

End Lab 01:20:08

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: student-01-cc1bb6995a7b@qwiklab

Password: m9J6tgf2hBW

GCP Project ID: qwiklabs-gcp-01-07ee1a4704c3

Region: us-central1

Zone: us-central1-a

# Site Reliability Troubleshooting with Stackdriver APM [ACE]

1 hour 30 minutes Free ★★★★☆ Rate Lab

## Overview

The objective of this lab is to familiarize yourself with the specific capabilities of Cloud Monitoring to monitor GKE cluster infrastructure, Istio, and applications deployed on this infrastructure.

## What you'll do

- Create a GKE cluster
- Deploy a microservices application to it
- Define latency and error SLIs and SLOs for it
- Configure Cloud Monitoring to monitor your SLIs
- Deploy a breaking change to the application and use Cloud Monitoring to troubleshoot and resolve the issues that result
- Validate that your resolution addresses the SLO violation

Overview
Environment Setup
Infrastructure setup
Deploy application
Develop Sample SLOs and SLIs
Configure Latency SLI
Configure Availability SLI
Deploy new release

## What you'll learn

- How to deploy a microservices application on an existing GKE cluster
- How to select appropriate SLIs/SLOs for an application
- How to implement SLIs using Cloud Monitoring features
- How to use Cloud Trace, Profiler, and Debugger to identify software issues

## Prerequisites

- Google Cloud Platform account and project with billing account
- Basic knowledge of Kubernetes
- Basic knowledge of Cloud Monitoring
- Basic knowledge of troubleshooting process

## Environment Setup

### Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click Start Lab, shows how long Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access the Google Cloud Platform for the duration of the lab.

### What you need

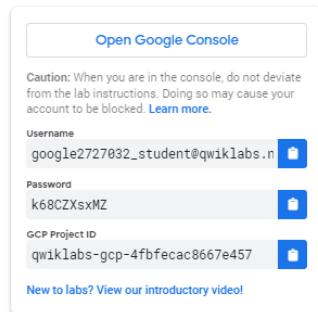
To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

**Note:** If you already have your own personal GCP account or project, do not use it for this lab.

### How to start your lab and sign in to the Console

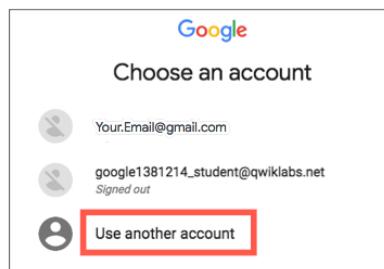
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



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.

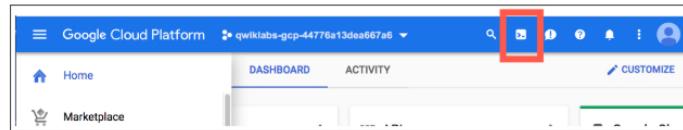
**Note:** You can view the menu with a list of GCP Products and Services by clicking the **Navigation menu** at the top-left, next to "Google Cloud Platform".



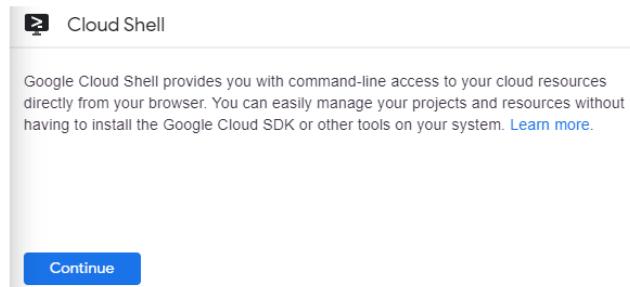
## Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Google Cloud Shell provides command-line access to your GCP resources.

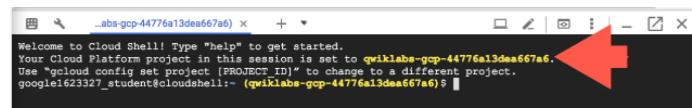
1. In GCP console, on the top right toolbar, click the Open Cloud Shell button.



2. Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT\_ID*. For example:



**gcloud** is the command-line tool for Google Cloud Platform. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

Output:

```
Credentialed accounts:
- <myaccount>@<mydomain>.com (active)
```

Example output:

```
Credentialed accounts:  
- google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

Output:

```
[core]  
project = <project_ID>
```

Example output:

```
[core]  
project = qwiklabs-gcp-44776a13dea667a6
```

Full documentation of **gcloud** is available on [Google Cloud gcloud Overview](#).

## Infrastructure setup

In this lab you will connect to a Google Kubernetes Engine cluster and validate that it's been created correctly.

Set the zone in `gcloud`:

```
gcloud config set compute/zone us-west1-b
```

Set the project ID:

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

Verify that the cluster named `shop-cluster` has been created:

```
gcloud container clusters list
```

Your cluster status will say PROVISIONING. Wait a moment and run the command above again. Repeat until the status is RUNNING. This could take several minutes.

While you're waiting, set up your Monitoring workspace to monitor the application on your cluster.

## Create a Monitoring workspace

You will now setup a Monitoring workspace that's tied to your Qwiklabs GCP Project. The following steps create a new account that has a free trial of Monitoring.

1. In the Google Cloud Platform Console, click on **Navigation menu > Monitoring**.

2. Wait for your workspace to be provisioned.

When the Monitoring dashboard opens, your workspace is ready.





Go back to your Cloud Shell and check again to see if the cluster has finished provisioning:

```
gcloud container clusters list
```

Once your cluster has RUNNING status, get the cluster credentials:

```
gcloud container clusters get-credentials shop-cluster --zone us-west1-b
```

Your output should look like this:

```
Fetching cluster endpoint and auth data.  
kubeconfig entry generated for shop-cluster.
```

Verify that the nodes have been created:

```
kubectl get nodes
```

Your output should look like this:

NAME	STATUS	ROLES	AGE
<b>VERSION</b>			
gke-shop-cluster-demo-default-pool1-24748028-3nwh	Ready	<none>	
4m v1.13.7-gke.8			
gke-shop-cluster-demo-default-pool1-24748028-3z1g	Ready	<none>	
4m v1.13.7-gke.8			
gke-shop-cluster-demo-default-pool1-24748028-4ksd	Ready	<none>	
4m v1.13.7-gke.8			
gke-shop-cluster-demo-default-pool1-24748028-f2f2	Ready	<none>	
4m v1.13.7-gke.8			
gke-shop-cluster-demo-default-pool1-24748028-gcb3	Ready	<none>	
4m v1.13.7-gke.8			

## Deploy application

In this module, you will deploy a microservices application called Hipster Shop to your cluster to create an actual workload you can monitor.

Run the following to clone the repo:

```
git clone -b APM-Troubleshooting-Demo-2  
https://github.com/blipzimmerman/microservices-demo-1
```

Download and install `skaffold`:

```
curl -Lo skaffold  
https://storage.googleapis.com/skaffold/releases/v0.36.0/skaffold-linux-amd64 && chmod +x skaffold && sudo mv skaffold /usr/local/bin
```

Install the app using `skaffold`:

```
cd microservices-demo-1  
skaffold run
```

Confirm everything is running correctly:

```
kubectl get pods
```

Your output should look like this:

NAME	READY	STATUS	RESTARTS

adservice-55t94ctd9c-4ivml	1/1	Running	0
cartservice-6f4946f9b8-6wtff	1/1	Running	2
checkoutservice-5688779d8c-16crl	1/1	Running	0
currencyservice-665d6f4569-b4sbm	1/1	Running	0
emailservice-684c89bcb8-h48sq	1/1	Running	0
frontend-67c8475b7d-vktsn	1/1	Running	0
loadgenerator-6d646566db-p422w	1/1	Running	0
paymentservice-858d89d64c-hmpkg	1/1	Running	0
productcatalogservice-bcd85cb5-d6xp4	1/1	Running	0
recommendationservice-685d7d6cd9-pxd9g	1/1	Running	0
redis-cart-9b864d47f-c9xc6	1/1	Running	0
shippingservice-5948f9fb5c-vndcp	1/1	Running	0
20m			

Re-run the command until all pods are reporting a Running status before moving to the next step.

Get the **external IP** of the application:

```
export EXTERNAL_IP=$(kubectl get service frontend-external | awk 'BEGIN {
cnt=0; } { cnt+=1; if (cnt > 1) print $4; }')
```

Finally, confirm that the app is up and running:

```
curl -o /dev/null -s -w "%{http_code}\n" http://$EXTERNAL_IP
```

**Note:** You may need to run this command a second time if you get a 500 error.

Your confirmation will look like this:

```
200
```

Download the source and put the code in the Cloud Source Repo:

```
./setup_csr.sh
```

Now that the application has been deployed, set up monitoring for the application.

## Resources

- [Microservices Demo Application](#)

NOTE: This lab uses a [fork](#) of this application build to aid in the troubleshooting exercises.

- [Skaffold](#)

## Develop Sample SLOs and SLIs

Before implementing any monitoring, review the introduction to the chapter on [Service Level Objectives from the SRE Book](#):

It's impossible to manage a service correctly, let alone well, without understanding which behaviors really matter for that service and how to measure and evaluate those behaviors. To this end, we would like to define and deliver a given level of service to our users, whether they use an internal API or a public product.

We use intuition, experience and an understanding of what users want to define **service level indicators (SLIs)**, **objectives (SLOs)** and **agreements (SLAs)**. These measurements describe basic properties of metrics that matter what values we

Measurements describe basic properties of metrics that matter, what values we want those metrics to have and how we'll react if we can't provide the expected service. Ultimately, choosing appropriate metrics helps to drive the right action if something goes wrong and also gives an SRE team confidence that a service is healthy.

An SLI is a service level indicator: A carefully defined quantitative measure of some aspect of the level of service that is provided.

Most services consider

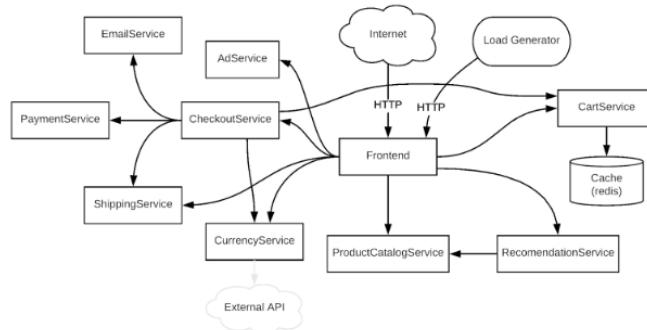
**request latency:** how long it takes to return a response to a request as a key SLI.

Other common SLIs include the **error rate**, often expressed as a fraction of all requests received and **system throughput**, typically measured in requests per second. Another kind of SLI important to SREs is **availability** or the fraction of the time that a service is usable. It is often defined in terms of the fraction of well-formed requests that succeed.

**Durability:** the likelihood that data will be retained over a long period of time is equally important for data storage systems. The measurements are often aggregated: i.e., raw data is collected over a measurement window and then turned into a rate, average, or percentile.

Now that you have established a basic understanding, define the SLIs and SLOs for your application. Given that the application itself serves end user ecommerce traffic, it's going to be very important that user experience remains constant and that performance is good. You will monitor SLIs for request latency, error rate, throughput and availability.

Application Architecture



It's impossible to develop SLIs without understanding how the application is built. Details are in the original [repository](#), but for this lab, it suffices to understand that:

- Users access the application through the Frontend.
  - Purchases are handled by CheckoutService.
  - CheckoutService depends on CurrencyService to handle conversions.
  - Other services such as RecommendationService, ProductCatalogService and AdsService are used to provide the frontend with content needed to render the page.

## Service Level Indicators and Objectives

The following SLIs and SLOs are selected based on the end-user experience and the theoretical impact to users and business objectives.

SLI	Metric	Description	SLO
Request latency	Front end latency	Measures how long a user is waiting for the page to load. A high latency typically correlates to a negative user experience	99% of requests from the previous 60 minute period are services in under 3 seconds
Error rate	Front end error rate	Measures the error rate experienced by users. A high error rate likely indicates an issue.	0 Errors in the previous 60 minute period

Error rate	Checkout error rate	Measures the error rate experienced by other services calling the checkout service. A high error rate likely indicates an issue.	0 Errors in the previous 60 minute period
Error rate	Currency Service error rate	Measures the error rate experienced by other services calling the currency service. A high error rate likely indicates an issue.	0 Errors in the previous 60 minute period
Availability	Front end success rate	Measures the rate of successful requests as a way to determine the availability of the service. A low success rate likely indicates that users are having a poor experience.	99% of requests are successful over the previous 60 minute period

## Configure Latency SLI

Now that you have SLOs and SLIs defined, you can implement cloud monitoring. The metrics you are interested in are already being collected. You will create alerting policies for each of your SLOs.

### Front End Latency

In the Monitoring tab, click **Alerting** and select **Create Policy**.

Name the policy **Latency Policy**.

Click **Add Condition** and specify the metric and condition that will be used to trigger the Alerting Policy. The condition will let you know when you're experiencing performance issues that are impacting user experience.

As described in the *Service Level Indicators and Objectives* table above, you will use the 99th percentile front end latency as the SLI.

Add the following into the **Find resource type and metric** field then select the following from the dropdown menu:

```
custom.googleapis.com/opencensus/grpc.io/client/roundtrip_latency
```

**Note:** You may need to refresh the page to see the above metric.

In the **Resource Type**, select the **Global** option.

Click into the **Filter** field and select the **opencensus\_task** item. Click on the first default Value, then click **Apply**.

Next, set the **Aggregator** to **99th percentile**.

Next, in the **Configuration** area, set the options as follows:

- Condition triggers if **Any time series violates**
- Condition: **is above**
- Threshold: **500**
- For: **Most recent value**

## Configuration

Condition triggers if

Any time series violates

Condition	Threshold	For
<b>is above</b>	<b>500</b>	<b>most recen...</b>

Click **Add**.

Click **Save**. You've configured Cloud Monitoring to monitor your frontend latency SLI!

## Configure Availability SLI

Next, configure Cloud Monitoring to monitor service availability by creating another Alerting Policy.

### Front End Availability

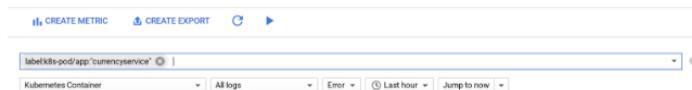
Start by monitoring the error rate for the front end service, since that's where user experience is going to be most directly impacted. As discussed above, you're going to consider any failures observed to be an SLO violation. Create an alerting policy that will trigger an incident if any failures are observed.

An easy way to trigger on a particular failure is to use log-based metrics.

In the Google Cloud Platform Console, click **Logging**.

Configure the filter as follows:

- In Resource type select **Kubernetes Container**.
- In Any Log Level select **ERROR**.
- In the filter bar add: `label:k8s-pod/app:"currencyservice"`



Click **Create Metric**.

Name the metric `Error_Rate_SLI` and click **Create Metric** to save the log based metric:

### Metric Editor

#### Name

#### Description

#### Labels

[+ Add item](#)

**Units** ⓘ (Optional)

**Type** ⓘ

**Create Metric** **Cancel**

You now see the metric listed on the Logs-based metrics page. To create an alert for this metric, click the 3 dots at the end of the row and select **Create Alert from Metric**.

User-defined Metrics

User defined logs-based metrics that count the number of log entries that match a given filter.

Name	Type	Description	Previous Month Usage	Usage (MTD)	Filter
user/error_rate_SLI	Counter		0 K	0 K	resource.type='file_container' severity='ERROR'

⋮

- [Edit metric](#)
- [Delete metric](#)
- [View logs for metric](#)
- [View in Metrics Explorer](#)
- [Create alert from metric](#) ⚡

Notice the resource type and metric have already been filled in.

Name the condition "Error Rate SLI".

Click the **Show Advanced Options** link and set the following:

- Aligner: **rate**

In Configuration, use **0.5** as your Threshold for **2 minutes**.

Then **Save** the condition.

In the subsequent screen, name your new policy "Error Rate SLI", and **Save it**.

As expected, there are no failures, and your application is currently meeting its availability SLO!

## Deploy new release

Now that you have configured SLI monitoring, you're ready to measure the impact of application changes on user experience. See what happens when you deploy a new release of the application.

Next you'll modify the Kubernetes manifests for the services which have new releases and then run skaffold to deploy the application again.

### Update YAML files

Open the Code Editor in Cloud Shell:



Find the **microservices-demo-1** folder and open the **kubernetes-manifests** folder within it.

**Update** the kubernetes manifests to pull the new images by:

1. Replacing the **accl-demo** image tag with **rel013019**

2. Adding: `imagePullPolicy: Always`

in the following files:

- kubernetes\_manifests/recommendationservice.yaml
- kubernetes\_manifests/currencyservice.yaml
- kubernetes\_manifests/frontend.yaml

As an example, here's the original version of the `recommendationservice.yaml` file:

```
 26   containers:
 27     - name: server
 28       image: gcr.io/accl-19-dev/recommendationservice:accl-demo
 29       ports:
 30         - containerPort: 8080
 31         readinessProbe:
 32           periodSeconds: 5
 33           exec:
 34             command: ["./bin/grpc_health_probe", "-addr=:8080"]
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 550
 551
 552
 553
 554
 555
 556
 557
 558
 559
 559
 560
 561
 562
 563
 564
 565
 566
 567
 568
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 580
 581
 582
 583
 584
 585
 586
 587
 588
 589
 589
 590
 591
 592
 593
 594
 595
 596
 597
 598
 599
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 609
 610
 611
 612
 613
 614
 615
 616
 617
 618
 619
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 659
 660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 679
 680
 681
 682
 683
 684
 685
 686
 687
 688
 689
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 709
 710
 711
 712
 713
 714
 715
 716
 717
 718
 719
 719
 720
 721
 722
 723
 724
 725
 726
 727
 728
 729
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 749
 750
 751
 752
 753
 754
 755
 756
 757
 758
 759
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768
 769
 769
 770
 771
 772
 773
 774
 775
 776
 777
 778
 779
 779
 780
 781
 782
 783
 784
 785
 786
 787
 788
 789
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809
 809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 859
 860
 861
 862
 863
 864
 865
 866
 867
 868
 869
 869
 870
 871
 872
 873
 874
 875
 876
 877
 878
 879
 879
 880
 881
 882
 883
 884
 885
 886
 887
 888
 889
 889
 890
 891
 892
 893
 894
 895
 896
 897
 898
 899
 899
 900
 901
 902
 903
 904
 905
 906
 907
 908
 909
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1019
 1020
 1021
 1022
 1023
 1024
 1025
 1026
 1027
 1028
 1029
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079
 1079
 1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1089
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1099
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1129
 1130
 1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1138
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1179
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1189
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1198
 1199
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1238
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1248
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1259
 1259
 1260
 1261
 1262
 1263
 1264
 1265
 1266
 1267
 1268
 1269
 1269
 1270
 1271
 1272
 1273
 1274
 1275
 1276
 1277
 1278
 1279
 1279
 1280
 1281
 1282
 1283
 1284
 1285
 1286
 1287
 1288
 1288
 1289
 1290
 1291
 1292
 1293
 1294
 1295
 1296
 1297
 1297
 1298
 1299
 1299
 1300
 1301
 1302
 1303
 1304
 1305
 1306
 1307
 1308
 1309
 1309
 1310
 1311
 1312
 1313
 1314
 1315
 1316
 1317
 1318
 1318
 1319
 1320
 1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328
 1329
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1338
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348
 1348
 1349
 1350
 1351
 1352
 1353
 1354
 1355
 1356
 1357
 1358
 1359
 1359
 1360
 1361
 1362
 1363
 1364
 1365
 1366
 1367
 1368
 1369
 1369
 1370
 1371
 1372
 1373
 1374
 1375
 1376
 1377
 1378
 1378
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1388
 1389
 1390
 1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1398
 1399
 1400
 1401
 1402
 1403
 1404
 1405
 1406
 1407
 1408
 1409
 1409
 1410
 1411
 1412
 1413
 1414
 1415
 1416
 1417
 1418
 1418
 1419
 1420
 1421
 1422
 1423
 1424
 1425
 1426
 1427
 1428
 1428
 1429
 1430
 1431
 1432
 1433
 1434
 1435
 1436
 1437
 1438
 1438
 1439
 1440
 1441
 1442
 1443
 1444
 1445
 1446
 1447
 1448
 1448
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457
 1458
 1459
 1459
 1460
 1461
 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1468
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1478
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1488
 1489
 1490
 1491
 1492
 1493
 1494
 1495
 1496
 1497
 1498
 1498
 1499
 1500
 1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508
 1509
 1509
 1510
 1511
 1512
 1513
 1514
 1515
 1516
 1517
 1518
 1518
 1519
 1520
 1521
 1522
 1523
 1524
 1525
 1526
 1527
 1528
 1529
 1529
 1530
 1531
 1532
 1533
 1534
 1535
 1536
 1537
 1538
 1538
 1539
 1540
 1541
 1542
 1543
 1544
 1545
 1546
 1547
 1548
 1548
 1549
 1550
 1551
 1552
 1553
 1554
 1555
 1556
 1557
 1558
 1559
 1559
 1560
 1561
 1562
 1563
 1564
 1565
 1566
 1567
 1568
 1568
 1569
 1570
 1571
 1572
 1573
 1574
 1575
 1576
 1577
 1578
