# Using Actions in Workflows

Actions in a workflow are reusable units of code that perform specific tasks within the context of a GitHub Actions workflow. These actions encapsulate a set of commands or operations and can be invoked as individual steps within a workflow.

Actions can be used to automate various aspects of your software development and deployment processes. They can perform tasks such as building, testing, deploying, linting, and more. Actions are typically shared and distributed through GitHub Marketplace or stored within repositories.

## Key characteristics of actions

**Modularity:** Actions are modular and self-contained, making it easy to reuse them across different workflows and repositories.

**Customizability:** Actions can be customized through inputs, outputs, and parameters, allowing users to tailor their behavior to specific requirements.

**Flexibility:** Actions can be written in various programming languages and technologies, such as JavaScript, Python, Docker, and shell scripts, providing flexibility to developers.

**Community-driven:** GitHub has a vast ecosystem of community-contributed actions available in the GitHub Marketplace, covering a wide range of use cases and scenarios.

## Actions used in Workflow

**Checkout Code:**

```yaml
- name: Checkout Code
  uses: actions/checkout@v4
```

- This action is used to clone your repository's codebase into the runner environment where your workflow is executing.
- It is represented by the actions/checkout@v4 reference, which specifies the version (v4) of the actions/checkout action to use.
- This action is essential at the beginning of most workflows to ensure that subsequent steps have access to the repository's code.

**Setup Java:**

```yaml
- name: Setup Java
  uses: actions/setup-java@v4
```

```
    with:
      distribution: 'adopt'
      java-version: '11'
```

- This action is used to set up a specific version of Java on the runner environment.
- It is represented by the actions/setup-java@v4 reference, specifying the version (v4) of the actions/setup-java action.
- In this case, the action is configured to set up Java version 11.
- This action is commonly used in Java-related workflows to ensure the correct Java version is available for building and testing Java applications.

## Testing Actions in Workflows

To create and test the workflow described in the provided YAML configuration, follow these steps:

**1. Create a Workflow File:**

- Create a new file named `06-01-Actions.yml` in the `.github/workflows` directory of your repository.

  ```
  name: 06-01-Actions

  on: workflow_dispatch

  jobs:
    build:
      runs-on: ubuntu-latest
      steps:
        - name: Checkout Code
          uses: actions/checkout@v4

        - name: Setup Java
          uses: actions/setup-java@v4
          with:
            distribution: 'adopt'
            java-version: '11'

        - name: Build with Maven
          run: mvn clean install -Dmaven.test.skip=true

        - name: Run tests
          run: mvn test

        - name: I will be skipped
          if: ${{ success() }}
          run: echo "I will print if previous steps succeed."
        - name: I will execute
          if: ${{ failure() }}
          run: echo "I will print if any previous step fails."
  ```

- Copy and paste the provided YAML configuration into this file.

**2. Understanding the Workflow:**

- The workflow is triggered manually (on: workflow_dispatch), meaning it will run when manually triggered by a user.
- It defines a single job named build that runs on an Ubuntu environment (runs-on: ubuntu-latest).
- The job consists of several steps:

**3. Checkout Code:**

- This step clones your repository's codebase into the runner environment using the `actions/checkout@v4` action.

**4. Setup Java:**

- This step sets up Java version 11 in the runner environment using the `actions/setup-java@v4` action.
- It specifies the distribution as `adopt` and the Java version as `11`.

**5. Build with Maven:**

This step runs the Maven command to clean the project and install dependencies. It skips running tests using the -Dmaven.test.skip=true flag.

**6. Run Tests:**

- This step runs the Maven command to execute tests.

**7. Conditional Steps:**

- These steps are conditional based on the outcome of the previous steps:
- "I will be skipped": This step will only execute if all previous steps succeed.
- "I will execute": This step will only execute if any previous step fails.

**8. Testing the Workflow:**

- Commit and push the workflow file (`06-01-Actions.yml`) to your repository.
- Go to the "Actions" tab in your GitHub repository to view the workflow.
- Manually trigger the workflow by clicking the "Run workflow" button and selecting the workflow named "06-01-Actions".
- Monitor the workflow execution and check the output of each step to ensure it completes successfully.