Build and Secure Your AWS Environment and Linux Server

1. Setting Up the Environment

Basics of AWS and set up a Linux server (EC2 instance) with security tools and Docker.

1.1 AWS Basics

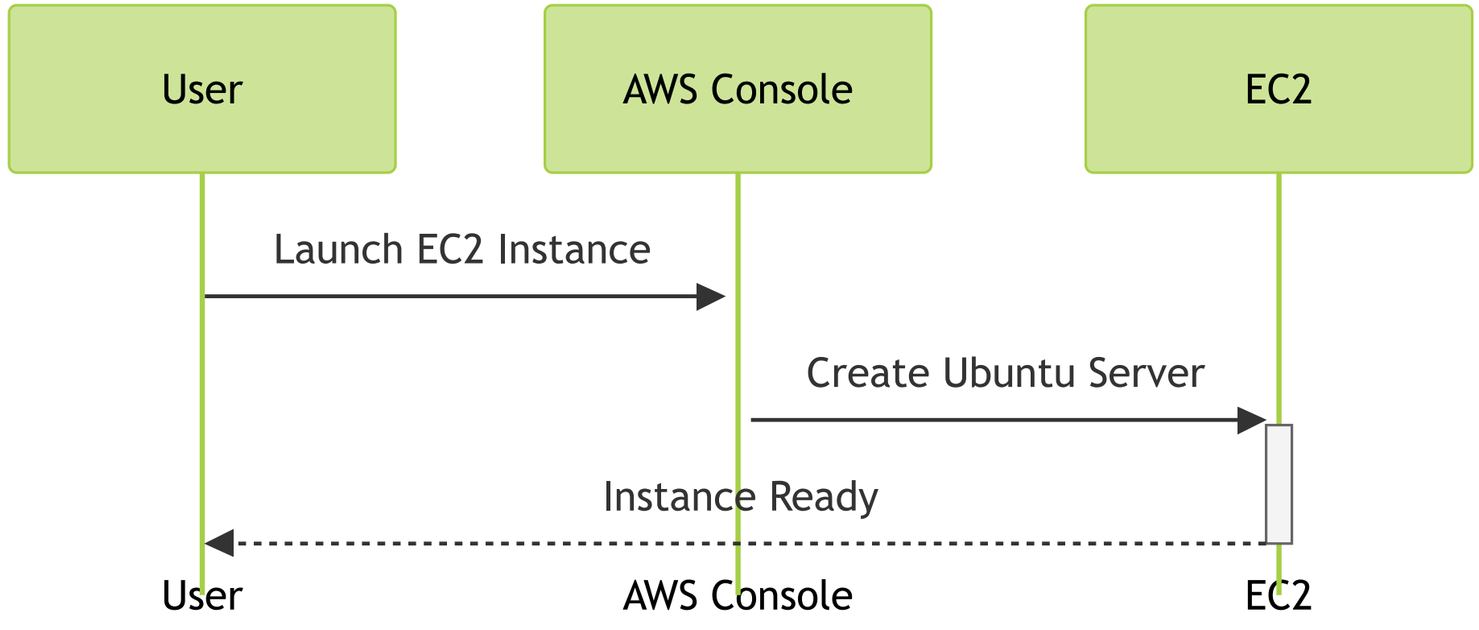
AWS provides cloud services like:

- **EC2**: Virtual servers
- **Security Groups**: Firewalls for traffic control
- **CloudTrail**: API activity logging

Reference: AWS Getting Started Guide

1.2 Launch an EC2 Instance

Set up an Ubuntu server with insecure settings for demonstration.



```
# SSH into the instance
ssh -i mykey.pem ubuntu@54.123.45.67
```

1.3 Install Security Tools and Docker

Install fail2ban, ufw, and Docker on the EC2 instance.

```
# Update packages
sudo apt update && sudo apt upgrade -y

# Install fail2ban and ufw
sudo apt install fail2ban ufw -y

# Install Docker
sudo apt install docker.io -y
sudo systemctl enable docker
sudo systemctl start docker
```

2. Assess and Harden the Linux Server

Use CIS benchmarks to assess security, then harden the server with Ansible.

