Automated Canary Analysis Workshop

Spinnaker Summit - 10/8/2018

Agenda

Canary Release Overview

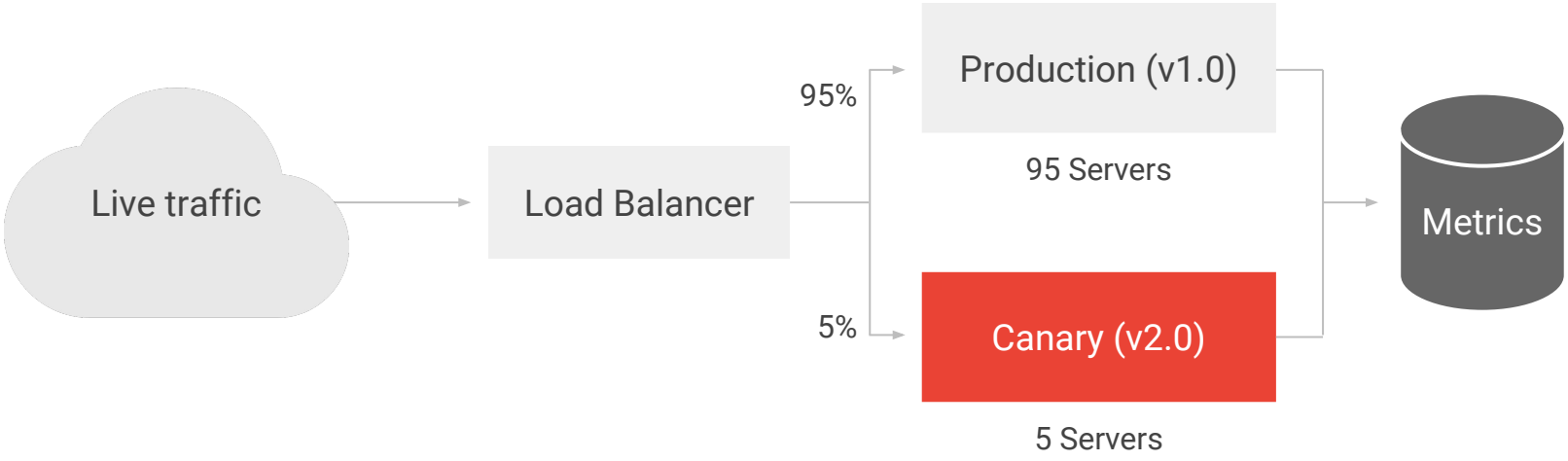
Spinnaker/Kayenta Overview

Provision Spinnaker/Kayenta & Sample Artifacts

Good/Bad Indicators Of Safety

Exercises

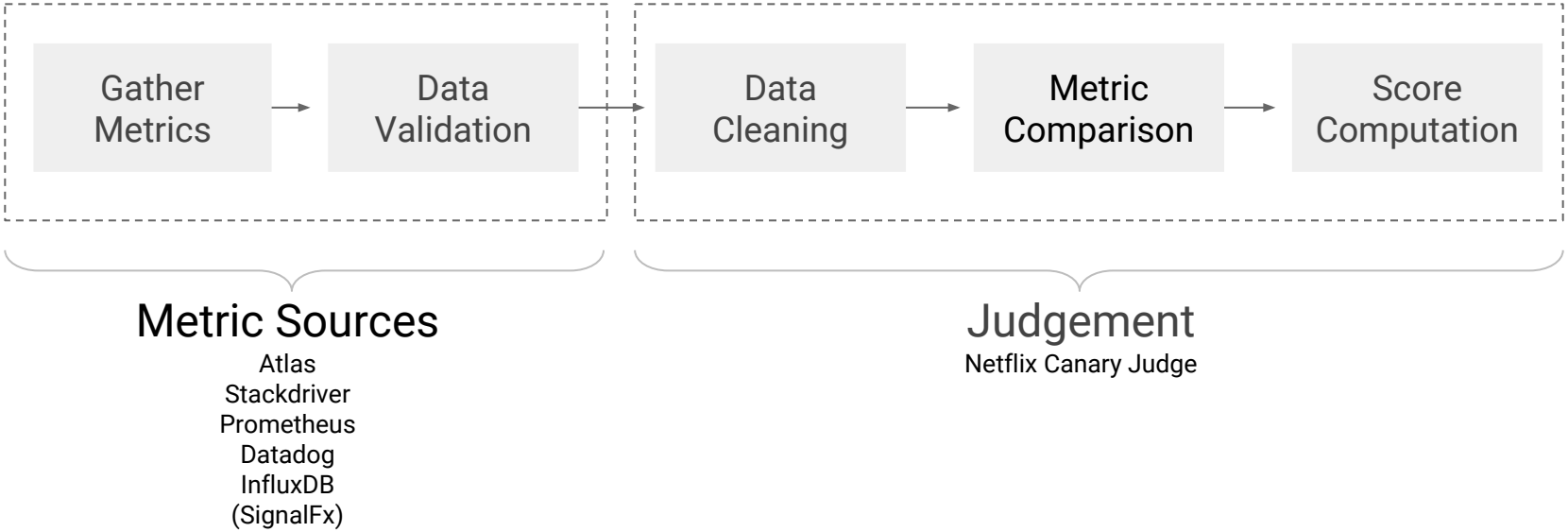
