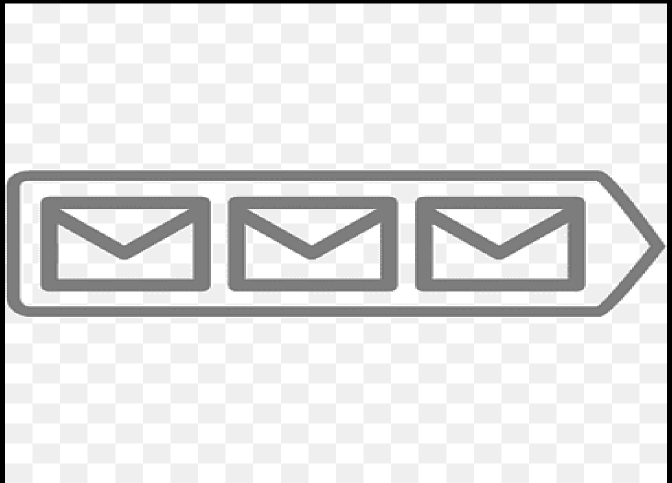
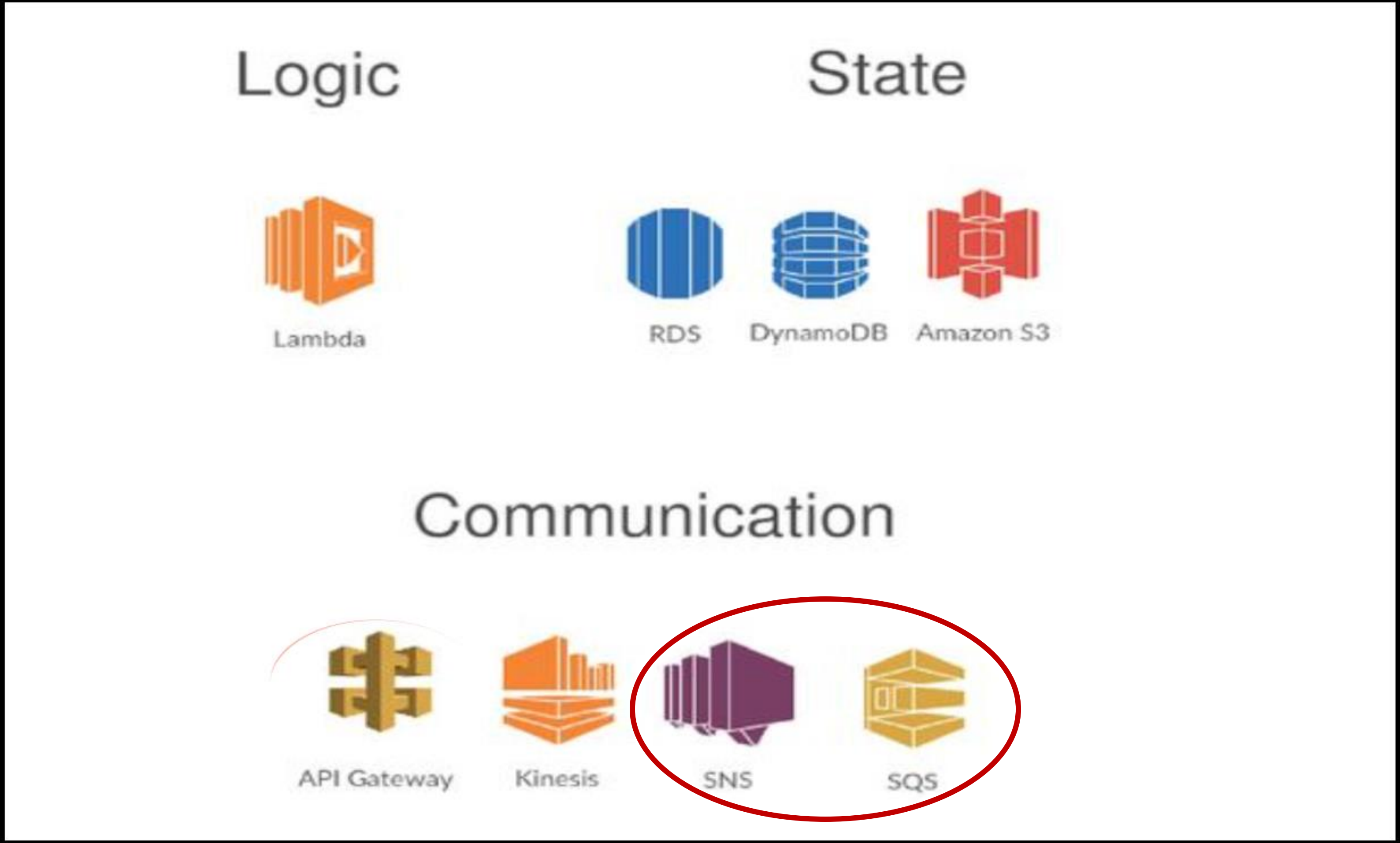


AWS Integration and Messaging Services (Contd).

Components of a Serverless, Message-Driven application

(aka Event Driven Architecture - EDA)



Simple Notification Service (SNS)

Amazon SNS

- Released in 2010.
- A serverless publish subscribe (pub/sub) messaging service.
- When you want to send a message to many receivers.
  - SQS is point-to-point, but SNS is pub/sub.
- The publisher (event producer) sends a message to an SNS topic.
- Many subscribers can listen to the topic.
- Each topic subscriber gets all the messages.
- Subscribers can be:
  - SQS, HTTP / HTTPS, Lambda function
  - Emails (SES)
  - SMS messages, Mobile Notifications

SNS - Features

- Integrates with lots of services (Publishers):
    - Lambda.
    - S3 (Bucket change notifications).
    - Cloudwatch (Alarm notification).
    - etc
- Encryption:
    - In-flight encryption using HTTPS API.
    - At-rest encryption using KMS keys.
- Message Filtering:
    - Subscriber can declare a filtering policy to limit the messages it receives to those of interest.

SNS - Features

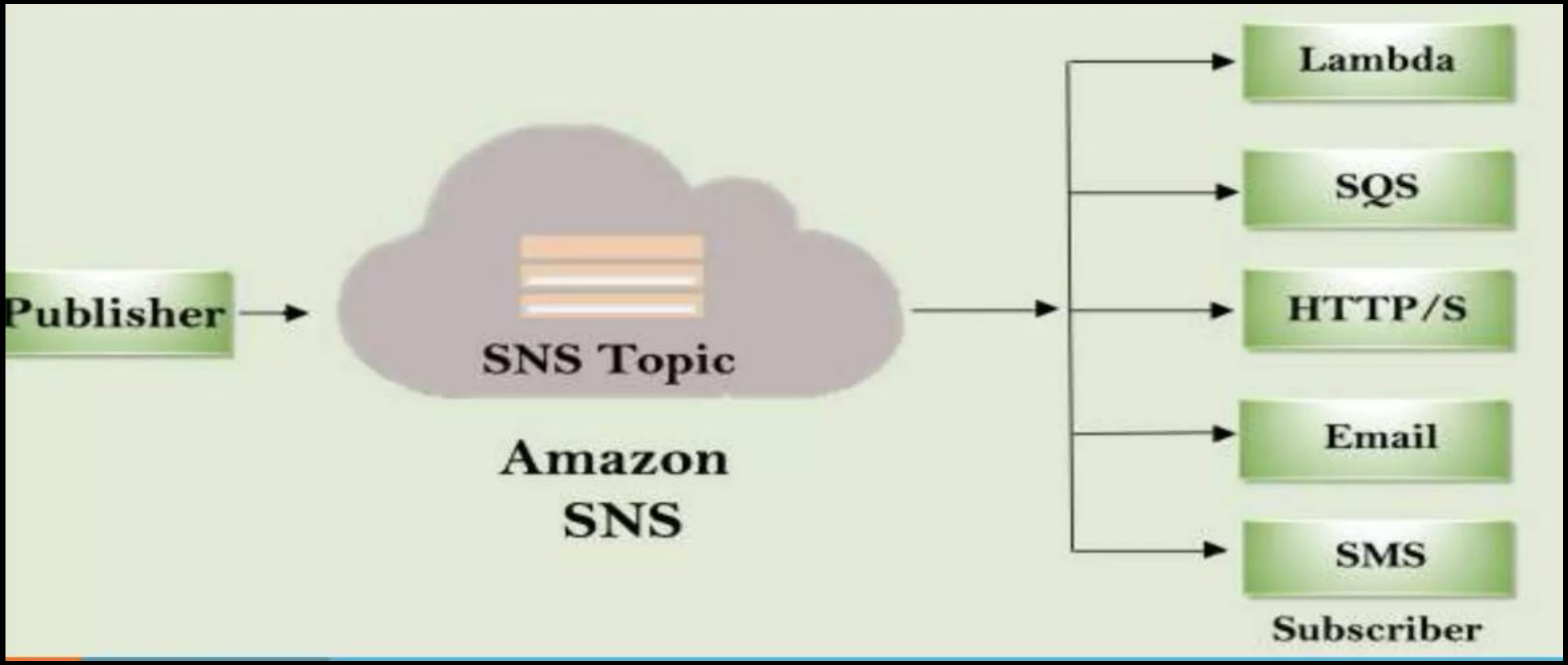
- Security:
  - Access Controls: IAM policies to regulate access to the SNS API.
  - SNS Access Policies (similar to S3 bucket policies):
    - Cross-account access to SNS topics.
    - Allowing other services (e.g. S3) to write to an SNS topic.

- Auto-scaling.

- DLQ – an SQS queue for messages that can't be delivered toa subscriber due to client errors or server errors.

Topics

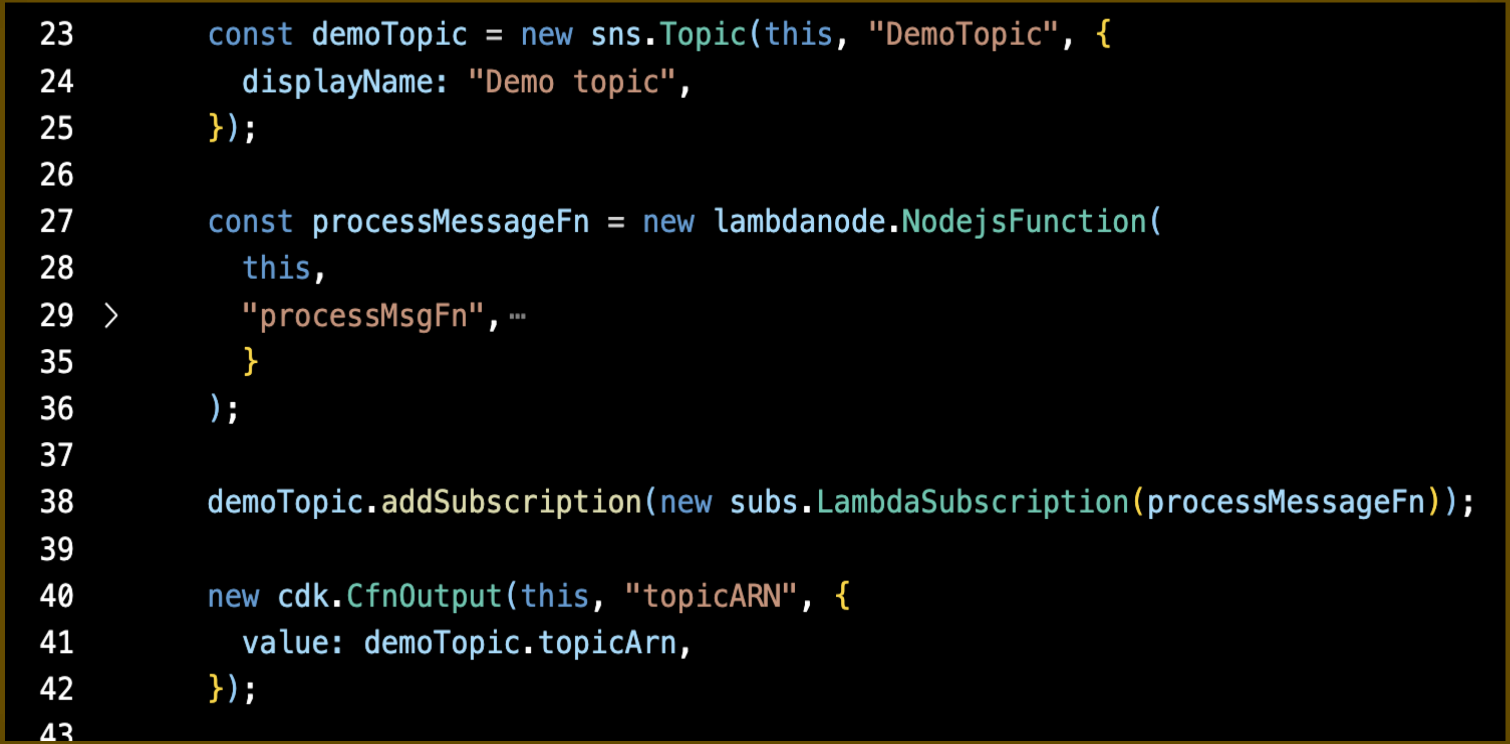
- An SNS topic is a logical access point that acts as a communication channel.
- A topic lets you group multiple endpoints, e.g. SQS, Lambda,



Demo – CDK provisioning code

- Architecture:

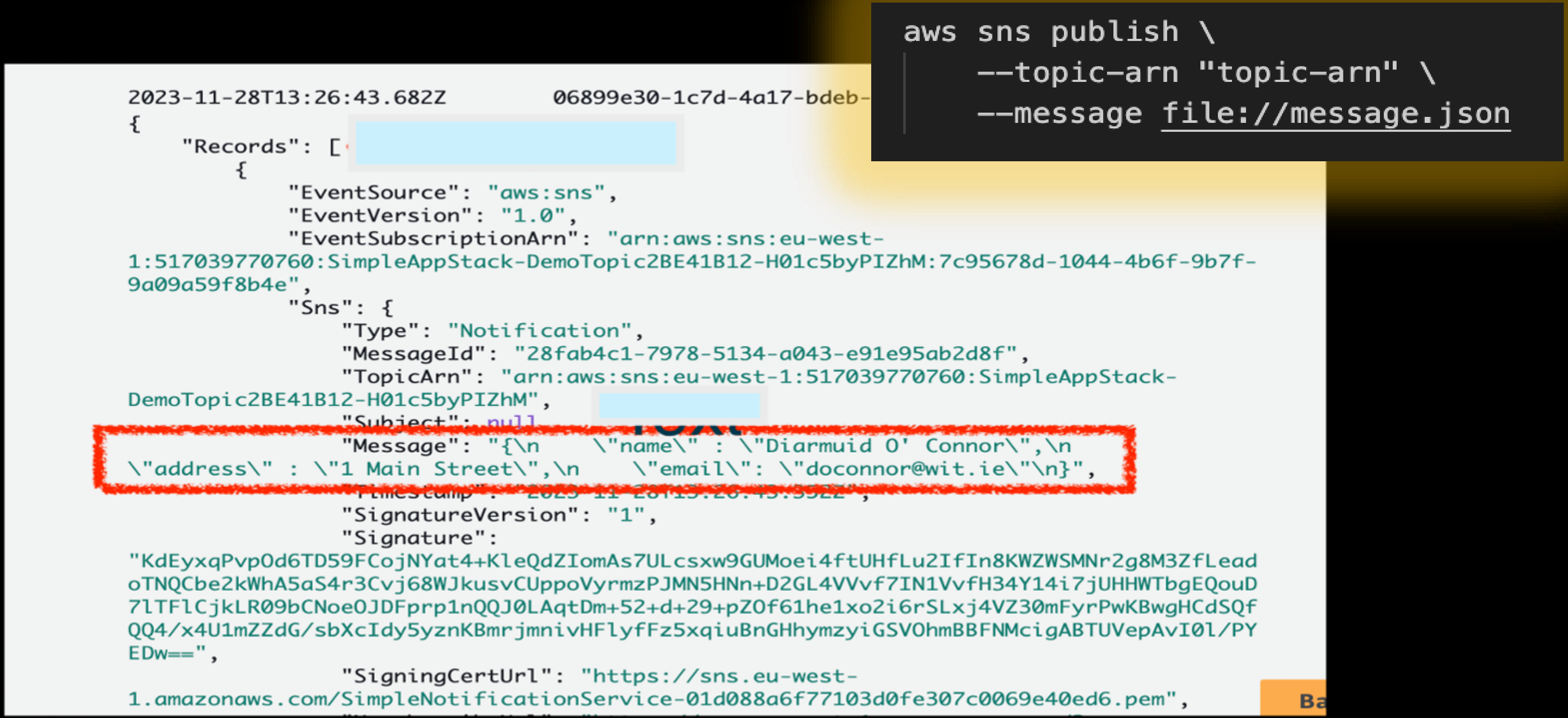
  AWS CLI (Publisher) -→ SNS Topic -→ Lambda (Subscriber)



```
const demoTopic = new sns.Topic(this, "DemoTopic", {
  displayName: "Demo topic",
});

const processMessageFn = new lambdanode.NodejsFunction(
  this,
  "processMsgFn",⋯
  }
);

demoTopic.addSubscription(new subs.LambdaSubscription(processMessageFn));

new cdk.CfnOutput(this, "topicARN", {
  value: demoTopic.topicArn,
});
```

Demo – Lambda subscriber

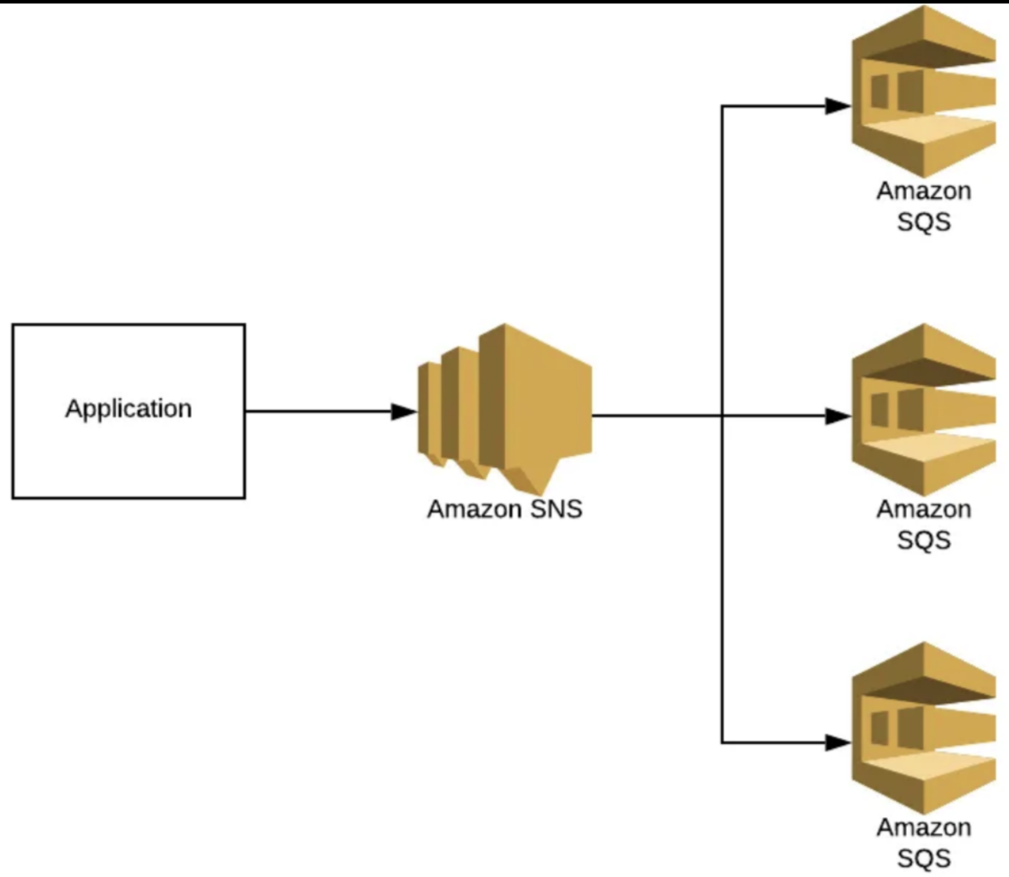
- Lambda subscriber receives a batch (?) of messages in its event parameter.



```
aws sns publish \
    --topic-arn "topic-arn" \
    --message file://message.json
```

```
2023-11-28T13:26:43.682Z        06899e30-1c7d-4a17-bdeb-
{
    "Records": [
        {
            "EventSource": "aws:sns",
            "EventVersion": "1.0",
            "EventSubscriptionArn": "arn:aws:sns:eu-west-
1:517039770760:SimpleAppStack-DemoTopic2BE41B12-H01c5byPIZhM:7c95678d-1044-4b6f-9b7f-
9a09a59f8b4e",
            "Sns": {
                "Type": "Notification",
                "MessageId": "28fab4c1-7978-5134-a043-e91e95ab2d8f",
                "TopicArn": "arn:aws:sns:eu-west-1:517039770760:SimpleAppStack-
DemoTopic2BE41B12-H01c5byPIZhM",
                "Subject": null,
                "Message": "{\n    \"name\" : \"Diarmuid O' Connor\",\n
\"address\" : \"1 Main Street\",\n    \"email\": \"doconnor@wit.ie\"\n}",
                "Timestamp": "2023-11-28T13:26:43.552Z",
                "SignatureVersion": "1",
                "Signature":
"KdEyxqPvpOd6TD59FCojNYat4+KleQdZIomAs7ULcsxw9GUMoei4ftUHfLu2IfIn8KWZWSMNr2g8M3ZfLead
oTNQCbe2kWhA5aS4r3Cvj68WJkusvCUppoVyrmzPJMN5HNn+D2GL4VVvf7IN1VvfH34Y14i7jUHHWTbgEQouD
7lTFlCjkLR09bCNoeOJDFprp1nQQJ0LAqtDm+52+d+29+pZOf61he1xo2i6rSLxj4VZ30mFyrPwKBwgHCdSQf
QQ4/x4U1mZZdG/sbXcIdy5yznKBmrjmnivHFlyfFz5xqiuBnGHhymzyiGSVOhmBBFNMcigABTUVepAvI0l/PY
EDw==",
                "SigningCertUrl": "https://sns.eu-west-
1.amazonaws.com/SimpleNotificationService-01d088a6f77103d0fe307c0069e40ed6.pem",
```

The Fan-out pattern



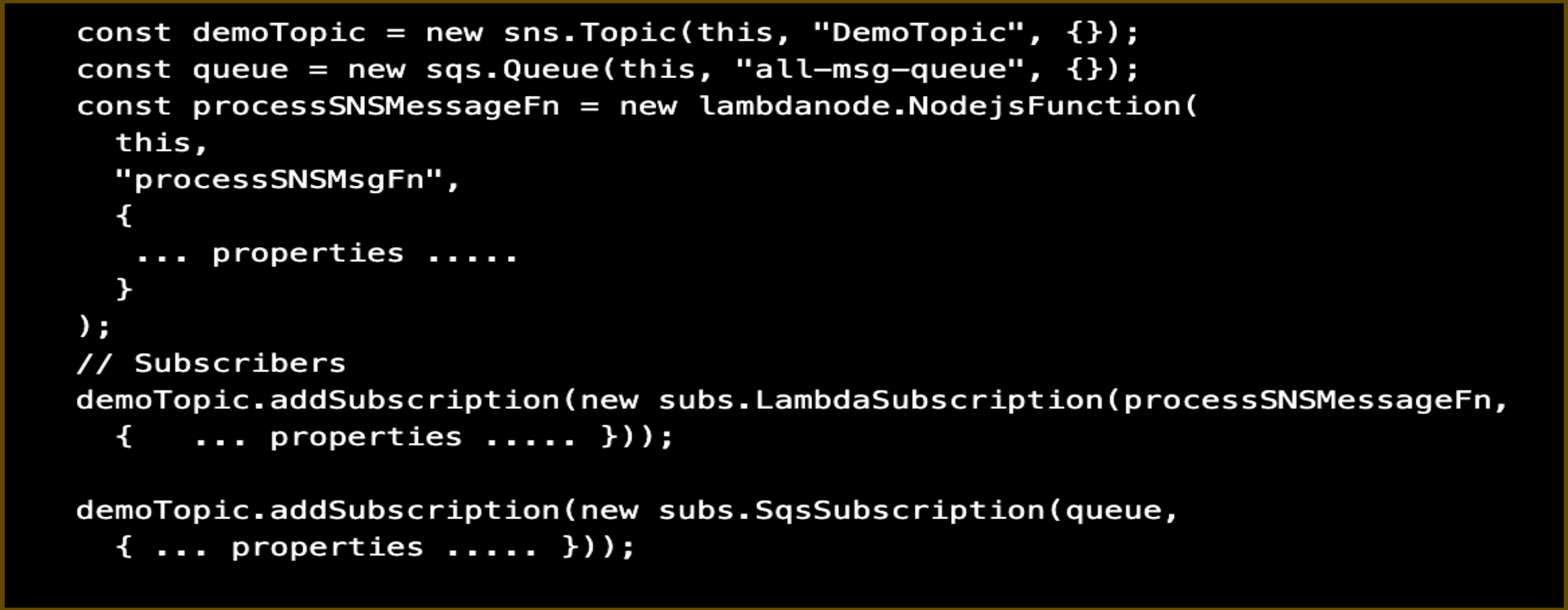
Each Queue has a different Lambda function consumer

Demo – Fan Out.

