



From Testing to Observing

Synthetic Monitoring Unpacked

Jessica Garson

ElasticON NYC Meetup | November 12, 2024

Jessica Garson

@JessicaGarson

@JessicaGarson@macaw.social

Senior Developer Advocate



Agenda

- Overview of testing and synthetic monitoring
- Introduce our demo application
- Synthetic monitoring in Elastic
- How to write and run tests
- Closing & Key Takeaways

What is testing?

Testing in software development allows us to examine individual functions or methods and ensure they behave as expected under various conditions.

A small orange and white kitten sits in a grassy field under a dark blue night sky filled with stars. The kitten has a white chest and paws, and orange and black markings on its back and head. It is looking directly at the camera with large, light-colored eyes.

A simple example

**Jest is a popular testing framework for
JavaScript**



```
function sum(a, b) {  
    return a + b;  
}  
  
module.exports = sum;
```



```
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

What is end-to-end (E2E) testing?

E2E testing checks an entire software application from beginning to end, mimicking fundamental user interactions and data, verifying that a software system performs as intended, covering the complete flow of the application.



```
import { test, expect } from '@playwright/test';

test('has title', async ({ page }) => {
  await page.goto('https://playwright.dev/');

  // Expect a title "to contain" a substring.
  await expect(page).toHaveTitle(/Playwright/);
});

test('get started link', async ({ page }) => {
  await page.goto('https://playwright.dev/');

  // Click the get started link.
  await page.getByRole('link', { name: 'Get started' }).click();

  // Expects page to have a heading with the name of Installation.
  await expect(page.getByRole('heading', { name: 'Installation' })).toBeVisible();
});
```



Synthetic monitoring



What is Synthetic Monitoring?

Synthetic monitoring allows you to track user pathways using global testing infrastructure, emulating the full user path to gauge the influence of web applications.

How does synthetic monitoring relate to testing?

Both testing and synthetic monitoring emulate the user path. This tooling can be used to build tests that can also provide production monitoring in web applications.

<https://demo-store-ivory.vercel.app/>

Demo Store [Home](#)



Coffee Mug
\$15
A coffee mug!
[Buy](#)



Set of Pens
\$10
Pens in assorted colors.
[Buy](#)



Plush Elk
\$16
Elastic Elk!
[Buy](#)



<https://github.com/JessicaGarson/demo-store>

Monitors

[+ Create Monitor](#)[⟳ Refresh](#)[Overview](#) [Management](#) Search by name, URL, host, tag, project or location[Up](#) [Down](#) [Disabled](#)[Type 2](#) [Location 1](#) [Tags 0](#) [Frequency 2](#) [Project 1](#)

Current status

3 Up **0** Down **0** Disabled

Last 6 hours

0 Errors

Last 12 hours

0 Alerts

Showing 3 Monitors

[Sort by Status](#) [Group by None](#)

Lightweight demo store

North America - US East

September 24, 2024 3:05 PM

Duration ⓘ

47 ms

Recorded journey Demo

North America - US East

Duration ⓘ

1 s

Simplified Test for Demo Store

North America - US East

Duration ⓘ

999 ms

"Simplified Test for Demo Store" (North America - US East) is down - Elastic Synthetics ▶



No Reply - Elastic Alerts <no-reply@alerts.elastic.co>

to me ▾

Wed, Sep 25, 12:35 PM (8 days ago)



"Simplified Test for Demo Store" is down from North America - US East. - Elastic Synthetics

Details:

- Monitor name: Simplified Test for Demo Store
- URL: <https://demo-store-ivory.vercel.app/>
- Monitor type: browser
- Checked at: Sep 25, 2024 @ 16:34:48.900
- From: North America - US East
- Error received: error executing step: Timed out 5000ms waiting for expect(locator).toBeVisible()
- Link: <https://getting-started.kb.us-east4.gcp.elastic-cloud.com:9243/app/synthetics/monitor/bf435caa-83eb-45b1-a876-a6c20b8035aa/errors/us-east4-a-1922a090bc4-0?locationId=us-east4-a>

This message was sent by Elastic. [View rule in Kibana](#).

How to get started



```
npm install -g @elastic/synthetics  
npx @elastic/synthetics init projects-test
```



changed 144 packages **in** 3s

26 packages are looking **for** funding

run `npm fund` **for** details

> Initializing Synthetics project in 'live-demo'

✓ Enter Elastic Kibana URL or Cloud ID • **your_cloud_id=**

✓ What is your API key • ****

✓ Select the locations where you want to run monitors • us_east

✓ Set default schedule **in** minutes **for** all monitors • 10

✓ Choose project id to logically group monitors • live-demo

✓ Choose the target Kibana space • default



> Setting up project using NPM...

Wrote to /Users/jessgarson/rehearsal-synth-mon/live-demo/package.json:

```
{  
  "name": "live-demo",  
  "version": "1.0.0",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "description": ""  
}
```

> Installing @elastic/synthetics library ...





