+ Nouveau

Partager

on git action how create a karate report

Sources

Integrating Karate tests into GitHub workflows - Packt Subscription
subscription.packtpub · 1

Testing a Java Spring Boot REST API with Karate - Semaphore CI
semaphoreci · 2

Running API tests
YouTube
youtube · 3

Réponse

To create a Karate test report as an artifact in a GitHub Actions workflow, you can follow these steps:

1. Run your Karate tests using Maven or the Karate CLI.
2. Use the `` `actions/upload-artifact@v3` `` action to upload the Karate report directory as an artifact.

Here's an example workflow that demonstrates this:

Poser un suivi

```yaml
name: Karate Tests

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:

  test:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3
    - name: Set up JDK
      uses: actions/setup-java@v3
      with:
        java-version: '11'
        distribution: 'adopt'

    - name: Run Karate Tests
      run: mvn test -Dtest=KarateTests

    - name: Upload Karate Report
      uses: actions/upload-artifact@v3
      with:
        name: karate-report
        path: target/karate-reports

    - name: Upload Karate Logs
      uses: actions/upload-artifact@v3
      with:
        name: karate-logs
        path: target/karate.log
```

In this example:

- The `Run Karate Tests` step runs the Karate tests using Maven.
- The `Upload Karate Report` step uploads the `target/karate-reports

artifact named `karate-report`. [4]

- The `Upload Karate Logs` step uploads the `target/karate.log` file as an artifact named `karate-logs`. [4]

You can view and download the artifacts from the Actions tab in your GitHub repository after the workflow run completes.

Partager

---

i need to start aa docker compose with git action before start karate compose

Sources

How can I wait for the container to be healthy in GitHub action?
stackoverflow · 1

Use docker and docker compose inside the action - Gitea
forum.gitea · 2

Using GitHub Acti
Deploy Docker ...
earthly · 3

Réponse

To start a Docker Compose setup with Karate tests in a GitHub Actions workflow, you can follow these steps:

1. Check out your repository code:

```yaml
- uses: actions/checkout@v3
```

2. Set up Java environment:

```yaml
- name: Set up JDK
  uses: actions/setup-java@v3
  with:
    java-version: '11'
    distribution: 'adopt'
```

3. Build and start Docker containers using Docker Compose:

```yaml
- name: Start Docker Compose
  run: docker-compose up -d
```

This will build and start all the containers defined in your `docker-compose.yml` file.

4. Once the containers are up, you can run your Karate tests:

```yaml
- name: Run Karate Tests
  run: mvn test -Dtest=KarateTests
```

This assumes your Karate tests are part of your Maven project.

Here's an example GitHub Actions workflow that puts it all together:

```yaml
name: Karate Tests

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:

  test:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Set up JDK
      uses: actions/setup-java@v3
      with:
        java-version: '11'
        distribution: 'adopt'

    - name: Start Docker Compose
      run: docker-compose up -d

    - name: Run Karate Tests
      run: mvn test -Dtest=KarateTests
```

This workflow will:

1. Check out the code
2. Set up Java 11 environment
3. Start Docker containers using `docker-compose up -d`
4. Run Karate tests using Maven

Make sure your `docker-compose.yml` file defines all the services/containers required for your Karate tests to run successfully. [1] [4]

Partager

but the git action unkonwn command docker compose

Sources

Run Docker Compose + in Github Action

gist.github · 1

context default compose reports unknown docker command ... - GitHub

github · 2

Use docker and dc action - Gitea

forum.gitea · 3

Réponse

To resolve the "unknown docker command: compose" error when using Docker Compose in a GitHub Actions workflow, you need to use the updated Docker Compose CLI syntax introduced in version 2.0.

Instead of running `docker-compose up` or `docker-compose down`, you should use:

```yaml
- name: Start Docker Compose
  run: docker compose up -d

- name: Stop Docker Compose
  run: docker compose down
```

The new syntax separates the `docker` command from the `compose` subcommand. [1]

Here's an example GitHub Actions workflow that demonstrates the correct usage:

yaml

```
name: Docker Compose

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:

  test:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Start Docker Compose
      run: docker compose up -d

    # Your test steps here

    - name: Stop Docker Compose
      if: always()
      run: docker compose down
```

This workflow will:

1. Check out the code
2. Start the Docker Compose setup using `docker compose up -d`
3. Run your tests (you need to add these steps)
4. Stop and remove the Docker Compose containers using `docker compose down`

The `if: always()` ensures that the `docker compose down` step runs even if the previous steps fail, to clean up the running containers. [1]

Partager

i need to add a docker file to do all that

Réponse

Sure, you can create a Dockerfile to encapsulate the setup and execution of your Karate tests within a Docker container. Here's an example of how you can structure your Dockerfile and GitHub Actions workflow:

First, create a Dockerfile in your project's root directory with the following content:

```dockerfile
# Use a Java base image
FROM openjdk:11

# Set the working directory
WORKDIR /app

# Copy the project files
COPY . /app

# Install Maven
RUN apt-get update && apt-get install -y maven

# Build the project
RUN mvn clean package

# Run the Karate tests
CMD ["mvn", "test", "-Dtest=KarateTests"]
```

This Dockerfile:

1. Uses the `openjdk:11` base image for Java 11.
2. Sets the working directory to `/app`.
3. Copies the project files to the container.
4. Installs Maven.
5. Builds the project using `mvn clean package`.
6. Runs the Karate tests using `mvn test -Dtest=KarateTests`.

Next, update your GitHub Actions workflow to build and run the Docker container:

```yaml
name: Karate Tests

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:

  test:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Build Docker Image
      run: docker build -t karate-tests .

    - name: Run Docker Container
      run: docker run karate-tests

    - name: Upload Karate Report
      uses: actions/upload-artifact@v3
      with:
        name: karate-report
        path: target/karate-reports

    - name: Upload Karate Logs
      uses: actions/upload-artifact@v3
      with:
        name: karate-logs
        path: target/karate.log
```

This workflow:

1. Checks out the code.
2. Builds a Docker image named `karate-tests` using the Dockerfile.
3. Runs the `karate-tests` Docker container, which executes the Kar

4. Uploads the Karate report and logs as artifacts.

Make sure to update the `path` values in the `actions/upload-artifact` steps to match the locations of your Karate report and log files within the Docker container.

With this setup, your Karate tests will run within a Docker container, and the reports will be available as artifacts in your GitHub Actions workflow run.

Partager

---

can i add the start of compose on docker file

Réponse

Yes, you can include the Docker Compose setup in your Dockerfile. Here's an example of how you can modify the Dockerfile to start Docker Compose before running your Karate tests:

```dockerfile
# Use a Java base image
FROM openjdk:11

# Install Docker
RUN apt-get update && apt-get install -y docker.io

# Set the working directory
WORKDIR /app

# Copy the project files and Docker Compose file
COPY . /app

# Install Maven
RUN apt-get update && apt-get install -y maven

# Build the project
RUN mvn clean package

# Start Docker Compose
RUN docker compose up -d

# Run the Karate tests
CMD ["mvn", "test", "-Dtest=KarateTests"]
```

Here's what's different in this Dockerfile:

1. It installs Docker using `apt-get install -y docker.io`.
2. It copies the Docker Compose file (`docker-compose.yml`) along with the project files to the container using `COPY . /app`.
3. It starts the Docker Compose setup using `docker compose up -d` before running the Karate tests.

Make sure you have a `docker-compose.yml` file in your project's root directory that defines the services/containers required for your Karate tests.

With this setup, your GitHub Actions workflow can be simplified as follows:

```yaml
name: Karate Tests

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:

  test:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Build Docker Image
      run: docker build -t karate-tests .

    - name: Run Docker Container
      run: docker run karate-tests

    - name: Upload Karate Report
      uses: actions/upload-artifact@v3
      with:
        name: karate-report
        path: target/karate-reports

    - name: Upload Karate Logs
      uses: actions/upload-artifact@v3
      with:
        name: karate-logs
        path: target/karate.log
```

This workflow:

1. Checks out the code.
2. Builds a Docker image named `karate-tests` using the Dockerfile.
3. Runs the `karate-tests` Docker container, which starts Docker Co
   executes the Karate tests.

4. Uploads the Karate report and logs as artifacts.

By encapsulating the Docker Compose setup and Karate test execution in a single Docker container, you can simplify your GitHub Actions workflow and ensure a consistent environment for running your tests.

