Ingress-based Canary for Mothership on Cloud

https://tutuplapak.atlassian.net/browse/DPT-412

Follow up from Percentage-based Canary for Microservices.

See also Mothership Cloud Deployment And Canary / Beta Proposal.

Summary

This document outlines how we will route canary traffic for Mothership using NGINX ingress rather than doing it in BLINX as we currently do now.

The new approach will enable us to expose Mothership canary to end-users in a staged manner: 0% at the beginning, enable access for `blcanary` user agents only `1%` `5%`, before we do a full rollout of Mothership.

It will be implemented only for Mothership on the cloud; Mothership on-premise will use the approach outlined here: Ingress NGINX Controller for Each Mothership Deployment Types and still using BLINX to split the traffic.

All of the mechanisms outlined here are tested in the proof of concepts.

Table of contents

- Summary
- Current implementation
  - User-agent based traffic splitting
  - Weighted traffic splitting
- Target implementation
  - NGINX ingress version
  - User-agent based traffic splitting
  - Weighted traffic splitting
  - Staged canary deployments
- Proof of concepts
  - User-agent based traffic splitting
  - Weighted traffic splitting
  - Traffic weight control from pipeline
  - Implementation

Current implementation

Currently, canary traffic splitting is configured in BLINX. BLINX has two upstreams: one for Mothership production deployment and one for Mothership canary deployment; each Mothership deployment (`mothership-web` and `mothership-api`) have their own upstream configuration.

User-agent based traffic splitting

All requests will be redirected to Mothership canary deployments by BLINX if the `User-Agent` HTTP header request contains `blcanary`.

In the last 30 days, these user agents are used to access canary deployments.

User agent	Requests	%
`blcanary`	998,351	94.73%
`blandroid blcanary`	30,227	2.87%
Other, including standard browser user agents containing `blcanary` string (e.g. `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36 blcanary`)	25,306	2.4%

(There's one instance of configuration where `blkubecanary` user agent will also be redirected. It is considered as a misconfiguration though, and we found no requests using `blkubecanary` in the last 30 days, so it won't be supported in the target implementation.)

Weighted traffic splitting

Aside from user-agent based splitting, `mothership-api` canary pods also receive end-user traffic, because BLINX uses the Kubernetes service of `mothership-api` that select both production and canary pods. The amount of traffic that will be received by the canary pods then will be 9 / 228 = ~3,94% of total traffic (calculated from the ratio between canary and production pods).

Note: `mothership-web` canary pods don't receive end-user traffic, as BLINX uses the Kubernetes service of `mothership-web` that only select production pods. This is also considered a misconfiguration. In the target implementation, we will include `mothership-web` canary pods to receive a small percentage of production traffic.

Target implementation

Mothership will be fronted by GCLB and NGINX ingress in the cloud. Canary traffic splitting will be done in NGINX ingress.

The GCLB implementation will be discussed in a separate document.

The implementation is based on the work outlined here: Percentage-based Canary for Microservices.

NGINX ingress version

Currently, we use NGINX ingress v0.29.0 in k8s-preproduction-2 and k8s-production-2 clusters. We choose to keep using this version for Mothership because it is already battle-tested to serve production traffic and we prefer stability on implementing Mothership stack on the cloud.

User-agent based traffic splitting

The NGINX ingress v0.29.0 doesn't support matching `User-Agent` HTTP header using regex. Instead, it can only do an exact match. It is only supported starting in v0.31.0, but we haven't tested it to make sure we don't get weird bugs.

Hence, we will only use `blcanary` as the user agent **without any other strings around it**.

Unless there are concerns from related teams about this, we will not try to upgrade the NGINX ingress to enable the regex use case (which only covers ~5% of the existing `blcanary` usage in the last 30 days).

Any concerns about this should be raised in the #mothership-cloud-migration channel. (We expect none, though.)

Weighted traffic splitting

NGINX ingress will split the production traffic based on a weight that we configure in the ingress. The amount of traffic going to canary pods then will not depend on the number of canary pods v. production pods anymore.

Staged canary deployments

The side effect of configuring the traffic in ingress is that we can do staged canary deployments for Mothership now.

The deployment pipeline now will do this:

- At the beginning of canary deployments, the ingress will be configured to send no production traffic to canary pods. This way, TEs can trigger smoke tests and do exploratory tests using the `blcanary` user agent.
- After TEs cleared the deployment, the ingress will be configured to send 1% of the production traffic.
- If metrics are fine and no new errors are found, the ingress will be configured to send 5% of the production traffic.
- In case of bad deployments, the ingress can be reset back to send no production traffic to canary pods.

