Semester Long Internship Report

On

DevOps

Submitted by

Saumitra Patil

Under the guidance of

Mr. Lee Thomas Stephen
and
Mr. Rohan Mhatre

July 31, 2024

Contents

Chapter 1

Acknowledgment

I want to express my sincere gratitude to everyone who contributed to the successful completion of my internship. Above all, I am truly thankful to Mr. Lee Thomas Stephen and Mr. Rohan Mhatre for their invaluable guidance, support, and encouragement throughout this journey. Their expertise, patience, and willingness to share their knowledge have been vital to my learning and growth, providing me with the tools and confidence to overcome challenges and succeed in my project.

I am also grateful to the FOSSEE organization for granting me this wonderful opportunity. This experience has shaped my career aspirations and provided me with a clearer perspective on the professional landscape.

I would also like to recognize my professors and academic advisors for their ongoing support and guidance, which established the foundation for my internship. Their encouragement and advice have been invaluable throughout this journey.

Chapter 2

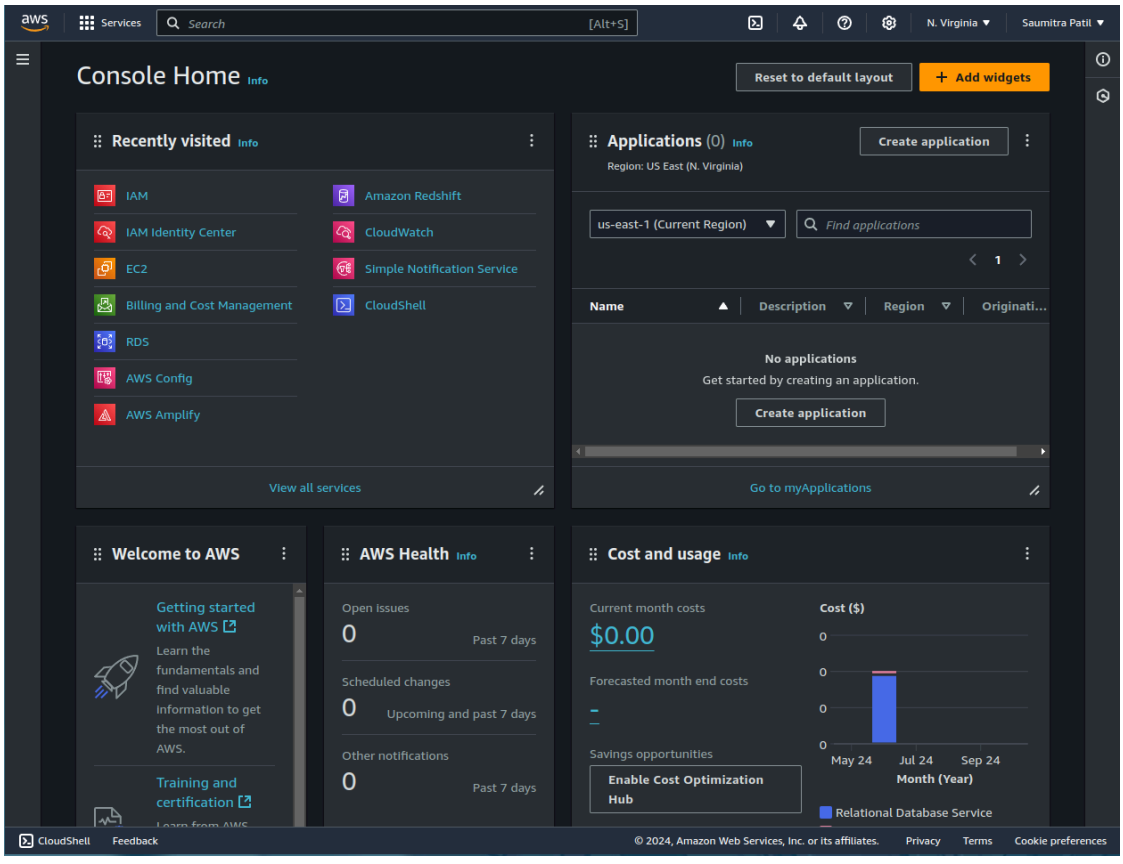
Introduction

AWS

Amazon Web Services (AWS) is a comprehensive cloud computing platform that Amazon provides. It offers a broad set of cloud-based products and services, including computing power (EC2), storage (S3), databases (RDS, DynamoDB), networking (VPC), and machine learning (SageMaker), among others. It allows organizations to deploy applications and services without needing physical servers or data centers, offering elasticity to scale resources up or down based on demand.