Partager

can i add karate report to docker file

Sources

How Can I run Karate Api tests via docker and then expose the ...	Run Karate tests from Docker - YouTube	Karate UI
stackoverflow · 1	youtube · 2	karatelabs.gith

Réponse

Yes, you can add the generation of the Karate test report to your Dockerfile. Based on the search results [1] and [3], here's how you can modify your Dockerfile to generate and expose the Karate test report:

dockerfile

```
# Use a Java base image
FROM openjdk:11

# Install Docker
RUN apt-get update && apt-get install -y docker.io

# Set the working directory
WORKDIR /app

# Copy the project files and Docker Compose file
COPY . /app

# Install Maven
RUN apt-get update && apt-get install -y maven

# Build the project
RUN mvn clean package

# Start Docker Compose
RUN docker compose up -d

# Run the Karate tests and generate report
RUN mvn test -Dkarate.options="--reports-dir=/app/target/karate-reports"

# Expose the report directory
VOLUME /app/target/karate-reports

# Keep the container running
CMD ["tail", "-f", "/dev/null"]
```

Here's what's different:

1. The `mvn test` command includes the `-Dkarate.options="--reports-dir=/app/target/karate-reports"` option to generate the Karate test report in the `/app/target/karate-reports` directory.
2. The `VOLUME /app/target/karate-reports` instruction exposes the `/app/target/karate-reports` directory as a volume, allowing you to access the report from outside the container.
3. The `CMD ["tail", "-f", "/dev/null"]` command keeps the conta

indefinitely, so you can access the report after the tests have completed.

With this Dockerfile, when you run the container, the Karate tests will execute, and the report will be generated in the `/app/target/karate-reports` directory inside the container. You can then access this directory from your host machine using Docker volume mapping.

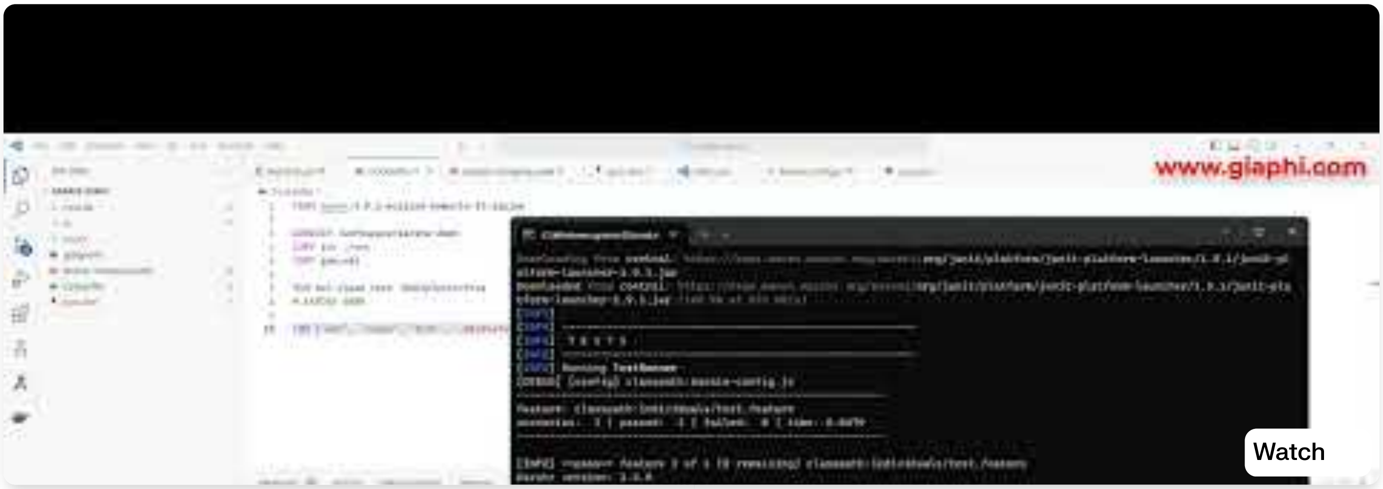
For example, if you run the container with:

```
docker run -v /path/on/host:/app/target/karate-reports karate-tests
```

The `/app/target/karate-reports` directory inside the container will be mapped to `/path/on/host` on your host machine, allowing you to access the Karate test report from your host.

