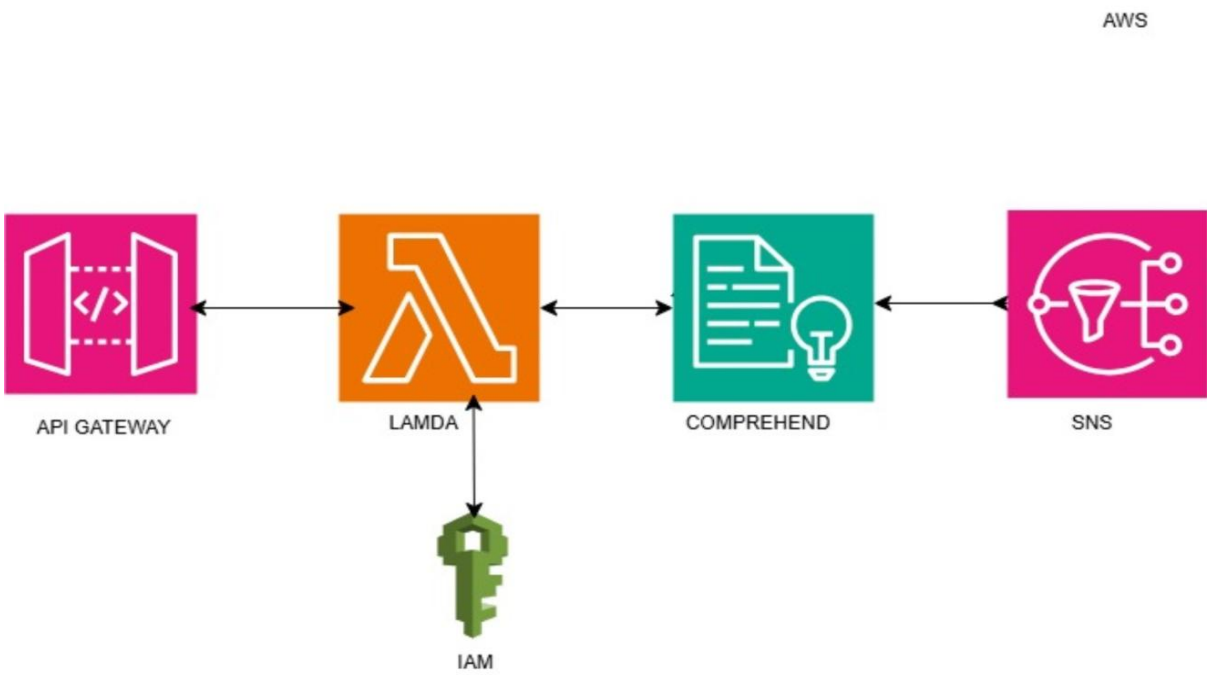
AWS serverless architecture that integrates API Gateway, Lambda, Comprehend, SNS, and IAM.



Project Idea

This project looks like a text sentiment analysis pipeline:

- A user sends text input via an API.
- AWS Comprehend analyzes the sentiment or meaning.
- The result is processed by Lambda.
- If needed, SNS (Simple Notification Service) sends alerts or notifications.
- IAM manages permissions for each service.

Workflow Recap

1. **Client → API Gateway** (send text).
2. **API Gateway → Lambda** (process request).
3. **Lambda → Comprehend** (analyze text sentiment).
4. **Lambda → SNS** (send results).
5. **SNS → Subscribers** (receive notification).
6. **IAM** controls access & security.

Prep

- Make sure you are in your chosen **Region** (top-right of the console).
- You'll need permissions to use **Lambda, API Gateway, SNS, IAM, Comprehend**.

STEP BY STEP GUIDE:

1) Create SNS Topic (email notifications)

1. **Amazon SNS → Topics → Create topic**

- **Type:** Standard
- **Create topic**

2. Open the topic you just created → copy the **Topic ARN**.

paste **the ARN** into Lambda later. (Use the **full ARN**, not just the name.)

3. **Subscribe your email**
   Topic page → **Create subscription**

- **Protocol:** Email

- **Endpoint:** your email (e.g., you@gmail.com) → **Create subscription**
- Go to your inbox and **Confirm subscription**.
  Status should become **Confirmed** (not "Pending confirmation").

2) Create the Lambda function

1. **Lambda → Create function**

- **Name:**
- **Runtime:** Python 3.11
- **Architecture:** x86_64
- **Execution role:** *Create a new role with basic Lambda permissions*
- **Create function**

2. **Environment variable** (this is where the SNS Topic ARN goes)
   Lambda → **Configuration → Environment variables → Edit → Add**

- **Key:** SNS_TOPIC_ARN
- **Value: your ARN** (ex:arn:aws:sns:us-east-1:<your-account-id>:FeedbackNotification)
  Save.