- The Fan Out subscribers can be a mixture of types.
- Architecture:

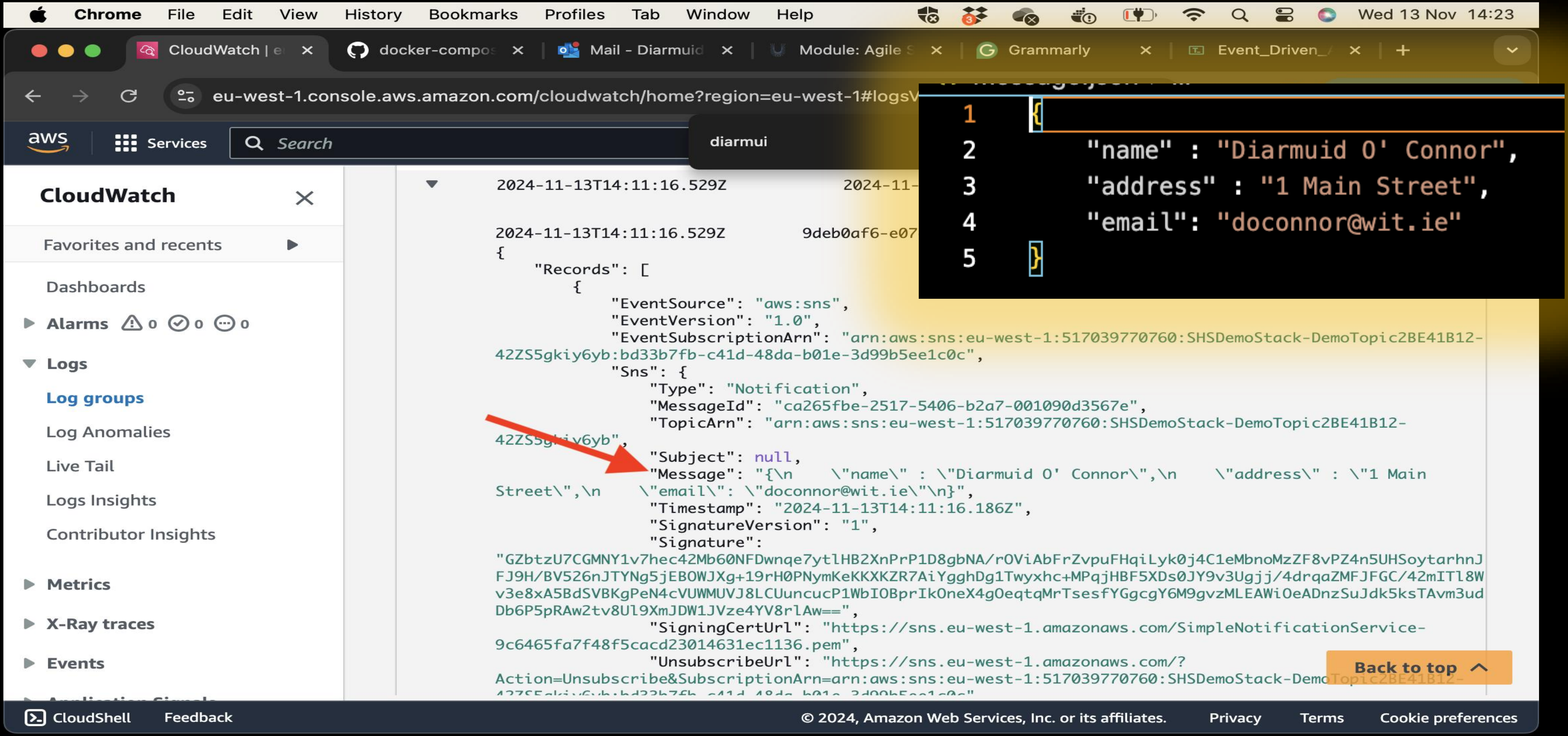
  AWS CLI (Pub) -→ SNS Topic -→ Lambda (Sub)

  -→ SQS (Sub). → Lambda (Consumer)



```
const demoTopic = new sns.Topic(this, "DemoTopic", {});
const queue = new sqs.Queue(this, "all-msg-queue", {});
const processSNSMessageFn = new lambdanode.NodejsFunction(
  this,
  "processSNSMsgFn",
  {
   ... properties .....
  }
);
// Subscribers
demoTopic.addSubscription(new subs.LambdaSubscription(processSNSMessageFn,
  {   ... properties ..... }));

demoTopic.addSubscription(new subs.SqsSubscription(queue,
  { ... properties ..... }));
```

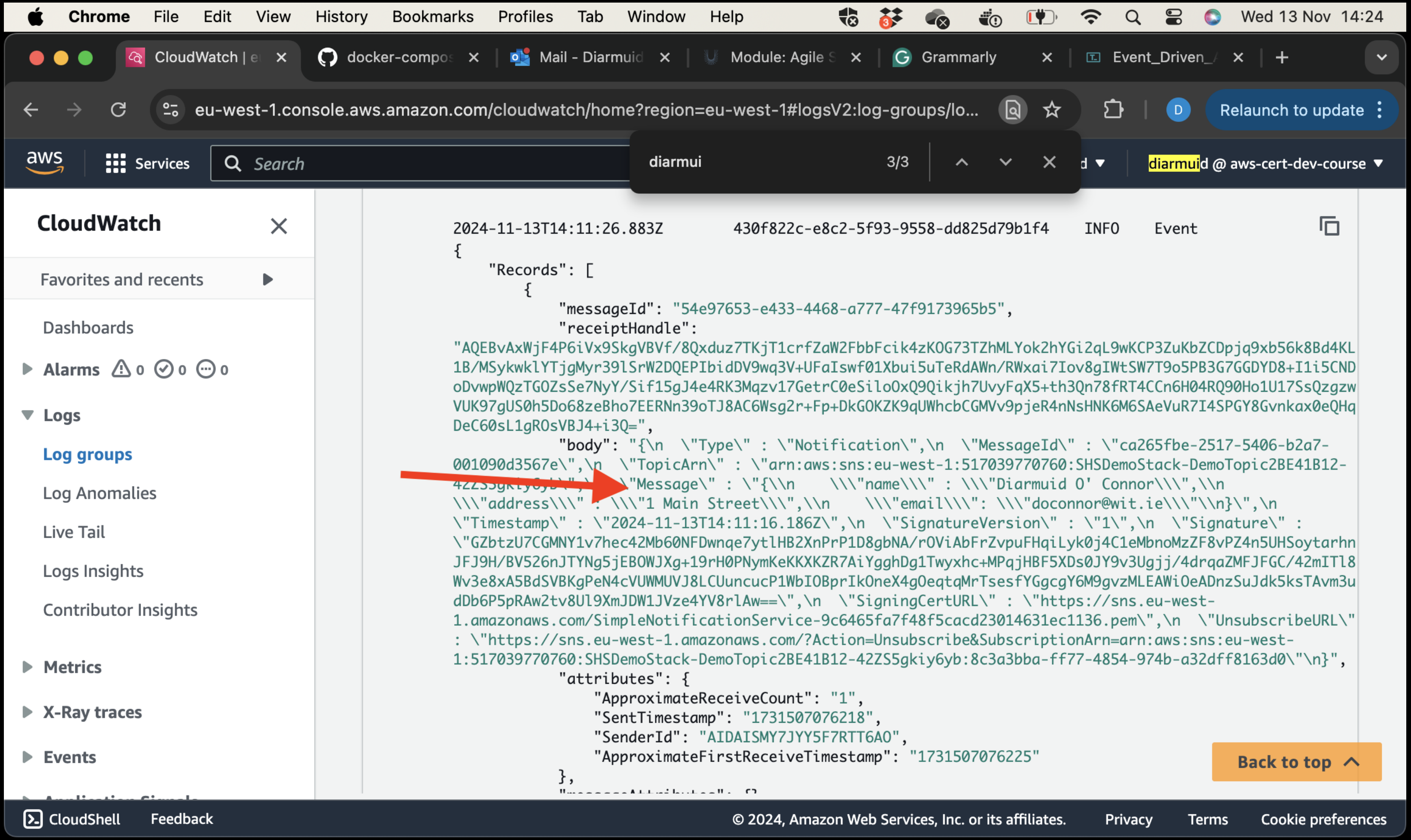
Demo – The Lambda subscriber event parameter



```
{
    "name" : "Diarmuid O' Connor",
    "address" : "1 Main Street",
    "email": "doconnor@wit.ie"
}
```

```
2024-11-13T14:11:16.529Z        2024-11-

2024-11-13T14:11:16.529Z        9deb0af6-e07
{
    "Records": [
        {
            "EventSource": "aws:sns",
            "EventVersion": "1.0",
            "EventSubscriptionArn": "arn:aws:sns:eu-west-1:517039770760:SHSDemoStack-DemoTopic2BE41B12-
42ZS5gkiy6yb:bd33b7fb-c41d-48da-b01e-3d99b5ee1c0c",
            "Sns": {
                "Type": "Notification",
                "MessageId": "ca265fbe-2517-5406-b2a7-001090d3567e",
                "TopicArn": "arn:aws:sns:eu-west-1:517039770760:SHSDemoStack-DemoTopic2BE41B12-
42ZS5gkiy6yb",
                "Subject": null,
                "Message": "{\n    \"name\" : \"Diarmuid O' Connor\",\n    \"address\" : \"1 Main
Street\",\n    \"email\": \"doconnor@wit.ie\"\n}",
                "Timestamp": "2024-11-13T14:11:16.186Z",
                "SignatureVersion": "1",
                "Signature":
"GZbtzU7CGMNY1v7hec42Mb60NFDwnqe7ytlHB2XnPrP1D8gbNA/rOViAbFrZvpuFHqiLyk0j4C1eMbnoMzZF8vPZ4n5UHSoytarhnJ
FJ9H/BV526nJTYNg5jEBOWJXg+19rH0PNymKeKKXKZR7AiYgghDg1Twyxhc+MPqjHBF5XDs0JY9v3Ugjj/4drqaZMFJFGC/42mITl8W
v3e8xA5BdSVBKgPeN4cVUWMUVJ8LCUuncucP1WbIOBprIkOneX4gOeqtqMrTsesfYGgcgY6M9gvzMLEAWiOeADnzSuJdk5ksTAvm3ud
Db6P5pRAw2tv8Ul9XmJDW1JVze4YV8rlAw==",
                "SigningCertUrl": "https://sns.eu-west-1.amazonaws.com/SimpleNotificationService-
9c6465fa7f48f5cacd23014631ec1136.pem",
                "UnsubscribeUrl": "https://sns.eu-west-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:eu-west-1:517039770760:SHSDemoStack-DemoTopic2BE41B12-
42ZS5gkiy6yb:bd33b7fb-c41d-48da-b01e-3d99b5ee1c0c"
```

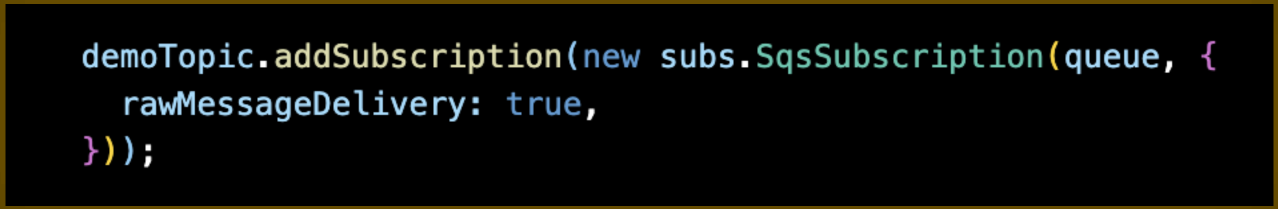
Demo – The Lambda Q consumer event parameter



```
2024-11-13T14:11:26.883Z        430f822c-e8c2-5f93-9558-dd825d79b1f4    INFO    Event
{
    "Records": [
        {
            "messageId": "54e97653-e433-4468-a777-47f9173965b5",
            "receiptHandle":
"AQEBvAxWjF4P6iVx9SkgVBVf/8Qxduz7TKjT1crfZaW2FbbFcik4zKOG73TZhMLYok2hYGi2qL9wKCP3ZuKbZCDpjq9xb56k8Bd4KL1B/MSykwklYTjgMyr39lSrW2DQEPIbidDV9wq3V+UFaIswf01Xbui5uTeRdAWn/RWxai7Iov8gIWtSW7T9o5PB3G7GGDYD8+I1i5CNDoDvwpWQzTGOZsSe7NyY/Sif15gJ4e4RK3Mqzv17GetrC0eSiloOxQ9Qikjh7UvyFqX5+th3Qn78fRT4CCn6H04RQ90Ho1U17SsQzgzwVUK97gUS0h5Do68zeBho7EERNn39oTJ8AC6Wsg2r+Fp+DkGOKZK9qUWhcbCGMVv9pjeR4nNsHNK6M6SAeVuR7I4SPGY8Gvnkax0eQHqDeC60sL1gROsVBJ4+i3Q=",
            "body": "{\n  \"Type\" : \"Notification\",\n  \"MessageId\" : \"ca265fbe-2517-5406-b2a7-001090d3567e\",\n  \"TopicArn\" : \"arn:aws:sns:eu-west-1:517039770760:SHSDemoStack-DemoTopic2BE41B12-42ZS5gkiy6yb\",\n  \"Message\" : \"{\\n    \\\"name\\\" : \\\"Diarmuid O' Connor\\\",\\n    \\\"address\\\" : \\\"1 Main Street\\\",\\n    \\\"email\\\": \\\"doconnor@wit.ie\\\"\\n}\",\n  \"Timestamp\" : \"2024-11-13T14:11:16.186Z\",\n  \"SignatureVersion\" : \"1\",\n  \"Signature\" : \"GZbtzU7CGMNY1v7hec42Mb60NFDwnqe7ytlHB2XnPrP1D8gbNA/rOViAbFrZvpuFHqiLyk0j4C1eMbnoMzZF8vPZ4n5UHSoytarhnJFJ9H/BV526nJTYNg5jEBOWJXg+19rH0PNymKeKKXKZR7AiYgghDg1Twyxhc+MPqjHBF5XDs0JY9v3Ugjj/4drqaZMFJFGC/42mITl8Wv3e8xA5BdSVBKgPeN4cVUWMUVJ8LCUuncucP1WbIOBprIkOneX4gOeqtqMrTsesfYGgcgY6M9gvzMLEAWiOeADnzSuJdk5ksTAvm3udDb6P5pRAw2tv8Ul9XmJDW1JVze4YV8rlAw==\",\n  \"SigningCertURL\" : \"https://sns.eu-west-1.amazonaws.com/SimpleNotificationService-9c6465fa7f48f5cacd23014631ec1136.pem\",\n  \"UnsubscribeURL\" : \"https://sns.eu-west-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:eu-west-1:517039770760:SHSDemoStack-DemoTopic2BE41B12-42ZS5gkiy6yb:8c3a3bba-ff77-4854-974b-a32dff8163d0\"\n}",
            "attributes": {
                "ApproximateReceiveCount": "1",
                "SentTimestamp": "1731507076218",
                "SenderId": "AIDAISMY7JYY5F7RTT6AO",
                "ApproximateFirstReceiveTimestamp": "1731507076225"
            },
```