Make sure to replace `/path/on/host` with the desired path on your host machine where you want to access the report. [1] [3]

Partager



please the pom file

Réponse

Sure, here's an example `pom.xml` file that you can use for your Karate tests project:

xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>karate-tests</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>11</java.version>
        <karate.version>1.3.1</karate.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>com.intuit.karate</groupId>
            <artifactId>karate-apache</artifactId>
            <version>${karate.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.intuit.karate</groupId>
            <artifactId>karate-junit5</artifactId>
            <version>${karate.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <testResources>
            <testResource>
                <directory>src/test/java</directory>
                <excludes>
                    <exclude>**/*.java</exclude>
                </excludes>
            </testResource>
        </testResources>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.10.1</version>
                <configuration>
                    <encoding>UTF-8</encoding>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

This `pom.xml` file:

1. Sets the project's `groupId`, `artifactId`, and `version`.
2. Defines the `java.version` and `karate.version` properties.
3. Includes the `karate-apache` and `karate-junit5` dependencies for running Karate tests with JUnit 5.
4. Configures the `maven-compiler-plugin` to use Java 11.
5. Includes the `src/test/java` directory as a test resource, excluding Java files (since Karate tests are written in a different format).

Make sure to update the `groupId`, `artifactId`, and `version` as per your project requirements.

You can place your Karate test files (e.g., `*.feature` files) in the `src/test/java` directory, and they will be included as test resources by Maven.

To run the Karate tests, you can use the following Maven command:

```
mvn test -Dtest=KarateTests
```

This will execute all tests in the `KarateTests` class (or any other class that runs your Karate tests).

Partager

for java 21

Sources

Maven in 5 Minutes
maven.apache · 1

Maven Build Phases and Basic Maven Commands - GeeksforGeeks
geeksforgeeks · 2

20+ Maven Comm
Sheet) - DigitalOc
digitalocean ·

Réponse

To configure Maven to use a specific Java version, you can set the `maven.compiler.source` and `maven.compiler.target` properties in your `pom.xml` file. Here's how you can do it for Java 21:

1. Open your `pom.xml` file.
2. Add the following properties inside the `<properties>` section:

```xml
<properties>
    <maven.compiler.source>21</maven.compiler.source>
    <maven.compiler.target>21</maven.compiler.target>
</properties>
```

3. Ensure you have the `maven-compiler-plugin` configured in the `<build><plugins>` section, with a version that supports Java 21:

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.10.1</version> <!-- Or a newer version that supports
        </plugin>
    </plugins>
</build>
```

Your `pom.xml` should look something like this:

xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven
    <modelVersion>4.0.0</modelVersion>

    <!-- Your project details -->

    <properties>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.10.1</version>
            </plugin>
        </plugins>
    </build>
</project>
```

After making these changes, Maven will use Java 21 to compile your project's source code.

Note that you need to have Java 21 installed on your system for this to work. If you don't have Java 21 installed, you can download it from the official Oracle website or use an OpenJDK distribution that provides Java 21.

Partager

---

karate apach 1.4.1 not exist

Sources

karate-demo build fails to find packages · Issue #463 - GitHub
github · 1

Karate | Test Automation Made Simple.
karatelabs.github · 2

1.0 upgrade guide
GitHub
github · 3

Réponse

Based on the search results, it seems that starting from Karate version 1.0, the `karate-apache` dependency no longer exists. Instead, there is now a single dependency called `karate-core` that includes the Apache HTTP Client by default.

Here are the key points regarding the changes in Karate 1.0 and above:

- The `karate-apache` and `karate-jersey` dependencies have been removed. There is now only one core dependency `karate-core` [4].
- If you are using JUnit 5, you only need to include the `karate-junit5` dependency in your `pom.xml` [5].
- If you are using an older version of JUnit, you can use `karate-junit4` instead.
- If you don't want to use JUnit at all, you can just use `karate-core`.
- The HTML reports and other artifacts are now generated in the `target/karate-reports` directory instead of `target/surefire-reports` [3].
- The `@KarateOptions` annotation is deprecated and will be removed in a future version. You should switch to using the `Runner` class instead [3].

So in summary, to upgrade to Karate 1.4.1 (latest version at the time of writing), you need to update your `pom.xml` to include only the `karate-core` and `karate-junit5` (or `karate-junit4`) dependencies, and update any paths referring to the report locations.

For example:

```xml
<dependency>
    <groupId>com.intuit.karate</groupId>
    <artifactId>karate-core</artifactId>
    <version>1.4.1</version>
</dependency>
<dependency>
```