Altair Engineering, Inc. makes no representation or warranties with respect to the contents of this manual or the associated software and especially disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Altair Engineering, Inc. reserves the right to revise this publication and make changes from time to time to its contents without obligation to notify anyone of such revisions or changes.

Altair Knowledge Hub Enterprise Server software is offered and is to be used in accordance with a SOFTWARE LICENSE AND MAINTENANCE AGREEMENT. This agreement stipulates that this software be used only in the computer system designated in that agreement. The agreement further stipulates that the customer shall not copy or alter, or permit others to copy or alter, the software or related materials in whole or in part, in any media for any purpose, except to make an archive (back-up) copy or to make a copy as an essential step in the use of the software with the customer's computer.

Altair Engineering, Inc. hereby grants the buyer the right to reprint this documentation for internal uses only. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, for any other purposes, without the prior written permission of Altair Engineering, Inc.

Altair Knowledge Hub Enterprise Server is a trademark of Altair Engineering, Inc. Other products mentioned herein may be trademarks or registered trademarks of their respective owners in the United States or other countries.

For U.S. Government End Users, the software is a “Commercial Item(s),” as that term is defined at 48 C.F.R. Section 2.101, consisting of “Commercial Computer Software” and “Commercial Computer Software Documentation,” as such terms are used in 48 C.F.R. Section 12.212 or 48 C.F.R. Section 227.7202, as applicable. Consistent with 48 C.F.R. Section 12.212 or 48 C.F.R. Sections 227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the Altair Software License and Maintenance Agreement.

SALES INFORMATION
US: + 1.800.445.3311
International: + 1.978.441.2200

Sales Email
US: sales@datawatch.com
Europe: sales_euro@datawatch.com
Asia Pacific: sales_apac@datawatch.com

SUPPORT CONTACT INFORMATION
Customer Portal: https://support.datawatch.com
Email: support@datawatch.com
US: +1 800.988.4739
International: +1 978.275.8350

2.3.2/2019.08.23
# Table of Contents

Altair Knowledge Hub Enterprise Server Installation Guide ........................................ 1

Components .......................................................................................................................... 1
  Knowledge Hub Enterprise Server Cluster components .................................................. 2
  Deployment Diagram ....................................................................................................... 4
Multitenancy Approach ...................................................................................................... 5
Horizontal Scalability Approach ........................................................................................ 6
  Scalability Per Component .............................................................................................. 6
Benefits of Kubernetes ...................................................................................................... 8

Pre-Installation Procedures ............................................................................................... 9

Amazon Elastic Container Service for Kubernetes (EKS) .................................................... 9
  Manual Setup with CloudFormation Templates .......................................................... 9
  Configuring VPN .......................................................................................................... 9
  Knowledge Hub Prerequisite Configuration for Amazon EKS .................................... 10
On-Prem Cluster Using Kubespray .................................................................................... 19
  Configuring kubectl ...................................................................................................... 21
  Knowledge Hub Prerequisite Configuration for on-Prem Kubespray .......................... 21
Google Compute Engine (GCE) ........................................................................................ 27

Knowledge Hub Helm Deployment .................................................................................... 30

Prerequisites ....................................................................................................................... 30
Setup Knowledge Hub ....................................................................................................... 31
Installing JDBC Drivers ...................................................................................................... 33

Spring Configuration ......................................................................................................... 34

  Core-API Properties ..................................................................................................... 35
  Data-Engine-API Properties ......................................................................................... 37
  Social-API Properties .................................................................................................. 40
  License-API Properties ................................................................................................. 41
Setting Up LDAP Authentication ....................................................................................... 42
Setting Up SSO Authentication ......................................................................................... 44

Deleting Knowledge Hub Enterprise .................................................................................. 51

Deleting Knowledge Hub Enterprise Server ..................................................................... 51
Deleting Modules .............................................................................................................. 51
Deleting the Tracer (Jaeger) ........................................................................ 51
Deleting the logger (ELK Stack) ................................................................. 52
Removing Monitoring (Grafana) .............................................................. 52
Utilities Configuration ............................................................................. 52