 1578
 1579
 1580
 1581
 1582
 1583
 1584
 1585
 1586
 1587
 1588
 1588
 1589
 1590
 1591
 1592
 1593
 1594
 1595
 1596
 1597
 1598
 1598
 1599
 1600
 1601
 1602
 1603
 1604
 1605
 1606
 1607
 1608
 1609
 1609
 1610
 1611
 1612
 1613
 1614
 1615
 1616
 1617
 1618
 1618
 1619
 1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1628
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1638
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1648
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1659
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1668
 1669
 1670
 1671
 1672
 1673
 1674
 1675
 1676
 1677
 1678
 1678
 1679
 1680
 1681
 1682
 1683
 1684
 1685
 1686
 1687
 1688
 1688
 1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1698
 1699
 1700
 1701
 1702
 1703
 1704
 1705
 1706
 1707
 1708
 1709
 1709
 1710
 1711
 1712
 1713
 1714
 1715
 1716
 1717
 1718
 1718
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727
 1728
 1728
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1738
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1778
 1779
 1780
 1781
 1782
 1783
 1784
 1785
 1786
 1787
 1788
 1788
 1789
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1798
 1799
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1809
 1809
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1818
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1828
 1829
 1830
 1831
 1832
 1833
 1834
 1835
 1836
 1837
 1838
 1838
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1848
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1868
 1869
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1878
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1888
 1889
 1890
 1891
 1892
 1893
 18
```

frontend-external service and click on the Endpoint URL.

Once on the Hipster Shop website, click on a **Buy** and/or **Add to Cart** for a couple of items to send some traffic.

## Latency SLO Violation - Find the Problem

In this exercise you will use Cloud Application Performance Management tools (APM) to identify and resolve an issue causing poor application latency.

First see if everything is still OK with the application after deploying the new version.

Go to **Monitoring Overview** page. Click the **autorefresh arrows** in the top ribbon so you will always be looking at the latest information.



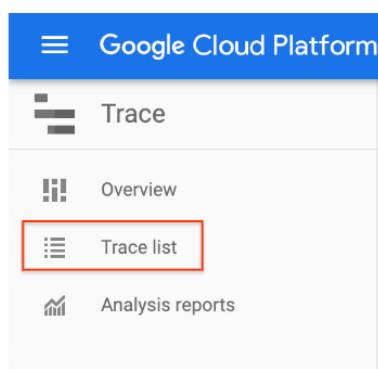
Click **Alerting** and select **Latency Policy** under Policies. You can see that your latency is significantly exceeding the threshold that was set up.

First, click **Acknowledged** so that further notification escalation does not take place.



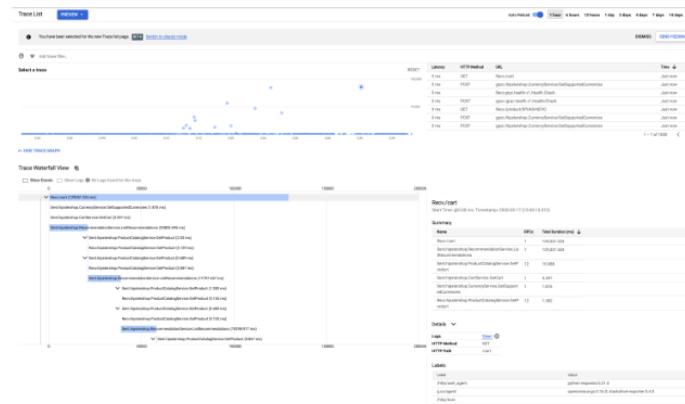
The best way to analyze latency issues is by using Trace. In the Google Cloud Platform Console, click **Trace**.

You're now in the Console. The initial overview is useful, but you need to get to the next level of detail. Open the **Trace List** page.



Click **Auto Reload**. Notice the scatter plot at the top of the page and that, around the time of the alert, there are a large number of outlier requests.

Wait a minute or two, to gather data, then click on one of the outlier traces to see the specifics about what is going on.



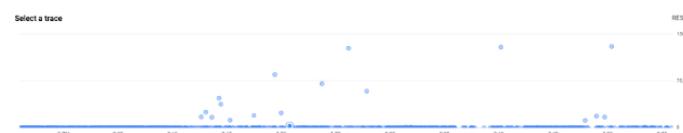
Notice the Span name (which represents the service or function that is being called) is either `/cart/` or `/cart/checkout/`.

To help understand how this trace compares with similar ones prior to the issue, enter `Recv./cart` in the **Add trace filter** to filter for all the cart operations and look for similar traces.

Set the time period to **1 hour** so that it includes traces that occurred before the issue.



Click on an example trace from before the issue occurred



Notice that in this similar trace `ListRecommendations` was only called once. However, after the most recent deploy, `ListRecommendations` is being called many times per request, causing significant additional latency.

You can conclude that the issue with these outliers is caused by multiple calls to `ListRecommendations`.

## Deploy Change to Address Latency

In order to address the latency issue that the last release created, you need to roll out another version that fixes the broken code. You will next modify the Kubernetes manifests for the services that contained the broken code.

To deploy a fix return to Cloud Shell and open the **Source Code Editor**. You'll be modifying the following files:

- `kubernetes_manifests/recommendationservice.yaml`
- `kubernetes_manifests/frontend.yaml`

Modify the image tag `rel013019` with `rel013019fix` so the `image` should look like this:

```
  containers:  
    - name: server
```

```
image: gcr.io/accl-19-dev/frontend:rel013019fix  
imagePullPolicy: Always
```

Cloud Shell

File Edit Selection View Go Help

Frontend

- ▶ cgroups-metadata
- ▶ datapath
- ▶ dataprep
- ▶ gke-enterprise-demo
- ▶ gs-spring-boot
- ▶ hooks
- ▶ info
- ▶ istio-1.0.1
- ▶ k8s-costelab
- ▶ kubernetes
- ▶ linux-amd64
- ▶ microservices-demo
- ▶ microservices-demo-1
- ▶ microservices-frontend
- ▶ docs
- ▶ img
- ▶ istio-manifests
- ▶ kubernetes-manifests
- ▶ adservice.yaml
- ▶ cartservice.yaml
- ▶ checkoutservice.yaml
- ▶ currencieservice.yaml
- ▶ emailservice.yaml
- ▶ frontend.yaml
- ▶ loadgenerator.yaml

frontend.yaml ▶

```
1 # Copyright 2018 Google LLC
2 #
3 # Licensed under the Apache License, Version 2.0 (the "License");
4 # you may not use this file except in compliance with the License.
5 # You may obtain a copy of the License at
6 #
7 #     http://www.apache.org/licenses/LICENSE-2.0
8 #
9 # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14
15 apiVersion: extensions/v1beta1
16 kind: Deployment
17 metadata:
18   name: frontend
19 spec:
20   template:
21     metadata:
22       labels:
23         app: frontend
24   spec:
25     containers:
26       - name: server
27         image: gcr.io/accl-19-dev/frontend:rel013019fix
28         imagePullPolicy: Always
```

**Save the files.**

Return to the Cloud Shell prompt and redeploy the images with the fixes in them:

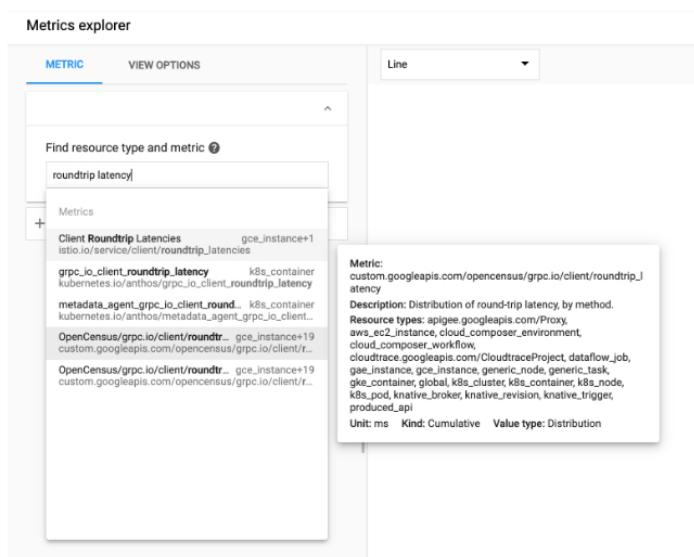
`skaffold run`

## Validate Fix

Now that you've rolled back the breaking change, verify that your application is back to a healthy state.

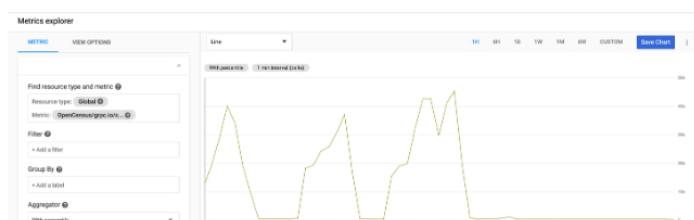
Return to **Monitoring** and click **Metrics Explorer**.

In the search field, enter **roundtrip latency** and select [custom.googleapis.com/opencensus/grpc.io/client/roundtrip\\_latency](https://custom.googleapis.com/opencensus/grpc.io/client/roundtrip_latency)



Select **Global** as a resource type. Click [here](#) link in the **Aggregator** field.

Change the chart type to **Line**. You should see a chart that shows an immediate decrease in latency (and if you don't, wait a minute).





Now, see if the incidents are resolved. Return to the [Monitoring Overview](#).

You should notice two things - you no longer have an incident and there are events letting you know that the incident has been resolved. Again, if you don't see an Incident resolved message, wait a couple of minutes.

Your monitoring was able to correctly identify a change that caused user experience (as measured by latency) to degrade, you were able to identify the root cause and you've rolled back the breaking change! In the next section, see how Monitoring can help you resolve an issue with availability.

## Error Rate SLO Violation - Find the Problem

In this exercise you will use Cloud Monitoring Application Performance Management tools (APM) to troubleshoot an issue causing ERRORS in your application violating your error budget.

Click **Alerting** in Monitoring.

Look for a Error Rate SLI incident and click the **incidents** to learn more about what's going on. Incidents can take several minutes to be confirmed and listed as an incident. If an incident has not yet arrived, you can skip the Incident step below click on **Error Reporting** in the GCP console.

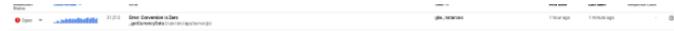
You can see that the pod is logging significantly more errors than it was previously.

**Acknowledged** the incident so that no further notification escalation takes place.

For an alert like this there are many places to start, but the easiest is Error Reporting.

Notice the Open Error Reporting incidents with a recent spike in occurrences. Click on the **Error: Conversion is Zero** to learn more about the error in question.



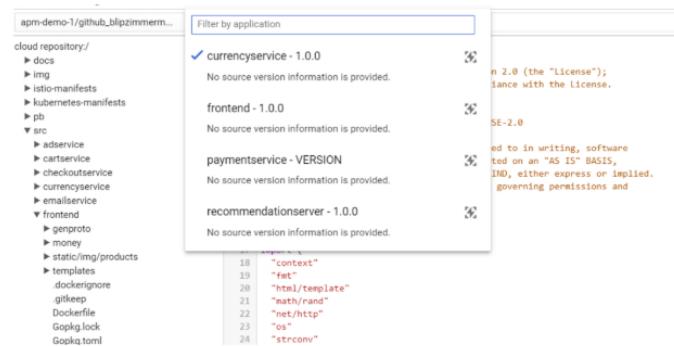


Look at the Stack Trace sample on the right. Here you can see what specific calls were related to the error.

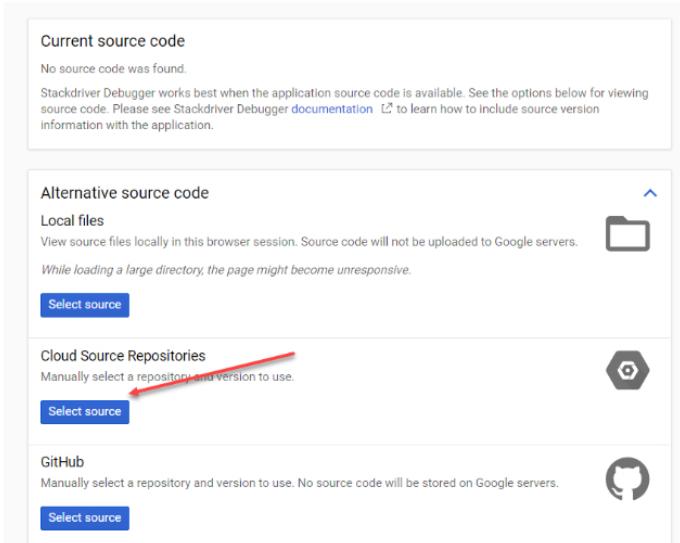


Click on the lowest call showing here: /usr/src/app/server.js:131

This will load you into Debugger. On the top bar click the **Service** and select the **currencyservice**.



Next, select the source code that is running from **Cloud Source Repositories**.

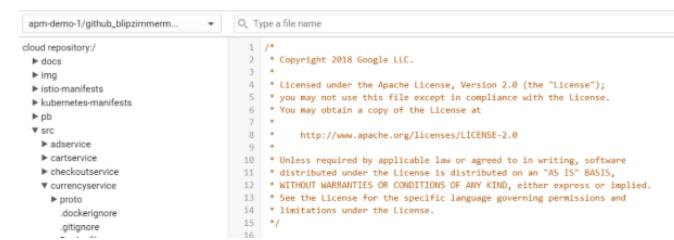


Select your source with:

- **Repository:** apm-qwiklabs-demo
- **Tagged version or branch:** APM-Troubleshooting-Demo-2

Then click **Select Source**.

In the left hand menu browse to `/src/currencyservice/server.js`.



```

17 require('@google-cloud/profiler').start({
18   serviceContext: {
19     service: 'currencyservice',
20     version: '1.0.0'
21   }
22 });
23 require('@google-cloud/trace-agent').start();
24 require('@google-cloud/debug-agent').start({
25   serviceContext: {
26     service: 'currencyservice',

```

Scroll down to around line 155 which is the function where the exception was thrown. You can see the logline **Conversion is Zero** that was referenced in error reporting.

```

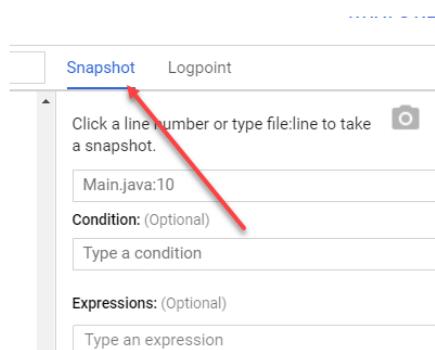
152 if (result.units > 0) {
153   logger.info(`conversion request successful`);
154 } else {
155   var stack = new Error(`Conversion is Zero`).stack;
156   logger.error(stack);
157 }
158 callback(null, result);

```



From the above code snippet you see this error is logged when `result.units < 0`. To troubleshoot this issue you'll use Snapshots to inspect the variables as the application progresses.

Make sure you have selected **Snapshot** in the top right:



Then click on the line number (155) you want to snapshot:

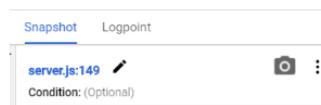
```

150 result.nanos = Math.floor(result.nanos);
151 result.currency_code = request.to_code;
152 if (result.units > 0) {
153   logger.info(`conversion request successful`);
154 } else {
155   var stack = new Error('Conversion is Zero').stack;
156   logger.error(stack);
157 }
158 callback(null, result);
159 });
160 } catch (err) {
161   logger.error(`conversion request failed: ${err}`);
162   callback(err.message);
163 }
164 }

```

For this exercise take snapshots at line 155, 141 and 149. Add additional snapshot points wherever you feel appropriate. The system will take a variable snapshot the next time that code is executed. While the application is waiting for the code to be executed next you can see a "Waiting for snapshot to hit...." notice.

When the snapshot is complete the right hand pane will display the variables for that given snapshot.



The screenshot shows a debugger interface with two main sections: 'Variables' and 'Call Stack'. The 'Variables' pane lists several objects and their properties. The 'Call Stack' pane shows the execution path from three specific lines of code.

Variables	Call Stack
<pre> ▶ data          #&lt;Object&gt; ▶ request      #&lt;Object&gt; ▶ from         #&lt;Object&gt; ▶ euros        #&lt;Object&gt; ▶ result       #&lt;Object&gt; stack          undefined callback       function wrappedCb() ▶ call         #&lt;ServerUnaryCall&gt; context        #&lt;Object&gt; </pre>	<pre> Call Stack _getCurrencyData    server.js:149 _getCurrencyData    server.js:100 convert            server.js:131 </pre>

Notice the Variable and Call Stack information. This information can provide extremely deep understanding of the path your code is taking including the variables and structures that exist as it takes that path, all without restarting the application or changing any code.

Click **result** to inspect all 3 snapshots finishing on line 155. Remember the error is triggered when `result.units` is NOT > 0. Inspecting the variables you can see that `result.units = NaN` (meaning 'not a number'). This is the issue that is causing the error.

The screenshot shows the 'Variables' pane with the 'result' object highlighted. The 'result' object has properties 'units' and 'nanos' both set to 'NaN'. Other variables like 'from' and 'euros' also have their 'units' properties highlighted.

Variables
<pre> ▶ data          #&lt;Object&gt; ▶ request      #&lt;Object&gt; ▶ from         #&lt;Object&gt;   currency_code USD   units         789   nanos        500000000 ▶ euros        #&lt;Object&gt; ▶ result       #&lt;Object&gt;   units         NaN   nanos        NaN stack          undefined callback       function wrappedCb() ▶ call         #&lt;ServerUnaryCall&gt; context        #&lt;Object&gt; </pre>

At this point you can conclude that the error is caused by a bug in the `convert` (or child) functions which sets the `result.units` to `0` resulting in a `0.00` price tag for the item being converted. Your troubleshooting along with the snapshot information and logs is a solid diagnosis of the issue.

So what is the bug that has caused this problem? From the code, `result.units` is set by the line 144 from `euros`, which was set in line 136 by operating on `from.units`

```

// Convert: from_currency --> EUR
const from = request.from;
const euros = _carry({
  units: from.units / data[from.currency_code],
  nanos: from.nanos / data[from.currency_code]
});

```

Inspecting the snapshots `euros.units` is also `NaN`, however, `from.units` is a valid number. Thus the issue happened when converting `from.units` to `euros`.

The screenshot shows the 'Variables' pane with the 'units' property of the 'from' and 'euros' objects highlighted. The 'from' object's 'units' is '8' and the 'euros' object's 'units' is 'NaN'.

Variables
<pre> ▶ data          #&lt;Object&gt; ▶ request      #&lt;Object&gt; ▶ from         #&lt;Object&gt;   currency_code USD   units         8   nanos        790000000 ▶ euros        #&lt;Object&gt;   units         NaN </pre>

nanos	NaN
result	undefined
stack	undefined
callback	function wrappedCb()
call	#<ServerUnaryCall>
context	#<Object>

You can conclude that the root cause is a bug in how `from.units` is converted into `euros.units` on line 137 and 8 is passed into `Data[]` which is actually a key value mapping of currency units (like EUR) into exchange rates. The corrected line 137 would use `from.to_currency` (aka USD) instead of `from.units` (aka 8).

Variables	
data	#<Object>
EUR	1.0
USD	1.1535
JPY	124.70
BGN	1.9558
CZK	25.626
DKK	7.4655
GBP	0.90423
HUF	321.16
PLN	4.2959
RON	4.6813

At this point you have determined the cause of the bug and can make the appropriate change. Based on the timing of the alert, this could have been caused by the latest deployment.

See if the previous branch **Master** had this code error on line 137.

Go back to the Console and inspect the code using Cloud **Source Repositories** (in the console menu under Tools).

Open the **apm-qwiklabs-demo** repository and select the **master** branch.

Browse on the left hand side to **src > currencieservice > server.js**. Notice on line 137 the proper dividend: `data[from.currency_code]` is used.