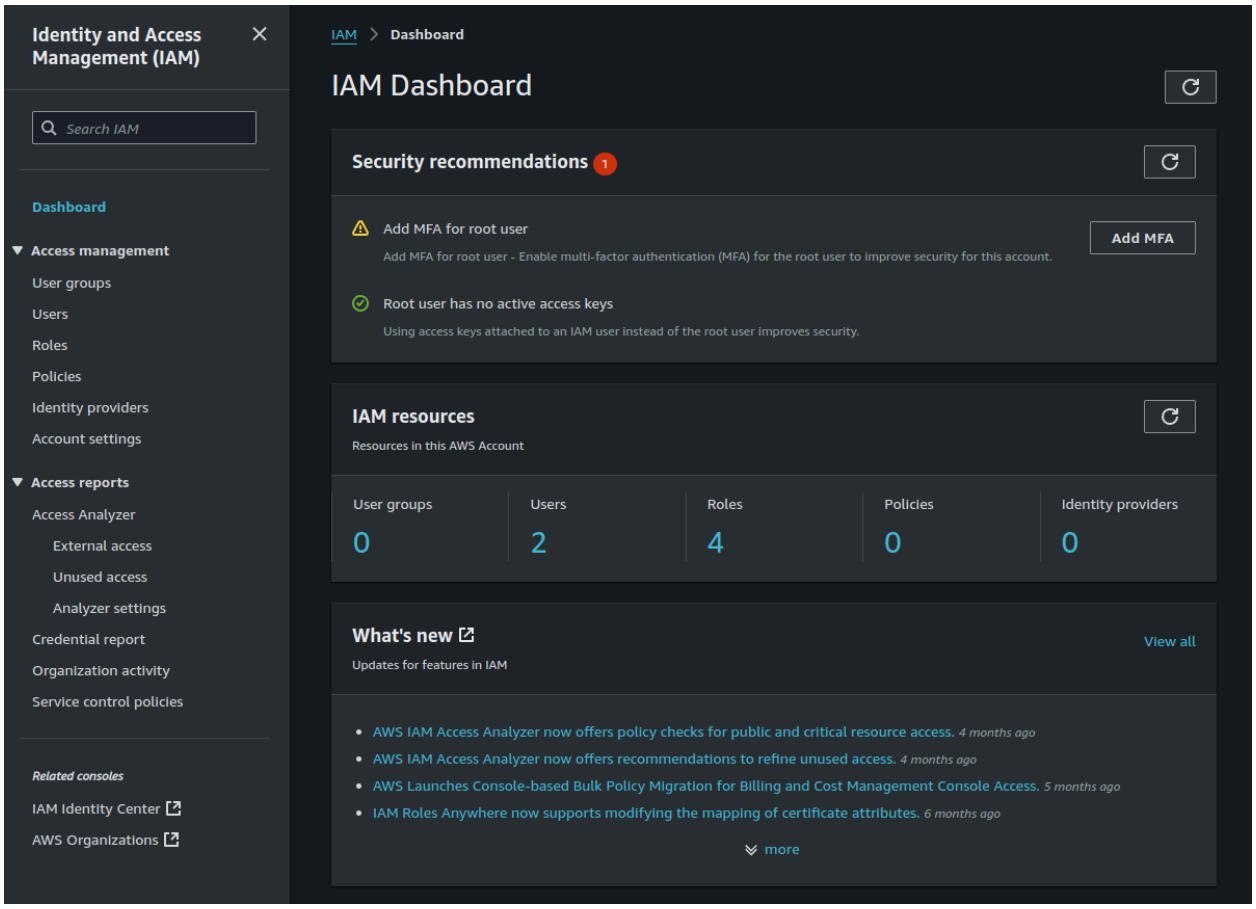
AWS operates on a global scale, with data centers (regions) around the world, ensuring high availability and low-latency access. It also provides advanced security features, compliance certifications, and identity management tools. Additionally, AWS integrates well with DevOps, MLOps, and containerization tools like Docker and Kubernetes, making it an essential platform for modern cloud-native architectures. Its pay-as-you-go pricing model ensures businesses only pay for what they use, providing cost efficiency and flexibility.

AWS is popular across industries for workloads like website hosting, big data analytics, artificial intelligence, and IoT applications.



