International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

An AWS-Based DevOps Automation Framework for Efficient Software Deployment

Vidushi
M. Tech Scholar, Ganga Institute of Technology & Management, Jhajjar, India

Dr. Priyanka Rani
Assistant Professor, Ganga Institute of Technology & Management, Jhajjar, India

***Abstract*—This paper aims to design and implement an AWS DevOps framework. DevOps is the process of developers and IT teams working together, and this research focuses on identifying cost-efficient solutions and deployment time in DevOps involving the AWS cloud platform. In today's software development, quick and secure deployment is very important. However, many companies still face issues due to manual work and human errors, and slow deployment. This paper suggests an AWS-based DevOps automation framework that reduces human effort and improves performance. The framework uses tools such as Docker for containers, GitHub for version control, and Amazon Web Services to run the processes of building, testing, and deploying applications. This system is designed using key AWS services such as Elastic Compute Cloud (EC2) for computing resources, Elastic Load Balancer (ELB) for distributing incoming traffic, and Virtual Private Cloud (VPC) for creating a private network in AWS. Software developers and IT operations specialists are helped to work together, communicate, and connect using DevOps. The results show that automation saves time, reduces errors, and provides stable performance. Companies can deliver applications faster and more reliably using this method. Future improvements may include AI-based monitoring and multi-cloud support. This framework is highly useful for today's cloud-based applications.**

Keywords— DevOps, AWS, CI/CD, Automation, Docker, Ansible

I. Introduction

Today, companies want fast software and servers because of the digital world, and there is high competition, which is why cloud computing is important. Earlier, companies used to buy and manage their own servers, and manual installation was very slow; the cost efficiency was high. But now they use internet-based servers.

Cloud computing provides resources like servers, storage, and databases over the internet, which is easier to run and manage software. A good example is AWS of cloud computing. Services include EC2(Virtual servers), RDS(Database), S3(Storage).

EC2 (Elastic Compute Cloud) is used to create virtual servers where the application is deployed and tested. It allows us to use a virtual computer in the cloud. These computers are called instances. We can use them like our own computer online. We can install programs, run apps, and save data on them. EC2 gives flexible power. We can choose small or big servers depending on your needs. Companies don't need to buy real servers. EC2 also allows scaling. This means we can make it bigger or smaller when traffic changes.

ELB (Elastic Load Balancer) is a tool from AWS. It shares website or app traffic to many servers. If many people visit at the same time, ELB divides the traffic. This keeps the server fast and stops it from crashing. ELB also checks if a server is not working. It sends traffic to other good servers. This makes the app stable and fast. ELB works automatically. You don't need to manage it manually. It is very useful for apps or websites that have many users and need to manage them manually. It is very useful for apps or websites that have many users and need to work all the time.

S3 (Simple Storage Service) is a storage service by AWS. It allows us to save files like photos, videos, documents, or backups in the cloud. We can get your files anytime from anywhere using the internet. S3 keeps multiple copies of your data, so we don't lose it. We only pay for the space we use. We can share files with others or keep them private. Many websites and apps use S3 to store data. It is fast, safe, and easy to use.

IAM (Identity and Access Management) is a service by AWS. We can use it to control who can access your AWS resources. We can create users, groups, and roles. We can give them only the permission they need. This keeps your data safe. IAM also gives extra security with passwords or codes. It stops people who are not allowed from using our resources. We can use IAM to manage teams, projects, and security easily. It is very important to keep the cloud system safe.

Virtual Private Cloud (VPC) is a service provided by AWS that allows us to create a private network within the cloud. It is like having your own small, safe space in the cloud, separated from other people. We can put our servers, databases, and applications inside this space. We can also control things like IP addresses, subnets, and who can access your network. VPC makes your cloud resources more secure and organized. We can connect it to the internet or your company's own network safely. Using a VPC helps developers and the IT team keep applications protected while working in the cloud.

