Module 06

Continuous Integration and Continuous Deployment (CI/CD) Pipelines

Job Connector Program

Outline

CI/CD Concepts & Pipeline Flow
Introduction to continuous integration and deployment, including automated workflows from code integration to production deployment.

Manual vs Automated Deployment
Understanding the differences between manual server deployment and automated CI/CD pipelines in real-world development.

CI/CD with GitHub and PM2
Implementing CI/CD pipelines using GitHub Actions, Docker-based builds, and PM2 for managing application processes on the server.

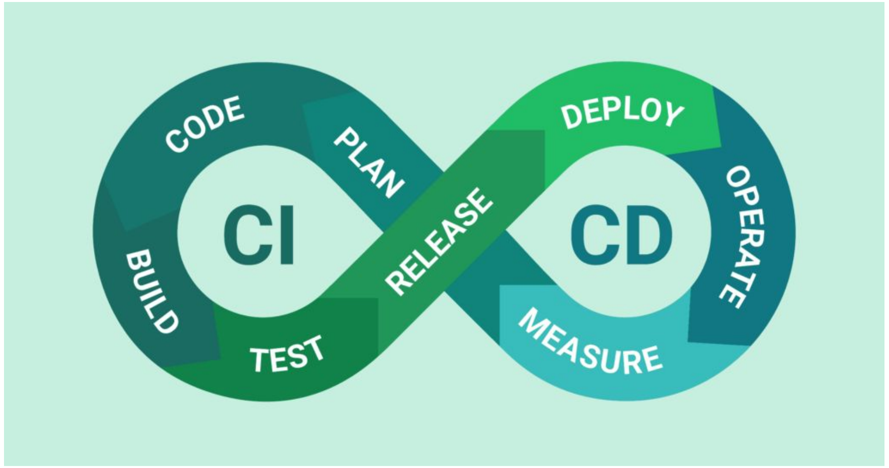
CI/CD Platforms with Coolify
Using Coolify as a self-hosted CI/CD and deployment platform that simplifies automation while leveraging Docker and Git integration.

Introduction to CI/CD

CI/CD is a development practice that automates how code changes are integrated and deployed to a server. Instead of manually updating applications, CI/CD allows deployment to happen automatically after code is pushed.

The main goal of CI/CD is to make deployment:

- Faster
- More consistent
- Less dependent on human actions



CI/CD Pipeline Flow

A CI/CD pipeline follows a simple but powerful flow:

1. **Git**
   Code changes are pushed to a repository (GitHub)

2. **Build**
   Dependencies are installed and the application is built

3. **Deploy**
   The application is updated and run on the server

This pipeline runs automatically without manual server access.

Manual Deployment (Previous Approach)

Manual Deployment Workflow

Before CI/CD, deployment is usually done manually:

- SSH into the server
- Pull the latest code
- Install dependencies
- Build the application
- Restart the service

This approach works, but it has limitations:

- Repetitive steps
- High chance of human error
- Difficult to scale for teams

Automated Deployment with CI/CD

Why Automated Deployment?

With CI/CD:

- Developers only push code
- The pipeline handles deployment automatically
- No repeated SSH commands

Benefits:

- Faster release cycle
- Consistent deployment process
- Production-ready workflow

CI/CD shifts deployment from a manual task to an automated system.

Tools for Implementing CI/CD

CI/CD is a process that can be implemented using various tools:

- GitHub Actions
- GitLab CI/CD
- Jenkins
- CircleCI

In this course, we use GitHub Actions because:

- It is built into GitHub
- Easy to configure
- Widely used in industry

What is GitHub Actions?

GitHub Actions is a CI/CD service that allows workflows to run automatically based on Git events.

Key characteristics:

- Uses YAML configuration files
- Triggered by push or pull request
- Runs commands on virtual machines (runners)

It turns GitHub into both a code repository and a deployment trigger.

Enabling GitHub Actions in a Project

Workflow Configuration Basics
GitHub Actions workflows are defined inside:
.github/workflows/

Each workflow file describes:
- When the pipeline runs
- What steps are executed

Example trigger:
```
on:
  push:
    branches:
      - main
```

This means deployment starts automatically when code is pushed to main.

Typical Pipeline Steps

A basic CI/CD pipeline usually includes:
- Checkout source code
- Install dependencies
- Build the project
- Deploy to server

These steps replace manual deployment commands with automated actions.

Connecting GitHub to the Server

How CI/CD Accesses the Server
To deploy automatically, GitHub Actions must connect to the server securely.
This is done using:

- SSH
- Key-based authentication

The pipeline uses SSH to execute deployment commands on the server without manual login.

Server-Side Requirements
Before CI/CD can work:

- SSH service must be active
- SSH port (22) must be open via firewall
- Deployment user must have access to project directory

SSH keys are used instead of passwords for security and automation.

