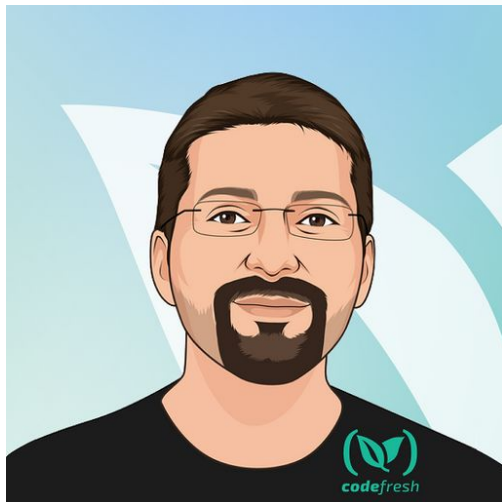




# Argo Rollouts and the Downward API

GitOpsCon US 2024

## Your host



Kostis Kapelonis

`kostis@codefresh.io`

Developer Advocate - Codefresh

Argo Maintainer

Co-author of GitOps certification with  
Argo -> <http://learning.codefresh.io>



# About Codefresh (acquired by Octopus Deploy)

## Modern Deployment Platform

Support for GitOps environments

## Enterprise Ready

Code-to-cloud visibility  
across apps and clusters

## Continuous Delivery

Progressive delivery without  
compromising stability powered  
by Argo CD and Argo Rollouts

The screenshot displays the Codefresh web interface for a release pipeline. The top navigation bar includes the Codefresh logo, a search bar, and user information. The left sidebar contains a navigation menu with sections like 'Getting Started', 'Home Dashboard', 'PIPELINES', 'WORKFLOWS', 'OPS', and 'ARTIFACTS & INSIGHT'. The main content area shows a release for 'my-product' (cloud deployment) with ID 'id281ghj3971' and version 'v.1.30.7'. It details a merge pull request and provides options for 'Summary', 'Retry', and 'Terminate'. The pipeline is visualized as a series of stages: 'Development (non-production)', 'Staging (production)', 'US-East (production)', 'EU (production)', 'Asia (production)', 'US-West (production)', 'US-Central (production)', and 'Asia-2 (production)'. Each stage lists its steps and their durations. The 'EU (production)' stage is highlighted with a red border and contains an 'Issues' section with a red '4' icon, indicating a failure in the 'e2e\_test' step.

# Agenda

1. Argo Rollouts and microservices
2. Kubernetes Downward API
3. Argo Rollouts ephemeral labels
4. Cooperation with developers
5. Demo

# Argo Rollouts basics



# The Argo Family of projects

argo

Workflows CD Rollouts Events Blog



## Argo Rollouts

  2405

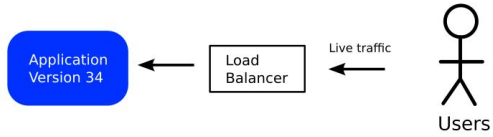
Advanced Kubernetes deployment strategies such as Canary and Blue-Green made easy.

[Learn More](#)

