APACHE SLING & FRIENDS TECH MEETUP

2 - 4 SEPTEMBER 2019

Deep-dive into cloud-native AEM deployments based on Kubernetes

Tomek Rękawek, Adobe

Cloud: what and why?

adaptTo()

We want to move from

to

Cloud: why?

- Running AEM at scale
- Hassle-free deployments
- The cloud provider (AWS, Azure) should worry about infrastructure

Agenda

- Docker & Kubernetes introduction
- Architecture overview
- Publish persistence
  - Cloud Segment Store
  - Golden publish
  - Compaction
- Sidecar services
- Jobs
  - Content migration
  - New indexes
- QA

Docker & Kubernetes introduction

adaptTo()

The power of standardization

Dockerized AEM

- AEM in Docker image
- Composite Node Store
  - /apps & /libs stored in the container
  - Actual content lives outside, in the VOLUME or MongoDB
- OSGi Feature Model
  - Defines AEM and customer application
  - Feature launcher starts it inside container
  - Covered in Karl & David's presentation
- See adaptTo() 2017 talk for more details

Mini intro to Kubernetes

- Launches Docker containers without worrying about the underlying VMs
- Dictionary

  - Pod – a group of containers starting together on a single machine

  - Service – internal load balancer, exposing a number of pod replicas under a single address

  - Ingress – exposes a service under an public http address

- Every entity is represented by a YAML object in K8s API server
- The YAML is the desired state, K8s knows how to get there

