

Automated Canary Analysis Workshop

Spinnaker Summit - 11/16/2019



Matt Duftler

Staff Software Engineer, Google



Chris Sanden

Senior Data Scientist, Netflix



Andrew Phillips

Product Manager, Google
(Assistant duct-taper)



Agenda

Canary Release Overview

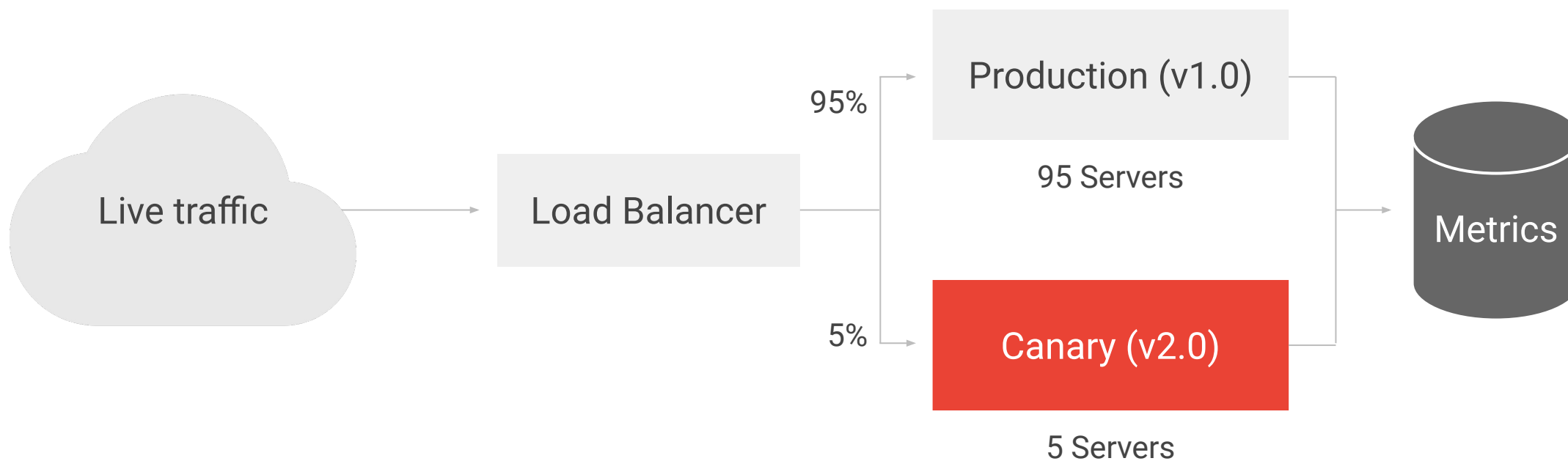
Spinnaker/Kayenta Overview

Provision Spinnaker/Kayenta/Prometheus & Sample Artifacts

Good/Bad Indicators Of Safety

Exercises

Canary Release Overview



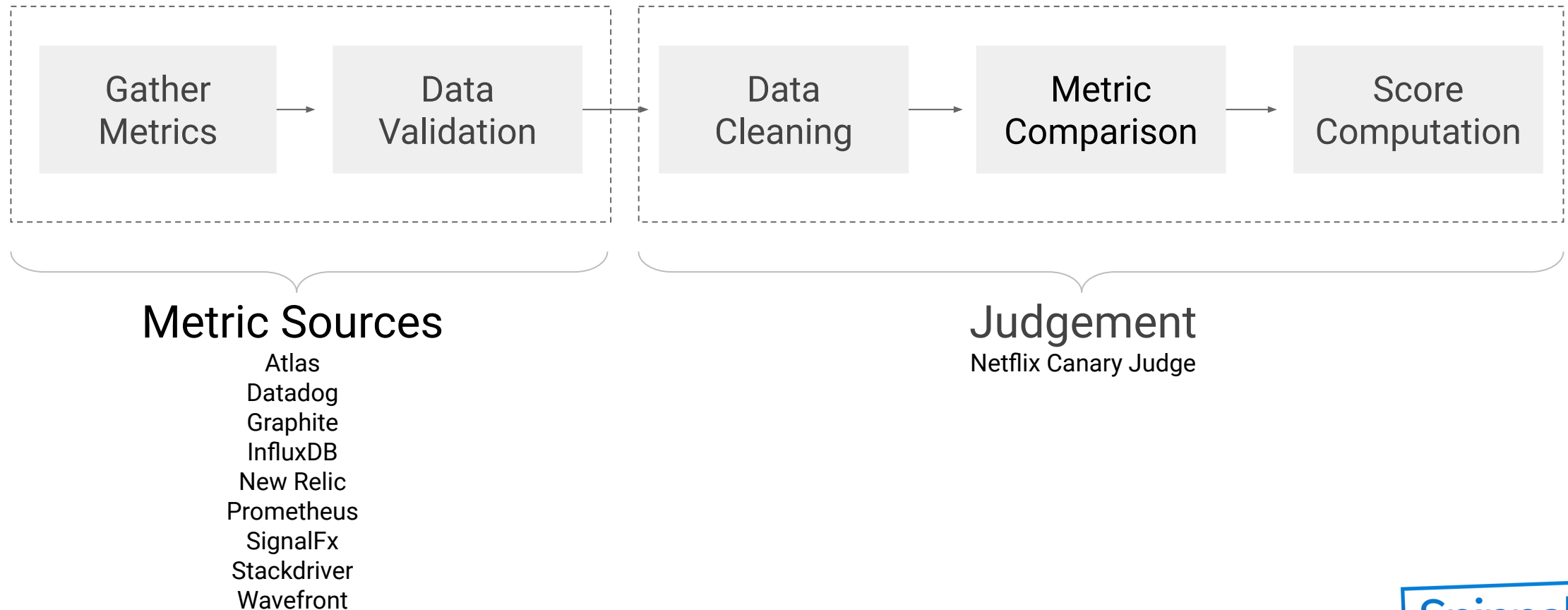
Canary Release Overview



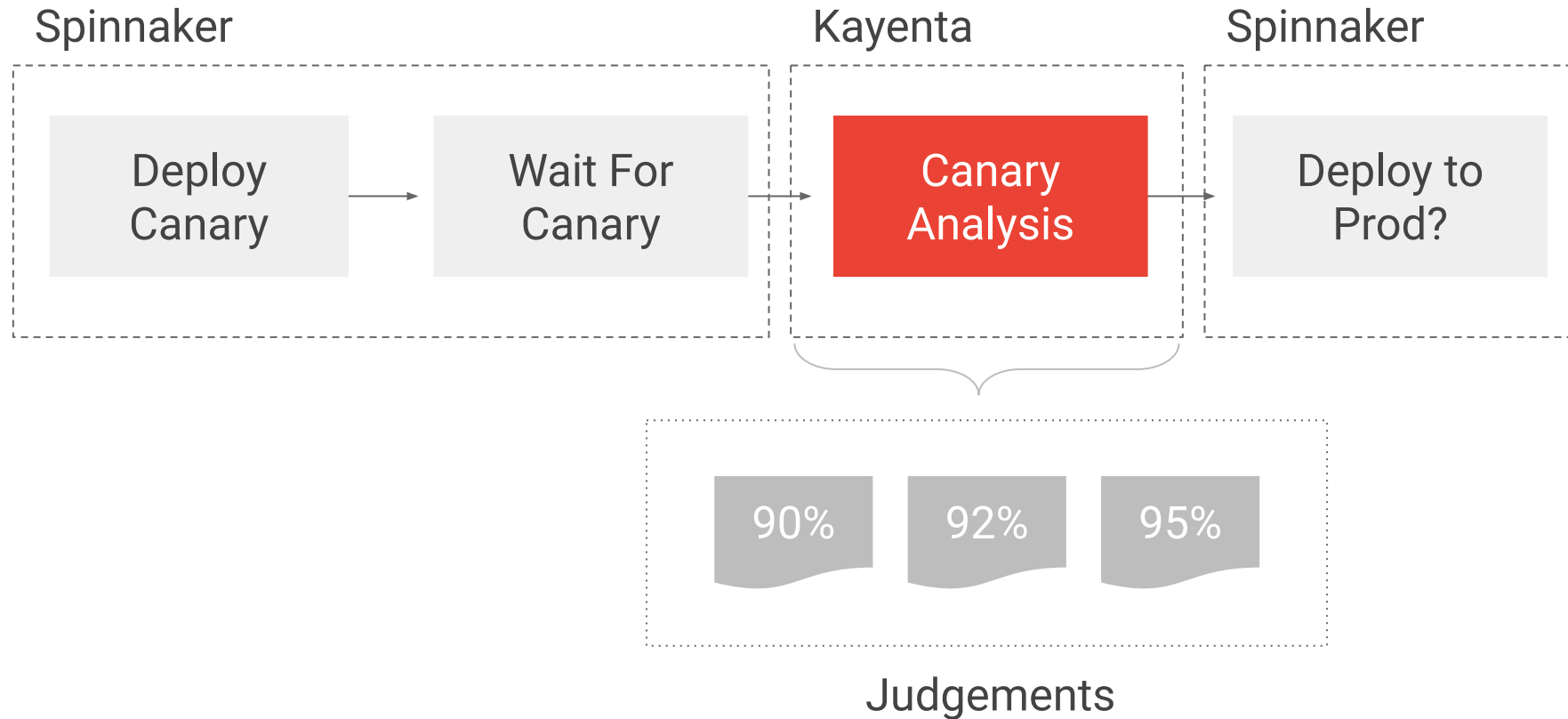
Canary Release Overview



Spinnaker/Kayenta Overview



Spinnaker/Kayenta Overview



Provision Spinnaker/Kayenta

1. Grab a temporary account id/pw.