All configuration will be done through the deployment pipeline that is discussed in a separate document.

As no production traffic will be sent to canary pods at the beginning of canary deployments, all Mothership production pods should be able to handle 100% of the traffic. This will impact the sizing of Mothership on the cloud.

Proof of concepts

User-agent based traffic splitting

Exact matching the user agent works well. The implementation can be seen here.

Weighted traffic splitting

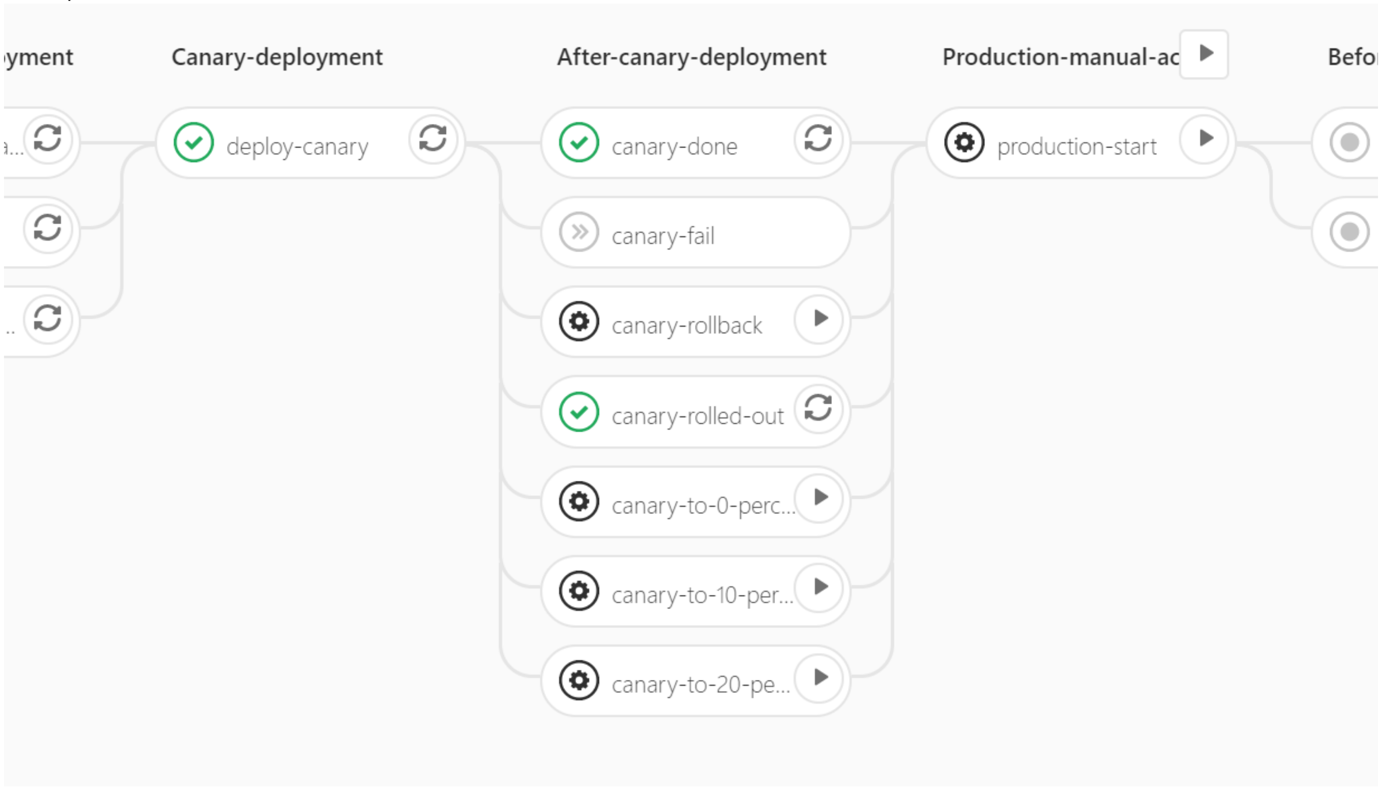
Weighted traffic splitting works flawlessly and tested in this document: Percentage-based Canary for Microservices. The implementation can be seen here.