```
> Writing live-demo/synthetics.config.ts.
> Writing live-demo/journeys/example.journey.ts.
> Writing live-demo/lightweight/heartbeat.yml.
> Writing live-demo/journeys/advanced-example-helpers.ts.
> Writing live-demo/journeys/advanced-example.journey.ts.
> Writing live-demo/.github/workflows/run-synthetics.yml.
> Writing live-demo/README.md.
> Writing live-demo/package.json.
```

All set, you can run below commands inside: /Users/jessgarson/rehearsal-synth-mon/live-demo:

Run synthetic tests: `npm run test`

Push monitors to Kibana: `SYNTHETICS_API_KEY=<value> npm run push`

Configure API Key via ``SYNTHETICS_API_KEY`` env variable or `--auth` CLI flag.

Visit <https://www.elastic.co/guide/en/observability/current/synthetic-run-tests.html> to learn more.

Key commands to keep in mind

Run synthetic tests in the journeys folder :



```
npm run test
```

Push monitors to Kibana:



```
SYNTHETICS_API_KEY=<value> npm run push
```

Heartbeat



```
heartbeat.monitors:  
- type: http  
  name: Lightweight demo store  
  id: demo-store-lightweight  
  enabled: true  
  urls: "https://demo-store-ivory.vercel.app"  
  schedule: '@every 3m'  
  timeout: 16s  
  alert.status.enabled: true
```

Elastic Synthetics  **Playwright**



```
import { journey, step, monitor, expect } from '@elastic/synthetics';

journey('Simplified Test for Demo Store', ({ page, params }) => {
  monitor.use({
    id: 'demo-store-check',
    schedule: 10,
  });

  step('Launch application', async () => {
    await page.goto(params.url);
  });

  step('Check for loading state', async () => {
    const loadingText = page.getByText('Loading...');
    if (await loadingText.isVisible({ timeout: 5000 })) {
      await loadingText.waitFor({ state: 'hidden', timeout: 60000 });
    }
  });
});
```





```
step('Validate page content', async () => {
  const header = page.getByRole('heading', { level: 1 });
  await expect(header).toHaveText('Demo Store');

  const firstProductCard = page.getByTestId('product-card').first();
  await expect(firstProductCard).toBeVisible();

  const buyButton = firstProductCard.getByTestId('buy-button');
  await expect(buyButton).toBeVisible();

  await page.screenshot({ path: 'page-content.png' });
});
```



```
step('Interact with the Buy button', async () => {
  page.once('dialog', async dialog => {
    const message = dialog.message();
    console.log('Alert message:', message);
    expect(message).toBe('Added to cart!');
    await dialog.accept();
  });
}

const firstProductCard = page.getByTestId('product-card').first();
const buyButton = firstProductCard.getByTestId('buy-button');
await buyButton.click();