2. Open an incognito window in Chrome.
3. Navigate to:
<https://console.cloud.google.com/marketplace/details/google-cloud-platform/spinnaker>
4. Click on "Go to Spinnaker for Google Cloud Platform" and follow the instructions.

- Provisioning & configuration should take around 20 minutes to complete. -

Once the `setup.sh` script completes, move to the next slide...

Install Prometheus/Grafana & Provision Sample Artifacts

1. ``exit`` out of the Cloud Shell window.

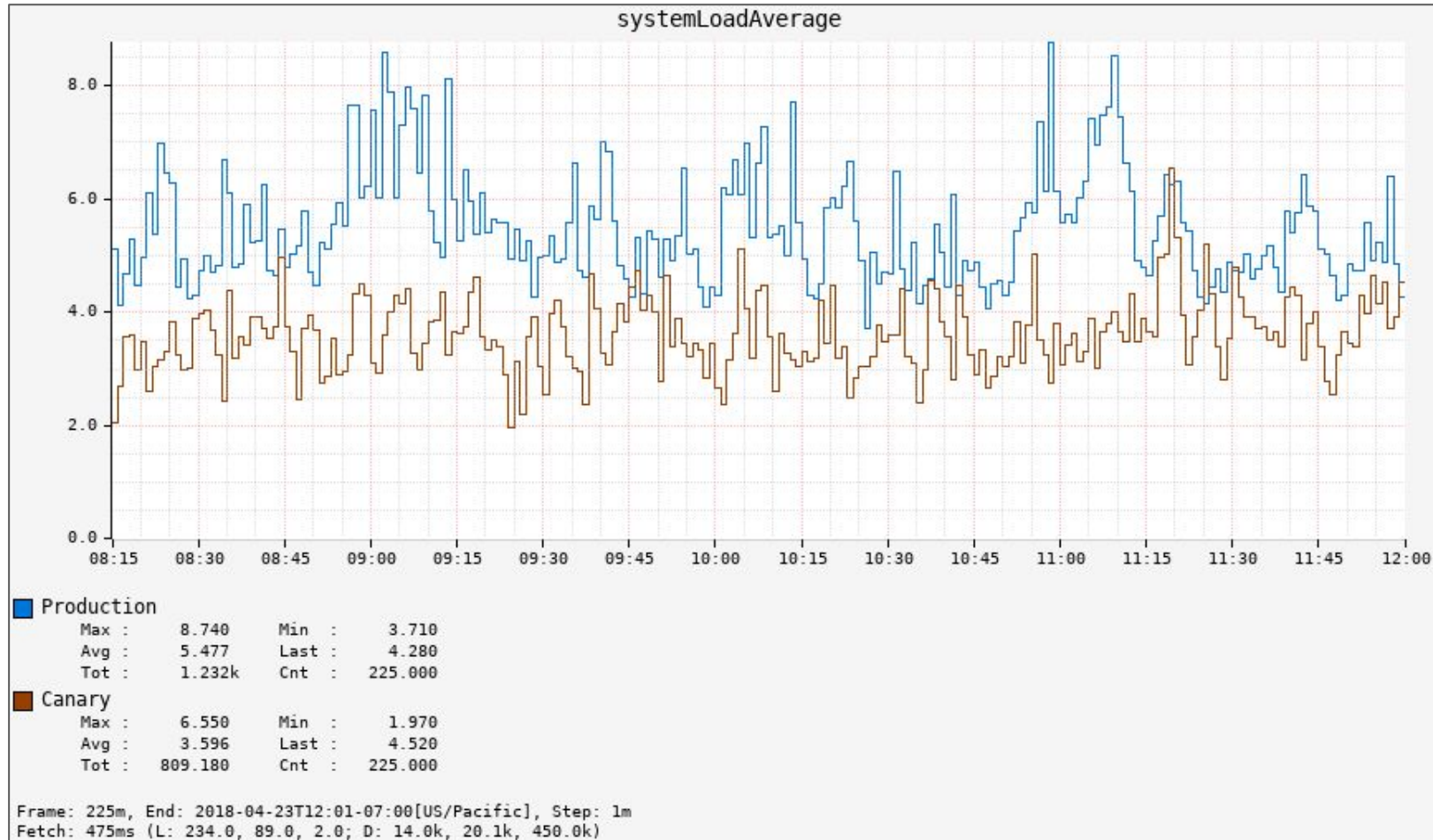
2. Launch the Canary Workshop tutorial:

https://console.cloud.google.com/cloudshell/editor?cloudshell_git_repo=github.com/duftler/canary-workshop.git&cloudshell_tutorial=tutorial.md&cloudshell_print=instructions.txt