IAM

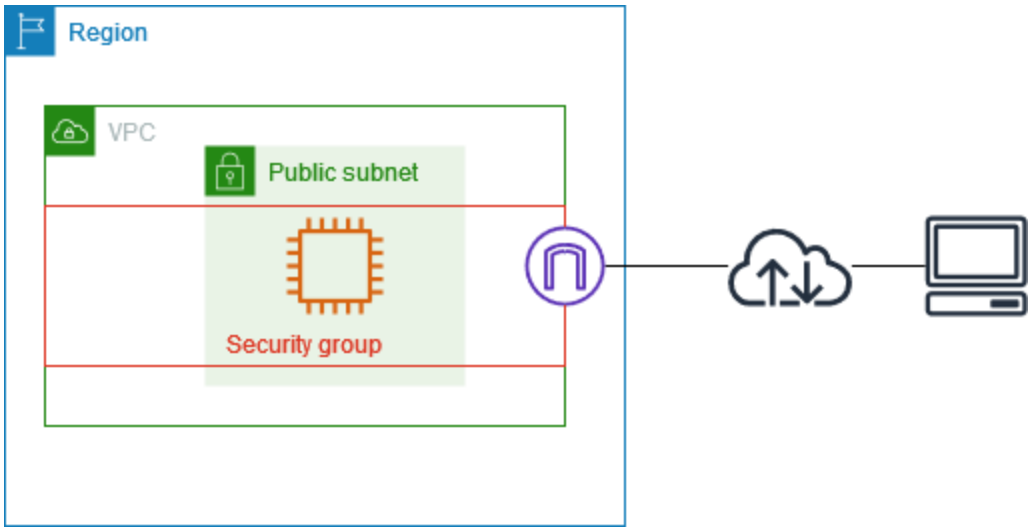
AWS Identity and Access Management (IAM) is a service that helps you securely manage access to AWS resources. It allows you to create and control permissions for users, groups, and roles, ensuring that only authorized users and services can access specific AWS resources.



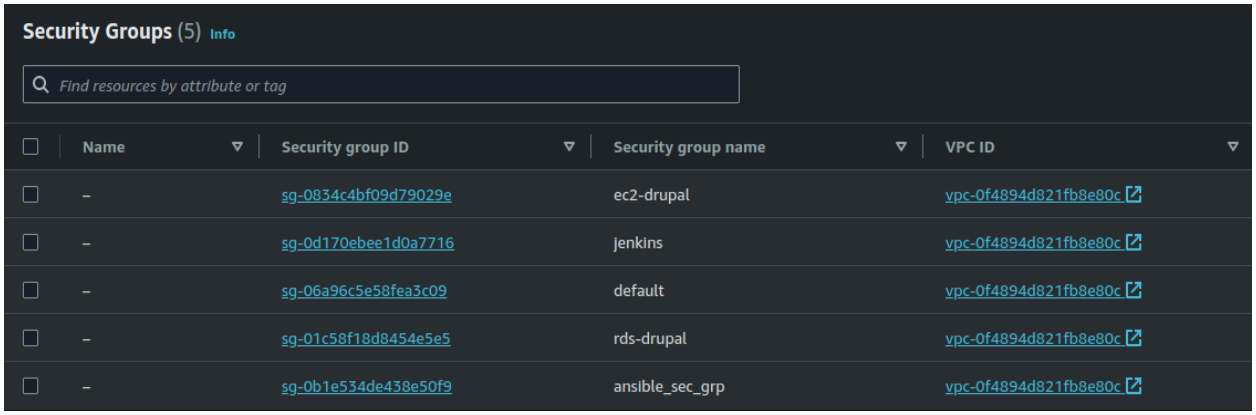
With IAM, you can define fine-grained access policies, enforce multi-factor authentication (MFA), and assign temporary credentials for tasks. This ensures secure access control and minimizes the risk of unauthorized actions, making it a crucial component for managing security in AWS environments.

Security Groups

AWS Security Groups are virtual firewalls that control inbound and outbound traffic to your Amazon EC2 instances and other resources. They help secure your cloud environment by defining rules that allow or block traffic based on IP addresses, protocols, and ports. Inbound rules specify the traffic allowed to reach your resources, while outbound rules control the traffic that can leave.



