National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Anuja Mahendrasingh Solanki
Student ID: x22139524

School of Computing
National College of Ireland

Supervisor: Prof. Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anuja Mahendrasingh Solanki
Student ID:	x22139524
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	25th April 2024
Project Title:	Configuration Manual
Word Count:	979
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Anuja Mahendrasingh Solanki
Date:	25th April 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Anuja Mahendrasingh Solanki
x22139524

1 IAM Role Setup

Log in to your AWS Account and perform the below:
Search for the IAM management console. Set up a separate IAM role with only the rights needed for a certain service or task.

- Click on 'Roles' in the sidebar as shown below and click on the 'Create Role' button to create a new IAM role named **"Drift-role-27s3qi0o"**. Select 'Custom role policy' for the role type and continue. Amazon Web Services (2024b)
- Policies are JSON documents in AWS that let you specify who has access to AWS resources and what actions they can perform on those resources.

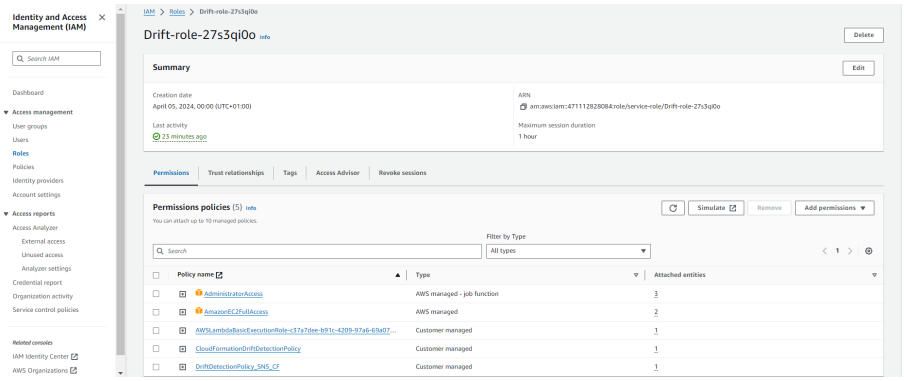

Figure 1: IAM Roles

- In the policy section, attach the already-available AWS-managed policies:
  1. AWS AdministorAccess
  2. AmazonEC2FullAccess
  3. AWS LambdaBasicExecutionrole
- As the account is logged in as the root user, any resource can be created in the cloudformation, and while detecting the drift, the Lambda function requires permission to use a variety of resources. Provisioning a resource without the permissions granted to Lambda will prevent the Lambda function from detecting the drift. In this case, AdministratorAccess is helpful. Grant the administrator restricted access to see if the cloudformation is unable to access permission for any resource; it can be identified. And then we can give access to that particular resource.

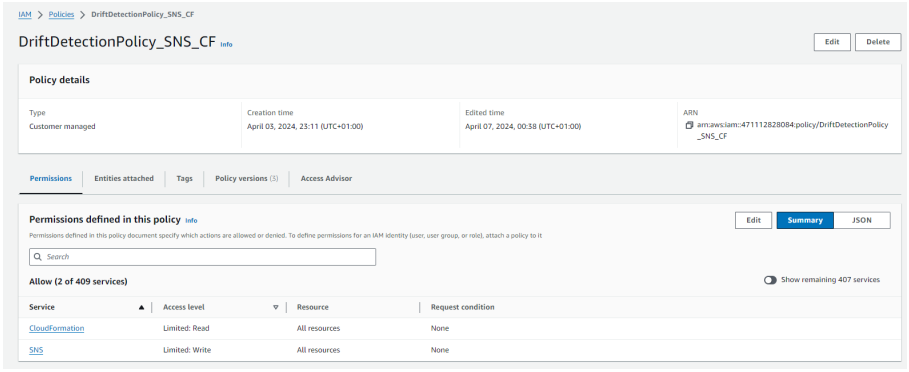

Figure 2: DriftDetectionPolicy_SNS_CF

- Addition to these AWS-managed policies, attach the customer-based policies:
  1. CloudFormationDriftDetectionPolicy
  2. DriftDetectionPolicy_SNS_CF

- Cloudformation and SNS are the two services attached to these policies, defining their permissions. When modifications are made to an IAM customer-managed policy, the new policy takes first place over the old one. IAM makes a new version for the controlled policy instead.