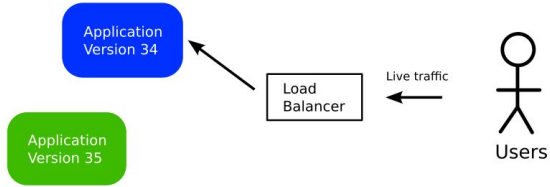
<https://argoproj.github.io/>



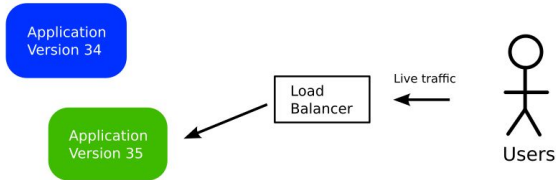
### 1- Initial version



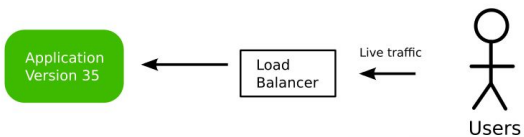
### 2- New version deployed



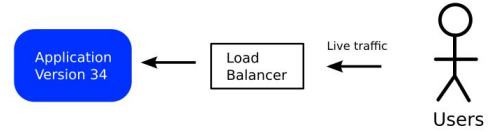
### 3- Switch Traffic



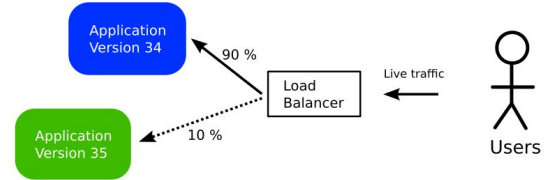
### 4- Finish



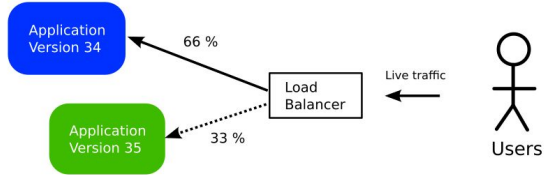
### 1- Initial version



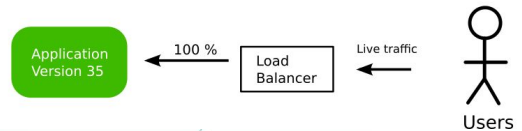
### 2- New version used by 10% of users



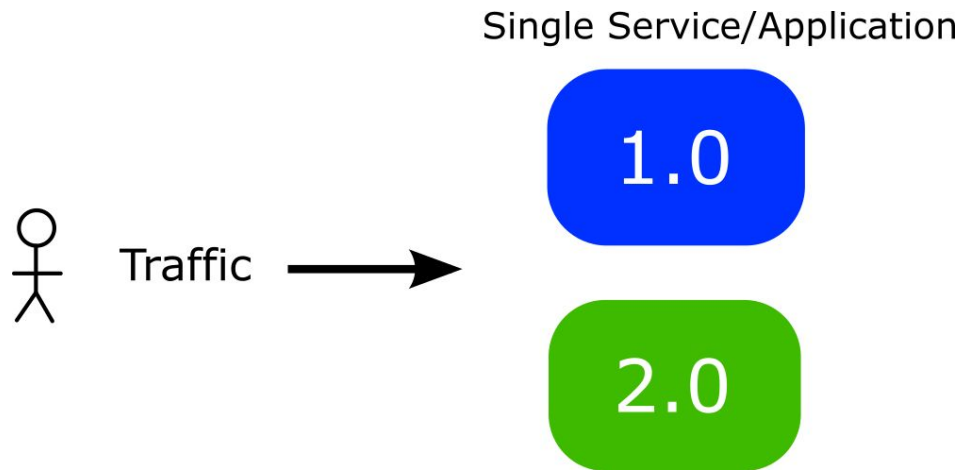
### 3- New version used by 33% of users



### 4- New version is used by all users



# Argo Rollouts is single service only

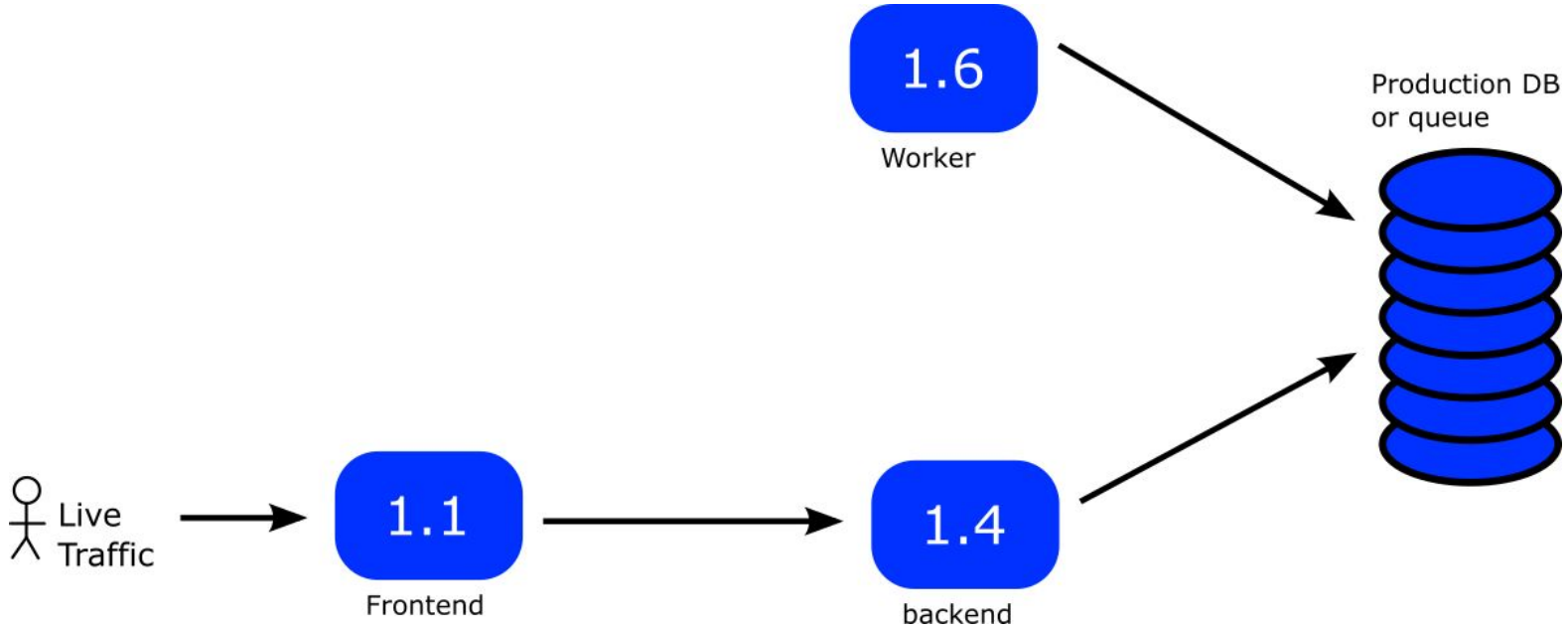




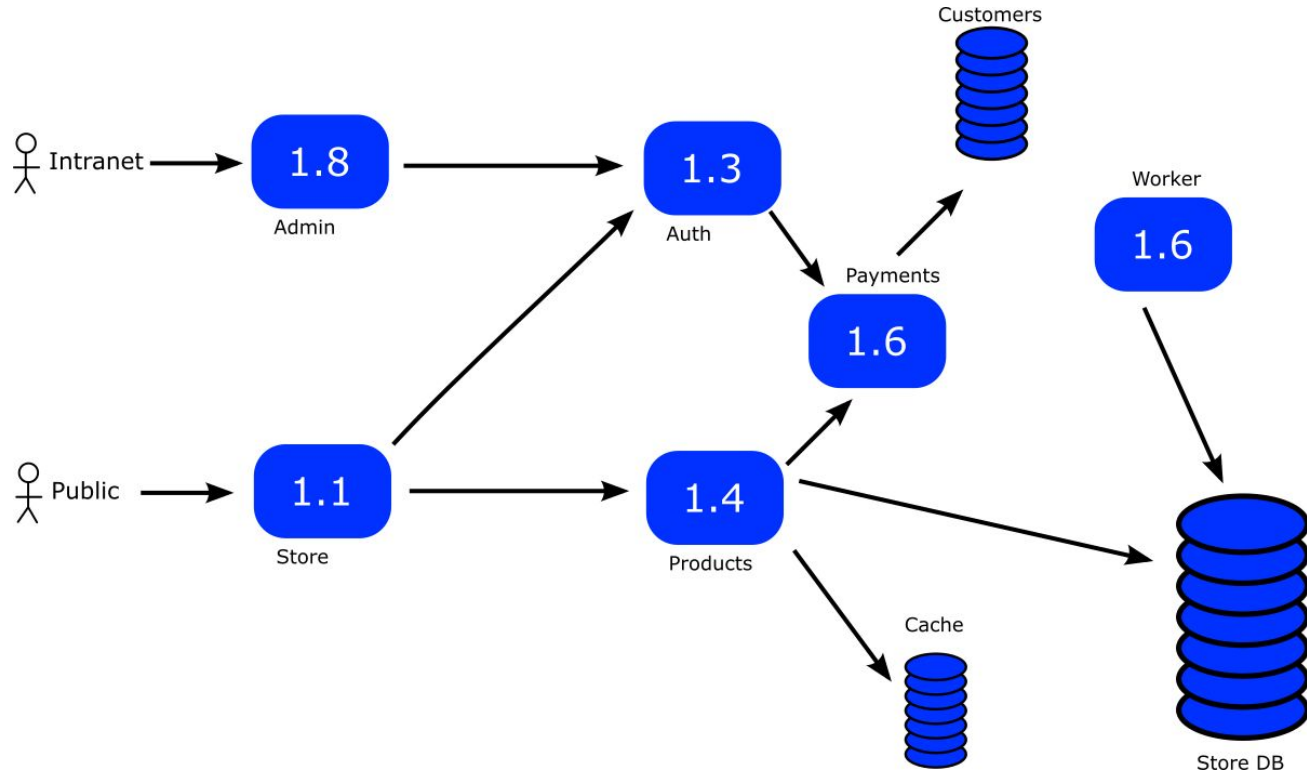
# Using Argo Rollouts in the real world



# Typical application



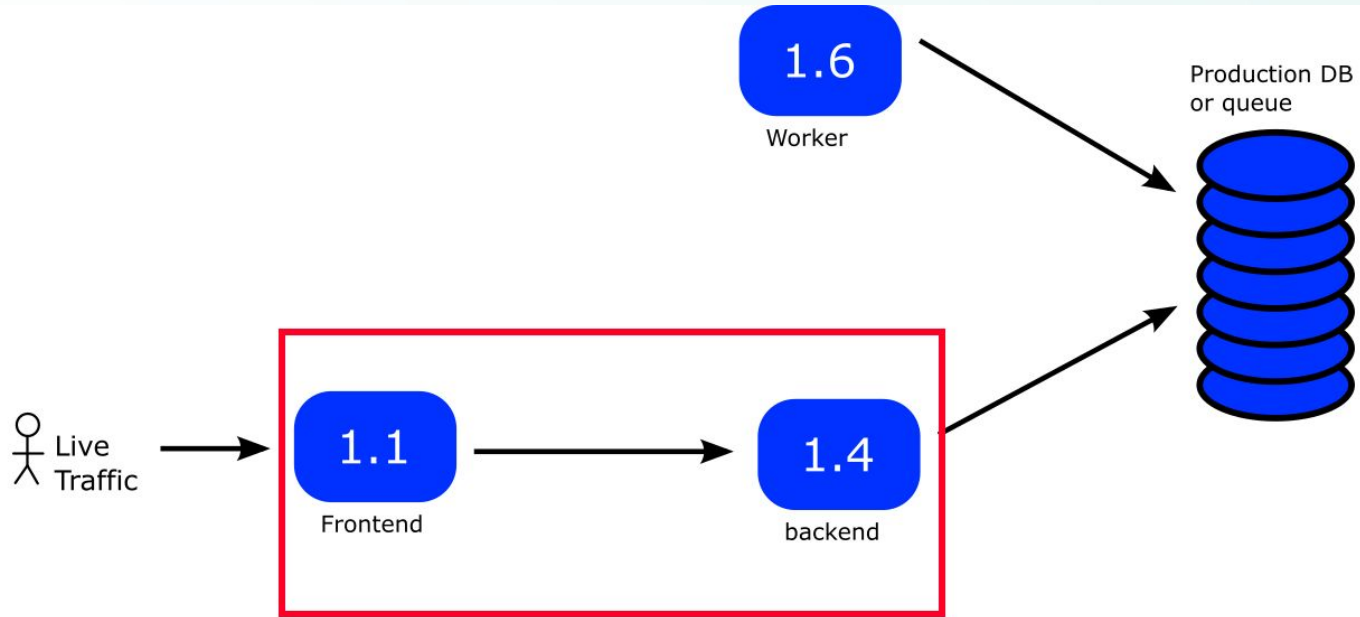
# Complex application



# Problem 1 – coordinating two services

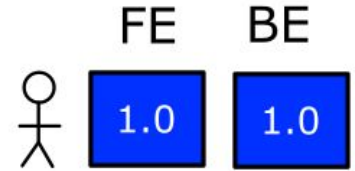
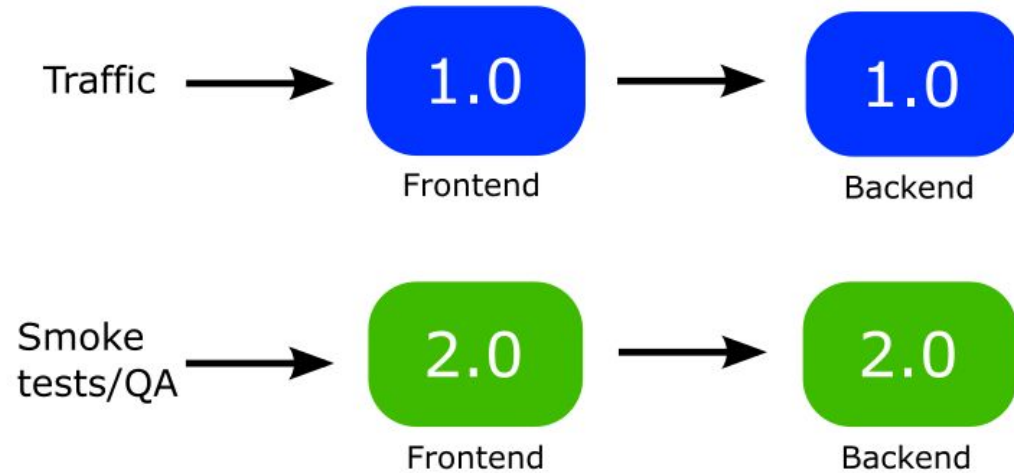


# Problem 1 – coordinate two services



Always upgrade in tandem

# Solution 1 - coordinate two services



# People want to coordinate backend-frontend

**Erik** 2 months ago

Hi @Kostis, part of the challenge is when a service is promoted and has dependencies on other services. For example, in this diagram where serviceA and serviceB must be deployed together because of an API breaking change. In the preview test scenario, we are able to validate the flow e2e because of a test entry point to preview serviceA which then points to preview serviceB. However, it is after the promotion that it is not clear. When serviceA is promoted and accessible via the stable entry point, it still points to preview serviceB.

image001.png

**Jakub Oškera** 1 year ago

Hello Argo users 😊

I have an interesting use case for Argo rollouts.

