Caldera Automation

Department of Engineering, Computing, and Cybersecurity

Candan
Marwin
Jacob
Professor: Dr. Gonzalo D. Parra

How to automate Caldera deployment using AWS

Outline

Why this project?

Summary of the Project: Automating Caldera deployment and management on AWS EC2 using Ansible and CloudFormation. This enables faster operations, enhanced security, and reduced human error in the deployment process

Rationale: To streamline the process of deploying Caldera on EC2 instances, improve maintainability and security of the system, and enhance efficiency through automation. This project was initiated to mitigate challenges associated with manual intervention and human errors

Challenges and Improvement: During the execution of the project, we faced challenges like command failures and missing libraries in Systems Manager. However, these were identified as areas of improvement and were resolved successfully, demonstrating adaptability and problem-solving skills

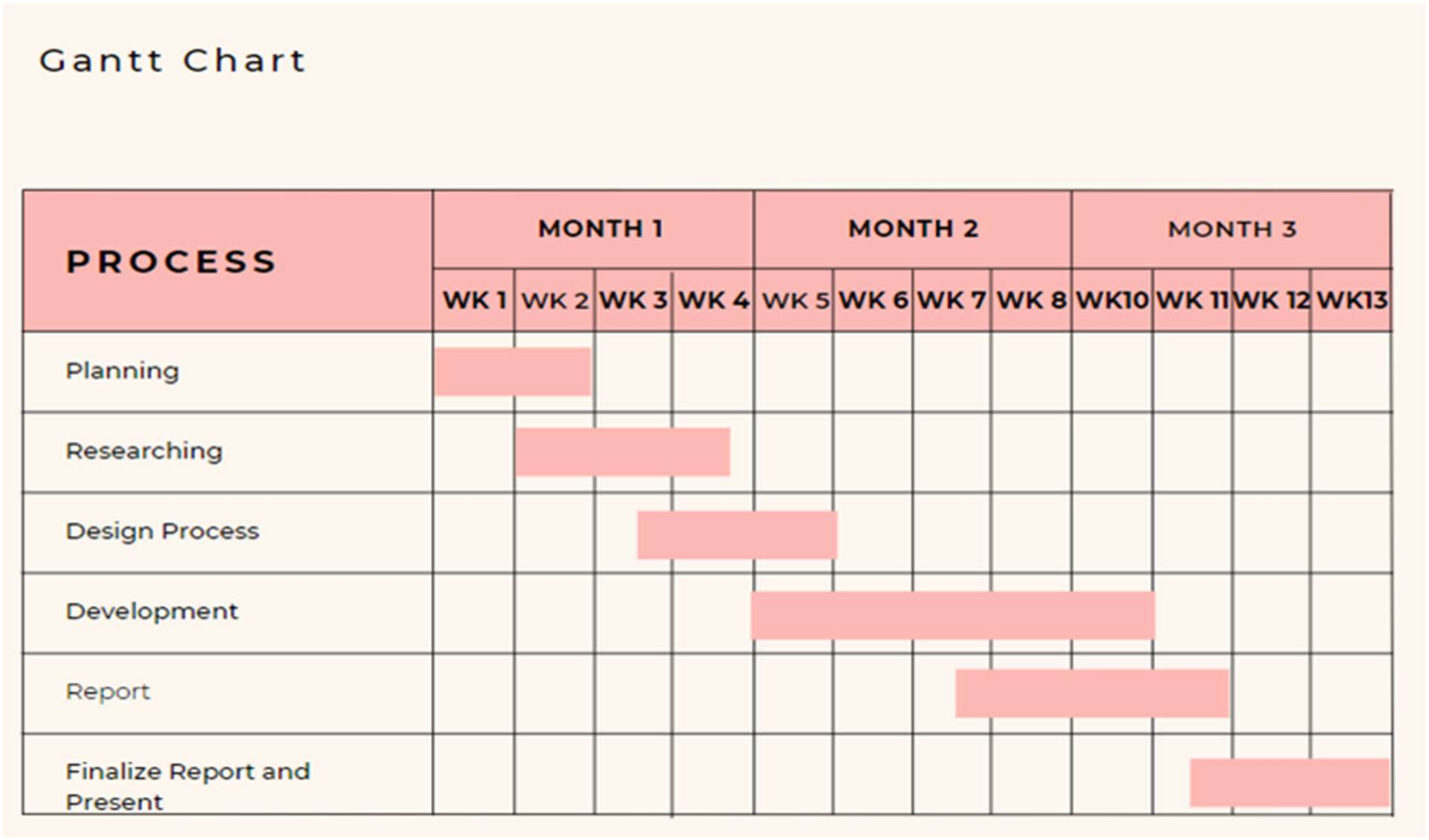
Professional Competence: The successful automation of the deployment process using CloudFormation and Ansible scripts showcases the ability to implement complex IT solutions. Moreover, troubleshooting the Systems Manager issues exhibits a strong understanding of AWS and DevOps tools.

Objective

Objective

- The main objective of this project is to automate the deployment and management of the Caldera platform on AWS EC2 instances using Ansible and CloudFormation, ultimately improving the efficiency, maintainability, and security of the system.