- DescribeStackDriftDetectionStatus, DetectStackDrift, and DetectStackResourceDrift and publish are defined in a version below:

```
"Statement": [
    {
        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": [
            "cloudformation:DetectStackResourceDrift",
            "cloudformation:DescribeStackDriftDetectionStatus",
            "cloudformation:DetectStackDrift",
            "sns:Publish"
        ],
        "Resource": "*"
    }
]
```

- The IAM role, along with these policies, will be granted the necessary authorizations for accessing AWS services such as EC2, Lambda, and CloudFormation to help with remediation and drift detection workflows.

2 Creation of Stacks using Cloudformation

Now Go to the AWS Console and search for "Cloudformation". For provisioning and setting up resources.

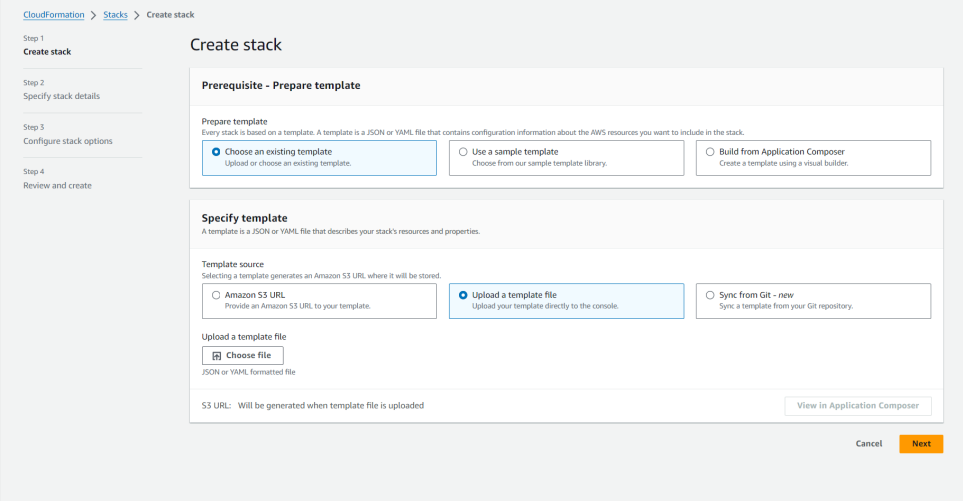

Figure 3: Create Stacks

- Click on "Create Stack" and you will be drawn to the below page:

- Select the requirements as shown in the Figure 3. Click on Upload a template file to upload the ".YAML" file from your local system that will be used to define the stack.
- Provide a stack name as "EC2instance" and click next.
- Review the stack details and click on Submit. Now Ec2Instance Stack is created as shown in Figure 4
- Similarly, upload the '.YAML' files for S3Bucket, and DynamoDB stacks and create them as shown in Figure 5
- Note the Stack ID(ARNs) of all three stacks. In this case for EC2 instance it is **arn:aws:cloudformation:ap-south-1:471112828084:stack/EC2instance/74c0e490-f209-11ee-a0af-0aac48631e59**

3 Configure SNS

Create an SNS Topic named "Detect_Drift" to send notifications whenever the drift is detected in the stacks. Amazon Web Services (2024a)

- While creating the subscription, select the protocol as EMAIL and provide an email ID as the endpoint (where the notifications will appear); here it is given as "x22139524@student.ncirl.ie"
- Copy the ARN Id: **arn:aws:sns:ap-south-1:471112828084:Detect_Drift**