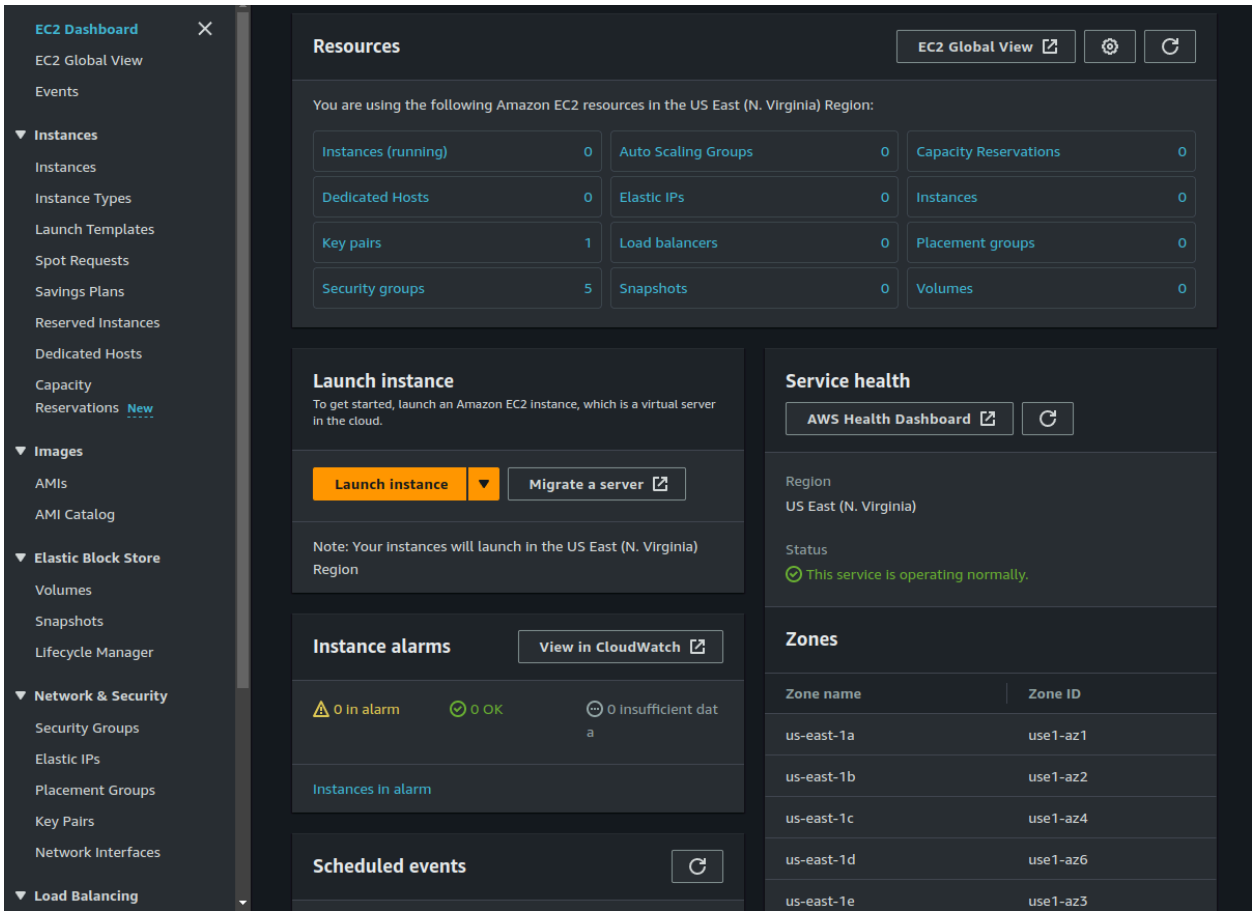
Security groups are stateful, meaning if an inbound request is allowed, the response is automatically allowed, and they can be modified at any time without interrupting the associated resources. This makes them an essential tool for managing network security in AWS.



EC2

Amazon EC2 (Elastic Compute Cloud) is a web service that provides scalable virtual servers in the cloud. It allows you to run applications and workloads by provisioning compute resources (called instances) with customizable CPU, memory, and storage configurations. EC2 offers flexible options, including on-demand, reserved, and spot instances, to suit different usage patterns and budgets.