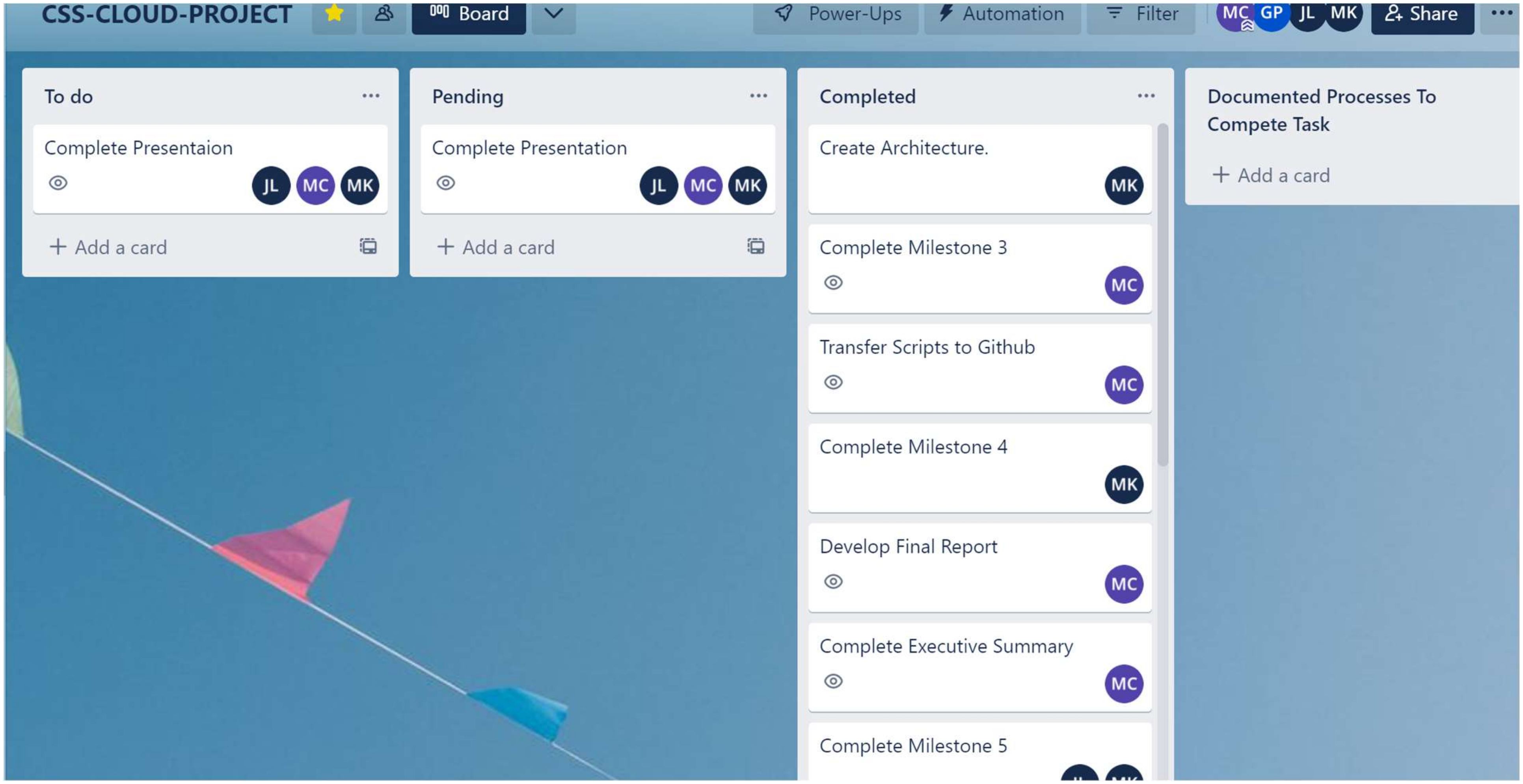
Timeline



Gantt Chart

PROCESS	MONTH 1				MONTH 2				MONTH 3			
	WK 1	WK 2	WK 3	WK 4	WK 5	WK 6	WK 7	WK 8	WK10	WK 11	WK 12	WK13
Planning	▨	▨										
Researching		▨	▨	▨								
Design Process			▨	▨	▨							
Development					▨	▨	▨	▨	▨			
Report							▨	▨	▨	▨		
Finalize Report and Present										▨	▨	▨

Management System



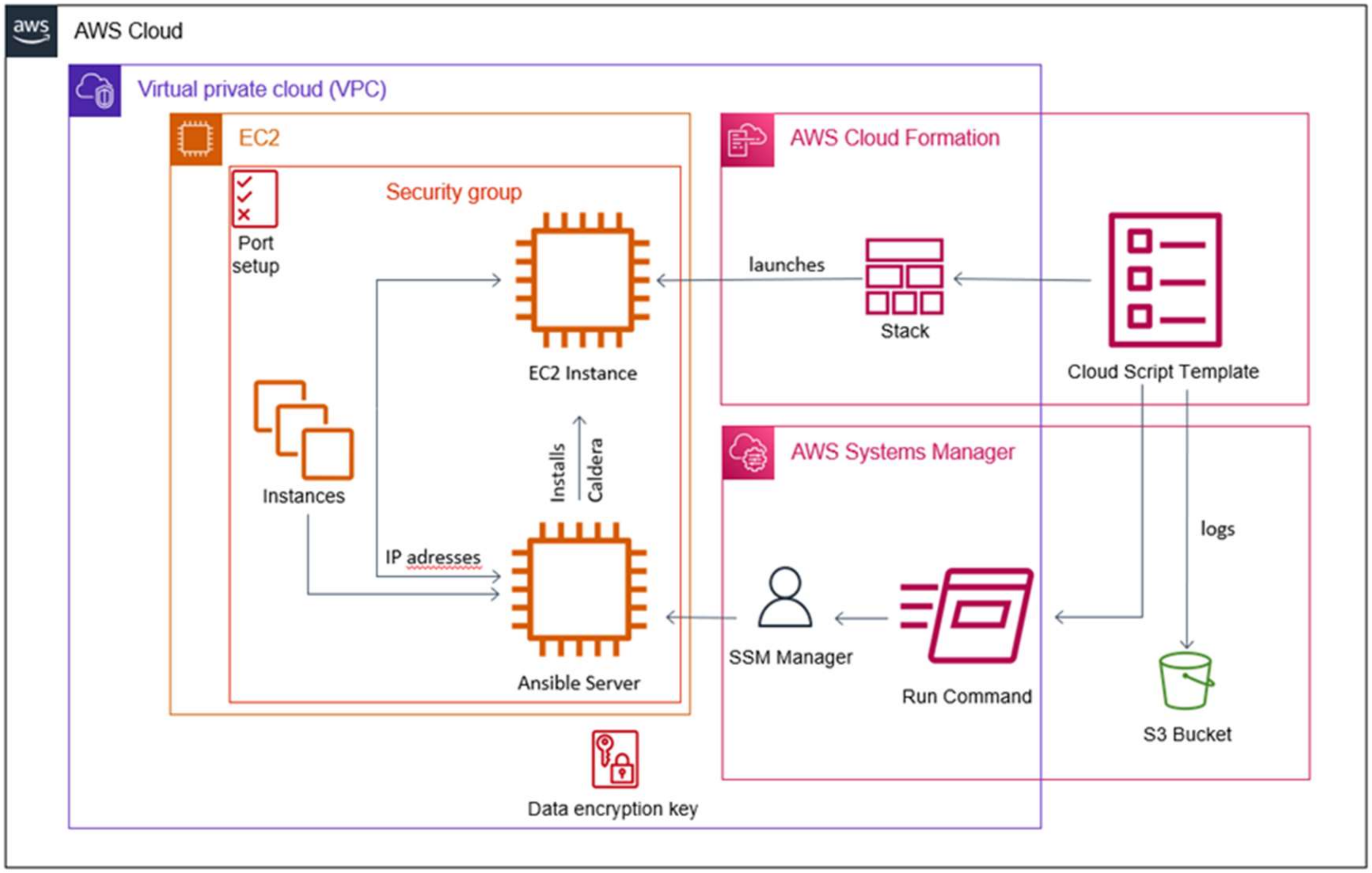
To do
- Complete Presentaion — JL MC MK
- + Add a card

Pending
- Complete Presentation — JL MC MK
- + Add a card

Completed
- Create Architecture. — MK
- Complete Milestone 3 — MC
- Transfer Scripts to Github — MC
- Complete Milestone 4 — MK
- Develop Final Report — MC
- Complete Executive Summary — MC
- Complete Milestone 5

