# Cloud Deployment Guide and Cloud Security IRP for Adversary Emulation Environment

12/2/24

# Introduction

## Purpose

The purpose of this document is to show how to set up an environment for testing red team abilities utilizing the mitre caldera platform. Mitre caldera is an open source software that can be downloaded on github. The platform is built on the mitre attack framework, which allows for abilities to be separated into different categories based on things like what type of attack it is, what it is affecting, etc. This environment will be deployed in AWS using many different resources like Vpc's, instances, internet gateways, security groups, and alarms.

## Scope

The scope of this guide will be for all of the resources you create in aws as well as anyone maintaining the environment.

## Maintenance

ADMIN, the creator of the environment will be in charge of maintenance on the environment. This includes the configuration of the environment, the abilities executed, as well as the measures used to secure the environment.

# Deployment Guide

## Overview

I have decided to create an AWS environment that will be able to support multiple EC2 instances. For the sake of this deployment I will only be deploying linux VM's, but if someone wanted to they could deploy caldera on a windows instance to emulate windows specific attacks. Each of these EC2 instances will serve a different role in the goal of emulating adversary activity:

- Instances that will serve as the executor of attacks
    - Ubuntu VM
        - One in each subnet for redundancy
- Instances that will serve as the target machine
    - Ubuntu VM
        - One in each subnet for redundancy

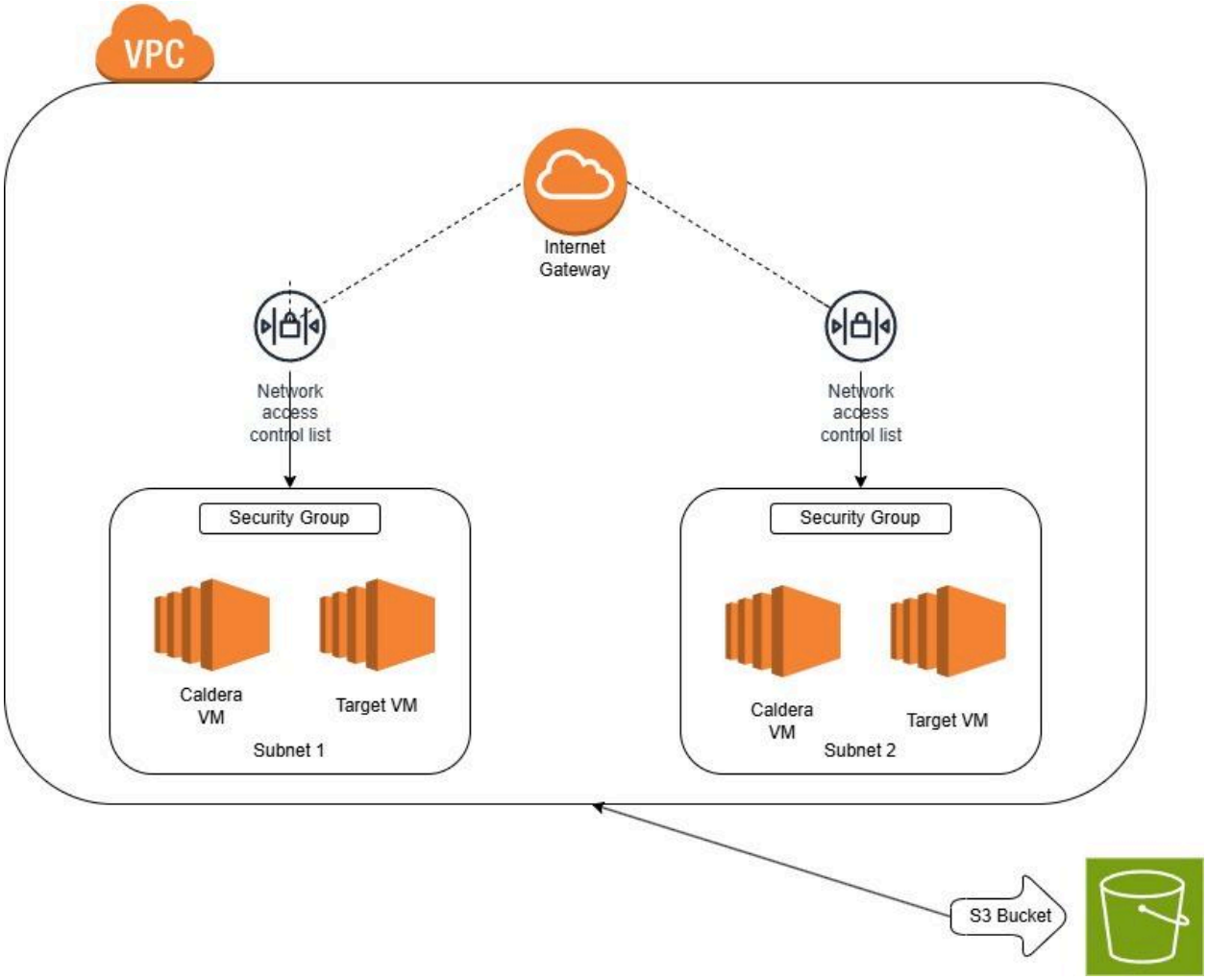
The environment will employ measures for security, as well as ensure high availability, redundancy, and scalability. This environment will also be built in AWS and utilize the following services.

(1) Security groups
(2) Network access control lists (nACLs)
(3) Subnet and ec2 instance redundancy
(4) Internet Gateways
(5) S3 bucket policies

The VPC will allow for redundancy and high availability in the environment. There are multiple ec2 instances in multiple subnets which will allow for multiple regions of use.

There will be high availability in the case that one of the instances in one subnet stops working, you will be able to switch to another one. An idea that should be implemented with this deployment is the idea of load balancing. If there are multiple users of an instance, they should automatically be moved to the other subnet for use of testing.

The environment will also use s3 buckets for high recovery as well as for storage. The s3 bucket will be set up to hold ova files of the working caldera linux VM's. In the event of an instance failing or needing to access a backup, the instance can be recreated from the ova file. This is done by creating an image and redeploying the instance. There are also stores on my local machine, so in the event that the files in the s3 bucket stop working, I could restore the vm's. This environment also employs scalability by allowing for instances to be created from different vm backups as well as for the instance's resources to be changed based on which AWS tier you are using. In regards to storage security, there will be no public access to the bucket, so only the administrator can access the contents. I also used DenyObjectsThatAreNotSSEKMS to make sure that every object written to the bucket has to be encrypted.




This figure shows the Virtual Private cloud and its resources including nACL's, multiple subnets, and multiple ec2 instances. Also shows how the s3 bucket will be used alongside the VPC.

## Configuration Of Environment

*I will only be deploying caldera in a local environment. This deployment will explain how to do so in a secure AWS environment.*

This configuration should be done in an administrator account that is under the root user. In the event of a disaster or incident where access to the admin account needs to be revoked, redeploy using the root user. Make sure the root user is locked when not in use.

**VPC and Subnet Setup and Configuration**

1. Sign into aws.amazon.com and navigate to VPC.
2. Click “create VPC”
3. Under IPv4 CIDR enter 192.168.0.0/24
4. Move to the subnet tab
5. Click “create subnet”
6. Select the VPC that was just created
7. Give the subnet a name like subnet one
8. Under IPv4 CIDR enter 192.168.0.0/26
    a. Set this subnet to an availability zone of your choosing
9. Repeat steps 5-8 except with the name subnet two and an IPv4 CIDR range of 192.168.0.64/26
    a. Also set this subnet to a different availability zone than the subnet one
    b. These steps create a virtual private cloud consisting of two subnets that we will use to ensure our environment has redundancy.
10. Next navigate to the internet gateway tab
11. Click create internet gateway and attach it to the VPC created in step 2
    a. This internet gateway will provide a way for the instances to reach the internet

**Configuring Security Groups**

1. Click on the security groups tab
2. Create security group
3. Name the security group
4. These inbound outbound rules will allow for the ec2 instances to communicate with each other for test execution. The rules allow for HTTP and HTTPS to communicate anywhere ingress or egress. We want this to be configured so the caldera and target machines can communicate with one another. We also have a SSH rule configured for ingress and egress to our IP. This will allow us to SSH into the instance from our computer. The last rule is an inbound rule for the port caldera will be running on. This port needs to be open on inbound for the Target Machine, with only traffic from the IP of the Caldera machine. This allows for a secure connection between the two.

| Security Group Inbound Rules | | | |
|---|---|---|---|
| Type | Protocol | Port range | Source |
| HTTPS | TCP | 443 | 0.0.0.0/0 |

| | | | |
|---|---|---|---|
| HTTP | TCP | 80 | 0.0.0.0/0 |
| SSH | TCP | 22 | <your IP address> |
| | TCP | 8888 | <IP of Caldera VM> |

| Security Group Outbound Rules | | | |
|---|---|---|---|
| Type | Protocol | Port range | Destination |
| HTTPS | TCP | 443 | 0.0.0.0/0 |
| HTTP | TCP | 80 | 0.0.0.0/0 |
| SSH | TCP | 22 | <your IP address> |