3. **Permissions for Comprehend & SNS**
   Lambda → **Configuration** → **Permissions** → click the **role**
   On the IAM role page:
4. **Add permissions** → **Attach policies**
    1. Add **AmazonComprehendReadOnly** (or **AmazonComprehendFullAccess** if you prefer)
    2. Add **AmazonSNSFullAccess** (or use the minimal inline policy below)

Least-privilege inline policy (recommended):
IAM Role → **Add permissions** → **Create inline policy** → **JSON** tab → paste and replace <your-account-id>:

```
{
  "Version": "2012-10-17",
  "Statement": [
    { "Effect": "Allow", "Action": "comprehend:DetectSentiment", "Resource": "*" },
    { "Effect": "Allow", "Action": "sns:Publish", "Resource": "arn:aws:sns:us-east-1:<your-account-id>:FeedbackNotification" }
  ]
}
```

Review policy → Name: <name>→ **Create policy**.

4. **Code** (handles both Lambda Test and API Gateway "proxy" events)

Lambda → **Code** tab → replace code → **Deploy**:

```
import os, json, base64, boto3

comprehend = boto3.client('comprehend')
sns = boto3.client('sns')
TOPIC_ARN = os.environ['SNS_TOPIC_ARN']

def _ok(payload):
    return {"statusCode": 200, "headers": {"Content-Type": "application/json"}, "body": json.dumps(payload)}

def _bad_request(msg):
```

```
    return {"statusCode": 400, "headers": {"Content-Type": "application/json"}, "body": json.dumps({"error": msg})}


def _parse_event(event):
    # Accept direct Lambda test: {"text": "..."} OR API Gateway proxy with string body
    if isinstance(event, dict) and "body" in event:
        body = event["body"]
        if event.get("isBase64Encoded"):
            body = base64.b64decode(body).decode("utf-8")
        try:
            return json.loads(body)
        except Exception:
            return {}
    return event if isinstance(event, dict) else {}


def lambda_handler(event, context):
    data = _parse_event(event)
    text = (data or {}).get("text")
    if not text:
        return _bad_request("text is required")

    # Sentiment via Comprehend
    r = comprehend.detect_sentiment(Text=text, LanguageCode='en')
    sentiment = r['Sentiment']

    # Notify via SNS
    sns.publish(
        TopicArn=TOPIC_ARN,                # MUST be full ARN
        Subject="Customer Feedback Result",
        Message=f"Text: {text}\nSentiment: {sentiment}"
    )
```

```
    return _ok({"feedback": text, "sentiment": sentiment})
```

3) Test the Lambda (no curl/Postman)

1. Lambda → **Test** → **Configure test event**

- **Event name:** <name>
- **JSON:**

```
{ "text": "I love this product!" }
```

Save → **Test**.

Expect Response body:

```
{"feedback":"I love this product!","sentiment":"POSITIVE"}
```

Check your email (subscribed to SNS). You should see **Customer Feedback Result**.

4) Create API Gateway (public endpoint)

1. **API Gateway** → **Create API** → **REST API (Build)**

- **API name:**<name>
- **Endpoint type:** Regional → **Create API**

2. **Resource**

- **Create Resource**
    - **Resource name:**<name>
    - Path becomes /<name>→ **Create**

3. **Method**

- Select /feedback → **Create Method** → choose **POST**
- **Integration type:** Lambda Function
- **Use Lambda Proxy integration: checked**
- **Lambda function:** <lambda function name>→ **Save** (allow permission)

4. (Optional) **Enable CORS** (useful for browsers later)

- Select /feedback → **Actions** → **Enable CORS** → **Enable CORS and replace existing CORS headers**

5. **Deploy**

- **Actions → Deploy API**
- **Stage:** create prod → **Deploy**
- Copy your **Invoke URL**, e.g. https://abc123.execute-api.us-east-1.amazonaws.com/prod/feedback

5) Test the API in the Console (no tools)

1. API Gateway → your API → **Resources** → select **POST** under /feedback → **Test**
2. **Request Body:**

```
{ "text": "this is really awesome" }
```

3. **Test** → Expect **200** and:

```
{"feedback":"this is really awesome","sentiment":"POSITIVE"}
```

4. You should also get the **SNS email**.

Drive Link:

https://drive.google.com/file/d/1PoSj5f2OQhGRSygKV0VpRctt6CkpWcVy/view