Documented Processes To Compete Task
- + Add a card

Cloud Architecture


AWS Cloud
Virtual private cloud (VPC)
EC2
Security group
Port setup
EC2 Instance
Instances
Installs Caldera
IP adresses
Ansible Server
Data encryption key
AWS Cloud Formation
launches
Stack
Cloud Script Template
AWS Systems Manager
logs
SSM Manager
Run Command
S3 Bucket

Project Milestones

- Milestone 1
  - Create Ansible script to launch a web server & create cloud formation script to launch ec2 instances
- Milestone 2
  - Develop an Ansible playbook to install Caldera on a VM in AWS
- Milestone 3
  - Create a configuration file template for Caldera using Ansible to improve security by adding user credentials.
- Milestone 4
  - Set up an Ansible server with dynamic inventory to manage the deployment of Caldera.
- Milestone 5
  - Set up cloud formation script that will create ec2 instances and run the caldera-install ansible script on the ansible server to deploy caldera on the created ec2 instances

Milestone 1

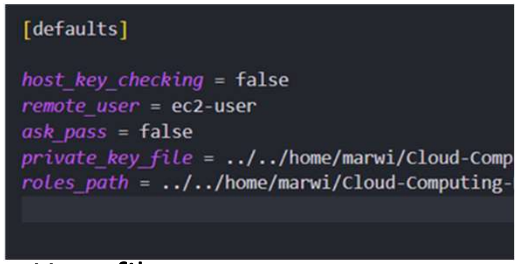
Create Ansible script to launch a web server & create cloud formation script to launch ec2 instances

Brief overview of Milestone 1 achievements:

- Set up Ansible on a local Windows machine using Cygwin
- Created and executed Ansible playbooks for various tasks
- Gained experience with Cloud Formation by creating a template for launching an EC2 instance with an Apache web server

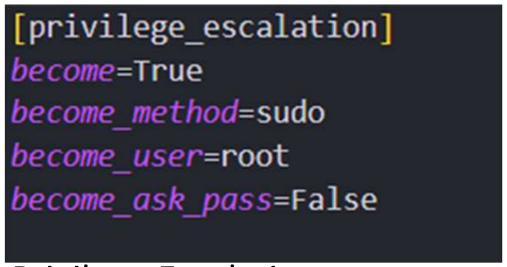
Setting up Ansible on a Local Machine

- Installed Cygwin to run Ansible on a Windows machine
- Followed guide to configure Ansible on Windows: [How to Install and Configure Ansible on Windows {3 Methods Explained} (phoenixnap.com)]
- Modified ansible.cfg, host file, and privilege escalation settings for seamless integration with AWS



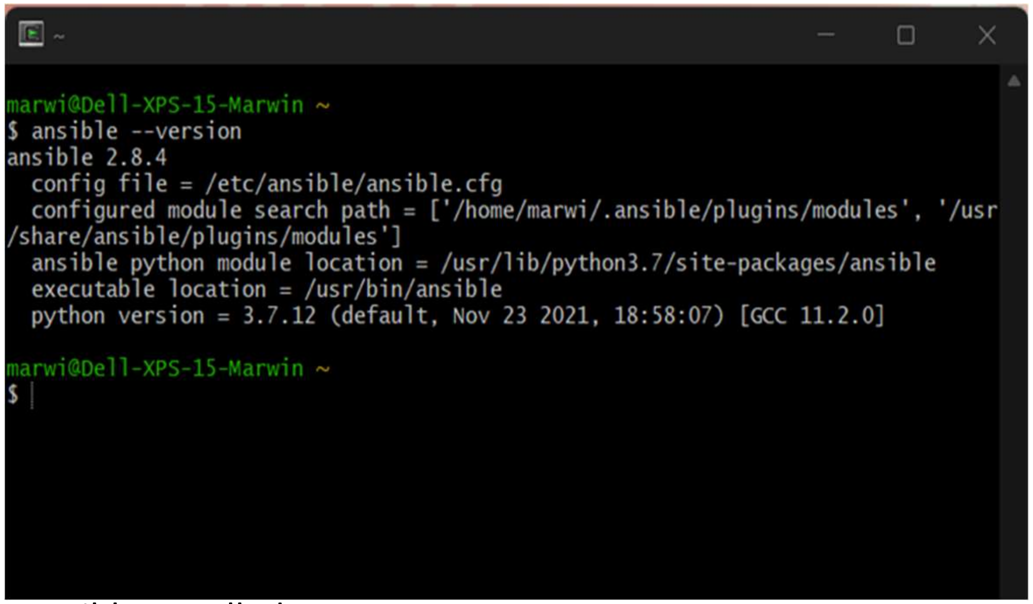
```
[defaults]

host_key_checking = false
remote_user = ec2-user
ask_pass = false
private_key_file = ../../home/marwi/Cloud-Comp
roles_path = ../../home/marwi/Cloud-Computing-
```

Host file



```
[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=False
```

Privilege Escalation

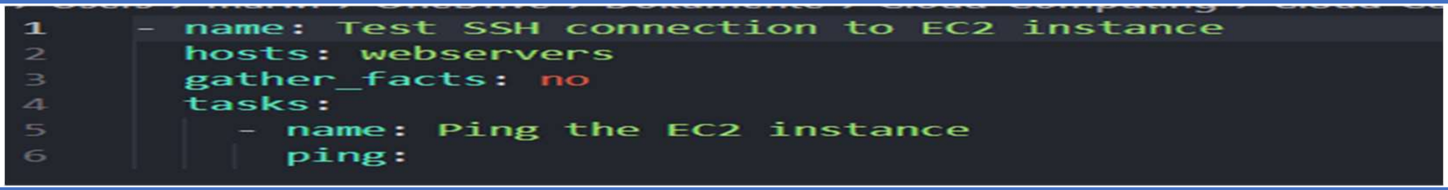


```
marwi@Dell-XPS-15-Marwin ~
$ ansible --version
ansible 2.8.4
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/marwi/.ansible/plugins/modules', '/usr
/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.7.12 (default, Nov 23 2021, 18:58:07) [GCC 11.2.0]

marwi@Dell-XPS-15-Marwin ~
$
```

Ansible Installed

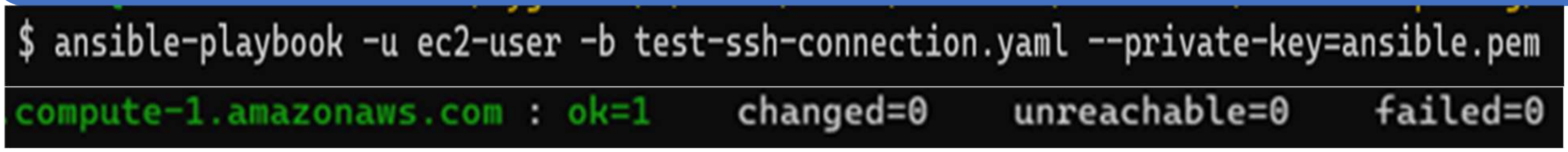
Creating Ansible Playbooks

Created a playbook to test SSH connection to EC2 instances



```
- name: Test SSH connection to EC2 instance
  hosts: webservers
  gather_facts: no
  tasks:
    - name: Ping the EC2 instance
      ping:
```