But now, DevOps in AWS helps us save time because instant deployment is now possible, but before, there was a problem with manual installation and time-consuming setup of servers and software. It means cloud computing is fast, flexible, and cheap. DevOps is a process in which developers and IT teams work together, and DevOps has core principles: Culture, Automation, Measurement, Sharing (CAMS). There are many challenges to DevOps, there is a shortage of technical skills, people are afraid to adopt new processes, and difficulty

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

in selecting tools & infrastructure.

Cloud computing also offers several key advantages. Scalability allows a company to adjust resources based on demand. Cost efficiency there is no upfront investment, only pay for what we use. Automation reduces manual updates and backups; these are performed automatically. Additionally, global access ensures that the application and data can be accessed from any location.

GitHub is used to save my code in one place. It helps me keep track of the changes I make. I can go back if something goes wrong. I can also work with my team easily. Everyone can see the latest version of the code. I can make branches to try new ideas without breaking the main code. GitHub helps me share my work and get feedback. It makes teamwork easier. Using GitHub, I don't lose my code, and I can work from anywhere. It is very important for DevOps and coding projects.

Jenkins is used to build and test my application automatically. It saves me from doing work manually. I can set up pipelines that run whenever I change my code. Jenkins tells me if my code has problems. I can test software before it goes live. It helps my team work faster and avoid mistakes. Jenkins works with other tools like GitHub and Docker. It is very helpful for DevOps automation and continuous integration.

CI/CD is used in DevOps and AWS to automate the process of software development. It helps in increasing the speed, stability, and security of modern software development. CI helps in merging the code continuously, and CD is used to deploy the code automatically. There is a continuous cycle: first, write a code; second, test the code; third, monitor the code; fourth, deploy the code, which means the software automatically updates repeatedly. By using this, all processes are done automatically.

Docker is used to run applications in containers. A container is like a small package with everything my app needs. Docker helps me avoid problems when moving apps from one place to another. I can create, run, and delete containers easily. The team can use the same container, so our apps work the same way. It is very important for testing and deploying software in a safe environment.

In the future, cloud computing will continue to grow as more, and now many companies are adopting it.

II. Literature Review

Many researchers have worked on DevOps and Cloud technologies, and their different aspects. One study focused on CI/CD pipelines, where the main goal of DevOps is to deliver software fast and with high quality, which is why we use a CI/CD pipeline. CI (Continuous Integration) is a process in which developer repeatedly merges their code into a shared repository, helping detect errors at an early stage and ensuring that new code does not break existing code. CD (Continuous Delivery) is the next step, where the code is automatically built, tested, and deployed. This process makes software delivery faster, more reliable, and efficient. In DevOps, multiple teams are involved: QA (Quality Assurance), Security, and IT, which work together to ensure continuous software delivery. In addition to CI/CD, Continuous Testing (CT) and Continuous Monitoring (CM) are integrated to ensure software and system quality and performance.[1]

In recent years, it is important to select the right EC2 instance because costs are high and workloads like AI are increasing. Different instance types are designed for different tasks, so performance depends on selecting the correct instance. The right choice helps in getting better performance at a lower cost, but a wrong choice can cause issues like high resource wastage and downtime. If we choose the correct instance, then we can use the cloud in a better way and save money.[2]

This study focuses on improving CI (Continuous Integration) using Docker containers, which make deployment and testing easier. The role of Docker is to package an application into a container, which means the application and its dependencies are bundled together so it can run anywhere.[3]

Banerjee and Mondal (2026) analyze the role of bash scripting in modern DevOps. Their study highlights that shell scripting acts as a glue, which means they are used to connect different tools. The author explains that shell scripting is widely used for automating system tasks such as user management, backup operations, and log analysis. It reduces manual effort and minimizes human errors. Also, the study focuses on the importance of shell scripting in CI/CD pipelines, where it is used to execute commands, run tests, build applications, and deploy software. Shell scripting is highly suitable for pipeline environments. The paper also discusses its role in container technologies like Docker and Kubernetes, where shell scripts are used for container setup and health checks. But shell scripting has some limits. It is not good for complex tasks, big data handling, or large applications. In these cases, languages like Python are better. The study says shell scripting is important for DevOps engineers. It helps in automating work, making the system better, and connecting old and new systems.[4]

