SOAR Step by Step Deployment Guide

Table of Contents

SOAR CAPSTONE: STEP-BY-STEP DEPLOYMENT VIA CLI

1. FOLDER STRUCTURE

After extracting PROJ603-Scripts.zip, you should have the following folder structure:

2. ASSUMPTIONS

i. AWS account with administrative-level permissions created, AWS CLI installed, and properly configured.

ii. All CloudFormation YAML (.yml) templates are valid and reference the correct AWS resources, file paths, and logical names.

iii. S3 buckets, DynamoDB tables, SNS topics, and all referenced resources have been created or their names have been updated in both the templates and scripts to match your actual project.

iv. All Lambda Python (.py) scripts are organized in the specified Lambda/Containment directory, and supporting scripts (e.g., for API or metadata logging) are present in the Lambda directory.

v. All required parameter values—such as S3 bucket names, EC2 key pair names, public IP addresses, and GuardDuty Detector IDs—are known and will be accurately substituted in each command before execution.

vi. IAM policy files (e.g., iam_policy.json) are located in the specified iam_policy directory and reference the appropriate resources for your deployment.

vii. You will upload Lambda ZIP packages to the correct S3 bucket paths as referenced by your CloudFormation templates.

viii. This deployment guide is intended to be executed phase by phase in a terminal shell or command-line interface, following the specified order of steps for a clean and repeatable deployment.

3. DEPLOYMENT STEPS AND COMMANDS

PHASE 1: Deploy VPC and EC2 Demo Instance

i. Change directory to CloudFormation:

```
cd CloudFormation
```

ii. Deploy VPC & EC2 with user data for web demo:

```
aws cloudformation deploy \
  --template-file vpc_ec2.yml \
  --stack-name SOARWebDemo \
  --capabilities CAPABILITY_NAMED_IAM \
  --parameter-overrides KeyName=<YOUR_EC2_KEYPAIR> MyIP=<YOUR_PUBLIC_IP>/32
```

NOTE:

Replace <YOUR_EC2_KEYPAIR> and <YOUR_PUBLIC_IP> accordingly.

Ensure the UserData section in your CloudFormation loads files from simple_web if needed.

II. PHASE 2: Deploy S3 Bucket and Enable GuardDuty

i. Deploy S3 bucket for Lambda code and logs:

```
aws cloudformation deploy \
  --template-file capstone-s3.yml \
  --stack-name capstone-soar-bucket-stack \
  --capabilities CAPABILITY_NAMED_IAM
```

ii. Deploy GuardDuty (if not already enabled):

```
aws cloudformation deploy \
  --template-file guardduty.yml \
  --stack-name GuardDutyStack \
  --capabilities CAPABILITY_NAMED_IAM
```

III. PHASE 3: Test GuardDuty with Sample Findings

i. Get your GuardDuty Detector ID:

```
aws guardduty list-detectors
```

ii. Simulate findings for quick check (Optional):

aws guardduty create-sample-findings --detector-id <YOUR_DETECTOR_ID>

Replace <YOUR_DETECTOR_ID> with your actual value.

IV. PHASE 4: Deploy DynamoDB and SNS

i. Deploy DynamoDB tables and SNS topics:

```
aws cloudformation deploy \
  --template-file dynamodb_sns.yml \
  --stack-name SOARDataLayer \
  --capabilities CAPABILITY_NAMED_IAM
```

V. PHASE 5: Package and Upload Lambda Functions

i. **Change directory to Lambda/Containment for playbook Lambdas:**

```
cd ../Lambda/Containment
```

ii. **Zip and upload each Lambda function to S3**

(Replace capstone-soar with your actual bucket name if different.)

```
zip geoip_threat_lambda.zip geoip_threat_lambda.py
aws s3 cp geoip_threat_lambda.zip s3://capstone-soar/lambda/geoip_threat_lambda.zip

zip iam_anomaly_lambda.zip iam_anomaly_lambda.py
aws s3 cp iam_anomaly_lambda.zip s3://capstone-soar/lambda/iam_anomaly_lambda.zip
```

zip iam_exfiltration_lambda.zip iam_exfiltration_lambda.py

aws s3 cp iam_exfiltration_lambda.zip s3://capstone-soar/lambda/iam_exfiltration_lambda.zip

zip port_scanning_lambda.zip port_scanning_lambda.py

aws s3 cp port_scanning_lambda.zip s3://capstone-soar/lambda/port_scanning_lambda.zip

zip s3_unauthorized_access_lambda.zip s3_unauthorized_access_lambda.py

aws s3 cp s3_unauthorized_access_lambda.zip s3://capstone-soar/lambda/s3_unauthorized_access_lambda.zip

zip ssh_brute_force_lambda.zip ssh_brute_force_lambda.py

aws s3 cp ssh_brute_force_lambda.zip s3://capstone-soar/lambda/ssh_brute_force_lambda.zip

zip tor_access_lambda.zip tor_access_lambda.py

aws s3 cp tor_access_lambda.zip s3://capstone-soar/lambda/tor_access_lambda.zip

zip web_login_abuse_lambda.zip web_login_abuse_lambda.py

aws s3 cp web_login_abuse_lambda.zip s3://capstone-soar/lambda/web_login_abuse_lambda.zip

iii. **Zip and upload supporting Lambda functions (from Lambda/):**

```
cd ..

zip logMetadataFunction.zip logMetadataFunction.py

aws s3 cp logMetadataFunction.zip s3://capstone-soar/lambda/logMetadataFunction.zip


zip nacl_cleanup_lambda.zip nacl_cleanup_lambda.py

aws s3 cp nacl_cleanup_lambda.zip s3://capstone-soar/lambda/nacl_cleanup_lambda.zip


zip api_soar_data.zip api_soar_data.py

aws s3 cp api_soar_data.zip s3://capstone-soar/lambda/api_soar_data.zip
```