Successfully executed the playbook and pinged the EC2 instance

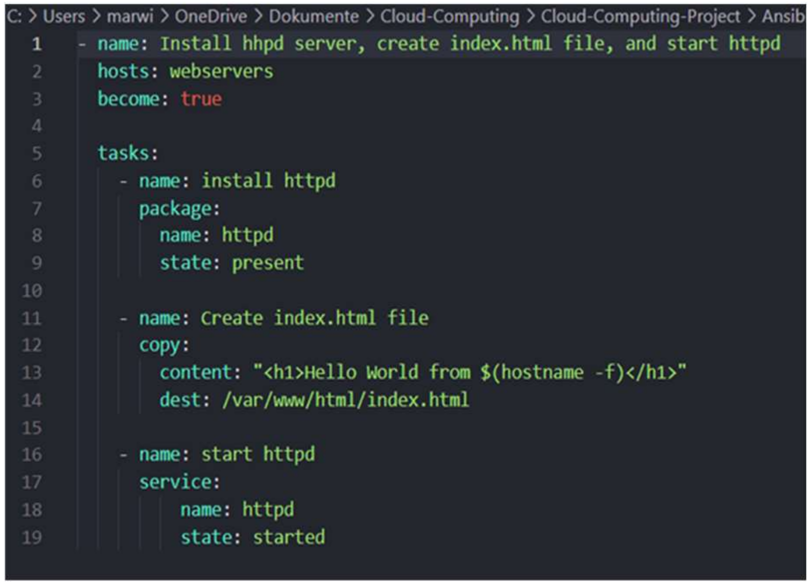


```
$ ansible-playbook -u ec2-user -b test-ssh-connection.yaml --private-key=ansible.pem
```

```
compute-1.amazonaws.com : ok=1    changed=0    unreachable=0    failed=0
```

Creating Ansible Playbooks

Created an Apache Server Ansible Script to launch an Apache server on an EC2 instance

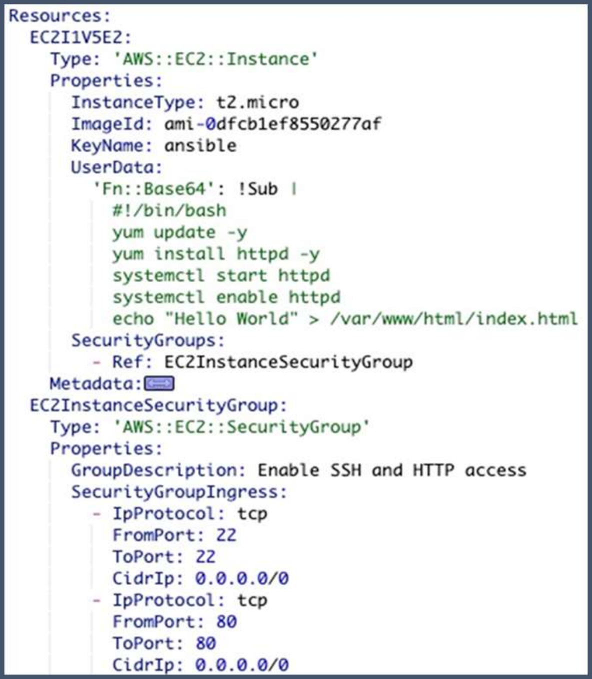

C: > Users > marwi > OneDrive > Dokumente > Cloud-Computing > Cloud-Computing-Project > Ansib

```
- name: Install hhpd server, create index.html file, and start httpd
  hosts: webservers
  become: true

  tasks:
    - name: install httpd
      package:
        name: httpd
        state: present

    - name: Create index.html file
      copy:
        content: "<h1>Hello World from $(hostname -f)</h1>"
        dest: /var/www/html/index.html

    - name: start httpd
      service:
         name: httpd
         state: started
```

Cloud Formation - Creating an EC2 Instance with Apache Web Server

EXPLORED CLOUD FORMATION TO ENHANCE AWS MANAGEMENT

CREATED A TEMPLATE TO LAUNCH AN EC2 INSTANCE WITH AN APACHE WEB SERVER INSTALLED



```
Resources:
  EC2I1V5E2:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: t2.micro
      ImageId: ami-0dfcb1ef8550277af
      KeyName: ansible
      UserData:
        'Fn::Base64': !Sub |
          #!/bin/bash
          yum update -y
          yum install httpd -y
          systemctl start httpd
          systemctl enable httpd
          echo "Hello World" > /var/www/html/index.html
      SecurityGroups:
        - Ref: EC2InstanceSecurityGroup
    Metadata:
  EC2InstanceSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Enable SSH and HTTP access
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
```

Milestone 2

Develop an Ansible playbook to install Caldera on a VM in AWS

Milestone 2: Caldera Playbook Development for AWS VMs

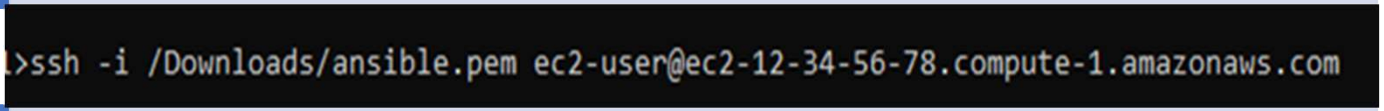
- Goal: Automate the installation and deployment of Caldera on an EC2 instance
- Manual installation process completed first to understand the process and lay the foundation for automation
- Script was developed and tested by manually installing Caldera on an EC2 instance and then automating the process using Ansible

Manual Installation Steps

Connecting

Connecting to the EC2 instance using SSH

- Installing dependencies: Python 3.6+, Pip for Python 3, Git

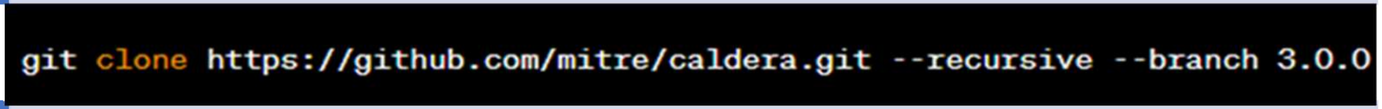


```
>ssh -i /Downloads/ansible.pem ec2-user@ec2-12-34-56-78.compute-1.amazonaws.com
```