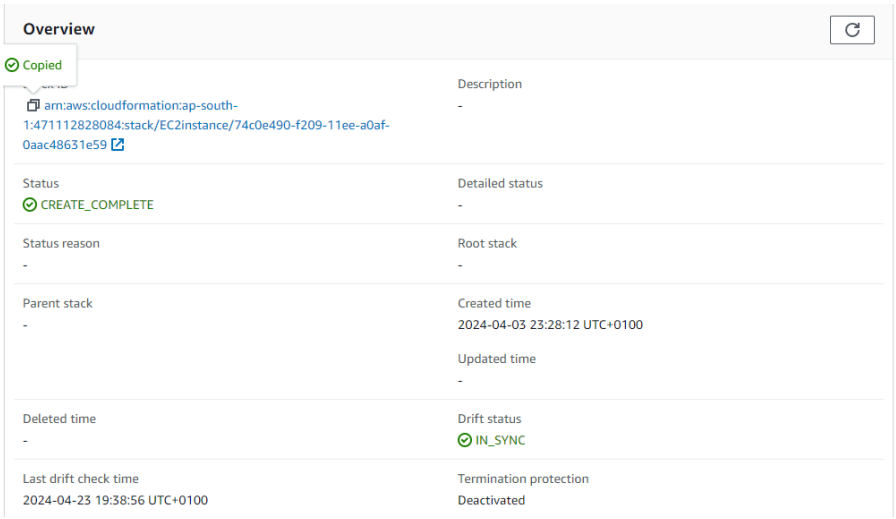

Figure 4: EC2 Stack Details

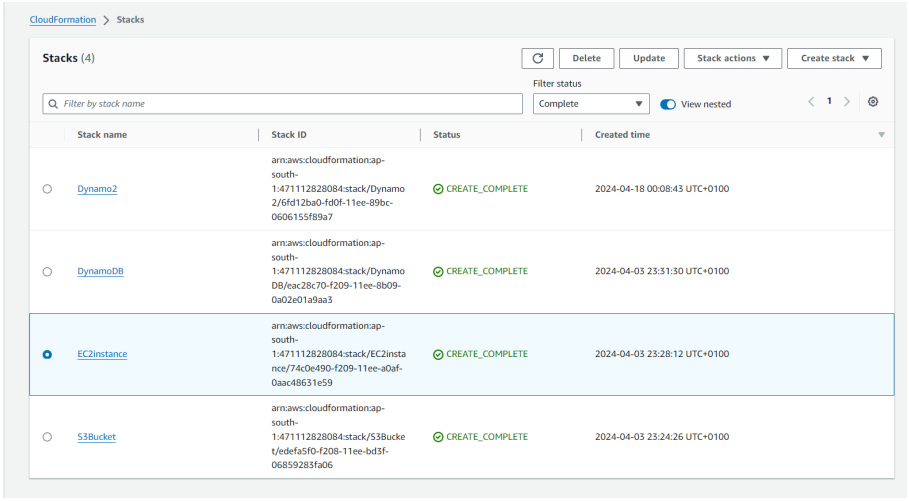

Figure 5: Created Stacks

4 Creation of Python script in Lambda

To create a Python function to write a program for Automating Drift Detection in the stacks, follow the steps below:

- In the Lambda Console, create a Lambda function with the name **Drift"**. Use the Runtime as Python. and click on the create function.
- Once the function is created, write your Python code for detecting drift in the three stacks mentioned in Section 2.
- Make sure you have the below installed before running your Python script for detecting the drift.
  1. Python
  2. Boto3 library

