TUNING ARGO ROLLOUTS FOR THOUSANDS OF WORKLOADS

Roxana Balasoiu / Carlos Sanchez

Roxana / Software Developer Engineer

problem-solving enthusiast, cloud engineering

balasoiuroxana / @balasoiuroxana

Carlos / Principal Scientist

OSS contributor, Jenkins Kubernetes plugin

csanchez.org / @csanchez

Adobe Experience Manager Cloud Service

ADOBE EXPERIENCE MANAGER

A Content and Digital Asset Management system

An existing distributed Java OSGi application

Using OSS components from Apache Software Foundation

A huge market of extension developers

Running on **Azure**

50+ clusters and growing

Multiple regions: US+, Europe+, Australia, Singapore, Japan, India, more coming

SERVICES

Multiple teams building services

Different requirements, different languages

You build it you run it

Using APIs or Kubernetes operator patterns

SCALE

17k+ environments

200k+ `Deployments`

10k+ namespaces

Already doing progressive rollouts at the environment and namespace level

CHALLENGES

How to avoid issues in production

deploying Adobe code / customer code

For 17k+ unique services

CHALLENGES

Full end to end testing is expensive

Does not cover all cases and does not scale

If a few environments fail it requires analysis

- is it an AEM release issue?
- is it a customer code issue?
- is it a temporary issue?

CHALLENGES

It is time consuming

Releases can get delayed

Issues can impact 100% of one environment traffic

OUR SETUP

Canary deployments with automatic rollback

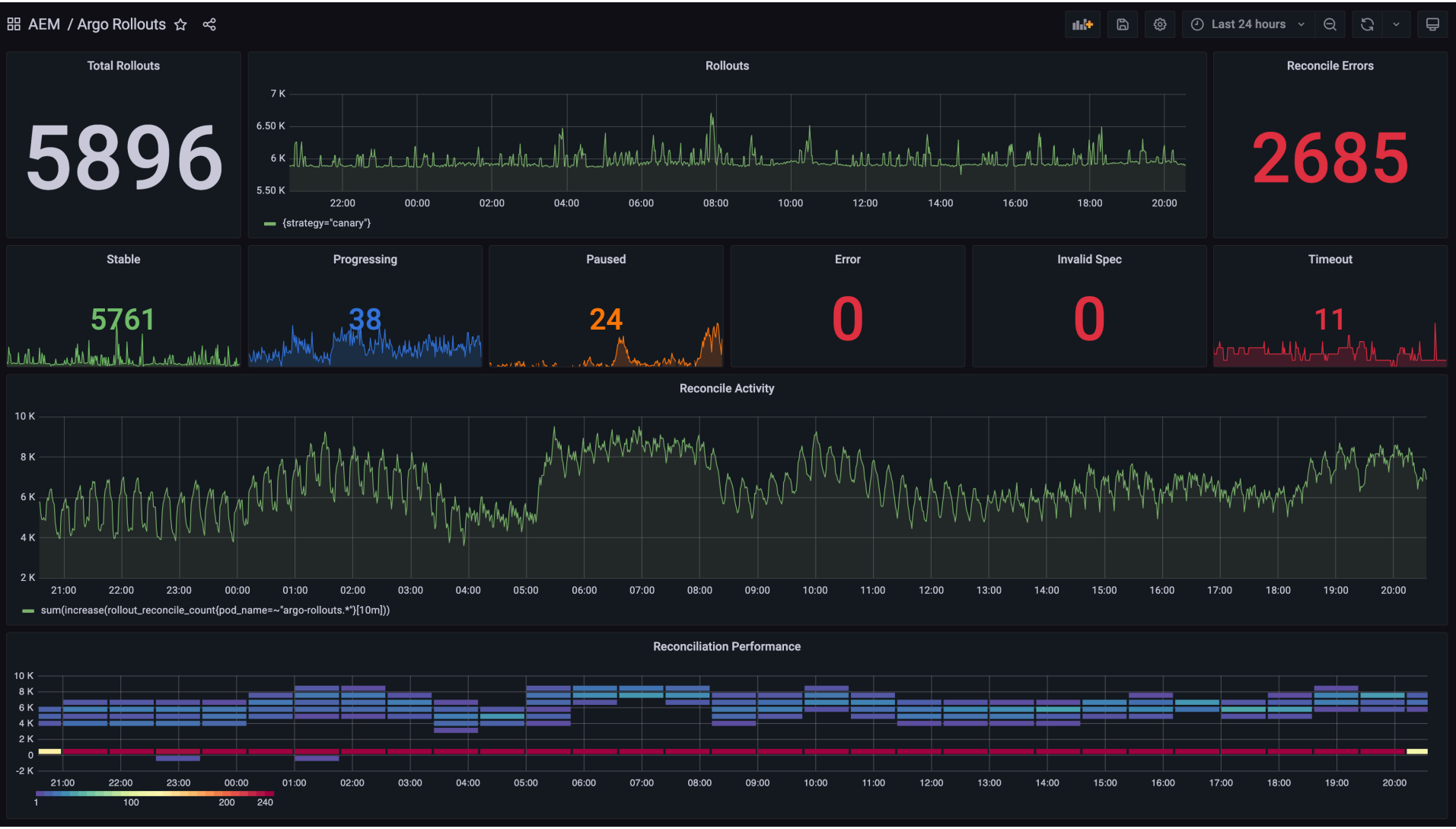
Based on real world traffic and error metrics

Using existing metrics from Prometheus

SCALE

6k `Rollout` objects

10k reconciliations in average (up to 5k per cluster)


AEM / Argo Rollouts
Last 24 hours
Total Rollouts
5896
Rollouts
7 K
6.50 K
6 K
5.50 K
22:00
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
{strategy="canary"}
Reconcile Errors
2685
Stable
5761
Progressing
38
Paused
24
Error
0
Invalid Spec
0
Timeout
11
Reconcile Activity
10 K
8 K
6 K
4 K
2 K
21:00
22:00
23:00
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
sum(increase(rollout_reconcile_count{pod_name=~"argo-rollouts.*"}[10m]))
Reconciliation Performance
10 K
8 K
6 K
4 K
2 K
0
-2 K
21:00
22:00
23:00
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
1
100
200
240

ROLLOUT OF THE ROLLOUT

Slow to avoid issues

Watch for quotas as Deployments are scaled down and Rollouts up

Dry run mode to look for issues

REVERTING THE ROLLOUTS

Disabling Rollouts require scaling up the `Deployment` object

MIGRATION

`Rollout` requires changing runbooks, tooling and training

Teaching engineers about `Rollouts`, `Deployments` scaled down to 0 are confusing

Two deployments tied together (author & publish)

Rolling both at the same time

In the future may consider a Helm rollback

ANALYSIS TEMPLATE

Combining 6 metrics:

Error ratio in stable vs canary

Number of errors in canary

Number of requests in both stable and canary

Combine two deployments together

ANALYSIS TEMPLATE USING AI

Easily implemented using canary analysis jobs

THE CUSTOMER VIEW

Differentiating customer triggered vs internal

Avoid confusing external users with no feedback

Rolling back changes without user feedback can create confusion

Be mindful of users who have strict requirements for fast deployments

BENEFITS

Automatic roll back on high error rates

Non blocking rollouts across environments and async investigation

Reduced blast radius

BENEFITS

Only a percentage of traffic is affected temporarily

More frequent releases

Validate with real traffic

More velocity

CHALLENGES

Migration requires orchestration to avoid downtime

A problem with 1000s of services

Better with `workloadRef` and `scaleDown` attribute

CHALLENGES

Scale `Deployment` down to zero and reference it from the `Rollout` with `workloadRef` field.