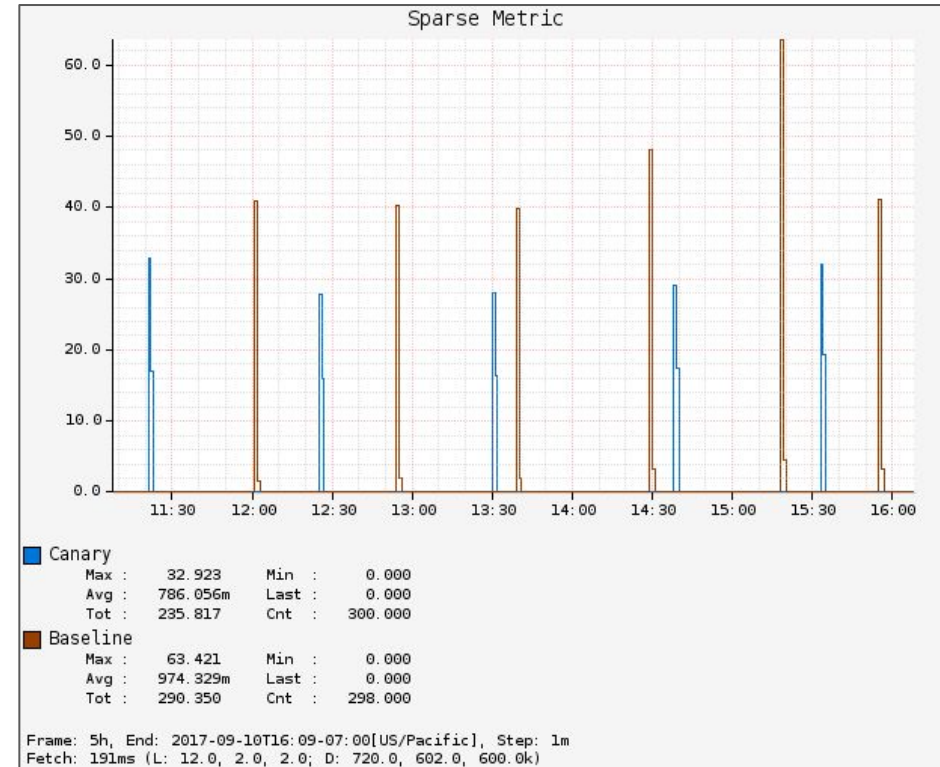
3. Follow the instructions.

- Provisioning & configuration should take around 5 minutes to complete. -

Good/Bad Indicators of Safety



Good/Bad Indicators of Safety



Install Prometheus/Grafana & Provision Sample Artifacts (continued)

Should see output like the following upon completion:

Access Prometheus here: `http://35.230.84.233:31269`

Access Grafana here: `http://35.230.84.233:30866`

If localhost:8080 is unreachable (at any point), re-run:

`~/spinnaker-for-gcp/scripts/manage/connect_unsecured.sh`



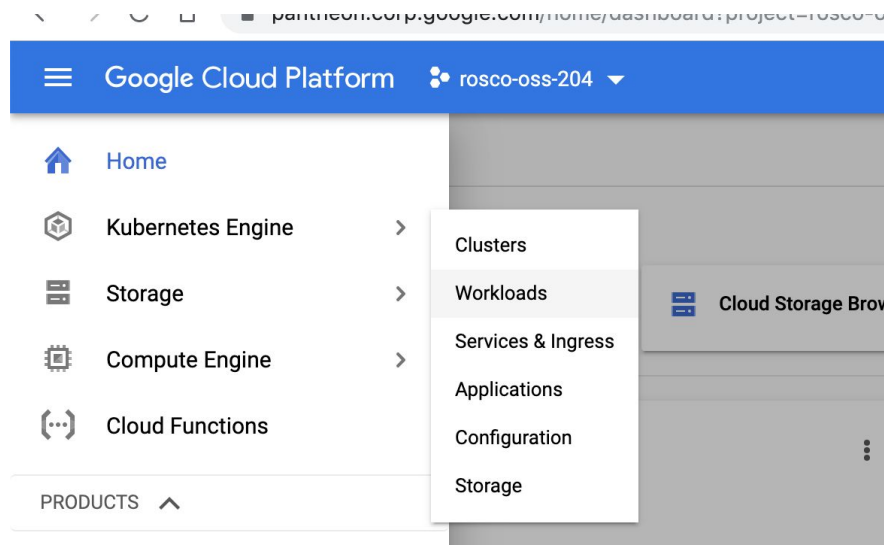
GCP Resources

Locate Relevant GCP Resources

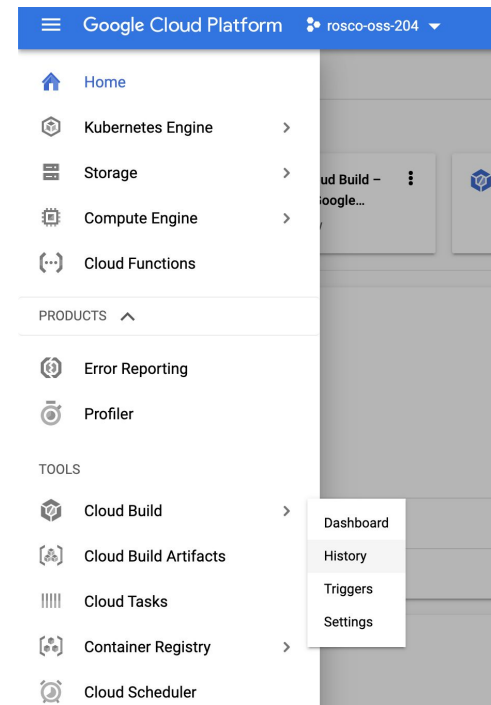
Navigate to: <https://console.cloud.google.com/>

Resources that may be helpful during the workshop:

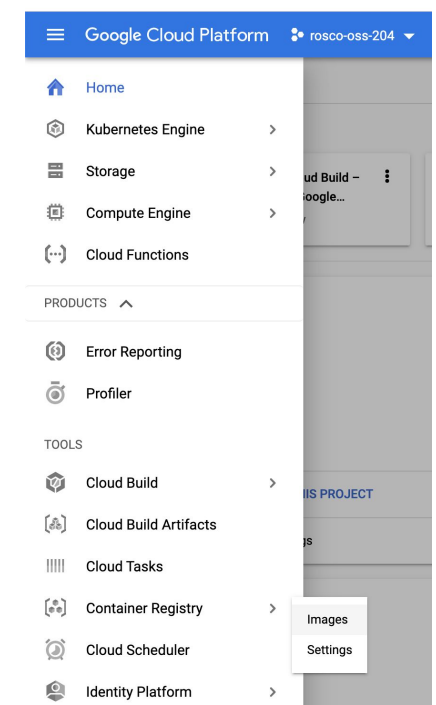
GKE->Workloads



Cloud Build->History



GCR->Images

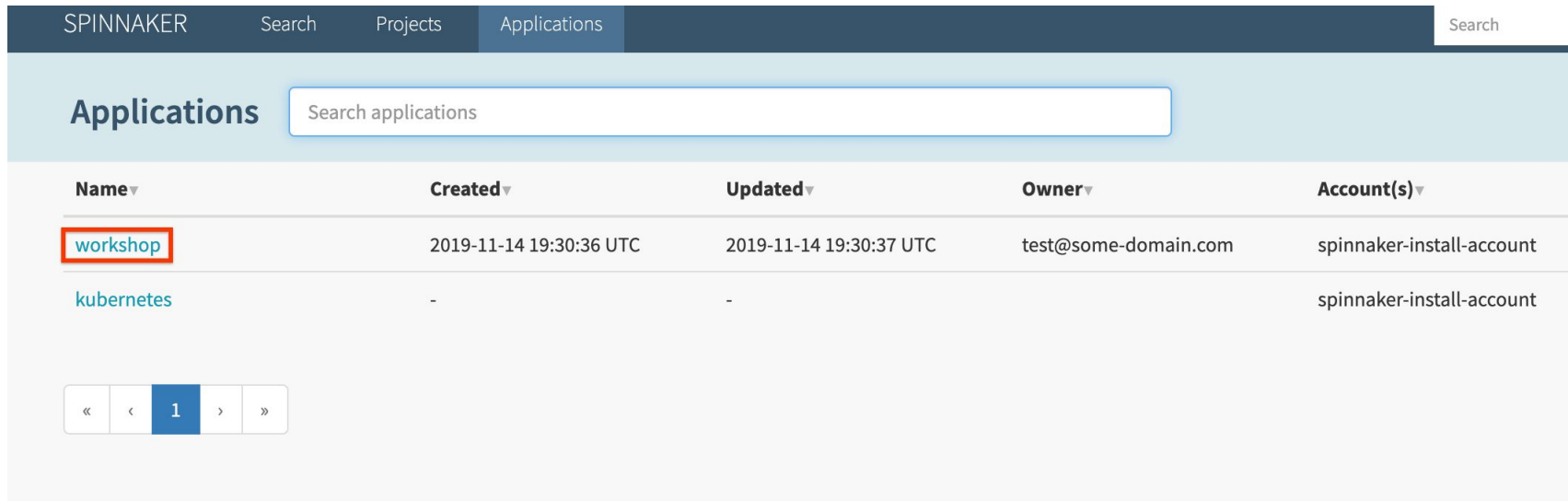


Exercises

Locate workshop Application

Follow the instructions in the console to connect to the Spinnaker UI.

Select the 'workshop' application on the Applications tab:

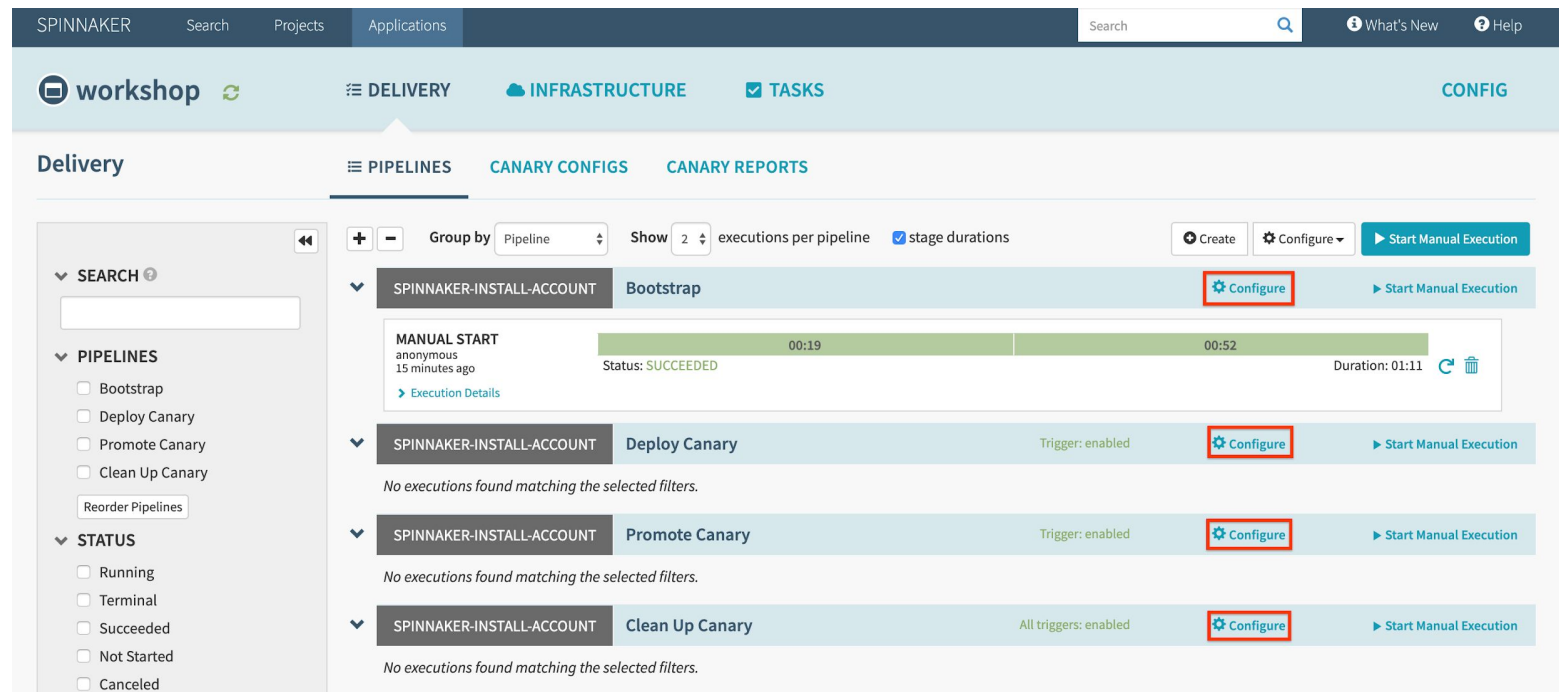


The screenshot shows the Spinnaker UI with the 'Applications' tab selected. The table lists two applications: 'workshop' and 'kubernetes'. The 'workshop' application is highlighted with a red box. The table has columns for Name, Created, Updated, Owner, and Account(s).

Name▼	Created▼	Updated▼	Owner▼	Account(s)▼
workshop	2019-11-14 19:30:36 UTC	2019-11-14 19:30:37 UTC	test@some-domain.com	spinnaker-install-account
kubernetes	-	-		spinnaker-install-account

Explore Pipelines

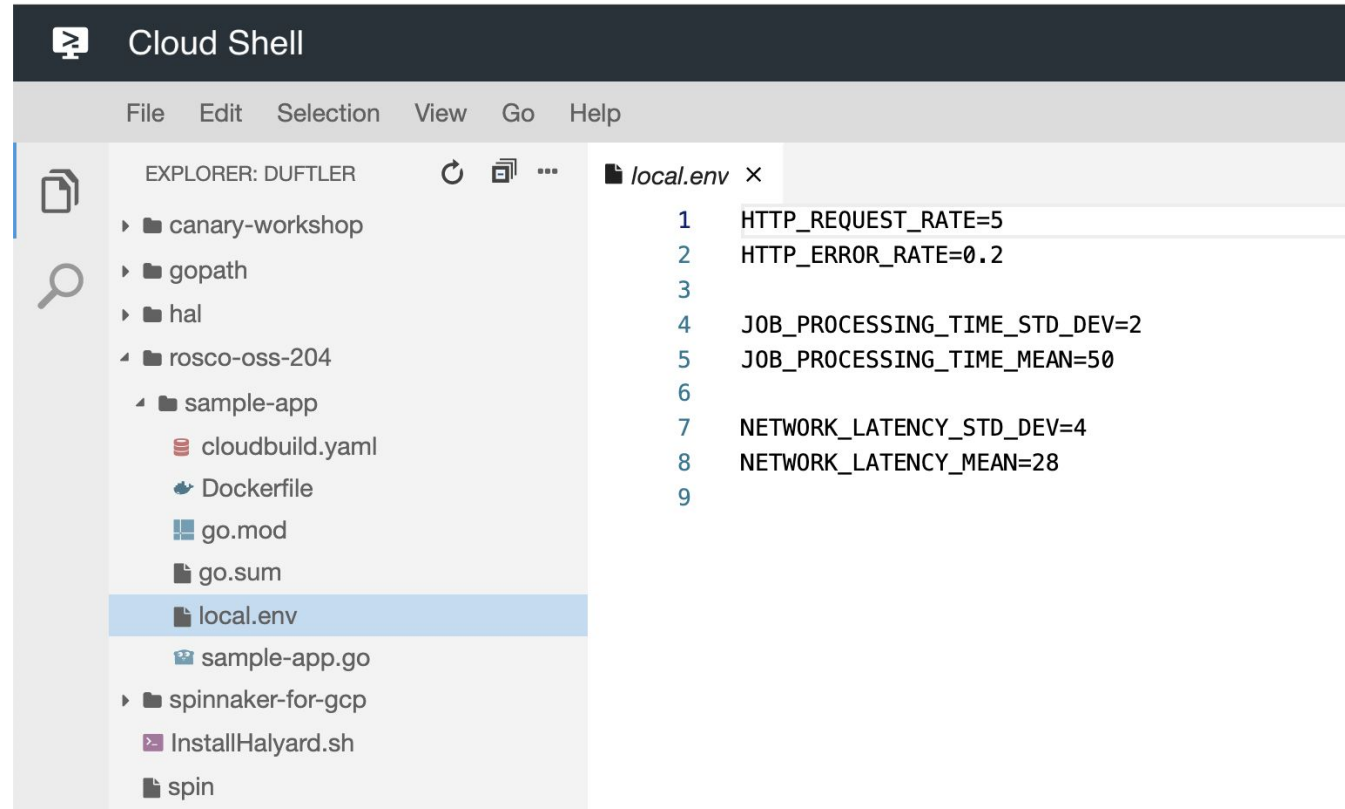
Navigate to the Pipelines tab and examine the configuration of each pre-populated pipeline:



The screenshot displays the Spinnaker web interface, specifically the Pipelines tab under the Delivery section. The interface includes a top navigation bar with 'SPINNAKER', 'Search', 'Projects', 'Applications', and a search input. Below this is a secondary navigation bar with 'workshop', 'DELIVERY', 'INFRASTRUCTURE', 'TASKS', and 'CONFIG'. The main content area is titled 'Delivery' and contains sub-tabs for 'PIPELINES', 'CANARY CONFIGS', and 'CANARY REPORTS'. The 'PIPELINES' tab is active, showing a list of pipelines. On the left, there is a sidebar with a search bar and filters for 'PIPELINES' (Bootstrap, Deploy Canary, Promote Canary, Clean Up Canary) and 'STATUS' (Running, Terminal, Succeeded, Not Started, Canceled). The pipeline list includes columns for 'Group by', 'Show', 'stage durations', and 'Create'. The pipelines listed are 'SPINNAKER-INSTALL-ACCOUNT Bootstrap', 'SPINNAKER-INSTALL-ACCOUNT Deploy Canary', 'SPINNAKER-INSTALL-ACCOUNT Promote Canary', and 'SPINNAKER-INSTALL-ACCOUNT Clean Up Canary'. Each pipeline has a 'Configure' button highlighted with a red box and a 'Start Manual Execution' button. The 'Bootstrap' pipeline shows a successful execution with a duration of 01:11. The other three pipelines show 'No executions found matching the selected filters.'

Locate Sample App and Sample Metric Parameters

The `sample-app` repo has been cloned into your home directory:



The screenshot shows a Cloud Shell interface. On the left is a file explorer titled 'EXPLORER: DUFTLER' showing a directory tree. The 'sample-app' directory is expanded, showing files like 'cloudbuild.yaml', 'Dockerfile', 'go.mod', 'go.sum', 'local.env', 'sample-app.go', and 'spin'. The 'local.env' file is selected. On the right is a code editor showing the contents of 'local.env' with line numbers 1 through 9. The environment variables are listed as follows:

```
1 HTTP_REQUEST_RATE=5
2 HTTP_ERROR_RATE=0.2
3
4 JOB_PROCESSING_TIME_STD_DEV=2
5 JOB_PROCESSING_TIME_MEAN=50
6
7 NETWORK_LATENCY_STD_DEV=4
8 NETWORK_LATENCY_MEAN=28
9
```

Manually Canary a Change

Adjust one of the sample metrics (e.g. change `JOB_PROCESSING_TIME_MEAN` to 55) and push your changes:



The screenshot shows a Cloud Shell interface. On the left, the 'EXPLORER: DUFTLER' sidebar displays a file tree with folders like 'canary-workshop', 'gopath', 'hal', 'rosco-oss-204', and 'sample-app'. The 'sample-app' folder is expanded, showing files like 'cloudbuild.yaml', 'Dockerfile', 'go.mod', 'go.sum', 'local.env', 'sample-app.go', 'spinnaker-for-gcp', 'InstallHalyard.sh', 'spin', and 'todo.txt'. The 'local.env' file is selected and open in the main editor. The file content is as follows:

```
1 HTTP_REQUEST_RATE=5
2 HTTP_ERROR_RATE=0.2
3
4 JOB_PROCESSING_TIME_STD_DEV=2
5 JOB_PROCESSING_TIME_MEAN=55
6
7 NETWORK_LATENCY_STD_DEV=4
8 NETWORK_LATENCY_MEAN=28
9
```

At the bottom, the terminal shows the command `duftler@cloudshell: ~/rosco-oss-204/sample-app (rosco-oss-204) $ git commit -am 'change metric' && git push`. The path `~/rosco-oss-204/sample-app` is highlighted with a red box.

Follow Progression of Change

- Change is pushed to Cloud Source Repo
- Cloud Build trigger builds new image
- Image is published to Container Registry
- Deploy Canary pipeline is triggered
- Baseline and Canary deployments are provisioned (locate image digests)
- Metrics are collected by Prometheus
- Grafana dashboard refreshes automatically
- Wait several minutes for dashboard to reflect change

Do Not Manually Approve Change Yet

Promote Change

Promote the change into production by clicking "Continue" on the "Manually Validate Canary Results" stage of the "Deploy Canary" pipeline. Successful completion of that pipeline will trigger the "Promote Canary" pipeline.

Successful completion of the "Promote Canary" pipeline will trigger the "Clean Up Canary" pipeline.

At this point, we will have covered 'manual' canary releases of binary changes, and we will next adapt the existing release workflow to include automated canary analysis.

Create Canary Configuration

Create a new Canary Config named "workshop-config":

The screenshot shows the Spinnaker web interface. The top navigation bar includes 'SPINNAKER', 'Search', 'Projects', 'Applications', and another 'Search' box. Below this, the 'workshop' application is selected, with tabs for 'DELIVERY', 'INFRASTRUCTURE', and 'TASKS'. The 'DELIVERY' tab is active, showing sub-tabs for 'PIPELINES', 'CANARY CONFIGS', and 'CANARY REPORTS'. The 'CANARY CONFIGS' tab is selected, displaying a list of configurations. A configuration named 'workshop-config' is shown, with a timestamp 'Edited: 2019-11-15 16:50:10 UTC'. Below the list, there is a dashed box with a plus icon and the text 'Add configuration'. The main form area is titled 'workshop-config' and 'Edited: 2019-11-15 16:50:09 UTC'. It has a section 'NAME AND DESCRIPTION' with two fields: 'Configuration Name' (containing 'workshop-config') and 'Description' (empty).

SPINNAKER Search Projects Applications Search

workshop DELIVERY INFRASTRUCTURE TASKS

Delivery PIPELINES CANARY CONFIGS CANARY REPORTS

workshop-config Edited: 2019-11-15 16:50:10 UTC

+ Add configuration

workshop-config Edited: 2019-11-15 16:50:09 UTC

NAME AND DESCRIPTION

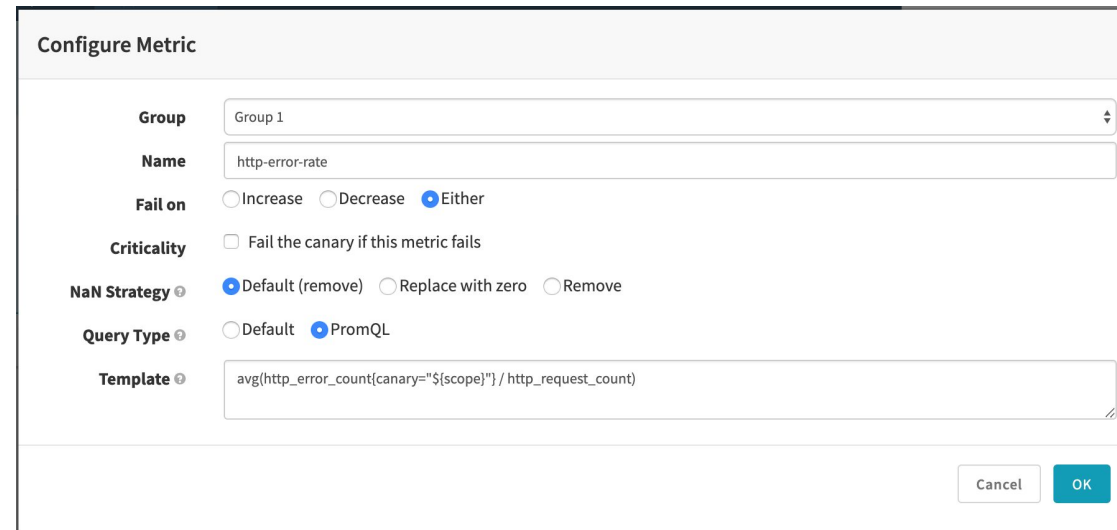
Configuration Name workshop-config

Description

Create Canary Configuration

Create a new Metric Config named "http-error-rate", with a PromQL template containing:

```
avg(http_error_count{canary="${scope}"} / http_request_count)
```



The screenshot shows the 'Configure Metric' dialog box with the following configuration:

- Group:** Group 1
- Name:** http-error-rate
- Fail on:** ☐ Increase ☐ Decrease ☒ Either
- Criticality:** ☐ Fail the canary if this metric fails
- NaN Strategy:** ☒ Default (remove) ☐ Replace with zero ☐ Remove
- Query Type:** ☐ Default ☒ PromQL
- Template:** avg(http_error_count{canary="\${scope}"} / http_request_count)

Buttons: Cancel, OK

Set the scoring weight of the group to 100 and save your Canary Config.

Create Automated Canary Analysis Stage

Edit the "Deploy Canary" pipeline and replace the "Manually Validate Canary Results" stage with a new "Canary Analysis" stage.

Make sure to select "Real Time (Manual)", and a reasonable lifetime (e.g. 5 minutes, for the purposes of this demonstration). 5 seconds is a good step size to start with.

Pay special attention to the values specified for Baseline & Canary, as they will be bound to `${scope}` when expanding the template.

Canary Analysis Configuration

Analysis Config

Analysis Type ?

☒ Real Time (Manual)

☐ Retrospective

Config Name

workshop-config

Lifetime ?

0

hours

5

minutes

Delay ?

minutes before starting analysis

Interval ?

5

minutes

Step

5

seconds

Baseline Offset ?

minutes

Lookback Type ?

Growing

Baseline + Canary Pair

Baseline ?

baseline

Baseline Location ?

location

Canary ?

canary

Canary Location ?

location

Metric Scope

Extended Params ?

Key

Value

Add Field

Scoring Thresholds

Marginal ?

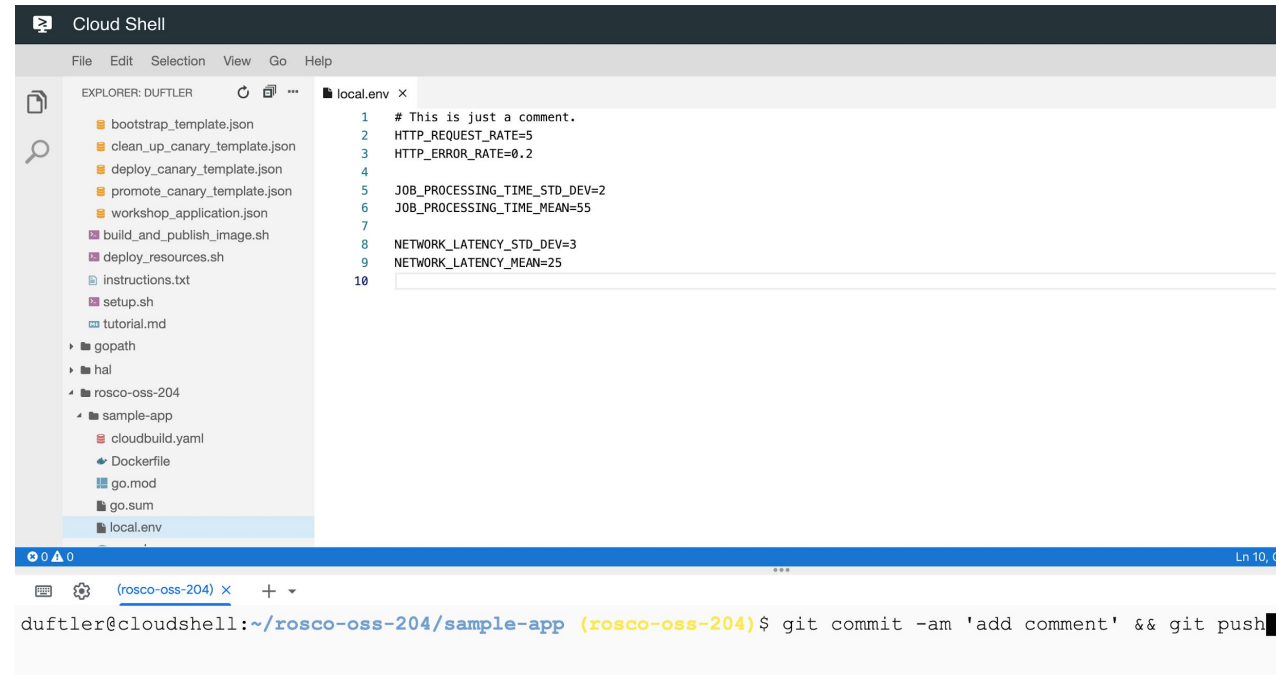
50

Pass ?

100

Canary a Change

Push an inconsequential change (e.g. add a comment to the local.env file):



The screenshot displays a Cloud Shell interface. On the left, the 'EXPLORER: DUFTLER' panel shows a file tree with various JSON templates, shell scripts, and a 'local.env' file at the bottom. The main editor area on the right shows the content of 'local.env' with several environment variables and a comment. The terminal at the bottom shows the user 'duftler@cloudshell' in the directory '~/rosco-oss-204/sample-app' executing the command 'git commit -am 'add comment' && git push'.

```
1 # This is just a comment.  
2 HTTP_REQUEST_RATE=5  
3 HTTP_ERROR_RATE=0.2  
4  
5 JOB_PROCESSING_TIME_STD_DEV=2  
6 JOB_PROCESSING_TIME_MEAN=55  
7  
8 NETWORK_LATENCY_STD_DEV=3  
9 NETWORK_LATENCY_MEAN=25  
10
```

```
duftler@cloudshell:~/rosco-oss-204/sample-app (rosco-oss-204)$ git commit -am 'add comment' && git push
```

Explore Canary Report and Underlying Queries

Click on the Canary Report link in the Pipeline Execution Details view:

SPINNAKER-INSTALL-ACCOUNT **Deploy Canary** Trigger: enabled [Configure](#) [Start Manual Execution](#)

PUBSUB
about 9 hours ago
Status: **SUCCEEDED**
Duration: 05:20
[View All Artifacts \(1\)](#)
[Execution Details](#)

00:16 00:00 00:16 05:04

Create Canary Deployment
Find Baseline Image
Create Baseline Deployment
Canary Analysis (100)

STAGE DETAILS: CANARY ANALYSIS
Duration: 05:04

Step	Started	Duration	Status
Canary Analysis	2019-11-16 06:58:01 UTC	05:04	SUCCEEDED

✓ **CANARY ANALYSIS**

Canary Summary Canary Config Task Status

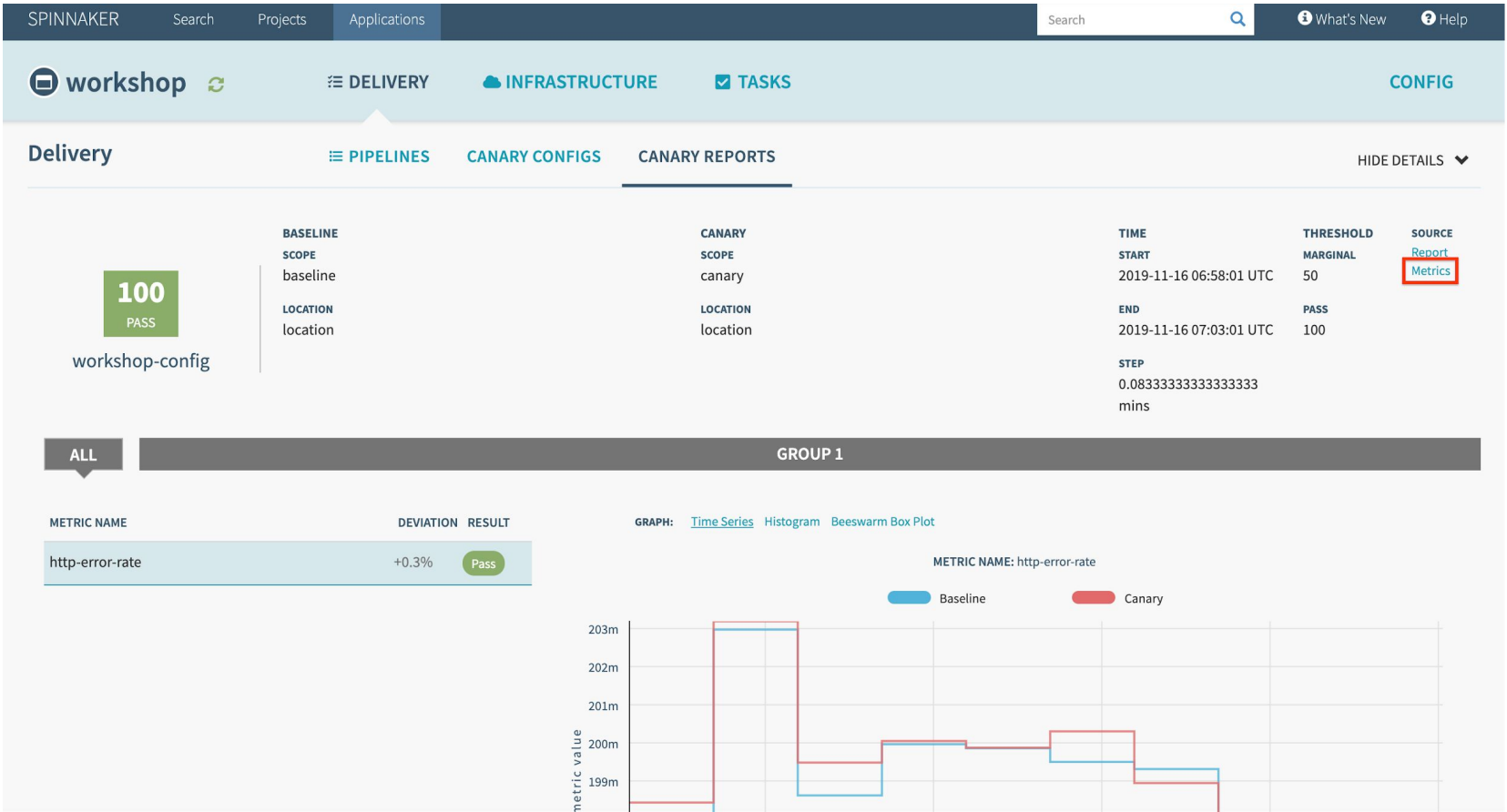
100

CANARY RESULT	DURATION	LAST UPDATED
100	PT5M	2019-11-16 07:03:04 UTC Canary Report ⓘ

Source | Permalink

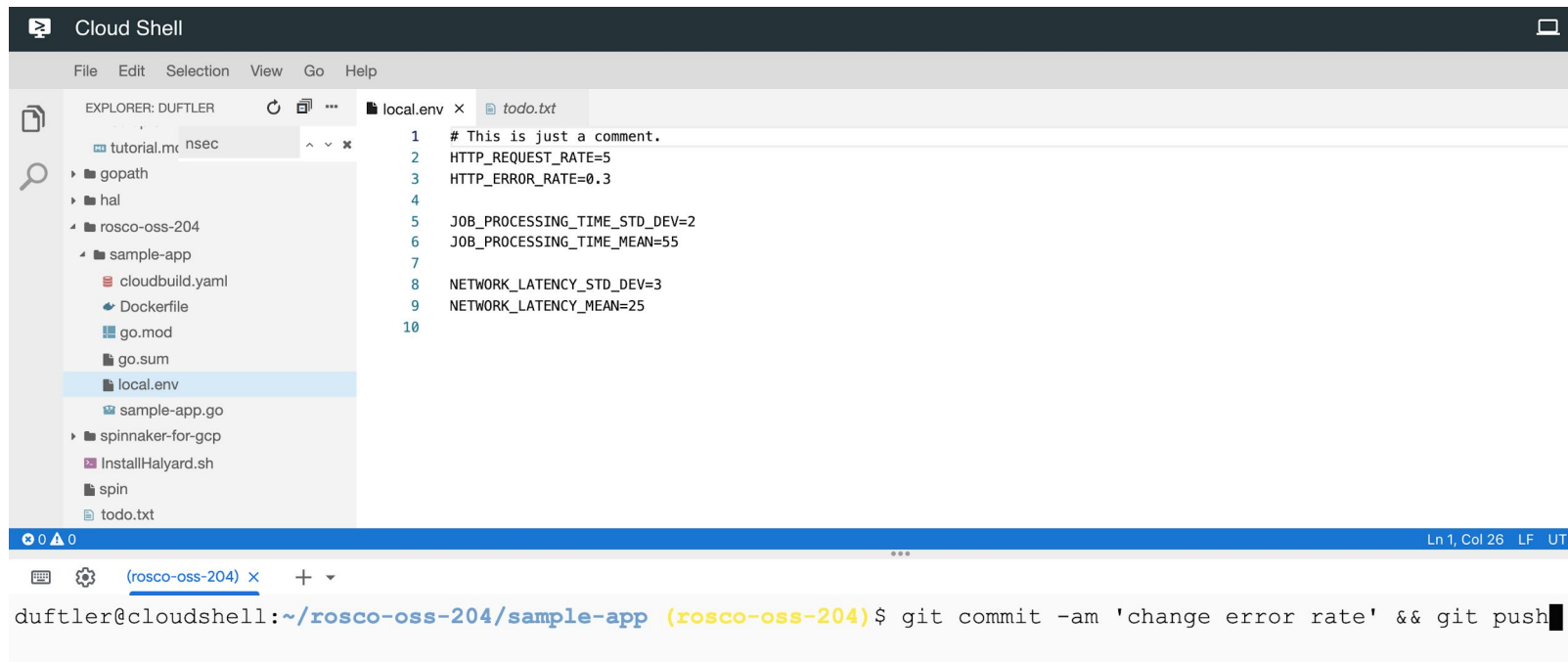
Explore Canary Report and Underlying Queries

For troubleshooting, you can find the expanded queries (and their actual results) via the Metrics link:



Canary a Change

Now push a change that is likely to fail the canary analysis (e.g. change `HTTP_ERROR_RATE` to `0.3`):



```
Cloud Shell
```

```
File Edit Selection View Go Help
```

```
EXPLORER: DUFTLER
```

```
local.env x todo.txt
```

```
1 # This is just a comment.
```

```
2 HTTP_REQUEST_RATE=5
```

```
3 HTTP_ERROR_RATE=0.3
```

```
4
```

```
5 JOB_PROCESSING_TIME_STD_DEV=2
```

```
6 JOB_PROCESSING_TIME_MEAN=55
```

```
7
```

```
8 NETWORK_LATENCY_STD_DEV=3
```

```
9 NETWORK_LATENCY_MEAN=25
```

```
10
```

```
duftler@cloudshell:~/rosco-oss-204/sample-app (rosco-oss-204)$ git commit -am 'change error rate' && git push
```

Configure Retrospective Analysis

Capture the timestamps from the analysis stage (using the copy-to-clipboard buttons) and use them to configure a new pipeline with a "Retrospective" canary analysis stage:

SPINNAKER-INSTALL-ACCOUNT **Deploy Canary** Trigger: enabled [Configure](#) [Start Manual Execution](#)

PUBSUB
10 minutes ago
Status: **TERMINAL**
Duration: 05:31
[View All Artifacts \(1\)](#)
[Execution Details](#)

Create Canary Deployment
Find Baseline Image
Create Baseline Deployment
Canary Analysis (0)

STAGE DETAILS: CANARY ANALYSIS
Duration: 05:03

Step	Started	Duration	Status
Run Canary #1	2019-11-16 16:22:17 UTC	00:03	TERMINAL
Run Canary Intervals	2019-11-16 16:17:17 UTC	05:03	TERMINAL
Canary Analysis	2019-11-16 16:17:17 UTC	05:03	TERMINAL

CANARY ANALYSIS

Canary Summary **Canary Config** **Task Status**

0

CANARY RESULT **DURATION** **LAST UPDATED**

0 PT5M 2019-11-16 16:22:21 UTC

Exception (Run Canary #1)
No reason provided.

Start: 2019-11-16 16:17:17 UTC
End: 2019-11-16 16:22:17 UTC

[Source](#) [Permalink](#)

Configure Retrospective Analysis

This technique can be used to quickly iterate on a canary analysis configuration, with reproducible results, without needing to repeatedly wait for infrastructure to be provisioned and torn down.

Canary Analysis Configuration

Analysis Config

Analysis Type

☐ Real Time (Manual)

☒ Retrospective

Config Name

workshop-config

Start Time

2019-11-16T16:17:17.509Z

End Time

2019-11-16T16:22:17.509Z

Interval

5

minutes

Step

5

seconds

Baseline Offset

minutes

Lookback Type

Growing

Baseline + Canary Pair

Baseline

baseline

Baseline Location

location

Canary

canary

Canary Location

location

Still Want More?

Additional Things to Explore

- Configure a canary using multiple metrics
- See if you can tune one of the metrics such that a human would have a hard time determining from the dashboard whether it is a safe push or not, but the statistical test still yields good results