Elastic Log Export/Import for Knowledge Hub Enterprise ....................... 54
Exporting Elasticsearch to AWS S3 .......................................................... 54
Importing Logs in k8s ELK ..................................................................... 55

Migrating (Upgrading) Windows Installations to a Cluster ....................... 56
Backing Up/Restoring Knowledge Hub ..................................................... 56
  Backing Up Windows Installations ......................................................... 56
  Backing Up Linux Single-Server Installations ....................................... 57
  Restoring Backed Up Installations via Kubernetes ............................... 57
Migrating Cipher Keys from Windows to Enterprise Server Installations .... 58

YAML File Templates ............................................................................. 59
  eks-vpc.yaml ..................................................................................... 59
  eks-nodes.yaml ................................................................................. 65
  eks-vpn.yaml .................................................................................... 75
  elasticsearch-export-job.yaml ............................................................ 78
Altair Knowledge Hub
Enterprise Server Installation Guide

This document will guide you through the steps of installing Altair Knowledge Hub Enterprise Server.

Components

Knowledge Hub Enterprise Server includes several components:

- The Knowledge Hub Enterprise Server application
- ELK Stack for logging
- Kubernetes Dashboard for system administration

Note that Knowledge Hub can be deployed to a private subnet for cloud deployments:

- The Knowledge Hub web UI can be deployed to a public internet or private subnet
- The Kubernetes API/Dashboard is only accessible through bastion instances or VPN

Helm is used to install and update server and Knowledge Hub Enterprise application. Knowledge Hub Enterprise server is built on a Kubernetes cluster. You can use the command line tool `kubectl` to deploy and manage applications in Kubernetes.
Using **kubectl**, you can:

- Inspect cluster resources
- Create, delete and update components
- Look at your new cluster
- Bring up example apps

These components are packaged in [Helm Package Manager](https://helm.sh) for Kubernetes with custom configurations.

## KNOWLEDGE HUB ENTERPRISE SERVER CLUSTER COMPONENTS

The following cluster components are included in Knowledge Hub Enterprise Server:

- **license-api** – [Spring Boot 2](https://spring.io) application
  - Used as external licensing microservice
- **ingress** – [Nginx](https://nginx.org) controller for the Knowledge Hub Enterprise server application
  - Ingress for Knowledge Hub clusters
  - The only component that is visible to the outer world
  - Used for TLS termination
- **core http-api** – [Spring Boot 2](https://spring.io) application
  - Serves all Knowledge Hub core functionalities (HTTP and WebSocket endpoints)
  - Swarm library (workspaces, data sources, connections, change lists, folders, etc.)
  - User management (users, roles, groups)
  - Process management (processes, schedules, etc.)
  - Data readers (CSV, excel, JDBC, etc.)
  - Data writers (CSV, excel, JDBC, etc.)
  - Report trapping
  - Data fetching and querying (design mode, batch mode, statistics)
- **core api postgres** – [PostgreSQL](https://www.postgresql.org) server
  - Used to store core api metadata objects
- **data-engine http-api** – [Spring Boot 2](https://spring.io) application
- **data-engine api postgres** – [PostgreSQL](https://www.postgresql.org) server
  - Used to store data-engine api metadata objects
- **data-engine worker** – *Spring Boot 2* application
  - Data Engine for design mode

- **data-engine postgres** – *PostgreSQL* server with *PostgreSQL PL/Java*
  - Data engine backend used to store data sources data and query workspaces relational tree for design mode

- **data-engine batch** – *Spring Boot 2* application and *PostgreSQL* server with *PostgreSQL PL/Java*
  - Data Engine for batch requests
  - Data engine backend used to store data sources data and query workspaces relational tree for export

- **machine-learning http-api** – *Spring Boot 2* application
  - Machine learning recommendations and social functionality (likes, follows, etc.) of the Knowledge Hub cluster

- **Distributed cache** – *Redis* server
  - Used in the core http-api distributed mode as a *hibernate l2 cache* (metadata storage), *spring cache* (data source preview)
  - Used in data-engine http-api distributed mode for *distributed locks, maps, counters* (data-engine).

