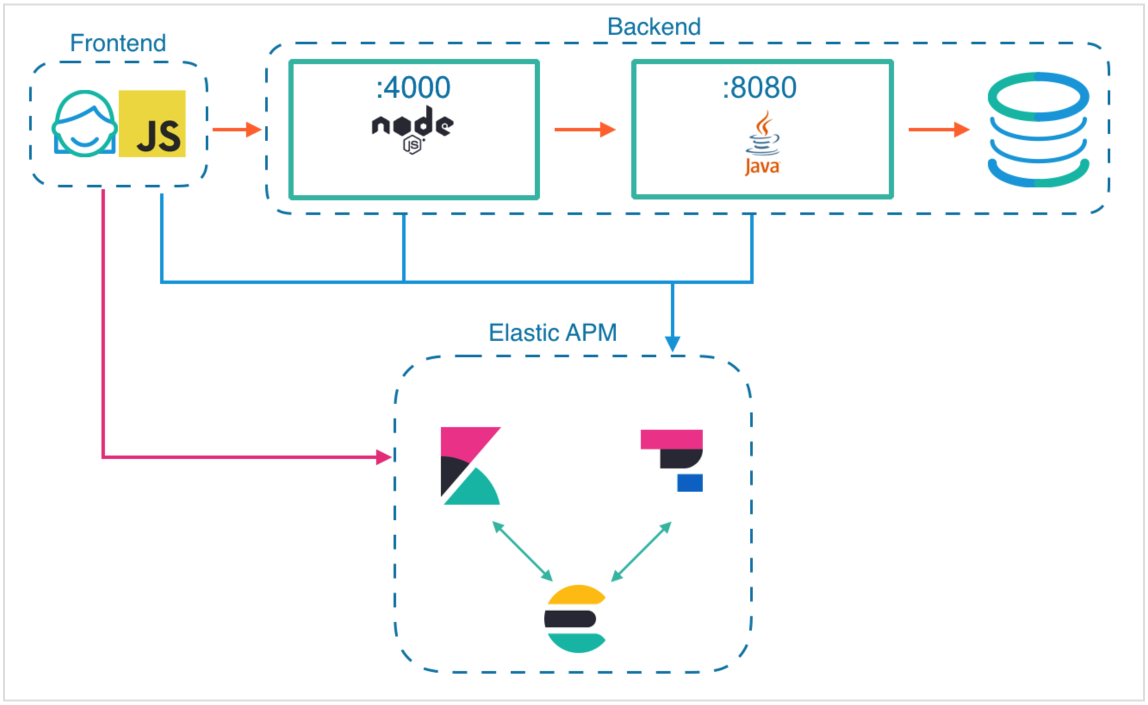
Lab 4: APM

Objective:

In this lab, you will learn how to use APM to instrument applications for collecting detailed performance information as well as errors. You will also explore the Kibana APM app and see how you can monitor application performance.

In these labs, you will learn how to create an observable system.

1. Before you get started, review the architecture you are implementing through the lab steps as described in the picture below.

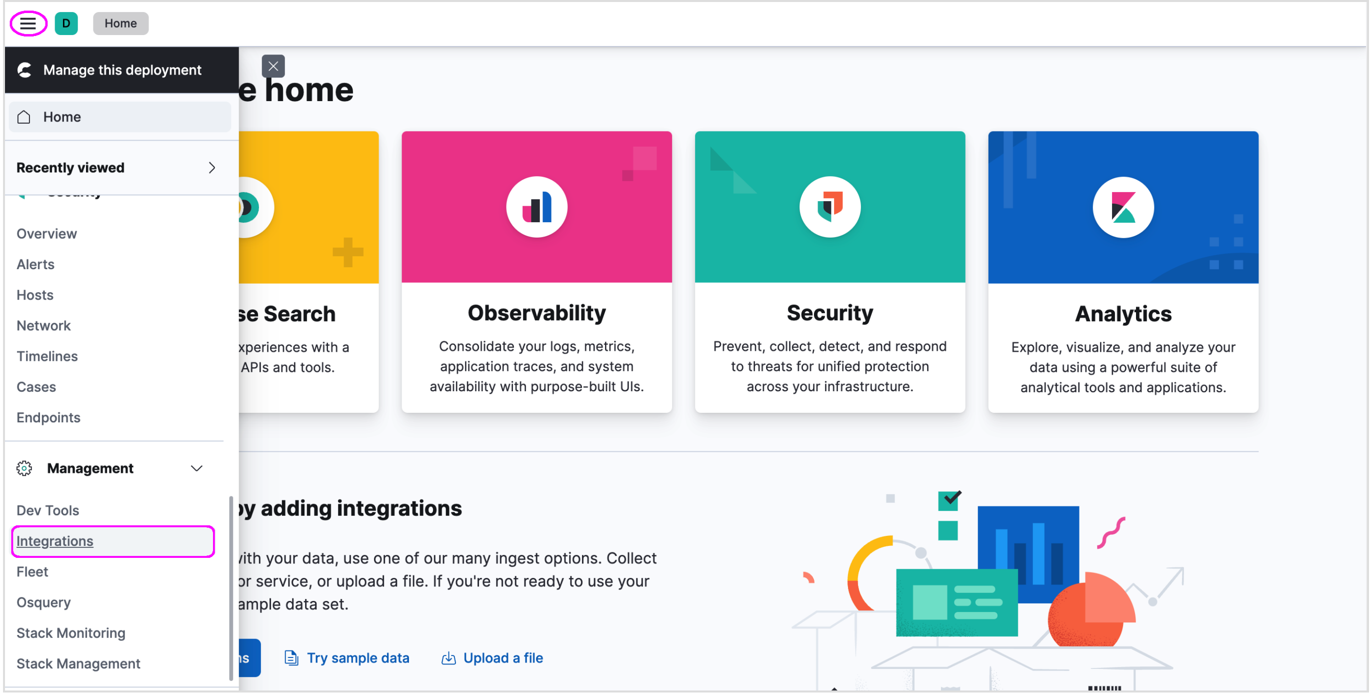


   You already have your Elasticsearch deployment running with APM integration set to send data there and Kibana to retrieve data from there. In the next steps you will instrument Petclinic to then explore distributed tracing through the APM app in Kibana. Petclinic is a demo application composed of several different applications and services.

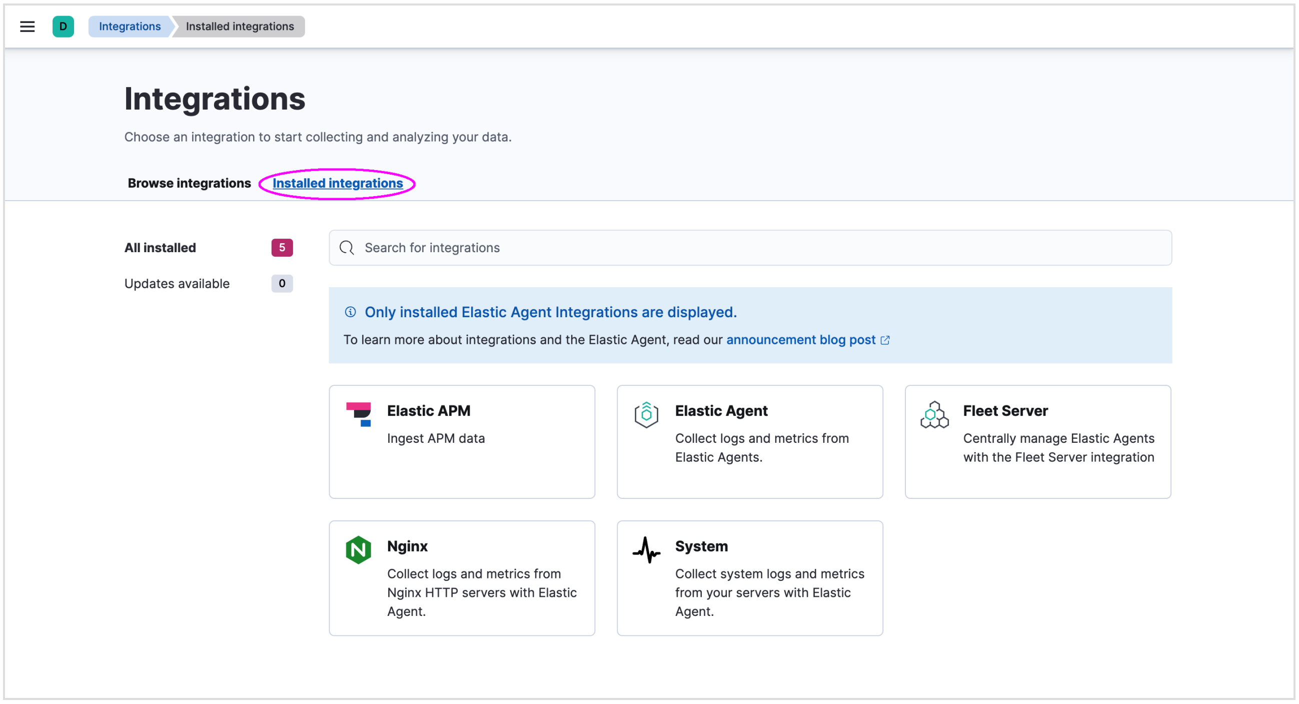
   - The frontend is a **React** application that runs on the client's browser.
   - The frontend is served by a **Node.js** backend server that listens on port `4000`.
   - The backend server proxies all frontend requests to the application core, so it can handle the requests.
   - The application core is a REST API implementation that connects to the databases. It is implemented with the **Spring** framework and listens on port `8080`.

     To monitor this architecture, you will use the **Java** agent for the application core, the **Node.js** agent for the backend server, and the **RUM** agent for the frontend.

2. Since you are using Elastic Cloud, the APM integration comes installed by default. An Elastic Agent has been enrolled to the **Elastic Cloud agent policy**. It is ready for collecting APM data and sending it to Elastic Cloud.
   First, you need to get the APM Server URL and secret token from your APM integration, so you can configure APM agents to send data to your deployment. To do that, select **Integrations** from the main menu in **Kibana**.


D
Home
Manage this deployment
Home
Recently viewed
Overview
Alerts
Hosts
Network
Timelines
Cases
Endpoints
Management
Dev Tools
Integrations
Fleet
Osquery
Stack Monitoring
Stack Management
e home
se Search
xperiences with a
APIs and tools.
Observability
Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.
Security
Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.
Analytics
Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.
y adding integrations
ith your data, use one of our many ingest options. Collect
or service, or upload a file. If you're not ready to use your
ample data set.
Try sample data
Upload a file

3. Then, select the **Installed integrations** tab and access the **APM** integration.



Integrations

Choose an integration to start collecting and analyzing your data.

Only installed Elastic Agent Integrations are displayed.

To learn more about integrations and the Elastic Agent, read our announcement blog post

Elastic APM Ingest APM data

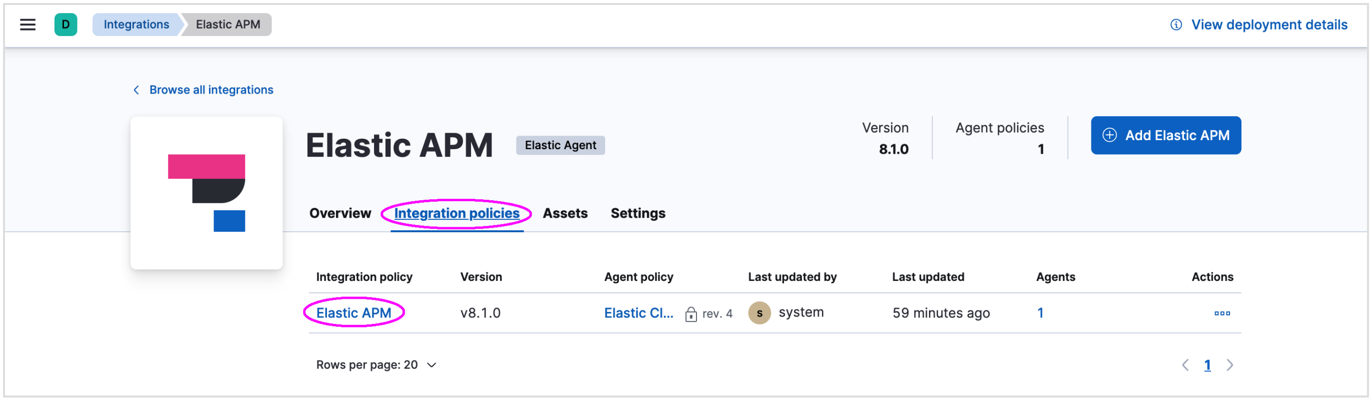
Elastic Agent Collect logs and metrics from Elastic Agents.

Fleet Server Centrally manage Elastic Agents with the Fleet Server integration

Nginx Collect logs and metrics from Nginx HTTP servers with Elastic Agent.

System Collect system logs and metrics from your servers with Elastic Agent.

4. Now, access the **Elastic Cloud agent policy** for **Elastic APM** to find the APM Server URL and secret token.



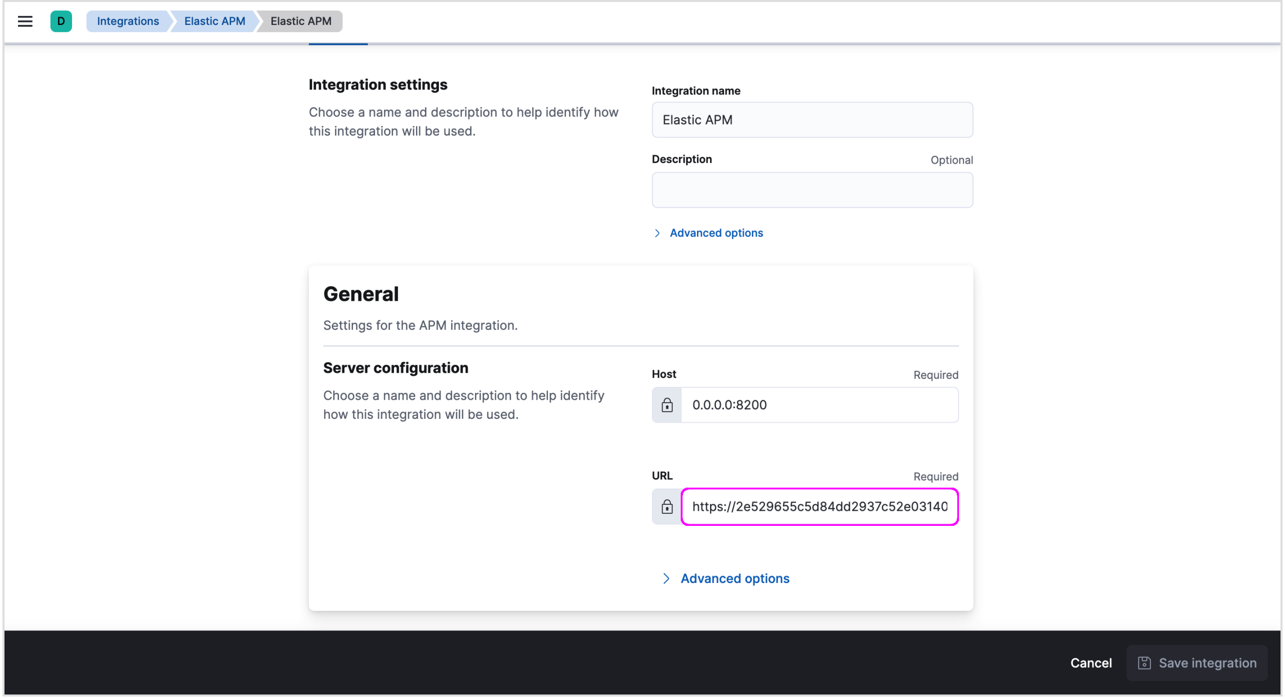
Elastic APM

Version 8.1.0 | Agent policies 1

Integration policy	Version	Agent policy	Last updated by	Last updated	Agents	Actions
Elastic APM	v8.1.0	Elastic Cl... rev. 4	system	59 minutes ago	1	...

Rows per page: 20

5. Copy the APM Server URL, as you will need it in the next lab steps.



Integration settings

Choose a name and description to help identify how this integration will be used.

Integration name: Elastic APM

Description (Optional):

General

Settings for the APM integration.

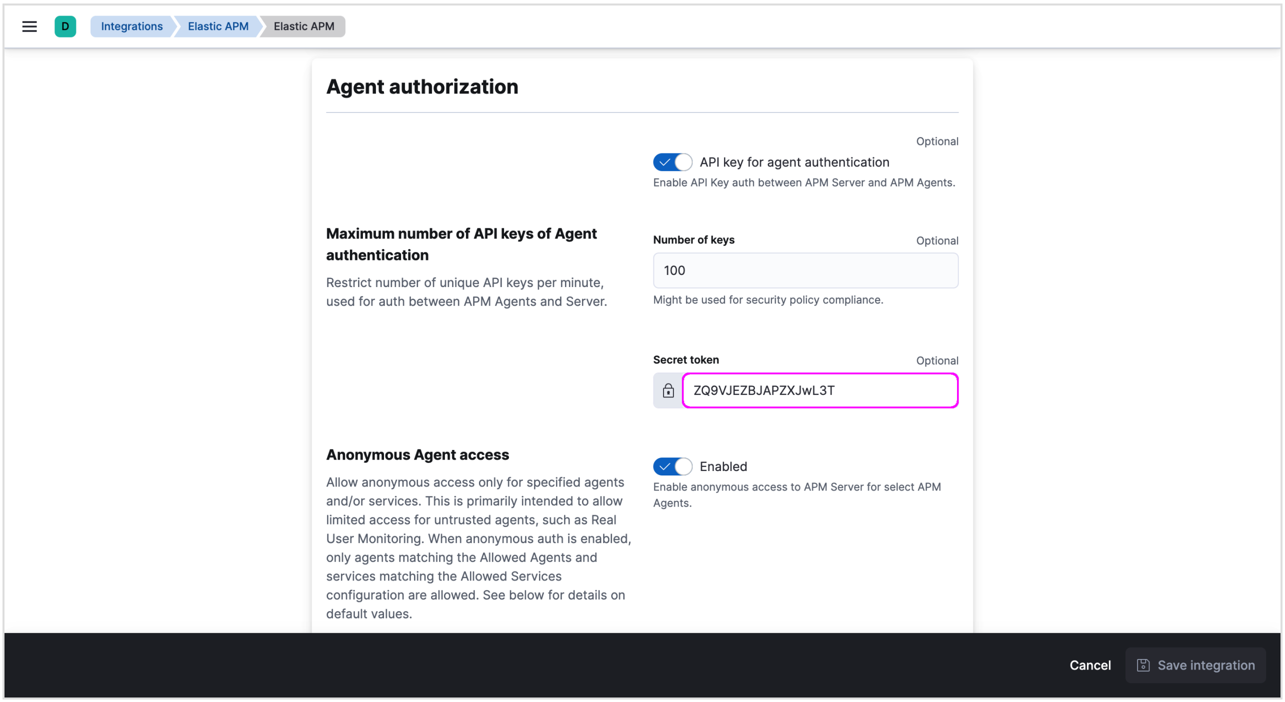
Server configuration

Choose a name and description to help identify how this integration will be used.

Host (Required): 0.0.0.0:8200

URL (Required): https://2e529655c5d84dd2937c52e03140

6. And copy the secret token, as you will need it in the next lab steps.



Agent authorization

Optional

API key for agent authentication

Enable API Key auth between APM Server and APM Agents.

Maximum number of API keys of Agent authentication

Restrict number of unique API keys per minute, used for auth between APM Agents and Server.

Number of keys (Optional): 100

Might be used for security policy compliance.

Secret token (Optional): ZQ9VJEZBJAPZXJwL3T

Anonymous Agent access

Allow anonymous access only for specified agents and/or services. This is primarily intended to allow limited access for untrusted agents, such as Real User Monitoring. When anonymous auth is enabled, only agents matching the Allowed Agents and services matching the Allowed Services configuration are allowed. See below for details on default values.

Enabled

Enable anonymous access to APM Server for select APM Agents.

7. Make sure you don't edit any configuration, as you will be using the default settings. Cancel editing to get back to the APM integration.
8. Next, you will start the application core. To do that, open a new terminal window.
9. The next step would be downloading the latest release of the agent jar file. However, the lab environment already has the version you need to run the labs.

```
ls petclinic/elastic-apm-agent-1.21.0.jar
```

10. Use the following command to start the Petclinic application core. Before running it replace `APM_SERVER_URL` and `APM_SERVER_TOKEN` with the APM Server URL and secret token you copied in the previous steps.

```
java -javaagent:/home/elastic/petclinic/elastic-apm-agent-1.21.0.jar \
  -Delastic.apm.service_name=petclinic-spring \
  -Delastic.apm.server_urls=APM_SERVER_URL \
  -Delastic.apm.secret_token=APM_SERVER_TOKEN \
  -Delastic.apm.environment=production \
  -Delastic.apm.application_packages=org.springframework.samples.petclinic \
  -jar /home/elastic/petclinic/spring-petclinic-1.5.16.jar
```

Note that you don't need to declare a dependency to the agent in your application. You only need to download the agent and add the `-javaagent` flag with the path to the jar agent when starting your application. You also need to specify:

- the `elastic.apm.service_name` setting as the name of the service that will appear in the APM app.
- `elastic.apm.server_urls` is the setting that defines where the APM Server is running.
- `elastic.apm.secret_token` provides the credentials to access the APM Server.
- the `elastic.apm.environment` and `elastic.apm.application_packages` settings are optional. The former helps navigating through APM data from a specific environment when you have more than one. The latter helps the APM app to collapse the stack frames of library code and highlight the stack frames originated from your application.

Solution

After replacing the APM Server URL and secret token, the command line should look like as follows:

```
java -javaagent:/home/elastic/petclinic/elastic-apm-agent-1.21.0.jar \
  -Delastic.apm.service_name=petclinic-spring \
  -Delastic.apm.server_urls=https://2e529655c5d84dd2937c52e031404f8a.apm.europe-west1.gcp.cloud.es.io:443 \
  -Delastic.apm.secret_token=ZQ9VJEZBJAPZXJwL3T \
  -Delastic.apm.environment=production \
  -Delastic.apm.application_packages=org.springframework.samples.petclinic \
  -jar /home/elastic/petclinic/spring-petclinic-1.5.16.jar
```

11. Next, you will start the application backend that serves the frontend. To do that, open a new terminal window.
12. Access the `petclinic/frontend` directory:

```
cd petclinic/frontend
```

13. Both the backend server and the frontend already have the Node.js and RUM agent installed. This means that you only need to make sure their configurations are correct before starting the backend server. So, check how the Node.js agent is required and started at the top of the `bin/www` main file.

```
head bin/www
```

You should see the following configurations:

```
#!/usr/bin/env node
const settings = require('../config')
var apm = require('elastic-apm-node').start({
  serviceName: settings.apm_service_name,
  serviceVersion: settings.apm_service_version,
  serverUrl: settings.apm_server,
  ....
```

Note how the `config.js` file is used to set the `serviceName`, `serviceVersion`, and `serverUrl` settings for the Node.js agent.

14. You also need to specify the `secretToken`, so the Node.js agent can access the APM Server that is running on your cloud deployment. Edit `bin/www` and add the following line to the configuration object.

```
  secretToken: settings.apm_server_token,
```

After editing the top of the `bin/www` file, it should look like this:

```
#!/usr/bin/env node
const settings = require('../config')
var apm = require('elastic-apm-node').start({
  serviceName: settings.apm_service_name,
  serviceVersion: settings.apm_service_version,
  serverUrl: settings.apm_server,
  secretToken: settings.apm_server_token,
  ...
```

15. Check the configurations set by the `config.js` file.

```
cat config.js
```

You should see the following object declaration:

```
var config = {
  apm_server: process.env.ELASTIC_APM_SERVER_URL || 'http://localhost:8200',
  apm_server_js: process.env.ELASTIC_APM_SERVER_JS_URL || 'http://localhost:8200',
  apm_service_name: process.env.ELASTIC_APM_SERVICE_NAME || 'petclinic-node',
  apm_client_service_name: process.env.ELASTIC_APM_CLIENT_SERVICE_NAME || 'petclinic-react',
  apm_service_version: process.env.ELASTIC_APM_SERVICE_VERSION || '1.0.0',
  api_server: process.env.API_SERVER || 'http://localhost:8080',
  api_prefix: process.env.API_PREFIX || '/petclinic/api',
  address_server: process.env.ADDRESS_SERVER || 'http://localhost:5000',
  distributedTracingOrigins: process.env.DISTRIBUTED_TRACINGS_ORIGINS || 'http://petclinic-
client:3000,http://petclinic-
server:8000,http://localhost:4000,http://localhost:8080,http://localhost:8081'
}
```

Note how `apm_service_name`, `apm_service_version`, and `apm_server` definitions relate to the Node.js agent configuration in the previous step. You will edit this configuration file in the next steps to make sure the Node.js and RUM agents can reach your APM Server.