**Configuring nACLs**

1. Click the network ACL's tab
2. Click create nACL
3. Name the nACL
4. Set the following rules
    a. HTTP and HTTPS will have the lowest rule # to allow for traffic between the Caldera and Target machines. Next we will allow ssh traffic for our IP so we can access the instances. We will also be specifying an ephemeral port range for communication with server applications. This range of ports is recommended for use with amazon linux, so make sure to change if using something different. These ports allow for a temporary transport protocol for communicating with IP. All other connections (Ingress or Egress) should be set to deny.

| nACL Inbound Rules | | | | | |
|---|---|---|---|---|---|
| Rule # | Type | Protocol | Port range | Source | Allow/Deny |
| 100 | HTTPS | TCP | 443 | 0.0.0.0/0 | Allow |

| | | | | | |
|---|---|---|---|---|---|
| 110 | HTTP | TCP | 80 | 0.0.0.0/0 | Allow |
| 120 | SSH | TCP | 22 | <your IP address> | Allow |
| 130 | Custom TCP | TCP | 32768-61000 | 0.0.0.0/0 | Allow |
| * | All traffic | All | All | 0.0.0.0/0 | Deny |

| nACL Outbound Rules | | | | | |
|---|---|---|---|---|---|
| Rule # | Type | Protocol | Port range | Destination | Allow/Deny |
| 100 | HTTPS | TCP | 443 | 0.0.0.0/0 | Allow |
| 110 | HTTP | TCP | 80 | 0.0.0.0/0 | Allow |
| 120 | SSH | TCP | 22 | <your IP address> | Allow |
| 130 | Custom TCP | TCP | 32768-61000 | 0.0.0.0/0 | Allow |
| * | All traffic | All | All | 0.0.0.0/0 | Deny |

Setting up these security measures promotes a defense in depth (DID) approach to configuring our environment. By securing our VPC with security groups, nACL's, AWS policies and rules we ensure multiple layers of security throughout the configuration process. This will leave less vulnerabilities for a threat actor to take advantage of.

**Ec2 Setup and Configuration**

1. Navigate to EC2
2. Click launch instance
3. Enter desired name for the Caldera instance
4. Select desired OS image and instance type. For this deployment make sure to use an OS image that supports a GUI so we can access our caldera interface through

http. For the instance type, I would use a medium tier at minimum.

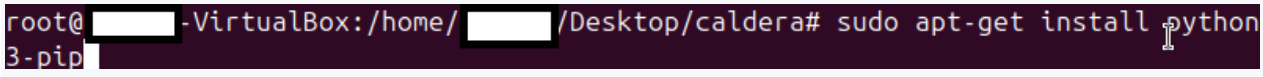
5. Create a new key pair that will be used to ssh into the instance.
6. Download pem file and save to a secure place
7. If using windows, navigate to the file properties and make sure your windows account is the only account with access to the pem file
    a. If this step is omitted, AWS will not let you ssh into the instance
8. Select the VPC that was created
9. Select the subnet with the CIDR range 192.168.0.64/26(Subnet 1)
10. Select the security group
11. Launch the instance
12. In order to create the other 3 instances you will need to repeat steps 2-4 and 8-11 with these modifications
    a. Target VM in 192.168.0.64/26 subnet
        i. Name it to something like Caldera Target
        ii. Select the 192.168.0.64/26 subnet
        iii. Select the same security group
    b. Caldera VM in 192.168.0.0/26(Subnet 2)
        i. Name it to something like Caldera subnet 2 machine
        ii. Select the 192.168.0.0/26 subnet
        iii. Select the same security group
    c. Target VM in 192.168.0.0/26 subnet
        i. Name it to something like Caldera Target subnet 2
        ii. Select the 192.168.0.0/26 subnet
        iii. Select the same security group

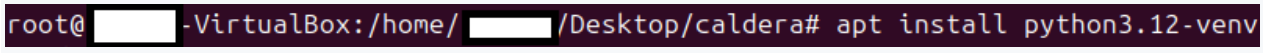
**SSH into the Caldera Machine**

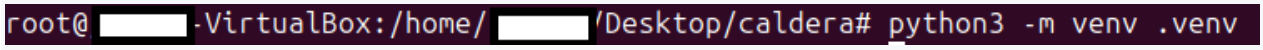
1. Copy the public IP address of the caldera subnet 1 instance
2. Navigate to your terminal/ssh client
3. Enter the following command
4. ssh -i /path/key-pair-name.pem instance-user-name@instance-public-dns-name
5. You should now be able to access the linux instance we will use for the Caldera machine

**Configuring Caldera Machine**

1. Navigate to the terminal of your machine
2. The configuration will be a series of terminal commands
3. Start with entering sudo su to make sure you are the root user

4. apt install git
    a. Will install the git command so you can install caldera directly from github
5. git clone https://github.com/mitre/caldera.git --recursive
    a. Clone the repository on your machine
6. cd caldera
    a. Change to the caldera directory that was just cloned on the machine
7. sudo apt-get update
    a. Will update any packages. Need to do this to proceed with next steps
8. 