- **Spark cluster** – *Apache Spark* analytics engine
  - Spark driver for machine-learning recommendations
  - Spark drivers for data-engine batch mode

- **Cassandra cluster** – *Apache Cassandra* database
  - Used as backend storage to import legacy workspaces from Monarch Complete
  - Used as backend storage for published data sources (internal library)
  - Used as backend storage for machine-learning http-api
  - Used as backend storage for machine-learning recommendations

- **rabbitmq server** – *RabbitMQ* message broker
  - Used for asynchronous request for batch and design mode requests
Multitenancy Approach

Our current multitenancy approach is highly based on virtualization with the aim of reducing costs for supporting both single-tenant and multi-tenant versions of the Knowledge Hub Enterprise Server application. Such an approach gives us relatively zero-development to support this functionality (with the help of Kubernetes namespaces).

The following approach is taken:

- Each tenant has its own virtual cluster (i.e., all of the components on the deployment diagram)
- All of the tenants' clusters are isolated between each other (i.e., each tenant has its own database, possibly its own codebase to allow some customization between tenants, etc.)
- Each tenant has own resource quotas (CPU, memory, storage) in the cluster
- Each tenant can scale components in cluster independently from other tenants (which allows more flexibility in setting up optimal resources for a particular tenant workload)
- The tenant administrator has full access to manage tenants (create, delete, scale, define resource quotas, etc.)

![Multitenancy Diagram]

Additional Information

- [Namespaces in Kubernetes](#)
- [Resource quotas in the namespace](#)
Horizontal Scalability Approach

Each component in the cluster can be scaled with known limitations.

Two types of scalability are supported:

- Manual scaling: driven by cluster/tenant administrator
- Automatic scaling: based on resource consumption (e.g., CPU) or specific metrics (e.g., request rate)

The scalability approach differs for stateful and stateless components:

- Stateful components (databases): Distributed cache, data-engine batch, cassandra cluster, rabbitmq-ha

Spark is a unique component that already has some clustered solutions with the scalability approach:

- [https://github.com/apache-spark-on-k8s/spark](https://github.com/apache-spark-on-k8s/spark)

**SCALABILITY PER COMPONENT**

- **license-api** — is not scalable by design
- **ingress** — Nginx is scalable by design
- **core http-api** — This component is stateless by design, so it can be scaled with some limitations
  - core api postgres connections limit
  - core api postgres throughput
- **core api postgres** — metadata storage for core http-api; this component is not scalable
- **data-engine http-api** — this component is stateless by design so can be scaled with some limitations:
  - data-engine postgres connections limit
  - data-engine postgres throughput
- **data-engine api postgres** — metadata storage for data-engine http-api, this component is not scalable
- **data-engine worker** — data-engine for design-mode requests is stateless and scalable by design
- **data-engine postgres** — this component is not scalable by design
- **data-engine batch** — data-engine for batch process can be scaled
- **machine-learning http-api** — is not scalable by design
- **spark cluster** — Spark is scalable by design
- **cassandra cluster** — Cassandra is scalable by design
- **rabbitmq-ha** - message broker is scalable by design
- **distributed cache** — Redis is scalable by design

**Additional Information**

- [https://kubernetes.io/docs/concepts/workloads/controllers/deployment/](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)
- [https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/](https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/)
- [https://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling](https://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling)
- [https://redis.io/topics/cluster-spec](https://redis.io/topics/cluster-spec)
- [https://www.nginx.com/blog/inside-nginx-how-we-designed-for-performance-scale/](https://www.nginx.com/blog/inside-nginx-how-we-designed-for-performance-scale/)
- [https://kubernetes.io/docs/tutorials/stateful-application/cassandra/](https://kubernetes.io/docs/tutorials/stateful-application/cassandra/)