16. Start by editing `config.js` to add `apm_server_token` to the object object declaration as follows. Note that you need to replace `APM_SERVER_TOKEN` with the secret token you copied before.

```
  apm_server_token: process.env.ELASTIC_APM_SECRET_TOKEN || 'APM_SERVER_TOKEN',
```

Solution

After editing `config.js` the object declaration should look like this:

```
var config = {
  apm_server_token: process.env.ELASTIC_APM_SECRET_TOKEN || 'ZQ9VJEZBJAPZXJwL3T',
  apm_server: process.env.ELASTIC_APM_SERVER_URL || 'http://localhost:8200',
  apm_server_js: process.env.ELASTIC_APM_SERVER_JS_URL || 'http://localhost:8200',
  apm_service_name: process.env.ELASTIC_APM_SERVICE_NAME || 'petclinic-node',
  apm_client_service_name: process.env.ELASTIC_APM_CLIENT_SERVICE_NAME || 'petclinic-react',
  apm_service_version: process.env.ELASTIC_APM_SERVICE_VERSION || '1.0.0',
  api_server: process.env.API_SERVER || 'http://localhost:8080',
  api_prefix: process.env.API_PREFIX || '/petclinic/api',
  address_server: process.env.ADDRESS_SERVER || 'http://localhost:5000',
  distributedTracingOrigins: process.env.DISTRIBUTED_TRACINGS_ORIGINS || 'http://petclinic-
client:3000,http://petclinic-server:8000,http://localhost:4000,http://localhost:8080,http://localhost:8081'
}
```

17. Then, edit `config.js` and change `apm_server` to use your APM Server URL with port `443` instead of `http://localhost:8200`. After doing this you configured the settings that make the Node.js agent be able to communicate with the APM integration running on your deployment.

Solution

After editing `config.js` the object declaration should look like this:

```
var config = {
  apm_server_token: process.env.ELASTIC_APM_SECRET_TOKEN || 'ZQ9VJEZBJAPZXJwL3T',
  apm_server: process.env.ELASTIC_APM_SERVER_URL || 'https://2e529655c5d84dd2937c52e031404f8a.apm.europe-
west1.gcp.cloud.es.io:443',
  apm_server_js: process.env.ELASTIC_APM_SERVER_JS_URL || 'http://localhost:8200',
  apm_service_name: process.env.ELASTIC_APM_SERVICE_NAME || 'petclinic-node',
  apm_client_service_name: process.env.ELASTIC_APM_CLIENT_SERVICE_NAME || 'petclinic-react',
  apm_service_version: process.env.ELASTIC_APM_SERVICE_VERSION || '1.0.0',
  api_server: process.env.API_SERVER || 'http://localhost:8080',
  api_prefix: process.env.API_PREFIX || '/petclinic/api',
  address_server: process.env.ADDRESS_SERVER || 'http://localhost:5000',
  distributedTracingOrigins: process.env.DISTRIBUTED_TRACINGS_ORIGINS || 'http://petclinic-
client:3000,http://petclinic-server:8000,http://localhost:4000,http://localhost:8080,http://localhost:8081'
}
```

18. Next, edit `config.js` and change `apm_server_js` to use your APM Server URL with port `443` instead of `http://localhost:8200`. After doing this you setup the RUM agent from `public/index.js` to collect APM data in the client's browser and send them to your APM integration.
IMPORTANT: Note that there are two configurations: `apm_server` and `apm_server_js`. The former is used by the backend server, while the latter is used by the frontend. There are two different configurations because the Node.js and RUM agents might access the APM integration through different endpoints. In particular, the frontend will be running on the client's browser and needs to know how to reach the APM integration through a public address. Also note that the secret token is not applicable for the RUM agent, because there is no way to prevent it from being publicly exposed.

Solution

After editing `config.js` the object declaration should look like this:

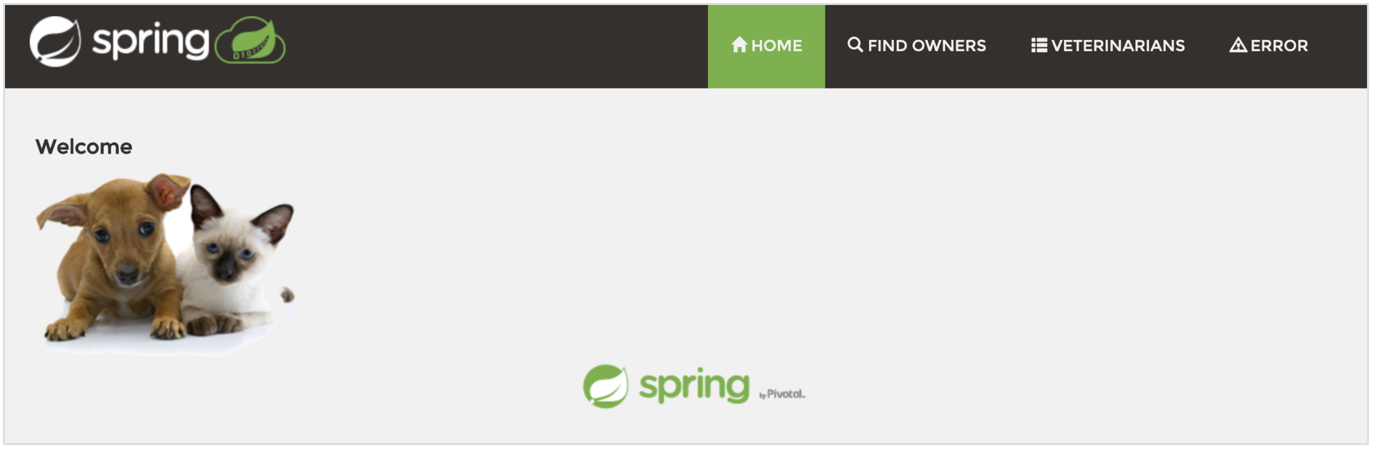
```
var config = {
  apm_server_token: process.env.ELASTIC_APM_SECRET_TOKEN || 'ZQ9VJEZBJAPZXJwL3T',
  apm_server: process.env.ELASTIC_APM_SERVER_URL || 'https://2e529655c5d84dd2937c52e031404f8a.apm.europe-west1.gcp.cloud.es.io:443',
  apm_server_js: process.env.ELASTIC_APM_SERVER_JS_URL || 'https://2e529655c5d84dd2937c52e031404f8a.apm.europe-west1.gcp.cloud.es.io:443',
  apm_service_name: process.env.ELASTIC_APM_SERVICE_NAME || 'petclinic-node',
  apm_client_service_name: process.env.ELASTIC_APM_CLIENT_SERVICE_NAME || 'petclinic-react',
  apm_service_version: process.env.ELASTIC_APM_SERVICE_VERSION || '1.0.0',
  api_server: process.env.API_SERVER || 'http://localhost:8080',
  api_prefix: process.env.API_PREFIX || '/petclinic/api',
  address_server: process.env.ADDRESS_SERVER || 'http://localhost:5000',
  distributedTracingOrigins: process.env.DISTRIBUTED_TRACINGS_ORIGINS || 'http://petclinic-client:3000,http://petclinic-server:8000,http://localhost:4000,http://localhost:8080,http://localhost:8081'
}
```

