Testing Cloud Applications Without Breaking the Bank

Nordic Testing Days

Fernando Teixeira

2026

How would you test an application that depends heavily on cloud services? Do you have a specific strategy for testing it?

The same way that I test other applications.

Infrastructure costs!!

Off-Prem ▸ SaaS

Google Cloud (over)Run: How a free trial experiment ended with a $72,000 bill overnight

Billing budget? Free plan? All useless when buggy code went into overdrive

Tim Anderson
Thu 10 Dec 2020 // 14:38 UTC

The $15K Dev Account Disaster: A Real-World Case Study and Solution

Case Study: A Startup's $450,000 Google Cloud Bill – Lessons for Startups

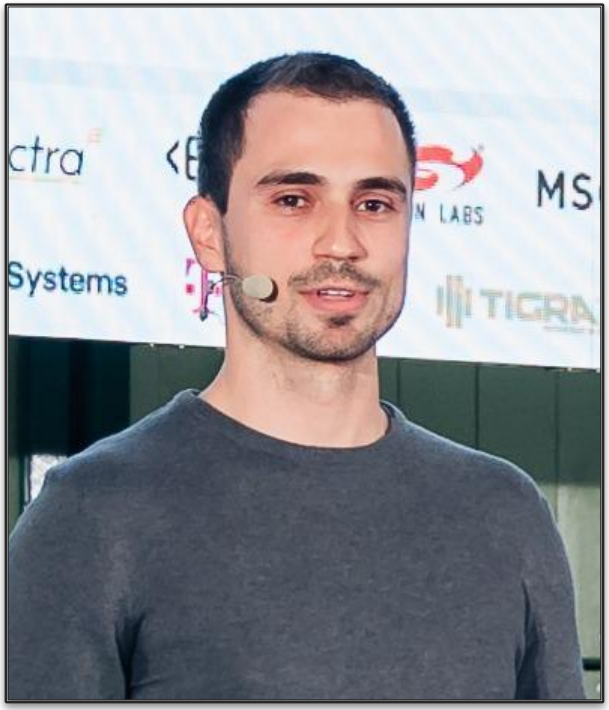
Updated on May 23, 2025 by Sash Ghosh

01

Introducing myself

Introducing myself

- Fernando Teixeira
- Lead QA Engineer in Verivox
- Living in Heidelberg, Germany
- Backend, microservices and DevOps
- Chess and cycling in my free time



02

Introduction to Cloud Computing

What is Cloud Computing?

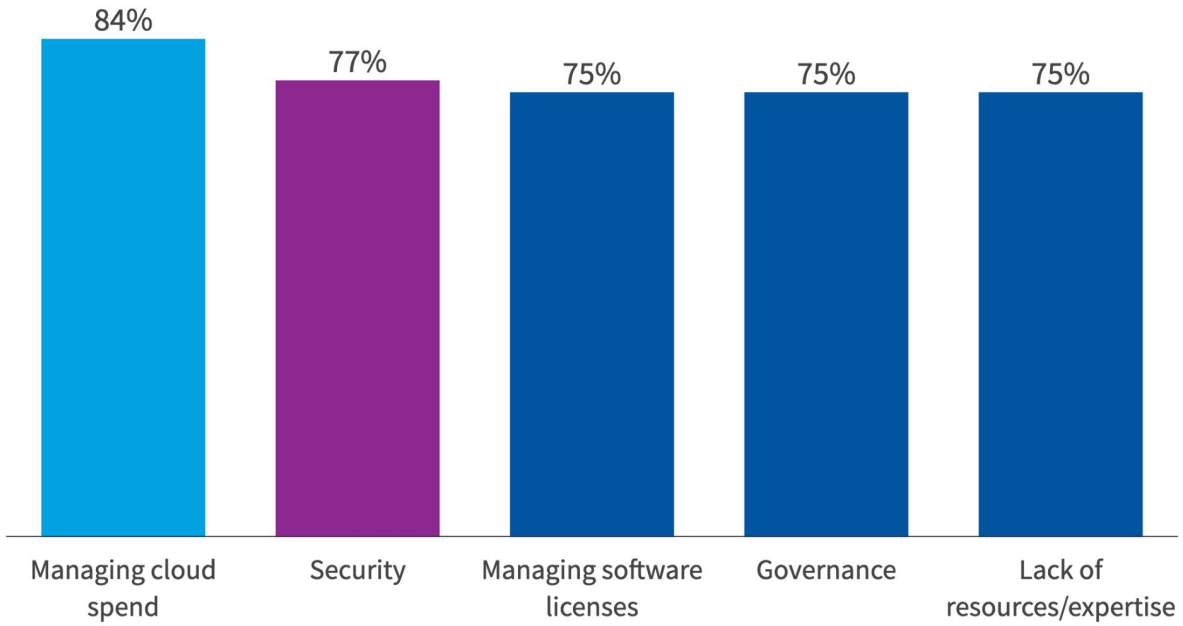
On-demand delivery of IT resources — such as servers, storage, databases, networking, and software — with pay-as-you-go pricing.

On-demand

Scalable

Pay-as-you-go

Top cloud challenges for all respondents



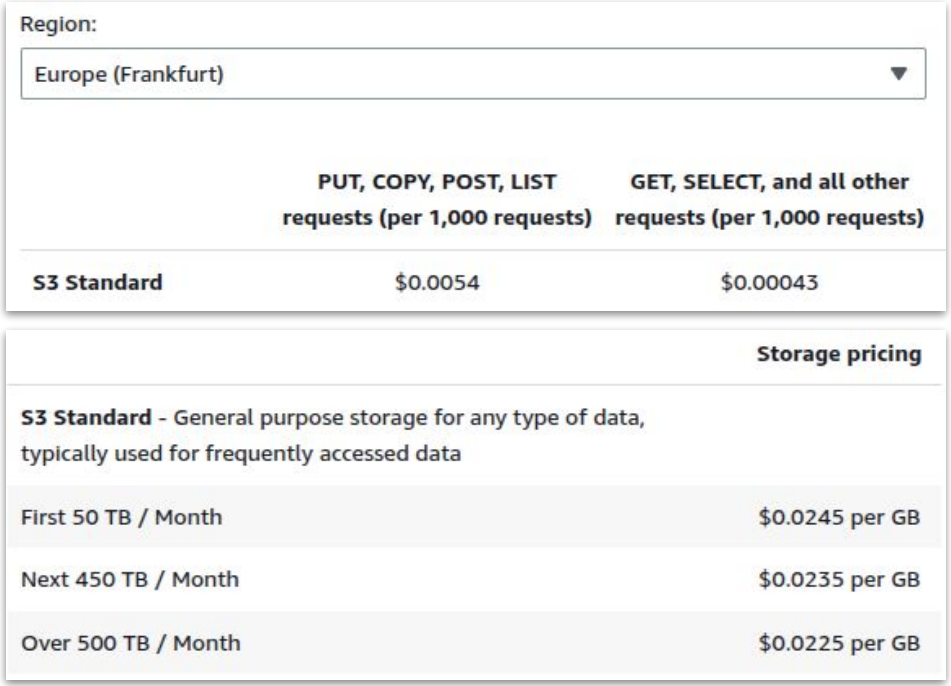
N=759

Source: Flexera 2025 State of the Cloud Report

flexera

Understanding how the cloud services are priced

- Most services are priced based on usage
- Examples:
  - Number of requests
  - Storage size usage
  - How long it is used



Region:

Europe (Frankfurt)

	PUT, COPY, POST, LIST requests (per 1,000 requests)	GET, SELECT, and all other requests (per 1,000 requests)
S3 Standard	$0.0054	$0.00043

	Storage pricing
S3 Standard - General purpose storage for any type of data, typically used for frequently accessed data	
First 50 TB / Month	$0.0245 per GB
Next 450 TB / Month	$0.0235 per GB
Over 500 TB / Month	$0.0225 per GB

Costs starts to rocket

- **Multiples Services being used:** databases, logs, file storages…
- **Multiple tests**
- **Multiple test executions**
- **Test environment instances**
- **Other hidden costs:** Networking is generally quite expensive

03

What are the options that we have to test those cloud applications?

Testing Cloud Applications - 3 Main Options

1. Use real cloud services
2. Use mocks
3. Use tools to emulate cloud services

Option 1 - Using Real Cloud Services

- **Pros ✔️**
  - High test confidence
  - Can detect configuration and infrastructure issues
  - Real word performance insights
- **Cons ❌**
  - High costs
  - Require cleanup strategies for test resources
  - Slow feedback loops

Option 2 - Using mocks

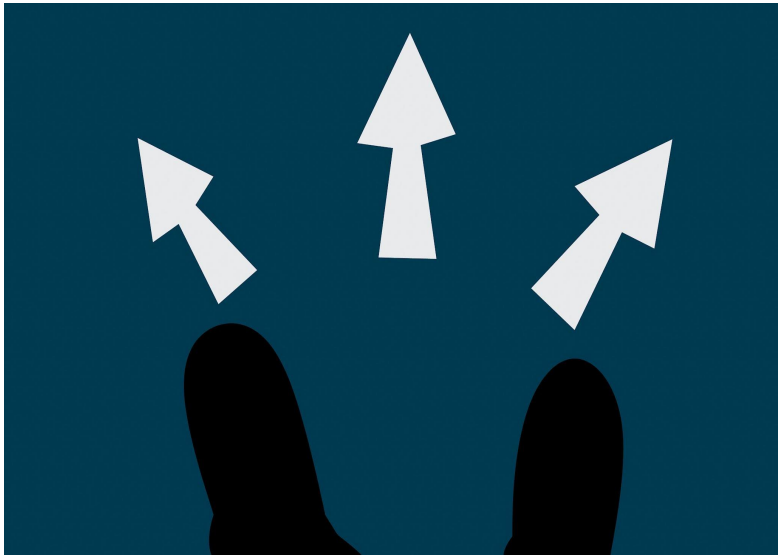
- **Pros ✔️**
  - It's free
  - Fast test run times
  - Tests are better isolated and more deterministic
- **Cons ❌**
  - Low confidence
  - Effort to keep mocks up to date

Option 3 - Use tools to emulate cloud services

- **Pros ✔️**
  - Accurate emulation of core cloud services behavior
  - Works locally and in CI/CD
  - Higher confidence than mocks
- **Cons ❌**
  - It is still just an emulation of services, so some behaviors can differ
  - The setup and configuration can be complex

Testing Cloud Applications - 3 Main Options

1. Use real cloud services
2. Use mocks
3. **Use tools to emulate cloud services**



04

Introducing Testcontainers and LocalStack

Introduction to LocalStack

- Cloud service emulator for development and testing
- Simulates cloud services on your own computer or CI/CD
- Works specifically for AWS services
- Easy cleanup of resources

💡 Think of it as your personal mini AWS in a Docker container.

LocalStack - Coverage Levels

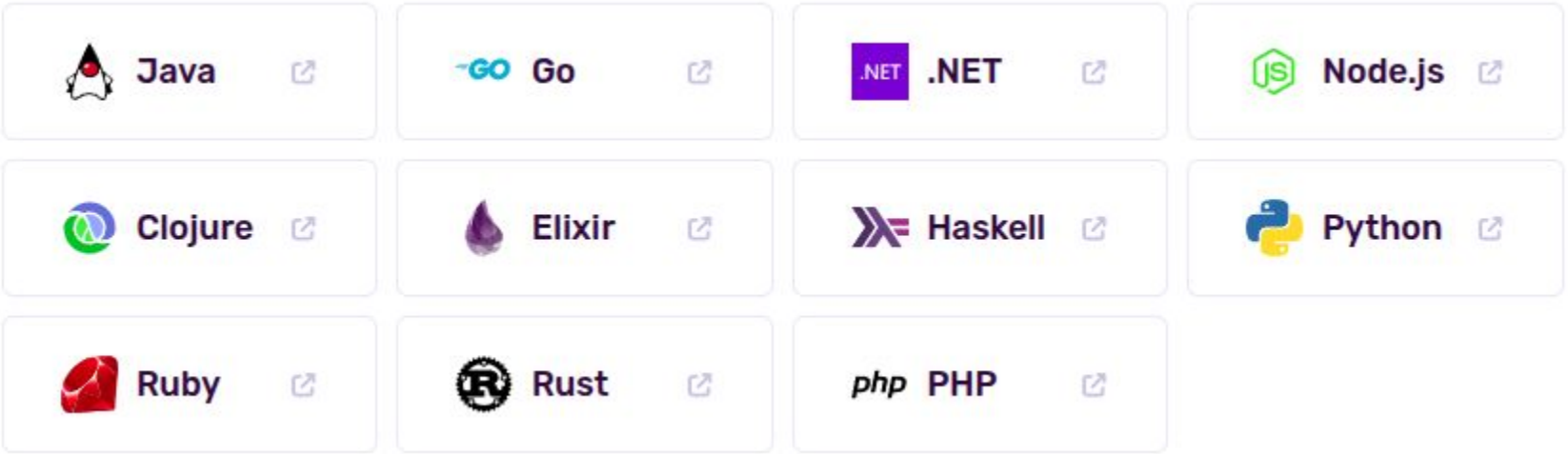
⭐⭐⭐⭐⭐	Feature fully supported by LocalStack maintainers; feature is guaranteed to pass all or the majority of tests
⭐⭐⭐⭐	Feature partially supported by LocalStack maintainers
⭐⭐⭐	Feature supports basic functionalities (e.g., CRUD operations)
⭐⭐	Feature should be considered unstable
⭐	Feature is experimental and regressions should be expected
-	Feature is not yet implemented

Reference:
https://docs.localstack.cloud/user-guide/aws/feature-coverage/
https://docs.localstack.cloud/references/coverage/

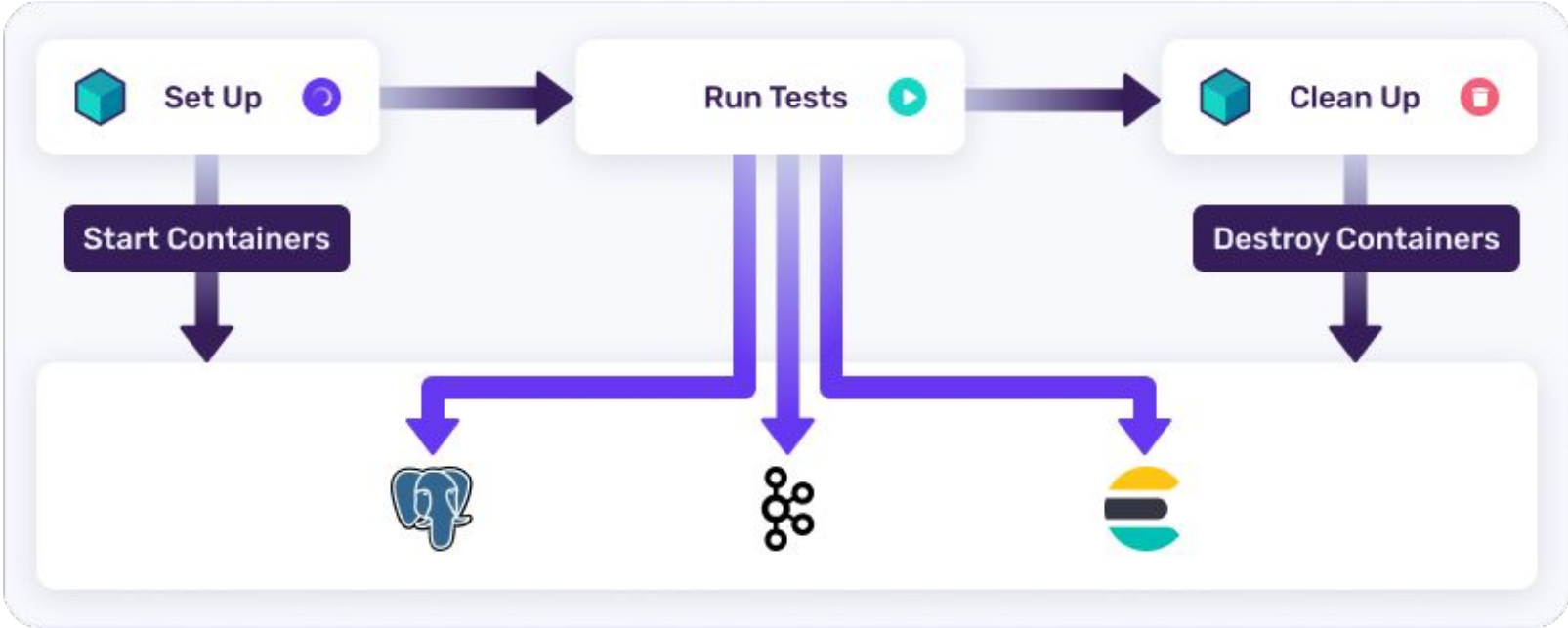
Introduction to Testcontainers

- Free and open-source tool
- Easily get your test dependencies wrapped in Docker containers inside your test code
- Official modules for easier integration, including **LocalStack**

Introduction to Testcontainers



Introduction to Testcontainers



Setup Example

```
@Testcontainers
public class TestContainersConfigurationForE2ETests {

   @Container
   protected static LocalStackContainer localStack = new
LocalStackContainer(DockerImageName.parse("localstack/localstack:4.0.3");
 //"LocalStackContainer" is a library from Testcontainers

   @BeforeAll
   static void beforeAll() throws IOException, InterruptedException {
       localStack.execInContainer("awslocal", "s3", "mb", "s3://" + "my-test-bucket");
       localStack.execInContainer(
               "awslocal",
               "sqs",
               "create-queue",
               "--queue-name",
               "my-test-queue"
       );
   }
}
```

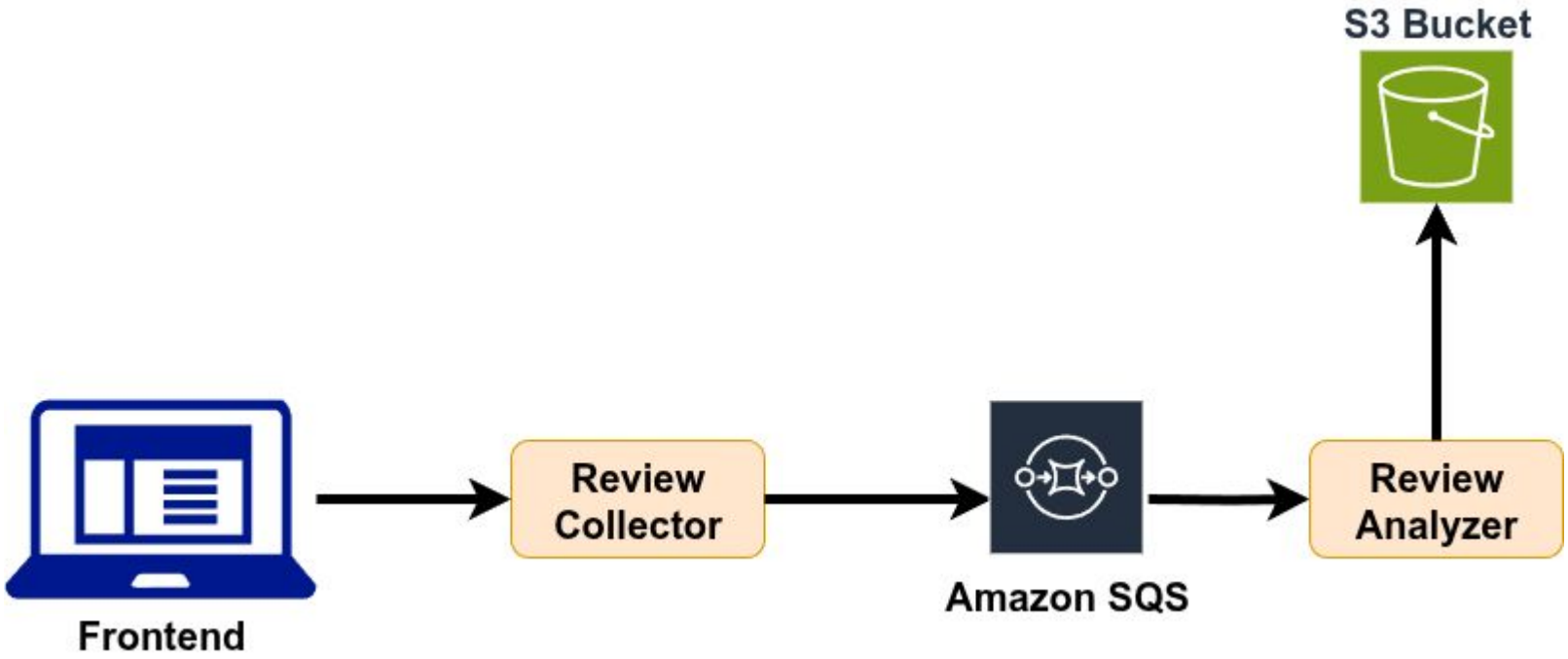
1 - Initialize Localstack container

2 - Create test prerequisites, including S3 Bucket and SQS Queue

05

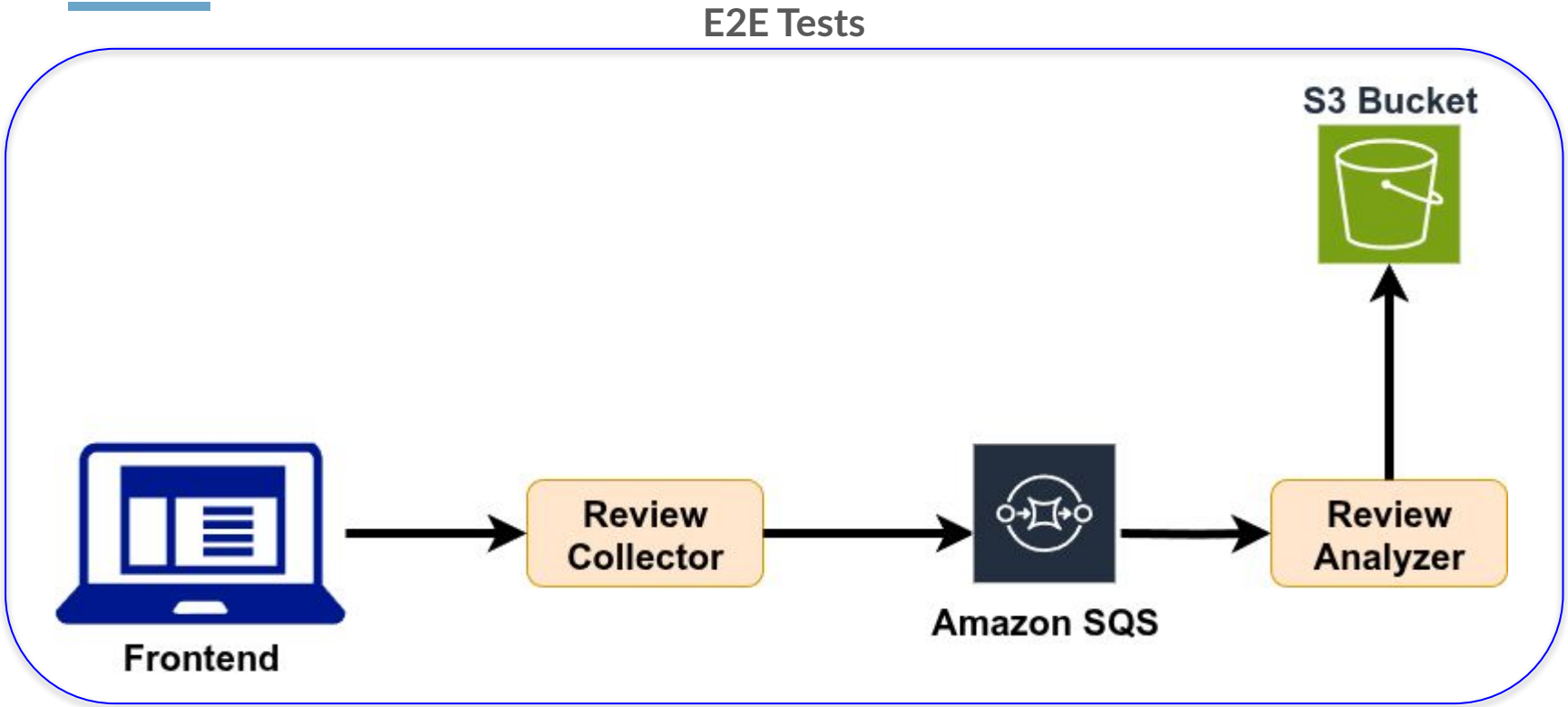
LocalStack and Testcontainers in action

Example - Review Analysis Application

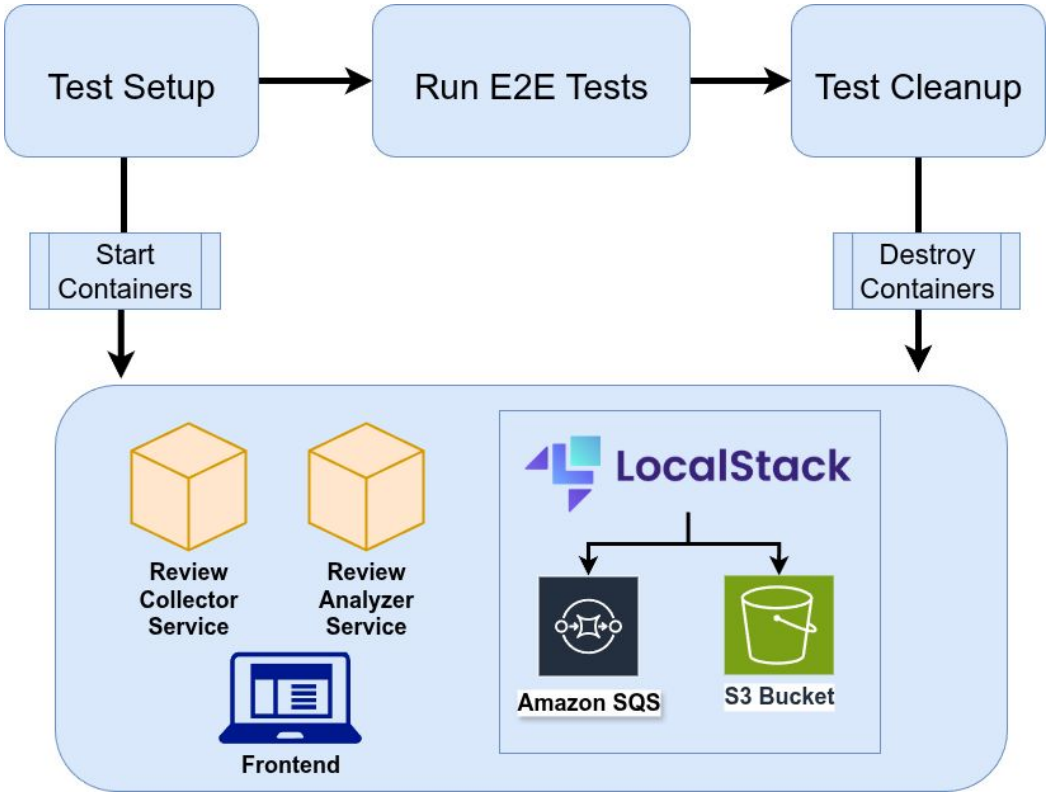


Reference: https://github.com/teixeira-fernando/Review-Analysis-Cloud-Microservices

Example - Review Analysis Application

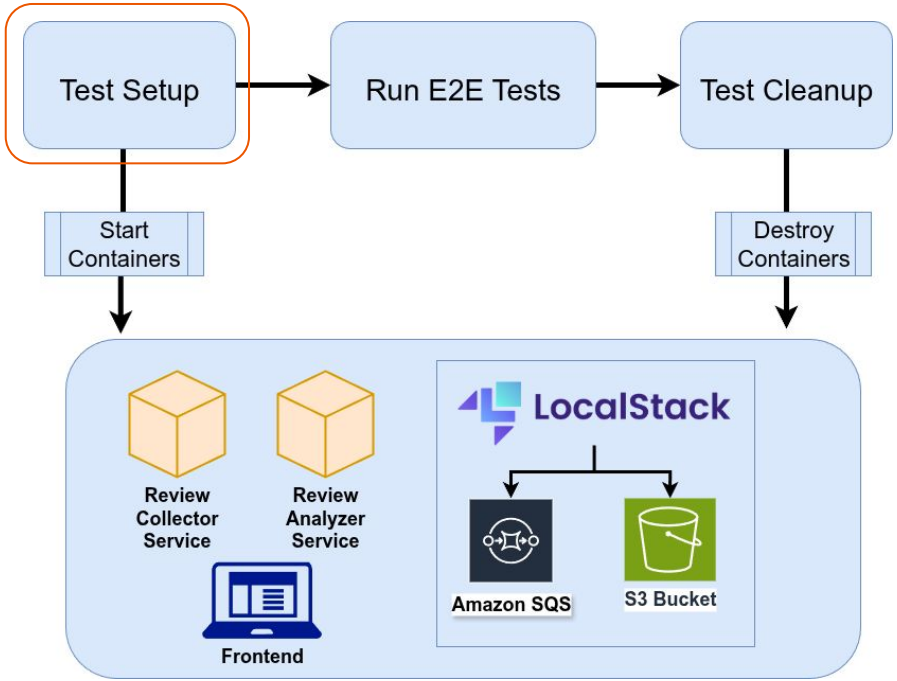


Test Flow - Review Analysis Application



Creating the Test Setup

The Heart of Our Test



```yaml
services:
  localstack:
    image: localstack/localstack:4.0.3
    ports:
      - "4566:4566"
    networks:
      - ls

  review-collector:
    image: teixeirafernando/review-collector:latest
    environment:
      AWS_ENDPOINT: http://localstack:4566
    depends_on:
      - localstack
    ports:
      - "8080:8080"
    networks:
      - ls

  review-analyzer:
    image: teixeirafernando/review-analyzer:latest
    environment:
      AWS_ENDPOINT: http://localstack:4566
    depends_on:
      - localstack
    ports:
      - "8081:8081"
    networks:
      - ls
```

```yaml
frontend-review:
    image: teixeirafernando/frontend-review:latest
    depends_on:
      - review-analyzer
      - review-collector
    ports:
      - "3000:80"
    networks:
      - ls

networks:
  ls:
    driver: bridge
```

Docker Compose File for Test Setup

1

```
const path = require('path');
const { DockerComposeEnvironment, Wait } = require('testcontainers');
const composeFile = path.resolve(__dirname, '../docker-compose.yml');
```

Import Testcontainers dependencies

2

```
let testContainersRuntime = await new DockerComposeEnvironment(
  path.dirname(composeFile),
  path.basename(composeFile)
)
  .withWaitStrategy('localstack', Wait.forLogMessage('Ready'))
  .withWaitStrategy(
    'review-collector',
    Wait.forHttp('/actuator/health', 8080).forStatusCode(200)
  )
  .withWaitStrategy(
    'review-analyzer',
    Wait.forHttp('/actuator/health', 8081).forStatusCode(200)
  )
  .up()
```