Pranav et al. (2021) explained that Ansible is used in DevOps to automate server management tasks. Ansible uses simple YAML playbooks, which are easy to read and write. It is agentless, so no extra software is needed on the systems. It helps the application to deploy quickly. This makes it very easy to create, read, and understand this instruction for automation. Using Ansible, teams can deploy applications quickly and reliably, ensuring that software is installed across all services. It also helps us to reduce errors caused by manual operations and improve efficiency in IT.[5]

Containerization is an important concept in DevOps. According to Docker documentation and The Docker Book by Turnbull (2014), Docker allows developers to package applications with all dependencies into containers. These containers can run on any system without changing the configuration. This helps to avoid errors and ensure consistency in the development and production environments [13],[19]. Studies also show that containers improve deployment reliability and reduce system failures [37].

Monitoring and security are also an important part of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

DevOps. Research on continuous monitoring (2023) shows that monitoring tools help detect errors early and improve system performance [17]. Security studies have shown that CI/CD pipelines must include secure practices like access control, encryption, and proper authentication to protect cloud systems [33] [35].

This paper focuses on deploying Deep Learning models using MLOps, AWS, and DevOps. By using Docker, they create custom images containing Python. A flash backend server takes user input and provides it from the model. The infrastructure is deployed on AWS using Terraform, ECS, and a load balancer for deployment.[6]

III. Problem Statement

Today, companies want to deliver software fast and without mistakes because there is high competition in the digital world. But many companies still face problems. They do a lot of work manually, which is slow and can be caused by human errors. Setting up servers and installing software manually takes a lot of time and increases costs. Cloud computing, like Amazon Web Services, helps to solve some of these problems. AWS provides virtual servers (EC2), storage (S3), and a database (RDS) that can be used anytime and scaled according to need. Even with cloud computing, it is important to choose the right servers or instances. The wrong choice can waste money and reduce performance.

DevOps practices help to make software deployment faster and more reliable. Tools like CI/CD pipelines, Docker containers, Bash scripting, and Ansible automate many tasks. CI/CD pipelines automatically integrate, test, and deploy code. A Docker application with all its requirements so they can run anywhere. Bash scripts automate system tasks, but are not always good for big or complex tasks. Ansible helps to manage servers automatically without installing extra software.

Even with these tools, many companies find it hard to combine everything in one system. They need a framework that uses cloud resources, automation tools, containers, and CI/CD pipelines together. Such a framework can save time, reduce errors, use resources efficiently, and help developers and the IT team work together. This will allow companies to deliver software faster, safer, and more reliably, improving overall performance and saving money.

If cloud and DevOps tools are not set up correctly, data can be unsafe, and rules may not be followed.

Automation tools make work easier, but the team must know how to use them. If the team is not trained, automation will not work properly.

Nowadays, many tasks are done automatically, but creating and deploying machine models is still hard and time-consuming. Normal methods need a lot of work, like writing code, training the model, testing it, and putting it on the server. It is also difficult to keep the model running and give reliable results to the user. We need a system that can automatically set up, run, and manage models.

Also, checking and maintaining applications is a big problem for many companies. After deployment, it is important to see if the application is working correctly or not. In manual systems, people check everything, so they can miss errors and problems. If any issues happen, it takes more time to find and fix them. This can make users unhappy and also affect the company's image.

Another problem is when the number of users increases. Without automation, it is hard to increase or decrease server resources quickly. This can make the server slow or waste extra resources. Security is also very important in a cloud system. If security settings are not done properly, data can be unsafe, and anyone can access it without permission.

Using many tools together is also difficult. Tools like Docker, Ansible, and CI/CD need a correct setup to work properly. If they are not connected in the right way, the system may not work as expected.

So there is a need for a completely automated system that can deploy, monitor, and manage applications easily. This will reduce manual work, improve performance, and help companies provide better and more reliable services.