19. Start the backend for serving the frontend:

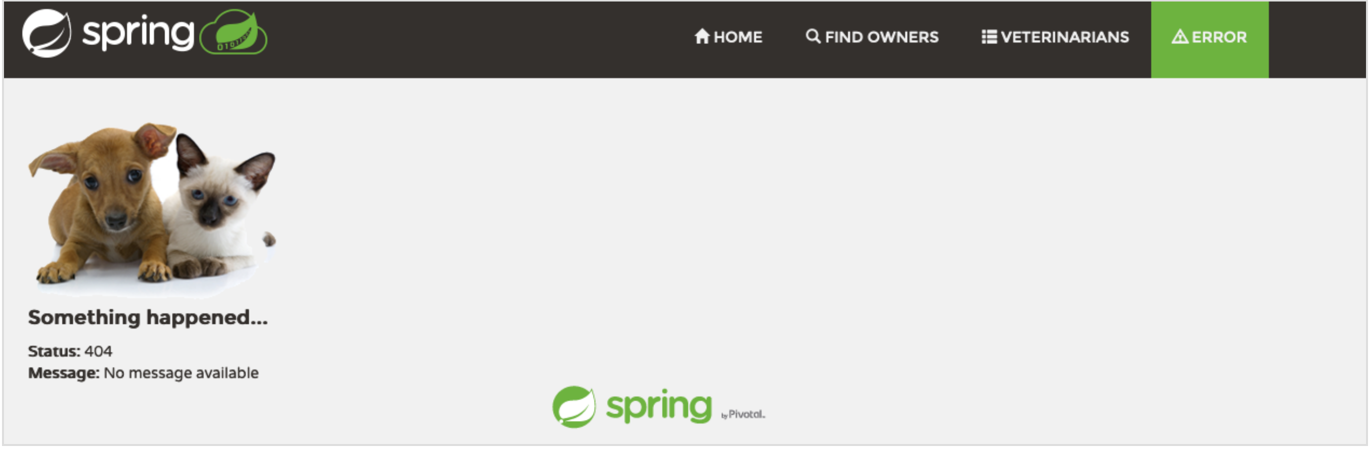
```
npm start
```

Note that the RUM and Node.js agents are already declared as dependencies to the frontend and backend applications, respectively. The settings `apm_client_service_name` and `apm_service_name` present in the `config.js` file define the name of the service as it will appear in the APM app.

20. Now that all the three micro services are running, access the Petclinic web page and you should see the following home page:



21. Click on **FIND OWNERS** and **VETERINARIANS** to generate performance data to be sent to the APM Server.
22. Click on **ERROR** to generate errors to be sent to the APM Server. Make sure you get the `404` status and the `No message available` error message as follows:



Something happened...

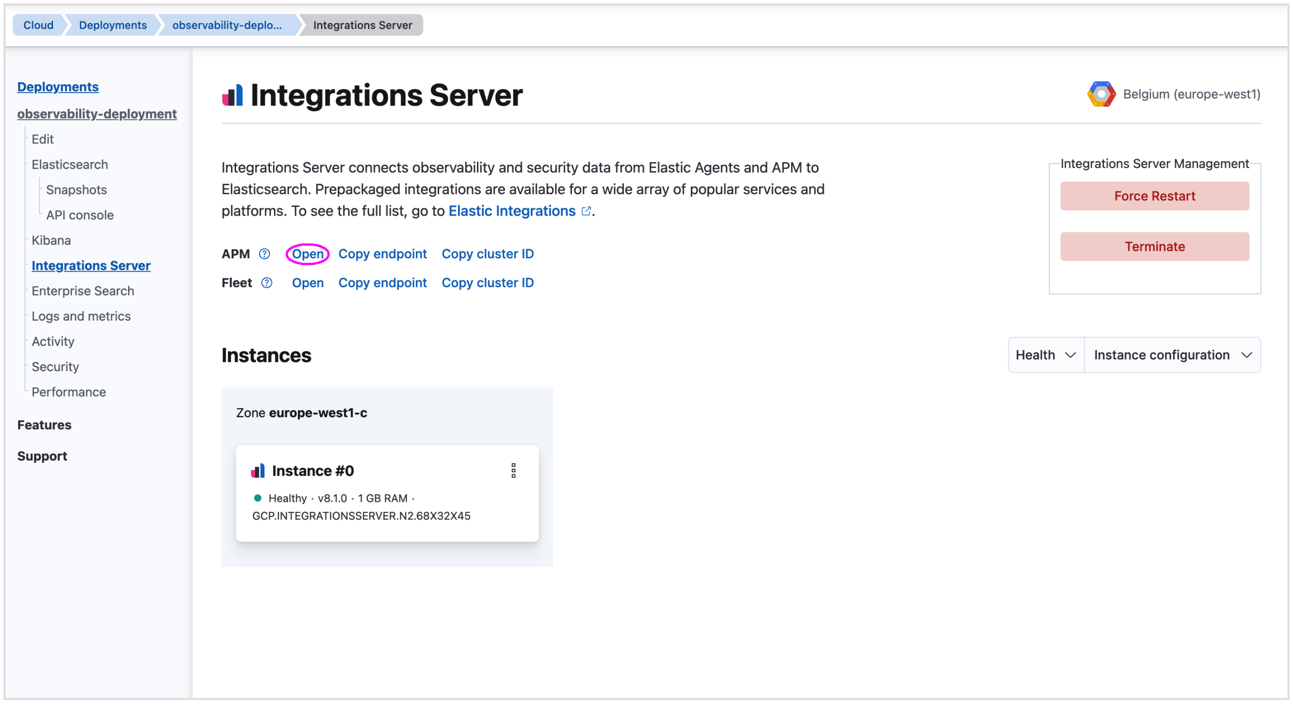
Status: 404
Message: No message available

If you don't get this error, click on **HOME** and click back on **ERROR** until you get this error.

23. Launch the APM app to start exploring the collected data.

Solution

To launch the APM app, in the Elastic Cloud console, access **Integrations Server** through the navigation menu of your `observability-deployment`. Next, click on the **Open** link right next to the APM label. You can also launch APM through your deployment main page.



Integrations Server

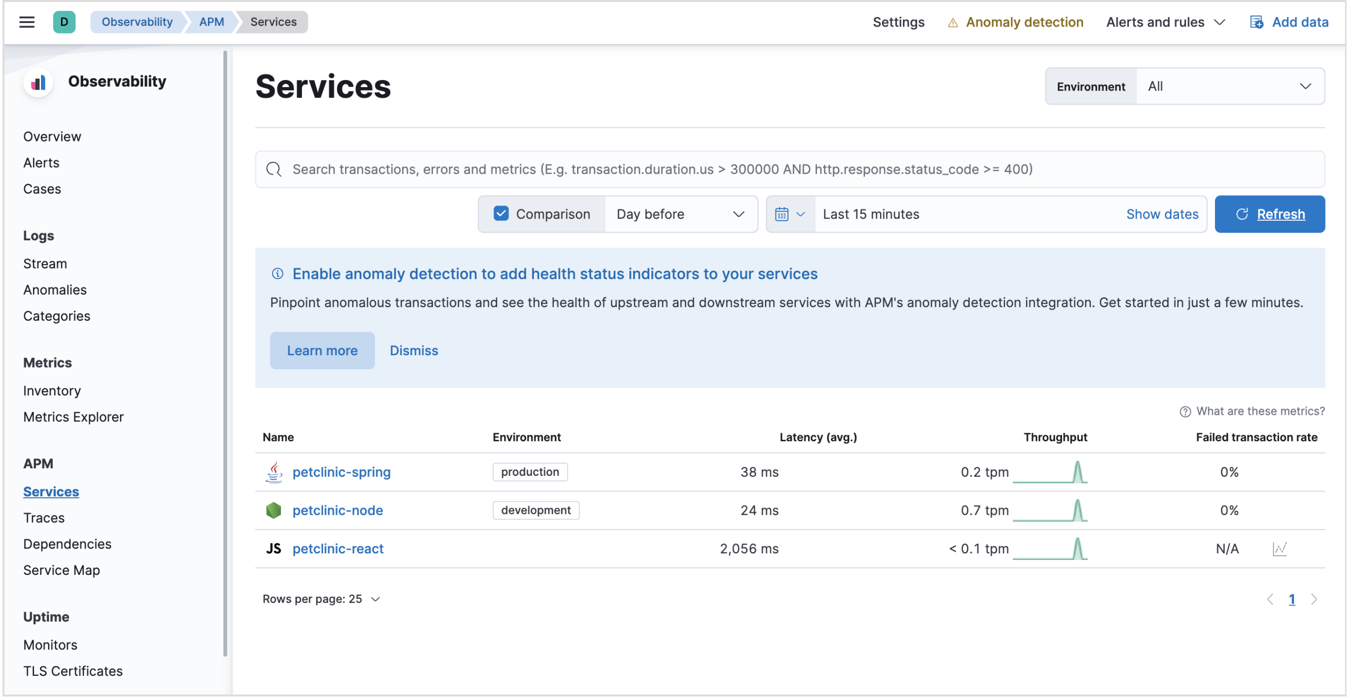
Integrations Server connects observability and security data from Elastic Agents and APM to Elasticsearch. Prepackaged integrations are available for a wide array of popular services and platforms. To see the full list, go to Elastic Integrations.