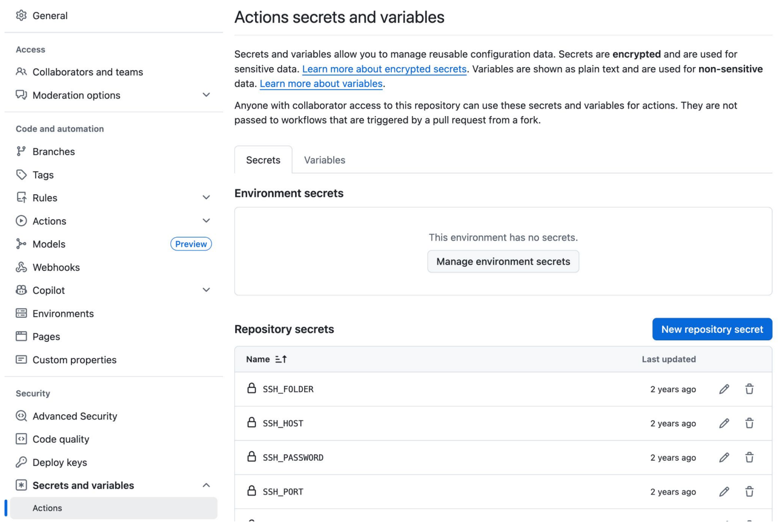
GitHub Secrets

Managing Sensitive Information
Sensitive data should never be written directly in pipeline files.

GitHub Secrets are used to store:

- Server IP or hostname
- SSH username
- Private SSH key

These secrets are injected into the pipeline securely during execution.



Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. Learn more about encrypted secrets. Variables are shown as plain text and are used for **non-sensitive** data. Learn more about variables.

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Secrets | Variables

Environment secrets

This environment has no secrets.

Manage environment secrets

Repository secrets New repository secret

Name	Last updated
SSH_FOLDER	2 years ago
SSH_HOST	2 years ago
SSH_PASSWORD	2 years ago
SSH_PORT	2 years ago

Basic GitHub Actions Pipeline (deploy.yml)

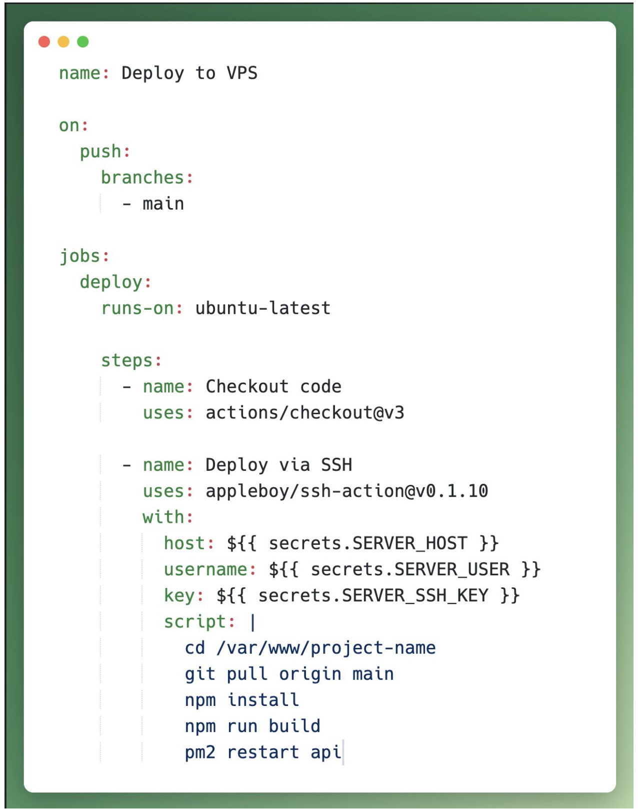
Purpose
Automatically deploy the application to the VPS whenever code is pushed to GitHub.

Trigger
Runs on every push to the main branch

How It Works

- GitHub Actions starts an Ubuntu runner
- Source code is checked out from the repository
- The runner connects to the VPS via SSH
- Server pulls the latest code
- Dependencies are installed and the app is built
- The application is restarted using PM2

Result
✔ No manual SSH
✔ Consistent deployment process
✔ Push to GitHub = Auto deploy to server



```yaml
name: Deploy to VPS

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Deploy via SSH
        uses: appleboy/ssh-action@v0.1.10
        with:
          host: ${{ secrets.SERVER_HOST }}
          username: ${{ secrets.SERVER_USER }}
          key: ${{ secrets.SERVER_SSH_KEY }}
          script: |
            cd /var/www/project-name
            git pull origin main
            npm install
            npm run build
            pm2 restart api
```

Coolify: Simplified CI/CD & Deployment Platform

Coolify is a **self-hosted platform** that helps developers **build, deploy, and manage applications automatically** using Git and Docker.

In simple terms:

Coolify replaces manual deployment scripts with an easy-to-use deployment platform.

https://coolify.io/docs/get-started/installation

Coolify in the CI/CD Context

Before Coolify, deployment usually involves:

- SSH into the server
- Pull code manually
- Run build commands
- Restart applications
- Write CI/CD scripts manually

Problems:

- Error-prone
- Hard to manage multiple projects
- Difficult for beginners

Coolify simplifies all of this.