Cloning

Cloning the CALDERA repository from GitHub

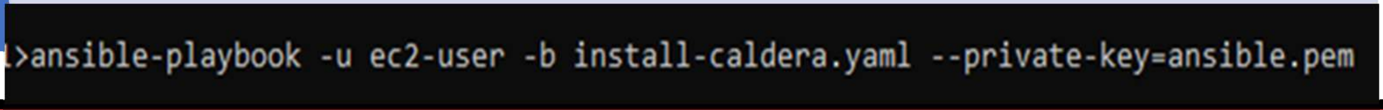
- Installing Python requirements



```
git clone https://github.com/mitre/caldera.git --recursive --branch 3.0.0
```

Starting

Starting the CALDERA server with an insecure connection



```
>ansible-playbook -u ec2-user -b install-caldera.yaml --private-key=ansible.pem
```

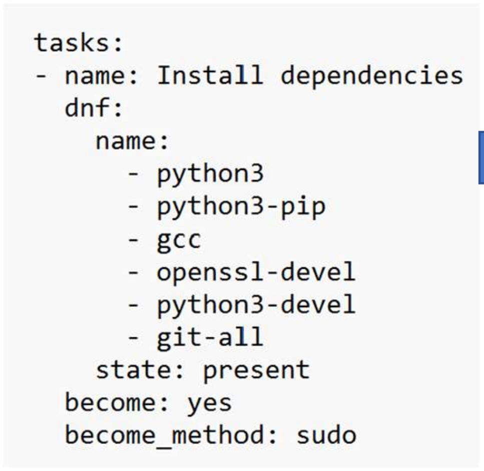
Automating Caldera Installation with Ansible

Ansible script divided into four main tasks:

- Installing dependencies
- Cloning the Caldera repository
- Installing Caldera requirements
- Starting Caldera

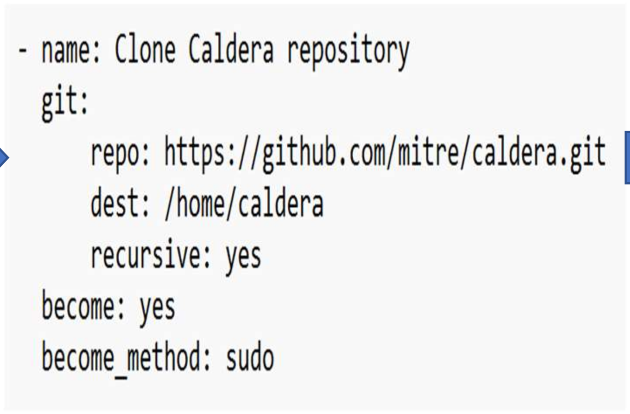
Ansible Script Implementation

Install Dependencies:



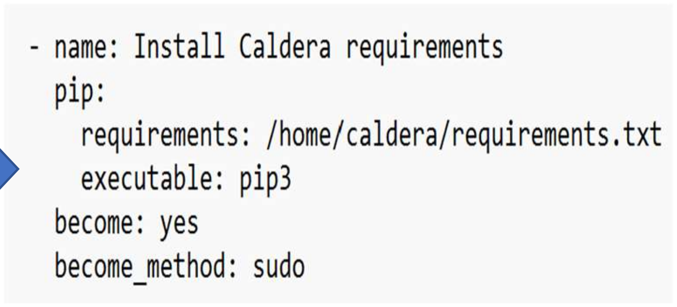
```
tasks:
- name: Install dependencies
  dnf:
    name:
      - python3
      - python3-pip
      - gcc
      - openssl-devel
      - python3-devel
      - git-all
    state: present
  become: yes
  become_method: sudo
```

Clone Caldera Repository:



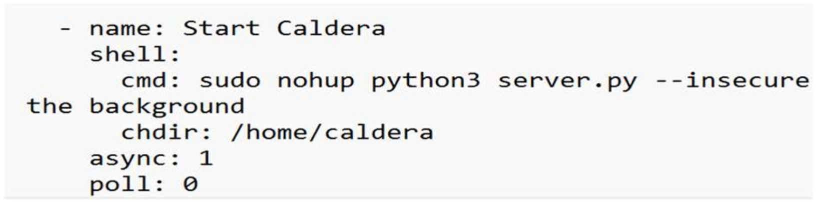
```
- name: Clone Caldera repository
  git:
     repo: https://github.com/mitre/caldera.git
     dest: /home/caldera
     recursive: yes
  become: yes
  become_method: sudo
```

Install Caldera Requirements:



```
- name: Install Caldera requirements
  pip:
    requirements: /home/caldera/requirements.txt
    executable: pip3
  become: yes
  become_method: sudo
```

Start Caldera: using shell module, sudo nohup, async, and poll options



```
  - name: Start Caldera
    shell:
      cmd: sudo nohup python3 server.py --insecure
the background
      chdir: /home/caldera
    async: 1
    poll: 0
```

Milestone 3

Create a configuration file template for Caldera using Ansible to improve security by adding user credentials.

Milestone 3 - Secure Caldera Configuration with Ansible-Managed Credentials

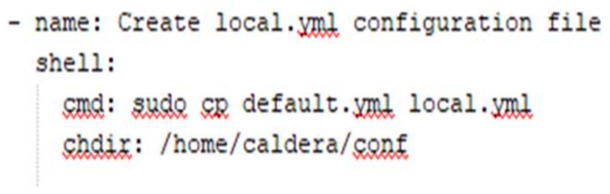
- Purpose: Improve the security of Caldera deployment by managing user credentials through Ansible
- Importance: Ensures that Caldera system access is properly controlled and secured

Modifying the Configuration File

Create File	Task 1: Create **local.yml configuration file** • Copies default.yml to create local.yml in /home/caldera/conf directory
Config. Admin	Task 2: Set **admin password in local.yml** • Replaces 'admin: admin' with 'admin: CloudProject2023'
Config. User Red	Task 3: Set **red password in local.yml** • Replaces 'red: admin' with 'red: CloudProject2023Red'
Config. User Blue	Task 4: Set **blue password in local.yml** • Replaces 'blue: admin' with 'blue: CloudProject2023Blue'

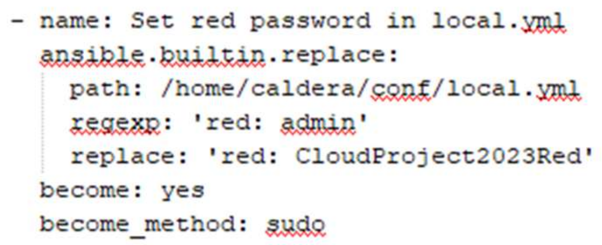
Images of Scripts

Create File



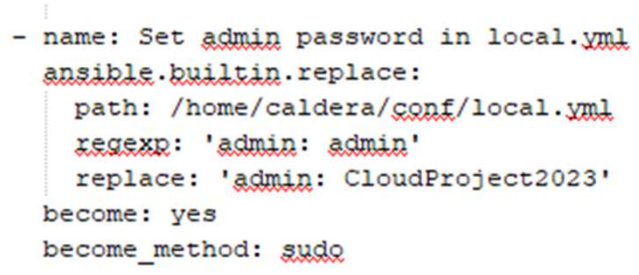
```
- name: Create local.yml configuration file
  shell:
    cmd: sudo cp default.yml local.yml
    chdir: /home/caldera/conf
```

