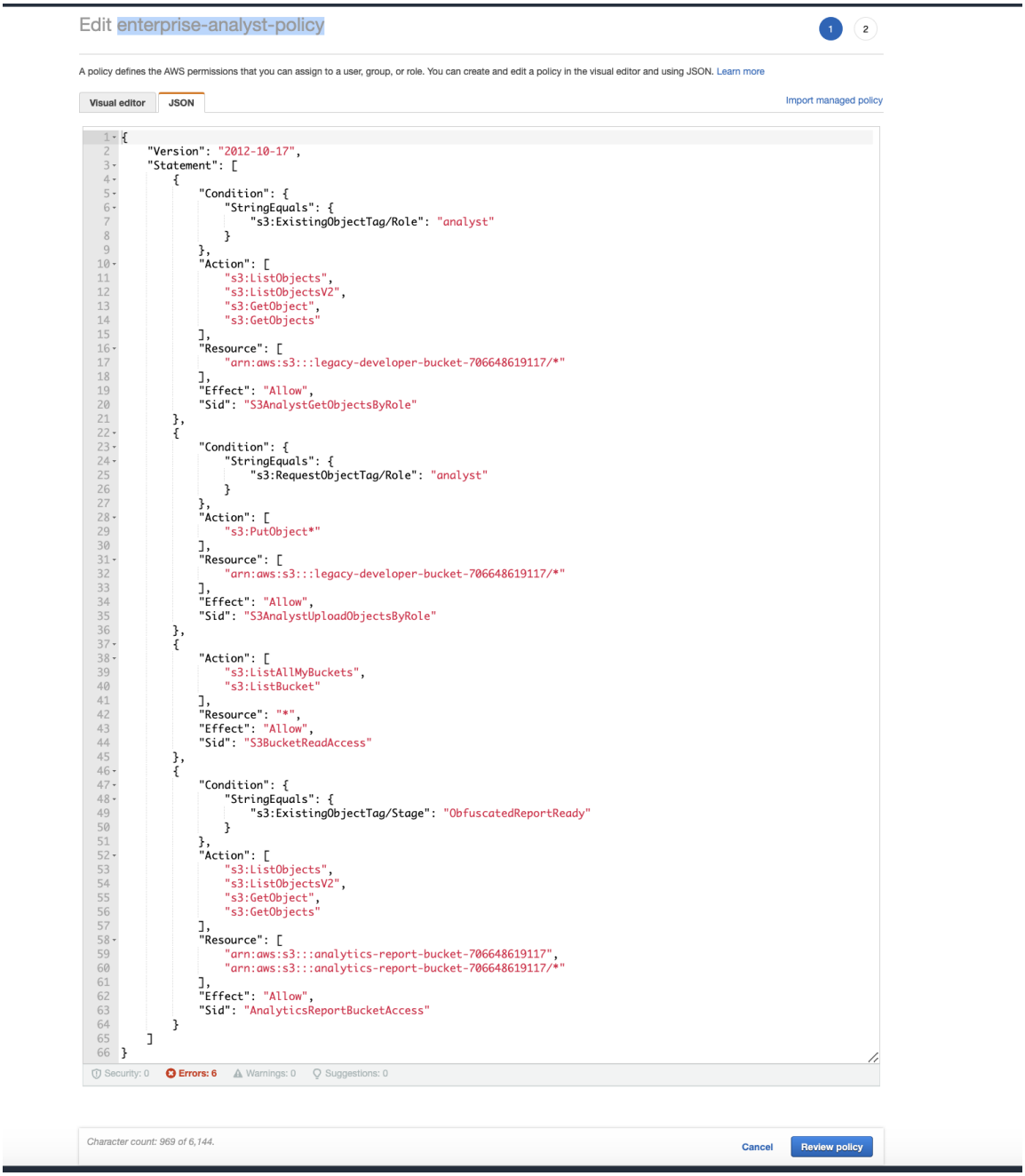
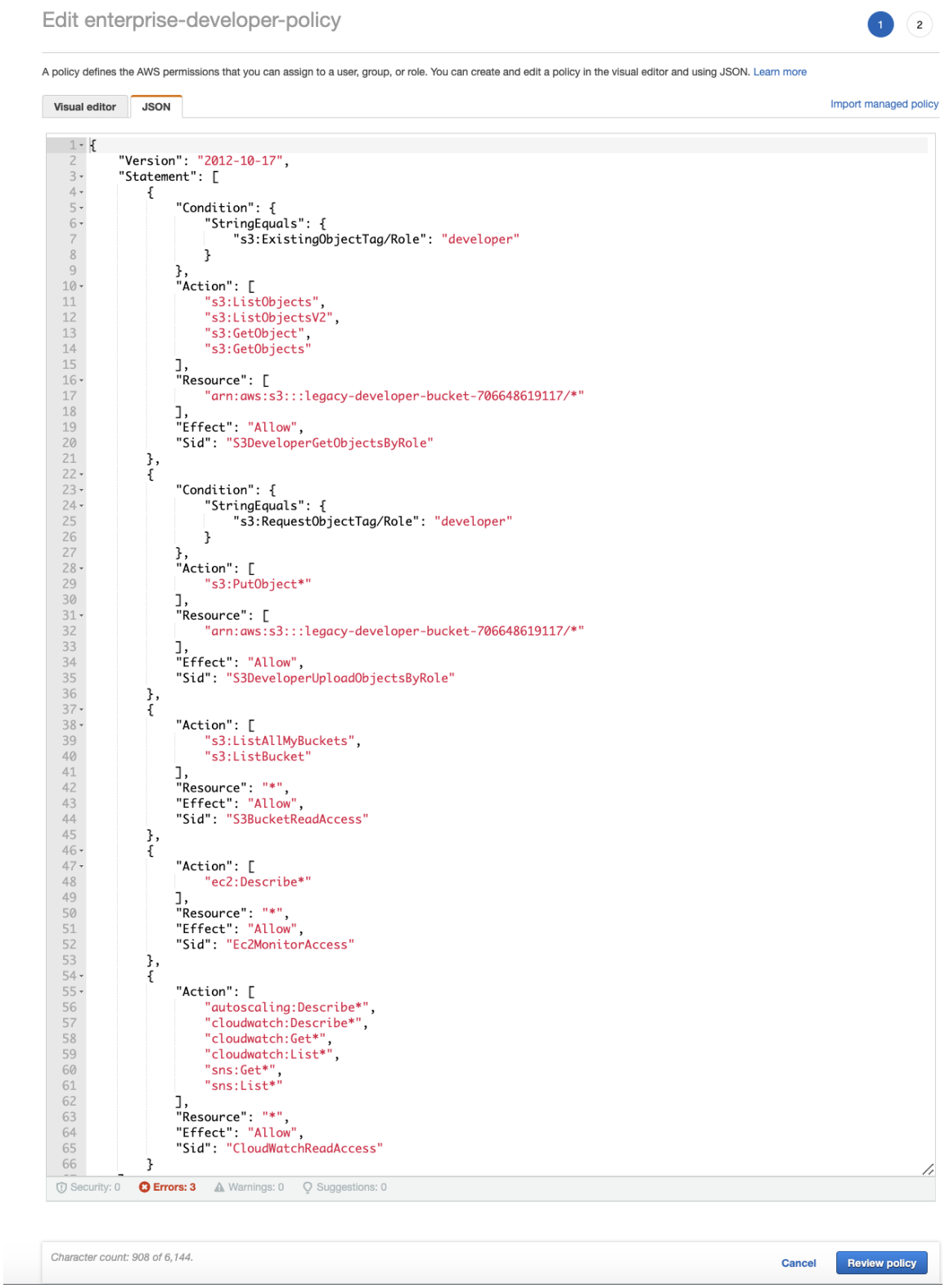
Purpose of this Folder