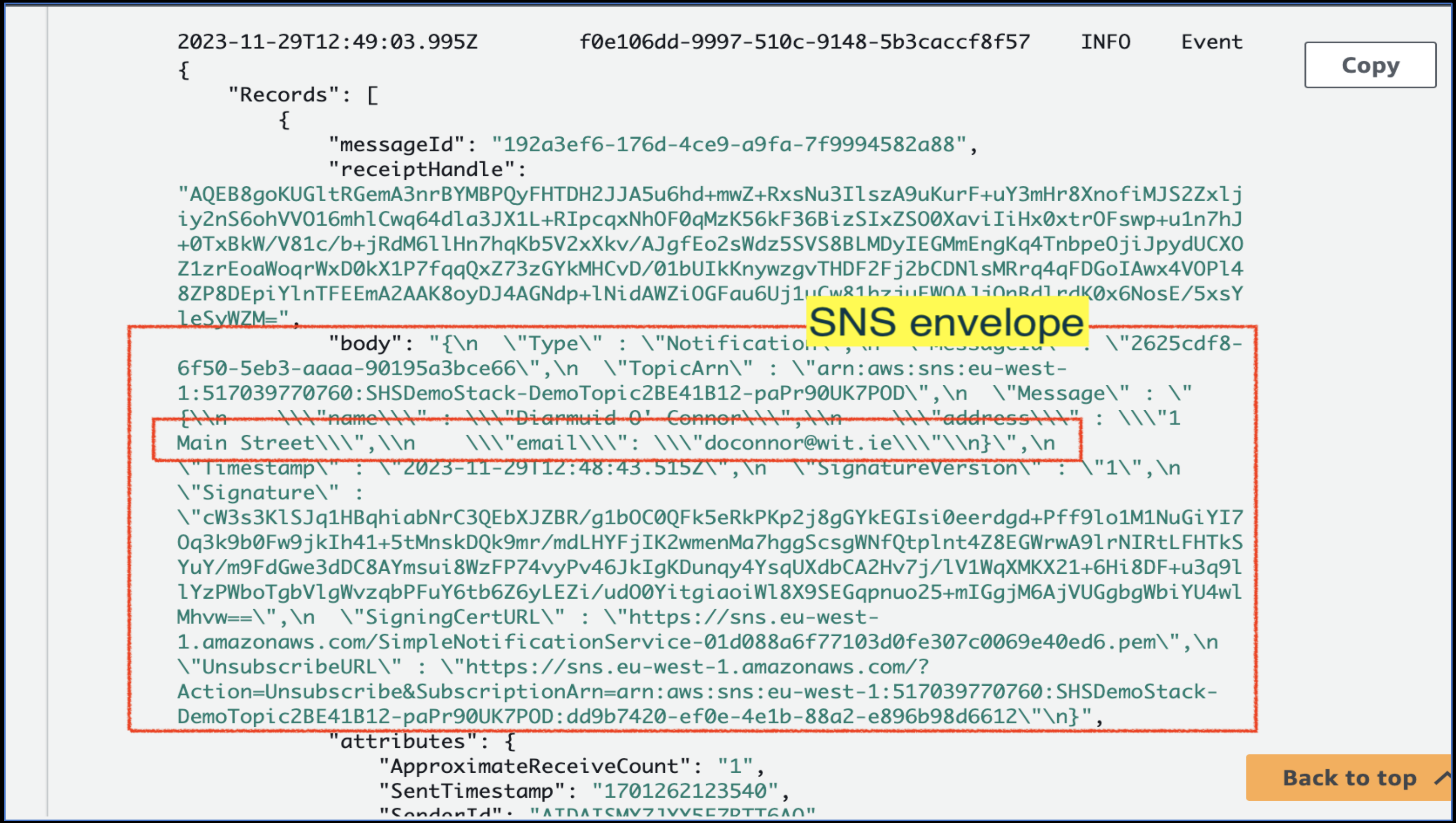
The SNS envelope.

- SNS wraps the source message in an envelope before sending it to an SQS queue subscriber.
    - Configurable



```
demoTopic.addSubscription(new subs.SqsSubscription(queue, {
  rawMessageDelivery: true,
}));
```

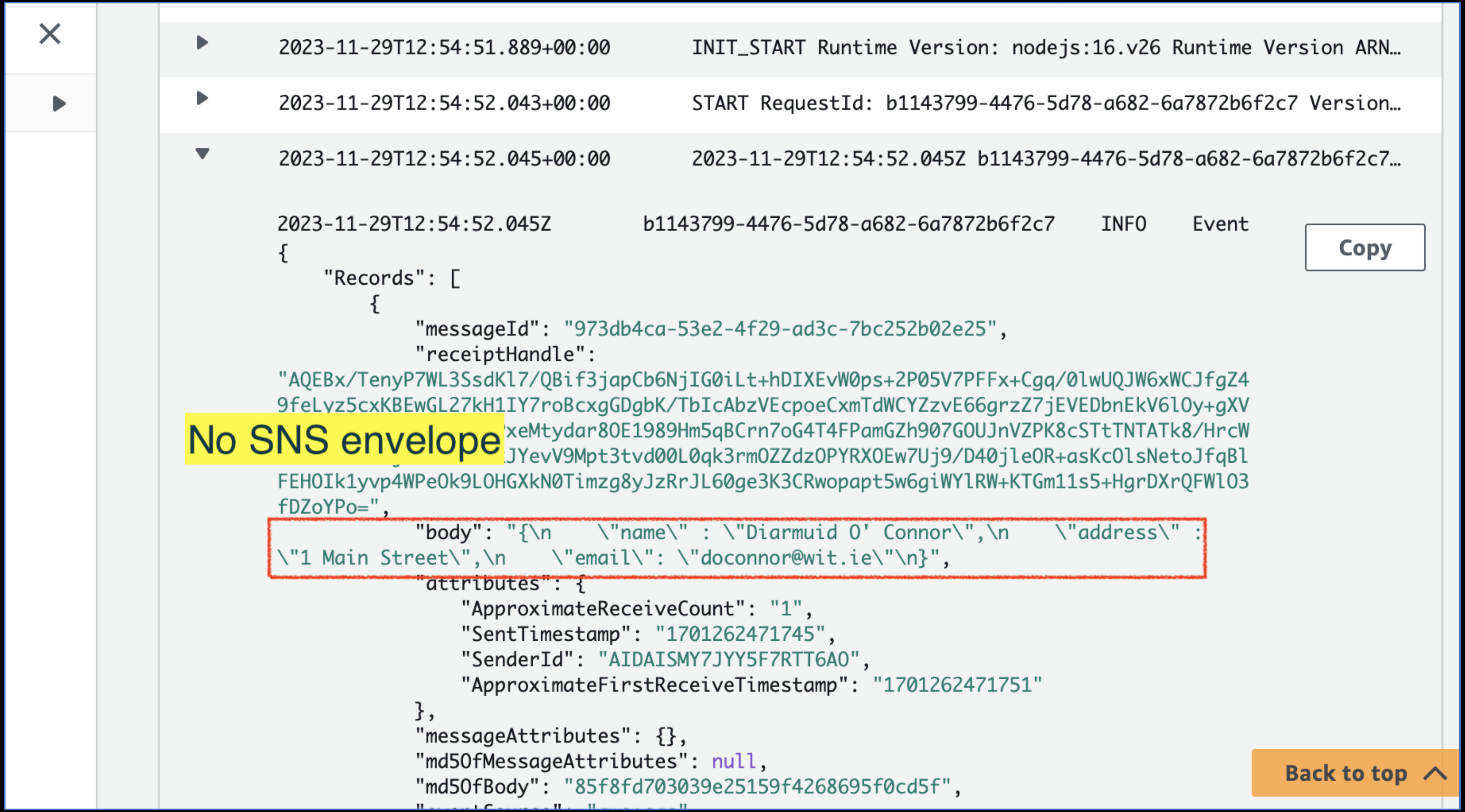
The SNS envelope.



```
2023-11-29T12:49:03.995Z	f0e106dd-9997-510c-9148-5b3caccf8f57	INFO	Event
{
    "Records": [
        {
            "messageId": "192a3ef6-176d-4ce9-a9fa-7f9994582a88",
            "receiptHandle":
"AQEB8goKUGltRGemA3nrBYMBPQyFHTDH2JJA5u6hd+mwZ+RxsNu3IlszA9uKurF+uY3mHr8XnofiMJS2Zxlj
iy2nS6ohVVO16mhlCwq64dla3JX1L+RIpcqxNhOF0qMzK56kF36BizSIxZSO0XaviIiHx0xtrOFswp+u1n7hJ
+0TxBkW/V81c/b+jRdM6llHn7hqKb5V2xXkv/AJgfEo2sWdz5SVS8BLMDyIEGMmEngKq4TnbpeOjiJpydUCXO
Z1zrEoaWoqrWxD0kX1P7fqqQxZ73zGYkMHCvD/01bUIkKnywzgvTHDF2Fj2bCDNlsMRrq4qFDGoIAwx4VOPl4
8ZP8DEpiYlnTFEEmA2AAK8oyDJ4AGNdp+lNidAWZiOGFau6Uj1uCw81hzjuFWOAljOnBdlrdK0x6NosE/5xsY
leSyWZM=",
            "body": "{\n  \"Type\" : \"Notification\",\n  \"MessageId\" : \"2625cdf8-
6f50-5eb3-aaaa-90195a3bce66\",\n  \"TopicArn\" : \"arn:aws:sns:eu-west-
1:517039770760:SHSDemoStack-DemoTopic2BE41B12-paPr90UK7POD\",\n  \"Message\" : \"
{\\n    \\\"name\\\" : \\\"Diarmuid O' Connor\\\",\\n    \\\"address\\\" : \\\"1
Main Street\\\",\\n    \\\"email\\\": \\\"doconnor@wit.ie\\\"\\n}\",\n
\"Timestamp\" : \"2023-11-29T12:48:43.515Z\",\n  \"SignatureVersion\" : \"1\",\n
\"Signature\" :
\"cW3s3KlSJq1HBqhiabNrC3QEbXJZBR/g1bOC0QFk5eRkPKp2j8gGYkEGIsi0eerdgd+Pff9lo1M1NuGiYI7
Oq3k9b0Fw9jkIh41+5tMnskDQk9mr/mdLHYFjIK2wmenMa7hggScsgWNfQtplnt4Z8EGWrwA9lrNIRtLFHTkS
YuY/m9FdGwe3dDC8AYmsui8WzFP74vyPv46JkIgKDunqy4YsqUXdbCA2Hv7j/lV1WqXMKX21+6Hi8DF+u3q9l
lYzPWboTgbVlgWvzqbPFuY6tb6Z6yLEZi/udO0YitgiaoiWl8X9SEGqpnuo25+mIGgjM6AjVUGgbgWbiYU4wl
Mhvw==\",\n  \"SigningCertURL\" : \"https://sns.eu-west-
1.amazonaws.com/SimpleNotificationService-01d088a6f77103d0fe307c0069e40ed6.pem\",\n
\"UnsubscribeURL\" : \"https://sns.eu-west-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:eu-west-1:517039770760:SHSDemoStack-
DemoTopic2BE41B12-paPr90UK7POD:dd9b7420-ef0e-4e1b-88a2-e896b98d6612\"\n}",
            "attributes": {
                "ApproximateReceiveCount": "1",
                "SentTimestamp": "1701262123540",
                "SenderId": "AIDAISMYZJYY5EZPTT6AO"
```

SNS envelope

The SNS envelope.



```
2023-11-29T12:54:51.889+00:00       INIT_START Runtime Version: nodejs:16.v26 Runtime Version ARN…
2023-11-29T12:54:52.043+00:00       START RequestId: b1143799-4476-5d78-a682-6a7872b6f2c7 Version…
2023-11-29T12:54:52.045+00:00       2023-11-29T12:54:52.045Z b1143799-4476-5d78-a682-6a7872b6f2c7…

2023-11-29T12:54:52.045Z      b1143799-4476-5d78-a682-6a7872b6f2c7     INFO     Event
{
    "Records": [
        {
            "messageId": "973db4ca-53e2-4f29-ad3c-7bc252b02e25",
            "receiptHandle":
"AQEBx/TenyP7WL3SsdKl7/QBif3japCb6NjIG0iLt+hDIXEvW0ps+2P05V7PFFx+Cgq/0lwUQJW6xWCJfgZ4
9feLyz5cxKBEwGL27kH1IY7roBcxgGDgbK/TbIcAbzVEcpoeCxmTdWCYZzvE66grzZ7jEVEDbnEkV6lOy+gXV
                          xeMtydar8OE1989Hm5qBCrn7oG4T4FPamGZh907GOUJnVZPK8cSTtTNTATk8/HrcW
                          JYevV9Mpt3tvd00L0qk3rmOZZdzOPYRXOEw7Uj9/D40jleOR+asKcOlsNetoJfqBl
FEHOIk1yvp4WPeOk9LOHGXkN0Timzg8yJzRrJL60ge3K3CRwopapt5w6giWYlRW+KTGm11s5+HgrDXrQFWlO3
fDZoYPo=",
            "body": "{\n    \"name\" : \"Diarmuid O' Connor\",\n    \"address\" :
\"1 Main Street\",\n    \"email\": \"doconnor@wit.ie\"\n}",
            "attributes": {
                "ApproximateReceiveCount": "1",
                "SentTimestamp": "1701262471745",
                "SenderId": "AIDAISMY7JYY5F7RTT6AO",
                "ApproximateFirstReceiveTimestamp": "1701262471751"
            },
            "messageAttributes": {},
            "md5OfMessageAttributes": null,
            "md5OfBody": "85f8fd703039e25159f4268695f0cd5f",
```