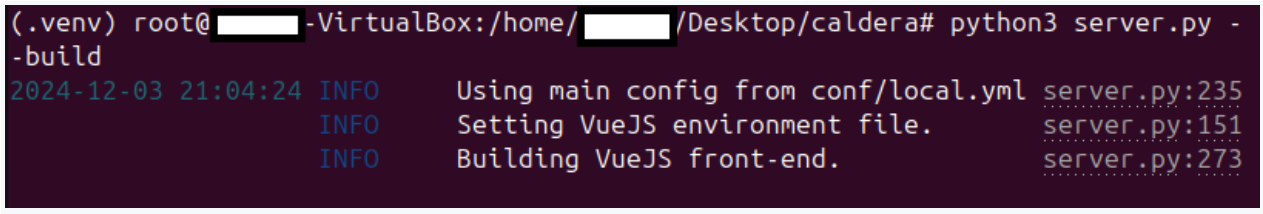
    a. Installs python pip package manager
9. 

    a. Installs the virtual environment capability
10. 

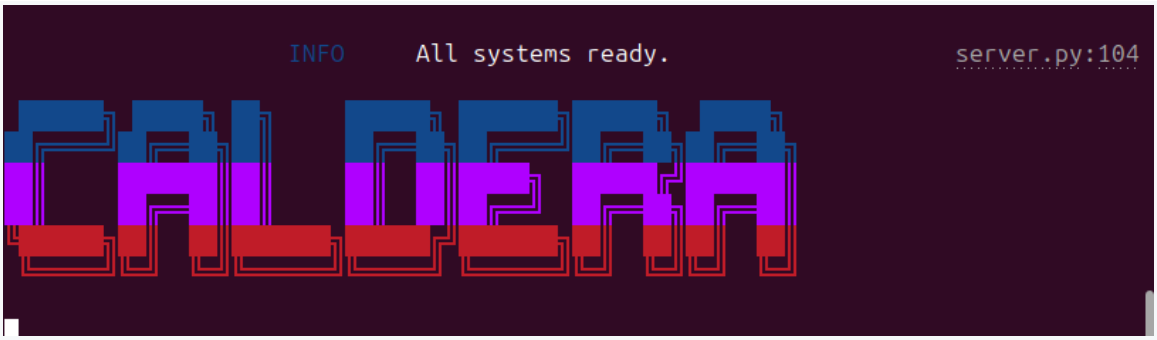
    a. Creates a new virtual environment
11. 

    a. Activates the virtual environment(Need to be in a virtual environment to install the requirements.txt file as well as build the server)
12. 

    a. Installs the requirements for building the Caldera server
13. 

    a. Installs NPM, which is a tool used during the server build
14. Change the http to your local host IP in the conf/local.yaml file
15. 

    a. This command will build the caldera server on the localhost IP

b. 

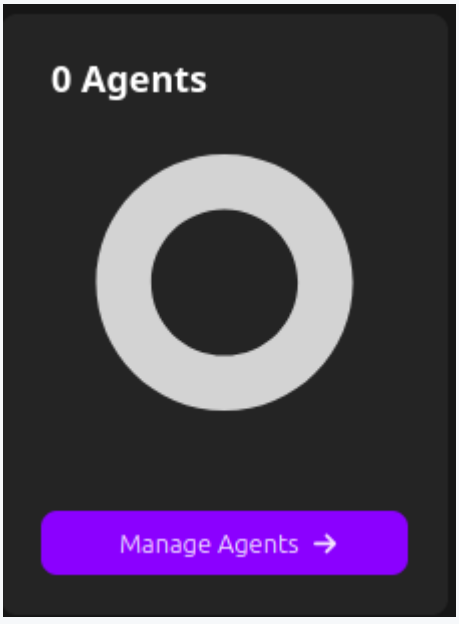

16. Navigate to the localhost on port 8888
17. Default password is in local yaml file

**SSH into the Caldera Target Machine**

6. Copy the public IP address of the Caldera Target subnet 1 instance
7. Navigate to your terminal/ssh client
8. Enter the following command
9. ssh -i /path/key-pair-name.pem instance-user-name@instance-public-dns-name
10. You should now be able to access the linux instance we will use for the Caldera Target Machine

**Configuring Caldera Target Machine**

1. To start the setup, you should be in the Caldera Machine
2. On the caldera vm navigate to the agents tab

a. 