This folder should contain a fully working project. This will be added to the reviewer toolkit for reviewers to use.

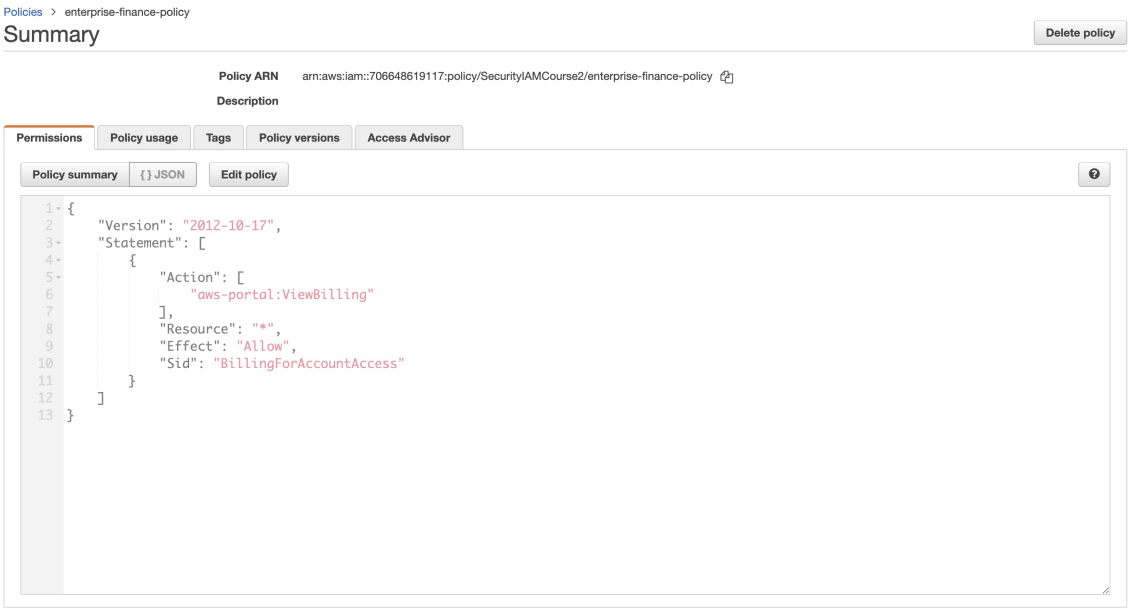
- Step 1 & 2 >
  - Submission 1:
    - enterprise-analyst-policy:



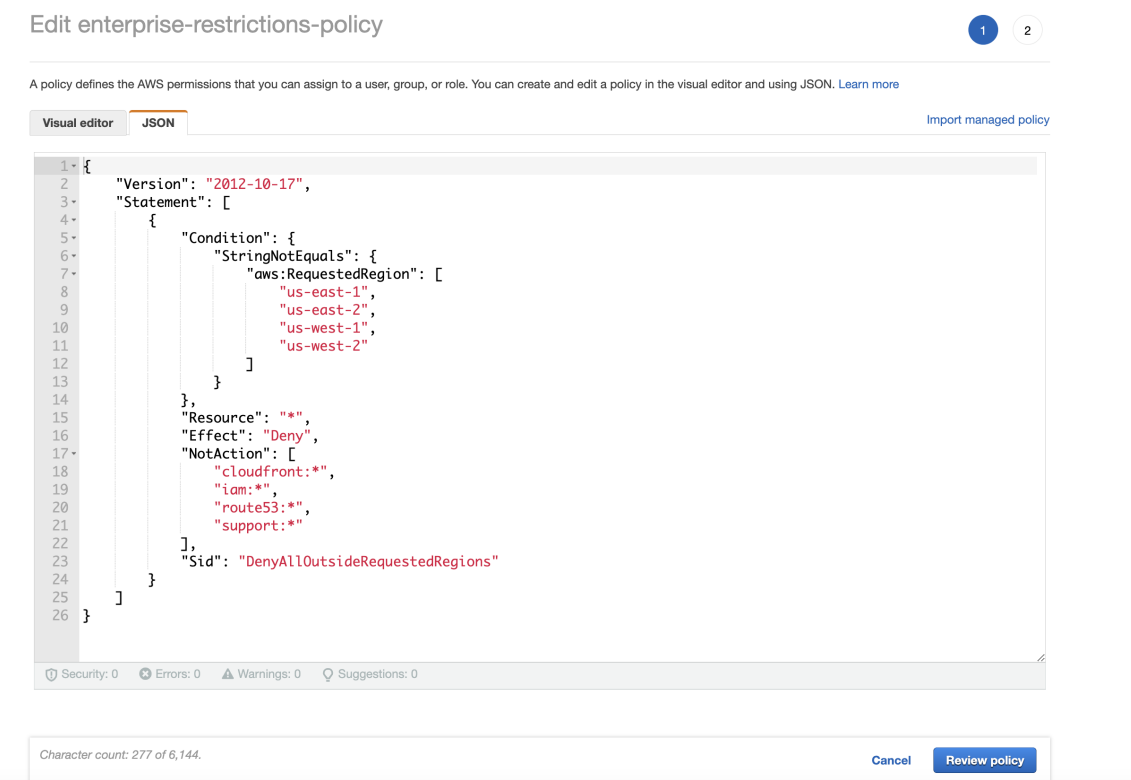
- enterprise-developer-policy:



- enterprise-finance-policy:

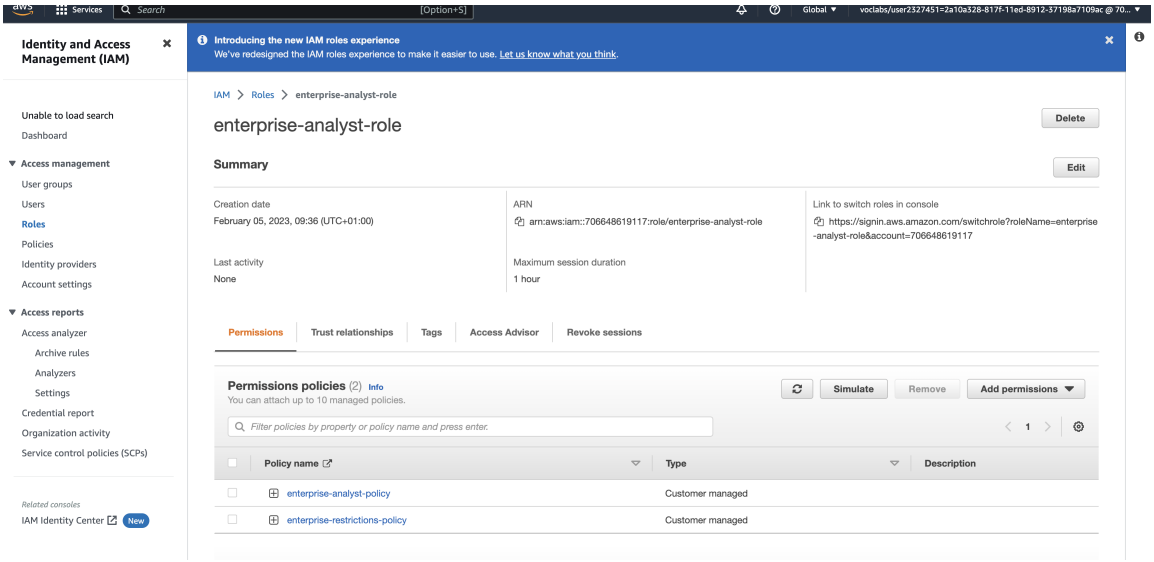


- enterprise-restrictions-policy:

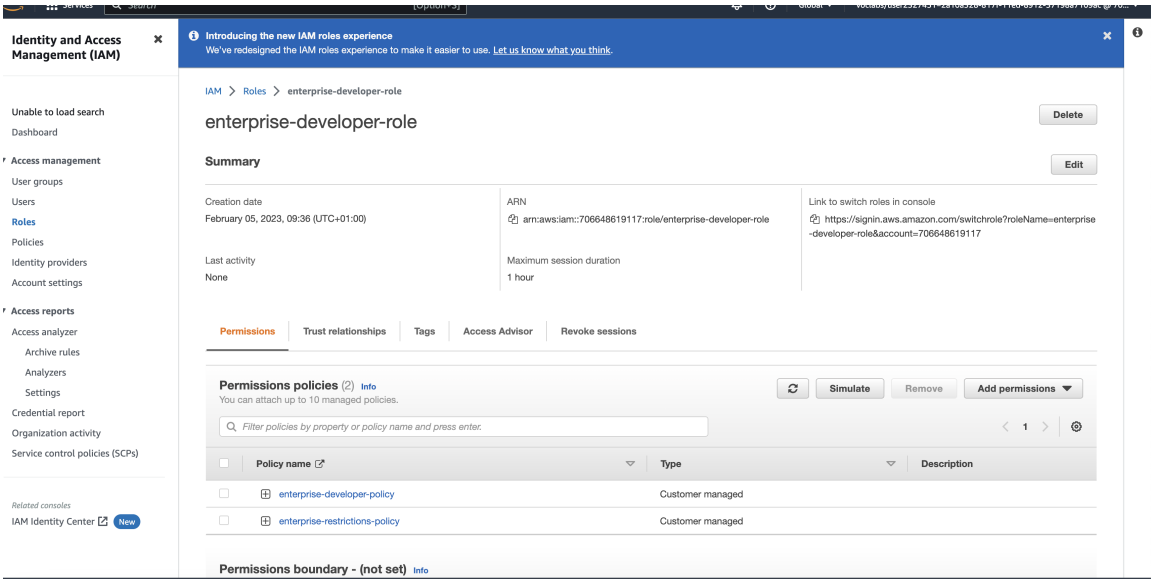


- Submission 2:

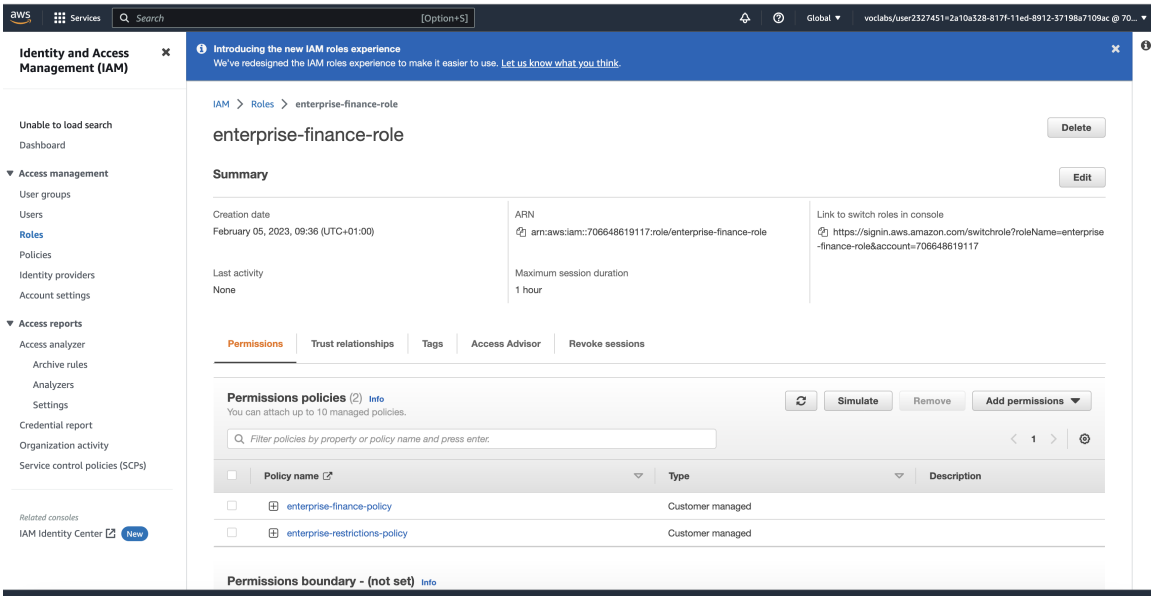
- enterprise-analyst-role:



- enterprise-developer-role:

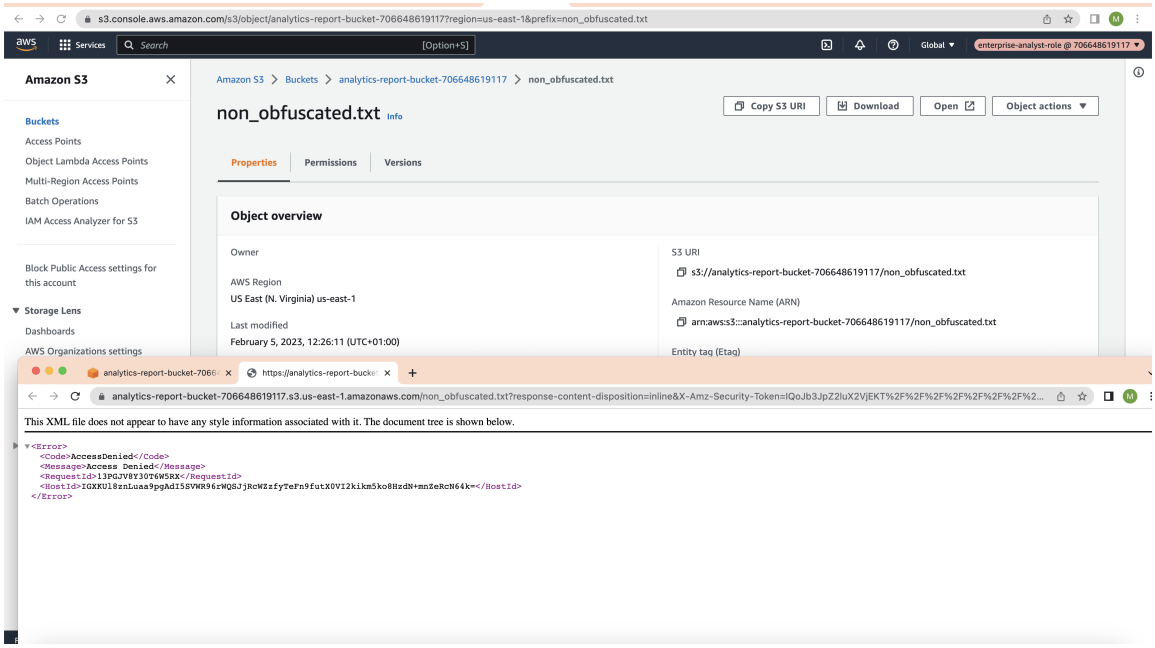


- enterprise-finance-role:

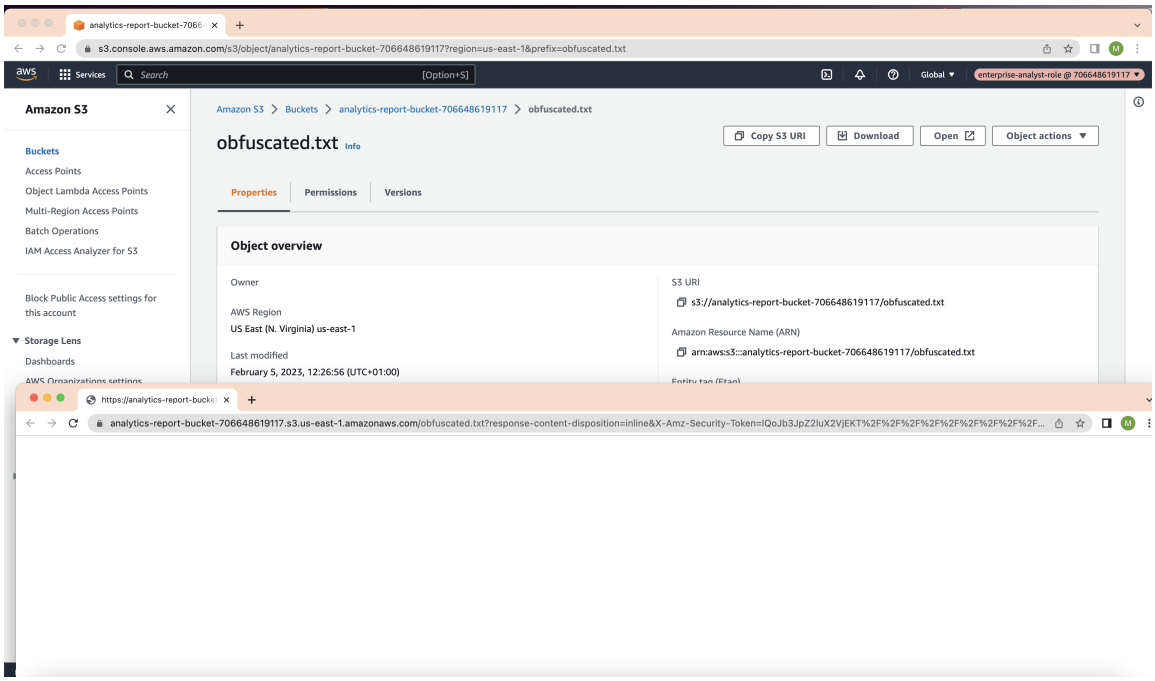


- Step 3 & 4 >
  - Submission 3:

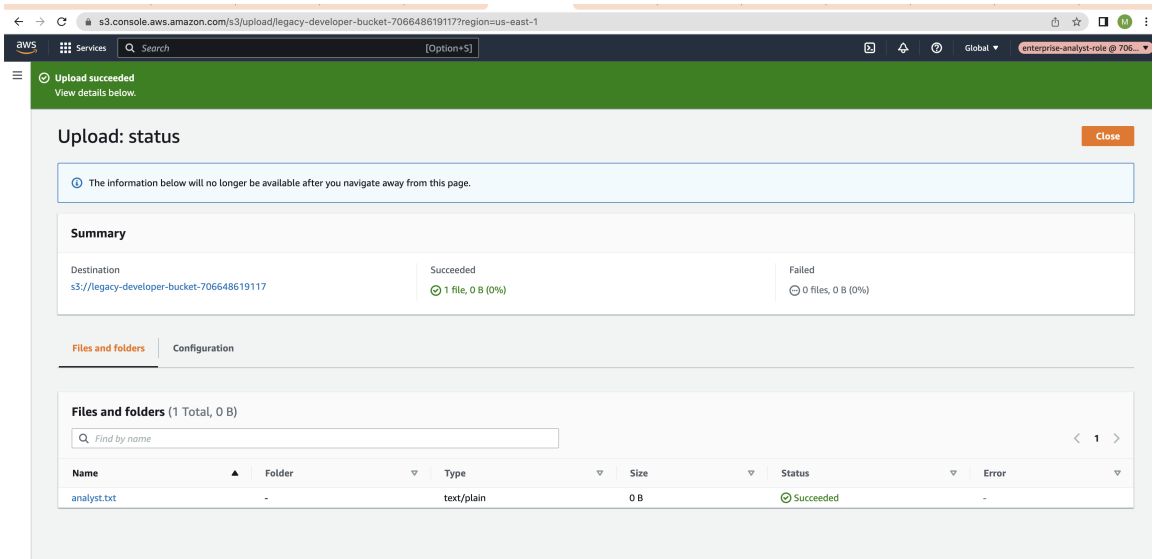
- non_obfuscated.txt:



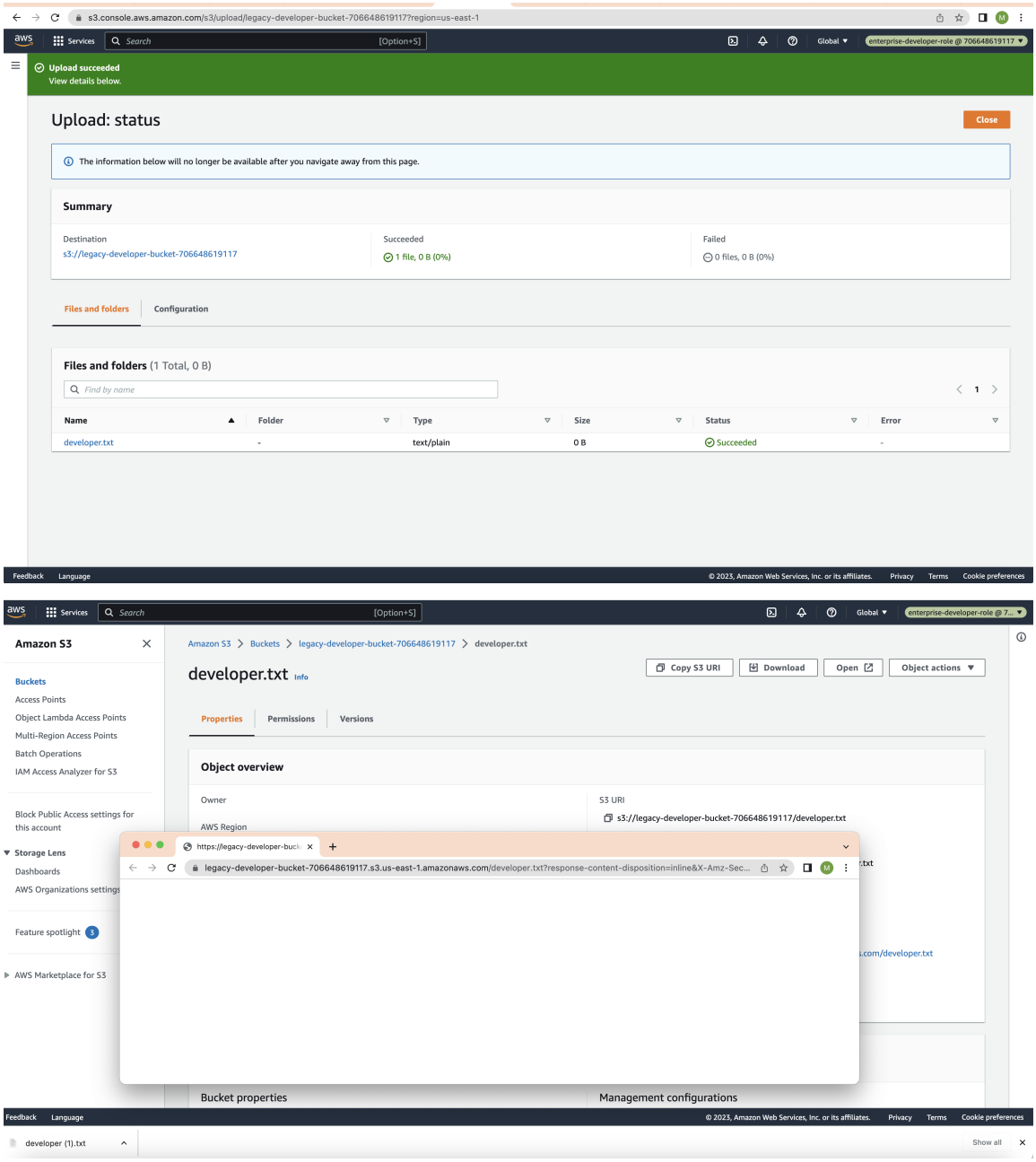
- obfuscated.txt:



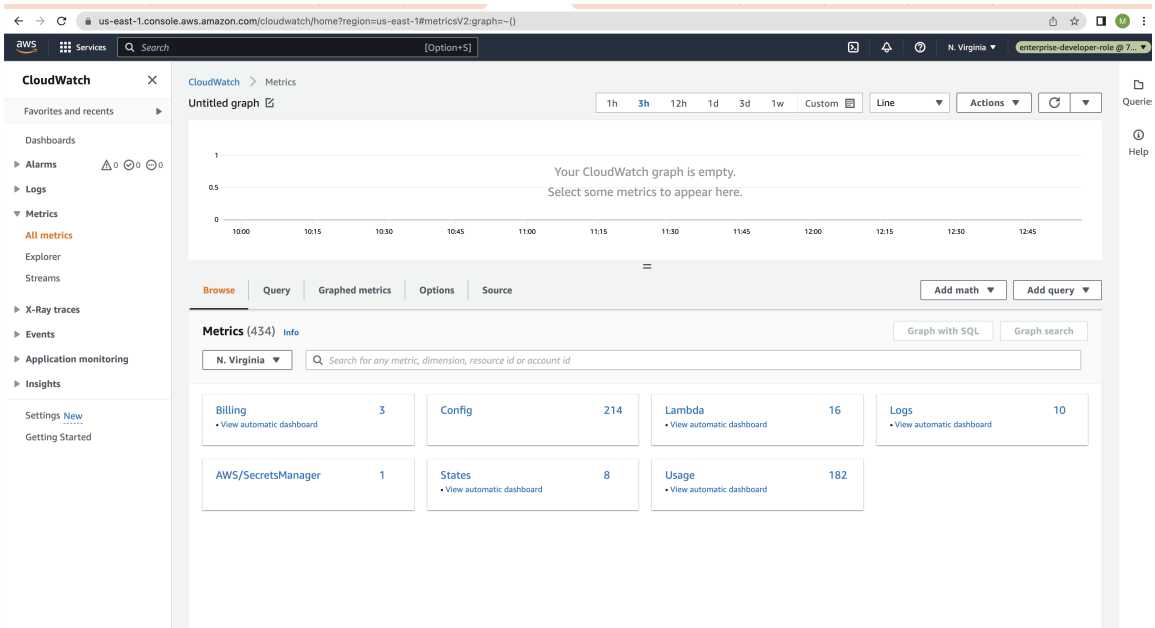
- analyst.txt:



- developer.txt:

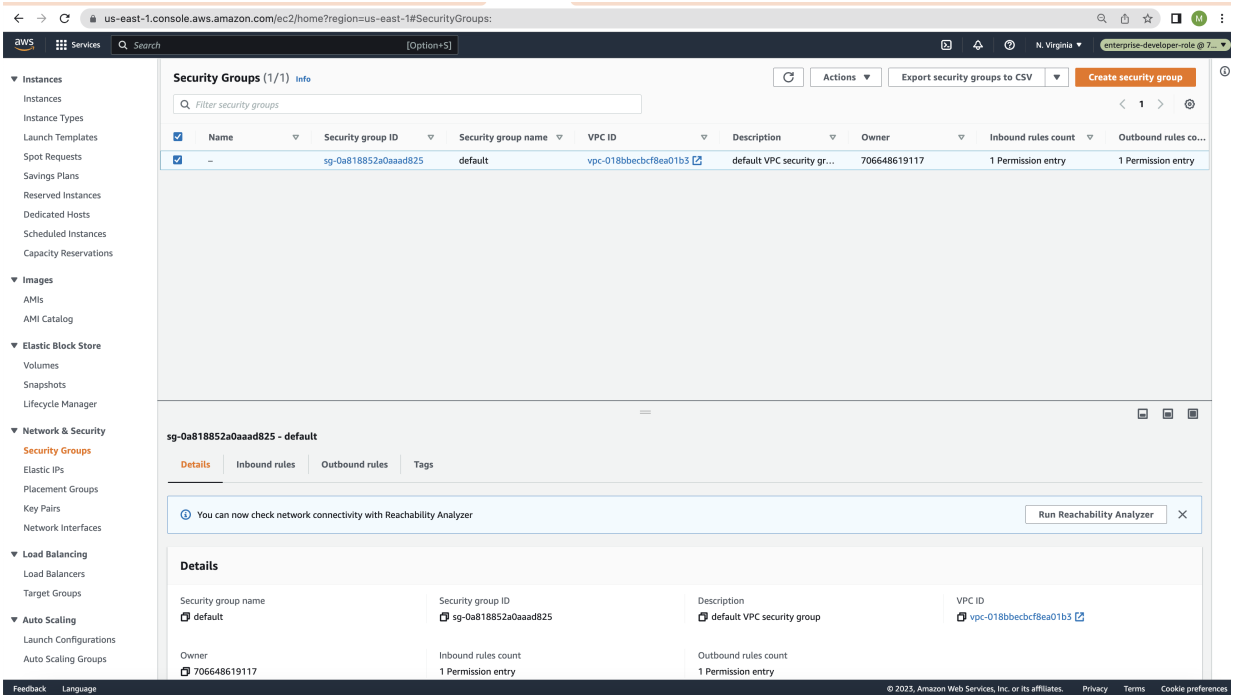


- cloudWatch-metric

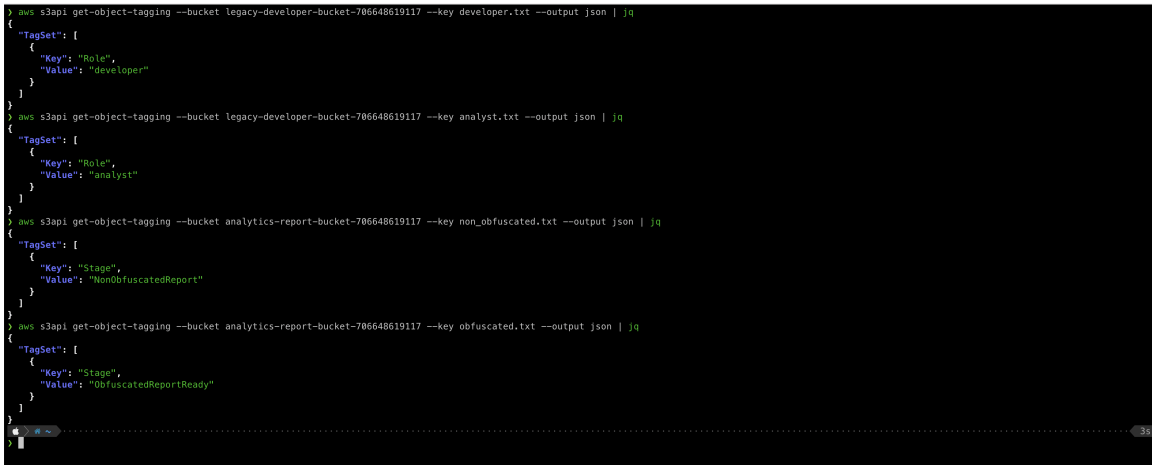


- Security Groups if this required then in the question we need to crrect it in task 4 the assignment