No SNS envelope

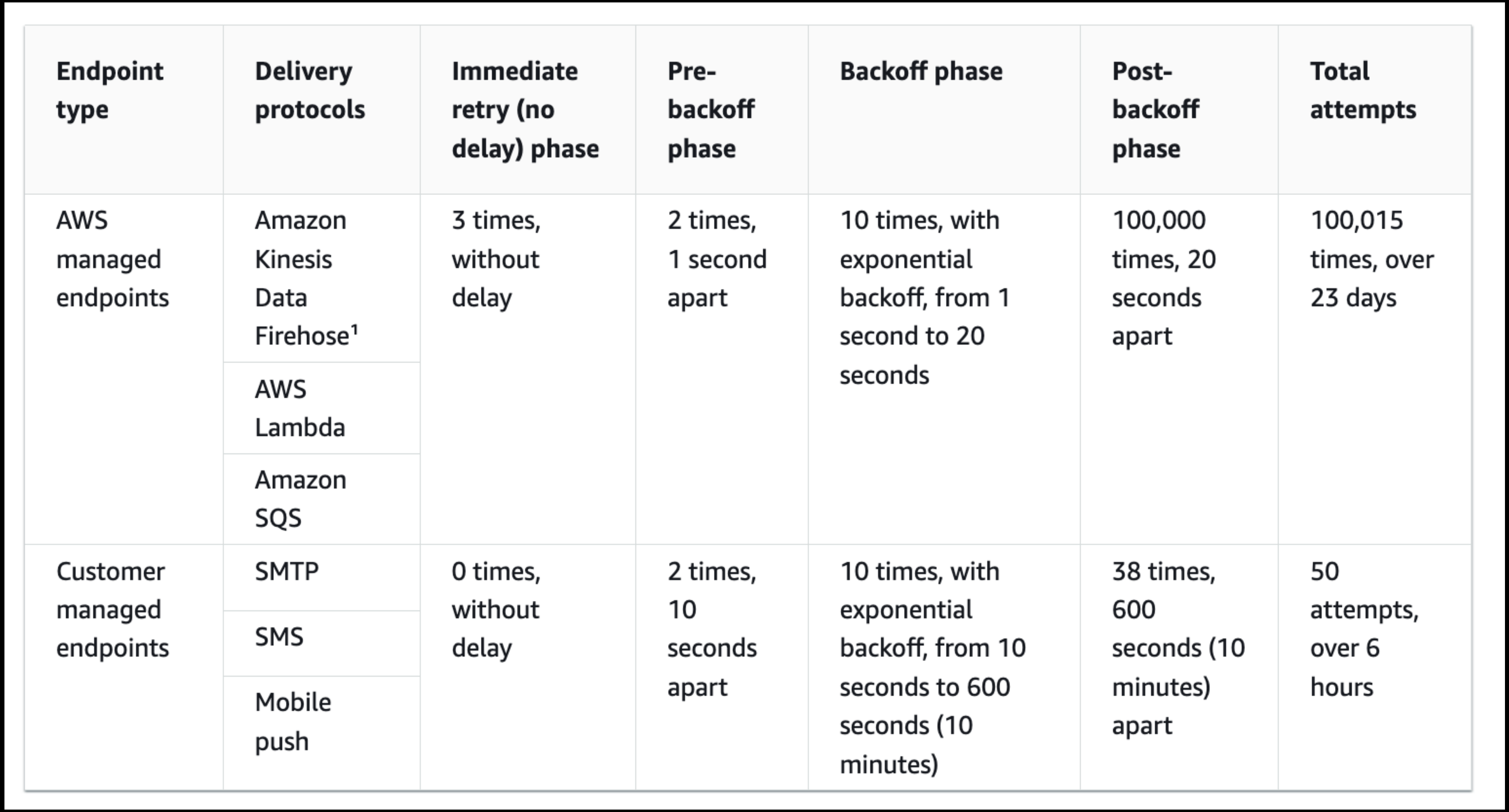
Lambda Vs { SQS → Lambda } subscribers

- Disadvantages (Lambda subscriber)
    - No Batching is available when processing messages from SNS.
    - No control on Lambda Concurrency, messages are processed one by one as soon as they arrive.
    - Lambda function is responsible for handling errors/retries
    - Lambda DLQ needs to be handled separately.
- Advantages (Lambda subscriber)
    - Good for time-critical processing.

SNS - Delivery protocols and policies.

- SNS defines a delivery policy for each delivery protocol.
- The policy defines how SNS retries the delivery of messages when server-side errors occur (when the system that hosts the subscribed endpoint becomes unavailable). When the delivery policy is exhausted, SNS stops retrying the delivery and discards the message.
    - A DLQ can be assigned for this case.

SNS - Delivery protocols and policies.



Endpoint type	Delivery protocols	Immediate retry (no delay) phase	Pre-backoff phase	Backoff phase	Post-backoff phase	Total attempts
AWS managed endpoints	Amazon Kinesis Data Firehose[1]	3 times, without delay	2 times, 1 second apart	10 times, with exponential backoff, from 1 second to 20 seconds	100,000 times, 20 seconds apart	100,015 times, over 23 days
	AWS Lambda					
	Amazon SQS					
Customer managed endpoints	SMTP	0 times, without delay	2 times, 10 seconds apart	10 times, with exponential backoff, from 10 seconds to 600 seconds (10 minutes)	38 times, 600 seconds (10 minutes) apart	50 attempts, over 6 hours
	SMS					
	Mobile push					

Lambda subscriber DLQ

- SNS invoks a lambda function subscriber asynchronously.
  - SNS does not wait for a response.

    ⇒ Lambda service must handle function failures cases.

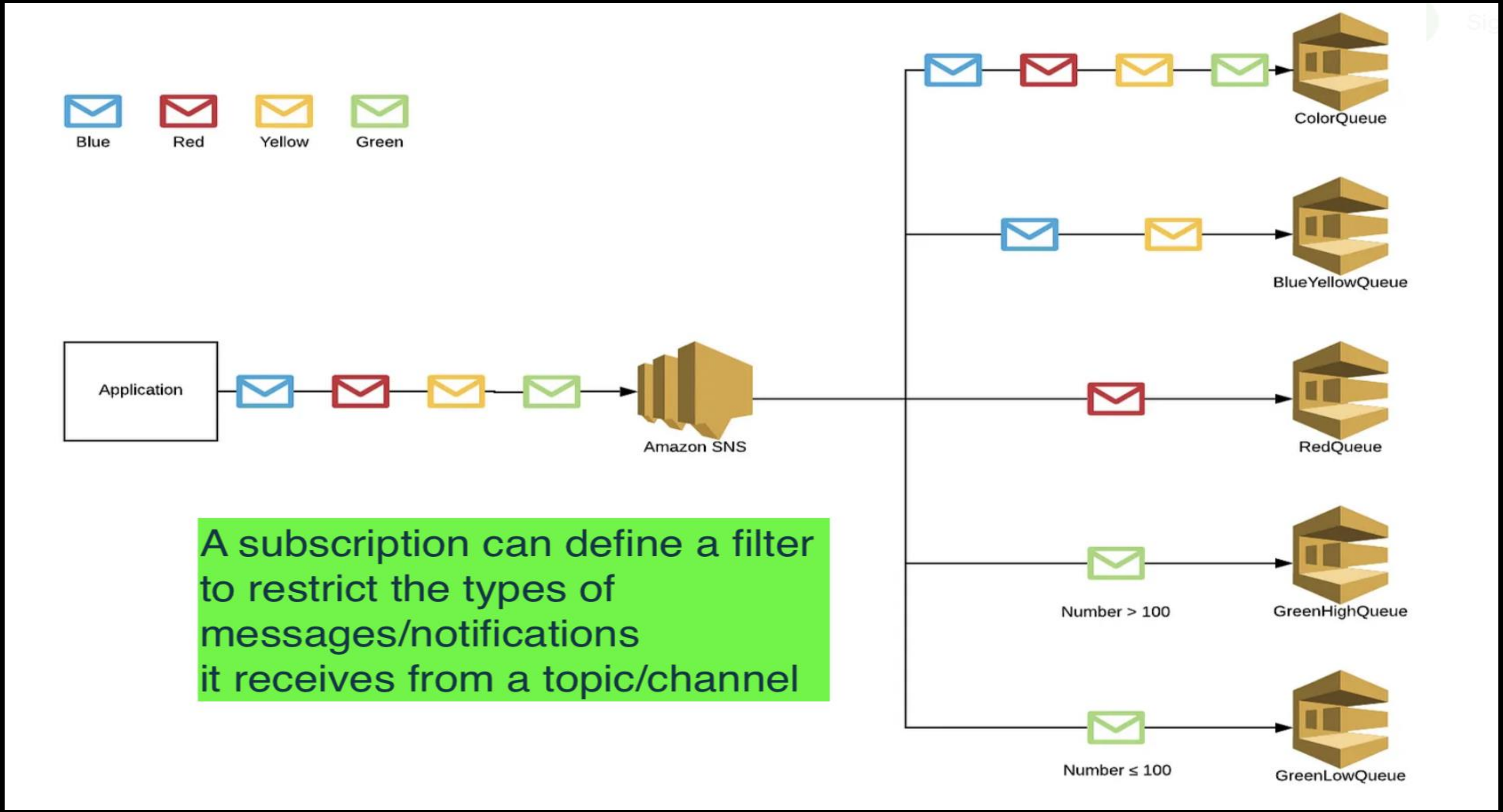
- Ex.: Architecture:

AWS CLI (Pub) → SNS Topic → Lambda (Sub\

|

|→ DLQ → Lambda (Consumer)

Fan Out pattern - Filtering.



A subscription can define a filter to restrict the types of messages/notifications it receives from a topic/channel

Filtering

- Filtering policy can be based on message <u>attributes</u> or the message <u>body</u>.

- Filtering criteria:
  1. String Filter.
     - Conditions - allowList, denyList, matchPrefixes
  2. Numeric Filter.
     - Conditions – allowList, greaterThan, lessThan, between. …
  3. Exists Filter.

→ DLQ → Lambda (Consumer)

Demo – Message Attribute Filtering

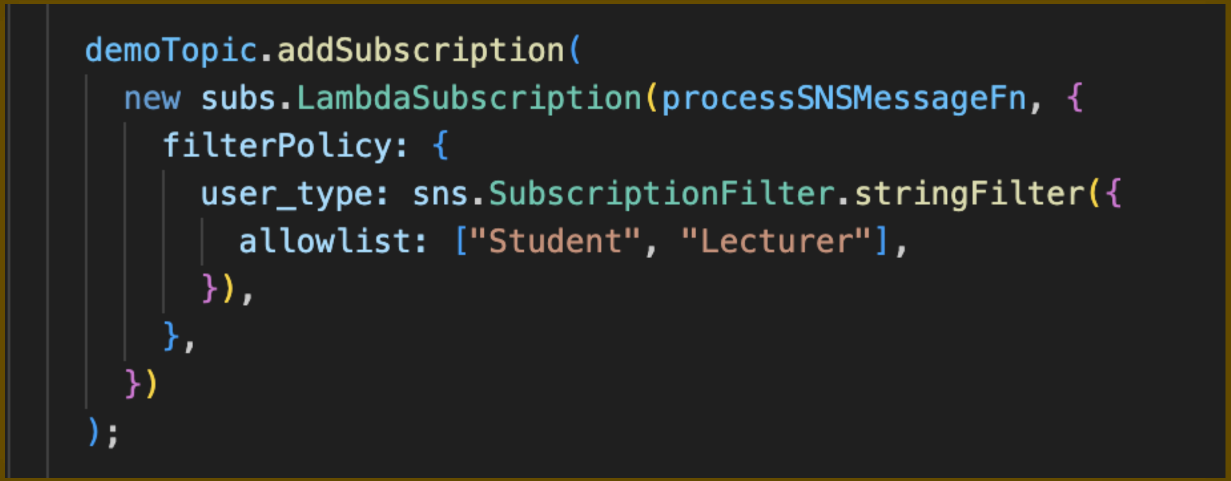
- Architecture:

  AWS CLI (Pub) → SNS Topic -- <Filter> → Lambda (Sub)

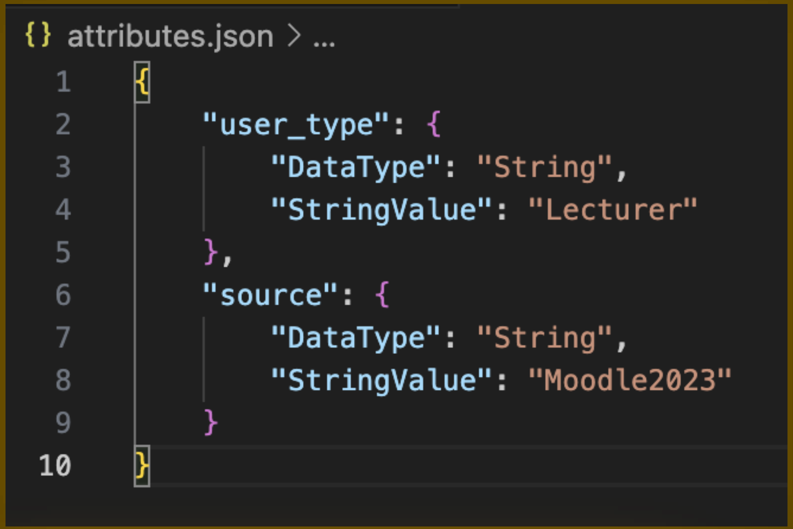
  -- <Filter> → SQS Q -→ Lambda (Con)

  - The Lambda subscription only concerns messages with the attribute user_type set to Student or Lecturer.