Instances

Zone europe-west1-c

Instance #0
Healthy · v8.1.0 · 1 GB RAM · GCP.INTEGRATIONSSERVER.N2.68X32X45

24. After launching the APM app you will reach the **Services** overview.



Services

Environment: All

Search transactions, errors and metrics (E.g. transaction.duration.us > 300000 AND http.response.status_code >= 400)

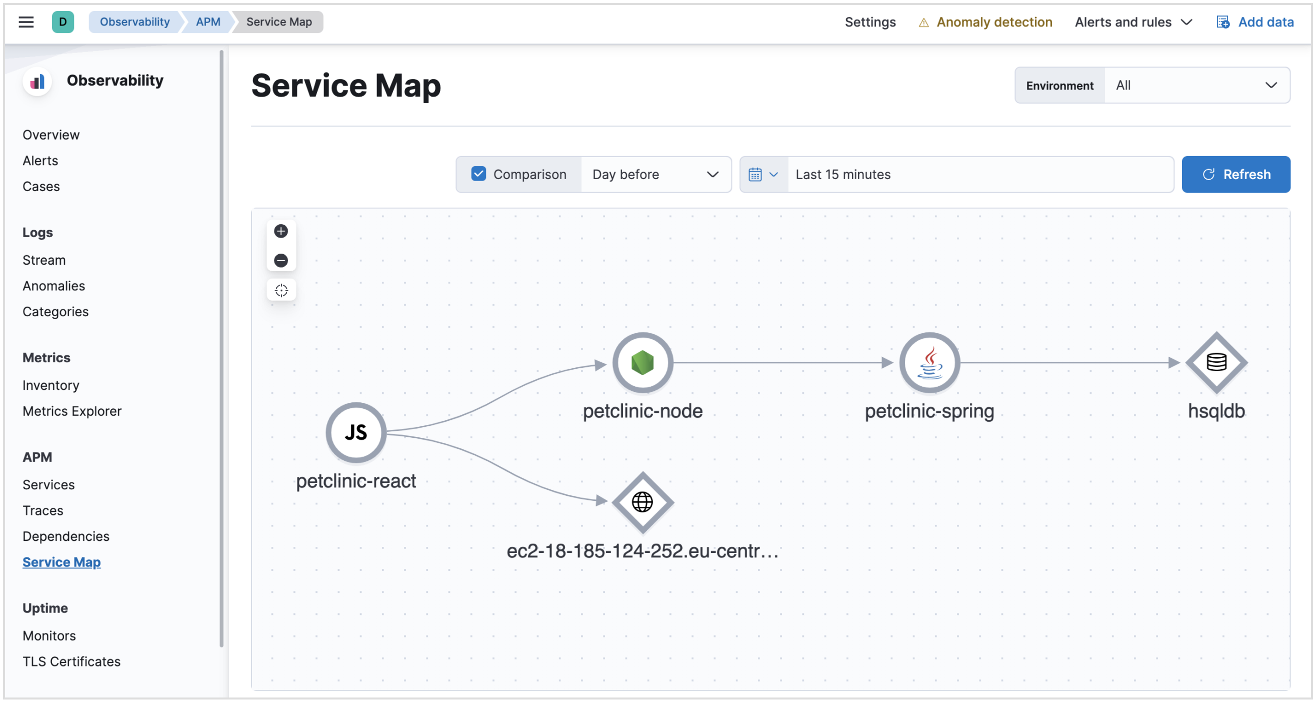
Enable anomaly detection to add health status indicators to your services

Pinpoint anomalous transactions and see the health of upstream and downstream services with APM's anomaly detection integration. Get started in just a few minutes.

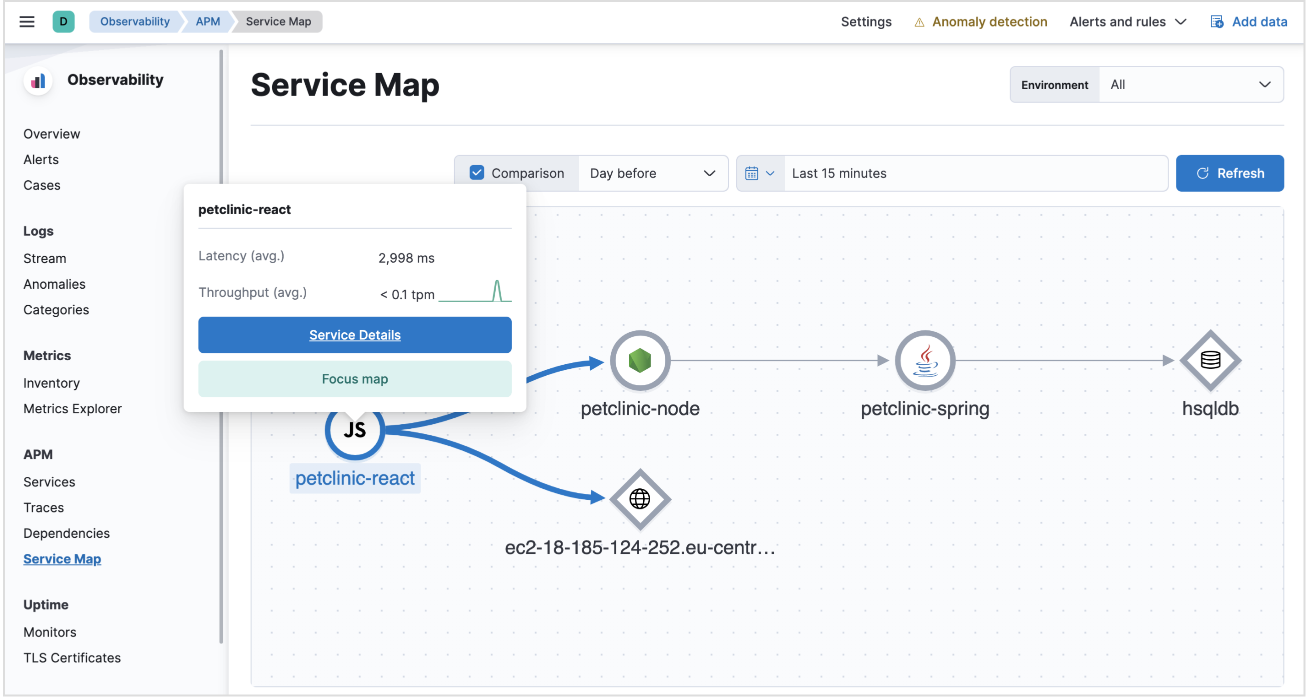
Name	Environment	Latency (avg.)	Throughput	Failed transaction rate
petclinic-spring	production	38 ms	0.2 tpm	0%
petclinic-node	development	24 ms	0.7 tpm	0%
JS petclinic-react		2,056 ms	< 0.1 tpm	N/A