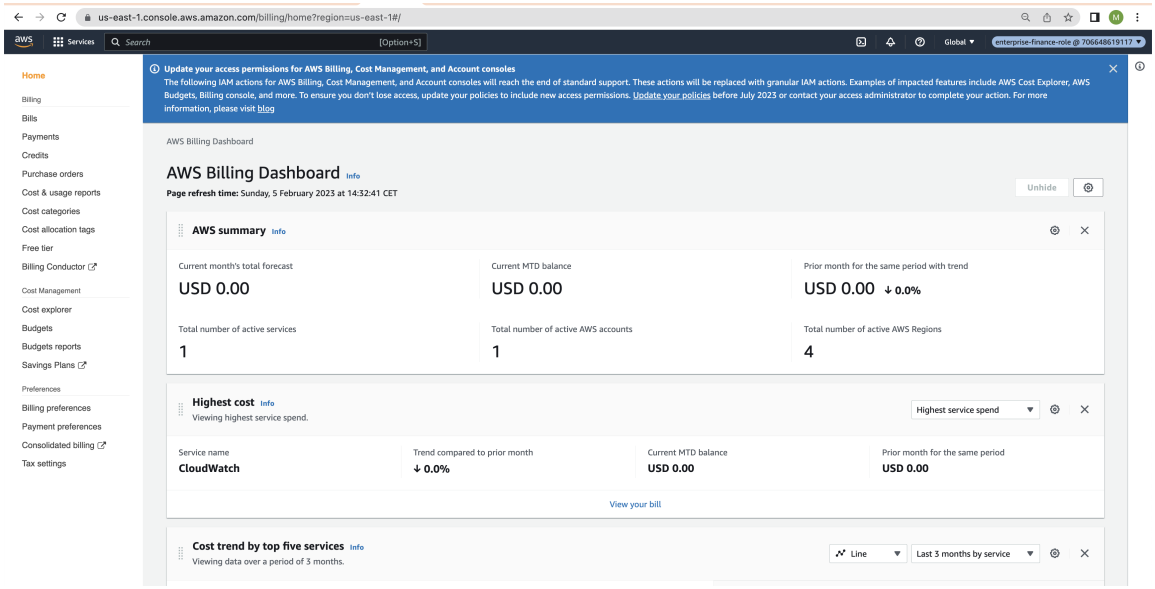
After validating the CloudWatch access granted to the role, navigate to the EC2 service and navigate to Security Groups. Within the submission template, under Steps 3 & 4 > Submission 3 provide a screenshot of successfully viewing a ~~CloudWatch metric~~ **Security Groups**.



- s3 objects tags



- Billing Console

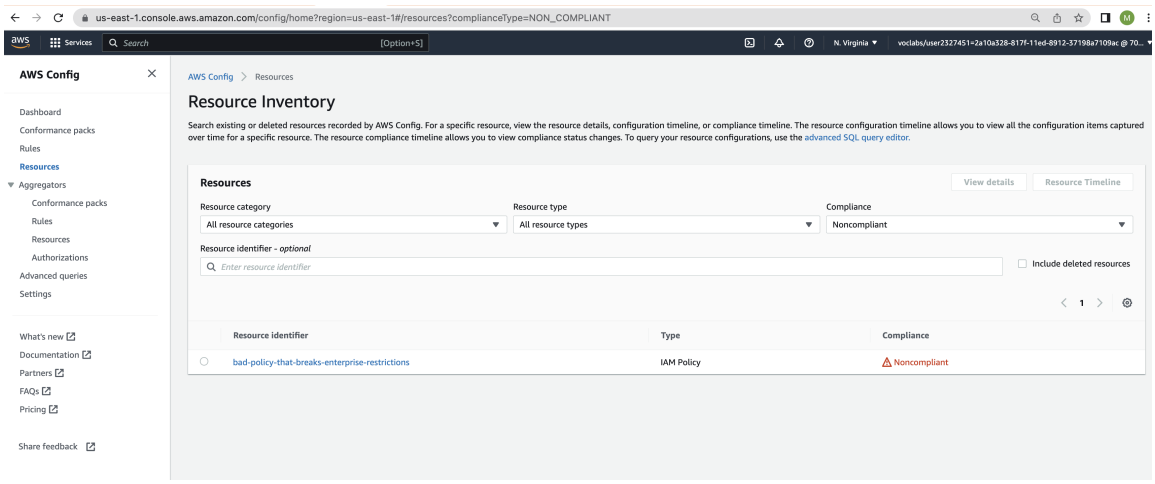


- Steps 5 & 6 >
  - Submission 4
    - Lambda Code

```
import os
import json
import boto3
import datetime
from urllib.parse import unquote

RESTRICTED_RESOURCES = ["arn:aws:s3:::super-secret-bucket"]
ALERTING_ENABLED = False

def handler(event, context):
    print(event)
    invoking_event = json.loads(event["invokingEvent"])
    resource_type = invoking_event["configurationItem"]
["resourceType"]
    resource_id = invoking_event["configurationItem"]
["resourceId"]
    resource_arn = invoking_event["configurationItem"]["ARN"]
    restricted_resources_enabled = []

    if resource_type == "AWS::IAM::Policy":
        configuration = invoking_event.get("configurationItem",
{}).get("configuration")
        if configuration:
          policy_version_list =
configuration.get("policyVersionList", [])
          default_version = False
          for version in policy_version_list:
              if version.get("isDefaultVersion", False):
                  default_version = version
          if default_version:
            policy =
```

```
json.loads(unquote(default_version["document"]))
            if type(policy["Statement"]) is list:
                for statement in policy["Statement"]:
                    if any(resource in RESTRICTED_RESOURCES for
resource in statement.get("Resource", [])):
                        client = boto3.client("config")
                        client.put_evaluations(
                            Evaluations=[
                                {
                                    'ComplianceResourceType':
resource_type,
                                    'ComplianceResourceId':
resource_id,
                                    'ComplianceType':
'NON_COMPLIANT',
                                    'Annotation': f'The policy has
a restricted resource listed',
                                    'OrderingTimestamp':
datetime.datetime.now()
                                },
                            ],
                            ResultToken=event["resultToken"]
                        )

restricted_resources_enabled.append(resource_arn)
    if ALERTING_ENABLED and restricted_resources_enabled:
      print("ALERTING SNS")
      sns_client = boto3.client("sns")
      sns_client.publish(
        TopicArn=os.environ["SNS_TOPIC"],
        Message=f'The following policies have restricted resources
enabled {restricted_resources_enabled}'
      )
```

- Steps 5 & 6 >
  - Submission 5
    - bad-policy-that-breaks-enterprise-restrictions:



- Steps 7 >
  - Submission 6

- organizational_role_diagram: