Agenda

- Community news
- Towards a sustainable deployment
Resources

Alfresco

Deploying Alfresco Outlook Transform Engine with Docker Compose @ GitHub
Script Object for RenditionService2 @ GitHub and thanks to @hi-ko
Alfresco GenAI @ GitHub

Resources to come

• Adapting your logging configuration to log4jv2
• How to set up messages for increased reliability
• Upgrading your addons to Jakarta EE 10 and Spring 6
• Upgrading to Apache Tomcat 10
• Using Control Center App with Community Edition
• Deploying Alfresco with Helm in Kubernetes
Collaboration

Blog posts

• Summarization of textual content in Alfresco repository with Amazon Bedrock by @abhinavmishra14

Contributions

• https://github.com/aborroy/alfresco-dockerx-builder/issues/11 by @uvukasinovic
• https://github.com/Alfresco/alfresco-docker-installer/issues/185 by @luca86r
• https://github.com/Alfresco/alfresco-docker-installer/issues/186 by @N00BTellaBrot

Conferences
TTL Speakers wanted!

- Take the opportunity to showcase your work with the community
- About Alfresco, Nuxeo, and associated technologies
- Best practices, integration, scaling, cloud, …
- In your native language
Today’s talk
Towards a sustainable deployment

Alexandre Chapellon and Giovanni Toraldo, Hyland
TTL #156
Towards a Sustainable deployment

Alexandre Chapellon, Valerio Provaggi, Giovanni Toraldo

March 22, 2024
Digital transformation has the potential to mitigate climate impacts.
So basically is just a matter of consuming less resources?
A bit more complex than that

Efficiency

Awareness
A bit more complex than that

Experience Design
Application Design
Application Deployment
Infrastructure
Agenda

- Helm charts modularity
- Graviton deployment
- Next steps
Helm charts modularity
Why modularity?

- Deploy only what you need and save resources
- Makes Alfresco platform composable
- More generic means of configuration, hence more flexibility
- Gives more freedom in choosing dependencies for 3rd party components
Comparison

**alfresco-content-services**
- CORE:
  - repository
  - sync-service
  - transform-service
- UI
  - share
- SEARCH
  - search-service

**digital-workspace**
**control-center**
**ai-transformer**
**connector-ms365**
**connector-msteams**
**repository**
**search-enterprise**
**search-service**
**share**
**sync-service**
**transform-service**

Must deploy at least all of
Can deploy any of
Components as building blocks for alfresco chart or your own chart
Creating your own chart is just plumbing

- ConfigMaps to share configuration parameters across charts (**alfresco-global.properties** is back)

- Secrets to share credentials

- **alfresco-common** library chart brings helpers to ease creation of ConfigMap entries

- No third-party dependencies with component charts
... dependencies:
- name: postgresql
  repository: oci://registry-1.docker.io/bitnamicharts
  version: 13.4.0
- name: activemq
  repository: https://alfresco.github.io/alfresco-helm-charts/
  version: 3.4.1
- name: alfresco-repository
  repository: https://alfresco.github.io/alfresco-helm-charts/
  version: 0.1.3
- name: alfresco-transform-service
  version: 1.2.0
  repository: https://alfresco.github.io/alfresco-helm-charts/
- name: my-custom-search-service
  repository: https://charts.partner.com/alfresco/
  version: 1.0.0
Example ConfigMap (ActiveMQ)

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: message-broker
data:
  {{- with .Values.activemq }}
  MQ_URL: {{ include "alfresco-common.activemq.url.withFailover" (printf "nio://%s-broker:%v" (include "activemq.fullname" $mqCtx) 61616) | quote }}
  {{- end }}
```

**Principles:**
- Build the subchart context ($mqCtx)
- Create a ConfigMap item to pass the expected information the target chart expects (Check individual README.md)
- Leverage common helpers (alfresco-common.activemq.url.withFailover) & source charts' named templates (activemq.fullname) to reliably build the ActiveMQ URL
Example ConfigMap (ActiveMQ)