Rows per page: 25

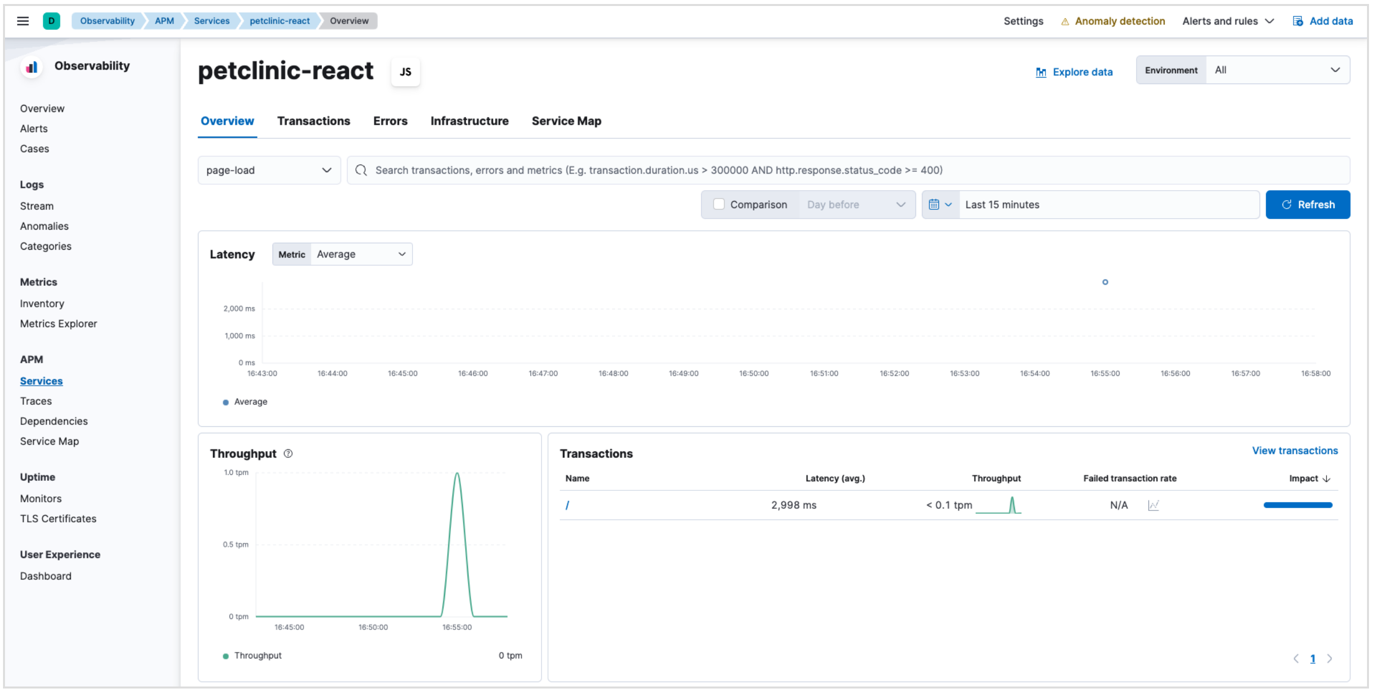
25. Now, click on **Service Map** to check the Petclinic architecture. Note how it describes that **petclinic-react** connects to **petclinic-node** that connects to **petclinic-spring**, which is connected to the database.



Click on **petclinic-react** to get high-level metrics like average latency and average throughput.

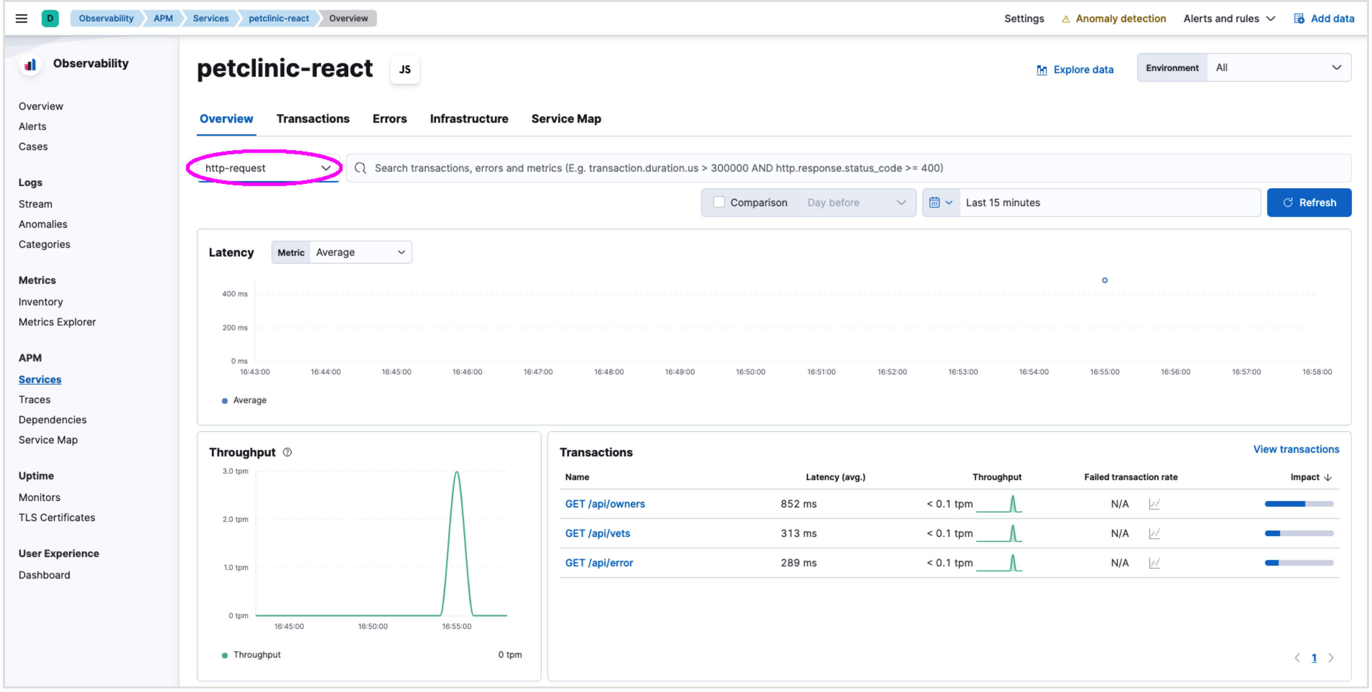


Click on **Service Details** to explore the overall health of the frontend by checking metrics about transactions, errors, and infrastructure.



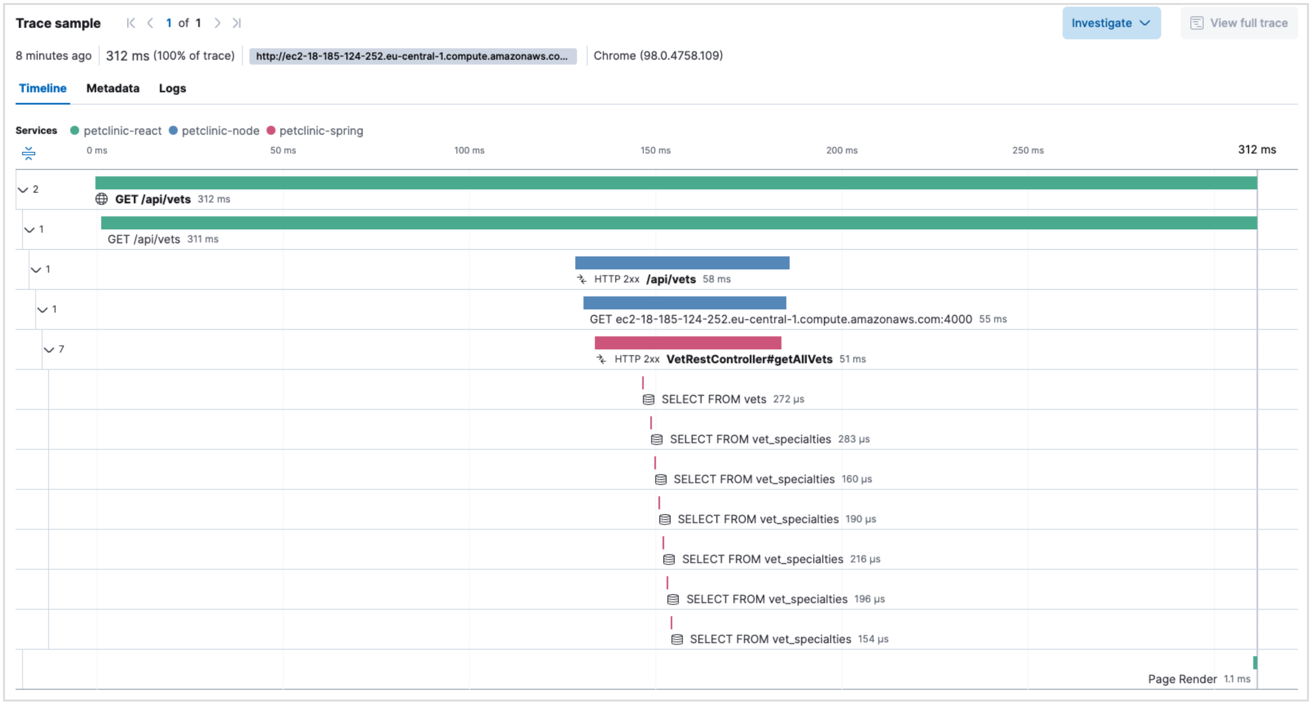
Name	Latency (avg.)	Throughput	Failed transaction rate	Impact
/	2,998 ms	< 0.1 tpm	N/A	

Next, select the `http-request` transaction type to explore the frontend requests.