IV. Methodology

1) First, I studied AWS cloud services like EC2, S3, and RDS. I learned how EC2 is used to create virtual servers, S3 is used for storing files, and RDS is used for managing databases. This helped me understand how cloud systems work.
2) After that, I used Docker to package my application. Docker helps to create a container that includes the application and all its required files. This make sure the application runs the same on every system without errors.
3) I used GitHub to store and manage my code. It helped me keep track of changes and save different versions of my project. It also made it easy to update code and connect it with other tools.
4) Then, I set up a CI/CD pipeline. The Pipeline automatically builds, tests, and deploys the application whenever I make changes to the code. It reduces manual work and saves time.
5) I also used Ansible and a Bash script to automate server setup and deployment. It helps to install software and configure the server automatically, while Bash scripts are used for simple system tasks
6) Finally, I tested the deployment process. I checked how fast the application was deployed, if there were any errors, and how efficiently the system was working. This helped me understand the benefits of automation.

V. Implementation (Practical + Code Flow)

A. Step 1: App Code

```
# app.py
print("Hello AWS DevOps")
```

B. Step 2: Dockerfile

```
FROM python:3
COPY . /app
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

```
WORKDIR /app
CMD ["python", "app.py"]
```

C. Step 3: Bash Script

```
#!/bin/bash
mkdir -p /app/logs
echo "Logs folder created"
```

D. Step 4: Ansible Playbook

```
- hosts: aws_ec2
  tasks:
    - name: Install Docker
      apt:
        name: docker.io
        state: present
    - name: Run app container
      command: docker run -d myapp
```

E. Step 5: CI/CD Pipeline (./yml)

```
name: AWS DevOps Pipeline
on: push
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout code

