**🚀 DevOps Learning Project**

A comprehensive DevOps learning project that demonstrates modern CI/CD practices, cloud deployment, and infrastructure automation.

**🌐 Live Application**

**Website**: http://54.255.152.69

**📚 What You'll Learn**

This project covers essential DevOps concepts and tools:

* ✅ **Web Development**: HTML, CSS, JavaScript
* ✅ **Version Control**: Git and GitHub
* ✅ **Cloud Infrastructure**: AWS EC2, Security Groups, Key Pairs
* ✅ **Server Management**: Linux, Apache, SSH
* ✅ **Infrastructure as Code**: Automated deployment scripts
* ✅ **CI/CD Pipelines**: GitHub Actions workflows
* ⏳ **Monitoring & Logging**: CloudWatch, application metrics
* ⏳ **Containerization**: Docker and container orchestration
* ⏳ **Infrastructure Automation**: Terraform or CloudFormation

**🏗️ Architecture**

┌─────────────────┐ ┌──────────────────┐ ┌─────────────────┐

│ Developer │ │ GitHub │ │ AWS Cloud │

│ │ │ │ │ │

│ Local Dev ────┼───▶│ Repository ────┼───▶│ EC2 Instance │

│ Git Commits │ │ GitHub Actions │ │ Apache Server │

│ │ │ CI/CD Pipeline │ │ Web Application │

└─────────────────┘ └──────────────────┘ └─────────────────┘

**🚀 Quick Start**

**Prerequisites**

* AWS Account with configured CLI
* GitHub Account
* Git installed locally
* SSH client (Git Bash on Windows)

**1. Clone Repository**

git clone https://github.com/zulfazlin/my-devops-webapp.git

cd my-devops-webapp

**2. Set Up AWS Infrastructure**

# Configure AWS CLI

aws configure

# Create EC2 key pair

aws ec2 create-key-pair --key-name my-devops-key --query 'KeyMaterial' --output text > scripts/my-devops-key.pem

# Create security group

aws ec2 create-security-group --group-name my-webapp-sg --description "Web application security group"

aws ec2 authorize-security-group-ingress --group-name my-webapp-sg --protocol tcp --port 22 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-name my-webapp-sg --protocol tcp --port 80 --cidr 0.0.0.0/0

# Launch EC2 instance

aws ec2 run-instances --image-id ami-0df7a207adb9748c7 --count 1 --instance-type t2.micro --key-name my-devops-key --security-groups my-webapp-sg --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=my-webapp-server}]'

**3. Deploy Manually (First Time)**

cd scripts

chmod +x \*.sh

./deploy.sh

**4. Set Up CI/CD Pipeline**

1. **Add GitHub Secrets** (Settings → Secrets → Actions):
   * AWS\_ACCESS\_KEY\_ID
   * AWS\_SECRET\_ACCESS\_KEY
   * AWS\_REGION
   * EC2\_SSH\_KEY
2. **Push code to trigger deployment**:
3. git add .
4. git commit -m "Update application"
5. git push origin main

**🛠️ Available Scripts**

**Deployment Scripts (/scripts)**

* **deploy.sh** - Automated deployment to EC2
* **server-status.sh** - Server health monitoring
* **rollback.sh** - Safe deployment rollback

**GitHub Actions Workflows (/.github/workflows)**

* **deploy.yml** - Main CI/CD pipeline
* **test.yml** - Pull request testing

**📁 Project Structure**

my-devops-webapp/

├── .github/

│ └── workflows/

│ ├── deploy.yml # Main CI/CD pipeline

│ └── test.yml # PR testing workflow

├── src/

│ └── index.html # Web application

├── scripts/

│ ├── deploy.sh # Deployment automation

│ ├── server-status.sh # Server monitoring

│ ├── rollback.sh # Rollback functionality

│ └── README.md # Scripts documentation

├── infrastructure/ # Future: IaC templates

└── README.md # This file

**🔄 Development Workflow**

**Making Changes**

1. **Create feature branch**:
2. git checkout -b feature/your-feature-name
3. **Make your changes** to src/index.html
4. **Test locally**:
5. # Open index.html in browser
6. open src/index.html
7. **Commit and push**:
8. git add .
9. git commit -m "Add: your feature description"
10. git push origin feature/your-feature-name
11. **Create Pull Request** on GitHub
12. **Automated testing** runs via GitHub Actions
13. **Merge to main** triggers automatic deployment

**Manual Operations**

# Deploy immediately

./scripts/deploy.sh

# Check server status

./scripts/server-status.sh

# Rollback if needed

./scripts/rollback.sh

# Connect to server

ssh -i scripts/my-devops-key.pem ec2-user@YOUR\_INSTANCE\_IP

**📊 Monitoring**

**GitHub Actions**

* **Deployment Status**: Check the Actions tab in GitHub
* **Build History**: View all deployment attempts
* **Logs**: Detailed logs for troubleshooting

**Server Monitoring**

# Real-time server status

./scripts/server-status.sh

# Apache access logs

ssh -i scripts/my-devops-key.pem ec2-user@YOUR\_INSTANCE\_IP 'sudo tail -f /var/log/httpd/access\_log'

# Apache error logs

ssh -i scripts/my-devops-key.pem ec2-user@YOUR\_INSTANCE\_IP 'sudo tail -f /var/log/httpd/error\_log'

**🔒 Security**

* **SSH Keys**: Stored securely as GitHub Secrets
* **AWS Credentials**: Never committed to repository
* **Security Groups**: Minimal required ports (22, 80)
* **File Permissions**: Proper ownership and permissions
* **Automated Backups**: Created before each deployment

**🚨 Troubleshooting**

**Common Issues**

**Deployment Fails**

# Check AWS credentials

aws sts get-caller-identity

# Verify instance is running

aws ec2 describe-instances --filters "Name=tag:Name,Values=my-webapp-server"

# Test SSH connectivity

ssh -i scripts/my-devops-key.pem ec2-user@YOUR\_INSTANCE\_IP

**Website Not Accessible**

# Check Apache status

./scripts/server-status.sh

# Verify security group

aws ec2 describe-security-groups --group-names my-webapp-sg

**GitHub Actions Failing**

* Check GitHub Secrets are correctly set
* Verify AWS credentials have proper permissions
* Review workflow logs in GitHub Actions tab

**🎯 Next Steps**

After completing this project, consider:

1. **Infrastructure as Code**: Implement Terraform or CloudFormation
2. **Containerization**: Dockerize the application
3. **Load Balancing**: Add multiple EC2 instances with ELB
4. **Database Integration**: Add RDS database
5. **Monitoring**: Implement CloudWatch alerts
6. **SSL/HTTPS**: Add certificate and secure connections
7. **CDN**: Implement CloudFront distribution
8. **Blue-Green Deployment**: Zero-downtime deployment strategy

**🤝 Contributing**

1. Fork the repository
2. Create your feature branch
3. Make your changes
4. Add tests if applicable
5. Create a Pull Request

**📄 License**

This project is for educational purposes. Feel free to use and modify for learning.

**🙏 Acknowledgments**

* AWS for providing free tier resources
* GitHub for Actions CI/CD platform
* The DevOps community for best practices and inspiration

**Happy Learning! 🚀**

*Remember: The goal is not just to make it work, but to understand how and why it works.*