Configure Red



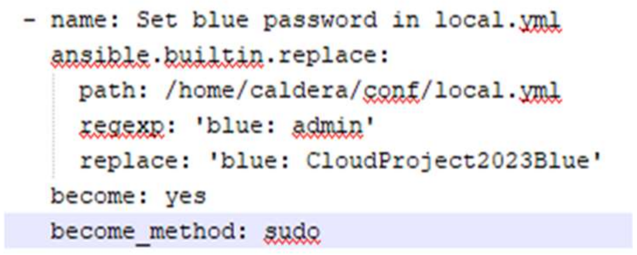
```
- name: Set red password in local.yml
  ansible.builtin.replace:
    path: /home/caldera/conf/local.yml
    regexp: 'red: admin'
    replace: 'red: CloudProject2023Red'
  become: yes
  become_method: sudo
```

Configure Admin



```
- name: Set admin password in local.yml
  ansible.builtin.replace:
    path: /home/caldera/conf/local.yml
    regexp: 'admin: admin'
    replace: 'admin: CloudProject2023'
  become: yes
  become_method: sudo
```

Configure Blue



```
- name: Set blue password in local.yml
  ansible.builtin.replace:
    path: /home/caldera/conf/local.yml
    regexp: 'blue: admin'
    replace: 'blue: CloudProject2023Blue'
  become: yes
  become_method: sudo
```

Enhanced Security and Conclusion

Impact: Updated and secure user credentials in the local.yml configuration file, increasing the overall security of Caldera deployment

Milestone 4

Set up an Ansible server with dynamic inventory to manage the deployment of Caldera.

Milestone 4 - Dynamic Inventory-Enabled Ansible Server for Caldera Management

Install Ansible on EC2 Instance

Set Up Dynamic Inventory

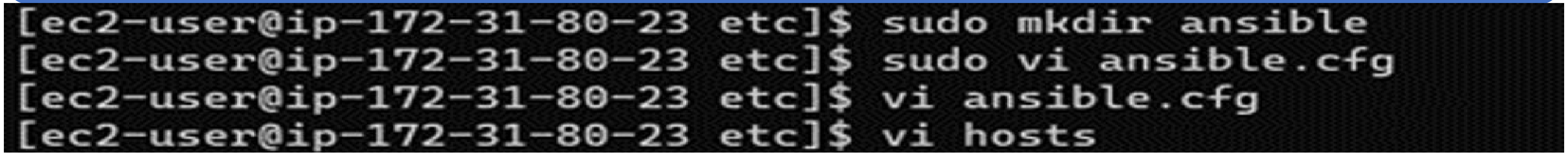
Import Necessary Files

Installed Ansible on EC2 Instance

Created an EC2 instance to host the Ansible server and enabled termination protection for continuous deployment management

Installed Ansible via SSH connection using the necessary commands following the official documentation

Addressing aws-linux yum installation limitations by manually creating ansible.cfg and host files using the CLI. Shown bellow

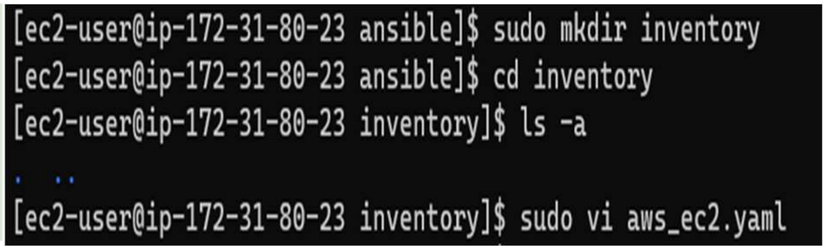


```
[ec2-user@ip-172-31-80-23 etc]$ sudo mkdir ansible
[ec2-user@ip-172-31-80-23 etc]$ sudo vi ansible.cfg
[ec2-user@ip-172-31-80-23 etc]$ vi ansible.cfg
[ec2-user@ip-172-31-80-23 etc]$ vi hosts
```

Steps to Set Up Dynamic Inventory

Create

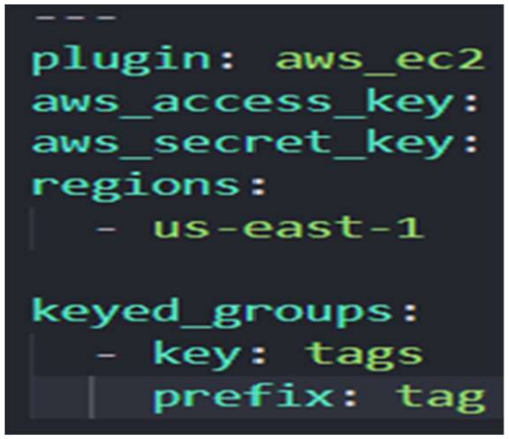
Create the aws_ec2.yaml file in a new inventory folder within the /etc/ansible directory



```
[ec2-user@ip-172-31-80-23 ansible]$ sudo mkdir inventory
[ec2-user@ip-172-31-80-23 ansible]$ cd inventory
[ec2-user@ip-172-31-80-23 inventory]$ ls -a
.  ..
[ec2-user@ip-172-31-80-23 inventory]$ sudo vi aws_ec2.yaml
```

Provide

Provide AWS access key, specify regions, and group instances by tags for targeted management

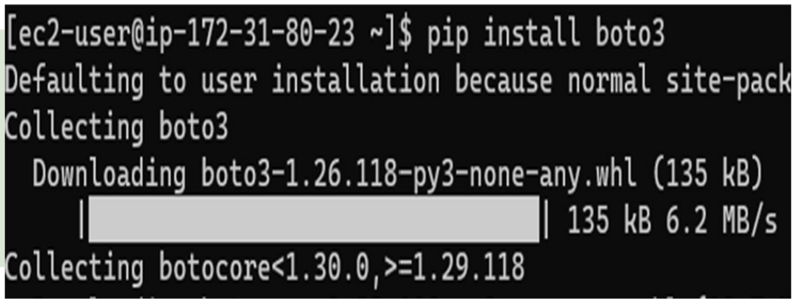


```
---
plugin: aws_ec2
aws_access_key:
aws_secret_key:
regions:
  - us-east-1

keyed_groups:
  - key: tags
    prefix: tag
```