Let's say that I have two rollout manifests within one Helm Chart, where one rollout manifest is for service A (frontend) and the other rollout manifest is for service B (backend). Each service has 2 service manifests created (active and preview). Pods of service A communicate with service B using internal k8s service.

How to ensure that when I deploy a new version of Helm Chart, the new (preview) version of service A communicates with the new (preview) version of service B, before I manually promoted both rollouts?

**JP** 2 years ago

Hi guys! We're evaluating the Argo Rollouts Blue-Green Deployment. Is it possible to use a different ConfigMap for previewService? The reason why I asked about this is because we have an application (`MAIN_APP`) that has a configuration or environment variable (`main-app-cm`) that proxies traffic to another service, something like this, for example: `OTHER_APP=http://service:port`, and this `OTHER_APP` is also a part of our deployment. I would like to tell the preview version `MAIN_APP_PREVIEW` to use the `main-app-cm-preview` ConfigMap that has `OTHER_APP=http://service-preview:port`.

## TLDR:

- `MAIN_APP -> main-app-cm -> OTHER_APP=http://service:port`
- `MAIN_APP_PREVIEW -> main-app-cm-preview -> OTHER_APP=http://service-preview:port`

**Jacob Hagstedt** 2 years ago

Is it possible for a pod in a canary release to know that it is the canary? When running analysis I only want to include the prometheus metrics for the canary release, so my idea is to include a metric-label that says if it is canary or not.

Is this a good approach or is there something else I can do?

**James Brady** 1 year ago

Hey everyone, general guidance question:

We have a frontend app and a backend app and we'd like to deploy new versions of them together (to avoid API versioning and typing headaches). Our desired deployment strategy is something like:

- Deploy new version of the backend
- Wait for it to be healthy (no frontend traffic is going to it yet)
- Deploy new version of the frontend