Canary Release Overview



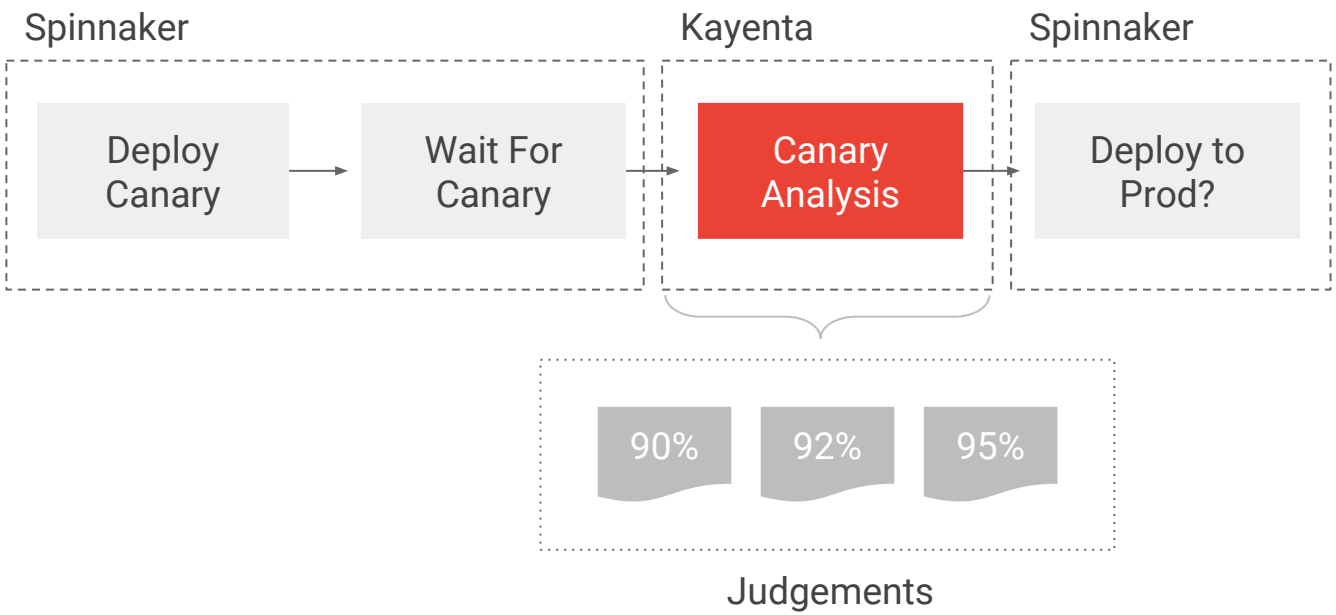
Canary Release Overview

Canary Release Overview

Spinnaker/Kayenta Overview



Spinnaker/Kayenta Overview



Provision Spinnaker/Kayenta & Sample Artifacts

Grab a temporary account id/pw.

