

JAVAMS04 Working with Stackdriver Trace

2 hoursFree

Rate Lab

Overview

In this series of labs, you take a demo microservices Java application built with the Spring framework and modify it to use an external database server. You adopt some of the best practices for tracing, configuration management, and integration with other services using integration patterns.

In a microservices architecture, you need distributed tracing to make complicated service calls more visible. For example, when service A calls B, and B calls C, which service has a problem? In Spring Cloud GCP, you can easily add distributed tracing by using Spring Cloud Sleuth. This typically requires you to run and operate your own Zipkin backend.

In this lab, you implement distributed tracing by using Spring Cloud GCP, Spring Cloud Sleuth, and Stackdriver Trace. Spring Cloud GCP provides a starter that can interoperate with Spring Cloud Sleuth, but it forwards the trace request to Stackdriver Trace instead.

Stackdriver Trace is a distributed tracing system that collects latency data from your applications and displays it in the Google Cloud Platform (GCP) console. You can track how requests propagate through your application and receive detailed, near-real-time performance insights. Stackdriver Trace automatically analyzes all of your application's traces to generate in-depth latency reports to surface performance degradations. And it can capture traces from all of your VMs, containers, or App Engine projects.

Language-specific SDKs for Stackdriver Trace can analyze projects running on VMs (even those not managed by GCP). The Trace SDK is currently available for Java, Node.js, Ruby, and Go. And Stackdriver Trace API can be used to submit and retrieve trace data from any source. A Zipkin collector is also available, which enables Zipkin tracers to submit data to Stackdriver Trace. Projects running on App Engine are automatically captured.

Objectives

In this lab, you learn how to perform the following tasks:

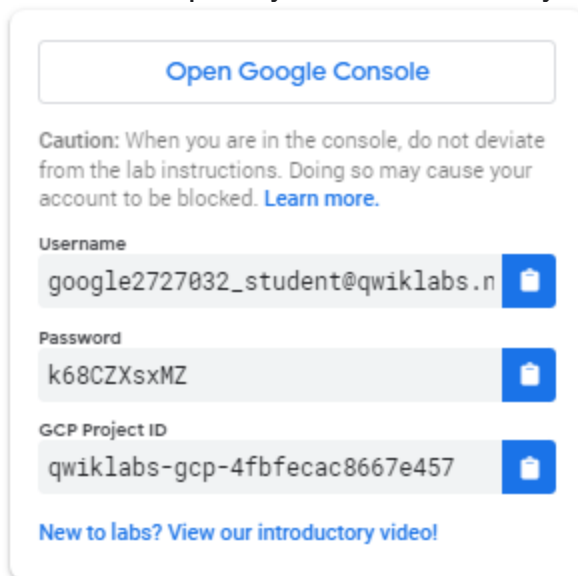
- Enable Stackdriver Trace API
- Use Spring to add Stackdriver Trace to your application
- Configure customized trace settings in an application
- Inspect the trace output

Task 0. Lab Preparation

Access Qwiklabs

How to start your lab and sign in to the Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



[Open Google Console](#)

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)

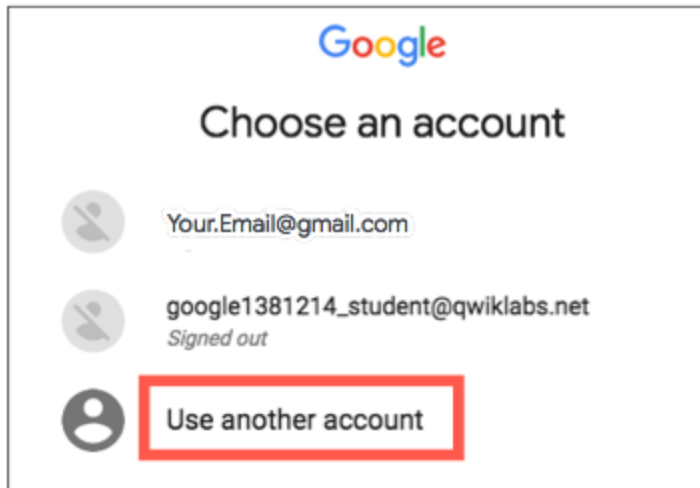
Username
google2727032_student@qwiklabs.n

Password
k68CZXsxMZ

GCP Project ID
qwiklabs-gcp-4fbfecac8667e457

[New to labs? View our introductory video!](#)

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Choose an account** page.
Tip: Open the tabs in separate windows, side-by-side.
3. On the Choose an account page, click **Use Another Account**.