Deployments with Helm

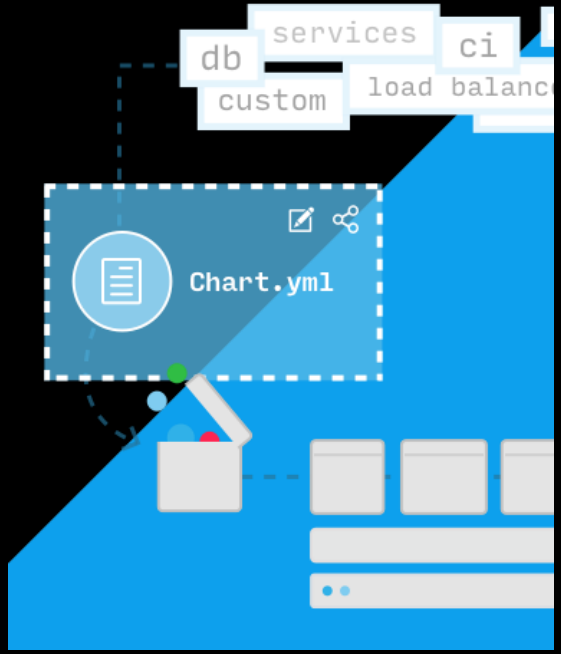


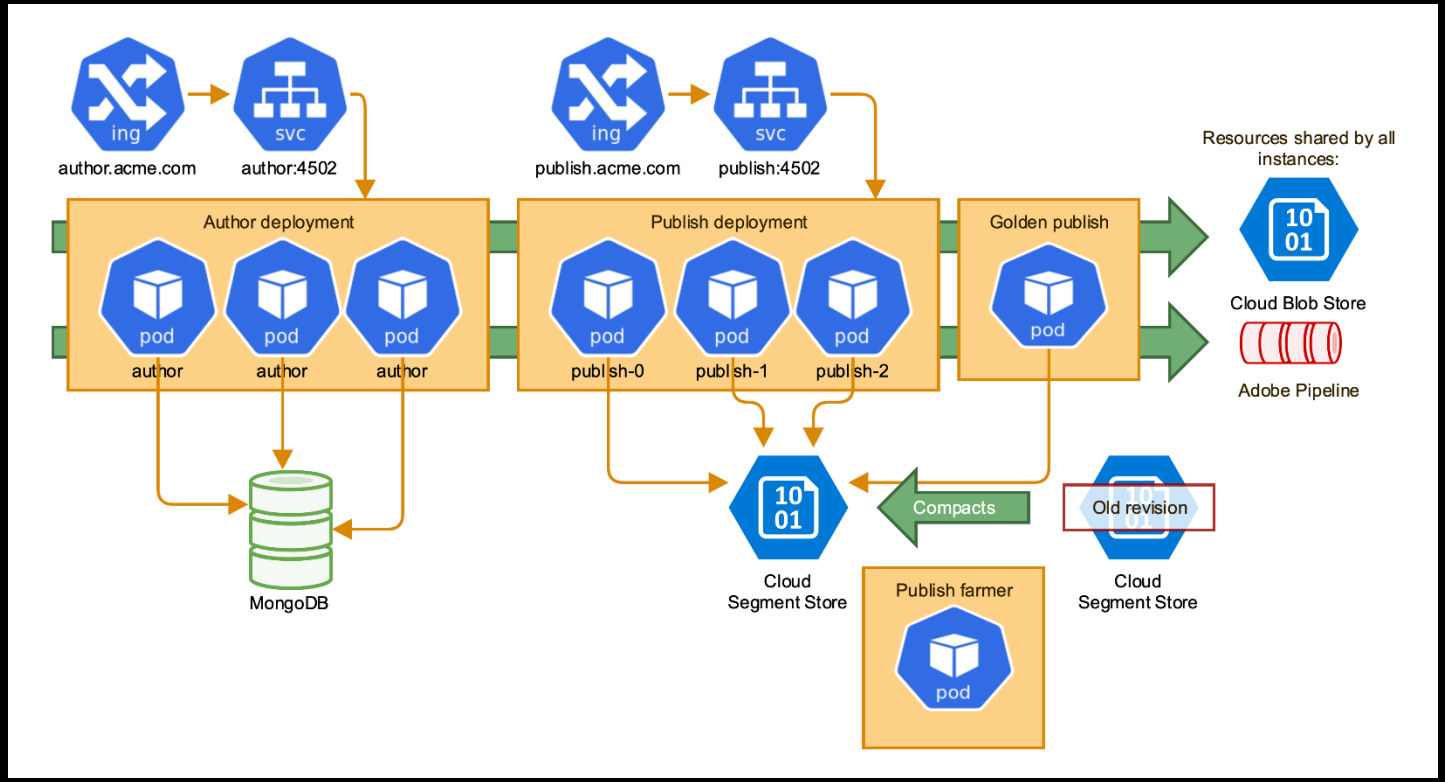
Image: https://helm.sh

- Helm – K8s apps management
- Chart – a bunch of K8s YAML descriptors, with a simple templating
- Whole AEM deployment can be installed/upgraded with a single command

adaptTo()

Architecture overview

AEM Kubernetes setup



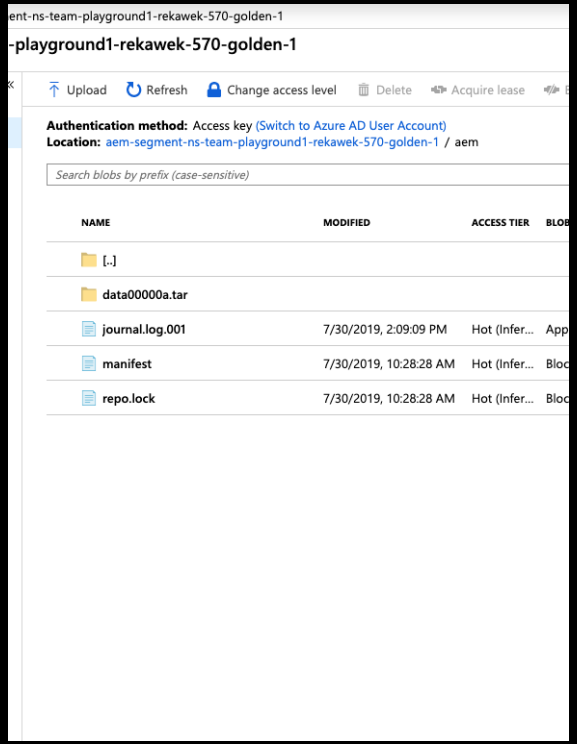
Publish persistence evolution

adaptTo()