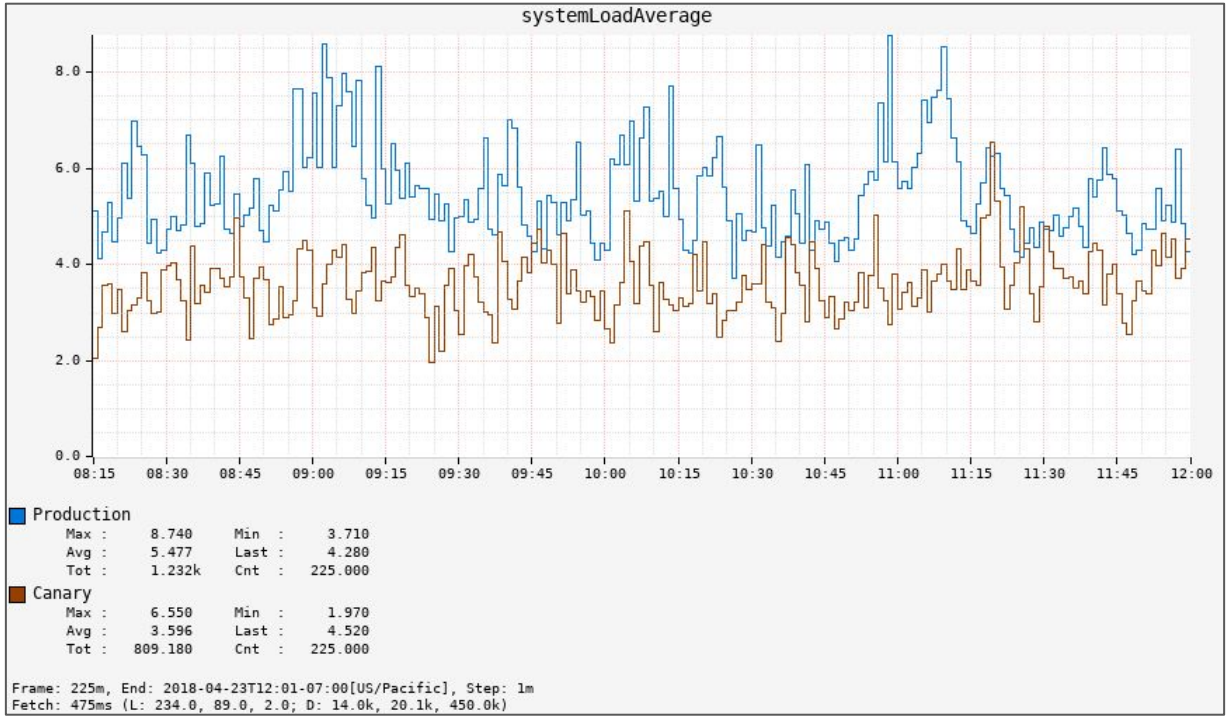
Navigate to:
https://codelabs.developers.google.com/codelabs/cloud-spinnaker-kubernetes-cd/index.html#0

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

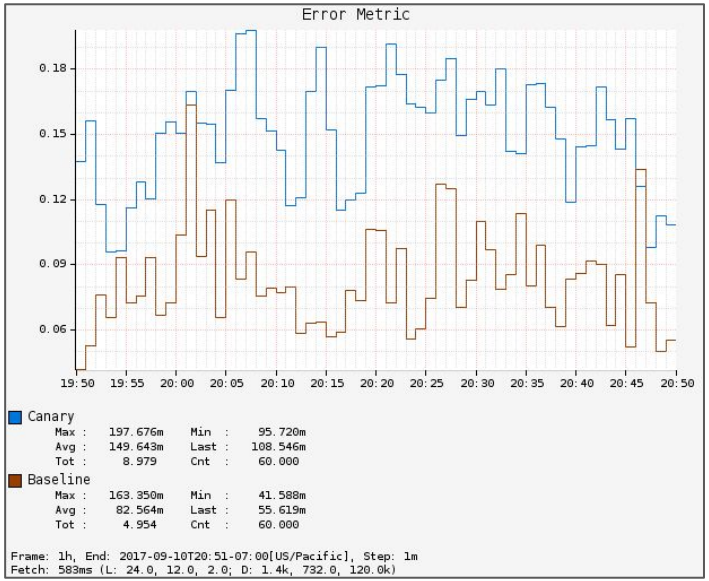
```
gsutil cp gs://gke-spinnaker-codelab/base/install.tar .
tar xvf install.tar
./setup.sh kayenta-workshop
```