4. The Sign in page opens. Paste the username that you copied from the Connection Details panel. Then copy and paste the password.

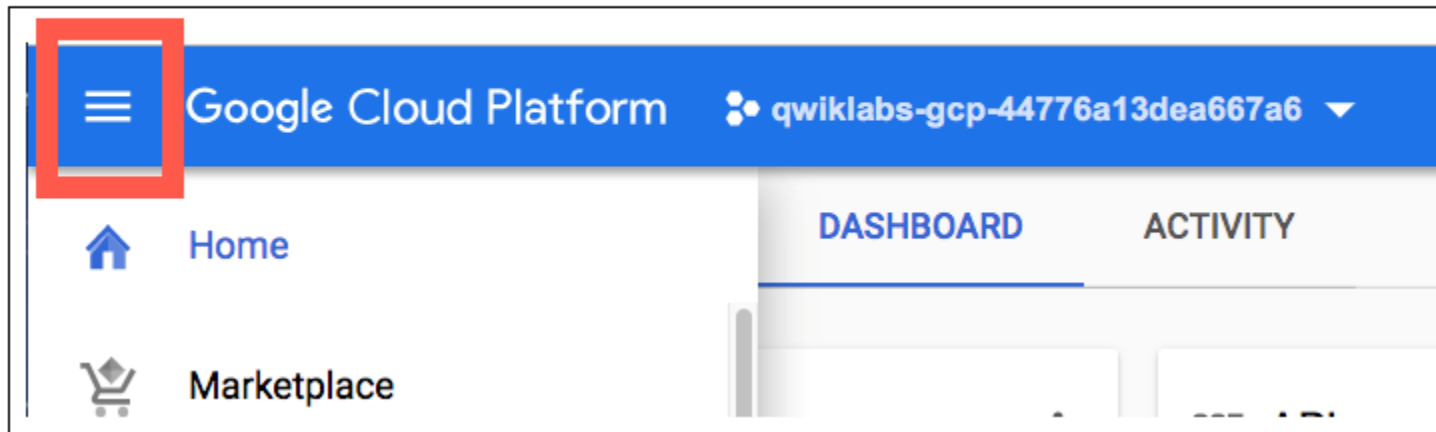
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own GCP account, do not use it for this lab (avoids incurring charges).

5. Click through the subsequent pages:
 - Accept the terms and conditions.
 - Do not add recovery options or two-factor authentication (because this is a temporary account).
 - Do not sign up for free trials.