`scaleDown` attribute: `never`, `onsuccess`, `progressively`

MIGRATION

feat: automatically scale down Deployment after migrating to Rollout #3111

Merged **zachaller** merged 2 commits into `argoproj:master` from `balasoiuroxana:deployments-scale-down` on Dec 5, 2023

Conversation 15 | Commits 2 | Checks 22 | Files changed 18

balasoiuroxana commented on Oct 17, 2023 • edited by zachaller — Contributor

fixes: #2748

Checklist:

- [x] Either (a) I've created an enhancement proposal and discussed it with the community, (b) this is a bug fix, or (c) this is a chore.
- [x] The title of the PR is (a) conventional with a list of types and scopes found here, (b) states what changed, and (c) suffixes the related issues number. E.g. `"fix(controller): Updates such and such. Fixes #1234"`.
- [x] I've signed my commits with DCO
- [x] I have written unit and/or e2e tests for my change. PRs without these are unlikely to be merged.
- [x] My builds are green. Try syncing with master if they are not.
- [ ] My organization is added to USERS.md.

👍 2

Reviewers
zachaller
kostis-codefresh

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone

argo-rollouts PR #3111

CHALLENGES

Start with simple rollouts, watch for degraded status

- prometheus is not reachable
- upgrades with object deletion

IMMUTABLE RESOURCES

Immutable `ConfigMap` and `Secret` are a solution for high load in the API server

Each change to them needs a new name (typically including a checksum of content)

ie. `mysecret-abcde`

Example:

- Helm upgrade
- new secret is created `mysecret-abcde1`
- old secret is deleted `mysecret-abcde0`
- new pods fail to start per Rollout config

- Argo does a rollback to previous deployment
- more pods of the old deployment cannot be created as old secret no longer exists
- outage as soon as existing pods are recycled

METRICS

Figuring out the correct metrics

Metrics need to account for `canary`/`stable` labels

Recognize low-traffic environments

Identify environments already experiencing errors

Check errors in multiple deployments if they work together

```
prometheus:
  query: |
    label_replace(
      avg(request_error_ratio_5m{pod_label_role="stable", aem_tier=~"author", aem_service=~"{{args.aem_service}}"} >=0) or vector(0),
      "metric", "request_error_ratio_stable_author", "", ""
    )
    or
    label_replace(
      avg(request_error_ratio_5m{pod_label_role="stable", aem_tier=~"publish", aem_service=~"{{args.aem_service}}"} >=0) or vector(0),
      "metric", "request_error_ratio_stable_publish", "", ""
    )
    or
    label_replace(
      avg(request_error_ratio_5m{pod_label_role="canary", aem_tier=~"author", aem_service=~"{{args.aem_service}}"} >=0) or vector(0),
      "metric", "request_error_ratio_canary_author", "", ""
    )
    or
    label_replace(
      avg(request_error_ratio_5m{pod_label_role="canary", aem_tier=~"publish", aem_service=~"{{args.aem_service}}"} >=0) or vector(0),
      "metric", "request_error_ratio_canary_publish", "", ""
    )
    or
    label_replace(
      sum(increase(red_error{pod_ready="true", pod_label_role="canary", aem_tier=~"author|publish", aem_service=~"{{args.aem_service}}", url=""}[5m])) or vector(0),
      "metric", "red_error_count", "", ""
    )
    or
    label_replace(
      sum(increase(red_rate{pod_ready="true", pod_label_role="canary", aem_tier=~"author|publish", aem_service=~"{{args.aem_service}}", url=""}[5m])) or vector(0),
      "metric", "red_requests_count", "", ""
    )
```

STEPS

Setting the steps correctly

Short steps may not catch issues

Long pauses or many steps can significantly increase deployment duration

FALSE POSITIVES / NEGATIVES

Review number of Rollouts that are not promoted

Fix and iterate

HANDLING FAILURES

Look for degraded rollouts: InvalidSpec, timeout (replicaset fails to be ready), error, abort

COSTS

Increase in cost for the added safety

Progressive Delivery is a great idea

Argo Rollouts is a great implementation

Some things to iron out and prepare for

balasoiuroxana

carlossg

balasoiuroxana

csanchez