# DevOps Case Study for Beginners: Automating Deployment with Jenkins

Scenario:

A small startup, QuickSite, wants to streamline their deployment process for their web application hosted on a single

server. They've already automated the deployment using Ansible but now want to integrate Jenkins for Continuous

Integration (CI) and Continuous Deployment (CD).

Goals:

1. Automate the build and deployment process using Jenkins.
2. Trigger the pipeline automatically whenever code is pushed to GitHub.
3. Use Jenkins to run tests and deploy the application.

Setup:

1. Server: One Linux server (e.g., Ubuntu).
2. Tools:
   - Jenkins for CI/CD.
   - Ansible for deployment.
   - GitHub for version control.
3. Application: A simple Node.js app.

Step 1: Install Jenkins

1. Install Jenkins on the server:

```
sudo apt update
sudo apt install openjdk-11-jdk -y
```

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins -y

sudo systemctl start jenkins

2. Access Jenkins:

Open http://<your_server_ip>:8080 in your browser.

3. Unlock Jenkins:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

4. Install suggested plugins and create an admin user.

Step 2: Create a Jenkins Pipeline

1. Set Up GitHub Integration:

- Go to Jenkins -> Manage Jenkins -> Manage Plugins.

- Install the Git Plugin and Pipeline Plugin.

2. Create a New Pipeline Job:

- Go to the Jenkins dashboard -> New Item -> Enter a name -> Select Pipeline -> Click OK.

3. Define the Pipeline Script:

```
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
```

```
            steps {
                git 'https://github.com/your-repo/quicksite.git'
            }
        }
        stage('Install Dependencies') {
            steps {
                sh 'npm install'
            }
        }
        stage('Run Tests') {
            steps {
                sh 'npm test'
            }
        }
        stage('Deploy Application') {
            steps {
                ansiblePlaybook credentialsId: 'ansible-ssh-key', playbook: 'deploy.yml'
            }
        }
    }
}
```

4. Save the job.

Step 3: Configure Ansible for Deployment

1. Ensure the Ansible playbook (deploy.yml) is available in the repository:

```
- name: Deploy QuickSite
```

```
    hosts: localhost
    tasks:
      - name: Pull latest code
        git:
          repo: 'https://github.com/your-repo/quicksite.git'
          dest: '/var/www/quicksite'

      - name: Install dependencies
        shell: npm install
        args:
          chdir: /var/www/quicksite

      - name: Restart application
        shell: pkill node; nohup node /var/www/quicksite/app.js &
```

2. Add the SSH key to Jenkins for Ansible to access the server.

Step 4: Automate the Workflow

1. Trigger Builds Automatically:

- Install the GitHub Integration Plugin in Jenkins.
- In the job configuration, go to Build Triggers and select GitHub hook trigger for GITScm polling.

2. Set Up GitHub Webhook:

- In your GitHub repository, go to Settings -> Webhooks -> Add webhook.
- Enter http://<your_jenkins_server_ip>:8080/github-webhook/ as the payload URL.

Results:

1. Developers push code to GitHub.

2. Jenkins automatically triggers the pipeline:

   - Clones the latest code.

   - Installs dependencies and runs tests.

   - Deploys the application using Ansible.

3. The web application is live with minimal downtime.

Key Takeaways:

1. Jenkins Pipelines: Simplify CI/CD by automating builds and deployments.

2. Integration with Ansible: Combines configuration management with CI/CD for a seamless workflow.

3. GitHub Webhooks: Enables automatic triggers for every code push.

This setup gives beginners hands-on experience with Jenkins, Ansible, and GitHub, covering the basics of CI/CD in a

practical, single-server scenario.