github actions interview questions fundamental and important ones please most asked with solutions

**ChatGPT said:**

Here are some of the most fundamental and commonly asked GitHub Actions interview questions, along with their solutions:

**1. What is GitHub Actions?**

**Answer:**  
GitHub Actions is an automation tool integrated into GitHub that allows you to automate workflows, such as CI/CD (Continuous Integration and Continuous Deployment), testing, deployment, or any other task related to software development. It uses workflows defined in YAML files, which specify jobs and actions to run on specified events (e.g., pushing code, creating pull requests).

**2. What is a GitHub Actions Workflow?**

**Answer:**  
A **workflow** is a configurable automated process made up of one or more **jobs** and **steps**. It is defined in a .yml file located in the .github/workflows directory of the repository. A workflow can be triggered by GitHub events like push, pull request, or manually through the GitHub Actions UI.

**Example:**

yaml

CopyEdit

name: CI Workflow

on: [push, pull\_request]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- run: echo "Running tests"

**3. What are Jobs in GitHub Actions?**

**Answer:**  
A **job** is a set of steps executed on the same runner. Jobs are executed in parallel by default, but they can be made to run sequentially if necessary using needs. A job can have a defined **runner** (e.g., ubuntu-latest, windows-latest) that specifies the environment the job will run in.

**Example:**

yaml

CopyEdit

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout Code

uses: actions/checkout@v2

- name: Run tests

run: npm test

**4. What is a GitHub Actions Step?**

**Answer:**  
A **step** is a single task that can either be an **action** or a **script**. A step is part of a job and can use actions like actions/checkout, actions/setup-node, or custom scripts to execute commands. Steps are executed sequentially in the job.

**Example:**

yaml

CopyEdit

jobs:

build:

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up Node.js

uses: actions/setup-node@v2

with:

node-version: '14'

- name: Install dependencies

run: npm install

**5. What are GitHub Actions Triggers?**

**Answer:**  
Triggers are events that start a workflow. Common triggers are:

* push: Triggered when a push occurs.
* pull\_request: Triggered when a pull request is created or updated.
* workflow\_dispatch: Triggered manually through the GitHub Actions interface.
* schedule: Triggered based on cron expressions.

**Example:**

yaml

CopyEdit

on:

push:

branches:

- main

pull\_request:

branches:

- main

**6. What is the Difference Between run and uses in GitHub Actions?**

**Answer:**

* **run**: Executes a script or command directly in the workflow.
* **uses**: Refers to an action defined by GitHub or the community (or a custom action), which can be reused.

**Example of run:**

yaml

CopyEdit

- name: Run a script

run: echo "Hello, world!"

**Example of uses:**

yaml

CopyEdit

- name: Checkout code

uses: actions/checkout@v2

**7. What is needs in GitHub Actions?**

**Answer:**  
needs is used to specify the dependency between jobs. It ensures that one job only runs after another job has successfully completed. This is useful when you want to create sequential workflows where later jobs depend on the outcome of earlier jobs.

**Example:**

yaml

CopyEdit

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout Code

uses: actions/checkout@v2

test:

runs-on: ubuntu-latest

needs: build

steps:

- name: Run Tests

run: npm test

**8. What are Runners in GitHub Actions?**

**Answer:**  
A **runner** is a server that has the GitHub Actions runner application installed. It is responsible for executing the jobs defined in a workflow. GitHub provides hosted runners with pre-installed software, or you can set up your own self-hosted runners.

**9. What are Artifacts in GitHub Actions?**

**Answer:**  
Artifacts are files that are uploaded after a job completes. They can be stored and used in later jobs or downloaded by users. This is useful for storing build outputs or test results.

**Example:**

yaml

CopyEdit

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Build project

run: npm run build

- name: Upload build artifact

uses: actions/upload-artifact@v2

with:

name: build-artifact

path: ./build

**10. How do you create a Custom Action in GitHub Actions?**

**Answer:**  
To create a custom action, you need to define a new repository with a Dockerfile or JavaScript code. Then, you can use that action within any workflow by referring to the repository and version.

**Example of a JavaScript Action (action.yml):**

yaml

CopyEdit

name: 'My Custom Action'

description: 'A simple custom action'

inputs:

name:

description: 'Your name'

required: true

default: 'world'

runs:

using: 'node12'

main: 'index.js'

**11. What is the checkout action used for in GitHub Actions?**

**Answer:**  
The checkout action is used to check out the code from the repository so that it can be accessed and worked on by subsequent steps in the workflow. This action is essential when working with GitHub Actions in any CI/CD pipeline.

