

Automated Canary Analysis Workshop

Spinnaker Summit - 10/8/2018

Agenda

Canary Release Overview

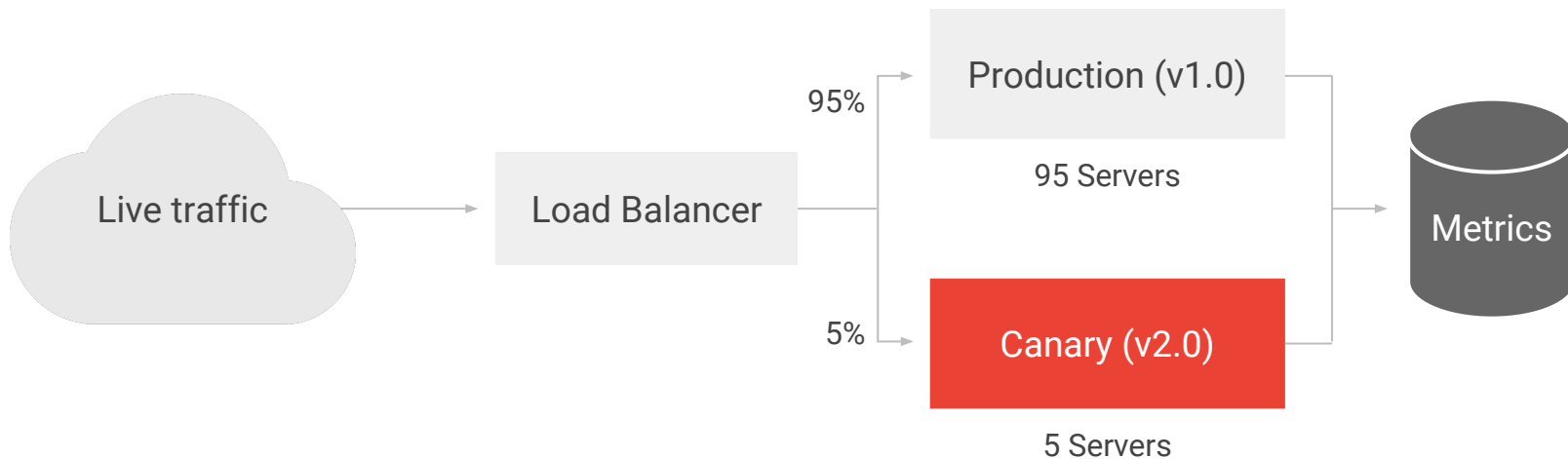
Spinnaker/Kayenta Overview

Provision Spinnaker/Kayenta & Sample Artifacts

Good/Bad Indicators Of Safety

Exercises

Canary Release Overview



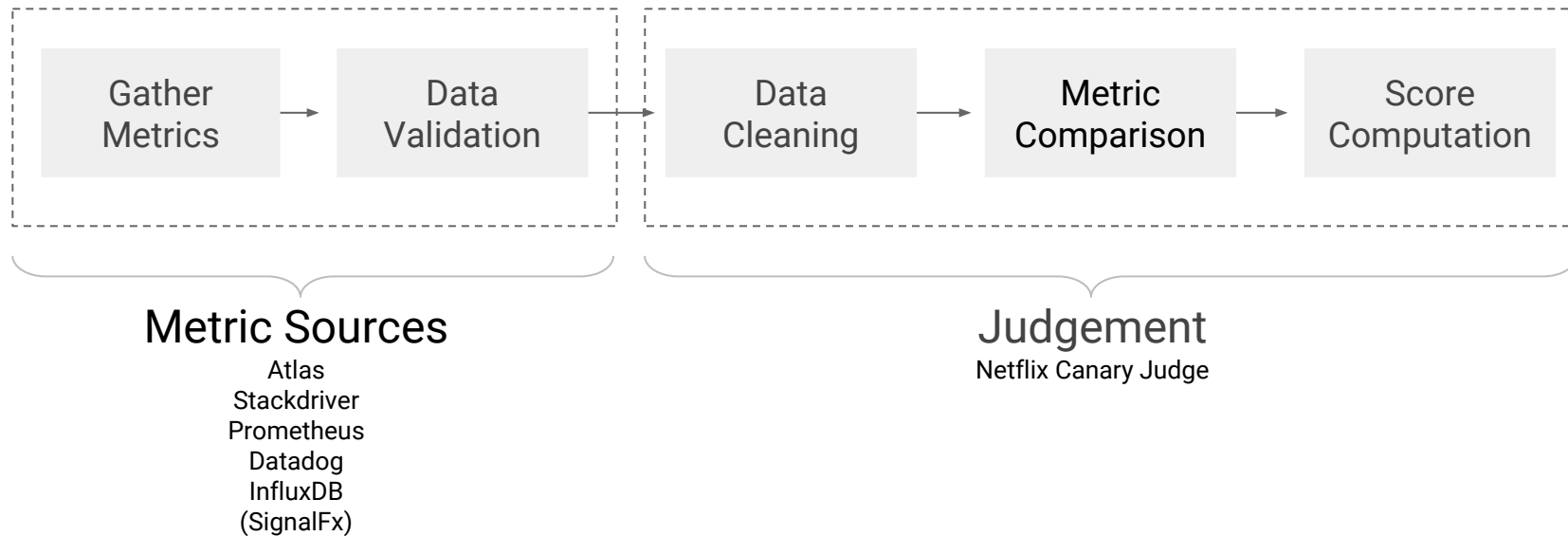
Canary Release Overview



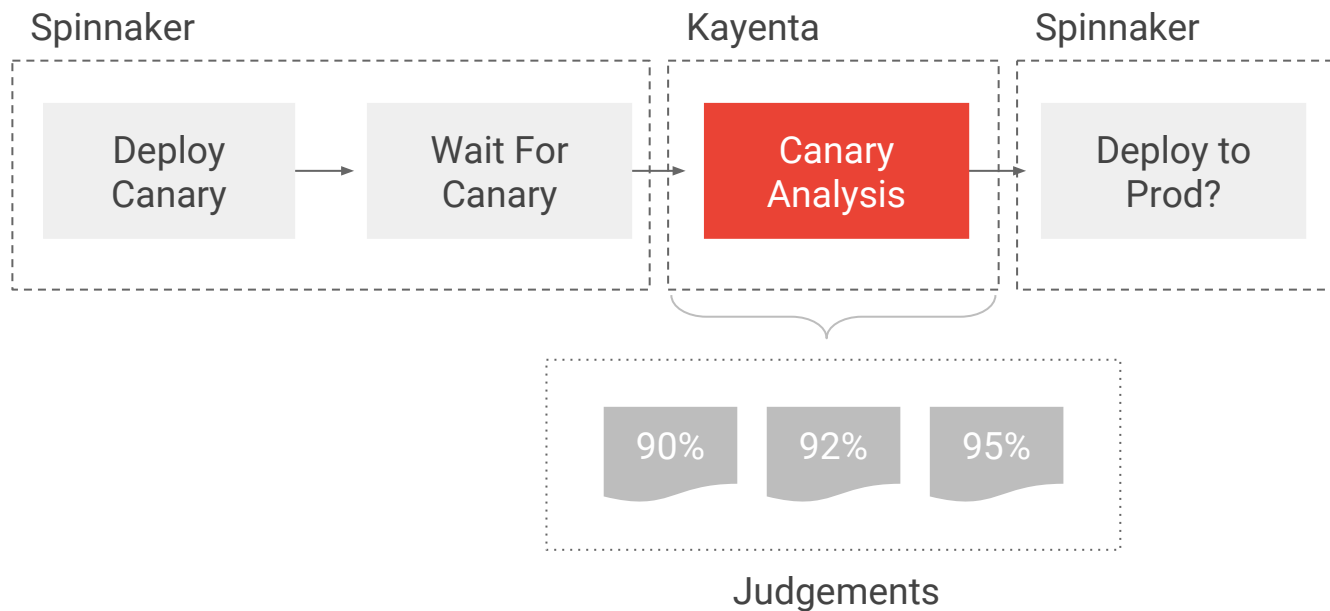
Canary Release Overview



Spinnaker/Kayenta Overview



Spinnaker/Kayenta Overview



Provision Spinnaker/Kayenta & Sample Artifacts

Grab a temporary account id/pw.

Navigate to:

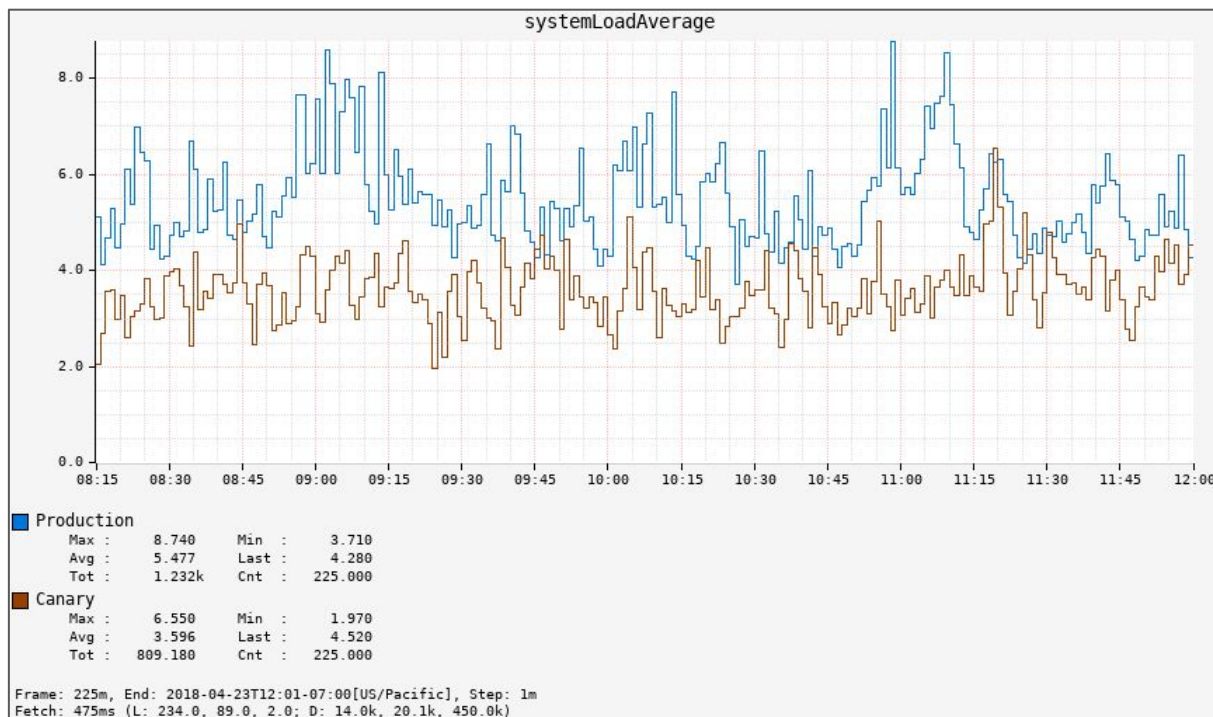
<https://codelabs.developers.google.com/codelabs/cloud-spinnaker-kubernetes-cd/index.html#0>

On the "2. Set up Spinnaker" page, use these commands instead of the existing commands under "Deploy Spinnaker":

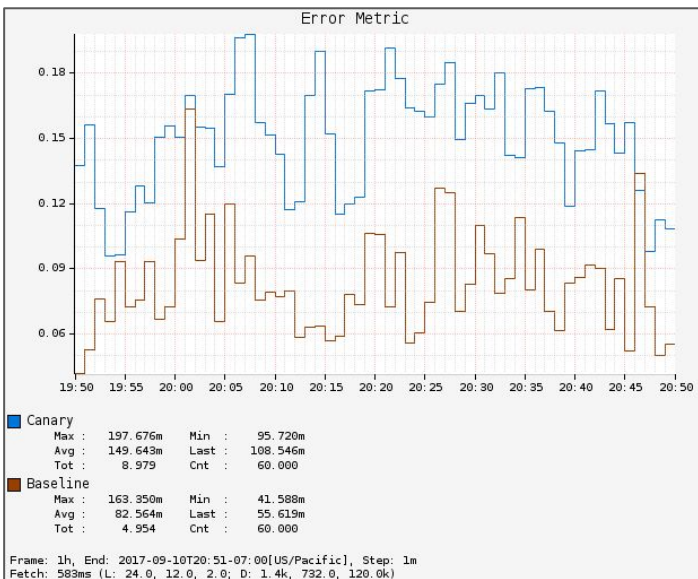
```
gsutil cp gs://gke-spinnaker-codelab/base/install.tar .  
tar xvf install.tar  
./setup.sh kayenta-workshop
```

- Provisioning & configuration should take 10-15 minutes to complete. -

Good/Bad Indicators of Safety



Good/Bad Indicators of Safety



Provision Spinnaker/Kayenta & Sample Artifacts (continued)

If you don't see this output upon completion:

```
Application save succeeded
```

```
Pipeline save succeeded
```

```
Pipeline save succeeded
```

```
Pipeline save succeeded
```

then re-run the 4 `spin` commands at the end of `overrides/publish_samples.sh`

If `localhost:8080` is unreachable (at any point), re-run: `./connect.sh`

Exercises

Follow the public codelab through "6. Canary a Config Change".

Repeat step 6, but this time modify `backend.yml` to reduce the `memory (requests & limits)` allocation from `128Mi` to `12Mi` and run: `./update-backend.sh`

Also, temporarily stop short of fully promoting the change into production so we can compare the baseline & canary metrics.

Exercises

Navigate to the Stackdriver Metrics Explorer:

<https://app.google.stackdriver.com/metrics-explorer>

(You will likely have to navigate through the initial Stackdriver initialization step. Just take the default options throughout and then navigate back to the original link above.)

Build this query:

Resource type: `k8s_container`

Metric: `kubernetes.io/container/memory/request_utilization`

Filter:

`cluster_name=spin-kayenta-workshop`

`namespace_name=production`

After poking around the dashboard, aggregate the results:

Group By: `label top_level_controller_name`

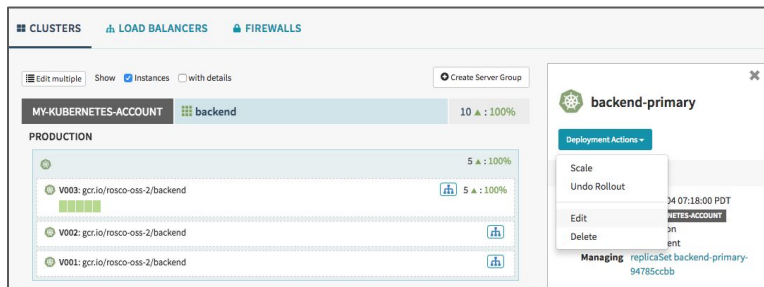
Aggregation: `mean`

Exercises

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

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

Manually edit the production deployment to reset the memory allocation to the higher amount (128Mi). We will want it reset to the original, higher amount so that we can exercise the automated canary stage we are about to add.

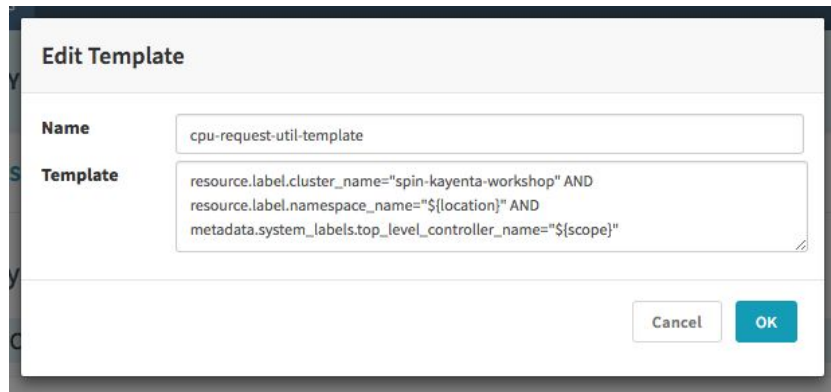


Exercises

Create a new Canary Config named "sample-canary-config".

Configure a new Template named "mem-request-util-template" with this definition:

```
resource.label.cluster_name="spin-kayenta-workshop" AND  
resource.label.namespace_name="${location}" AND  
metadata.system_labels.top_level_controller_name="${scope}"
```



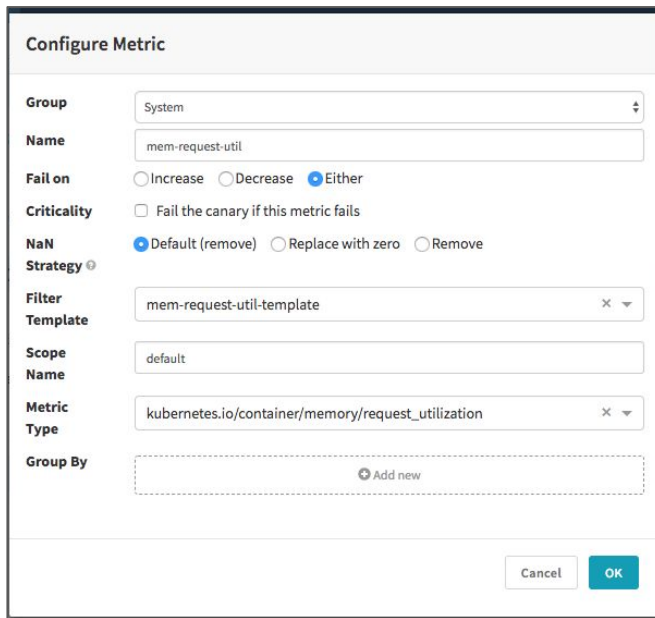
Edit Template

Name

Template

Exercises

Configure a new Metric named "mem-request-util" for Metric Type "kubernetes.io/container/memory/request_utilization" , using the template we just configured:



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

- Group:** System
- Name:** mem-request-util
- Fail on:** ☐ Increase ☐ Decrease ☒ Either
- Criticality:** ☐ Fail the canary if this metric fails
- NaN Strategy:** ☒ Default (remove) ☐ Replace with zero ☐ Remove
- Filter Template:** mem-request-util-template
- Scope Name:** default
- Metric Type:** kubernetes.io/container/memory/request_utilization
- Group By:** Add new

Buttons: Cancel, OK

Exercises

Set the scoring weight of the group to 100 and save your changes:

NAME AND DESCRIPTION

Configuration Name

sample-canary-config

Description

METRICS

ALL

SYSTEM

Add Group

METRIC NAME

mem-request-util

GROUPS

System

Add Metric

FILTER TEMPLATES

TEMPLATE NAME

mem-request-util-template

Add Template

SCORING

Thresholds

Marginal

50

Pass

75

Metric Group Weights

System

100

Exercises

Edit the "Deploy Simple Canary to Production" pipeline and replace the "Wait For Canary Results" and "Manually Validate Canary Results" stages 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).

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

The screenshot displays the 'Analysis Config' form, which is used to configure a canary analysis stage. The form is organized into several sections:

- Analysis Type:** Three radio buttons are present: 'Real Time (Automatic)' (unselected), 'Real Time (Manual)' (selected), and 'Retrospective' (unselected).
- Config Name:** A dropdown menu showing 'sample-canary-config'.
- Lifetime:** Input fields for '0' hours and '5' minutes.
- Delay:** A dropdown menu set to '2' minutes before starting analysis.
- Interval:** A dropdown menu set to 'minutes'.
- Step:** Input fields for '60' seconds.
- Lookback Type:** A dropdown menu set to 'Growing'.
- Baseline & Canary Server Groups:** A section with four input fields: 'Baseline' (backend-primary), 'Baseline Location' (production), 'Canary' (backend-canary), and 'Canary Location' (production).
- Metric Scope:** A dropdown menu set to 'k8s_container'.
- Extended Params:** A table with columns 'Key' and 'Value'. Below the table is a dashed box with an 'Add Field' button.
- Scoring Thresholds:** Two input fields: 'Marginal' (50) and 'Pass' (75). Below these is a horizontal bar chart with three segments: red (0-50%), grey (50-75%), and green (75-100%).
- Advanced Settings:** Three input fields: 'Metrics Account' (my-google-account), 'Storage Account' (my-google-account), and 'Scope Name' (default).

Exercises

Repeat step 6 (`backend.yml` should already have the reduced memory (requests & limits) allocation of 12mi):

```
./update-backend.sh
```

The Canary Analysis stage should fail after about 7 minutes (2 minute warmup + 5 minute lifetime).

Navigate to the canary report:

The screenshot shows the Spinnaker interface for a deployment titled "Deploy Simple Canary to Production". The top bar indicates the trigger is enabled and provides links to "Configure" and "Start Manual Execution".

The main section displays a "PIPELINE" with a progress bar showing a duration of 07:38. The status is "TERMINAL". The pipeline steps are listed as follows:

- go://spin-gcs-bucket-d80dayth...
- go://spin-gcs-bucket-d80dayth...
- go://repo-oss-2/terraform
- go://repo-oss-2/backend

Below the pipeline steps, a "Canary Analysis" stage is highlighted. The "STAGE DETAILS: CANARY ANALYSIS" section shows a duration of 07:04. A table lists the stage details:

Step	Started	Duration	Status
Canary Analysis	2018-10-05 14:18:53 PDT	07:04	TERMINAL

The "CANARY ANALYSIS" section shows a "Canary Summary" with a score of 0. The "Canary Result" section shows a "Canary score is not above the marginal score threshold." message. The "Last Updated" timestamp is 2018-10-05 14:20:06 PDT.

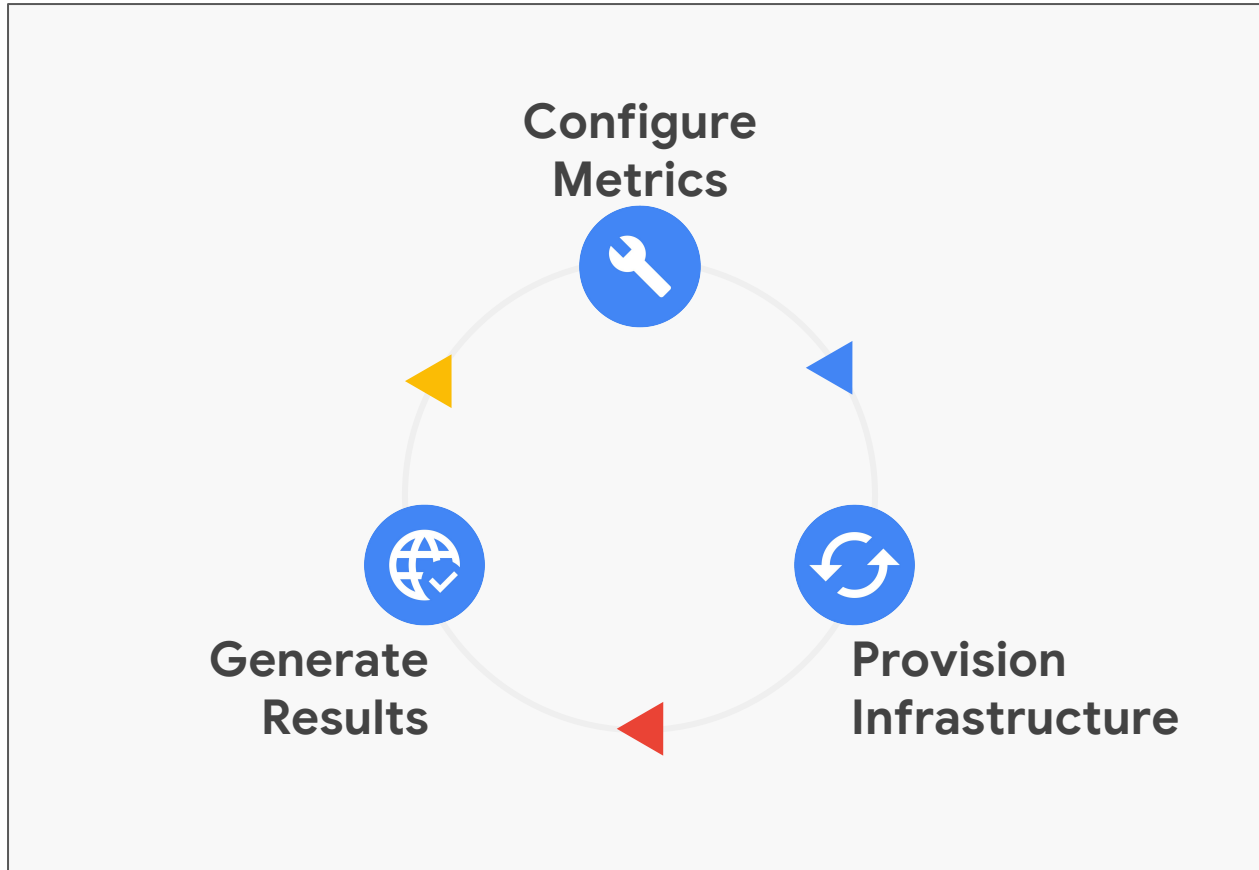
Exercises

Set the memory allocation back to `128Mi` and re-run `./update-backend.sh`.

The Canary Analysis stage should succeed this time.

You may need to extend the warmup and/or lifetime periods to give the pods time to stabilize.

Pro-tip: Use a Retrospective analysis to try experimenting with the canary stage configuration without having to re-provision resources and wait.



Exercises

Want to add another metric?

The backend service publishes a custom metric `custom.googleapis.com/my_app_metric` for resource type `k8s_pod`. Its value can be explicitly controlled via the `MY_APP_METRIC_VALUE` env var in `backend.yml`.

Try adding the custom application metric to your existing canary config.

Hint: Probably makes sense to follow a similar path of first querying it via the metrics explorer.

Another hint: You should be able to reuse the existing template for the new metric.

Note: The `resourceType` configuration has moved from the canary stage to the metric config within the canary config. The ui changes aren't reflected in the workshop installation yet, so you'll need to manually edit the json.