- The frontend app's requests hit the new version of the backend
- After some period of time / reduction in requests, the old backend is shut down

So similar to blue-green deploys, except that at the moment the front- and backends are separate Deployments. I don't see a way to configure rollouts to work across deployments in this way.

One option would be to combine the front- and backend apps into one pod, but I was wondering if there is a different/better option which keeps the two apps more separate, and yet in lockstep with version updates? (edited)

**Anil Kumar** 10 months ago

Hello , I have a requirement where I need to deploy two rollouts and promote them simultaneously because both rollouts are dependent on each other i.e. frontend and backend so if they do not promote at the same time it will be a downtime till the time both comes with the same version of code . I am achieving this by a shell script to check for rollouts and promote when both deployed with new version .Any good solution would be appreciated .

Also sometime I have seen rollouts remain in paused state even after I promoted that rollout and I need to run promote command again to get that rollout in healthy state .

Thanks in Advance . (edited)

**nvcnvn** 2 years ago

Hello, please excuse me for my bad English.

I'm looking for a best practice to do progressive rollout in case you need to coordinate API sever with Web App.

For example, (1) in case we have new feature.

- Backend service have new DB tables, new REST endpoints
- Frontend have new screen, new feature


But another case, (2) only some performance improvement/refactoring with backwards compatible from backend side.

For case (2), I guess it really straightforward that progressive rollout can reduce the risk.

But for case (1), how can we coordinate the rolling out of BE and FE? From FE side we are using Unleash & Firebase Remote Config to show the feature to a small set of user before 100% rolling out.

