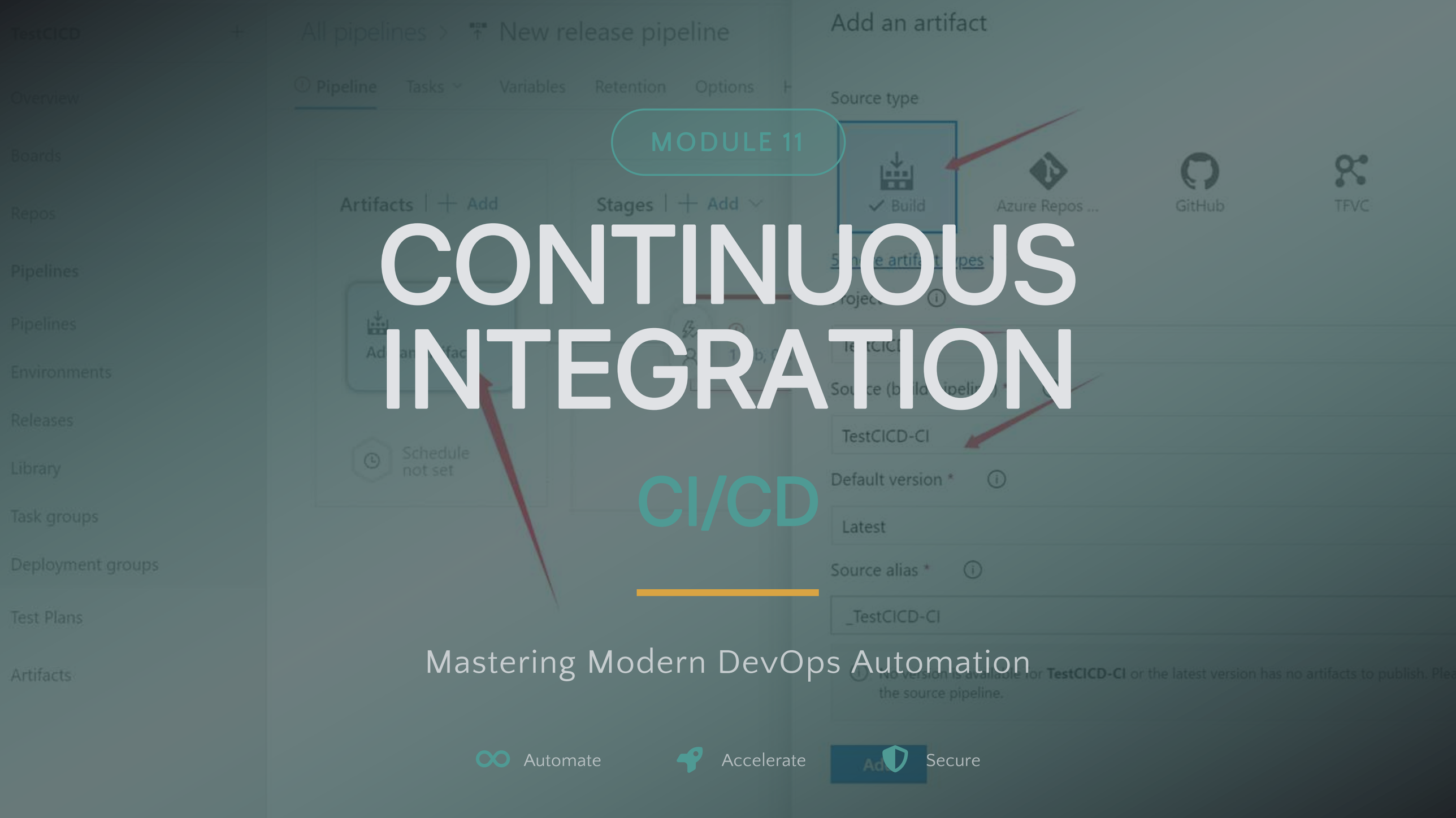

MODULE 11
CONTINUOUS INTEGRATION
CI/CD
Mastering Modern DevOps Automation
Automate
Accelerate
Secure

What is CI/CD?

Understanding the Foundation of Modern Software Delivery

CI/CD

Continuous Integration + Continuous Delivery/Deployment

CI Continuous Integration

Automatically building and testing code whenever a developer pushes code changes to the repository.

