## 1. What is GitHub Actions and how does it work in CI/CD pipelines?

GitHub Actions is a CI/CD (Continuous Integration/Continuous Deployment) platform that automates workflows directly within GitHub repositories. It allows you to build, test, and deploy code automatically when certain events (e.g., push, pull\_request) occur.

CI (Continuous Integration): Automatically runs tests on code changes.

CD (Continuous Deployment): Automatically deploys code to production/staging environments.

Workflows are defined in YAML files (.github/workflows/) and executed on GitHub-hosted or self-hosted runners.

## 2. What are the main components of a GitHub Actions workflow?

**Workflow**: A YAML file defining automation (e.g., ci.yml).

**Job**: A set of steps that run on the same runner (can run sequentially or in parallel).

**Step**: A single task in a job (can run a command or an action).

**Action**: A reusable unit of code (e.g., actions/checkout@v4).

**Runner**: The machine (GitHub-hosted or self-hosted) that executes jobs.

Example:

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- run: npm install

## 3. How do you trigger a workflow in GitHub Actions?

Workflows can be triggered by:

push: Code pushed to a branch.

pull\_request: PR opened or updated.

schedule: Cron-based triggers (e.g., 0 0 \* \* \*).

workflow\_dispatch: Manual trigger from GitHub UI/API.

repository\_dispatch: External events via API.

Example:

on:

push:

branches: [ main ]

pull\_request:

branches: [ main ]

workflow\_dispatch:

## 4. How do you pass secrets into a GitHub Actions workflow securely?

Secrets (e.g., API keys) are stored in GitHub Secrets (Settings → Secrets) and accessed in workflows using ${{ secrets.NAME }}.

Example:

yaml

Copy

steps:

- name: Deploy

run: ./deploy.sh

env:

API\_KEY: ${{ secrets.API\_KEY }}

Best Practice: Never hardcode secrets in workflow files.

## 5. What’s the difference between jobs.<job\_id>.runs-on and container in a workflow?

runs-on: Specifies the OS (e.g., ubuntu-latest) where the job runs.

container: Runs the job inside a Docker container (useful for isolated environments).

Example:

jobs:

build:

runs-on: ubuntu-latest

container: node:16

steps:

- run: node --version

## 6. Explain matrix builds and how to use them.

A matrix runs a job in multiple configurations (e.g., different OSes, Node.js versions).

Example:

jobs:

test:

runs-on: ubuntu-latest

strategy:

matrix:

node-version: [14, 16, 18]

steps:

- uses: actions/setup-node@v3

with:

node-version: ${{ matrix.node-version }}

This runs the job 3 times (once for each Node.js version).

## 7. How do you cache dependencies between workflow runs?

Use actions/cache to save dependencies (e.g., node\_modules).

Example (Node.js):

steps:

- uses: actions/cache@v3

with:

path: node\_modules

key: ${{ runner.os }}-node-${{ hashFiles('package-lock.json') }}

## 8. How can you reuse workflows using workflow\_call?

The workflow\_call trigger allows a workflow to be called by another workflow.

Example (reusable.yml):

on:

workflow\_call:

inputs:

version:

required: true

type: string

jobs:

deploy:

steps:

- run: echo "Deploying ${{ inputs.version }}"

Calling workflow:

jobs:

call-reusable:

uses: ./.github/workflows/reusable.yml

with:

version: "1.0"

## 9. What’s the difference between uses: and run: in GitHub Actions?

uses: Runs a prebuilt action (e.g., actions/checkout@v4).

run: Executes a shell command (e.g., npm install).

Example:

steps:

- uses: actions/checkout@v4 # Action

- run: npm install # Command

## 10. Have you written a custom GitHub Action before? If yes, how did you do it?

Yes! Custom actions can be:

JavaScript/TypeScript: Uses @actions/core for inputs/outputs.

Docker containers: For complex environments.

Composite actions: Combines shell commands.

Example (JavaScript action action.yml):

name: 'Greet'

description: 'Prints a greeting'

inputs:

name:

required: true

runs:

using: 'node16'

main: 'dist/index.js'

Steps to create:

Write code in index.js.

Bundle with @vercel/ncc.

Push to a repo and reference it in workflows.