SNS and Lambda Setup

✅ 1. SNS Topic Creation

We created a new SNS topic to publish alerts:

```
aws sns create-topic --name cicd-alerts-topic
```

Get the ARN:

```
SNS_TOPIC_ARN="arn:aws:sns:us-east-1:ACCOUNT_ID:cicd-alerts-topic"
```

✅ 2. Subscribe Email to SNS Topic

We added an email subscription to receive alerts:

```
aws sns subscribe \
  --topic-arn $SNS_TOPIC_ARN \
  --protocol email \
  --notification-endpoint your_email@example.com
```

👉 **Note:** You must **confirm the subscription** from your email.

✅ 3. Create the Lambda Function

We created a Lambda function named (for example) `cicd-status-notifier` that parses the event and publishes a cleaned alert to SNS.

🔁 Lambda Code (Python 3.9):

```
import json
import boto3
```

```
import os

sns = boto3.client('sns')
SNS_TOPIC_ARN = os.environ.get("SNS_TOPIC_ARN")

def lambda_handler(event, context):
    print("Received event:", json.dumps(event))
    print("SNS_TOPIC_ARN:", SNS_TOPIC_ARN)

    source = event.get("source")
    detail = event.get("detail", {})
    detail_type = event.get("detail-type", "Unknown Event")

    message = ""
    subject = "CI/CD Alert"

    if source == "aws.codebuild":
        project = detail.get("project-name", "Unknown Project")
        status = detail.get("build-status", "UNKNOWN")
        phase = detail.get("additional-information", {}).get("phases", [])

        failed_phase = "N/A"
        for p in phase:
            if p.get("phase-status") == "FAILED":
                failed_phase = p.get("phase-type", "Unknown")
                break

        message = f"🔨 Build Alert\nProject: {project}\nStatus: {status}\nFailed P
hase: {failed_phase}"
        subject = f"[Build {status}] {project}"

    elif source == "aws.codedeploy":
        deployment_id = detail.get("deploymentId", "Unknown Deployment")
        status = detail.get("status", "UNKNOWN")
        app = detail.get("application", "Unknown App")
        deployment_group = detail.get("deploymentGroupName", "Unknown Gro
```

```
up")

        message = f"🚀 Deploy Alert\nApplication: {app}\nDeployment Group: {d
eployment_group}\nDeployment ID: {deployment_id}\nStatus: {status}"
        subject = f"[Deploy {status}] {app}"

    else:
        message = json.dumps(event, indent=2)
        subject = "Unknown Event Received"

    sns.publish(
        TopicArn=SNS_TOPIC_ARN,
        Message=message,
        Subject=subject
    )

    return {
        'statusCode': 200,
        'body': json.dumps('Alert sent successfully!')
    }
```

In lambda → Permissions → Execution Role add:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-1:975050195505:cicd-alerts-topic"
        },
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:GetDeployment",
```

```
            "codedeploy:ListDeployments"
          ],
          "Resource": "*"
        },
        {
          "Effect": "Allow",
          "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents"
          ],
          "Resource": "*"
        }
    ]
}
```

✅ 4. Attach SNS_TOPIC_ARN as Environment Variable

In Lambda configuration:

```
Key: SNS_TOPIC_ARN
Value: arn:aws:sns:us-east-1:ACCOUNT_ID:cicd-alerts-topic
```

✅ 5. EventBridge Rule for CodeBuild

We created a rule to catch **build state changes**:

```
aws events put-rule \
  --name CodeBuildStatusRule \
  --event-pattern '{
    "source": ["aws.codebuild"],
    "detail-type": ["CodeBuild Build State Change"]
  }'
```

Then connected Lambda:

```
aws events put-targets \
  --rule CodeBuildStatusRule \
  --targets "Id"="cb-target","Arn"="arn:aws:lambda:us-east-1:ACCOUNT_ID:function:cicd-status-notifier"
```

Granted permission:

```
aws lambda add-permission \
  --function-name cicd-status-notifier \
  --statement-id AllowCodeBuildInvoke \
  --action lambda:InvokeFunction \
  --principal events.amazonaws.com \
  --source-arn arn:aws:events:us-east-1:ACCOUNT_ID:rule/CodeBuildStatusRule
```

✅ 6. EventBridge Rule for CodeDeploy

```
aws events put-rule \
  --name CodeDeployStatusRule \
  --event-pattern '{
    "source": ["aws.codedeploy"],
    "detail-type": ["CodeDeploy Deployment State-change Notification"]
  }'
```

Connect the Lambda:

```
aws events put-targets \
  --rule CodeDeployStatusRule \
  --targets "Id"="cd-target","Arn"="arn:aws:lambda:us-east-1:ACCOUNT_ID:function:cicd-status-notifier"
```

Permission:

```
aws lambda add-permission \
  --function-name cicd-status-notifier \
  --statement-id AllowCodeDeployInvoke \
  --action lambda:InvokeFunction \
  --principal events.amazonaws.com \
  --source-arn arn:aws:events:us-east-1:ACCOUNT_ID:rule/CodeDeployStatus
Rule
```

Give sample test cases in Lambda’s Test tab

Final Result

- 📩 Whenever CodeBuild or CodeDeploy runs, this Lambda gets invoked.
- 📧 Lambda formats the message and pushes it to SNS.
- 💌 SNS sends an email alert to the devs/team instantly.