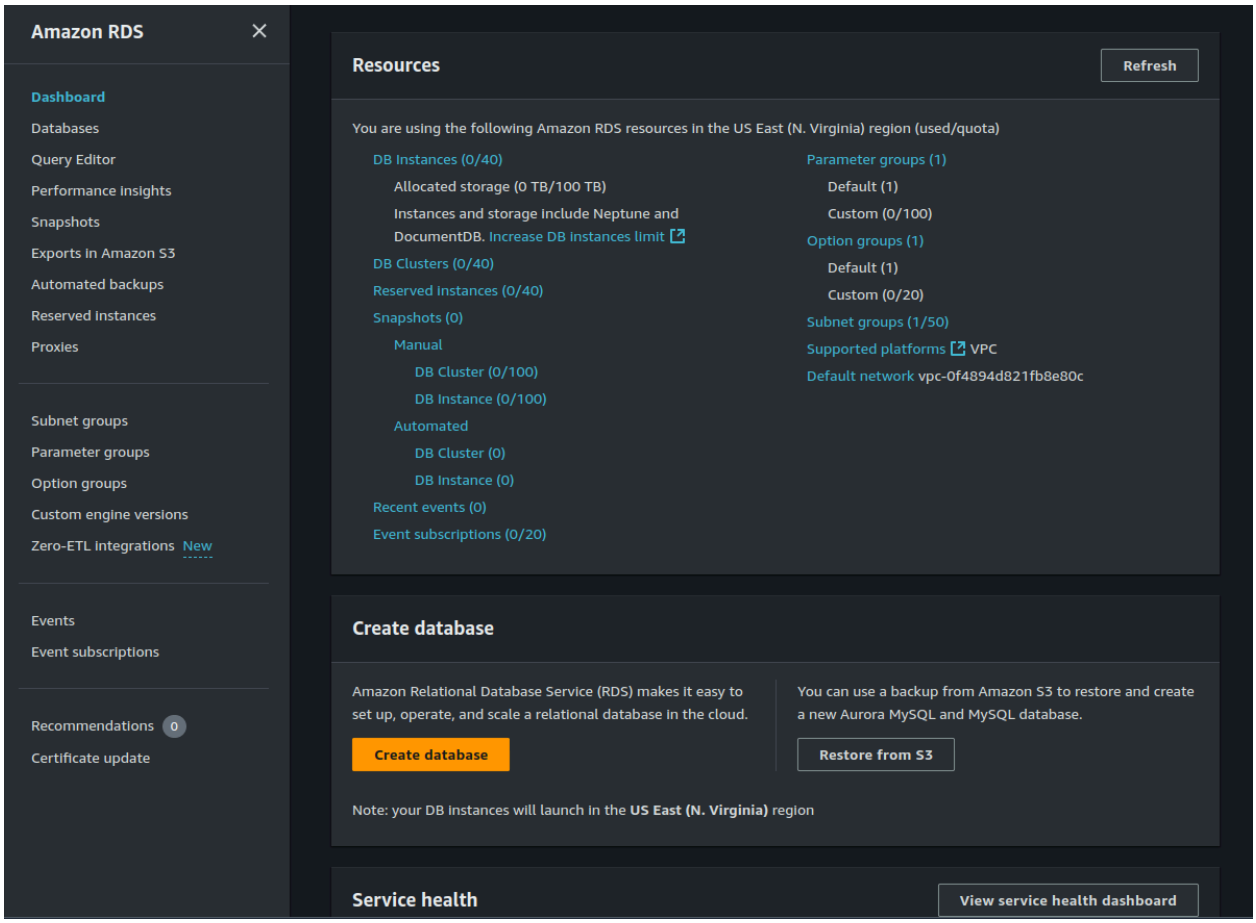
With EC2, you can easily launch, stop, or scale instances based on demand, making it ideal for hosting websites, running applications, or performing data processing. It integrates with other AWS services and provides control over your virtual servers while reducing the need for physical infrastructure.



RDS

Amazon RDS (Relational Database Service) is a managed service that simplifies setting up, operating, and scaling relational databases in the cloud. It supports multiple database engines like MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB. RDS automates routine database tasks such as backups, patching, and scaling, allowing you to focus on your application rather than database management.

RDS provides high availability, security, and automatic failover through features like Multi-AZ deployments, read replicas, and encryption. It’s ideal for use cases where you need a reliable, scalable, and fully managed relational database solution.



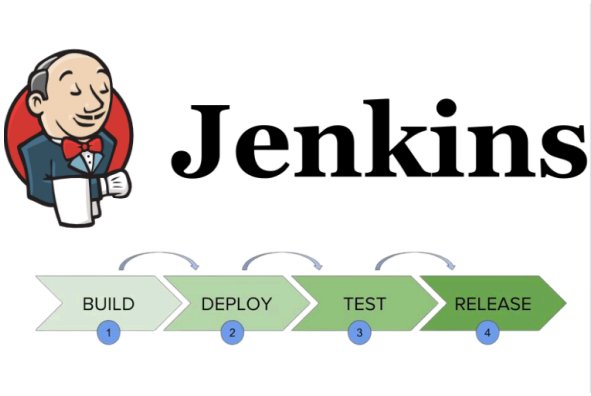
Ansible

Ansible is an open-source automation tool used for configuration management, application deployment, and task automation. It allows users to manage IT infrastructure and automate repetitive tasks across multiple systems, including cloud environments, networks, and containers. Ansible operates without needing agents on the target systems, using SSH for communication, making it lightweight and easy to deploy.

With its simple, human-readable YAML-based playbooks, Ansible enables users to define the desired state of their systems and automate the provisioning, configuration, and management of complex environments. It's widely used in DevOps to automate processes like server setup, software installation, and continuous delivery.