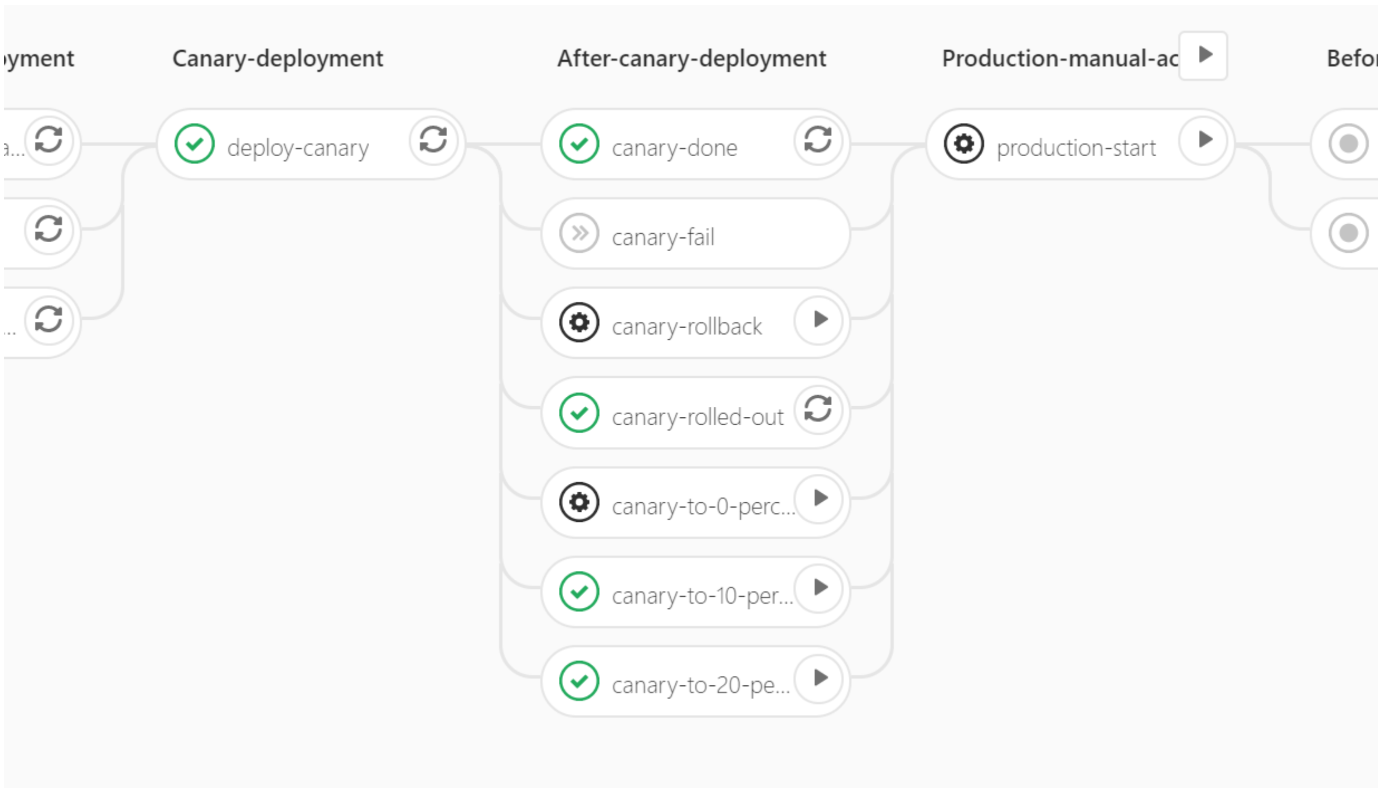
Traffic weight control from pipeline

To control canary traffic weight from the deployment pipeline, we tried to implement it in this way:

1. First job, `deploy-canary`, will deploy canary pods and set the canary traffic weight to 0%.
2. After it is fully rolled out, we can access the canary using the specified user agent.
3. Three additional jobs are provided to adjust the canary traffic weight without downtime:
   a. `canary-to-10-percent` will set the canary traffic weight to 10%.
   b. `canary-to-20-percent` will set the canary traffic weight to 20%.
   c. `canary-to-0-percent` will set the canary traffic back to 0%, in case of a bad canary deployment.

The implementation can be seen here.





Implementation

This is already tested in Mothership preproduction: https://gitlab.cloud.bukalapak.io/bukalapak/mothership/pipelines/562428.