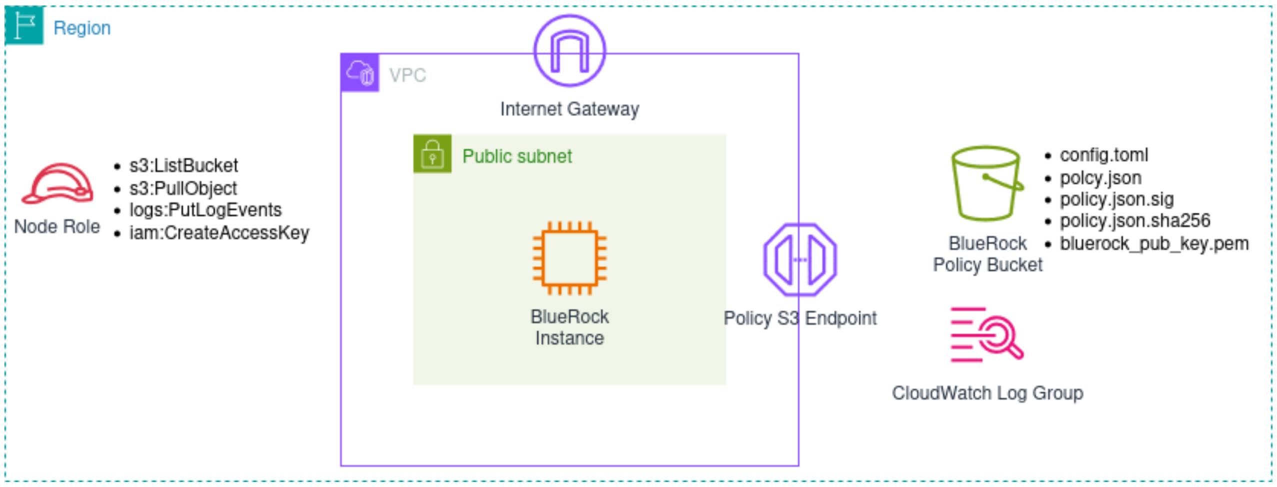
BlueRock AWS CloudFormation Deployment

Introduction

The following is a guide that shows how to create and deploy a BlueRock Node using a CloudFormation template in AWS.



Architecture Diagram

Above is the architecture created by the CloudFormation Template. It contains the following components:

- The BlueRock Node Instance. This instance and its workloads are protected by BlueRock, it contains the following:
  - BlueRock Rule Processing Engine: This container manages the node's policy and collects information about the node for rule enforcement. BlueRock policies can be configured for application and container runtime as well as process and file level controls.
  - Trex: Trex is an internal tool that turns simple json policy files into signed BlueRock consumable policy files. After writing a new policy file it needs to be processed by Trex before being uploaded to the policy bucket
  - OTel Collector: BlueRock manages its logs using Open Telemetry Receivers, Processors and Collectors. An intermediate collector has been placed as a

container on this instance to allow for ease of access in the log management for this marketplace listing. All BlueRock Logs are sent through this intermediary collector on their way to Cloudwatch.

- Additional AWS services: This template utilizes additional Amazon services for configuration and event monitoring
  - Amazon S3: This service is used to store signed policies in a BlueRock Policy S3 Bucket
  - Amazon CloudWatch: BlueRock sends events to a CloudWatch Log Group via the OTel Collector

Installation

Prerequisites

The following NodeInstanceTypes are supported with BlueRock. Larger instance sizes should also be compatible, but the below list has been validated:

```
t3.2xlarge
t3.xlarge
t3.large
t3.medium
t3.micro
t3.nano
t3.small
m5.2xlarge
m5.xlarge
m5.large
r5.2xlarge
r5.xlarge
r5.large
r5n.2xlarge
r5n.xlarge
r5n.large
i4i.2xlarge
i4i.xlarge
i4i.large
d3.2xlarge
d3.xlarge
```

The BlueRock Installation requires customers obtain the following artifacts:

- BlueRock CloudFormation Template
- BlueRock Node AMI: Shared through AWS accounts

To use this Cloud Formation template, ensure that the calling entity has the permissions necessary to call for `CAPABILITY_NAMED_IAM` capabilities. Additionally, ensure that an AWS managed ssh key is present in the region. As a parameter of the template, a key managed by the EC2 service is needed for developers to access the nodes. Developers should have access to the name of their key and the private and public key files.

Using CloudFormation

Through the AWS Web Console

1. Navigate to the CloudFormation Service Page
2. Select **Create stack** → **With new resources (standard)**
3. Select **Choose an existing template**
    a. Specify template source by selecting the **Upload a template file**
4. Upload the BlueRock CloudFormation Template
5. Select **Next**
6. Fill in all template parameters (see Parameter table below)
7. Select **Next**
8. Under **Capabilities** Acknowledge the creation of IAM Roles / Policies (see Policy Table Below)
9. Select **Next**
10. Confirm Stack creation and **Submit**

Through the AWS CLI

Make sure the CLI is installed and has privileges to specify `CAPABILITY_NAMED_IAM`

Call `create-stack` specifying the parameter file, template file and capabilities

```
aws cloudformation create-stack --stack-name <string-name> --template-
body file://<template-file-location> --capabilities
CAPABILITY_NAMED_IAM --parameters file://<json-param-file-location>
```

Example Parameters File

```
[
  {
    "ParameterKey": "AllowIp",
    "ParameterValue": "<your_ip_address>/32"
  },
  {
    "ParameterKey": "ConfigBucket",
    "ParameterValue": "my-s3-bucket"
  },
  {
    "ParameterKey": "NodeInstanceType",
    "ParameterValue": "t3.xlarge"
  },
  {
    "ParameterKey": "NodeAmi",
    "ParameterValue": "ami-xxxxx"
  },
  {
    "ParameterKey": "Prefix",
    "ParameterValue": "unplugged-single-1"
  },
  {
    "ParameterKey": "SampleHostName",
    "ParameterValue": "bru-host"
  },
  {
    "ParameterKey": "SshKeyName",
    "ParameterValue": "my-ssh-key"
  },
  {
    "ParameterKey": "TrexVersion",
    "ParameterValue": "trex-25.28_bru-release-beta"
  },
]
```

CloudFormation Template Parameters

Parameter Name	Description
`AllowIp`	Developer's IP address to whitelist in the BlueRock Instance for SSH access
`ConfigBucket`	Name of S3 bucket generated to hold BlueRock policy files (must be globally unique)
`NodeInstanceType`	Size of BlueRock node to be created.
`NodeAmi`	Image AMI ID for BlueRock EC2 Instance
`Prefix`	Unique Identifier appended to AWS resource names
`SampleHostName`	Unique Identifier for BlueRock UC Container name
`SshKeyName`	Name of AWS managed SSH key, used for authorizing access to the BlueRock Instance
`TrexVersion`	Version name of trex package tied to release

This stack takes about 5 minutes to build.

Installation Validation Checks

SSH into BlueRock Instance

Once the stack has completed, select it (**CloudFormation** → **<Stack-Name>**) and view the **Resources** tab. Here you can search for the `NodeInstance` instance. Connect to `NodeInstance` via an SSH command using the key specified in `SshKeyName`.

SSH Command

```
ssh -i <priv-key> ec2-user@<instance-pub-ip>
```

Check Services

The BlueRock Services are initialized through the `uc-docker.service` service. Ensure that the service and the Rule Engine container are running and enabled. Additionally, use the provided script to view the logs exported by the Rule Engine.

```
$sudo systemctl status uc-docker.service
$ docker ps
CONTAINER ID   IMAGE
COMMAND                    CREATED       STATUS       PORTS     NAMES
5b5e00d1881f   ultracontrol:latest
"/opt/bluerock/sbin/…"   7 hours ago   Up 7 hours             uc
e0186c875227   public.ecr.aws/aws-observability/aws-otel-
collector:latest   "/awscollector --con…"   7 hours ago   Up 7 hours
otel-collector

$/opt/bluerock/bin/uc-docker.sh logs
```

Finally, ensure the OTEL collector service is running and enabled.

```
$docker logs otel-collector
```

Policy Configuration

To enable the enforcement of policies, BlueRock Instances need a reachable policy file in place. In this architecture, we will load the policy file into the CF generated S3 bucket.

First we will create the signed policy file on the BlueRock Instance.

Once connected, navigate to `~/policy` where a sample policy (`bru_policy.json`) with all protection mechanisms set to observe mode is present. Once the policy is modified we need to sign it using Trex.

```
$ python3 /opt/bluerock/trex/trex.py <policy-file>
$ tar xvf buv_policy.tar
```

Ensure that the `bru-venv` is activated when using any python commands

```
$ source /home/ec2-user/build/bru-venv/bin/activate
```

This will generate a tar file containing the new policy, the policy's signature and a sha256 sum of the policy. All three of these files need to be uploaded to the BlueRock Policy Bucket.

The Services Instance also generates a new public key on creation, this key is specified in the `trex.toml` file located in the `policy` directory. This public key also must be uploaded to the BlueRock Policy Bucket.

Pushed Item	Description
policy.json	Json version of buv_policy.yaml, this includes additional configs supplied by Trex
policy.json.sig	signature file for policy.json, needed for verifying authenticity of policy file
policy.json.sha256	sha256 sum of policy.json, needed for verifying integrity of policy file

```
aws s3 sync . s3://<bluerock-cfg-s3-bucket-name> --exclude "*" --
include "policy.json*"
aws s3 cp ./bluerock_pub_key.pem s3://<bluerock-cfg-s3-bucket-name>
```

These policy files take ~10 mins to propagate to each node.

Refer to the Configuring BlueRock Security Policies section for instructions on editing and tuning policies.

Any time the policy file is updated the tar needs to be re-created and pushed to the policy bucket.