Name	Latency (avg.)	Throughput	Failed transaction rate	Impact
GET /api/owners	852 ms	< 0.1 tpm	N/A	
GET /api/vets	313 ms	< 0.1 tpm	N/A	
GET /api/error	289 ms	< 0.1 tpm	N/A	

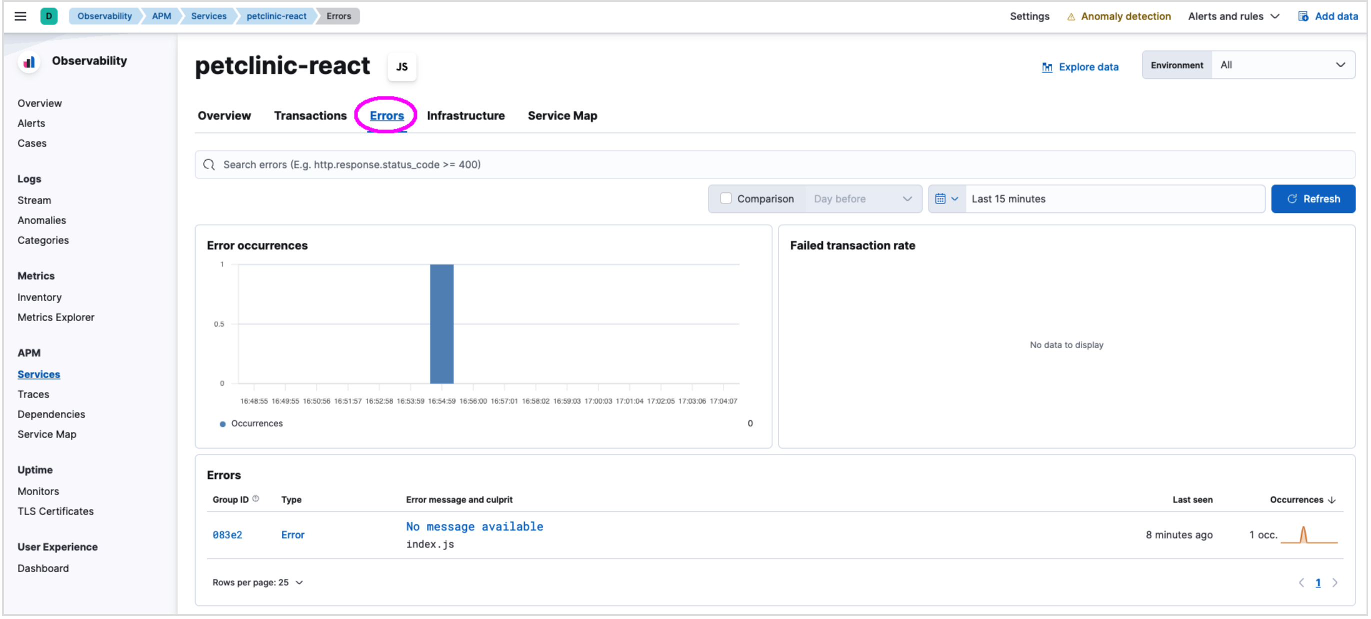
Then, click on `GET /api/vets` to explore distributed tracing and see how the applications and services interact with each other.



8 minutes ago | 312 ms (100% of trace) | http://ec2-18-185-124-252.eu-central-1.compute.amazonaws.co... | Chrome (98.0.4758.109)

```
2 GET /api/vets 312 ms
1 GET /api/vets 311 ms
1 HTTP 2xx /api/vets 58 ms
1 GET ec2-18-185-124-252.eu-central-1.compute.amazonaws.com:4000 55 ms
7 HTTP 2xx VetRestController#getAllVets 51 ms
SELECT FROM vets 272 μs
SELECT FROM vet_specialties 283 μs
SELECT FROM vet_specialties 160 μs
SELECT FROM vet_specialties 190 μs
SELECT FROM vet_specialties 216 μs
SELECT FROM vet_specialties 196 μs
SELECT FROM vet_specialties 154 μs
Page Render 1.1 ms
```

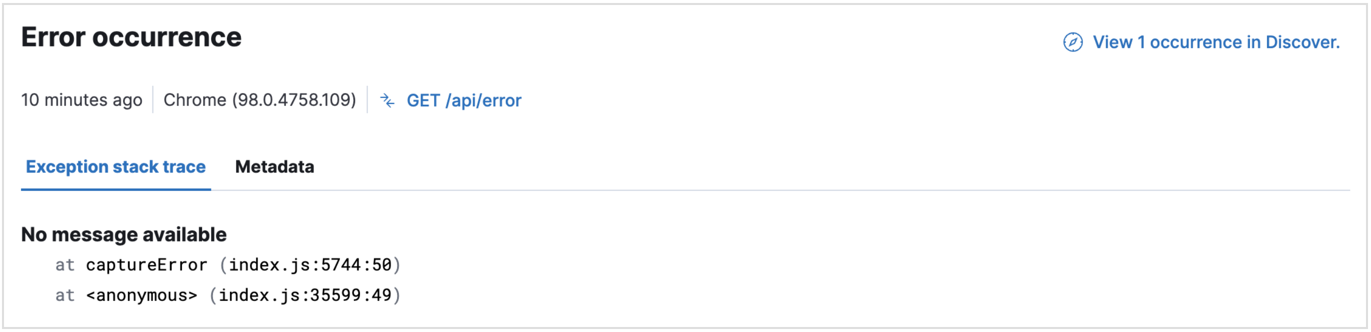
Next, click on the **Errors** overview to explore the application errors.



Errors

Group ID	Type	Error message and culprit	Last seen	Occurrences
083e2	Error	`No message available` `index.js`	8 minutes ago	1 occ.

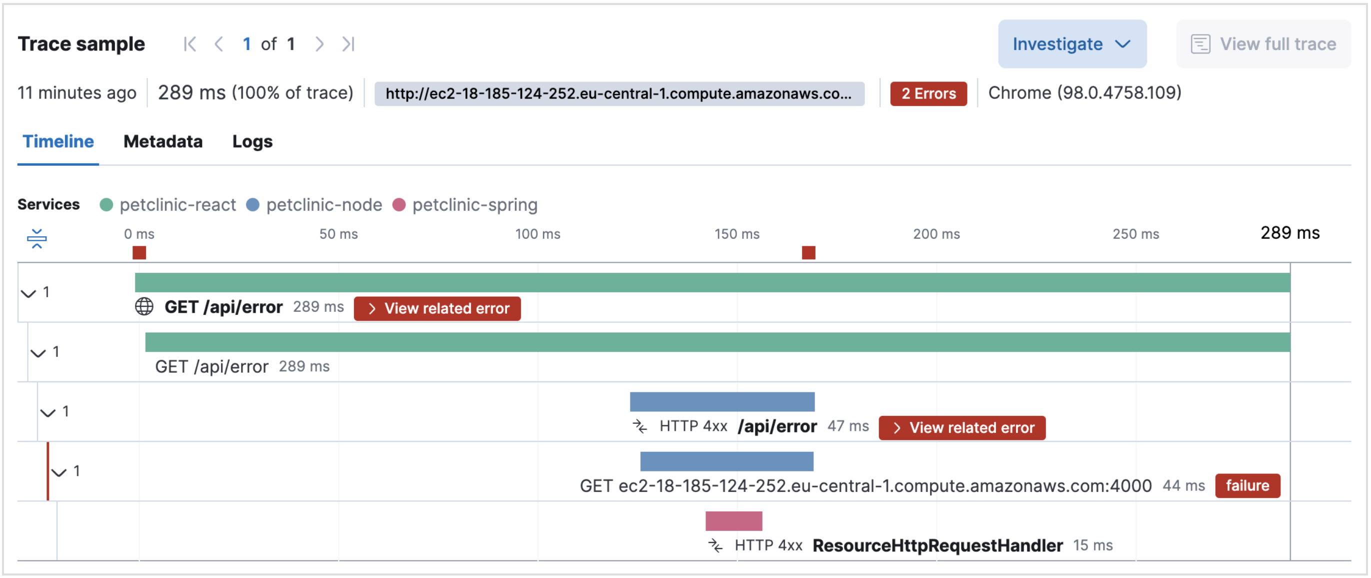
Then, click on the `No message available` error to see its stack trace that points to where the error originated.



No message available

```
    at captureError (index.js:5744:50)
    at <anonymous> (index.js:35599:49)
```

Finally, click on the transaction `GET /api/error` to see the related errors and how they were propagated among the services.



Summary:

In this lab, you learned how to setup APM agents for collecting trace information and errors to index them into Elasticsearch. You also explored the Kibana APM app and saw how you can monitor application performance.