2.1 Assess Security with CIS Benchmarks

Use Lynis to audit the server against CIS-like standards.

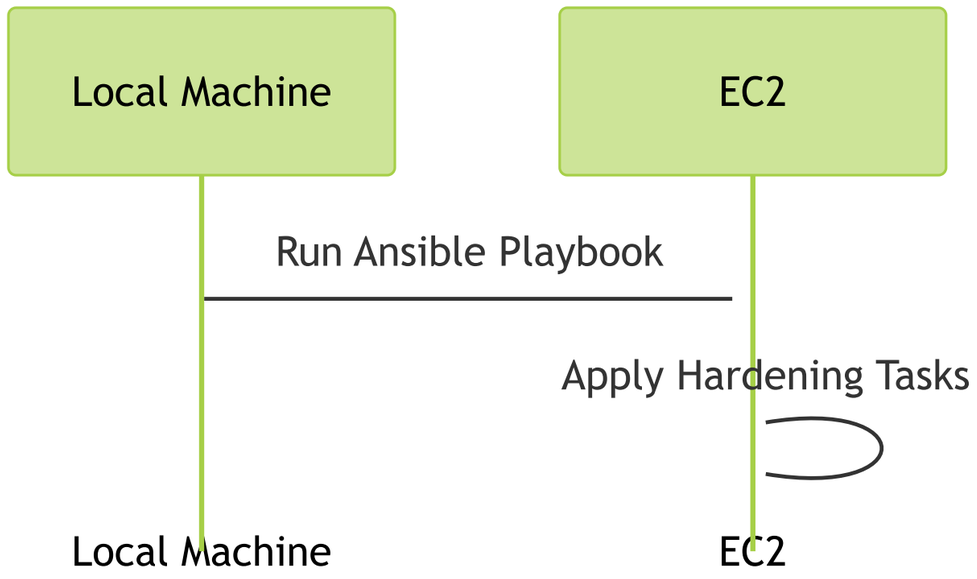
```
# Install Lynis
sudo apt install lynis aha -y

# Run assessment
sudo lynis audit system
sudo less /var/log/lynis.log
sudo lynis audit system | aha > lynis-report.html
#
sudo apt install python3-pip python3.12-venv
python3 -m venv .seas
source .seas/bin/activate

pip install prowler awscli
aws configure
prowler
```

2.2 Harden with Ansible

Write an Ansible playbook to fix identified issues.



2.3 Re-run CIS Assessment

Verify improvements by running Lynis again.

```
# Re-run Lynis
sudo lynis audit system
```

3. Assess and Secure the AWS Environment

Use Prowler to assess AWS security, then fix issues.

3.1 Assess AWS Security with Prowler

Install and run Prowler to identify security issues.

```
# Install Prowler
git clone https://github.com/toniblyx/prowler
cd prowler

# Run assessment
./prowler
```

4. Resolving Critical Network Security Issues in AWS

Issue: Critical Ports Exposed to the Internet

- **Cassandra**: Ports 7000, 7001, 7199, 9042, 9160
- **Elasticsearch/Kibana**: Ports 9200, 9300, 5601
- **Memcached**: Port 11211
- **SQL Server**: Ports 1433, 1434
- **SSH**: Port 22
- **MySQL**: Port 3306
- **MongoDB**: Ports 27017, 27018
- **Redis**: Port 6379

Exposing these critical ports publicly can lead to unauthorized access, data breaches, and potential service outages.

Resolution Steps

1. Identify security groups allowing public access to critical ports.
2. Modify AWS security groups to restrict ingress to specific IPs or remove internet accessibility entirely.
3. Validate the changes to ensure security compliance.

```
# Remove insecure ingress rule (example for port 22 SSH)
aws ec2 revoke-security-group-ingress \
    --group-id sg-0123456789abcdef0 \
    --protocol tcp \
    --port 22 \
    --cidr 0.0.0.0/0
```

Repeat this process for each critical port exposed.

Security Best Practices

- Regularly audit security groups and firewall rules.
- Use automated security scanning tools like Prowler.
- Implement Infrastructure as Code (IaC) with Terraform to maintain secure configurations.