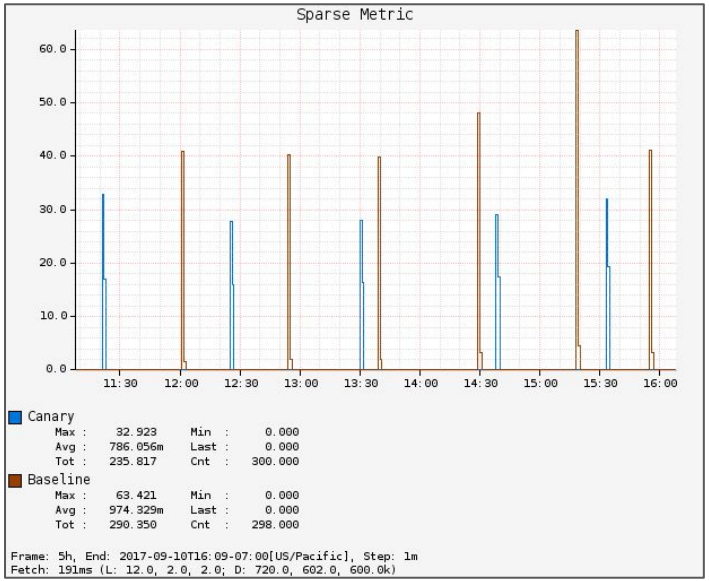
- Provisioning & configuration should take 10-15 minutes to complete. -

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:

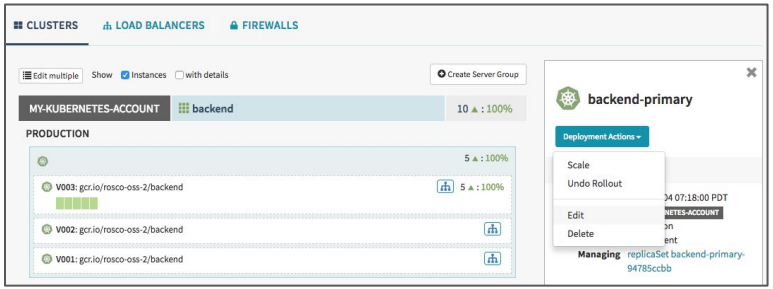
```
Group By: label top_level_controller_name
Aggregation: mean
```

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.

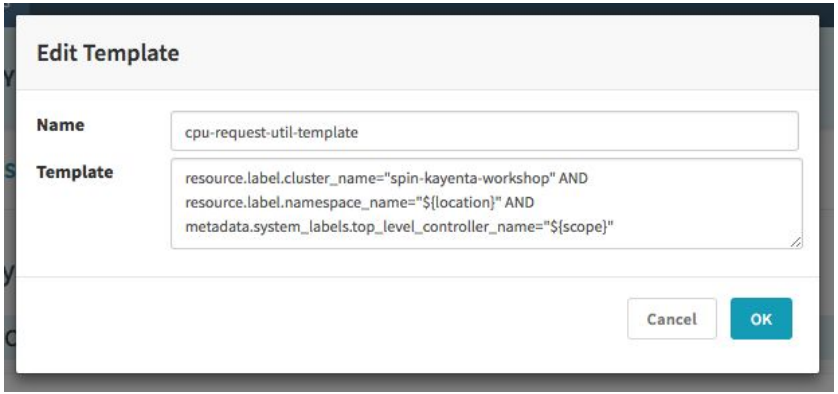


Exercises

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

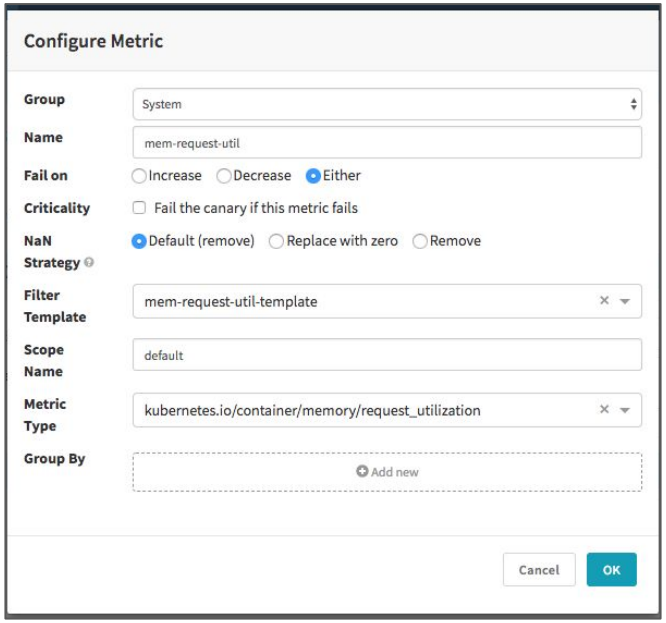
Configure a new Template named "mem-request-util-template" with this definition:

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



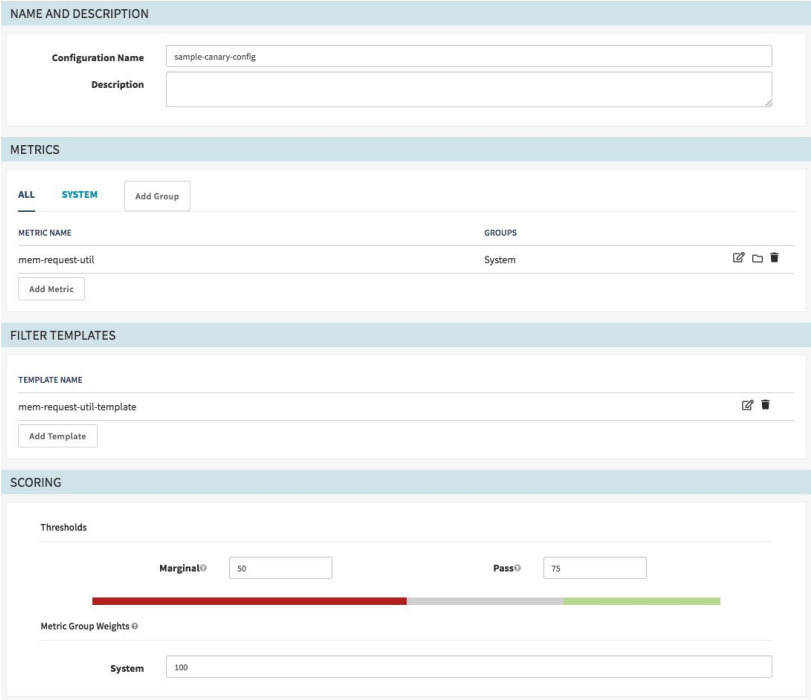
Exercises

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



Exercises

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

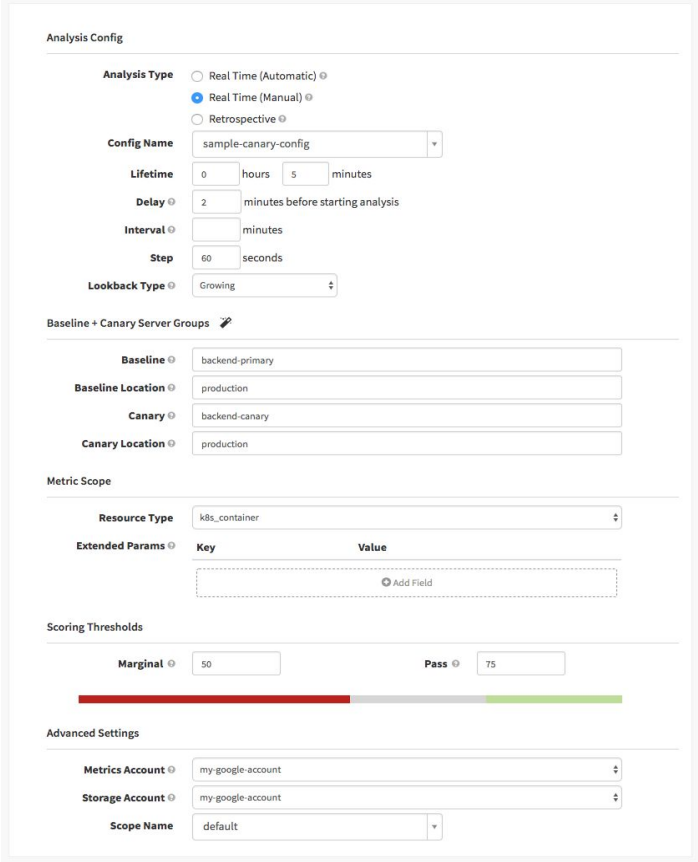


Exercises

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.