Kubernetes deployments/statefulset with scaling support:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Support Scaling</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>license-api</td>
<td>stateless</td>
<td>false</td>
<td>-</td>
</tr>
<tr>
<td>core-api</td>
<td>stateless</td>
<td>true</td>
<td>-</td>
</tr>
<tr>
<td>core-api-postgres</td>
<td>stateless</td>
<td>false</td>
<td>core-api database</td>
</tr>
<tr>
<td>data-engine-api</td>
<td>stateless</td>
<td>true</td>
<td>Should be scaled on each node in the cluster</td>
</tr>
<tr>
<td>data-engine-api-postgres</td>
<td>stateless</td>
<td>false</td>
<td>data-engine-api database</td>
</tr>
<tr>
<td>data-engine-worker</td>
<td>stateless</td>
<td>true</td>
<td>Should be scaled on each node in the cluster</td>
</tr>
<tr>
<td>data-engine-batch</td>
<td>stateful</td>
<td>true</td>
<td>Should be scaled on each node in the cluster</td>
</tr>
<tr>
<td>rabbitmq-ha</td>
<td>stateful</td>
<td>true</td>
<td>Should be scaled on each node in the cluster</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Support</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>---------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>machine-learning</td>
<td>stateless</td>
<td>false</td>
<td>–</td>
</tr>
<tr>
<td>depostgres</td>
<td>stateless</td>
<td>false</td>
<td>Data engine for design mode, work with data-engine-worker.</td>
</tr>
<tr>
<td>cassandra</td>
<td>stateful</td>
<td>false</td>
<td>Support replication</td>
</tr>
<tr>
<td>redis</td>
<td>stateful</td>
<td>false</td>
<td>Support master-slave replication with persistence on file system</td>
</tr>
</tbody>
</table>

**Benefits of Kubernetes**

*Kubernetes (k8s)* is used as a cluster container orchestrator for:

- Automatic deployment
- Horizontal scaling
- Multitenancy (*namespace*-based)
- Storage orchestration
- Release management (*helm*-based)
- Self-healing
- Service discovery and load balancing
- Secret and configuration management
Pre-Installation Procedures

Amazon Elastic Container Service for Kubernetes (EKS)

The following steps describe how to set up Amazon EKS Cluster and Knowledge Hub Enterprise with custom modules.

MANUAL SETUP WITH CLOUDFORMATION TEMPLATES

Use the link https://docs.aws.amazon.com/eks/latest/userguide/getting-started.html to set up Amazon EKS.

Set up Virtual Private Cloud (VPC) by using eks-vpc.yaml.

Set up the Kubernetes worker nodes using eks-nodes.yaml.

Use the PrivateSubnets output from the first template as the Subnets parameter.

CONFIGURING VPN

To set up a simple VPN server, use the eks-vpn.yaml CloudFormation template. The VPN endpoint, username, and password are listed in CloudFormation Stack outputs.
KNOWLEDGE HUB PREREQUISITE CONFIGURATION FOR AMAZON EKS

This section describes how to set up several modules from the Knowledge Hub Enterprise installer via helm. Note that Amazon EKS must be properly set up to configure these modules successfully.

Requirements:

- Kubernetes 10.x version
- Helm version ≥ 2.12
- Configured kubectl for Amazon EKS cluster

Configuring helm on the Cluster

Home URL: https://github.com/helm/helm/releases/tag/v2.12.3

Commands:

```bash
kubectl apply -f tiller.yaml
helm init --service-account tiller --upgrade --wait
```

`tiller.yaml`

```yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller-clusterrolebinding
subjects:
  - kind: ServiceAccount
    name: tiller
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: cluster-admin
```
Configuring External Ingress

**REQUIRED**


Command:

```
helm upgrade --install --namespace kube-system --values nginx_ingress_values.yaml --version 1.0.1 --wait --timeout 600 lb stable/nginx-ingress
```

**nginx_ingress_values.yaml**

```
nameOverride: lb
controller:
  config:
    proxy-body-size: 2048m
  publishService:
    enabled: true
  service:
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-backend-protocol: tcp
      service.beta.kubernetes.io/aws-load-balancer-connection-idle-timeout: 1800
      service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled: true
      service.beta.kubernetes.io/aws-load-balancer-ssl-ports:
        https
    targetPorts:
      http: http
      https: https
    type: LoadBalancer
  load_balancer_type: aws-single
```
Configuring EFS Provisioner

**REQUIRED**


Command:

```
helm upgrade --install --namespace kube-system --values efs_provisioner_values.yaml --version 0.1.1 --wait --timeout 600 efs-provisioner stable/efs-provisioner
```