Steps to Set Up Dynamic Inventory

Install

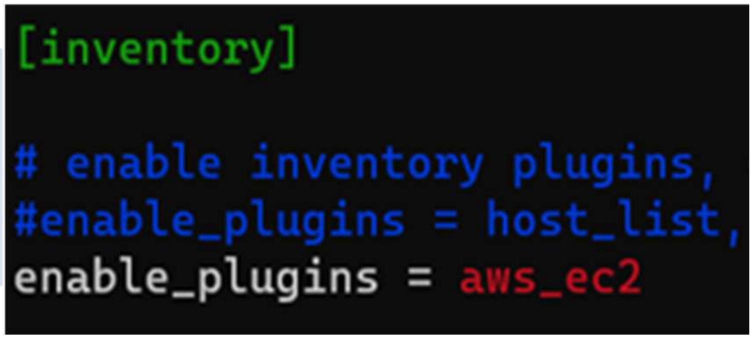
3. Install the boto3 plugin for Python using the appropriate command to access AWS services



```
[ec2-user@ip-172-31-80-23 ~]$ pip install boto3
Defaulting to user installation because normal site-pack
Collecting boto3
  Downloading boto3-1.26.118-py3-none-any.whl (135 kB)
     |████████████████████████████████| 135 kB 6.2 MB/s
Collecting botocore<1.30.0,>=1.29.118
```

Enable

4. Enable the plugin and set the default host directory in the ansible.cfg file for seamless integration



```
[inventory]

# enable inventory plugins,
#enable_plugins = host_list,
enable_plugins = aws_ec2
```

Milestone 5

Set up cloud formation script that will create ec2 instances and run the caldera-install ansible script on the ansible server to deploy caldera on the created ec2 instances

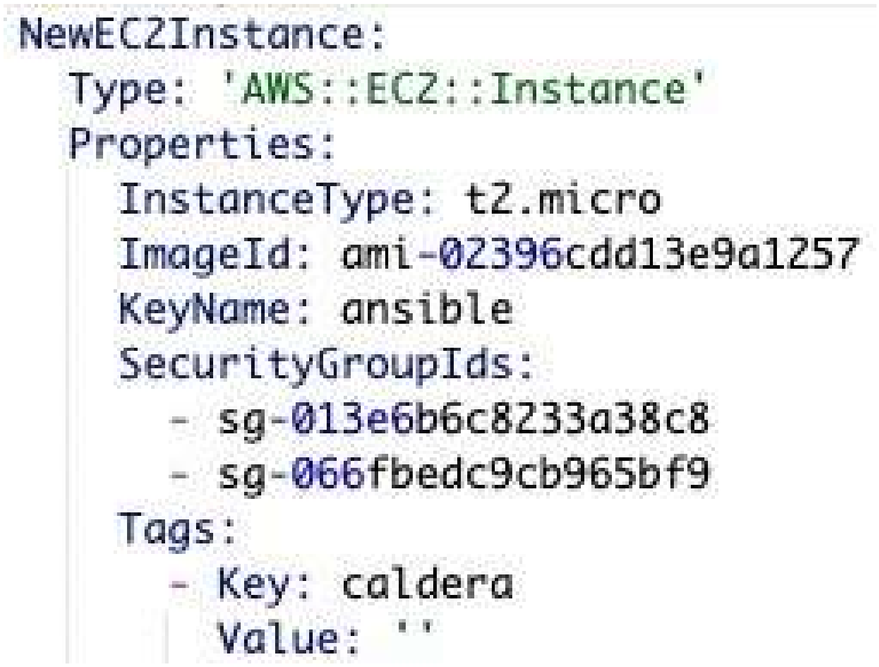
Milestone 5 - Automated Caldera Deployment on EC2 Instances via CloudFormation & Ansible

- Create Cloud Formation Script
- Setup task to create EC2 instance
- Setup task to execute a shell command

Cloud Formation Task One

The first task of our Cloud Formation script was the creation of an EC2 Instance. Some Key properties include:

- Security Groups: responsible for port 22 to allow SSH traffic for Ansible and port 8888 for Caldera.
- Tags: Using "caldera" tag so the ansible server can locate and run the playbook.

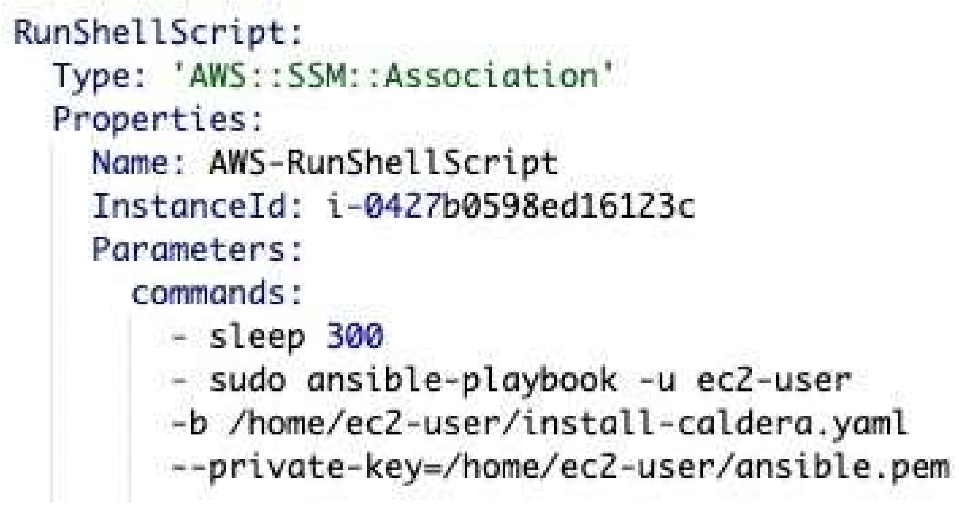


```
NewEC2Instance:
  Type: 'AWS::EC2::Instance'
  Properties:
    InstanceType: t2.micro
    ImageId: ami-02396cdd13e9a1257
    KeyName: ansible
    SecurityGroupIds:
      - sg-013e6b6c8233a38c8
      - sg-066fbedc9cb965bf9
    Tags:
      - Key: caldera
        Value: ''
```

Cloud Formation Task Two

The second task is responsible to run a shell command using Server Manager's Run a Command. Properties to note:

- SSM Command Name: AWS-RunShellScript
- Sleep 300: Utilizing a delay in the execution, allows the EC2 Instance to finish setup and initialize.
- Ansible-playbook: Executes the Ansible playbook on targeted instances, which we saw in the last slide with tag "caldera".