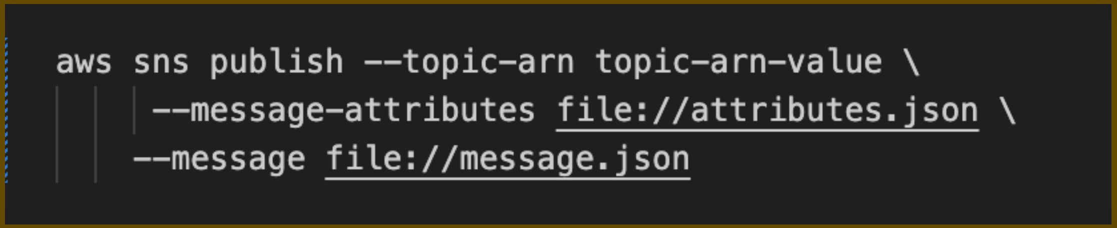
Demo – Message Attribute Filtering



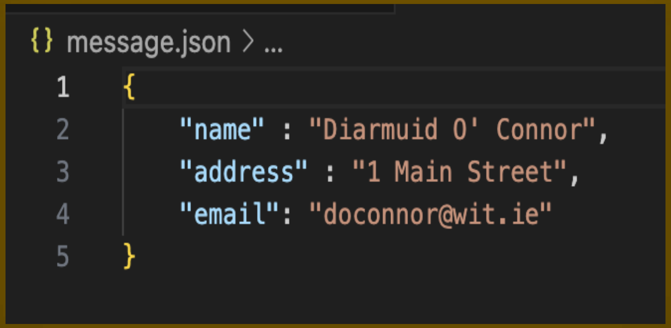
```
demoTopic.addSubscription(
  new subs.LambdaSubscription(processSNSMessageFn, {
    filterPolicy: {
      user_type: sns.SubscriptionFilter.stringFilter({
        allowlist: ["Student", "Lecturer"],
      }),
    },
  })
);
```



```
{
    "user_type": {
        "DataType": "String",
        "StringValue": "Lecturer"
    },
    "source": {
        "DataType": "String",
        "StringValue": "Moodle2023"
    }
}
```

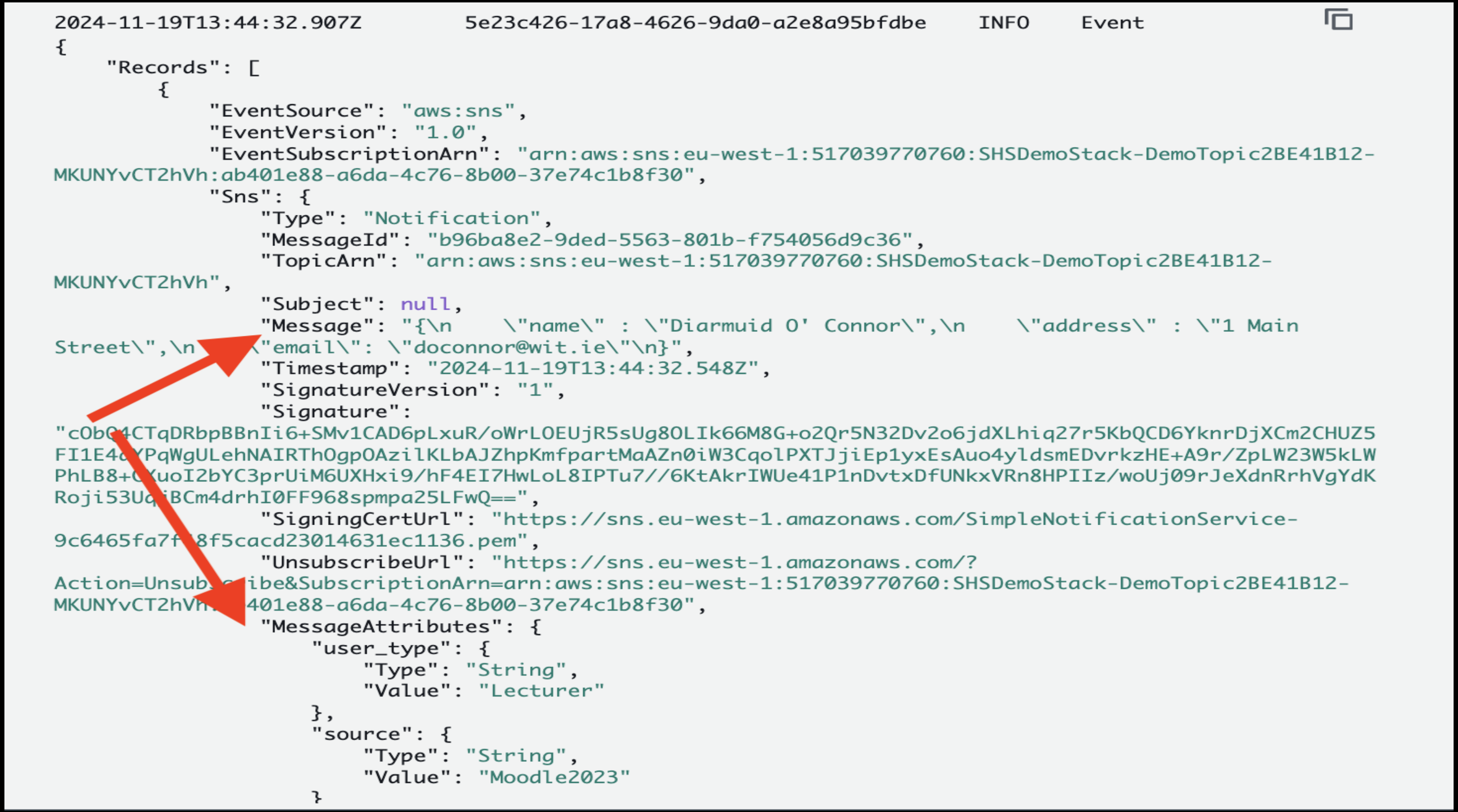


```
aws sns publish --topic-arn topic-arn-value \
     --message-attributes file://attributes.json \
    --message file://message.json
```



```
{
    "name" : "Diarmuid O' Connor",
    "address" : "1 Main Street",
    "email": "doconnor@wit.ie"
}
```

Demo – Message Attribute Filtering



```
2024-11-19T13:44:32.907Z        5e23c426-17a8-4626-9da0-a2e8a95bfdbe    INFO    Event
{
    "Records": [
        {
            "EventSource": "aws:sns",
            "EventVersion": "1.0",
            "EventSubscriptionArn": "arn:aws:sns:eu-west-1:517039770760:SHSDemoStack-DemoTopic2BE41B12-
MKUNYvCT2hVh:ab401e88-a6da-4c76-8b00-37e74c1b8f30",
            "Sns": {
                "Type": "Notification",
                "MessageId": "b96ba8e2-9ded-5563-801b-f754056d9c36",
                "TopicArn": "arn:aws:sns:eu-west-1:517039770760:SHSDemoStack-DemoTopic2BE41B12-
MKUNYvCT2hVh",
                "Subject": null,
                "Message": "{\n    \"name\" : \"Diarmuid O' Connor\",\n    \"address\" : \"1 Main
Street\",\n    \"email\": \"doconnor@wit.ie\"\n}",
                "Timestamp": "2024-11-19T13:44:32.548Z",
                "SignatureVersion": "1",
                "Signature":
"cObQ4CTqDRbpBBnIi6+SMv1CAD6pLxuR/oWrLOEUjR5sUg8OLIk66M8G+o2Qr5N32Dv2o6jdXLhiq27r5KbQCD6YknrDjXCm2CHUZ5
FI1E4aYPqWgULehNAIRThOgpOAzilKLbAJZhpKmfpartMaAZn0iW3CqolPXTJjiEp1yxEsAuo4yldsmEDvrkzHE+A9r/ZpLW23W5kLW
PhLB8+CXuoI2bYC3prUiM6UXHxi9/hF4EI7HwLoL8IPTu7//6KtAkrIWUe41P1nDvtxDfUNkxVRn8HPIIz/woUj09rJeXdnRrhVgYdK
Roji53UqjBCm4drhI0FF968spmpa25LFwQ==",
                "SigningCertUrl": "https://sns.eu-west-1.amazonaws.com/SimpleNotificationService-
9c6465fa7f48f5cacd23014631ec1136.pem",
                "UnsubscribeUrl": "https://sns.eu-west-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:eu-west-1:517039770760:SHSDemoStack-DemoTopic2BE41B12-
MKUNYvCT2hVh:ab401e88-a6da-4c76-8b00-37e74c1b8f30",
                "MessageAttributes": {
                    "user_type": {
                        "Type": "String",
                        "Value": "Lecturer"
                    },
                    "source": {
                        "Type": "String",
                        "Value": "Moodle2023"
                    }
```

Message Body Filtering

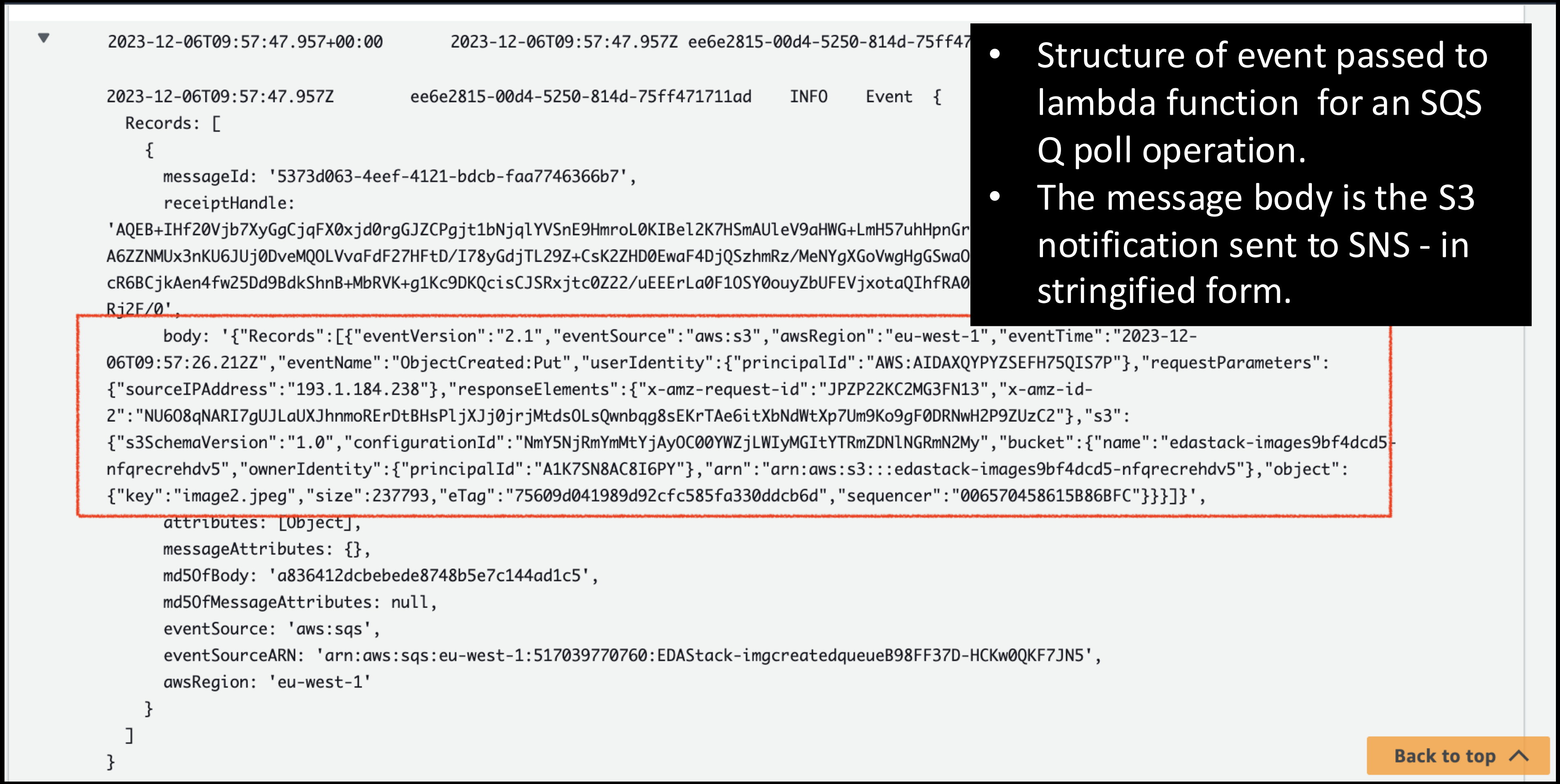
- Filtering based on properties of the message body.
  - Body is a JSON structure.
- Same filtering criteria as attribute filtering

- Demo – Architecture:

  S3 (Pub) → SNS --<Filter>→ SQS (Sub) → Lambda fn (Con)

  The filter only allows messages for objects named ‘image*’ uploaded to an S3 bucket.