**Example:**

yaml

CopyEdit

steps:

- name: Checkout code

uses: actions/checkout@v2

**12. What are Secrets in GitHub Actions?**

**Answer:**  
**Secrets** are sensitive data like API keys, passwords, or tokens that are stored in GitHub's encrypted storage and used within workflows. They are accessed via secrets in your workflow file. Secrets help keep sensitive information safe.

**Example:**

yaml

CopyEdit

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: Deploy

run: echo ${{ secrets.MY\_SECRET\_KEY }}

**13. How can you run a GitHub Action only for specific branches?**

**Answer:**  
You can specify the branches in the on section of your workflow file to run the action only for certain branches.

**Example:**

yaml

CopyEdit

on:

push:

branches:

- main

- feature/\*

**14. What is a Matrix in GitHub Actions?**

**Answer:**  
A **matrix** allows you to run a job in multiple environments or with different configurations. This is useful for testing across various versions of a language, different operating systems, or multiple configurations.

**Example:**

yaml

CopyEdit

jobs:

build:

runs-on: ubuntu-latest

strategy:

matrix:

node-version: [12, 14, 16]

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up Node.js

uses: actions/setup-node@v2

with:

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

- name: Install dependencies

run: npm install

**15. How do you handle failures in GitHub Actions?**

**Answer:**  
GitHub Actions allows you to control the flow of your workflows with features like if, continue-on-error, and fail-fast.

* continue-on-error: Allows a step to fail without failing the entire job.
* if: Specifies conditional execution of steps based on job outcomes.

**Example:**

yaml

CopyEdit

jobs:

test:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Run Tests

run: npm test

continue-on-error: true

These questions cover essential GitHub Actions topics that are fundamental for understanding and mastering the platform.

4o

**You said:**

continue please with more fundametnal questions that interviewer can asked

**ChatGPT said:**

Sure! Here are more fundamental GitHub Actions interview questions that interviewers commonly ask, along with their solutions:

**16. What is the difference between workflow\_run and workflow\_dispatch triggers in GitHub Actions?**

**Answer:**

* **workflow\_run**: It triggers one workflow based on the success or completion of another workflow. This is useful for chaining workflows together.

**Example:**

yaml

CopyEdit

on:

workflow\_run:

workflows: ["CI Workflow"]

types:

- completed

* **workflow\_dispatch**: It allows you to manually trigger a workflow, either through the GitHub UI or using the GitHub API. You can also pass input parameters when triggering the workflow.

**Example:**

yaml

CopyEdit

on:

workflow\_dispatch:

inputs:

branch:

description: 'Branch to deploy'

required: true

default: 'main'

**17. How do you cache dependencies in GitHub Actions to speed up builds?**

**Answer:**  
You can use the actions/cache action to cache dependencies and speed up subsequent runs of your workflows. The cache is typically used to store dependencies like node\_modules, maven caches, or pip dependencies, preventing them from being reinstalled every time.

**Example:**

yaml

CopyEdit

steps:

- name: Cache Node modules

uses: actions/cache@v2

with:

path: ~/.npm

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

restore-keys: |

${{ runner.os }}-node-modules-

- name: Install dependencies

run: npm install

**18. How do you pass data between jobs in GitHub Actions?**

**Answer:**  
You can pass data between jobs using **artifacts** or **outputs**.

* **Artifacts** are used when you want to store files and pass them to other jobs.
* **Outputs** are used when you want to pass values (such as strings or variables) between jobs.

**Example using outputs:**

yaml

CopyEdit

jobs:

build:

runs-on: ubuntu-latest

outputs:

build\_version: ${{ steps.version.outputs.version }}

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set build version

id: version

run: echo "::set-output name=version::1.0.0"

deploy:

runs-on: ubuntu-latest

needs: build

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Deploy build version

run: echo "Deploying version ${{ needs.build.outputs.build\_version }}"

**19. What are the benefits of using GitHub Actions for CI/CD?**

**Answer:**

* **Integration with GitHub**: GitHub Actions is natively integrated into GitHub, making it easier to manage and track CI/CD pipelines alongside your code.
* **Customization**: You can create custom workflows, actions, and jobs to fit your specific CI/CD needs.
* **Free for public repositories**: GitHub Actions offers free minutes for public repositories, making it an excellent choice for open-source projects.
* **Matrix builds**: You can easily run tests or builds across multiple environments (e.g., different versions of Node.js or different OS platforms).
* **Extensive marketplace**: GitHub Actions has a marketplace of pre-built actions that you can use to speed up your workflows.

**20. How do you make a GitHub Action run conditionally?**

**Answer:**  
You can use the if condition to control whether a step or job runs based on the outcome of previous steps or other conditions.