VI. PHASE 6: Deploy Lambda Functions and EventBridge Rules

i. **Change to CloudFormation directory:**

```
cd ../CloudFormation
```

ii. **Deploy all Lambda and event rules using the master and individual stack files:**

```
aws cloudformation deploy \

  --template-file soar_containments_master.yml \
```

```
  --stack-name SOARContainmentsStack \
  --capabilities CAPABILITY_NAMED_IAM

aws cloudformation deploy \
  --template-file log_metadata_function.yml \
  --stack-name SOARLogMetadataFunctionStack \
  --capabilities CAPABILITY_NAMED_IAM

aws cloudformation deploy \
  --template-file nacl_cleanup_event_rule.yml \
  --stack-name SOARNaclCleanupEventRuleStack \
  --capabilities CAPABILITY_NAMED_IAM

aws cloudformation deploy \
  --template-file geoip_threat_event_rule.yml \
  --stack-name SOARGeoIPThreatEventRuleStack \
  --capabilities CAPABILITY_NAMED_IAM
```

```
aws cloudformation deploy \
  --template-file iam_anomaly_event_rule.yml \
  --stack-name SOARIAMAnomalyEventRuleStack \
  --capabilities CAPABILITY_NAMED_IAM

aws cloudformation deploy \
  --template-file iam_exfiltration_event_rule.yml \
  --stack-name SOARIAMExfiltrationEventRuleStack \
  --capabilities CAPABILITY_NAMED_IAM

aws cloudformation deploy \
  --template-file port_scanning_event_rule.yml \
  --stack-name SOARPortScanningEventRuleStack \
  --capabilities CAPABILITY_NAMED_IAM
```

```
aws cloudformation deploy \
  --template-file s3_unauthorized_access_event_rule.yml \
  --stack-name SOARS3UnauthorizedAccessEventRuleStack \
  --capabilities CAPABILITY_NAMED_IAM

aws cloudformation deploy \
  --template-file ssh_brute_force_event_rule.yml \
  --stack-name SOARSSHBruteForceEventRuleStack \
  --capabilities CAPABILITY_NAMED_IAM

aws cloudformation deploy \
  --template-file tor_access_event_rule.yml \
  --stack-name SOARTorAccessEventRuleStack \
  --capabilities CAPABILITY_NAMED_IAM
```

```
aws cloudformation deploy \
  --template-file web_login_abuse_event_rule.yml \
  --stack-name SOARWebLoginAbuseEventRuleStack \
  --capabilities CAPABILITY_NAMED_IAM
```

VII. PHASE 7: Attach IAM Policy to Each Lambda Role

i. **Change directory to iam_policy**

```
cd ../iam_policy
```

ii. **Attach policy to all Lambda roles**

```
for role in SSHBruteForceResponderRole PortScanningResponderRole
IAMAnomalyResponderRole IAMExfiltrationResponderRole TorAccessResponderRole
WebLoginAbuseResponderRole GeoIPThreatResponderRole
S3UnauthorizedAccessResponderRole
do
  aws iam put-role-policy \
    --role-name $role \
    --policy-name LambdaSOARPolicy \
    --policy-document file://iam_policy.json
  echo "Policy attached to $role"
done
```

Adjust the Lambda role names accordingly.

VIII. PHASE 8: Deploy API Gateway and Dashboard

i. **Change directory to Lambda:**

```
cd ../Lambda
```

ii. **Zip and upload API Lambda:**

```
zip api_soar_data.zip api_soar_data.py
aws s3 cp api_soar_data.zip s3://capstone-soar/lambda/api_soar_data.zip
```

iii. **Change directory to CloudFormation:**

```
cd ../CloudFormation
```

iv. **Deploy API Gateway and Lambda stack:**

```
aws cloudformation deploy \
 --template-file soar_data_api.yml \
 --stack-name SOARDataAPIStack \
 --capabilities CAPABILITY_NAMED_IAM
```

v. **Host dashboard (Github_UI/) via GitHub Pages or S3 (optional):**

- Place index.html, script.js, and style.css in your GitHub repo or S3 static website bucket.
- Update API endpoint in script.js if your API Gateway URL change

IX. PHASE 9: Testing

i. Run simulated and real attacks from Kali or your test host:

- SSH brute force: hydra -l demo -P /usr/share/wordlists/rockyou.txt ssh://<EC2_PUBLIC_IP>
- Port scanning: nmap -sS -T4 -p- <EC2_PUBLIC_IP>
- Web login abuse: use login.html or login.php with repeated failed logins.
- S3 unauthorized: change S3 permissions on a test bucket.
- IAM anomaly: use key from a different geo/location.

ii. Verify results:

- GuardDuty findings
- EventBridge and Lambda triggers
- DynamoDB/S3/SNS logging
- Dashboard population and notifications