Module 1: Introduction to Github Actions

- What is GitHub Actions
- GitHub Actions vs Jenkins — key differences
- Core benefits — no server, no plugins, no maintenance
- Where GitHub Actions fits in the DevOps lifecycle
- GitHub Actions pricing — free tier, minutes, storage
- Overview of the course project — i27Academy DevOps Dashboard
- How this course is structured — learning files + project milestones

Module 2: Github Actions Building Blocks

- Workflow
- Events
- Jobs
- Steps
- Actions
- Runners
- How they connect — Event → Workflow → Job → Step → Runner
- Workflow file location — .github/workflows/*.yml
- Workflow file structure — name, on, jobs
- runs-on vs run (common mistake)
- Your first workflow — hello world
- Parallel jobs vs sequential jobs (needs:)
- Job status — success, failure, cancelled, skipped

Module 3: Event that trigger workflows

- What is an event
- push
- pull_request
- workflow_dispatch
- schedule
- workflow_run
- repository_dispatch
- Combining multiple triggers
- Activity types and filters

Module 4: Workflow Runners

- GitHub-hosted runners
- ubuntu-latest, windows-latest, macos-latest
- Runner specifications (CPU, RAM, storage)
- Runner environment — what's pre-installed
- Self-hosted runners (preview — covered in depth in Module 16)
- runs-on — targeting specific runners
- Running on multiple OS
- Runner labels

Module 5: Using Third-Party Actions

- What is an action
- Types of actions — composite, JavaScript, Docker
- Finding actions in the marketplace
- Using an action — uses, with
- Version pinning — @v4, @main, SHA
- Evaluating third-party actions
- Official actions — actions/*, docker/*, github/*
- Dependabot for actions

Module 6: Environment Variables & Secrets

- Environment variables — env:
  i. Workflow level
  ii. Job level
  iii. Step level
- Repository Variables (vars.*)
- Secrets (secrets.*)
  i. Repository secrets
  ii. Environment secrets
  iii. Organization secrets
- GITHUB_TOKEN
- Secrets masking
- Dynamic Variables ($GITHUB_ENV)

Module 7: Step Outputs and Job Outputs

- Step Outputs
  i. Setting a step output ($GITHUB_OUTPUT)
  ii. Reading a step output
  iii. id: field
- Job Outputs
  i. Setting a job output (outputs: block)
  ii. Reading a job output (needs.JOB.outputs.key)
  iii. $GITHUB_ENV vs $GITHUB_OUTPUT

Module 8: Contexts

- What is a context
- github.*
- env.*
- vars.*
- secrets.*
- steps.*
- needs.*
- runner.*
- matrix.*
- inputs.*
- Debugging — toJson(github)

Module 9: Expressions & Functions

- Expression syntax — ${{ }}
- Operators — ==, !=, &&, ||, !
- String functions — contains, startsWith, endsWith, format, join
- Data functions — toJson, fromJson, hashFiles
- Status functions — success, failure, always, cancelled
- if: conditions

Module 10: Artifacts and Caching

- What are artifacts
  i. Upload-artifact
  ii. Download-artifact
  iii. Retention days
  iv. Passing artifacts between jobs
- What is caching
  i. actions/cache
  ii. Cache key strategies
  iii. hashFiles()
  iv. Cache restore keys
- Maven .m2 caching
- Cache hit vs cache miss

Module 11: Working with Matrices

- What is a matrix strategy
  i. Single dimension matrix
  ii. Multi-dimensional matrix
- include / exclude
- fail-fast
- max-parallel
- Dynamic matrix — fromJson()
- matrix.* context

Module 12: Environments & Approval Gates

- What are environments
- Creating environments in GitHub UI
  i. development, staging, production
- Protection Rules
  i. Required reviewers
  ii. Wait timer
  iii. Deployment branches and tags
- Environment secrets
- Environment URL
- dev → staging → production promotion
- Concurrency in deployments
- Tag-based deployments

Module 13: Implementing Custom Actions

- What is a Custom Action
- Types of Actions
  i. JavaScript
  ii. Docker
  iii. Composite
- Composite Action Structure
  i. action.yml
  ii. inputs
  iii. outputs
- runs: using: composite
- shell: bash
- Local composite action
- Remote composite action
- Sharing actions across repos

Module 14: Creating Reusable Workflows

- What is a reusable workflow
- workflow_call trigger
- Inputs
- Secrets
  i. Explicit secret
  ii. Secrets: inherit
- outputs
- secrets: inherit
- Caller workflow
- Called workflow
- Passing outputs across reusable workflows
- Composite action vs reusable workflow

Module 15: Self-Hosted Runners

- What is a self-hosted runner
- GitHub-hosted vs self-hosted
- Installing runner software
- Registering a runner
- Runner labels
- Runner groups
- Security considerations
- Use cases — private network, custom software, cost

Module 16: OIDC-Based Cloud Authentication

- What is OIDC
- Why OIDC — no long-lived secrets
- How OIDC works with GitHub Actions
- id-token: write permission
- AWS — configure-aws-credentials
- GCP — google-github-actions/auth
- Azure — azure/login
- Replacing stored cloud credentials with OIDC

Module 17: Managing Concurrency

- What is concurrency
- concurrency: group
- cancel-in-progress: true vs false
- Per-branch concurrency
- Per-PR concurrency
- Global deploy concurrency
- CI vs deployment concurrency patterns

Module 18: Security Hardening

- GITHUB_TOKEN permissions
- Least privilege
  i. Workflow level permissions
  ii. Job level permissions
- Workflow level vs job level permissions
- Script injection
  i. Unsafe patterns
  ii. Safe patterns
- CodeQL security scanning
- Dependabot
  i. Dependency Updates
  ii. Action Updates
- Secret scanning

- Third-party action security

Module 19: Final Real-Time Project Pipeline

- Course recap — what we built module by module
- Project pipeline evolution — v1 through final
- Complete CI/CD pipeline on DevOps Dashboard
- CI — test, build, Docker image
- CD — dev → staging → production with approval gate
- Self-hosted runner for deployment
- OIDC for cloud authentication
- Security hardening applied
- Final pipeline walkthrough end to end