**Example using if:**

yaml

CopyEdit

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Build project

run: npm run build

deploy:

runs-on: ubuntu-latest

if: success() # Deploy only if the previous job was successful

steps:

- name: Deploy to production

run: ./deploy.sh

**21. What is the continue-on-error key in GitHub Actions?**

**Answer:**  
The continue-on-error key allows you to define whether a step failure should cause the entire job to fail. If set to true, a failed step will not stop the workflow from continuing.

**Example:**

yaml

CopyEdit

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Run Tests

run: npm test

continue-on-error: true # This will allow the job to continue even if tests fail

**22. What is the purpose of the matrix strategy in GitHub Actions?**

**Answer:**  
The matrix strategy allows you to run multiple configurations of a job in parallel, such as testing different operating systems, versions of a language, or other dependencies.

**Example:**

yaml

CopyEdit

jobs:

build:

runs-on: ubuntu-latest

strategy:

matrix:

node-version: [12, 14, 16]

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up Node.js

uses: actions/setup-node@v2

with:

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

- name: Install dependencies

run: npm install

- name: Run tests

run: npm test

**23. How do you schedule a GitHub Actions workflow to run at specific intervals?**

**Answer:**  
You can use the schedule trigger in GitHub Actions with cron syntax to schedule workflows to run at fixed times or intervals.

**Example:**

yaml

CopyEdit

on:

schedule:

- cron: '0 0 \* \* \*' # Runs daily at midnight

**24. What is the role of the setup-node action in GitHub Actions?**

**Answer:**  
The setup-node action is used to set up a specific version of Node.js in the GitHub Actions environment. This allows you to run your Node.js-related tasks (like npm install, npm test) in a controlled environment with the desired Node.js version.

**Example:**

yaml

CopyEdit

steps:

- name: Set up Node.js

uses: actions/setup-node@v2

with:

node-version: '14'

**25. How do you use GitHub Actions for deployments?**

**Answer:**  
GitHub Actions can be used to automate the deployment of your application to various platforms (AWS, Azure, Google Cloud, etc.). You can configure a workflow to deploy your code after passing tests or after the code is merged into the main branch.

**Example for AWS deployment:**

yaml

CopyEdit

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up AWS CLI

uses: aws-actions/configure-aws-credentials@v1

with:

aws-access-key-id: ${{ secrets.AWS\_ACCESS\_KEY\_ID }}

aws-secret-access-key: ${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

aws-region: 'us-east-1'

- name: Deploy to AWS

run: aws s3 sync ./dist s3://my-bucket/

**26. How can you trigger a GitHub Action workflow manually?**

**Answer:**  
You can use the workflow\_dispatch trigger to manually start a workflow. This can be done through the GitHub UI or through the GitHub API, and you can also pass input parameters to the workflow.

**Example:**

yaml

CopyEdit

on:

workflow\_dispatch:

inputs:

environment:

description: 'Deployment environment'

required: true

default: 'staging'

**27. Explain how to handle multi-stage deployments using GitHub Actions.**

**Answer:**  
Multi-stage deployments can be handled by creating multiple jobs that represent different stages, such as build, test, staging, and production. Each job can be configured to run only after the successful completion of the previous one using the needs keyword.

**Example:**

yaml

CopyEdit

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Build project

run: npm run build

test:

runs-on: ubuntu-latest

needs: build

steps:

- name: Run tests

run: npm test

deploy:

runs-on: ubuntu-latest

needs: test

steps:

- name: Deploy to production

run: ./deploy.sh

**28. What are the best practices when using GitHub Actions?**

**Answer:**

* **Use secrets for sensitive data**: Avoid hardcoding API keys or credentials in your workflow files; use GitHub Secrets.
* **Limit permissions**: Restrict permissions for workflows to minimize security risks.
* **Use matrix builds**: Leverage matrix strategies to run your jobs on multiple platforms or configurations to ensure compatibility.
* **Use caching**: Cache dependencies to speed up your builds.
* **Write clean, modular workflows**: Break down complex workflows into smaller, reusable steps or actions.
* **Keep workflows DRY (Don't Repeat Yourself)**: Use reusable workflows or actions to avoid code duplication.

These questions cover additional fundamental topics that are often tested in interviews and give a solid understanding of GitHub Actions.

4o

**You said:**

name: Deploy on: [push, workflow\_dispatch] jobs: test: runs-on: ubuntu-latest steps: - name: Checkout code uses: actions/checkout@v4 # You can use @v4 directly here instead of v3 # - name: Print GitHub context and clone repo # run: | # echo "${{ toJson(github) }}" # echo "Cloning the repo" # git clone https://github.com/${{ github.repository }}.git - name: isntall nodejs uses: action/setup-node@v4 with: node-version: 20 - corect ident and explain

**ChatGPT said:**

​