how run docker compose via karate -config

Sources

A prime example of Karate tests run within Dockerized containers
github • 1

Component Testing of Frontends: Karate Netty Project - DZone
dzone • 2

Spinning up Karate test cl Overflow
stackoverflow • 3

Réponse

To run Docker Compose via Karate with a configuration file, you can follow these steps:

1. Create a `karate-config.js` file with the necessary configuration properties, such as the Docker Compose command and any environment variables needed. For example:

```javascript
function() {
  var env = karate.env; // get system property 'karate.env'
  karate.log('karate.env system property was:', env);
  if (!env) {
    env = 'dev'; // a custom 'intelligent' default
  }
  var config = { // base config JSON
    dockerComposeCommand: 'docker-compose up --exit-code-from e2e-tests',
    env: env
  }
  if (env == 'dev') {
    // customize for dev environment
  } else if (env == 'e2e') {
    // customize for e2e env
  }
  return config;
}
```

2. In your Karate test script (e.g., `my-test.feature`), you can call the Docker Compose command using the `karate.call` function and pass the configuration file as an argument:

```cucumber
Feature: My Test

Background:
  * configure driverBinary = { docker: { executable: '/usr/local/bin/docker' } }
  * def config = callonce read('classpath:karate-config.js')

Scenario: Run Docker Compose
  * docker().start(config.dockerComposeCommand)
  * match docker.logs('#regex').count > 0
```

In this example, the `configure driverBinary` line sets the path to the Docker executable. The `def config` line reads the configuration from the `karate-config.js` file using `callonce read('classpath:karate-config.js')`.

The `docker().start(config.dockerComposeCommand)` line runs the Docker Compose command specified in the configuration file. You can then perform assertions or additional steps as needed, such as checking the Docker logs using `match docker.logs('#regex').count > 0`. [1] [2]

Poser un suivi Pro

Partager

an other méthode

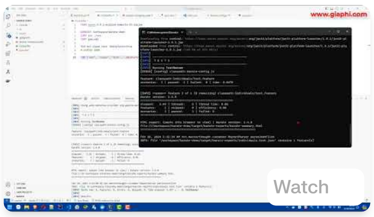
Sources

A prime example of Karate tests run within Dockerized containers
github · 1

Docker · karatelabs/karate Wiki - GitHub
github · 2

Karate UI
karatelabs.github · 3

Réponse



There is no clear mention of another method to run Docker Compose via Karate in the given search results. The main approach described is:

1. Create a `karate-config.js` file with the Docker Compose command and any required configuration.
2. In the Karate test script, read the configuration using `callonce read('classpath:karate-config.js')`.
3. Start the Docker Compose setup by calling `docker().start(config.dockerComposeCommand)` from the configuration. [1]

The search results do not provide any alternative methods beyond this approach of using a configuration file and the `docker().start()` function in Karate. If you need additional details or have a specific query, please provide more context.

Partager

i need to run my data base servevia docker compose into the application karate before start testing

Sources

A prime example of Karate tests run within Dockerized containers
github · 1

Docker · karatelabs/karate Wiki - GitHub
github · 2

Karate UI
karatelabs.github · 3

Réponse

To run your database server via Docker Compose and integrate it with the Karate testing framework before starting your tests, you can follow these steps:

1. Create a `docker-compose.yml` file that defines your database service and any other required services. For example:

```yaml
version: '3'
services:
  db:
    image: postgres:13
```

Pro

```
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
    ports:
      - "5432:5432"
```

This defines a PostgreSQL database service named `db`.

2. In your Karate test script (e.g., `my-test.feature`), you can start the Docker Compose setup before running your tests:

```cucumber
Feature: My Test

Background:
  * def dockerCompose = docker.compose('/path/to/docker-compose.yml')
  * dockerCompose.up()

Scenario: Run Tests
  # Your test scenarios here
  * def dbConfig = { uri: 'jdbc:postgresql://localhost:5432/mydb', user: 'myuser', password: 'mypa
  * def db = Java.type('com.myapp.DatabaseUtils').getConnection(dbConfig)
  # Use the db object to interact with the database

  # Clean up
  * dockerCompose.down()
```

In this example:

- `def dockerCompose = docker.compose('/path/to/docker-compose.yml')` creates a Docker Compose instance from the `docker-compose.yml` file.
- `dockerCompose.up()` starts the Docker Compose setup, which will run the database service.
- You can then create a database connection using the appropriate configuration and interact with the database as needed.
- After your tests, `dockerCompose.down()` stops and removes the Docker Compose setup. [1] [2]

By integrating Docker Compose with Karate, you can ensure that your database server is running and available before executing your tests. This approach allows you to test your application against a real database instance, making your tests more realistic and reliable.

Partager

and if i need to start it one time before all testes

Sources

Docker Compose Quickstart docs.docker.com · 1
If you want to run your services in the background, you can pass the -dflag (for "detached" mode) to docker compose upand use docker compose psto see what is currently running: $ docker compose up -d Starting...

Setup integration test environment GitHub
github · 2

Réponse

If you need to start the Docker Compose setup only once before running all tests, you can use the `@pytest.fixture(scope="session")` decorator in pytest. This will ensure that the Docker Compose environment is started before the test session begins and stopped after all tests have completed.