Traditional CI/CD
Git ➔ GitHub Actions ➔ SSH ➔ Server ➔ Build ➔ Restart

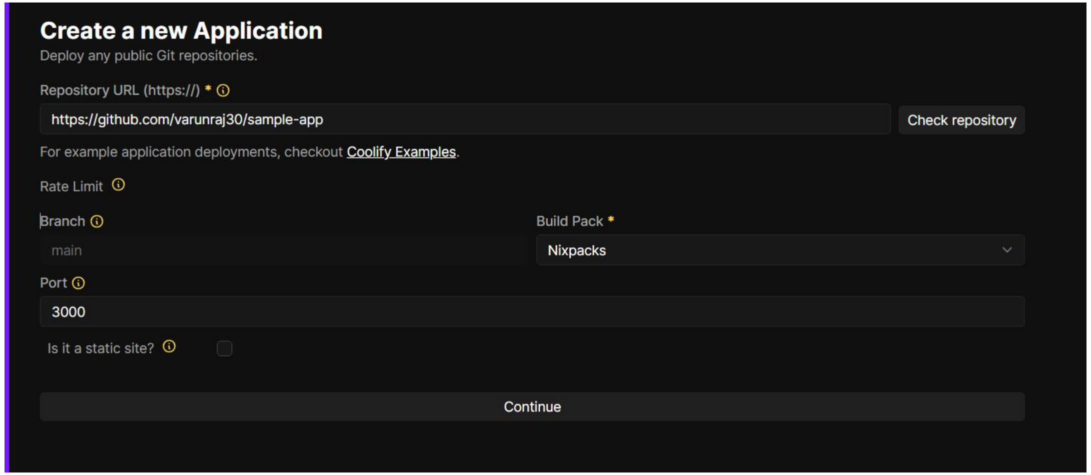
With Coolify
Git ➔ Coolify ➔ Build ➔ Deploy ➔ Run

Coolify vs Manual Deployment

Manual Deployment	Coolify
SSH access required	No SSH for daily deploy
Manual build steps	Automated build
PM2 commands	Container-based
Custom scripts	UI-based configuration
Hard to scale	Easier to manage multiple apps

Deploying on Coolify

Create a project and **select a public repository**. Provide the public URL of the repository and click on continue.



Create a new Application

Deploy any public Git repositories.

Repository URL (https://) *

https://github.com/varunraj30/sample-app

For example application deployments, checkout Coolify Examples.

Rate Limit

Branch	Build Pack *
main	Nixpacks

Port

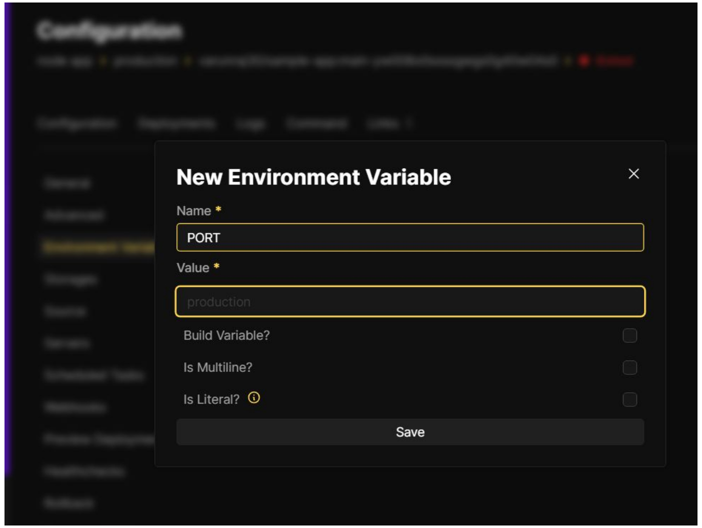
3000

Is it a static site? ☐

Deploying on Coolify

Go to environment variables and add environment variables (if your project uses any) click on save and then deploy.

Wait for the container to be created and then you can visit the link.



New Environment Variable

Name *: PORT

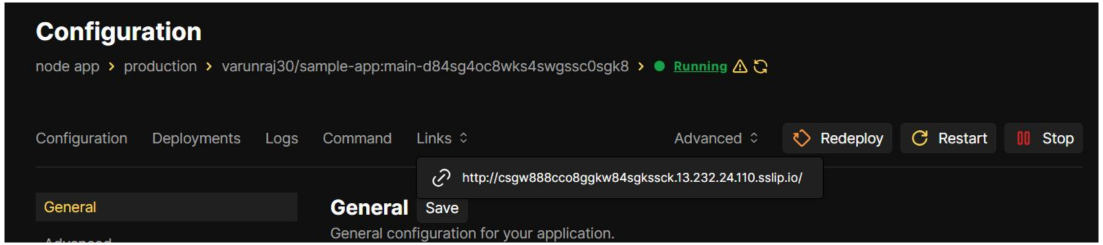
Value *: production

Build Variable? ☐

Is Multiline? ☐

Is Literal? ☐

Save



Configuration

node app › production › varunraj30/sample-app:main-d84sg4oc8wks4swgssc0sgk8 › ● Running

Configuration | Deployments | Logs | Command | Links | Advanced | Redeploy | Restart | Stop

http://csgw888cco8ggkw84sgkssck.13.232.24.110.sslip.io/

General

General Save

General configuration for your application.

Thank you