    - name: Build Docker image
      run: docker build -t myapp .
    - name: Run container
      run: docker run myapp
    - name: Run Bash script
      run: bash setup.sh
    - name: Deploy using Ansible
      run: ansible-playbook -i hosts deploy.ym
```

F. Step 6: AWS Deployment

· Created an EC2 instance · Install Docker and other required tools. · Ansible automatically set up the server and ran the app · App got deployed automatically

G. Final Flow

· Pushed code to GitHub
· Pipeline started automatically
· Docker image was built
· Bash script ran system setup tasks.
· Ansible deployed the app on EC2
· App ran in the container

H. Output:

Hello AWS DevOps

VI. Architecture Diagram

The workflow from code development to deployment and monitoring is:

Developer
↓
GitHub (Code Push)
↓
CI/CD Pipeline (Automated Build & Test)
↓
Bash Scripts / Ansible (Automation Tasks)
↓
Docker (Package app in a container)
↓
AWS EC2 (Deploy Container)
↓
Application Live
↓
Monitoring (Continuous Monitoring)

Fig 1: AWS DevOps Architecture

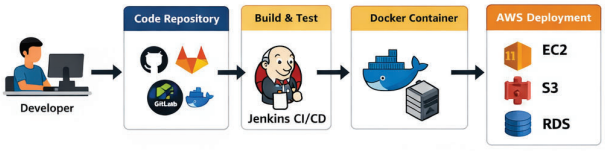


Fig 2: CI/CD Pipeline Workflow

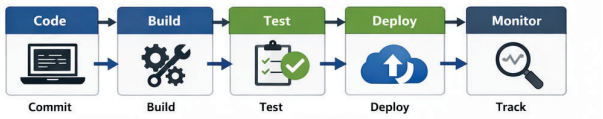
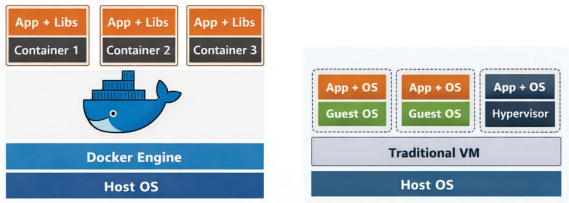


Fig 3: Docker Containerization



VII. Results

The result clearly shows the difference between manual deployment and using an automated DevOps framework on AWS with Docker, CI/CD pipelines, a Bash script, and Ansible.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

In the manual process, developers and the IT team have to set up servers, install software, run, test, and deploy applications by hand. This takes a lot of time, sometimes hours or even days. Manual work also increases the chance of human errors, like missing a step or installing the wrong software version. The speed of deployment is slow, the system is less reliable, and the use of resources is not efficient. For example, if the wrong EC2 instance is chosen, it may not handle the workload properly, causing slow performance.

In the automated framework, most tasks are done automatically. When code is pushed to GitHub, the CI/CD pipeline automatically builds and tests it. Docker packages the app and runs it in a container, which ensures it works the same everywhere. Bash Script creates necessary folders, and Ansible installs software and deploys the app on AWS EC2 automatically. This reduces errors, saves a lot of time, and makes deployment faster and more reliable

Automation also uses resources more efficiently. The right EC2 instance is selected, containers run smoothly, and developers can focus on coding instead of manual tasks. Overall, automated DevOps on AWS improves speed, reliability, and cost efficiency, and makes software delivery much easier compared to manual deployment.

A. Comparison Table

TABLE I
COMPARISON BETWEEN MANUAL AND AUTOMATED DEPLOYMENT

Feature	Manual Deployment	Automated DevOps
Time	High (hours/days)	Low (minutes)
Errors	High	Low
Cost	High	Optimized
Speed	Slow	Fast
Reliability	Low	High

VIII. Conclusion

This study shows that using cloud services like AWS EC2, S3, and RDS together with DevOps tools and automation scripts makes software delivery much better. Automation with Docker, Bash scripts, Ansible, and CI/CD pipelines helps deploy applications faster, more safely, and more reliably.

This framework helps developers and the IT team work together easily, reduces human mistakes, and saves a lot of deployment time. Security and training are still important because a wrong setup or lack of knowledge can reduce the benefits. In the future, adding AI-based monitoring and using multi-cloud systems could make things better.

Automation helps to save time, reduce human errors, and make the system more reliable. Docker helps to run applications in containers, so they work the same in every environment.

However, there are some challenges. If the system is not set up properly, it can cause a security issue. Also, the team must have proper knowledge and training to use these tools correctly.

Overall, using an automated DevOps framework makes cloud software deployment faster, more stable, and cheaper.

IX. Limitations

This study has some limitations. First, the system needs a skilled team to use tools like Docker, Ansible, and CI/CD pipelines. If the team is not trained, the system may not work properly.

Second, the setup of AWS and DevOps tools can be complex and time-consuming. It takes time to find all the services correctly.

Third, there can be security risks if services like IAM and VPC are not configured properly. The wrong setting can make data unsafe.