await page.screenshot({ path: 'after-click.png' });
});
```





```
step('Monitor page performance', async () => {
  const performance = await page.evaluate(() => {
    return JSON.stringify(window.performance);
  });
  console.log(performance);
})`;
});
```

Let's run our tests!



Simplified Test for Demo Store

[Overview](#) [History](#) [Errors](#)

Location
North America - US East **Status** Succeeded **Last run**
9 seconds ago

[Run test manually](#)[Edit monitor](#)[Refresh](#)**Monitor details**Enabled (all locations) 

URL

<https://demo-store-ivory.vercel.app/>

Last run

Oct 1, 2024 @ 2:26:14 PM

Last modified

Sep 25, 2024 @ 1:19:31 PM

Project ID

projects-test

Monitor ID

demo-store-check-projects-test-default

Monitor type

 Journey

Frequency

Every 10 minutes

Locations

 North America - US East

Tags

--

Summary To date**100.0%****1.0 s**

Median duration

**1**

Errors



Simplified Test for Demo Store-Test results

North America - US East		COMPLETED	Took 1 s	View test result details	
5 steps completed					
Steps	#	Screenshot	Step name	Result	Duration
>	1		Launch application	Complete	688 ms
>	2		Check for loading state	Complete	102 ms
>	3		Validate page content	Complete	184 ms
>	4		Interact with the Buy button	Complete	238 ms
>	5		Monitor page performance	Complete	47 ms



Another option



Monitors

[+ Create Monitor](#)[⟳ Refresh](#)

[Overview](#) [Management](#)

 Search by name, URL, host, tag, project or location

Up

Down

Disabled

Type

2

Location

1

Tags

0

Frequency

2

Project

1

Current status

3 Up **0** Down **0** Disabled

Last 6 hours

0 Errors

Last 12 hours

0 Alerts

Showing 3 Monitors

Sort by [Status](#)

Group by [None](#)



3

Add a script

Use Elastic Synthetics Recorder to generate a script and then upload it. Alternatively, you can write your own [Playwright](#) script and paste it in the script editor.

[Launch Synthetics Recorder](#)[Download Synthetics Recorder](#)[Upload script](#)[Script editor](#)

Select or drag and drop a .js file

Parameters

1



Use JSON to define parameters that can be referenced in your script with `params.value`

[Advanced options](#)[Cancel](#)[Run test](#)[Create monitor](#)

[▶ Start](#)● Not recording[!\[\]\(0f56135fbafc50fedfcfff94393fe0f3_img.jpg\) Test](#)[!\[\]\(240c12821e227464ff6b7614924c0018_img.jpg\) Export](#)

No steps recorded yet

Click on [Start recording](#) to get started with your script.

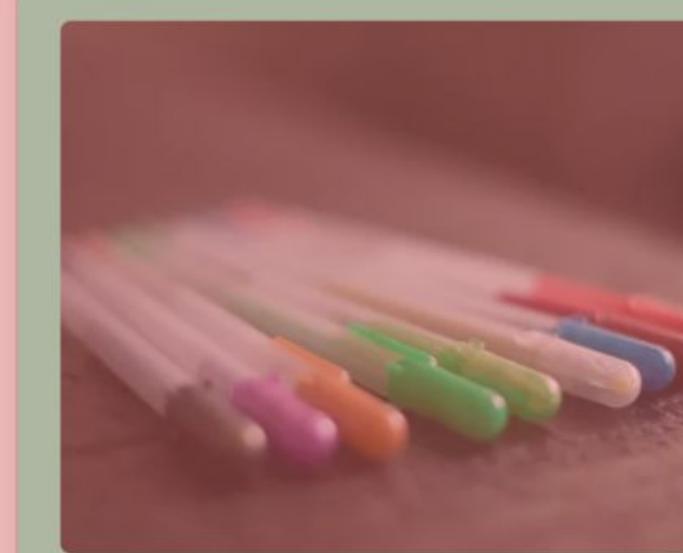


Demo Store

[Home](#)**Coffee Mug**

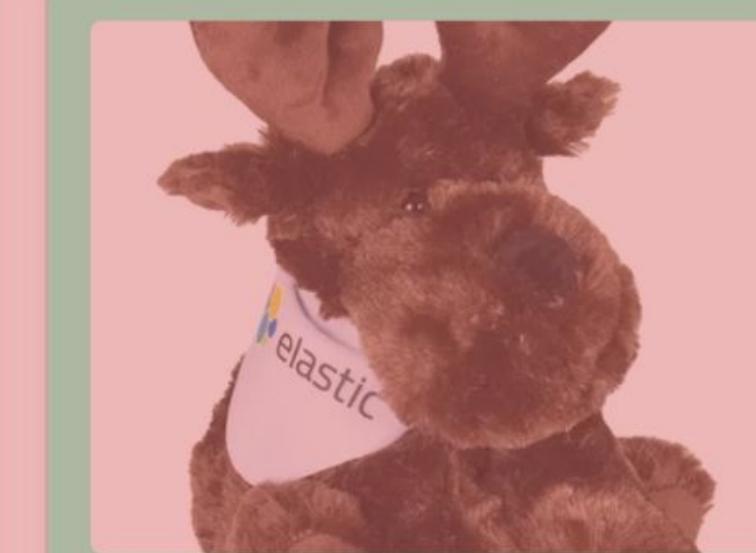
\$15

A coffee mug!

[Buy](#)**Set of Pens**

\$10

Pens in assorted colors.

[Buy](#)**Plush Elk**

\$16

Elastic Elk!

[Buy](#)

```
getByText('Coffee Mug$15A coffee mug!BuySet of Pens$10Pens in assorted  
colors.BuyPlush Elk$')
```

[Pause](#)[Stop](#)

● Recording

[Test](#)[Export](#)▼ Step 1 **navigate** https://demo-store-ivory.vercel.app/products**click** div >> internal:has-text=/^Coffee Mug\\$15A coffee mug!Buy\$/ >> [data-test="buy-button"]**click** div >> internal:has-text=/^Plush Elk\\$16Elastic Elk!Buy\$/ >> [data-test="buy-button"]**click** div >> internal:has-text=/^Set of Pens\\$10Pens in assorted colors!.Buy\$/ >> [data-test="buy-button"]**click** internal:role=link[name="Home"]

Tradeoffs with this option



Next steps



Resources

Get started

<https://www.elastic.co/guide/en/observability/current/synthetics-get-started-project.html>

Two sides of the same coin: Unit testing and monitoring with Synthetic Monitoring

<https://www.elastic.co/observability-labs/blog/testing-monitoring-synthetic-monitoring>

My coworker Carly's talk from DevOps.js

<https://portal.gitnation.org/contents/synthetic-monitoring-and-e2e-testing-2-sides-of-the-same-coin-1979>



<https://github.com/JessicaGarson/Synthetic-Monitoring-Unpacked>



Thank you!