```yaml
alfresco-repository:
  configuration:
    messageBroker:
      existingConfigMap:
        name: message-broker
        keys:
          url: MQ_URL
      existingSecret:
        name: message-broker
        keys:
          username: MQ_USER
          password: MQ_PASS
```

- **Principles:**
  - Leverage configuration values as expected by the target chart (Check individual README.md) to point to the right key in the right ConfigMap.
There's more than one way to do it
More details:

acs-sso-example available in acs-deployment with step-by-step documentation
Standardized means of configuration

- `extraVolumes` & `extraVolumeMounts`
  - Can be used to mount additional files from ConfigMaps (e.g. config files, keystores)

- `extraInitContainers` & `extraSideContainers`
  - Preliminary actions
  - System integration

- `extraEnv`
  - Additional config options through environment variables

- `nodeSelector`
- `tolerations`
- `affinity`
- `podLabels` & `podAnnotations`
Compositions examples

- alfresco-content-services
  - Repository
  - Share
  - Search-Enterprise
  - Digital-workspace
  - Transform-service
  - Sync-service

- Repository
- Transform-service
- Search-enterprise

- Repository
- Transform-service
- Search-Enterprise
- OpenSearch Cluster
- ActiveMQ broker

- Repository (self hosted)
- Database (self hosted)
- ActiveMQ (self hosted)
- Transform-service
- Search service (custom chart)
- Intelligence-service
Charts Repositories

• New git repository: https://github.com/Alfresco/alfresco-helm-charts
• New chart registry: https://alfresco.github.io/alfresco-helm-charts/
• Each chart now has its own documentation
  o Main README.md for values
  o /docs/* for specific use cases
• Charts are tested & released using standard Helm tooling on KinD
  o Chart Tester (ct) & Chart Releaser (cr)
• Semver:
  o Major bump: expect breaking changes in architecture or configuration interface
  o Minor bump: new features or non-breaking changes
  o Patch bump: bugfix only
• Deployments tend to be sized to cope with the peak workload
• Autoscaling can help lowering the resources impact & costs
• Alfresco repository and Alfresco Transform Service good first candidates
• CPU based scaling & investigate other metrics
Graviton deployment
Docker multi-arch support requirement

- Alfresco ACS 23.2 includes multi-arch docker images (almost)
- Check if a specific image is multi-arch:

```bash
$ docker buildx imagetools inspect quay.io/alfresco/alfresco-content-repository:23.2.1
```

- Multi-arch docker image can be built easily with:

```bash
$ docker buildx build --platform linux/amd64,linux/arm64 -t my-namespace/my-image:latest
```
Graviton testbed architecture

- Single EKS cluster
  - Graviton (arm64) node group with t4g.xlarge
    - 4 burstable vCPUs, 16.0 GiB of memory and up to 5 Gibps of bandwidth, starting at $0.1344 per hour - $96.76 per month per node
  - Classic (amd64) node group with t3a.xlarge
    - 4 burstable vCPUs, 16.0 GiB of memory and up to 5 Gibps of bandwidth, starting at $0.1504 per hour - $108.28 per month per node

- Helm install of Alfresco against each node group
  - (Deployment would fit 2x nodes each with room for scaling up)
Provision EKS cluster with graviton nodes

- Deploy with Terraform EKS module

```typescript
[..]
  eks_managed_node_groups = {
    graviton = {
      min_size = 0
      max_size = 2
      desired_size = 1

      instance_types = ["t4g.xlarge"]
      ami_type = "AL2_ARM_64"
    }
  }
[..]
```
Cluster Autoscaler

Automatically adjust the desired size of node groups so that all pods have a place to run and there are no unneeded nodes.

• Runs as a deployment within the cluster
• Should have permissions to examine and modify EC2 Auto Scaling Groups
• Do not require additional configuration except for fine tuning

Links:
• https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/cloudprovider/aws/README.md
• https://github.com/lablabs/terraform-aws-eks-cluster-autoscaler
**nodeSelector to target nodeGroup**

- nodeSelector is a Pod spec to tell K8s scheduler to run only on nodes that match that constraint
- Helm charts allow specifying a nodeSelector via values for each component:

```yaml
[..]
alfresco-repository:
  nodeSelector:
    "kubernetes.io/arch": arm64
share:
  nodeSelector:
    "kubernetes.io/arch": arm64
[..]
```
Install alfresco on EKS

- Values oriented for testing purposes only:
  - No persistency
  - 1 replicas

$ helm upgrade --install acs alfresco/alfresco-content-services --version ~8.0.0
  --namespace alfresco
  --set global.search.sharedSecret=$(openssl rand -hex 24)
  --values helm-install/acs_values.yml
  --values helm-install/arm64_selector.yml
alfresco-repository:
ingress:
  annotations:
    cert-manager.io/cluster-issuer: letsencrypt-prod
persistence:
  enabled: false
configuration:
  search:
    flavor: elasticsearch
[..]
alfresco-repository:
    nodeSelector:
      "kubernetes.io/arch": amd64
    ingress:
      hosts:
        - host: FQDN
          paths:
            - path: /
              pathType: Prefix
      tls:
        hosts:
          - FQDN
          secretName: letsencrypt-prod-hydra

Sources at https://github.com/gionn/alfresco-eks-graviton-example
<table>
<thead>
<tr>
<th>State</th>
<th>Name</th>
<th>Roles</th>
<th>Version</th>
<th>External/Internal IP</th>
<th>OS</th>
<th>CPU</th>
<th>RAM</th>
<th>Pods</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>ip-172-16-8-200.eu-west-1.compute.internal</td>
<td>Worker</td>
<td>v1.27.7-eks-e71965b</td>
<td>- / 172.16.8.200</td>
<td>Linux</td>
<td>83%</td>
<td>34%</td>
<td>21%</td>
<td>2.2 hours</td>
</tr>
<tr>
<td>Labels: eks.amazonaws.com/capacityType=ON_DEMAND</td>
<td>eks.amazonaws.com/nodegroup=classic-202312211012140632000000001</td>
<td>eks.amazonaws.com/nodegroup-image=ami-0086c4f3f1b824d82</td>
<td>eks.amazonaws.com/sourceLaunchTemplateVersion=1</td>
<td>kfls.ik존Provider-aws-d7ab6c90e13c2dc866e6564b6549a</td>
<td>topologyvts.csi.aws.com/zones/eu-west-1a</td>
<td>eks.amazonaws.com/sourceLaunchTemplateId-ft-03861ae3200048875</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Active | ip-172-16-15-126.eu-west-1.compute.internal | Worker | v1.27.7-eks-e71965b | - / 172.16.15.126 | Linux | 95% | 43% | 59% | 6 days |
| Labels: eks.amazonaws.com/capacityType=ON_DEMAND | eks.amazonaws.com/nodegroup=graviton-20231214173855942001200000001 | eks.amazonaws.com/nodegroup-image=ami-0ff1888b3bcb4e12d | eks.amazonaws.com/sourceLaunchTemplateVersion=1 | kfls.ik존Provider-aws-d7ab6c90e13c2dc866e6564b6549a | topologyvts.csi.aws.com/zones/eu-west-1a | eks.amazonaws.com/sourceLaunchTemplateId-ft-0339aa6d944261655 |

| Active | ip-172-16-27-199.eu-west-1.compute.internal | Worker | v1.27.7-eks-e71965b | - / 172.16.27.199 | Linux | 98% | 45% | 29% | 2.2 hours |
| Labels: eks.amazonaws.com/capacityType=ON_DEMAND | eks.amazonaws.com/nodegroup=graviton-20231214173855942001200000001 | eks.amazonaws.com/nodegroup-image=ami-0ff1888b3bcb4e12d | eks.amazonaws.com/sourceLaunchTemplateVersion=1 | kfls.ik존Provider-aws-d7ab6c90e13c2dc866e6564b6549a | topologyvts.csi.aws.com/zones/eu-west-1a | eks.amazonaws.com/sourceLaunchTemplateId-ft-0339aa6d944261655 |

| Active | ip-172-16-38-251.eu-west-1.compute.internal | Worker | v1.27.7-eks-e71965b | - / 172.16.38.251 | Linux | 83% | 33% | 24% | 2.2 hours |
| Labels: eks.amazonaws.com/capacityType=ON_DEMAND | eks.amazonaws.com/nodegroup=classic-202312211012140632000000001 | eks.amazonaws.com/nodegroup-image=ami-0086c4f3f1b824d82 | eks.amazonaws.com/sourceLaunchTemplateVersion=1 | kfls.ik존Provider-aws-d7ab6c90e13c2dc866e6564b6549a | topologyvts.csi.aws.com/zones/eu-west-1a | eks.amazonaws.com/sourceLaunchTemplateId-ft-03861ae3200048875 |
Deployment/statefulset overview

$ kubectl get deploy -n alfresco

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>UP-TO-DATE</th>