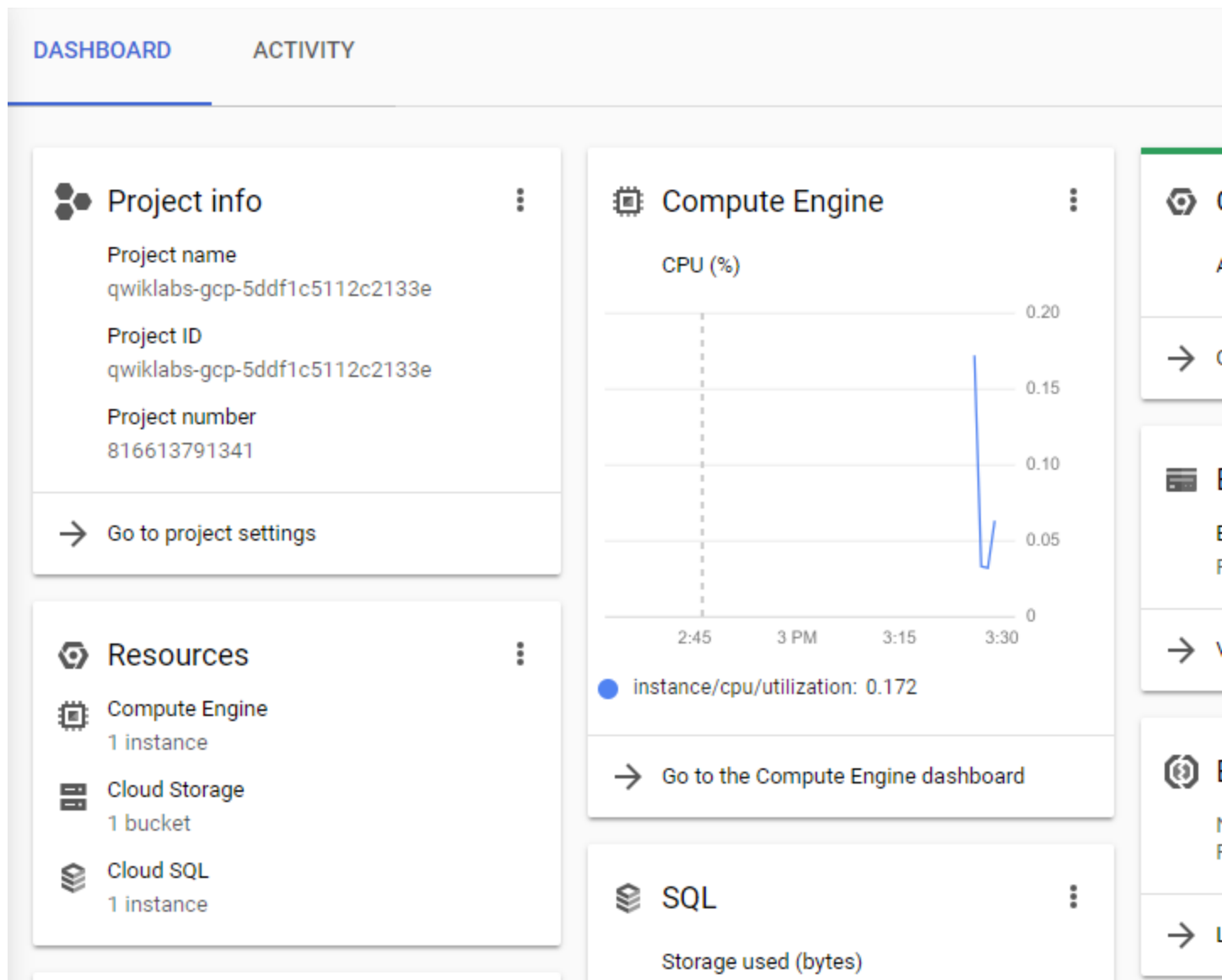
After a few moments, the GCP console opens in this tab.

Note: You can view the menu with a list of GCP Products and Services by clicking the **Navigation menu** at the top-left, next to "Google Cloud

Platform”.



After you complete the initial sign-in steps, the project dashboard appears.



Fetch the application source files

The lab setup includes automated deployment of the services that you configured yourself in previous labs. When the setup is complete, copies of the demo application (configured so that they are ready for this lab session) are put into a Cloud Storage bucket named using the project ID for this lab.

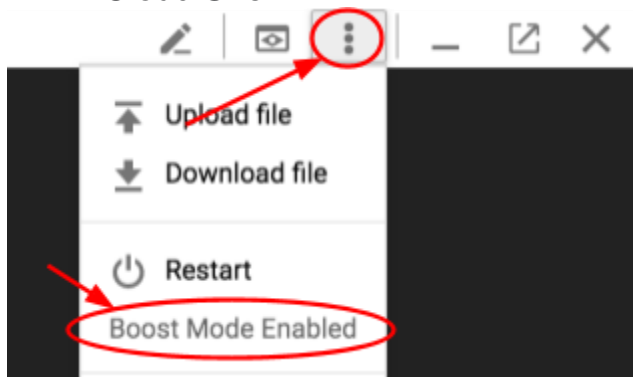
Before you proceed with the tasks for this lab, you must first copy the demo application into Cloud Shell so you can continue to work on it.

1. In the upper-right corner of the screen, click **Activate Cloud**



Shell () to open Cloud Shell.

2. Click **Start Cloud Shell**.
3. If **Boost Mode Enabled** is not available (bold), enable boost mode for Cloud Shell.



4. In the Cloud Shell command line, enter the following command to create an environment variable that contains the project ID for this lab:

```
export PROJECT_ID=$(gcloud config list --format 'value(core.project)')
```

5. Verify that the demo application files were created.

```
gsutil ls gs://$PROJECT_ID
```

Repeat the last step if the command reports an error or if it does not list the two folders for the `guestbook-frontend` application and the `guestbook-service` backend application.

Note

A Cloud Storage bucket that is named using the project ID for this lab is automatically created for you by the lab setup. The source code for your applications is copied into this bucket when the Cloud SQL server is ready. You might have to wait a few minutes for this action to complete.

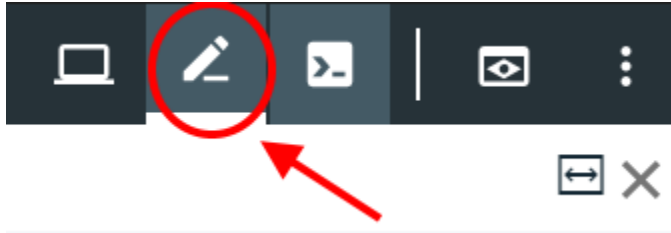
6. Copy the application folders to Cloud Shell.

```
gsutil -m cp -r gs://$PROJECT_ID/* ~/
```

7. Make the Maven wrapper scripts executable.

```
chmod +x ~/guestbook-frontend/mvnw
chmod +x ~/guestbook-service/mvnw
```

8. Click the pencil icon to open the Cloud Shell code editor.



Task 1. Enable Stackdriver Trace API

In this task, you enable Stackdriver Trace API in order to use Stackdriver Trace to store your trace data.

1. In the Cloud Shell, enter the following command:

```
gcloud services enable cloudtrace.googleapis.com
```

Task 2. Add the Spring Cloud GCP Trace starter

In this task, you add the Spring Cloud GCP Trace starter to the `pom.xml` files for the guestbook service and the guestbook frontend applications.

1. In the Cloud Shell code editor, open `~/guestbook-service/pom.xml`.
2. Insert the following new dependency at the end of the dependencies section, just before the closing `</dependencies>` tag:

```
<dependency>  
  <groupId>org.springframework.cloud</groupId>
```



```
<artifactId>spring-cloud-gcp-starter-trace</artifactId>
</dependency>
```

3. In the Cloud Shell code editor, open `~/guestbook-frontend/pom.xml`.
4. Insert the following new dependency at the end of the dependencies section, just before the closing `</dependencies>` tag:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-gcp-starter-trace</artifactId>
</dependency>
```

Task 3. Configure applications

You get full trace capability simply by including the starters. However, only a small percentage of all requests have their traces sampled and stored by default.

In this task, you disable trace completely for the default profile and configure trace sampling for all requests in the `cloud` profile.

Disable trace for testing purposes

For testing purposes, you disable trace in the `application.properties` files used for the local profile.

1. In the Cloud Shell code editor, open `guestbook-service/src/main/resources/application.properties`.
2. Add the following property to disable tracing in the guestbook service:

```
spring.cloud.gcp.trace.enabled=false
```

3. In the Cloud Shell code editor, open `guestbook-frontend/src/main/resources/application.properties`.
4. Add the following property to disable tracing in the guestbook frontend application:

```
spring.cloud.gcp.trace.enabled=false
```

Enable trace sampling for the cloud profile for the guestbook backend

For the `cloud` profile for the guestbook backend, you enable trace sampling for all of the requests in the `application.properties` file used for the `cloud` profile.

1. In the Cloud Shell code editor, open the guestbook service `cloud` profile: `guestbook-service/src/main/resources/application-cloud.properties`.
2. Add the following properties to enable the tracing detail needed in the guestbook service:

```
spring.cloud.gcp.trace.enabled=true
spring.sleuth.sampler.probability=1
spring.sleuth.web.skipPattern=(^cleanup.*|.+favicon.*)
```

Enable trace sampling for the cloud profile for the guestbook frontend

For the `cloud` profile for the frontend application, you enable trace sampling for all of the requests in the `application.properties` file used for the `cloud` profile.

1. In the Cloud Shell code editor, create a properties file for the guestbook frontend application `cloud` profile: `guestbook-frontend/src/main/resources/application-cloud.properties`.
2. Add the following properties to enable the tracing detail needed in the guestbook frontend application:

```
spring.cloud.gcp.trace.enabled=true
spring.sleuth.sampler.probability=1
spring.sleuth.web.skipPattern=(^cleanup.*|.+favicon.*)
```

Task 4. Set up a service account

In this task, you create a service account with permissions to propagate trace data to Stackdriver Trace.

1. In Cloud Shell enter the following commands to create a service account specific to the guestbook application.

```
gcloud iam service-accounts create guestbook
```

2. Add the Editor role for your project to this service account.

```
export PROJECT_ID=$(gcloud config list --format 'value(core.project)')
gcloud projects add-iam-policy-binding ${PROJECT_ID} \
  --member serviceAccount:guestbook@${PROJECT_ID}.iam.gserviceaccount.com \
  --role roles/editor
```

Warning

This action creates a service account with the Editor role. In your production environment, you should assign only the roles and permissions that the application needs.

3. Generate the JSON key file to be used by the application to identify itself using the service account.

```
gcloud iam service-accounts keys create \
  ~/service-account.json \
  --iam-account guestbook@${PROJECT_ID}.iam.gserviceaccount.com
```

This command creates service account credentials that are stored in the `$HOME/service-account.json` file.

Warning

Treat the `service-account.json` file as your own username/password. Do not share this information.

Task 5. Run the application locally with your service account

In this task you start both the frontend application and backend service application using the additional `spring.cloud.gcp.credentials.location` property to specify the location of the service account credential that you created.

1. Change to the `guestbook-service` directory.

```
cd ~/guestbook-service
```

2. Start the guestbook backend service application.

```
./mvnw spring-boot:run -Dserver.port=8081 -Dspring.profiles.active=cloud \
-Dspring.cloud.gcp.credentials.location=file:///HOME/service-account.json
```

This takes a minute or two to complete and you should wait until you see that the GuestbookApplication is running.

```
Started GuestbookApplication in 20.399 seconds (JVM running...)
```

3. Open a new Cloud Shell tab by clicking the plus (+) icon to the right of the title of the current Cloud Shell tab.

4. Change to the `guestbook-frontend` directory.

```
cd ~/guestbook-frontend
```

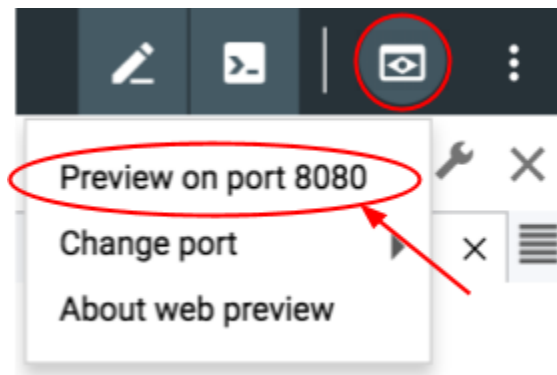
5. Start the guestbook frontend application.

```
./mvnw spring-boot:run -Dspring.profiles.active=cloud \
-Dspring.cloud.gcp.credentials.location=file:///HOME/service-account.json
```

6. Click **Web Preview** ().

7. Select **Preview on port 8080** to access the application on port 8080.

A new browser tab displays the connection to the frontend application. You may have to wait for a minute or two for both the frontend application and backend service application to start.



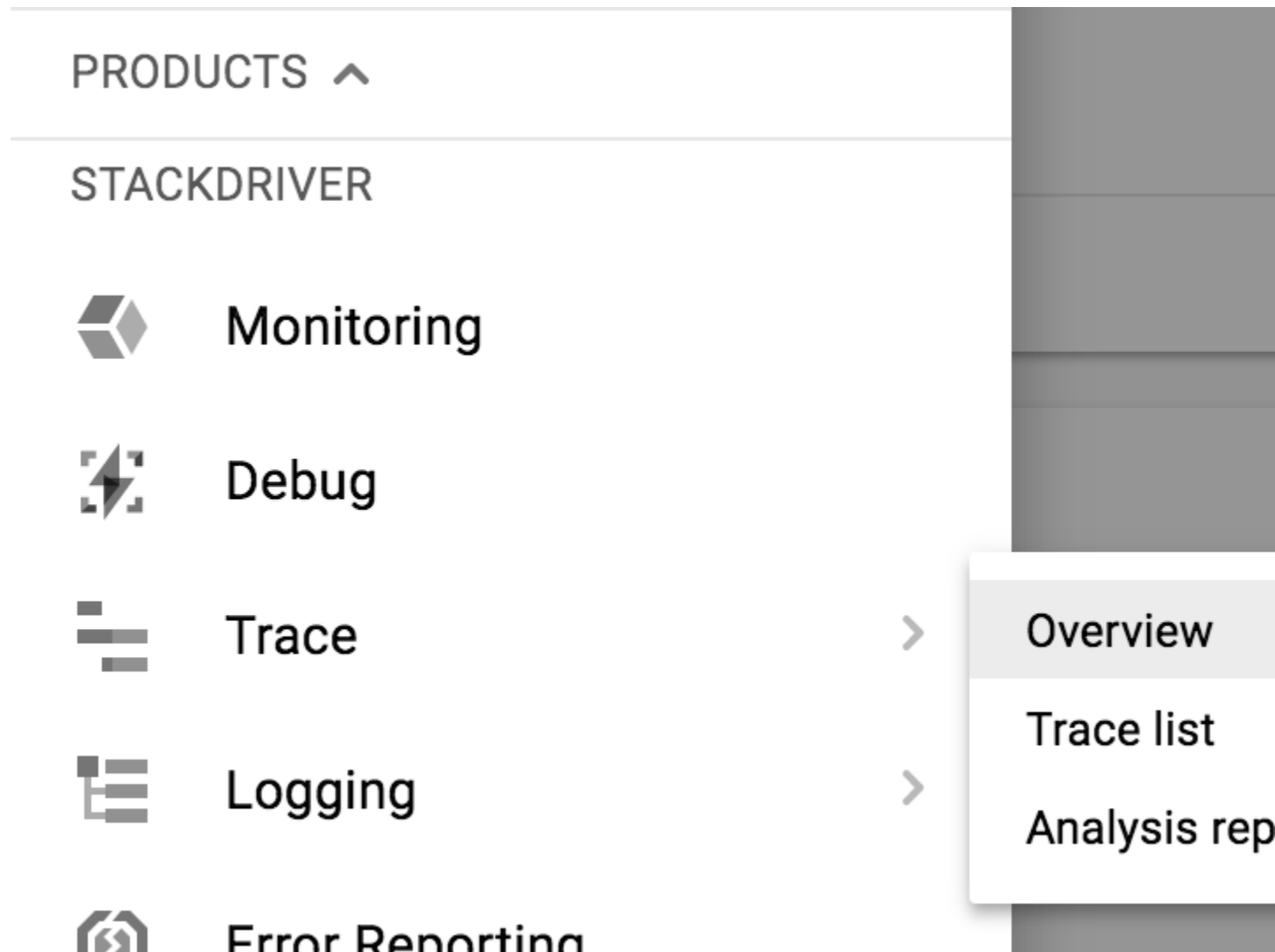
8. Post a message using the Cloud Shell web preview of the application to generate trace data or open a third Cloud Shell tab and make a request from Cloud Shell.

```
curl http://localhost:8080
```

Task 6. Examine the traces

In this task, you examine trace data using Stackdriver Trace to confirm that the data was propagated by Spring Cloud Sleuth to Trace.

1. Open the GCP console browser tab.
2. In the Navigation Menu navigate to **Stackdriver > Trace > Trace List**.
The traces for your requests are listed.



3. At the top of the window, set the time range to 1 hour.
By default, **Auto reload** is on. New trace data will take up to 30 seconds to appear.

Trace list

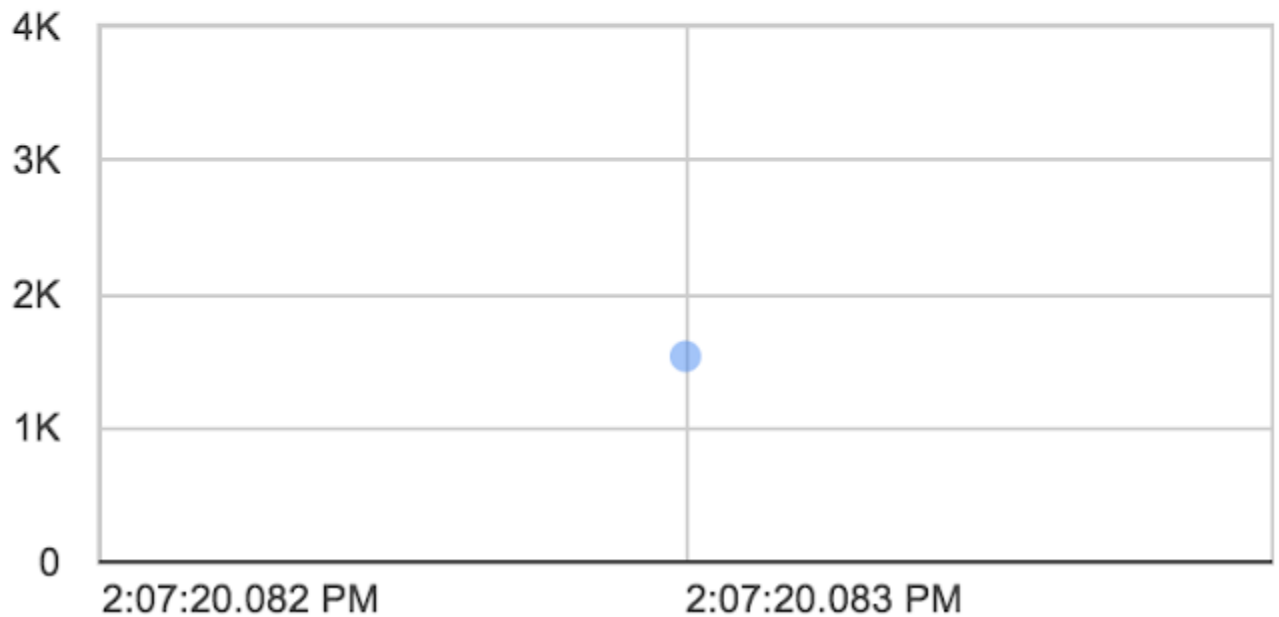