Here's an example of how you can implement this using the `pytest-docker-compose` plugin:

python

Pro

```
@pytest.fixture(scope="session")
def docker_compose_file(pytestconfig):
    return pytestconfig.rootpath / "docker-compose.yml"

@pytest.fixture(scope="session")
def docker_services(docker_compose_file):
    executor = DockerComposeExecutor(docker_compose_file)
    executor.up()
    yield executor
    executor.down()

def test_something(docker_services):
    # Use the docker_services fixture to interact with the Docker Compose setup
    # and run your tests
    pass
```

In this example:

1. The `docker_compose_file` fixture returns the path to your `docker-compose.yml` file.
2. The `docker_services` fixture uses the `DockerComposeExecutor` from `pytest-docker-compose` to start the Docker Compose setup before the test session begins (`executor.up()`).
3. The `yield executor` statement allows your tests to use the `docker_services` fixture and interact with the Docker Compose setup.
4. After all tests have completed, the `executor.down()` method stops and removes the Docker Compose setup.

In your test functions, you can use the `docker_services` fixture to interact with the Docker Compose setup, such as retrieving container logs, executing commands, or accessing service URLs.

By using the `scope="session"` for the `docker_services` fixture, you ensure that the Docker Compose environment is started only once before the entire test session and stopped after all tests have completed. This can save time and resources compared to starting and stopping the environment for each individual test. [1] [2] [5]

Partager

on karate config

Sources

karate docker compose implementation on linux os is possible or not?
stackoverflow • 1

A prime example of Karate tests run within Dockerized containers
github • 2

Karate UI
karatelabs.github • 3

Réponse

Based on the provided search results, here are the key points about using the `karate-config.js` file in Karate:

The `karate-config.js` file is a configuration file in Karate that allows you to:

1. Store global variables that can be accessed from an, ature file in your project.

Pro

Creating and Using `karate-config.js`

1. Create a new file named `karate-config.js` in the `src/test/java` folder of your project. [4]
2. In this file, define a JavaScript function that returns a JSON object containing your configuration properties:

```javascript
function fn() {
  var config = {
    name: "test123",
    baseURL: 'https://reqres.in/api'
  }

  // Environment-specific configurations
  var env = karate.env;
  if (env == 'qa') {
    config.baseURL = 'https://reqres.in/api/qa';
  } else if (env == 'dev') {
    config.baseURL = 'https://reqres.in/api/dev';
  }

  // Configure timeouts
  karate.configure('connectTimeout', 5000);
  karate.configure('readTimeout', 5000);

  return config;
}
```

3. You don't need to explicitly call this file; Karate reads it automatically before executing any scenario. [4]
4. You can pass the environment value (`karate.env`) from the test runner or command line to switch configurations:
   - From test runner: `System.setProperty("karate.env", "qa");`
   - From command line: `mvn test -Denv=qa` [4]
5. In your feature files, you can access the configuration values defined in `karate-config.js` directly, e.g., `baseURL`. [1] [4]

The `karate-config.js` file allows you to centralize and manage your test configurations, making it easier to switch between different environments or settings without modifying the test scripts themselves. [1] [2] [4]

Partager

i need to run my docker compose one time automatiquely begore all testes

Sources

Docker Compose Quickstart docs.docker.com · 1
If you want to run your services in the background, you can pass the -dflag (for "detached" mode) to docker compose upand use docker compose psto see what is currently running: $ docker compose up -d Starting...

Running tests in containers with do
JetThoughts
jetthoughts · 2

Réponse

To run your Docker Compose setup automatically once before all tests in Karate, you can use the `@Startup` annotation along with the `karate.fork()` method. Here's how you ca nplement it:

Pro

```javascript
function() {
  var env = karate.env; // get the environment
  karate.log('Initializing Docker Compose setup for env:', env);

  // Start Docker Compose
  var compose = karate.fork({
    command: 'docker-compose',
    args: ['-f', 'path/to/docker-compose.yml', 'up', '-d'],
    showOutput: true
  });

  // Wait for Docker Compose to complete
  compose.join();

  // Return configuration if needed
  return { /* config properties */ };
}
```

Replace `'path/to/docker-compose.yml'` with the actual path to your Docker Compose file.

2. In your `karate-config.js` file, call the `docker-compose.js` function using the `read` function:

```javascript
function() {
  var config = karate.read('classpath:docker-compose.js')();
  // Additional configuration if needed
  return config;
}
```

3. In your feature files, annotate the `Background` or a `Scenario` with `@Startup` to run the Docker Compose setup before executing the tests:

```cucumber
@Startup
Feature: My Feature

Background:
  * configure driverBinary = { docker: { executable: '/usr/local/bin/docker' } }
  * def config = callonce read('classpath:karate-config.js')

Scenario: Test scenario
  # Your test steps here
```

The `@Startup` annotation ensures that the `docker-compose.js` script is executed before any scenarios in the feature file. The `karate.fork()` method runs the Docker Compose command in a separate process, and the `join()` method waits for it to complete.

With this setup, your Docker Compose environment will be started automatically before running any tests in Karate. The environment will remain running until all tests have completed, and you can access any configuration properties returned by `docker-compose.js` through the `config` variable. [1] [2] [4]

Note: Make sure you have Docker and Docker Compose installed and accessible from the Karate test runner. Also, adjust the paths and configurations as per your project structure and requirements.

Partager

Pro

how can I automate running Docker Compose before my tests +

what is the best way to ensure Docker Compose services are up before running tests +

can I use a script to run Docker Compose and then my tests +