**efs_provisioner_values.yaml**

```
global:
  deployEnv: prod

image:
  tag: v1.0.0-k8s1.10

efsProvisioner:
  efsFileSystemId: <efs_fsid>
  awsRegion: <efs_aws_region>
  path: /
  storageClass:
    name: default-nfs
    isDefault: false
    reclaimPolicy: Retain
```

Configuring External DNS

**REQUIRED if Route53 Hosted Zone is used**


Command:

```
helm upgrade --install --namespace kube-system --values external_dns_values.yaml --version 0.7.6 --wait --timeout 600 external-dns stable/external-dns
```
**external_dns_values.yaml**

```yaml
domainFilters: []
nameOverride: external-dns
policy: upsert-only
provider: aws
publishInternalServices: true
rbac:
  create: true
sources:
  - ingress
```

**Configuring Internal ingress**

**REQUIRED if Optional modules are used**


Command:

```
helm upgrade --install --namespace kube-system --values internal_ingress_values.yaml --version 1.0.1 --wait --timeout 600 lb-internal stable/nginx-ingress
```

**internal_ingress_values.yaml**

```yaml
controller:
  ingressClass: internal
config:
  proxy-body-size: 2048m
publishService:
  enabled: true
service:
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-internal: 0.0.0.0/0
    service.beta.kubernetes.io/aws-load-balancer-backend-protocol: tcp
    service.beta.kubernetes.io/aws-load-balancer-connection-idle-timeout: '1800'
    service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled: 'true'
    service.beta.kubernetes.io/aws-load-balancer-ssl-ports: https
targetPorts:
  http: http
```
Configuring Kubernetes Dashboard

**OPTIONAL**


Generate certificates for the domain or for dashboard URL only (tls.key, tls.crt) and then create a secret using these certificates:

```bash
kubectl create secret tls dashboard-tls-cert -n kube-system --key tls.key --cert tls.crt
```

Configure basic authorization:

```bash
kubectl create secret generic ops-auth -n kube-system --from-literal=username='USERNAME' --from-literal=password='PASSWORD'
```

Execute command:

```bash
helm upgrade --install --namespace kube-system -f dashboard_values.yaml --force --wait dashboard stable/kubernetes-dashboard
```

dashboard_values.yaml

```yaml
fullnameOverride: kubernetes-dashboard
image:
  tag: v1.8.3
ingress:
  annotations:
    kubernetes.io/ingress.class: internal
    nginx.ingress.kubernetes.io/auth-realm: Authentication Required!
    nginx.ingress.kubernetes.io/auth-secret: ops-auth
    nginx.ingress.kubernetes.io/auth-type: basic
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
    nginx.ingress.kubernetes.io/rewrite-target: /
  enabled: true
  hosts:
  - <dashboard url>
  path: /
```
ELK Charts

**OPTIONAL**

Home URL: - (available in Knowledge Hub Enterprise archive)

Create namespace for logging:

```bash
kubectl create ns logging
```

Generate certificates for domain or for Kibana URL only (tls.key, tls.crt) and create secret using these certificates:

```bash
kubectl create secret tls logs-tls-cert -n logging --key tls.key --cert tls.crt
```

Configure basic authorization; you must set your own username and password:

```bash
kubectl create secret generic ops-auth -n logging --from-literal=username='USERNAME' --from-literal=password='PASSWORD'
```

Command:

```bash
helm upgrade --install --namespace logging -f logging_values.yaml --wait --timeout 600 elk ./elk/elk-*tgz
```

Define hostname `<host>` values in yaml file:

```
logging_values.yaml

elasticsearch:
  data:
    persistence:
      size: 120Gi

kibana:
  ingress:
    annotations:
      kubernetes.io/ingress.class: internal
      kubernetes.io/tls-acme: "true"
```
NFS

**OPTIONAL**


Command:

```
helm upgrade --install --namespace kube-system --values nfs_values.yaml --version 0.2.1 --wait --timeout 600 nfs stable/nfs-server-provisioner
```