Jenkins

Jenkins is an open-source automation server primarily used for continuous integration (CI) and continuous delivery (CD) in software development. It automates application building, testing, and deployment, helping developers integrate code changes frequently and detect issues early. Jenkins supports various plugins, making it highly customizable for different workflows and technologies.



Chapter 3

Role Overview

1. **Learning and applying DevOps methodologies to automate software development and deployment processes:**
    - Gaining expertise in continuous integration, continuous delivery, and infrastructure as code (IaC) to automate building, testing, and deploying applications.
    - Using tools like Jenkins, GitHub Actions, and Ansible to streamline workflows, improve collaboration between development and operations, and enhance the software delivery process.
2. **Working with AWS, AlmaLinux, and AmazonLinux for hosting projects:**
    - Leveraging AWS services (e.g., EC2, RDS) for reliable hosting and using AlmaLinux and AmazonLinux for managing Linux-based servers.
    - Automating server provisioning, security, and performance management using tools like Ansible.
3. **Contributing to implementing CI/CD pipelines using tools like Jenkins and GitHub Actions:**
    - Setting up CI/CD pipelines to automate code integration, testing, and deployment, ensuring faster, more reliable software releases with minimal manual intervention.
4. **Gaining experience with project-specific DevOps tools and technologies:**
    - Working with cloud platforms (AWS, DigitalOcean), using Ansible for configuration management, Git for version control, and Nginx for web traffic management.
    - Continuously learning new tools to improve DevOps processes and infrastructure management.

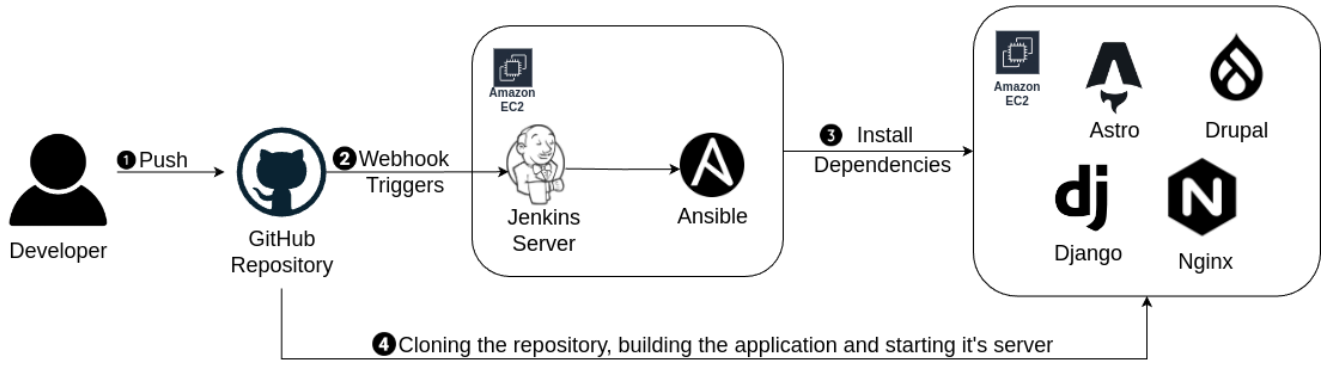


Figure 1. The basic flow for deployment

Chapter 4

Projects

Alma-Ansible-Vagrant:

- **Source:** [GitHub - FOSSEE-Intern/Alma-Ansible-Vagrant]
- **Project Title:** Alma-Ansible-Vagrant
- **Description:** This project sets up a Continuous Integration/Continuous Deployment (CI/CD) pipeline using Jenkins to automate the deployment of a Django application on AWS EC2 instances. It integrates Jenkins with GitHub and Ansible to streamline the process of creating EC2 instances and deploying the Django application.
- **DevOps Principles Applied:** Infrastructure as code (IaC) with Jenkins, CI using Jenkins and GitHub, and CD to AWS EC2 instances running AlmaLinux.
- **Challenges:**
    - Learning Ansible syntax and modules.
    - Ensuring playbooks were idempotent and reusable.
    - Learning Jenkins and its integration with AWS and Ansible.
    - Ensuring that the environmental variables are stored safely.
    - Learning Nginx.
- **Solutions:**
    - Used **shell scripts** and **systemd** services with playbooks such that they become idempotent.
    - Used Jenkins inbuilt environment variable option to store environment variables.
- **Learnings:** Gained experience with Ansible playbooks, modules, and inventories. Understood the benefits of IaC tools like Jenkins and GitHub Actions for managing server configurations consistently and efficiently.

```yaml
tasks:
  - name: Create and start an EC2 instance
    amazon.aws.ec2_instance:
      name: "managed-node"
      key_name: "{{ key_name }}"
      instance_type: "{{ instance_type }}"
      region: "{{ aws_region }}"
      security_group: "{{ sec_grp }}"
      network:
        assign_public_ip: true
      access_key: "{{ aws_access_key_id }}"
      secret_key: "{{ aws_secret_access_key }}"
      image_id: "{{ aws_ami }}"
      state: "running"
      tags:
        Projects: webserver
```

```yaml
- hosts: aws_ec2
  gather_facts: no
  become: yes
  remote_user: ec2-user

  tasks:
    - name: install python3 and nginx
      yum:
        name:
          - git
          - python3
          - python3-pip
          - nginx
        state: present

    - name: clone repo
      git:
        repo: 'https://github.com/saumitrapatil/sysad_intern.git'
        dest: /srv/checkout/ansible-nginx-gunicorn
        clone: yes
        update: yes
```