... strategy in this case, maybe Blue/Green and then some how send the traffic of new feature to the new

# How to coordinate two services



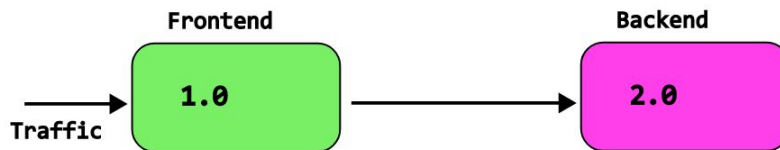
**Frontend microservice (loans)**

My version is 1.0 also available at [version](#)

I use a backend at [backend-preview:8080](#) Backend has version 2.0

- Liveness endpoint is at [health/live](#)
- Readiness endpoint is at [health/ready](#)

Enter your loan amount to see the interest. \$

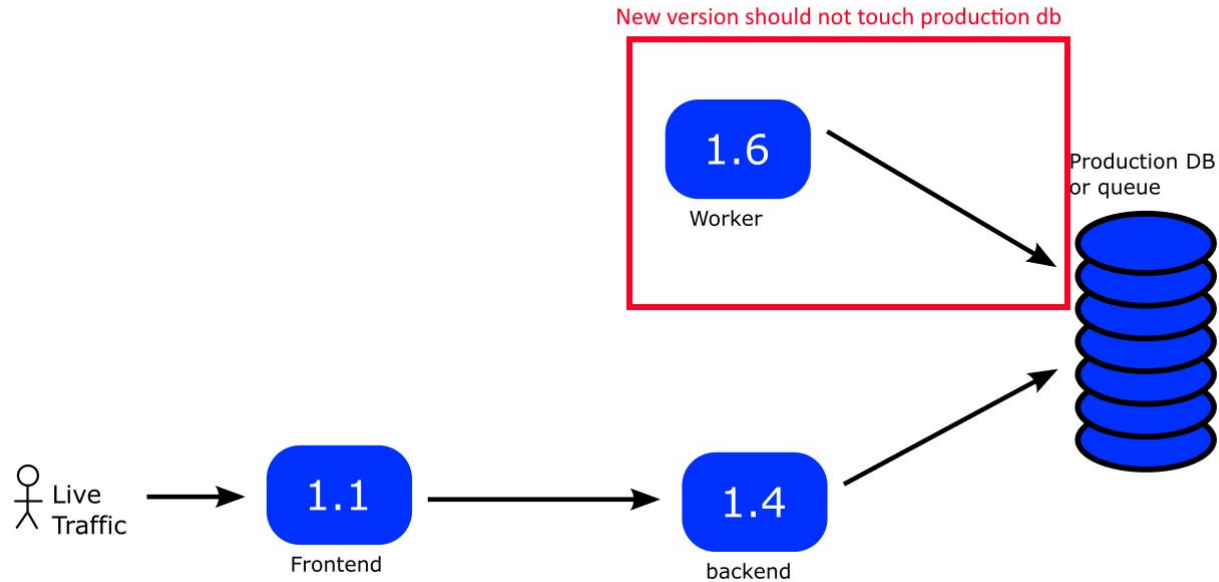


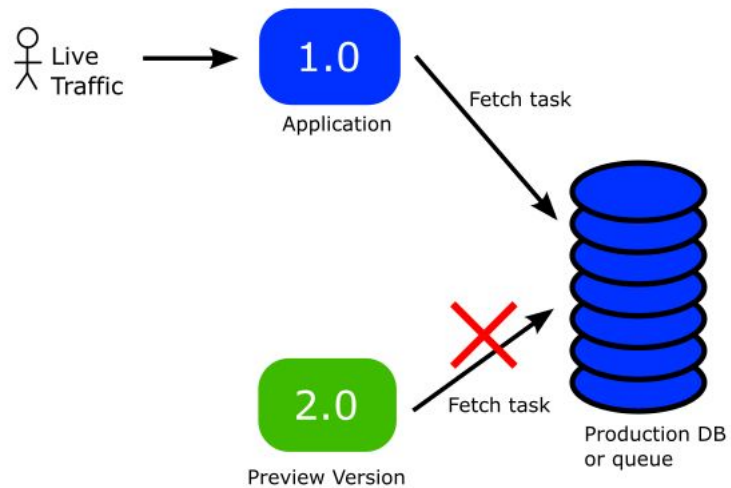
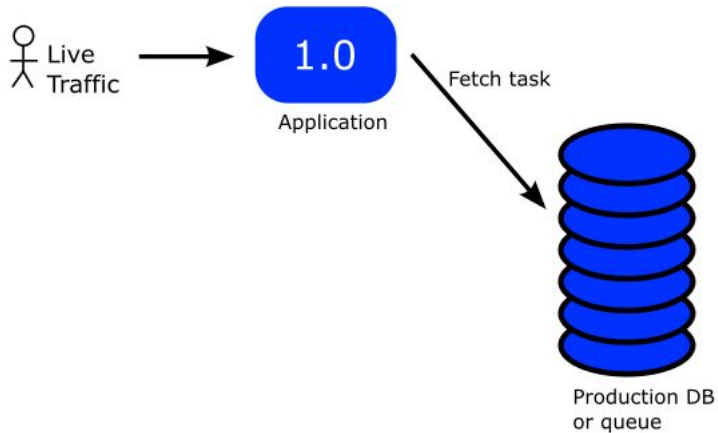