<table>
<thead>
<tr>
<th>nfs_values.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>nameOverride: nfs</td>
</tr>
<tr>
<td>persistence:</td>
</tr>
<tr>
<td>enabled: true</td>
</tr>
<tr>
<td>size: 150Gi</td>
</tr>
<tr>
<td>storageClass:</td>
</tr>
<tr>
<td>storageClass:</td>
</tr>
<tr>
<td>name: default-nfs</td>
</tr>
<tr>
<td>reclaimPolicy: Retain</td>
</tr>
</tbody>
</table>

Heapster

**OPTIONAL**


Command:

```
helm upgrade --install --namespace kube-system -f heapster_values.yaml --force --wait --timeout 600 heapster stable/heapster
```
**Monitoring**

**OPTIONAL**

Home URL: (available in Knowledge Hub Enterprise archive)

Create namespace for logging:

```shell
kubectl create ns monitoring
```

Generate certificates for domain or for Grafana URL only (tls.key, tls.crt) and then create secret using these certificates:

```shell
kubectl create secret tls logs-tls-cert --namespace monitoring
```

Configure basic authorization, please set your own username and password:

```shell
kubectl create secret generic ops-auth --namespace monitoring
```

Command:

```
helm upgrade --install --namespace monitoring -f monitoring_values.yaml
```

Define hostname `<host>` values in yaml file:

```yaml
grafana:
  env:
    GF_SERVER_ROOT_URL: <host>
  ingress:
    annotations:
```
kubernetes.io/ingress.class: internal
kubernetes.io/tls-acme: "true"
nginx.ingress.kubernetes.io/auth-realm: Authentication Required!
  nginx.ingress.kubernetes.io/auth-secret: ops-auth
  nginx.ingress.kubernetes.io/auth-type: basic

enabled: true
hosts:
  - <host>
path: /
tls:
  - hosts:
    - <host>
  secretName: monitoring-tls-cert

**Autoscaler**

**OPTIONAL**


Command:

run kubectl

```
values.yaml
```

nameOverride: cluster-autoscaler

cloudProvider: aws
dacloudRegion: <aws region>
autoDiscovery:
  autoDiscovery:
    clusterName: <eks cluster name>

sslCertPath: /etc/ssl/certs/ca-bundle.crt

extraArgs:
  - skip-nodes-with-system-pods: 'false'
  - skip-nodes-with-local-storage: 'false'

rbac:
  create: true
  pspEnabled: true

resources:
requests:
  cpu: 0.1
  memory: 300Mi
limits:
  cpu: 0.3
  memory: 600Mi

On-Prem Cluster Using Kubespray

Download latest version of Kubespray from https://github.com/kubernetes-sigs/kubespray/releases

Make sure that your system and Kubernetes cluster meet Kubespray requirements.

The following instructions are based on the Kubespray documentation: https://github.com/kubernetes-sigs/kubespray/blob/master/docs/getting-started.md

Instructions

After downloading Kubespray, perform the following actions inside the Kubespray directory:

```
pip install -r requirements.txt
cp -r inventory/sample inventory/mycluster
```

Make the following changes:

- Change `dashboard_enabled: true` to `dashboard_enabled: false` in inventory/mycluster/group_vars/k8s-cluster/addons.yml. kubernetes-dashboard will be installed later using helm chart.

- Specify the kubernetes version in inventory/mycluster/group_vars/k8s-cluster/k8s-cluster.yml via `kube_version` variable. A list of supported kubespray versions may be found in roles/download/defaults/main.yml.

- Change other variables in inventory/mycluster/group_vars if needed. Documentation on these variables may be found in https://github.com/kubernetes-sigs/kubespray/blob/master/docs/vars.md

- Configure the inventory/mycluster/host.ini file.

```
[all]
## Configure 'ip' variable to bind kubernetes services
## on a different ip than the default iface set to 'ansible_host'
```
node1 ansible_host=<ip> etcd_member_name=etcd1 # ip=<private ip>
node2 ansible_host=<ip> etcd_member_name=etcd2 # ip=<private ip>
node3 ansible_host=<ip> etcd_member_name=etcd3 # ip=<private ip>

[kube-master]
node1
node2
node3

[etcd]
node1
node2
node3

[kube-node]
node1
node2
node3

[k8s-cluster:children]
kube-master
kube-node

[all:vars]
kubeconfig_localhost=true

Additional instructions for modifying the host.ini file may be found at
https://github.com/kubernetes-sigs/kubespray/blob/master/docs/ansible.md

Run Kubespray playbook:
```
ansible-playbook -i inventory/mycluster/hosts.ini cluster.yml
-b -v --private-key=<path to key> -u <remote user>
```
CONFIGURING KUBECTL

Accessing Kubernetes API

For public IP configuration, copy the file `inventory/mycluster/artifacts/admin.conf` to `~/.kube/config`. Doing so will overwrite existing configuration.