Code Integration

CD Continuous Delivery

Code is automatically prepared for deployment and released to production with manual approval.

Ready to Deploy

CD Continuous Deployment

Code is automatically deployed to production without manual intervention.

Auto Deploy

CI/CD is the backbone of DevOps

Enabling rapid, reliable, and repeatable software delivery

Why Do We Need CI/CD?

Transforming Software Development from Chaos to Confidence

Without CI/CD

Manual Testing

Time-consuming, error-prone, and inconsistent testing processes

Late Bug Detection

Issues discovered only after deployment, causing delays

Slow Releases

Months or years between releases, missing market opportunities

Deployment Failures

High-risk deployments with frequent rollbacks and downtime

✕ Inefficient & Risky

With CI/CD

Faster Delivery 🚀

Deploy multiple times daily with rapid feedback loops

Automated Testing ✓✓

Comprehensive test suites run automatically on every change

Fewer Bugs 🐞

Early detection and prevention of defects before production

Reliable Deployments 🔁

Repeatable, predictable processes with instant rollback capability

✓✓ Efficient & Safe

Key Insight: CI/CD transforms software delivery from a high-risk, manual process to a reliable, automated pipeline

Common Types of CI/CD Systems

Three Approaches to Building Your Pipeline

1 Self-Hosted

Definition

Installed and managed on your own infrastructure. Provides maximum control and customization.

Key Characteristics

- ✓ Full control over infrastructure
- ✓ Highly customizable and extensible
- ✓ On-premises data security
- ✗ Requires maintenance and updates

Example: Jenkins

Most popular self-hosted automation server with 2000+ plugins

2 VCS Embedded

Definition

Integrated directly into your version control platform. Seamless workflow with your codebase.

Key Characteristics

- ✓ Native integration with Git platform
- ✓ No separate infrastructure needed
- ✓ Simple YAML-based configuration
- ✗ Limited to platform ecosystem

Example: GitHub Actions

Native CI/CD built into GitHub with 10,000+ free minutes/month

3 External Managed

Definition

Fully managed cloud-based CI/CD service. Focus on code, not infrastructure.

Key Characteristics

- ✓ Zero infrastructure maintenance
- ✓ Fast setup and scaling
- ✓ Enterprise-grade security
- ✗ Usage-based pricing model

Example: CircleCI

Cloud-based CI/CD with 2+ million workflows daily

Popular CI/CD Tools Overview

Choosing the Right Tool for Your Pipeline

Tool	Type	Key Features
Jenkins Open-source automation server	Self-hosted	Highly customizable: 2000+ plugins available Pipeline as Code: Jenkinsfile for version-controlled pipelines Multi-branch support: Automatically builds all branches
GitHub Actions GitHub CI/CD	Native VCS Embedded	Native integration: Built directly into GitHub Marketplace: 10,000+ pre-built actions available YAML-based: Simple, readable workflow syntax
CircleCI Cloud-based CI/CD platform	External Managed	Fast & cloud-based: Parallel execution across multiple environments Docker-native: First-class container support Orbs ecosystem: Reusable configuration packages

Best for:
Maximum customization & control

Best for:
GitHub-native workflows

Best for:
Fast, managed cloud CI/CD

Jenkins Deep Dive

The Open-Source Automation Powerhouse

What is Jenkins?

Jenkins is an open-source automation server that enables developers to reliably build, test, and deploy software.

Core Capabilities

Build
Compile & package applications

Test
Automated testing & quality checks

Deploy
Automated deployment to any environment

Jenkins Setup (Basic)

1. **Install Java**
   Jenkins requires Java runtime
2. **Download Jenkins**
   Get latest stable version
3. **Start Service**
   Launch Jenkins daemon
4. **Browser Access**
   `localhost:8080`

Jenkins Jobs

Freestyle

Basic jobs via UI. Good for simple builds.

Pipeline

Complex workflows as code (Jenkinsfile). Version controlled.

Multibranch

Auto creates jobs for each branch/PR.