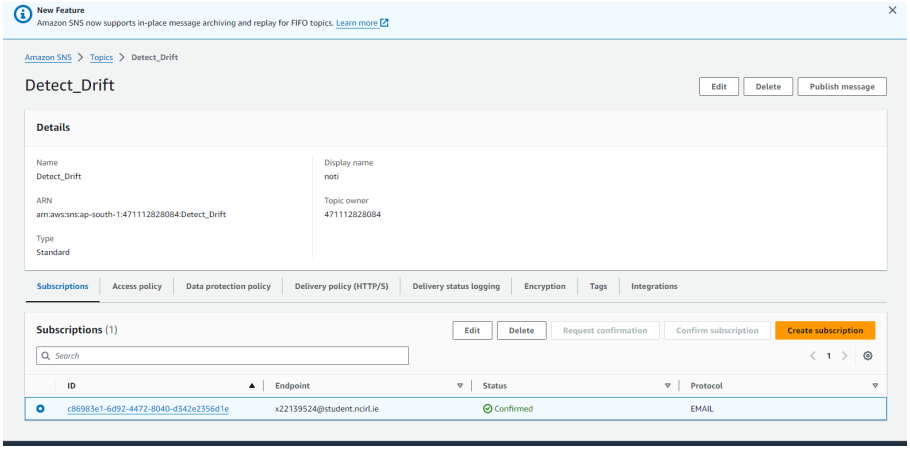

Figure 6: Created Subscription

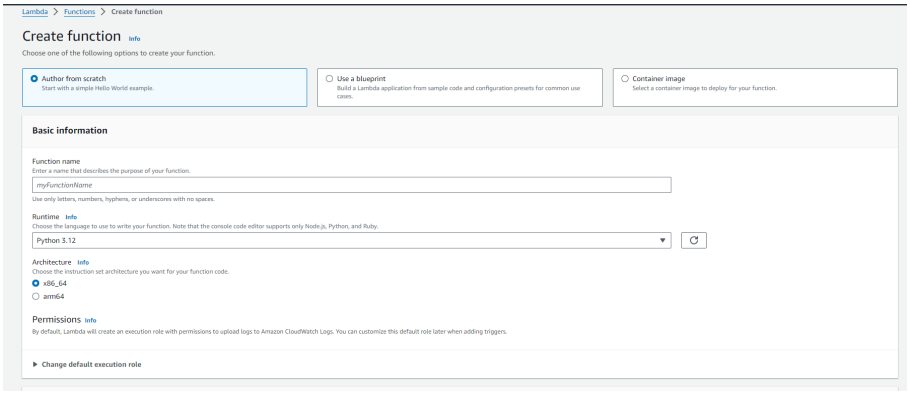

Figure 7: Creation of Lambda Function

- If boto3 is not installed, use this command to install it. **"pip install boto3"**. Amazon Web Services (2023a)
- Copy and paste the ARN IDs for the three stacks and the SNS subscription from sections 2 and 3 in the code to identify the resources. as shown in Figure 8

```
# Specify your stack names and SNS topic ARN
stack_names = ['arn:aws:cloudformation:ap-south-1:471112828084:stack/DynamoDB/eac28c70-f209-11ee-8b09-0a02e01a9aa3', '
#stack_names = ['arn:aws:cloudformation:ap-south-1:471112828084:stack/EC2instance/74c0e490-f209-11ee-a0af-0aac48631e59
# Add your SNS ARN names here
sns_topic_arn = 'arn:aws:sns:ap-south-1:471112828084:Detect_Drift'  # Change to your SNS topic ARN
# response = cf_client.detect_stack_drift(StackName="arn:aws:cloudformation:ap-south-1:471112828084:stack/EC2instance/
```

Figure 8: Stack ARN and SNS topic ARN

5 EventBridge Schedule for Triggering Lambda Function

- Create an Eventbridge schedular named "**Driftrule**" to trigger the Lambda function "**Drift**" and run the drift detection Python program with the below specifications.
- A cron expression generates a time-sensitive recurring schedule with precise configuration. Set the corn expression for the schedule as **0 0,8,16 * ? *** Figure 9. Amazon Web Services (2023b) It means that the lambda function will trigger every 8 hours to check the drift of the resources.

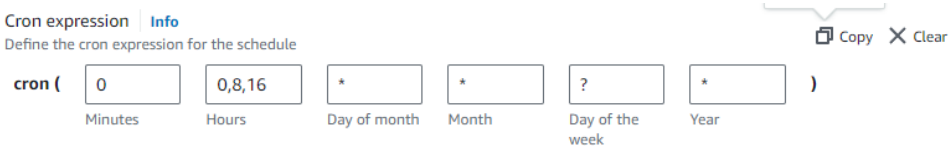

Figure 9: Cron Expression

- Click on Next and search for AWS Lambda from the list of APIs provided. Select the target API as an AWS Lambda from the given options. Now scroll down and write the payload specifications with the below script. Figure 10
- The JSON Script:

```
{
  "comment": "Scheduled drift detection trigger",
  "stackName": "DynamoDB, EC2instance, S3Bucket"
}
```

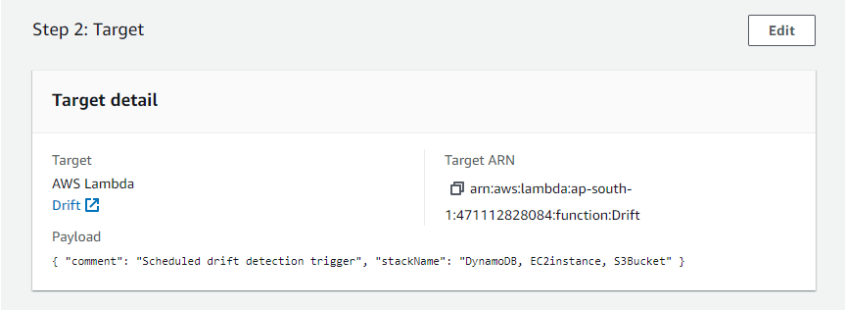

Figure 10: Set Target

- Click on Next and select the permission: Select as an existing role and choose "**Amazon_EventBridge_Scheduler_LAMBDA_84821d0ca4**" Figure 11