3. Click deploy an agent
4. Select the type of agent(I chose a sandcat agent that communicates through http)
5. Select the os of the target vm (linux)
6. Enter the IP of the Caldera VM for the app contact http field

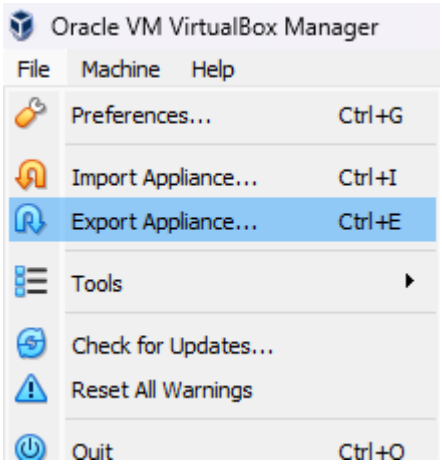
7. Copy the command in the default field
8. Move over to the target VM
9. Paste and run the command
10. After running the command the agent will be deployed and will be available to run abilities

# Repeat this process of configuration for Subnet 2

**S3 Bucket Setup and Configuration**

For the s3 bucket setup I am going to use the bucket to store an export of the configured VM's from my local Machine. The bucket will hold copies of both the configured Caldera Machine and Target Machine.

1. Download and run the installation of the AWS CLI
    - On windows run the following command
        i. C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
    - Next run this command to verify the installation
        i. C:\> aws --version aws-cli/2.19.1 Python/3.11.6 Windows/10 exe/AMD64 prompt/off
2. Next navigate to the s3 bucket tab of AWS
3. Click Create bucket
4. Give the bucket a name
5. This bucket will only be accessed by this AWS account for maximum security
    - ACLs should be disabled
    - Public Access should be blocked
    - Enable Bucket Versioning
    - Enable Bucket Key
    - Create the Bucket
6. Next I exported the VM as an OVA file from virtualbox




7. The next step is to upload the ova file to the bucket
    - Start by clicking into the bucket that was created
    - Click add files
    - Click the upload button
8. To import the vm from the s3 bucket container you will have to specify directions in a containers.json file

```
[
  {
    "Description": "My Server OVA",
    "Format": "ova",
    "UserBucket": {
        "S3Bucket": "amzn-s3-demo-import-bucket",
        "S3Key": "vms/my-server-vm.ova"
    }
  }
]
```

9. You then can run this command to import the image into aws
    - aws ec2 import-image --description "My server VM" --disk-containers "file://C:\import\containers.json"
10. After doing this you will be able to create an instance with this ova by selecting it in the AMI section.

**Monitoring and Logging**

For this environment I will be utilizing AWS monitoring services. I will be using AWS cloud watch to monitor the target machines. This will be a good way to monitor how tests are affecting both the performance and logging events. I will also be using AWS cloudtrail for monitoring all resources.

# Incident Response/Disaster Recovery Plan

## Incident Response Coordinator

ADMIN, the coordinator of this environment, will also serve as the coordinator for incident response. The plan below is composed to ensure environment continuity in the event of an incident. If ADMIN is transferring this environment to someone else, they can follow this deployment guide as well as this IRP to maintain the environment.

## Incident Response

The incidence response plan starts with making sure everything is as secure as possible from launch. We have utilized many different security measures to ensure the environment is secure. Using security groups and nACL's allows our instances to only be accessed by the necessary parties. We also have utilized many different availability rules to make sure that in the event of an incident, we can still use the environment for testing. By implementing these security measures before running the environment, we ensure that incidents can be avoided.

The next important aspect of the IRP is the use of backups and storage. There are multiple points where you can start in the event of both an incident or a disaster. Since we have backups of the instances in a s3 bucket, if one of the instances is rendered unusable, we can restore from the backup. Follow the steps to deploy again from an AMI. This should not take long as the backups are in working states. If the s3 buckets are not accessible or become corrupted, restore from one of the multiple local backups. These are saved in different locations for redundancy purposes. This would take a little bit more time, but may be useful

in certain situations.

In the event of a threat actor taking over parts of the environment, the first step is to evaluate. Look for what access they have and determine how you want to go about remediating the issue. If the threat actor gains access to an instance, destroy the instance and evaluate the security rules. In the event of a threat actor gaining access to an AWS account or admin functions, delete all resources and rebuild from the local paths.