astro-ansible-aws:

- **Source:** [GitHub - FOSSEE-Intern/astro-ansible-aws]
- **Project Title:** astro-ansible-aws
- **Description:** This project sets up a fully automated Jenkins and Ansible pipeline to deploy an Astro website on AWS. The process is initiated when Jenkins detects new commits via a GitHub webhook. Jenkins triggers Ansible playbooks, ec2.create.yml to spin up an EC2 instance and deploy.yml to clone the Astro website from GitHub, install dependencies (npm, Node.js, nginx, git), build the website, and configure nginx.
- **DevOps Principles Applied:** Infrastructure as code (IaC) with Jenkins, CI using Jenkins and GitHub, and CD to AWS EC2 instances running AlmaLinux.
- **Challenges:**
    - Ensuring playbooks were idempotent and reusable.
    - Learning Jenkins and its integration with AWS and Ansible.
    - Ensuring that the environmental variables are stored safely.
    - Learning Nginx.
- **Solutions:**
    - Used **shell scripts** and **systemd** services with playbooks such that they become idempotent.
    - Used Jenkins inbuilt environment variable option to store environment variables.
- **Learnings:** Through this project, I gained hands-on experience with Jenkins and Ansible for setting up automated CI/CD pipelines, including deploying and managing AWS infrastructure. Additionally, I developed skills in scripting, version control, troubleshooting, and workflow automation.

```yaml
- name: clone repo
  git:
    repo: 'https://github.com/FOSSEE/animate2024.git'
    dest: /srv/animate2024
    clone: yes
    update: yes

- name: build astro website
  shell: |
    npm ci
    npm run build
  args:
    chdir: /srv/animate2024

- name: create nginx sites-enabled folder
  command:
    cmd: mkdir /etc/nginx/sites-enabled
    creates: /etc/nginx/sites-enabled
```

drupal-mariadb-aws-ansible:

- **Source:** GitHub - FOSSEE-Intern/drupal-mariadb-aws-ansible
- **Project Title:** drupal-mariadb-aws-ansible
- **Description:** This project automates the deployment of a Drupal website on an Amazon EC2 instance using Jenkins and Ansible. Jenkins is configured on an EC2 instance with a GitHub repository linked via a webhook. Two Ansible playbooks are used: ec2.create.yml provisions an EC2 instance for deployment, while deploy.yml installs required dependencies such as PHP, Nginx, and Git, sets up Composer, and creates a Drupal project directory. It also configures Nginx using a template and manages Nginx and PHP-FPM services. The pipeline is triggered by GitHub webhooks, providing a seamless and efficient deployment process.
- **DevOps Principles Applied:** Infrastructure as code (IaC) with Jenkins, CI using Jenkins and GitHub, and CD to AWS EC2 instances running *AlmaLinux* with AWS RDS running *MariaDB*.
- **Challenges:**
    - Ensuring composer was installed in a directory where it is accessible.
    - Ensuring the Drupal directory exists and is empty.
    - Ensuring the php-fpm and nginx services are reloaded properly.
- **Solutions:**
    - Using the composer module with
    - Using a port for php-fpm instead of a socket.
- **Learnings:** I learned how to set up a Jenkins server on an EC2 instance and integrate it with GitHub for automated deployment. Configuring Jenkins to run Ansible playbooks taught me to automate the creation of EC2 and RDS instances and deploy a Drupal

