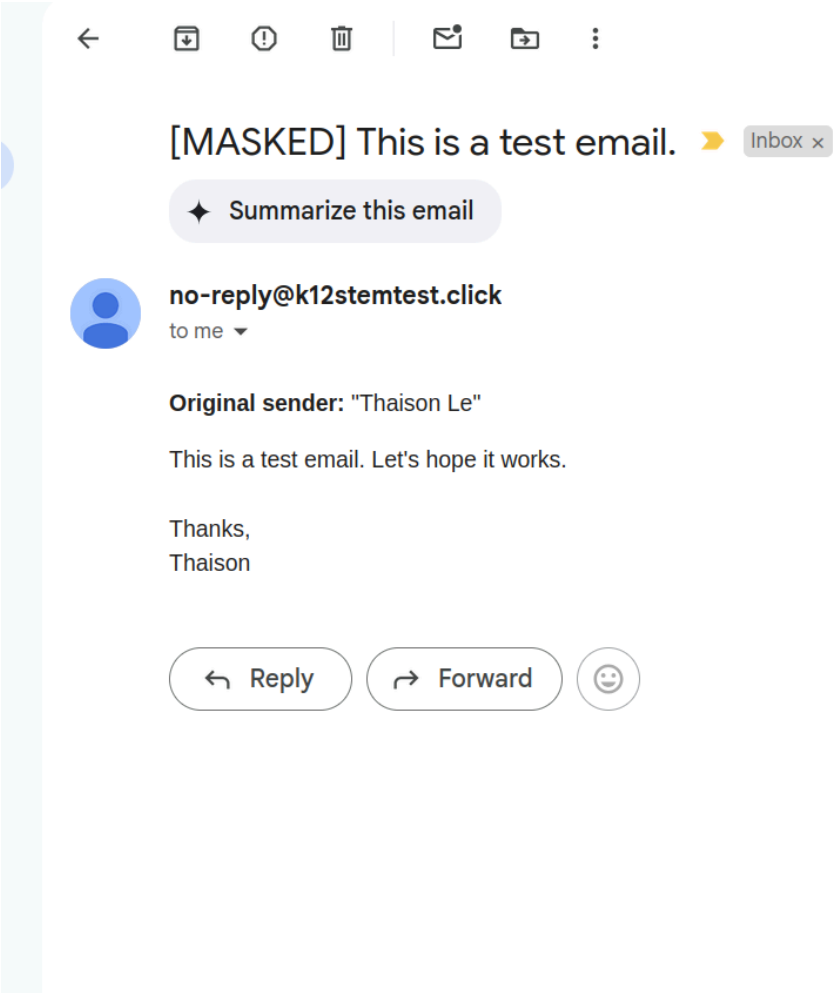
Test the final product

Now that everything is set up, try sending an email to your masked email. If you don't remember it, you can find it in the dynamoDB table.



Wait 10-20 seconds. Now check your inbox.

You should receive an email that looks like this. Congratulations!



Potential Future improvements for you:

Two way communication? Replying to the masked email, similar to something like craigslist.

Web ui to create masked emails easily? You would also need to have the ability to authenticate (AWS Cognito)