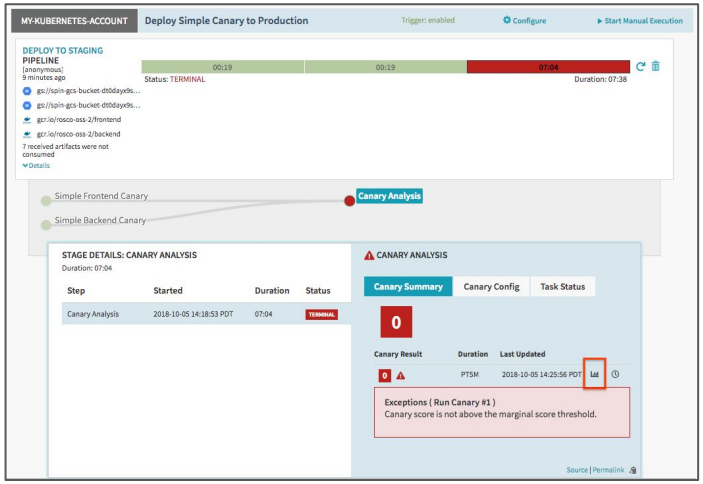
Exercises

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:



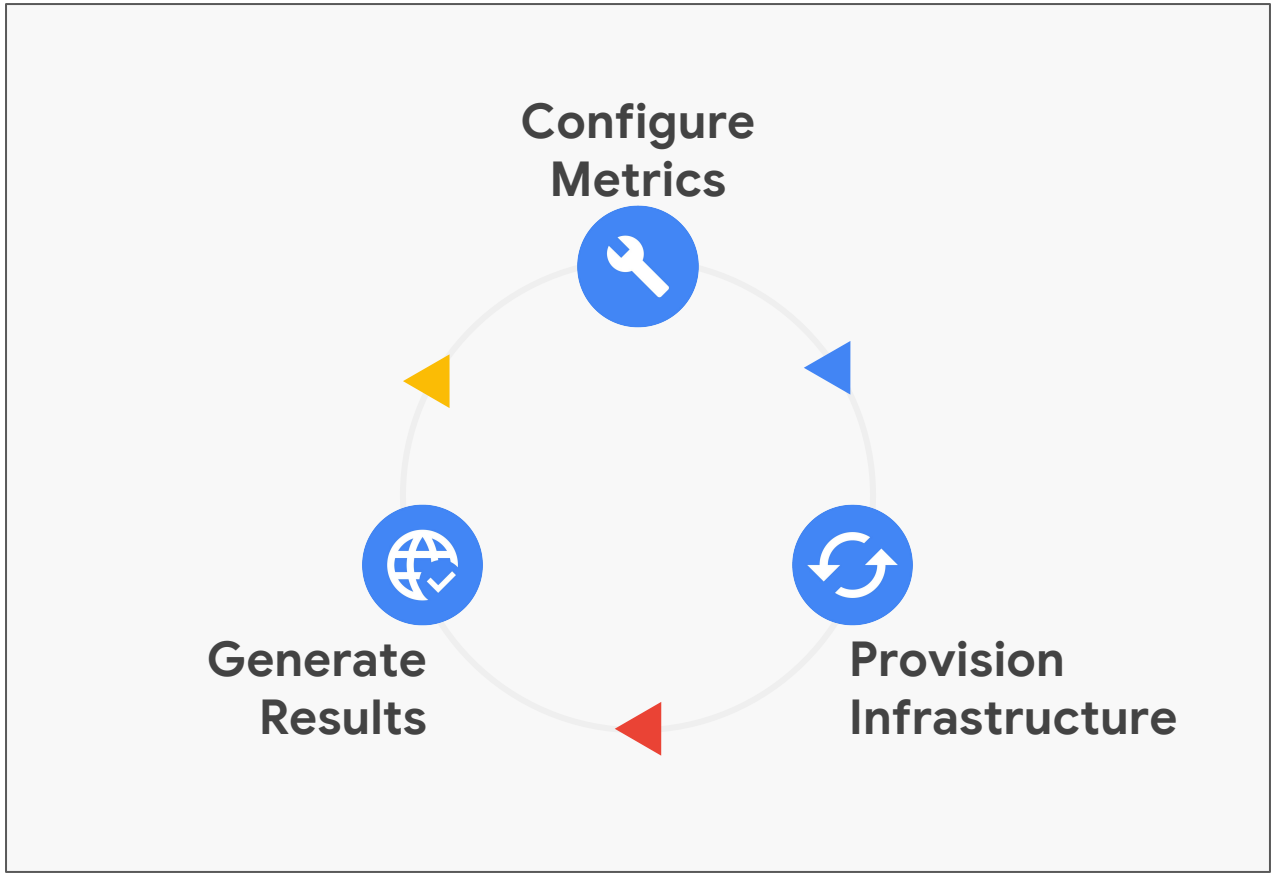
Exercises

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.


Configure Metrics
Provision Infrastructure
Generate Results

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.