# Problem 2 – queue/task workers

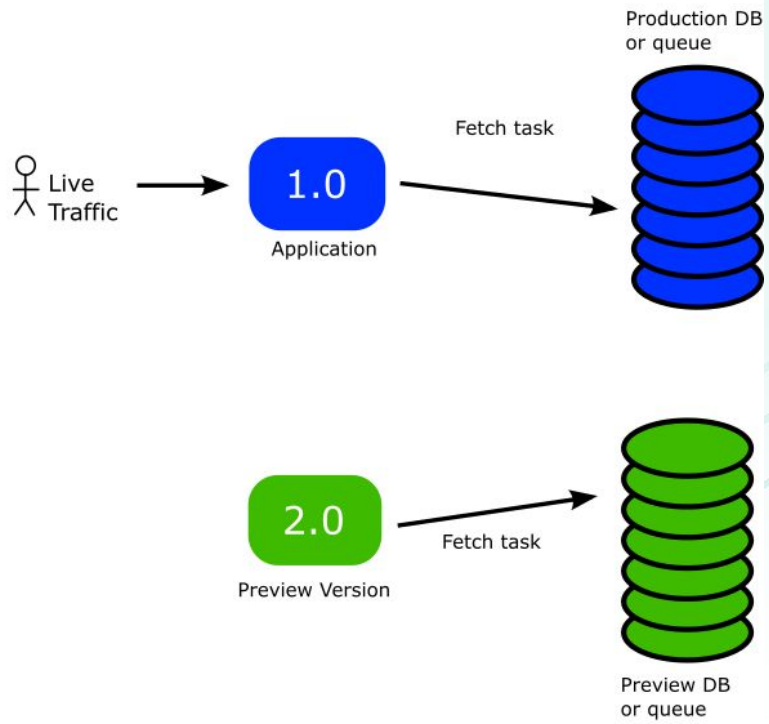
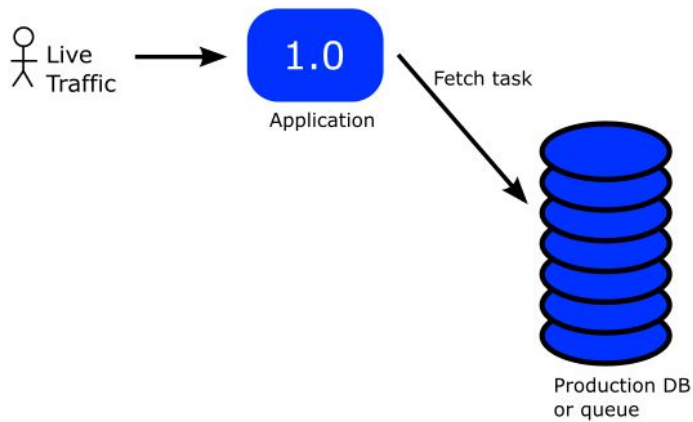


## Problem 2 - queue/task workers





Time



Time

# People want to use Rollouts for workers

## Support queue worker paradigm #438



derekperkins opened this issue on Mar 11, 2020 - 6 comments



derekperkins commented on Mar 11, 2020

Contributor · ···

Currently argo-rollouts mainly targets http/grpc services, which is great for that use case. Most companies also have queue driven workers that would benefit from the same type of deployment strategy. Rather than managing the traffic directly like you can with a service / mesh, this would require more integration from the application, but I think it could be handled without major architectural changes to rollouts as is.

My hope would be able to set up a watch on the Rollout object as it progresses through its various states, and configure my application accordingly. For example, if my current replicaset is in the preview state for blue/green, I would expect to be able to watch an annotation or a label on the replicaset itself, or alternatively watch the status field of the core rollout object. While in preview mode, I might set my worker to run as normal, but rollback any database transactions rather than committing them. When promotion happens, my app would see that in the watch and start committing those transactions.

I understand that it might not be as magical as being able to migrate live traffic. I believe it is a common use case that isn't being served today. As is, I could probably hack rollouts to do what I'm suggesting by deploying two services that don't serve any purpose except to be the meta that my app would watch. It just seems like unnecessary churn and ip allocation.



Thread # argo-rollouts

Ariel Halm 27 days ago

Hey all, using B/G strategy I want to expose the current rollout state from the pod to the container. Using Kubernetes downward API I set an environment variable on the container with the value of the 'role' label of the pod, this works well until I promote the rollout. When promoting the rollout new pods are created in blue state and then updated to green when the desired number of pods are created, so all of them are initialized with an environment variable with a value of 'preview'. Is there a different way to expose the rollout state to the container or creating new pods directly in active mode when promoting a rollout? (edited)

6 replies



Andre Marcelo-Tanner 8 months ago

---

How do some people use ArgoRollouts for Workers? eg things consuming from a queue?

I read <https://github.com/argoproj/argo-rollouts/issues/438>

An idea I was thinking about

- Have the worker use a special queue for preview/canary and grab this from the canary/previewMetadata fields passed in as an env var



Deepak Mishra 5 months ago

Hi #argo-rollouts, I am trying to achieve blue green strategy in my organisation and find myself stuck in a situation. I have a pod which polls DB every 30 seconds. I have deployed both active and preview version of this pod. Problem is both the pods are fetching db and which ever gets it first it starts to process it. Not sure how to handle this situation.



John Thompson 2 years ago

If we've got a bunch of workers processing messages from SQS queues (or kafka topics or similar) -- how would you handle canary deploys in that case?



Illia Sokalau 1 month ago

@Kostis The blue app is connected to the blue database, but the green (experiment) should be connected to the green database. Then after testing is completed I'd like to replace blue database credentials and canary pods will use the same (new) credentials as the blue app. Then green database will become a new blue one. This is the idea. To test migrations on new database instance in-flight



Dan Bunker 5 months ago

basically run an integration test against new app code before exposing it to prod traffic, one of our apps is worker-based, so it doesn't receive http requests, but rather plucks messages off a queue and processes them. I wanted a way to use a canary queue name for the canary replica in an experiment, and then for the stable replica set I wanted to use the prod queue name. Unfortunately rollouts with integrated experiments don't scale down the canary replica set after experiment passes; It simply hot swaps the metadata labels/annotations and converts the pod into a stable pod without restarting it. but my app already started up with the canary queue name and doesn't support live-reloading of env vars (I tried using the k8s downward api to reference the canary/stable metadata values), and now standalone experiments can't be synced to a healthy state when you update it; you have to delete it and submit a new experiment, which doesn't work for my CI/CD process.