Fourth, this framework is designed for medium or large-scale applications. It is not suitable for very small projects.

Finally, this study does not include advanced technologies like AI-based monitoring, which can improve the system.

X. Future Work

- Use AI with Kubernetes to fix problems and apps automatically.
- Add automatic security checks for data and rules.
- Use AI to predict and stop system failures before they happen.
- Support both on-premise and cloud together (hybrid cloud).
- Make automatic setup of servers and resources smarter with AI.
- Show a dashboard that gives suggestions to improve the system.
- Save energy by smartly using cloud resources.
- Make automatic backup and recovery with AI help.
- Help the DevOps team work faster with AI assistants.
- Use AI to predict costs and performances and choose the best server.

References

[1] Bari et al., "Study of AWS DevOps CI/CD pipeline," IJISAE, 2023. [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/3812
[2] Doshi et al., "Study to analyze the performance of AWS EC2," IJSci, 2025. [Online]. Available: https://ijsci.com/index.php/home/article/view/679
[3] A. P. Akarsha, "Continuous Integration Research Based on Docker," IJERT, 2016. [Online]. Available: https://www.ijert.org/research/continuous-integration-research-based-on-docker-IJERTV5IS050443.pdf
[4] R. Banerjee and K. Mondal, "Optimizing DevOps workflow with shell scripting," AJRCOS, 2026. [Online]. Available: https://journalajrcos.com/index.php/AJRCOS/article/view/801/1900
[5] Pranav et al., "Using Ansible to automate servers in DevOps," IJASI, 2021. [Online]. Available: https://ijasi.org/index.php/ijasi/article/view/17
[6] Pavani et al., "Deploying deep learning using AWS and DevOps," IJERT, 2021. [Online]. Available: https://www.ijert.org/research/deploying-deep-learning-using-aws-and-devops-IJERTV10IS060111.pdf
[7] "A systematic literature review on Continuous Integration and Deployment for secure Cloud computing," ResearchGate, 2023.
[8] K. K. Paul and S. K. Pau, "Controller-Light CI/CD with Jenkins: Remote Container Builds," arXiv, 2025.
[9] "Improving Software Development with CI/CD for Agile DevOps," ResearchGate, 2024.
[10] R. Kusumadewi and R. Adrian, "Performance Analysis of DevOps with CI/CD Using Jenkins," MATICS Journal, 2023.
[11] Amazon Web Services, "AWS DevOps Guide." [Online]. Available: https://aws.amazon.com

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

[12] Docker Inc., ”Docker Documentation.” [Online]. Available: https://docs.docker.com
[13] GitHub, ”GitHub Actions Documentation.” [Online]. Available: https://docs.github.com
[14] Jenkins Community, ”Jenkins User Documentation.”
[15] Jenkins Project, ”Pipeline Syntax in Jenkins.”
[16] ”Continuous Monitoring Strategies in DevOps,” DevOps World Conference, 2023.
[17] Red Hat, ”Ansible Automation Platform Documentation,” 2022.
[18] J. Turnbull, *The Docker Book*, 2014.
[19] M. Huttermann, *DevOps for Developers*, Apress, 2012.
[20] GitHub, ”GitHub Actions CI/CD Guide,” 2023.
[21] Amazon Web Services, ”Amazon EC2 User Guide.”
[22] Amazon Web Services, ”Amazon S3 Developer Guide.”
[23] Amazon Web Services, ”Amazon VPC Networking Guide.”
[24] Amazon Web Services, ”Amazon IAM Best Practices.”
[25] V. Vangala, ”Multi-Cloud DevOps Automation,” ResearchGate, 2018.
[26] M. Baqar, S. Naqvi, and R. Khanda, ”AI-Augmented CI/CD Pipelines,” arXiv, 2025.
[27] ”Automation Practices in Cloud Platforms: Best Practices,” Cloud Engineering Conference, 2023.
[28] ”Challenges in DevOps Toolchain Adoption,” Software Engineering Review, 2022.
[29] ”Survey of Container and CI/CD Integration,” Int. J. Computer Science Engineering, 2023.
[30] M. Tammik, ”Cost Optimization Strategies for AWS Infrastructure,” 2024.
[31] R. N. Ghimire, ”Deploying Software in the Cloud with CI/CD Pipelines,” 2020.
[32] B. Mangla, ”Securing CI/CD with Automation Tools,” Packt Publishing, 2022.
[33] ”Cross-Cloud DevOps Frameworks and Case Studies,” IRE Journals, 2025.
[34] ”Security in Cloud CI/CD Pipelines,” Cyber Security Journal, 2024.
[35] ”Comparative Study of DevOps Tools for Cloud Deployment,” Int. J. Cloud Computing, 2024.
[36] ”Impact of Containerization on Deployment Reliability,” ContainerTech Journal, 2023.
[37] ”Ansible Use in Automated Infrastructure Management,” J. Recent Trends Comp Sci Eng., 2023.
[38] ”IaC Infrastructure Automation Patterns,” CloudPatterns Journal, 2023.
[39] ”Review of Cloud-Native DevOps and Automation Methods,” Int. J. Cloud Software Engineering, 2024.