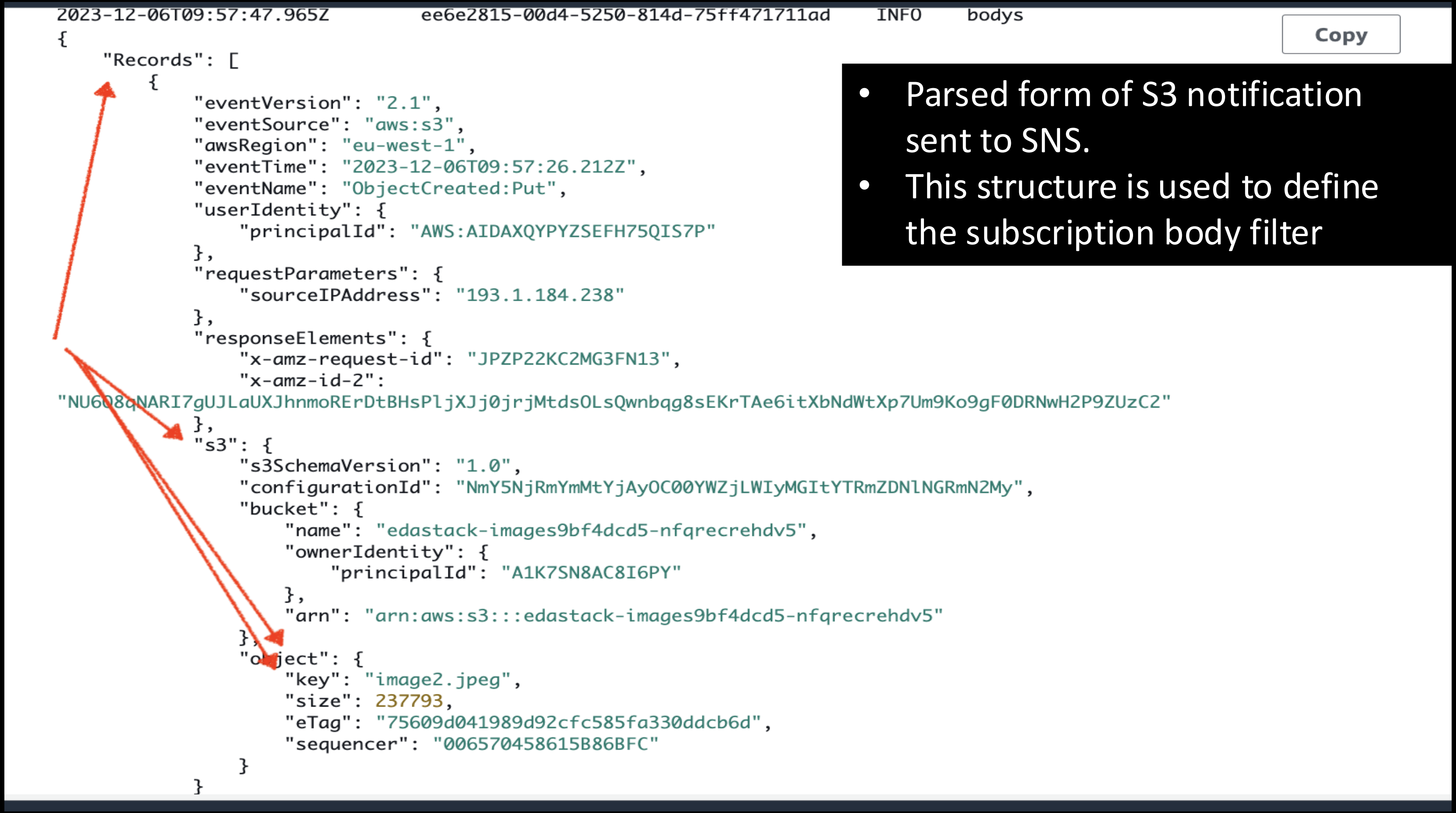
Demo - Understand the message structure



```
▼     2023-12-06T09:57:47.957+00:00       2023-12-06T09:57:47.957Z ee6e2815-00d4-5250-814d-75ff47

      2023-12-06T09:57:47.957Z        ee6e2815-00d4-5250-814d-75ff471711ad    INFO    Event  {
        Records: [
          {
            messageId: '5373d063-4eef-4121-bdcb-faa7746366b7',
            receiptHandle:
      'AQEB+IHf20Vjb7XyGgCjqFX0xjd0rgGJZCPgjt1bNjqlYVSnE9HmroL0KIBel2K7HSmAUleV9aHWG+LmH57uhHpnGr
      A6ZZNMUx3nKU6JUj0DveMQOLVvaFdF27HFtD/I78yGdjTL29Z+CsK2ZHD0EwaF4DjQSzhmRz/MeNYgXGoVwgHgGSwaO
      cR6BCjkAen4fw25Dd9BdkShnB+MbRVK+g1Kc9DKQcisCJSRxjtc0Z22/uEEErLa0F1OSY0ouyZbUFEVjxotaQIhfRA0
      Rj2F/0',
            body: '{"Records":[{"eventVersion":"2.1","eventSource":"aws:s3","awsRegion":"eu-west-1","eventTime":"2023-12-
      06T09:57:26.212Z","eventName":"ObjectCreated:Put","userIdentity":{"principalId":"AWS:AIDAXQYPYZSEFH75QIS7P"},"requestParameters":
      {"sourceIPAddress":"193.1.184.238"},"responseElements":{"x-amz-request-id":"JPZP22KC2MG3FN13","x-amz-id-
      2":"NU6O8qNARI7gUJLaUXJhnmoRErDtBHsPljXJj0jrjMtdsOLsQwnbqg8sEKrTAe6itXbNdWtXp7Um9Ko9gF0DRNwH2P9ZUzC2"},"s3":
      {"s3SchemaVersion":"1.0","configurationId":"NmY5NjRmYmMtYjAyOC00YWZjLWIyMGItYTRmZDNlNGRmN2My","bucket":{"name":"edastack-images9bf4dcd5-
      nfqrecrehdv5","ownerIdentity":{"principalId":"A1K7SN8AC8I6PY"},"arn":"arn:aws:s3:::edastack-images9bf4dcd5-nfqrecrehdv5"},"object":
      {"key":"image2.jpeg","size":237793,"eTag":"75609d041989d92cfc585fa330ddcb6d","sequencer":"00657045861 5B86BFC"}}}]}',
            attributes: [Object],
            messageAttributes: {},
            md5OfBody: 'a836412dcbebede8748b5e7c144ad1c5',
            md5OfMessageAttributes: null,
            eventSource: 'aws:sqs',
            eventSourceARN: 'arn:aws:sqs:eu-west-1:517039770760:EDAStack-imgcreatedqueueB98FF37D-HCKw0QKF7JN5',
            awsRegion: 'eu-west-1'
          }
        ]
      }
```

- Structure of event passed to lambda function for an SQS Q poll operation.
- The message body is the S3 notification sent to SNS - in stringified form.

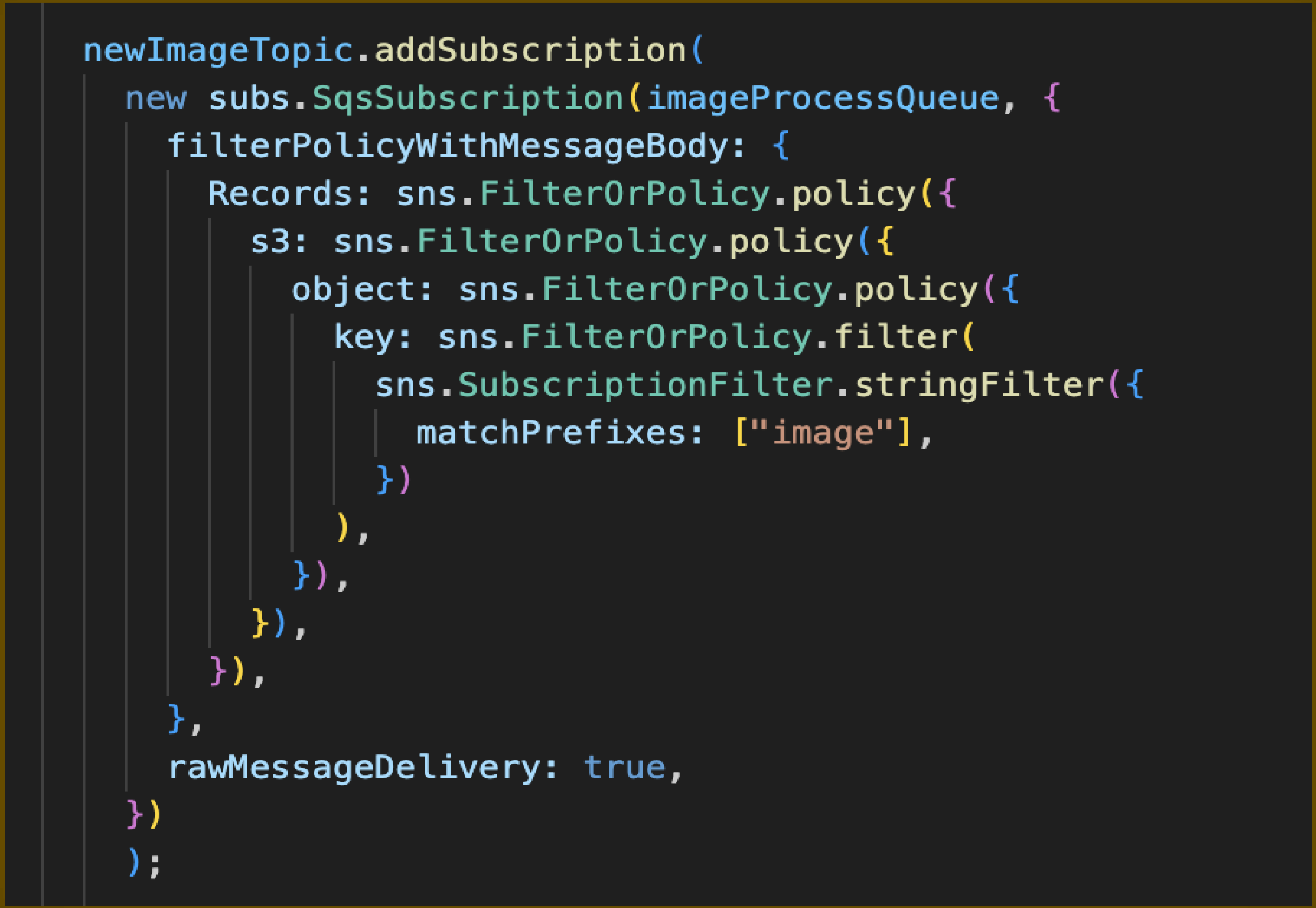
Demo - Understand the message structure



```
2023-12-06T09:57:47.965Z        ee6e2815-00d4-5250-814d-75ff471711ad    INFO    bodys
{
    "Records": [
        {
            "eventVersion": "2.1",
            "eventSource": "aws:s3",
            "awsRegion": "eu-west-1",
            "eventTime": "2023-12-06T09:57:26.212Z",
            "eventName": "ObjectCreated:Put",
            "userIdentity": {
                "principalId": "AWS:AIDAXQYPYZSEFH75QIS7P"
            },
            "requestParameters": {
                "sourceIPAddress": "193.1.184.238"
            },
            "responseElements": {
                "x-amz-request-id": "JPZP22KC2MG3FN13",
                "x-amz-id-2":
"NU6Q8qNARI7gUJLaUXJhnmoRErDtBHsPljXJj0jrjMtdsOLsQwnbqg8sEKrTAe6itXbNdWtXp7Um9Ko9gF0DRNwH2P9ZUzC2"
            },
            "s3": {
                "s3SchemaVersion": "1.0",
                "configurationId": "NmY5NjRmYmMtYjAyOC00YWZjLWIyMGItYTRmZDNlNGRmN2My",
                "bucket": {
                    "name": "edastack-images9bf4dcd5-nfqrecrehdv5",
                    "ownerIdentity": {
                        "principalId": "A1K7SN8AC8I6PY"
                    },
                    "arn": "arn:aws:s3:::edastack-images9bf4dcd5-nfqrecrehdv5"
                },
                "object": {
                    "key": "image2.jpeg",
                    "size": 237793,
                    "eTag": "75609d041989d92cfc585fa330ddcb6d",
                    "sequencer": "006570458615B86BFC"
                }
            }
```

- Parsed form of S3 notification sent to SNS.
- This structure is used to define the subscription body filter