website. Managing AWS credentials as environment variables in Jenkins was a practical lesson in a secure setup. Setting up GitHub webhooks to trigger Jenkins builds helped me understand how to automate deployment processes efficiently. Overall, this project improved my skills in Jenkins, Ansible, and AWS.

```yaml
tasks:
  - name: Create and start a DB instance in RDS
    amazon.aws.rds_instance:
      engine: "{{ engine}}"
      db_instance_identifier: "{{ db_identifier }}"
      db_instance_class: "{{ instance_class}}"
      allocated_storage: "{{ storage }}"
      publicly_accessible: false
      storage_encrypted: true
      vpc_security_group_ids: "{{ rds_sec_grp }}"
      state: running

      master_username: "{{ username }}"
      master_user_password: "{{ password }}"
      db_name: "{{ db_name }}"
    register: rds_info
```

```yaml
vars:
  rds_info: "{{ lookup('file', 'JSON/out.rds.create.json') | from_json }}"
  db_url: "{{ rds_info | json_query('endpoint.address') }}"

tasks:
  - name: Install php and other dependencies
    dnf:
      name:
        - git
        - php
        - php-fpm
        - php-gd
        - php-mysqlnd
        - nginx
      state: present

  - name: Download Composer installer.
    get_url:
      url: https://getcomposer.org/installer
      dest: /tmp/composer-installer.php
      mode: 0755
```

drupal-mariadb-aws-ansible-update:

- **Source:** [GitHub - FOSSEE-Intern/drupal-mariadb-aws-ansible-update]
- **Project Title:** drupal-mariadb-aws-ansible-update
- **Description:** This script is meant to be run on a previously setup Drupal environment, which will update Drupal to its latest version as well as update all its modules.
- **DevOps Principles Applied:** Infrastructure as code (IaC) with Jenkins Ansible to update Drupal and the modules used in the website.
- **Learnings:** This project taught me how to set up a Jenkins server on EC2, integrate it with GitHub, and automate the deployment of a Drupal website using Ansible playbooks. I also learned how to manage AWS credentials securely in Jenkins and use GitHub webhooks to trigger automated builds.

```
- name: Update drupal with it's dependencies
  composer:
    command: update
    arguments: "drupal/core-* --with-all-dependencies"
    working_dir: "{{ drupal_site_path }}"
  when: drupal_composer_json.stat.exists

- name: Start nginx
  systemd_service:
    name: php-fpm.service
    state: reloaded

- name: Start nginx
  systemd_service:
    name: nginx.service
    state: reloaded
```

Chapter 5

Learnings

Technical Skills Developed

- Ansible:
    - Configuration management.
    - Writing automation scripts using various modules.
    - Dynamic inventory using `aws_ec2` plugin.
    - Using automation script to manage AWS EC2 and RDS instances.
- Vagrant:
    - Setting up a local environment for testing.
    - Manage the firewall to allow traffic through the host machine.
    - Provisioning Ansible playbooks directly through Vagrantfile.
- AWS:
    - Working with EC2 instances.
    - Working with RDS instances.

    - Managing security groups for incoming and outgoing traffic as well as connecting EC2 instances with RDS.
    - Creating billing alerts.
- Jenkins:
    - Creating a pipeline for the deployment of Django applications, building and deploying Astro website, and installing and updating Drupal websites.
    - Use of environment variables in Jenkins to securely store sensitive information.
- Linux:
    - Features and usage of RHEL9, AlmaLinux specifically.

Soft Skills Developed

- Explaining and reporting my work:
    - Gaining the ability to clearly communicate complex technical concepts to both technical and non-technical team members.
    - Creating detailed reports and documentation to showcase progress, challenges, and solutions in a structured manner.
    - Enhancing collaboration through effective verbal and written communication, ensuring stakeholders are informed about project status and decisions.
- Learning new tools and integrating them with one another:
    - Quickly adapting to and mastering new tools and technologies relevant to DevOps and automation.
    - Demonstrating proficiency in integrating different tools and platforms to create seamless workflows, improving efficiency and productivity.
    - Continuously staying updated with emerging trends, enhancing the ability to implement modern practices and solutions across projects.

Chapter 6

Conclusion

My internship at The FOSSEE provided an invaluable learning experience in DevOps. During the internship, I gained hands-on experience with key DevOps tools and methodologies, including Ansible for automation and configuration management. I further developed my technical skills by working on real-world projects applying DevOps practices like continuous integration and continuous deployment.

In addition to my technical growth, I honed my soft skills, such as communication and teamwork, which were crucial for effectively collaborating with team members in a fast-paced environment. These experiences not only deepened my understanding of DevOps but also prepared me for future challenges, making me more confident in applying these skills in professional settings. The technical and collaborative abilities I developed will significantly enhance my career prospects.