Problem definition

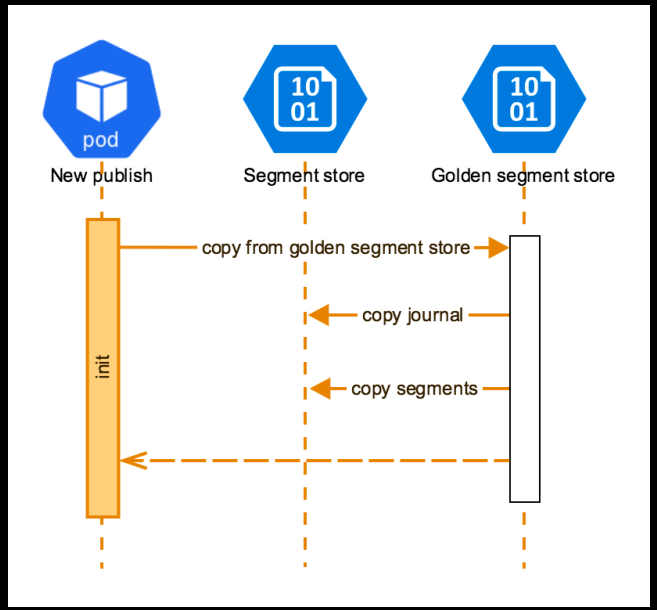
- Author persistence is easy-ish with MongoDB
- Publish is harder – local SegmentMK, no clustering
- The publish farm is kept up-to-date with replication
- However:
  - we need to provide the new publish instances with a segment store,
  - copy it from another instance.
- Problems:
  - copying files between pods is hard and hacky,
  - what if there's no publish to copy from?

Cloud Segment Store



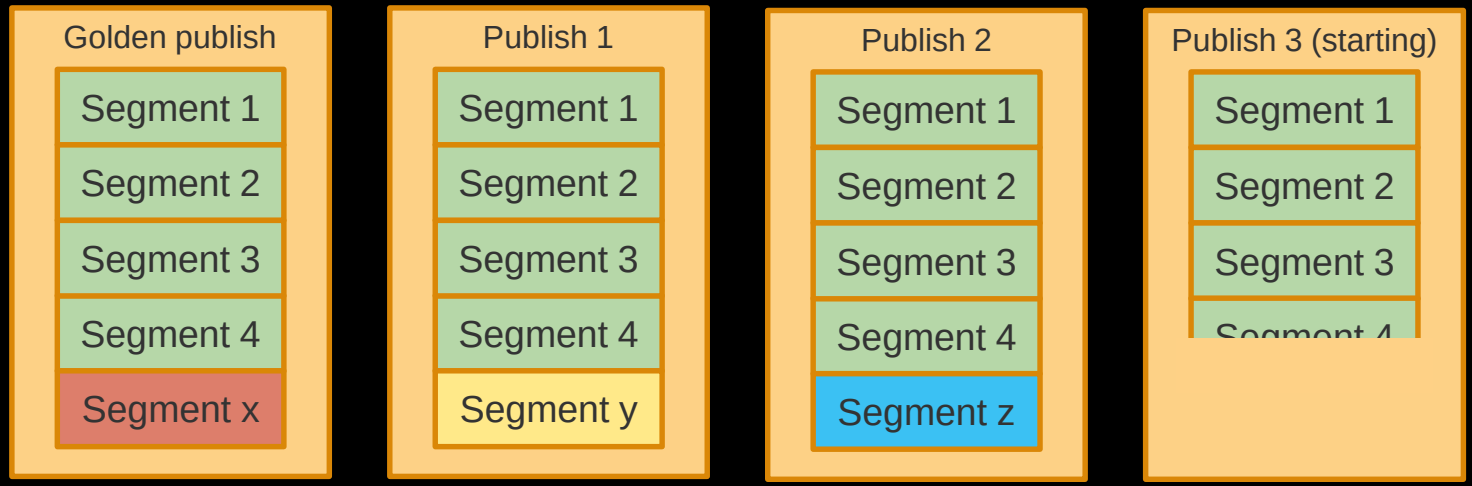
- A new plugin for the Segment Node Store
- Nodes are stored in a cloud storage service
- No tar files, raw segments grouped in dirs
- Can be used in RW or RO modes