UNDO ZOOM

|| AUTO RE

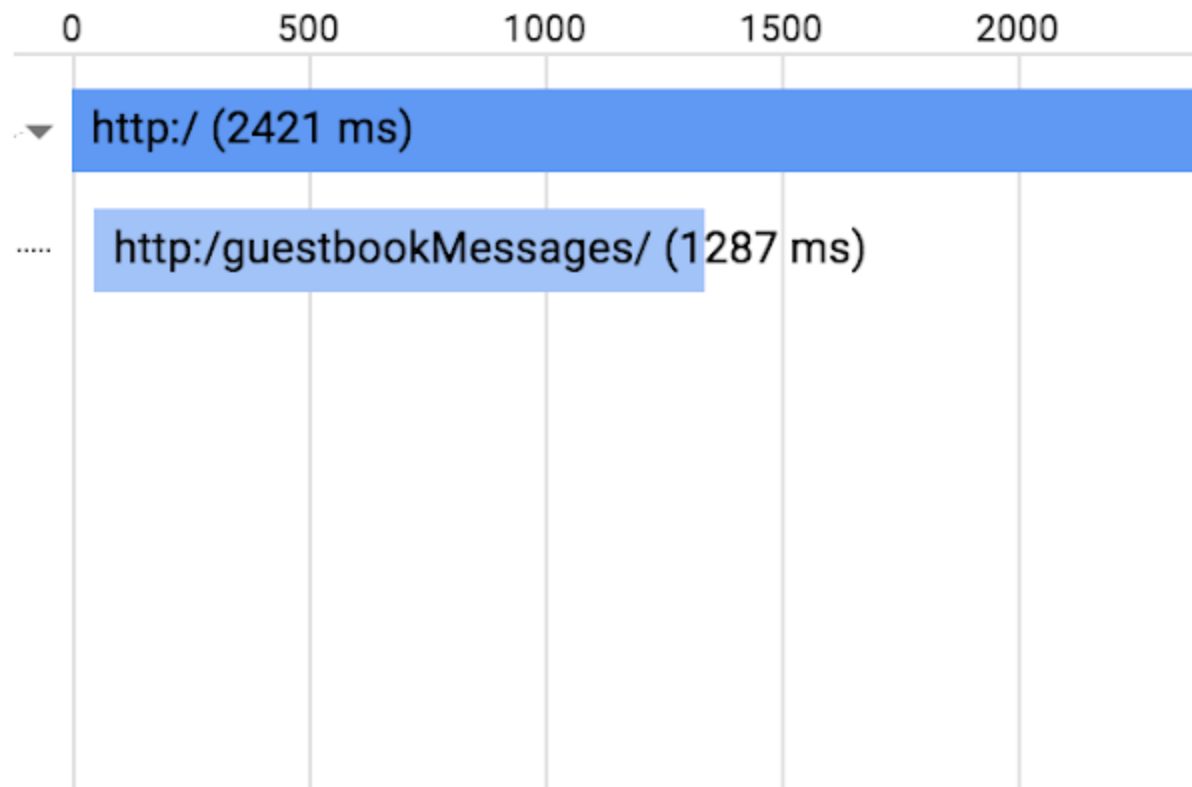
Request filter

(ms) Click and drag along the graph to zoom in to the selected area



4. Click the blue dot to view trace detail.

Timeline



End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you. You'll be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**. The number of stars indicates your rating:

- 1 star = Very dissatisfied

- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, use the **Support** tab.

Copyright 2019 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.