Michael Glaesemann 3 months ago

Hello! We're successfully using Argo Rollouts with Istio for traffic management for services. What are some equivalent types of things that we can use for progressive delivery of deployments that aren't services? For example, queue workers. I'd love to be able to shift the amount of work that's being done between different versions of workers. I don't think Argo Rollouts is really a solution for this (other than adjusting the ratio of worker pod replicas—nothing more fine-grained than that), but I'd love to be wrong. Are there alternative strategies people are using?



ebenezer popoola 1 year ago

Please, we have one helm chart that has two deployments. one of the deployment is an api server while the other is a celery deployment. Please, i need an example of how to use blue-green deployment on this because the celery deployment does not need a service.

in summary, i would say i need to know how to use argo-rollout for a "deployment" that does not need a kubernetes service? (edited)

# Argo Rollouts and workers/queues



The screenshot shows a worker's status page with the Codefresh logo at the top. The title is "Queue worker - 2.0 with role: preview", where "role: preview" is highlighted with a red box. Below the title are several informational sections:

- "My version is 2.0 also available at [version](#)"
- "My settings as read from /etc/podinfo/labels"
  - My current role is **preview** also available at [role](#)
  - My RabbitMQ server is at [rabbitmq:5672](#)
  - Reading messages from queue named **myPreviewQueue**
- Liveness endpoint is at [health/live](#)
- Readiness endpoint is at [health/ready](#)
- "I have processed 0 messages so far. Also available at [count](#)"

At the bottom, there is a section titled "Live log of messages processed" with a button labeled "Send dummy message" and a status indicator "Sent 0".

# The Solution



# Don't leave your applications in the dark



Photo by [Thanos Pal](#) on [Unsplash](#)



# Let your applications know where they are



Photo by [Meriç Dağlı](#) on [Unsplash](#)

“Hey – frontend – while you are running in canary mode make sure to use the endpoint of the canary backend.

Now that the canary is finished please use the production backend”

“Hey – worker– you are running in blue/green now.  
DO NOT touch the production queue.

Ok. We are ready. Please use the production queue  
now”

# Make your applications smarter

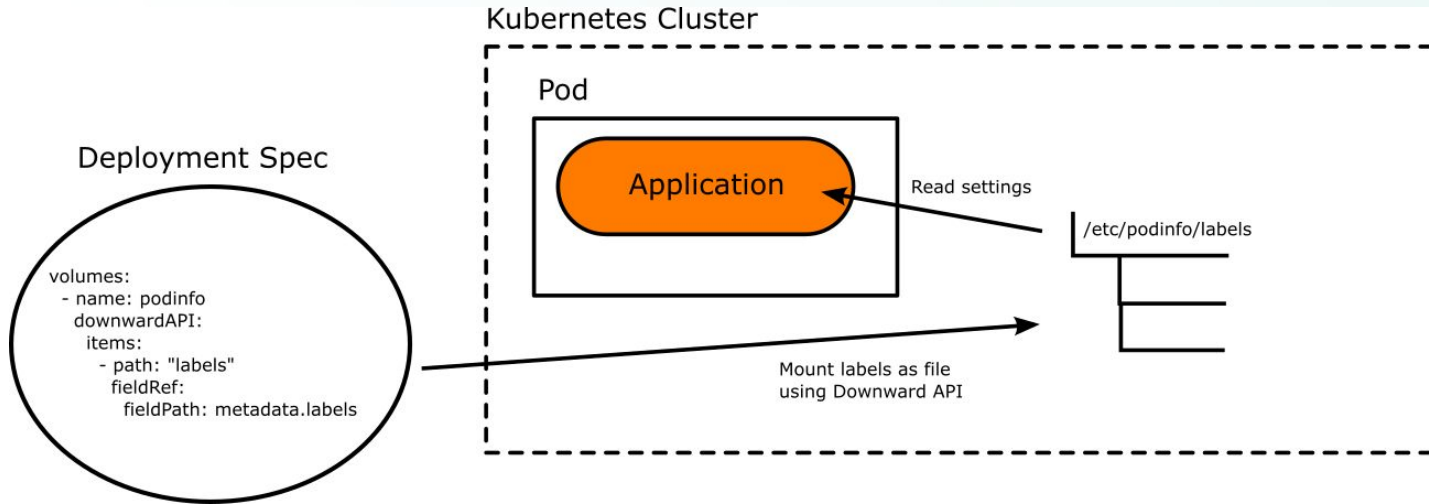
- Using the Kubernetes Downward API
- Argo Rollouts ephemeral labels
- Auto-reloading of configuration
- Co-operation with developers



# The Kubernetes Downward API



# Mount your labels as files (or env vars)



<https://kubernetes.io/docs/concepts/workloads/pods/downward-api/>

# Make your configuration smarter

1. Have labels that denote role (stable or canary)
2. Mount these labels to your application
3. Have the application source code read them
4. Do NOT use environment variables. Load from files

# Argo Rollouts Ephemeral labels





# Let Argo Rollouts instruct the app automatically

Instruct Blue/green app of its “color”