```

▼ src
  ► adbservice
  ► cartservice
  ► checkoutservice
  ▼ currencieservice
    ► proto
      □ .dockerignore
      □ .gitignore
      □ Dockerfile
      □ client.js
      □ genproto.sh
      □ package-lock.json
      □ package.json
      □ server.js
    ► emailservice
    ► frontend
    ► loadgenerator
    ► paymentservice

  125 /**
  126 * Converts between currencies
  127 */
  128 function convert (call, callback) {
  129   logger.info('received conversion request');
  130   try {
  131     _getCurrencyData(data) => {
  132       const request = call.request;
  133
  134       // Convert: from_currency --> EUR
  135       const from = request.from;
  136       const euros = _carry(
  137         units: from.units / data[from.currency_code],
  138         nanos: from.nanos / data[from.currency_code]
  139       );
  140
  141       euros.nanos = Math.round(euros.nanos);
  142
  143       // Convert: EUR --> to_currency
  144       const result = _carry(
  145         units: euros.units * data[request.to_code],
  146         nanos: euros.nanos * data[request.to_code]
  147       );
  148
  149       result.units = Math.floor(result.units);
  150       result.nanos = Math.floor(result.nanos);
  151       result.currency_code = request.to_code;
  
```

You have confirmation this bug was introduced in the latest push. To mitigate this problem you need to roll back to the previous version.

## Deploy Change to Address Error Rate

In order to fix this issue, you'll need to deploy a fix to your application. To do that, you'll need to modify the Kubernetes manifest for the service that contained the broken code.

### Deploy Fix

Return to **Cloud Shell** and in the **Source Code Editor** open the **currencyservice.yaml** file in the **kubernetes\_manifests** folder.

Replace the image tag `rel013019` with `rel013019fix` so the `image` should look like this:

```
containers:
- name: server
  image: gcr.io/accl-19-dev/frontend:rel013019fix
  imagePullPolicy: Always
```

**Close** the file to save it and **return** to the Cloud Shell prompt.

Redeploy the image with the fix in it:

`skaffold run`

## Validate Fix

Now that you've rolled back the breaking change, verify that the application is back to a healthy state.

As before, start by verifying that your incident is resolved. Go to the **Alerting** in the Monitoring UI and verify that the error rate incident is resolved.

Next, return to **Error Reporting**. Open the error previously observed and verify that it is no longer occurring (the timeline should show no further occurrences since the last deployment).



Congratulations - your monitoring was able to correctly identify a change that caused user experience (as measured by application errors) to degrade, you were able to identify the root cause and you've rolled back the breaking change!

Move on to the next section to learn about how you can optimize resource utilization using Cloud Monitoring.

## Application optimization with Cloud Monitoring APM

In this exercise use Cloud Monitoring Application Performance Management tools (APM) to identify opportunities for improvement that will help your application run faster and use less compute resources.

In this scenario, the Director of Cloud Operations is disappointed with the recent rise in compute costs. Specifically, the **currencyservice** service is using more CPU than expected based on the usage of the system.

Your team has been tasked with finding optimization opportunities. APM tools will be used to analyze the service and ensure your team's efforts are focused on the right areas of the application.

From the Console, open **Profiler** from the left hand menu.

## OPERATIONS

-  **Monitoring**
-  **Debugger**
-  **Trace**
-  **Logging**
-  **Error Reporting**
-  **Profiler**

Change the Timespan in the upper right to 30 minutes. If there is no data, wait a minute or 2 for the data to populate.

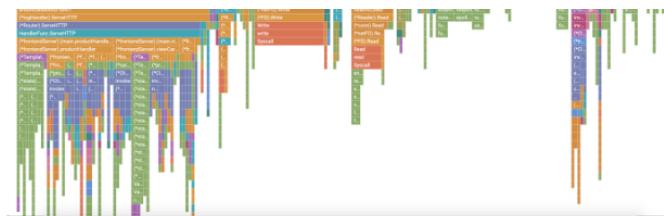
**NOTE:** Profiler takes a random sample of calls to build an aggregate call stack. If you don't see the data you expect, it's because not enough time has elapsed during this lab and you completed it faster than expected. Feel free to use the screenshots below during the exercise.

In the filter options select the **frontend** service, the **CPU time** Profile type.

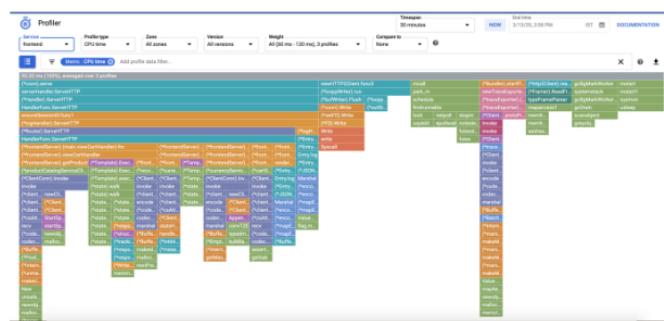


Profiler takes random sample profiles of the system and combines the data to show you what functions are using the most resources. The flame graph below shows the function calls grouped by their use of the resource (in this case CPU) where the X-axis is the amount of CPU and the Y-axis shows parent child relationships.





In this case the majority of the CPU is used by the ServeHTTP call on the left hand side. Click on this call to drill into the cause.



The expanded view shows almost half of this is caused by `viewCartHandler`, which in turn is mostly caused by `getRecommendations`.

The opportunity here is in the `getRecommendations` and in turn `getProduct`.

Thinking back to your earlier exercise, remember that the recommendation service and `getProduct` were being called often in a loop due to an error in retry logic. The resolution for that issue will likely decrease compute cost by as much as 20%.

## 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 will 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 the following:

- 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, please 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.