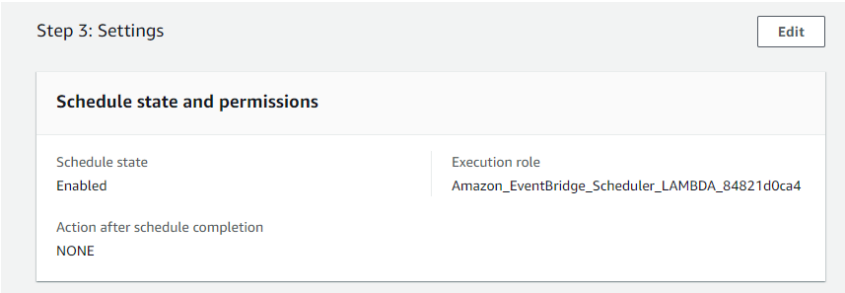

Figure 11: Setting up the permissions

6 Performing the Drift Detection

Once the above configurations are performed, drift detection will start as per the schedule defined in the AWS Eventbridge. It will then trigger the AWS Lambda template, where the drift detection script will start running.

- The first step is initializing Cloudformation and SNS in the Python script.

```
def lambda_handler(event, context):
    # Initialize the CloudFormation and SNS clients
    cf_client = boto3.client('cloudformation')  # Initialize CloudFormation client
    sns_client = boto3.client('sns')  # Initialize SNS client
```

Figure 12: Initilization of Cloudformation and SNS

- Provide the stack ARN and SNS ARN for locating the resources. Check Figure 8
- After the stacks are defined, the program will trigger the drift detection for all the stacks, one after the other for drift.

```
for stack_name in stack_names:
    # Step 1: Trigger drift detection
    logger.info(f"Triggering drift detection for stack: {stack_name}")
    response = cf_client.detect_stack_drift(StackName=stack_name)  # Trigger drift detection for the stack
    drift_detection_id = response['StackDriftDetectionId']  # Get drift detection ID
    time.sleep(30)
    # Step 2: Wait for drift detection to complete
    wait_interval = 5  # Initial wait interval
    max_wait_time = 300  # Maximum wait time (5 minutes)
    while True:
```

Figure 13: Triggering Drift Detection

- Once the drift detection is completed, it extracts the details from the stack and sends the status to the "endpoint".
- The below response will be received after a successful run of the Python script. Figure 15

```python
# Step 5: Prepare and send the notification
s = stack_name.split(':')[-1].split('/')
stack_name = s[-2]
message = f"Drift status for stack {stack_name} is {drift_status}. Details:\n{json.dumps(drift_details, indent=2)}"
sns_response = sns_client.publish(
    TopicArn=sns_topic_arn,
    Message=message,
    Subject=f'CloudFormation Stack Drift Status: {stack_name}'
)
```

Figure 14: Drift Detection SNS Notification

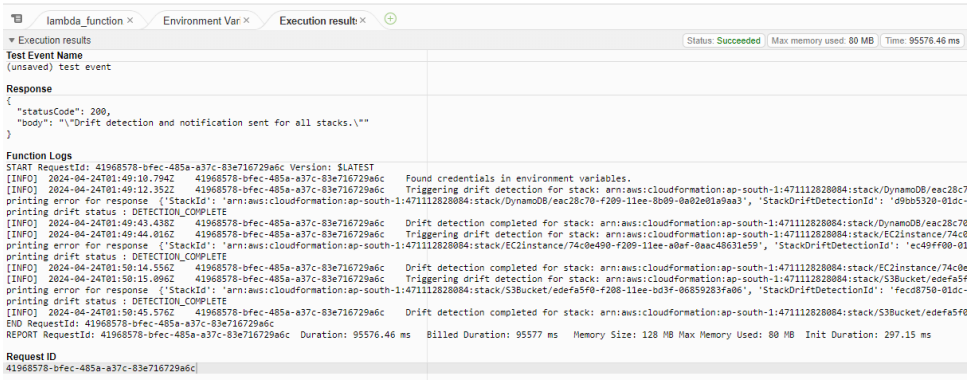

Figure 15: Response after successful run

References

Amazon Web Services (2023a). *Boto3 Documentation: AWS CloudFormation.*

Amazon Web Services (2023b). Schedule types in aws scheduler.

Amazon Web Services (2024a). Amazon sns developer guide.

Amazon Web Services (2024b). Iam roles - aws identity and access management.