Golden publish



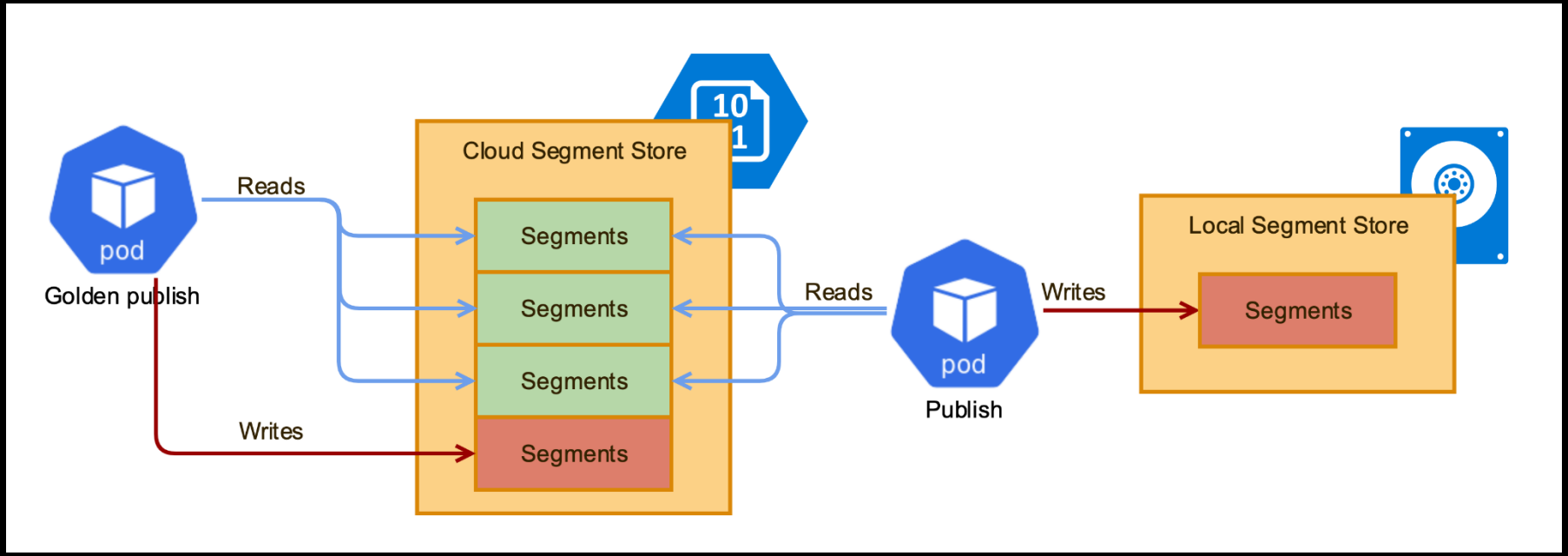
- A designated publish instance
- Not connected to LB
- It maintains a “golden copy” of the segment store
- New publish just clone it

Problem: duplicated binaries and startup time



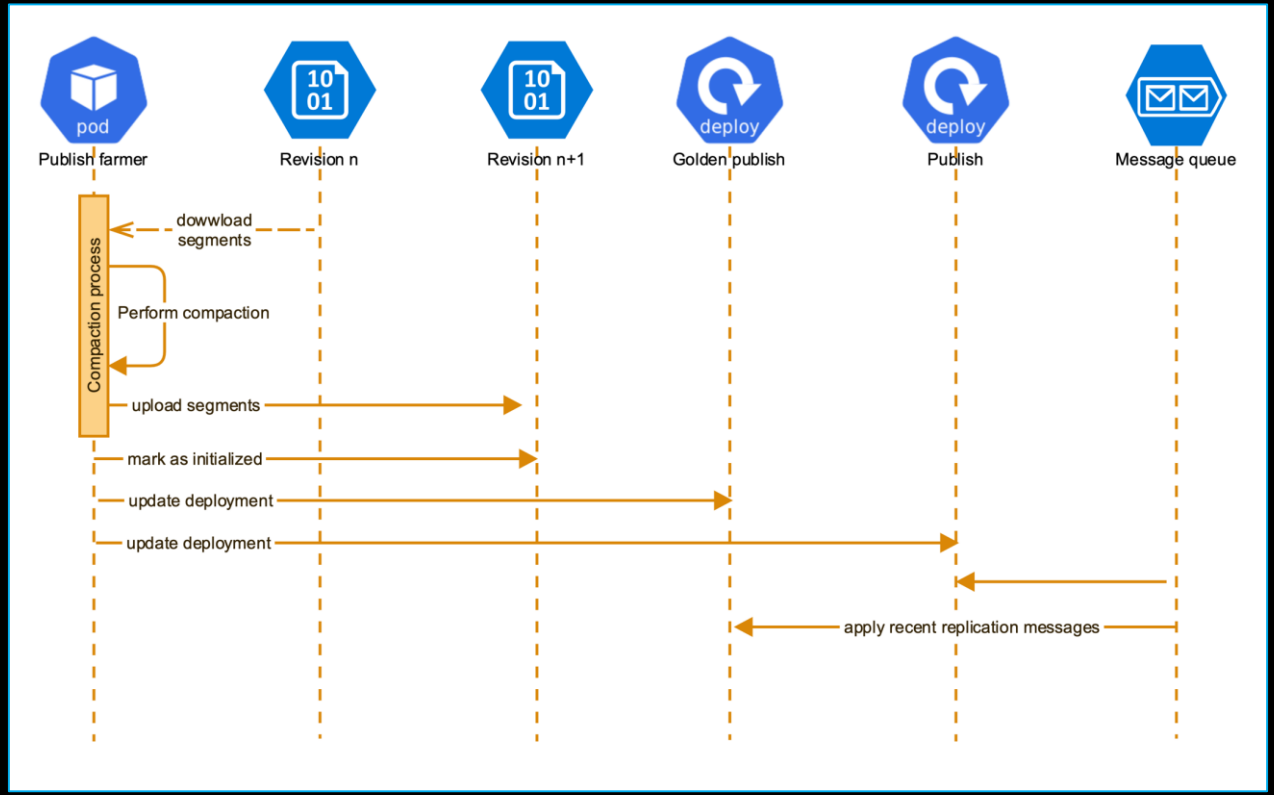
- Multiple copies of the same segments 1-4 ($$$)
- Cloning a bucket takes time during the publish start (🕒🕒🕒)