<th>AVAILABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>acs-activemq</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-alfresco-cc</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-alfresco-dw</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-alfresco-repository</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-alfresco-search-enterprise-content</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-alfresco-search-enterprise-metadata</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-alfresco-search-enterprise-path</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-filestore</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-imagemagick</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-libreoffice</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-pdfrenderer</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-share</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-tika</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-transform-misc</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acs-transform-router</td>
<td>1/1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$ kubectl -n alfresco get statefulset

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
</tr>
</thead>
<tbody>
<tr>
<td>acs-postgresql-acs</td>
<td>1/1</td>
</tr>
<tr>
<td>alfresco-search-enterprise-mediation</td>
<td>1/1</td>
</tr>
<tr>
<td>elasticsearch-master</td>
<td>1/1</td>
</tr>
</tbody>
</table>
Verify deployment with DTAS

- Deployment Test Automation Scripts (DTAS) is an internal (not yet opensource) pytest suite to verify the correctness of an Alfresco install
- API testing
  - content lifecycle: create user, create a new site, create a folder structure, uploading documents
  - Alfresco Transformation Service (ATS): requesting renditions for various document types such as docx, xlsx, pdf, odt, jpeg, and png
- Basic benchmarking using pytest-benchmark library
  - pytest suites executed multiple times within a configurable time frame or number of rounds
Run DTAS via helm charts on Alfresco Enterprise

- Helm install as usual plus `--set dtas.enabled=true`
- Run the suite via `helm test RELEASE`

NAME: acs
LAST DEPLOYED: Mon Mar 11 17:28:56 2024
NAMESPACE: alfresco
STATUS: deployed
REVISION: 1
TEST SUITE: dtas-hbusb
Last Started: Mon Mar 11 17:36:58 2024
Last Completed: Mon Mar 11 17:49:34 2024
Phase: Succeeded
CPU/Memory overview (arm64 vs amd64)
Network bandwidth overview (arm64 vs amd64)
## Benchmark results (arm64 vs amd64)

<table>
<thead>
<tr>
<th>Name (time in ms)</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Name (time in ms)</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_post_person_performance</td>
<td>793.1918</td>
<td>1,086.6737</td>
<td>841.8284</td>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>1,011.1304</td>
<td>1,234.0511</td>
<td>1,116.0950</td>
</tr>
<tr>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>899.8614</td>
<td>1,068.4193</td>
<td>925.2980</td>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>1,018.7815</td>
<td>1,655.8906</td>
<td>1,184.0600</td>
</tr>
<tr>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>901.5965</td>
<td>1,031.4085</td>
<td>926.1439</td>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>1,877.6391</td>
<td>1,606.7332</td>
<td>1,194.7080</td>
</tr>
<tr>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>901.7826</td>
<td>1,195.2217</td>
<td>932.4559</td>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>1,878.6347</td>
<td>1,452.8360</td>
<td>1,223.1594</td>
</tr>
<tr>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>934.2809</td>
<td>1,079.4848</td>
<td>969.5077</td>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>1,995.7999</td>
<td>1,354.1743</td>
<td>1,266.4598</td>
</tr>
<tr>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>1,152.6486</td>
<td>1,344.4858</td>
<td>1,190.3074</td>
<td>test_redirects_performance[dtas.docx-impreview]</td>
<td>1,400.5495</td>
<td>1,817.3643</td>
<td>1,575.0191</td>
</tr>
</tbody>
</table>
Key takeaway

• Alfresco can run smoothly on arm64
• EKS cluster can be configured easily to run arm64 nodes
• Performances/latency is slightly better on AWS Graviton2 instances vs cost-near t3a instances
  • Increased cost efficiency
• Migrating to arm64 reduce workloads energy consumption
  • AWS claim up to 60 percent for the same performance of comparable EC2
Next Steps
Alfresco roadmap

Efficiency and Awareness lead to cost savings

Share and create ideas
Thank you