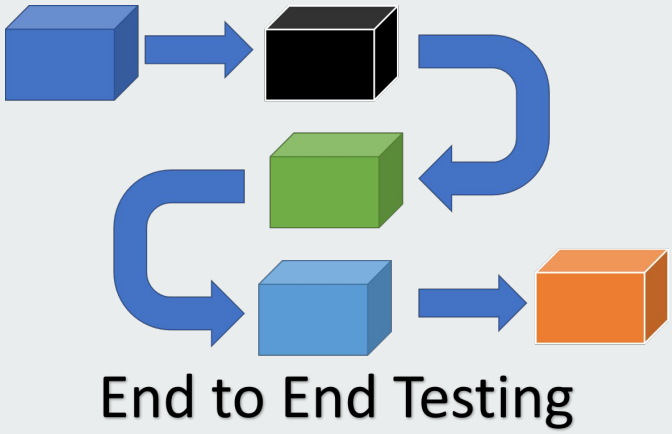
Spin up docker compose using Testcontainers and wait until containers are started

3

```
testContainersRuntime
  .getContainer('localstack')
  .exec(['awslocal', 's3', 'mb', 's3://review-analysis-bucket'])
testContainersRuntime
  .getContainer('localstack')
  .exec(['awslocal','sqs','create-queue','--queue-name','review-analysis-queue',
  ])
```

Execute commands inside Localstack container

Creating the E2E tests



- With the test setup ready, we can properly execute the E2E tests
- Playwright to support our E2E tests, with API requests monitoring

1

```
const { test, expect } = require('@playwright/test');

// E2E test: submit a review through the UI and verify backend response

test('submit review and verify backend analysis', async ({ page, request }) => {
  await page.goto('/');

  // Intercept the POST request to /api/review and capture the response
  let reviewId = null;
  const [createReviewResponse] = await Promise.all([
    page.waitForResponse(resp =>
      resp.url().includes('/api/review') && resp.request().method() === 'POST'),
    // Fill out and submit the form
    (async () => {
      await page.getByTestId('product-name-input').fill('Test Product');
      await page.getByTestId('customer-name-input').fill('Playwright User');
      await page.getByTestId('comment-input').fill('This is a Playwright E2E test review.');
      await page.getByTestId('rating-select').selectOption('5');
      await page.getByTestId('submit-review-btn').click();
    })()
  ]);

  // Wait for success message
  await expect(page.getByTestId('form-message')).toHaveText(/Review submitted
successfully!/i);
```

Using Playwright, we fill out a review form and intercept the POST request that happens to the backend

Validate whether a success message was displayed

2

```
// Verify the response status and content for the review submission
  expect(createReviewResponse.status()).toBe(200);
  const reviewResponse = await createReviewResponse.json();
  expect(reviewResponse).toBeDefined();
  expect(reviewResponse).toHaveProperty('id');

  // Extract the id from the response JSON
  reviewId = reviewResponse.id;
```

Verify the response from the API request triggered by the frontend

3

```
// Get from S3 Bucket the new review using Playwright's expect
with retry mechanism
  await expect(async () => {
    const S3response = await
request.get(`http://localhost:8081/api/messages/${reviewId}`);
    expect(S3response.status()).toBe(200);
    const jsonResponse = await S3response.json();
    expect(jsonResponse).toBeDefined();
    expect(jsonResponse).toHaveProperty('id');
    expect(jsonResponse.id).toBe(reviewId);
    expect(jsonResponse).toHaveProperty('productName');
  }).toPass({
    timeout: 15000, // total time to wait (15s)
    intervals: [3000] // poll every 3s
  });
```

Do one final request to review-analyzer service to check if the review was processed as expected

07

Conclusion

Conclusion

- Good option to save costs, being more effective than mocks
- Easy and powerful integration of LocalStack and Testcontainers
- Emulation ≠ real services
- LocalStack free usage*
- **Personal opinion:** Try to balance real tests and emulated tests. You can have both and implement a better test strategy

Thanks!!

Do you have any questions?

Find me around the web with this QR code

References

- https://www.launchableinc.com/blog/cloud-cost-and-your-software-testing/
- LocalStack Feature Coverage: https://docs.localstack.cloud/references/coverage/
- Feature Reliability: https://docs.localstack.cloud/user-guide/aws/feature-coverage/
- Test Environment Costs: https://www.plutora.com/blog/test-environment-cost
- Code Syntax highlighter: https://carbon.now.sh