Jenkins Pipeline & Docker

Building and Deploying with Containers

</>Basic Pipeline Example

```
pipeline {
  agent anystages {
    stage('Build') {
      steps {
        echo'Building...'
      }
    }
    stage('Test') {
      steps {
        echo'Testing...'
      }
    }
    stage('Deploy') {
      steps {
        echo'Deploying...'
      }
    }
  }
}
```

Build	Test	Deploy
Compile	Validate	Release

Jenkins + Docker Workflow

1. **Build Docker Image**
   Create container with application

2. **Push to Registry**
   Upload to Docker Hub or ECR

3. **Deploy Container**
   Run on target environment

Key Benefits

- Consistent environments
- Simplified dependency management
- Scalable & portable deployments

Pipeline as Code: Jenkins pipelines are defined in a Jenkinsfile, stored in your repo, enabling version control, code review, and audit trails.

GitHub Actions Overview

Native CI/CD Built Into Your Git Platform

What is GitHub Actions?

GitHub Actions is a CI/CD tool built directly into GitHub, enabling you to automate workflows directly from your repository.

Workflow Triggers

- Code is pushed
- Pull request created
- Manual trigger

Workflow Syntax

Workflows are written in YAML and stored in your repository.

.github/workflows/ci.yml

</>Basic GitHub Actions Workflow

```
name: CI Pipelineon: [push]
jobs:
  build:
    runs-on: ubuntu-lateststeps:
      - uses: actions/checkout@v3
      - name: Run Testsrun: echo "Testing code"
```

name	on	jobs	steps
Workflow name	Trigger event	Workflow tasks	Job commands

GitHub-native: No separate server needed. Your code and CI/CD live together.

CircleCI Overview

Cloud-Based CI/CD with Speed and Scale

What is CircleCI?

CircleCI is a cloud-based CI/CD platform that enables rapid software development and deployment.

Key Features

- **Fast pipelines:** Parallel execution across multiple environments
- **YAML configuration:** Simple, version-controlled pipeline definitions
- **Docker-native:** First-class container support

Config File

CircleCI uses a YAML configuration file to define your pipeline.

```
.circleci/config.yml
```

Orbs in CircleCI

What are Orbs?

Orbs are reusable packages of CI configuration that simplify complex workflows.

Docker Orb
Build and push Docker images

AWS Orb
Deploy to AWS services

Workflow & Jobs

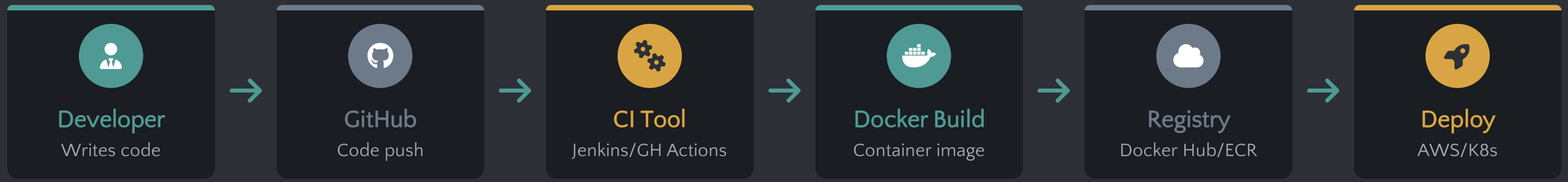
J **Job = Single Task**
A job is a single unit of work (e.g., build, test).

W **Workflow = Job Orchestration**
A workflow connects multiple jobs with dependencies.

Real DevOps CI Workflow

End-to-End Automation from Code to Production

Complete CI/CD Pipeline Flow



1 Push Code

Developer pushes to Git repository

2 CI Triggers

Automatic pipeline execution begins

3 Build & Test

Code compilation and automated testing

4 Containerize

Docker image creation and registry push

5 Deploy

Container deployment to production

6 Monitor

Application performance monitoring

Tools Used

GitHub, Jenkins/GitHub Actions, Docker, Kubernetes/AWS

Project 1: Jenkins + Docker + AWS

Real-World CI/CD Pipeline with Flask Application

Tech Stack

GitHub	Jenkins	Docker
AWS EC2	Docker Hub	Flask

Project Structure

```
jenkins-docker-aws/
├── app.py
├── Dockerfile
├── requirements.txt
└── Jenkinsfile
```

Sample Flask App (app.py)

Dockerfile

```
FROM python:3.10
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

Jenkinsfile