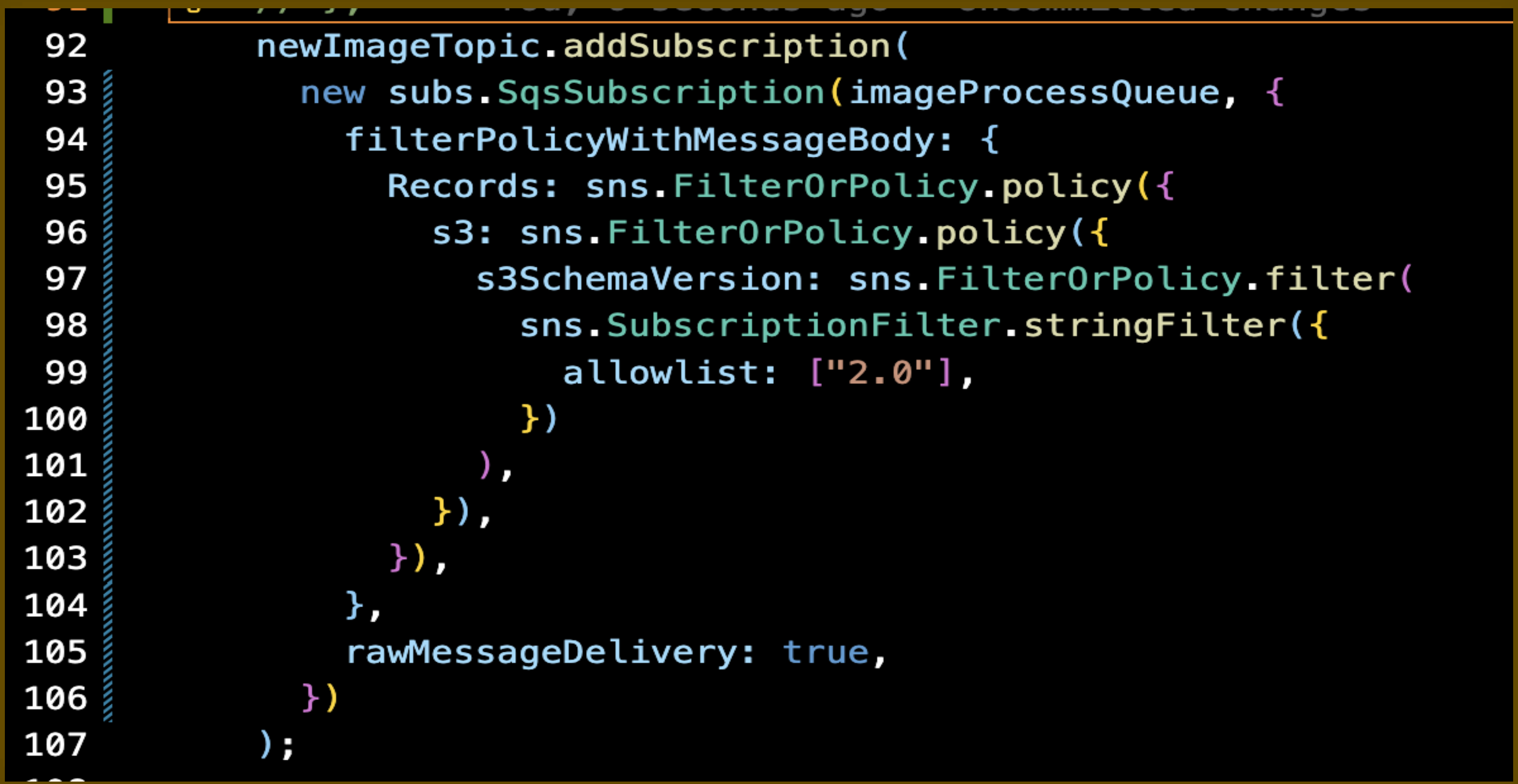
Demo – Filter Policy



```
newImageTopic.addSubscription(
  new subs.SqsSubscription(imageProcessQueue, {
    filterPolicyWithMessageBody: {
      Records: sns.FilterOrPolicy.policy({
        s3: sns.FilterOrPolicy.policy({
          object: sns.FilterOrPolicy.policy({
            key: sns.FilterOrPolicy.filter(
              sns.SubscriptionFilter.stringFilter({
                matchPrefixes: ["image"],
              })
            ),
          }),
        }),
      }),
    },
    rawMessageDelivery: true,
  })
);
```

Demo – Filter Policy

- Ex. 2 Only allow messages with the S3 schema version of 2.0.



```
newImageTopic.addSubscription(
  new subs.SqsSubscription(imageProcessQueue, {
    filterPolicyWithMessageBody: {
      Records: sns.FilterOrPolicy.policy({
        s3: sns.FilterOrPolicy.policy({
          s3SchemaVersion: sns.FilterOrPolicy.filter(
            sns.SubscriptionFilter.stringFilter({
              allowlist: ["2.0"],
            })
          ),
        }),
      }),
    },
    rawMessageDelivery: true,
  })
);
```

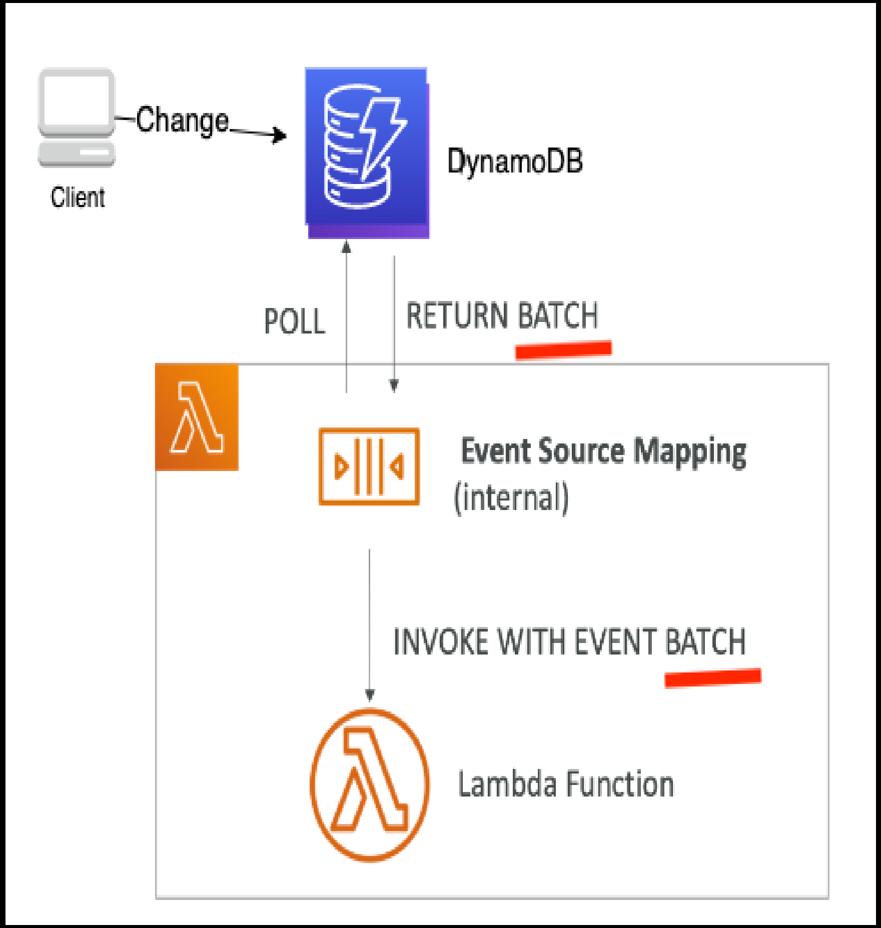
Reminder (Why)

- It’s better to decouple an app’s compute components by using a range of AWS services/techniques:
  - SQS: queue model.
  - SNS: pub/sub/subscribe model.
  - Data streams.
  - Triggers.

- These techniques result in:
  - Reduces latency; Increase availability; Reduces complexity (by decreasing dependency).

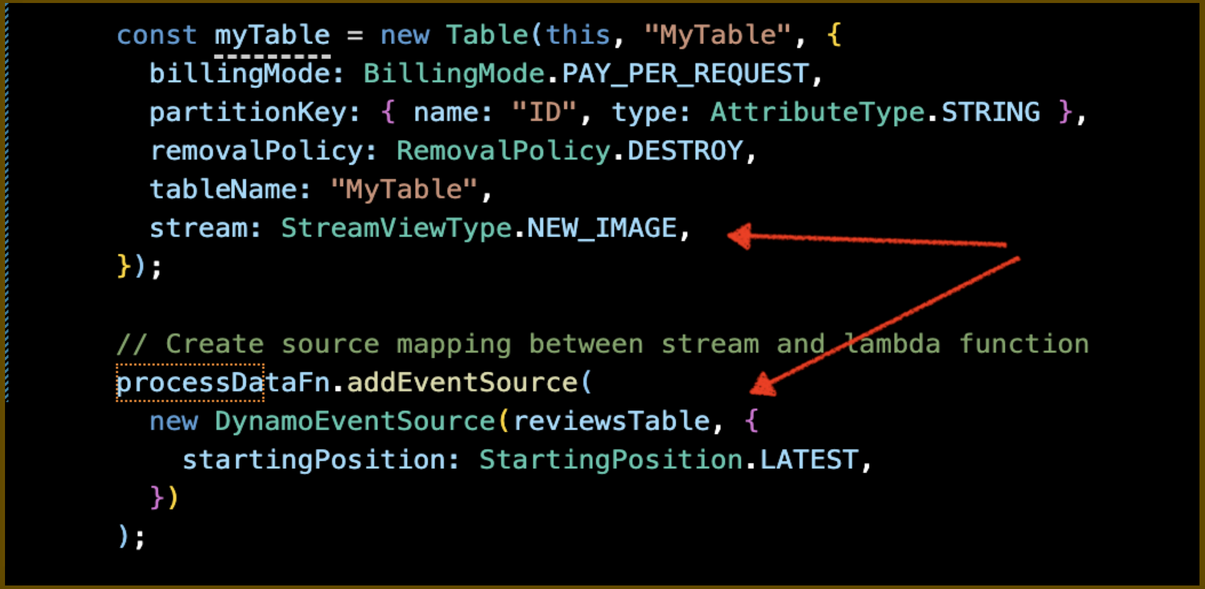
Lambda – Event source mappings

- Sources: DynamoDB streams, SQS, Kinesis streams.
- Common denominator: events are polled from the source.
- Your Lambda function is invoked synchronously by the Lambda service. The events are placed in an Event Queue.
- Lambda service retries retry on errors.3 tries with exponential backoff



DynamoDB streams

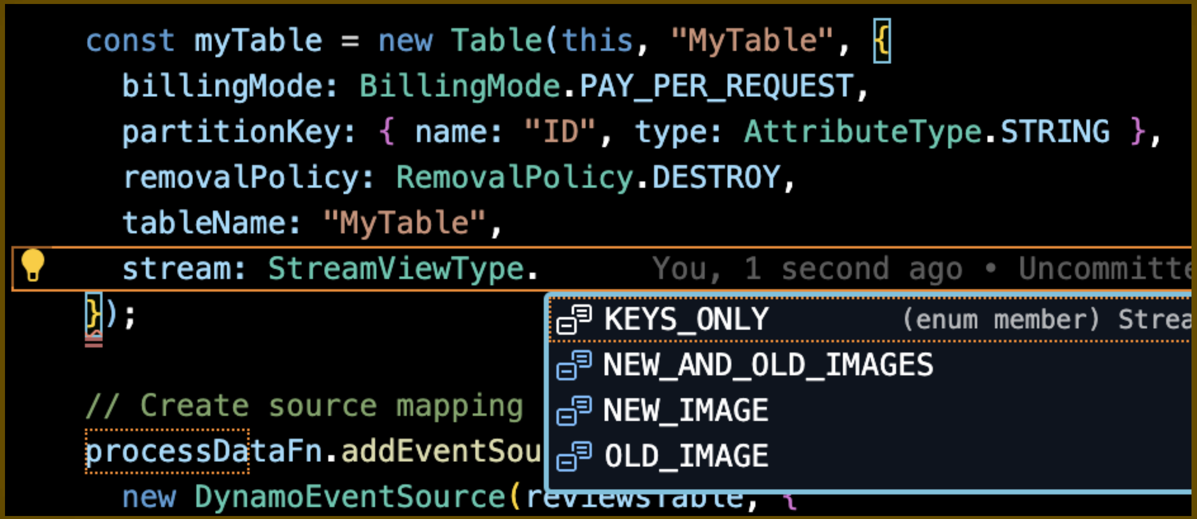
- Declare the stream in the Table.
- Declare the stream as an event source (trigger) for the lambda function.
- Tip: console.log the event in the function to see its structure



```
const myTable = new Table(this, "MyTable", {
  billingMode: BillingMode.PAY_PER_REQUEST,
  partitionKey: { name: "ID", type: AttributeType.STRING },
  removalPolicy: RemovalPolicy.DESTROY,
  tableName: "MyTable",
  stream: StreamViewType.NEW_IMAGE,
});

// Create source mapping between stream and lambda function
processDataFn.addEventSource(
  new DynamoEventSource(reviewsTable, {
    startingPosition: StartingPosition.LATEST,
  })
);
```

DynamoDB stream types

- DynamoDB stream type controls what is written to the stream.
  - NEW_IMAGE – The entire table item, after it was modified.
  - NEW_AND_OLD_IMAGES - Both the pre- and post-modification states of the item.
  - OLD_IMAGE - The tableitem, before it was modified
  - KEYS_ONLY - Only the key attributes of the modified item



```
const myTable = new Table(this, "MyTable", {
  billingMode: BillingMode.PAY_PER_REQUEST,
  partitionKey: { name: "ID", type: AttributeType.STRING },
  removalPolicy: RemovalPolicy.DESTROY,
  tableName: "MyTable",
  stream: StreamViewType.
});

// Create source mapping
processDataFn.addEventSou
  new DynamoEventSource(reviewsTable, {
```

DynamoDB stream types

- The event object passed to the lambda function includes information on the type of table operation associated with the stream entry, i.e. insert, update, delete.
  - Use console.log (and JSON parse/stringify) to examine the structure of a stream event.