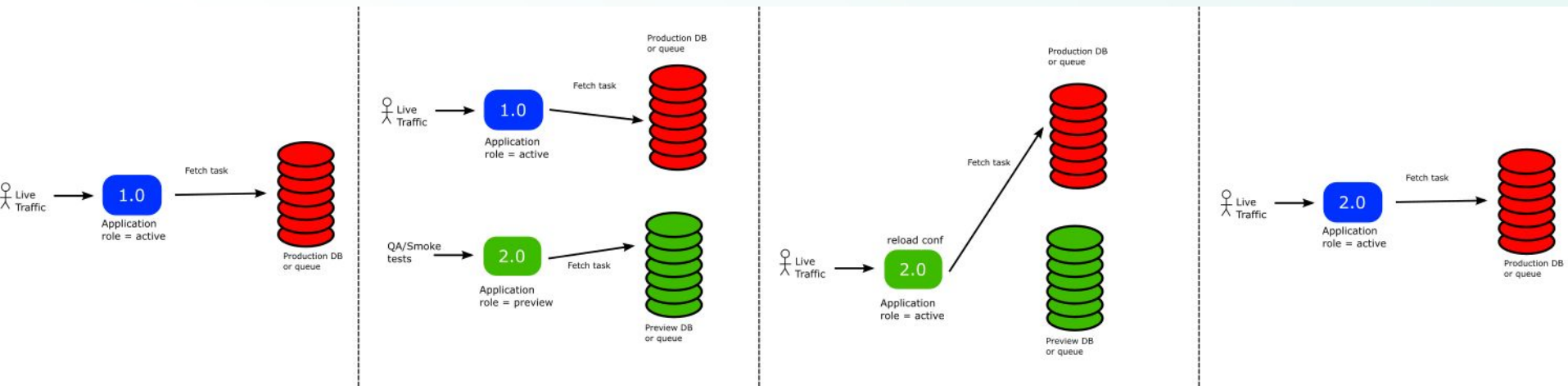
```
spec:
  strategy:
    blueGreen:
      activeMetadata:
        labels:
          role: active
      previewMetadata:
        labels:
          role: preview
```

Instruct Canary application for its promotion “status”

```
spec:
  strategy:
    canary:
      stableMetadata:
        labels:
          role: stable
      canaryMetadata:
        labels:
          role: canary
```



# Full Process



Time

# Auto-reloading of Configuration



# Make your application smarter

1. The application should read conf from files
2. DB/Queue URL must be configurable
3. Application should auto-reload conf on its own
4. You need to coordinate with your developers for this

```
viper.SetDefault("role", "unknown")  
viper.SetDefault("rabbitHost", "localhost")  
viper.SetDefault("rabbitPort", "5672")  
viper.SetDefault("rabbitQueue", "devReadQueue")
```

# Popular languages support

1. Viper Conf (Golang)
2. RefreshScope (Spring/Java)
3. chokidar/config (Node.js)
4. configparser/watchdog (Python)
5. yaml/listen (Ruby)
6. config/watchservice (Kotlin)
7. config/config-watch (Rust)
8. symfony/config-filesystem (PHP)



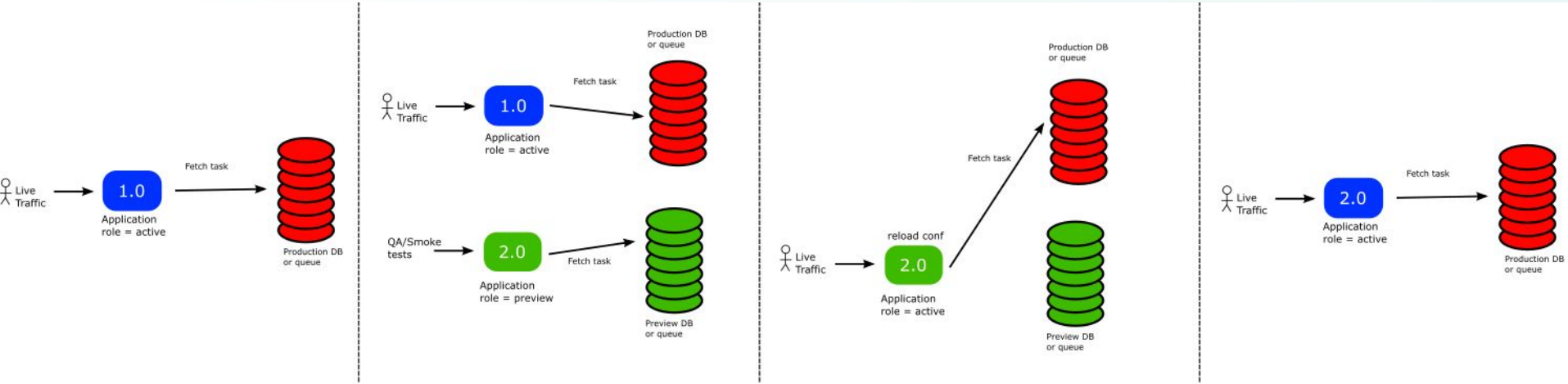
# Coordinate with the Developers



- Photo by [Sylvain Mauroux](#) on [Unsplash](#)



# Live demo



Time



# Demo Time

Argo Rollouts, Downward API, RabbitMQ, golang viper autoreload

<https://github.com/kostis-codefresh/argo-rollouts-stateful-example>



# Argo Rollouts and microservices



1. Use Kubernetes Downward API
2. Use Argo Rollouts ephemeral labels
3. Application should read configuration from files
4. Application should auto-reload its configuration
5. Enjoy !

# Questions?

[kostis@codefresh.io](mailto:kostis@codefresh.io)



Argo Rollouts



Downward API