```
pipeline {
  agent any
  stages {
    stage('Clone') {
      steps {
        git 'https://github.com/user/repo.git'
      }
    }
    stage('Build') {
      steps {
        sh 'docker build -t myapp .'
      }
    }
    stage('Push') {
      steps {
        sh 'docker push username/myapp'
      }
    }
    stage('Deploy') {
      steps {
        sh 'docker run -d -p 80:5000 myapp'
      }
    }
  }
}
```

Project 2: GitHub Actions → K8s

Cloud-Native CI/CD with Kubernetes Deployment

Tech Stack

GitHub | Actions | Docker

Kubernetes | kubectl | Flask

GitHub Actions Workflow

```
name: CI-CDon: [push]
jobs:
  deploy:
    runs-on: ubuntu-lateststeps:
      - uses: actions/checkout@v3
      - name: Build Docker Imagerun: docker build -t myapp .
      - name: Push Imagerun: docker push username/myapp
      - name: Deploy to K8srun: kubectl apply -f deployment.yml
```

Kubernetes Deployment YAML

```
apiVersion: apps/v1kind: Deploymentmetadata:
  name: flask-appspec:
  replicas: 2selector:
    matchLabels:
      app: flasktemplate:
    metadata:
      labels:
        app: flaskspec:
      containers:
      - name: flaskimage: username/myappports:
        - containerPort: 5000
```

Real DevOps Learning

- GitHub-native CI automation

Interview Q&A

Real & Practical Questions for DevOps Roles

Q1 What is CI?

A: CI automatically builds and tests code after every change, enabling early bug detection.

Q2 Difference: Jenkins vs GitHub Actions?

A: Jenkins is self-hosted and customizable. GitHub Actions is simpler and GitHub-native.

Q3 What is a pipeline?

A: A pipeline is a sequence of automated steps (build, test, deploy) that code changes go through.

Q4 Workflow in GitHub Actions?

A: A workflow is an automated process defined in YAML, triggered by events like a push.

Q5 What are Orbs?

A: Orbs are reusable CircleCI configuration packages that simplify pipeline setup.

Q6 Docker vs Kubernetes?

A: Docker packages apps. Kubernetes manages and scales containers.

Q7 How does CI help DevOps?

A: CI detects bugs early by automatically building and testing code on every change.

Q8 Why use Docker in CI/CD?

A: Docker ensures consistent environments, eliminating "it works on my machine" issues.

Interview Tip: Focus on practical experience and real-world scenarios to demonstrate hands-on knowledge.

Quick Reference

Cheat Sheet & MCQs for Exam Preparation

CI/CD Cheat Sheet

Jenkins

- **Job:** Build task
- **Pipeline:** Code-defined workflow
- **Jenkinsfile:** Pipeline definition

GitHub Actions

- **Workflow:** Automated process
- **Jobs:** Workflow units
- **Steps:** Job commands
- **Runners:** Job executors

CircleCI

- **config.yml:** Main config file
- **Jobs:** Single tasks
- **Workflows:** Job connections
- **Orbs:** Reusable configs

Exam Focus Points

- **CI/CD Definition:** Integration vs Delivery vs Deployment
- **Tool Comparison:** Jenkins vs GitHub Actions trade-offs
- **YAML Syntax:** Workflow structure and keywords
- **Pipeline Concepts:** Stages, jobs, steps, workflows
- **Automation Benefits:** Speed, quality, reliability
- **Real-World Usage:** Docker, K8s, cloud deployment

MCQs with Answers

1. Jenkins is:

A) Cloud only
B) Self-hosted CI/CD ✓
C) Database

2. GitHub Actions files are stored in:

A) /ci
B) .github/workflows ✓
C) /pipelines

3. CircleCI uses:

A) XML
B) YAML ✓
C) JSON

4. CI mainly focuses on:

A) Monitoring
B) Testing & Build ✓

MODULE COMPLETE

Master CI/CD, Master DevOps

CI/CD automates development from code to production

Jenkins
Powerful & Flexible

GitHub Actions
Simple & Native

CircleCI
Fast & Cloud-Based

A Mandatory Skill for DevOps Engineers

Transform your career with automation expertise

Automate Everything Deploy Faster Build Reliably