Optimization: a single segment store

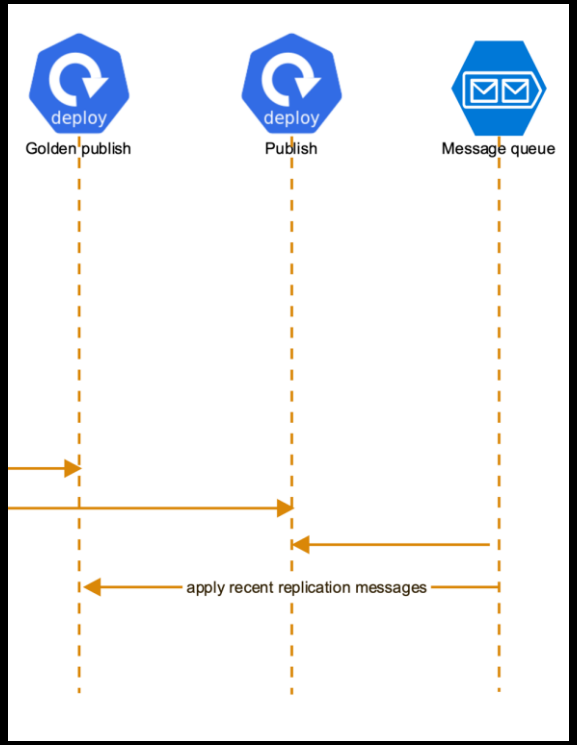


Publish persistence: compaction

Compaction


pod
Publish farmer
10
01
Revision n
10
01
Revision n+1
deploy
Golden publish
deploy
Publish
Message queue
Compaction process
dowwload
segments
Perform compaction
upload segments
mark as initialized
update deployment
update deployment
apply recent replication messages

Out-of-band publish update



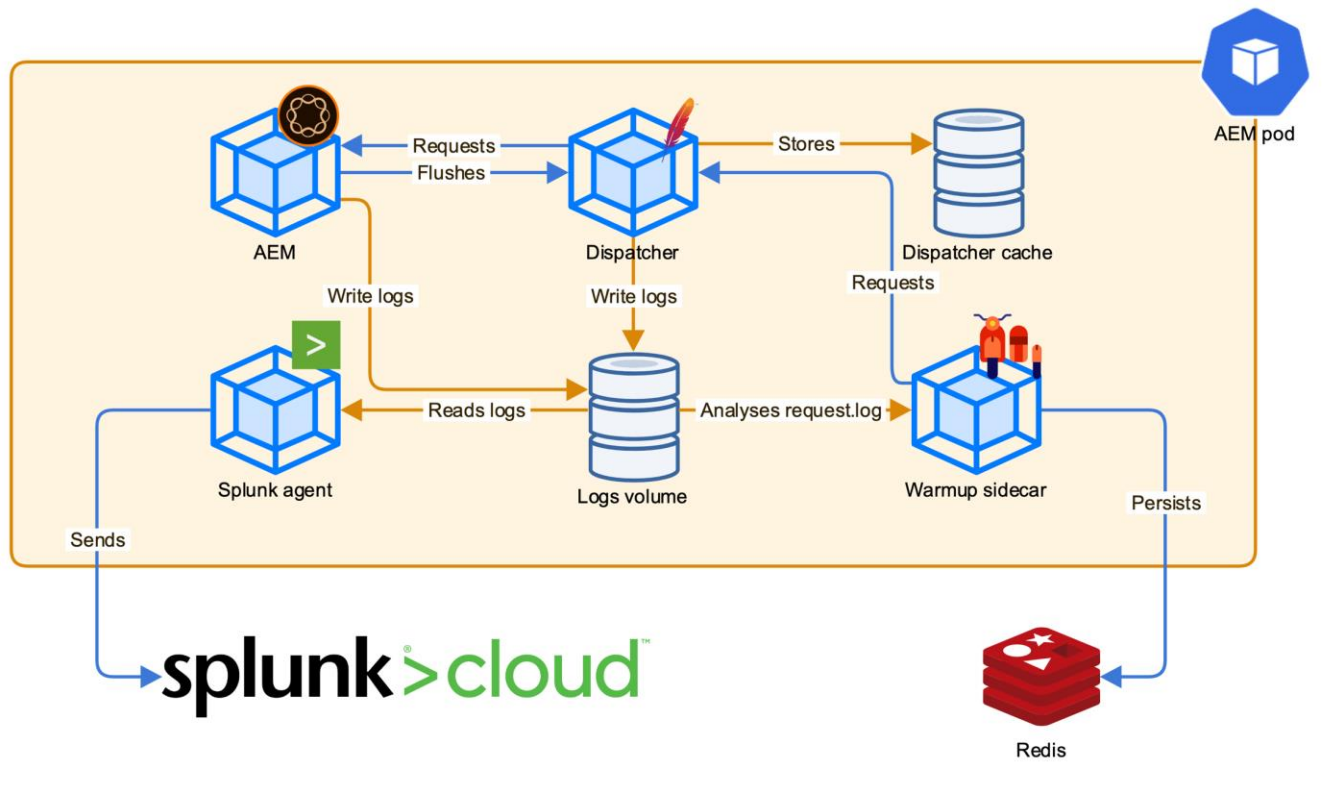
- This pattern will be useful in many cases
- We may clone the publish repository, modify it and re-deploy instances on top of it
- Persisted message queue will apply missing changes

Sidecar services

Sidecar approach

- A single pod can run many containers, sharing their volumes and localhost interface
- We can use them for the auxiliary services (sidecars):
  - Dispatcher in the publish pod
  - Upload logs to Splunk
  - Warmup service

AEM pod with sidecars



Jobs

Kubernetes job

- Starts a pod
- Meant to perform a specific task and finish
  - Unlike the deployment, which run indefinitely
- Will be restarted if fails
- Jobs provides a way to interact with the deployed AEM

adaptTo()

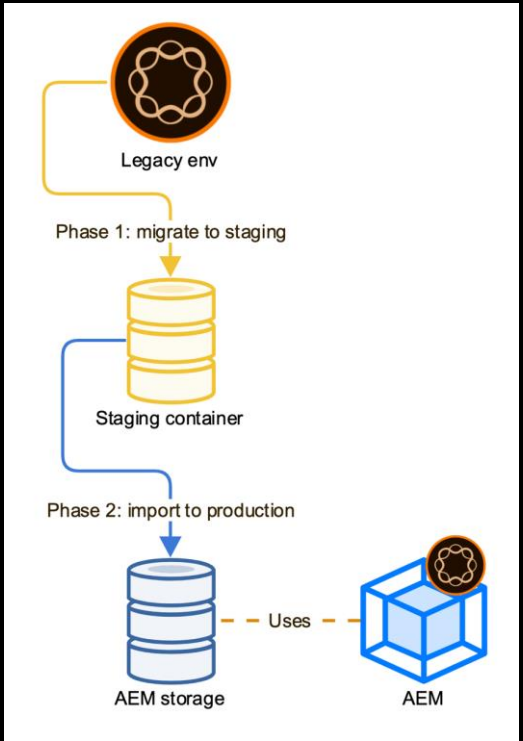
Content migration

adaptTo()