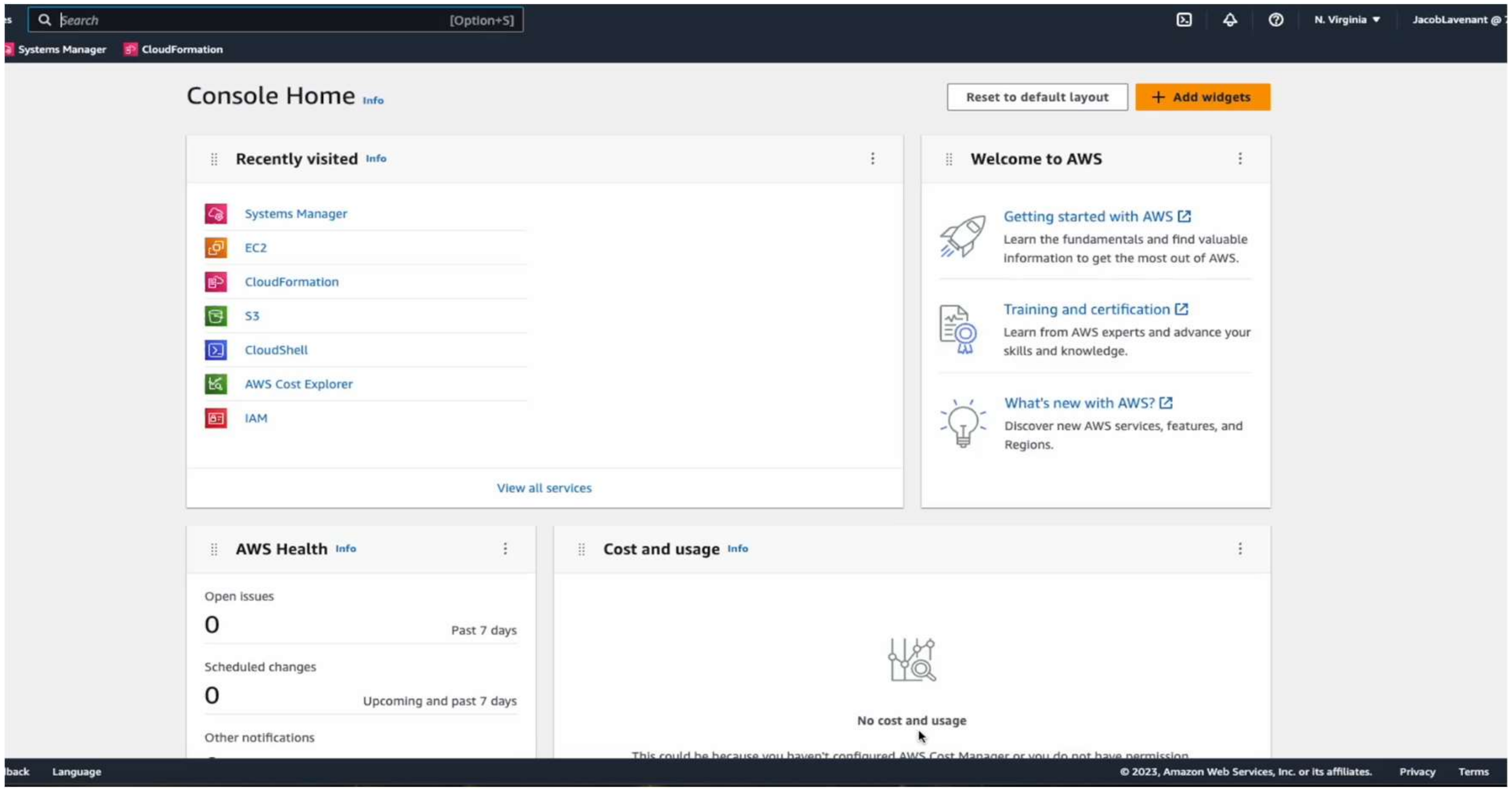


```
RunShellScript:
  Type: 'AWS::SSM::Association'
  Properties:
    Name: AWS-RunShellScript
    InstanceId: i-0427b0598ed16123c
    Parameters:
      commands:
        - sleep 300
        - sudo ansible-playbook -u ec2-user
        -b /home/ec2-user/install-caldera.yaml
        --private-key=/home/ec2-user/ansible.pem
```

Prototype and Demo

Video Demonstration

Video Demonstration of Project



Console Home Info

Recently visited Info

- Systems Manager
- EC2
- CloudFormation
- S3
- CloudShell
- AWS Cost Explorer
- IAM

View all services

Welcome to AWS

Getting started with AWS
Learn the fundamentals and find valuable information to get the most out of AWS.

Training and certification
Learn from AWS experts and advance your skills and knowledge.

What's new with AWS?
Discover new AWS services, features, and Regions.

AWS Health Info

Open issues	
0	Past 7 days
Scheduled changes	
0	Upcoming and past 7 days
Other notifications	

Cost and usage Info

No cost and usage

This could be because you haven't configured AWS Cost Manager or you do not have permission

Challenges and Limitations

Challenges

Ansible Challenges:

1. Directory Control in Ansible Scripts
   - We encountered the error "directory/file doesn't exist" frequently even though we had already created it.
2. Configuring Ansible to run on the Ansible server with AWS systems manager run command
   - We encountered the error "Ansible command not found" even though Ansible was installed

Cloud Challenges:

1. Getting the cloud formation script to run a command from Systems Manager.
   - This halted progress temporarily and made it difficult for us to get script for function

Limitations

Absence of Ansible AWS Integration

This forced us to create an Ansible server in order to integrate Ansible into AWS

Funding

Lack of funding limited our access to cheaper and less effective AWS instances

And limited how long we could leave our environments up for testing purposes

Future Work

Future Work

- **Scalability and Load Balancing:** Enhance the current infrastructure to support horizontal scaling and load balancing, ensuring Caldera can handle increased demand and distribute traffic efficiently among multiple instances.
- **Continuous Integration and Deployment (CI/CD):** Implement a CI/CD pipeline to automatically build, test, and deploy updates to the Caldera platform, ensuring a streamlined development process and reduced time to deploy new features or patches.
- **Integration with Other Security Tools:** Explore the possibility of integrating Caldera with other cybersecurity tools and platforms to provide a more comprehensive and unified security testing and defense solution.

Conclusion

Conclusion

- Successful automation of Caldera deployment and management on AWS EC2 instances using Ansible and CloudFormation
- Streamlined process, enhanced security, and improved maintainability achieved through integration
- Increased efficiency, security, and ease of management, enabling better management and monitoring of cybersecurity tools in cloud environments
- Project milestones accomplished, showcasing the potential of combining cloud automation technologies for effective deployment of cybersecurity platforms
- Contribution to the broader goal of securing and optimizing cloud-based infrastructures for modern organizations

- Our project's scripts can be open-sourced in the following repository: ccmarti1/CSS-CLOUD-PROJECT (github.com)

Resources

Resources

- *Installing ansible*. Installing Ansible - Ansible Documentation. (2023, March 30). Retrieved April 24, 2023, from https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#installing-and-upgrading-ansible
- Pejsar, R. J. (1975). *The systems manager: A study in conflict*. Amazon. Retrieved April 24, 2023, from https://docs.aws.amazon.com/systems-manager/latest/userguide/what-is-systems-manager.html
- *Welcome to caldera's documentation!¶*. Welcome to CALDERA's documentation! - caldera documentation. (n.d.). Retrieved April 24, 2023, from https://caldera.readthedocs.io/en/latest/