Content migration

- How to migrate old content to K8s?
- Access problem
  - old AEM envs shouldn’t have access to the K8s (encapsulation)
  - K8s shouldn’t be able to access old AEMs (they can be installed anywhere)
- Solution: demilitarized zone

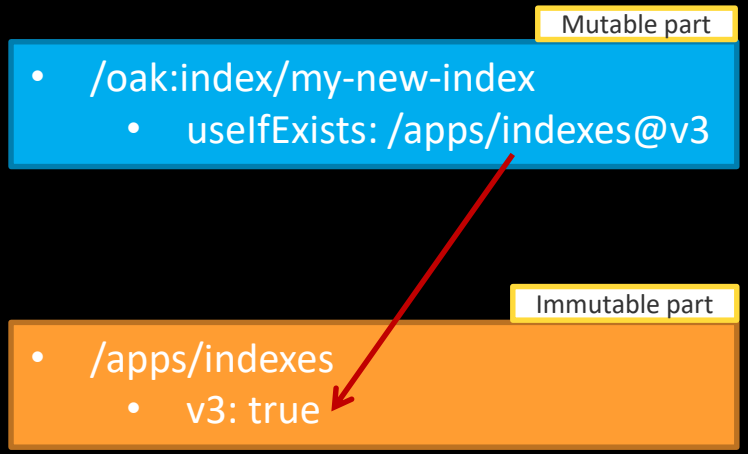
2-phase migration



- Phase 1
  - The migrator tool (crx2oak-like jar) is used to export old AEM content into a cloud storage service
- Phase 2
  - A Kubernetes job is used to apply the migrated content on the cloud instances

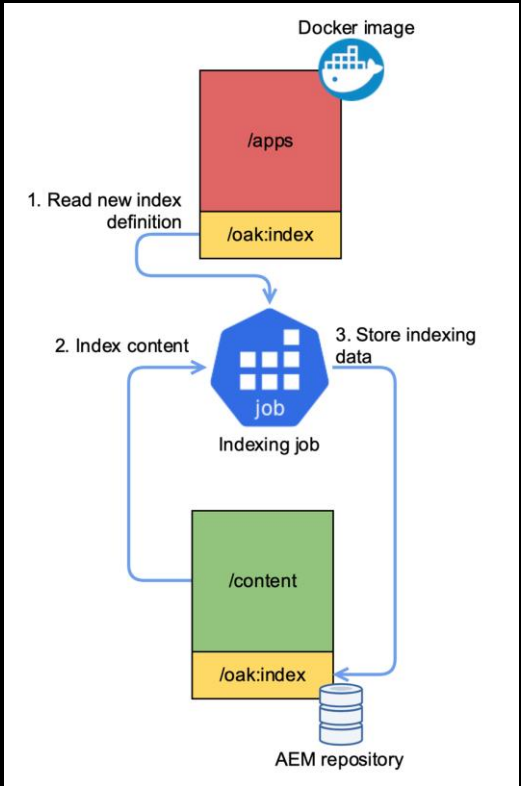
New indexes

Adding new index



- Indexes are tricky
  - /oak:index is a part of the mutable content
  - But index definitions belongs to the application
- Only adding new indexes is supported
- When a new index is added in /oak:index, it'll have an extra useIfExists property referencing immutable part /apps
  - This bounds the index definition to the application version and Docker image
- This /apps path have to be added as well

Mutable content: adding new index



- Indexing job should be run before the actual app deployment
- The job will:
  1. Look for the new index defs in /oak:index
  2. Perform out of band indexing of the content
  3. Save the new indexing content to the production repository
- The useIfExists will make sure that the new index is ignored, until the new image is installed
- When the new image is deployed, the /apps part will be updated and the new index will be used

Other topics

Related topics

- Feature model usage in Docker (covered in David's and Karl's talk)
- Replication (covered in Timothee's talk)
- Monitoring with Prometheus and Grafana
- CI/CD pipeline
- Network policies
- …

Thanks!