KNOWLEDGE HUB PREREQUISITE CONFIGURATION FOR ON-PREM KUBESPRAIY

This section describes how to set up several modules from the Knowledge Hub Enterprise installer via helm. Note that Kubespray must be properly set up to configure these modules successfully.

Requirements:

- Kubernetes 10.x version
- Helm version >= 2.12
- Configured `kubectl` for On-Prem cluster

Configuring helm on the Cluster

Home URL: [https://github.com/helm/helm/releases/tag/v2.12.3](https://github.com/helm/helm/releases/tag/v2.12.3)

Commands:

```bash
cubectl apply -f tiller.yaml
helm init --service-account tiller --upgrade --wait
```

tiller.yaml

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
```
kind: ClusterRoleBinding
metadata:
  name: tiller-clusterrolebinding
subjects:
  - kind: ServiceAccount
    name: tiller
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: ''

Configuring External Ingress

REQUIRED


Command:

helm upgrade --install --namespace kube-system --values nginx_ingress_values.yaml --version 1.0.1 --wait --timeout 600 lb stable/nginx-ingress

<table>
<thead>
<tr>
<th>nginx_ingress_values.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>nameOverride: lb</td>
</tr>
<tr>
<td>controller:</td>
</tr>
<tr>
<td>kind: DaemonSet</td>
</tr>
<tr>
<td>hostNetwork: true</td>
</tr>
<tr>
<td>config:</td>
</tr>
<tr>
<td>proxy-body-size: 2048m</td>
</tr>
<tr>
<td>publishService:</td>
</tr>
<tr>
<td>enabled: true</td>
</tr>
<tr>
<td>service:</td>
</tr>
<tr>
<td>targetPorts:</td>
</tr>
<tr>
<td>http: http</td>
</tr>
<tr>
<td>https: https</td>
</tr>
<tr>
<td>type: ClusterIP</td>
</tr>
</tbody>
</table>
Configuring NFS

**REQUIRED**


Command:

```
helm upgrade --install --namespace kube-system --values nfs_values.yaml --version 0.2.1 --wait --timeout 600 nfs stable/nfs-server-provisioner
```

```
nfs_values.yaml
```

```
nameOverride: nfs
persistence:
  enabled: true
  size: 150Gi # adjust if needed
  storageClass: ""
storageClass:
  name: default-nfs
reclaimPolicy: Retain
```

Configuring Kubernetes Dashboard

**OPTIONAL**


Generate `tls.key` and `tls.crt` certificates for domain `<host>` (e.g.: dashboard-2.aws.dev-altair.com) and upload them to the cluster:

```
kubectl create secret tls dashboard-tls-cert -n kube-system -key tls.key --cert tls.crt
```

Configure basic authorization:

```
kubectl create secret generic ops-auth -n kube-system --from-literal=username='USERNAME' --from-literal=password='PASSWORD'
```

Execute command:

```
helm upgrade --install --namespace kube-system -f dashboard_values.yaml --force --wait dashboard stable/kubernetes-dashboard
```
**dashboard_values.yaml**

```
fullnameOverride: kubernetes-dashboard
image:
  tag: v1.8.3
ingress:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/auth-realm: Authentication Required!
    nginx.ingress.kubernetes.io/auth-secret: ops-auth
    nginx.ingress.kubernetes.io/auth-type: basic
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
    nginx.ingress.kubernetes.io/rewrite-target: /
  enabled: true
  hosts:
    - <host>
  path: /
  tls:
    hosts:
      - <host>
      secretName: dashboard-tls-cert
```

**Configuring ELK Charts**

**OPTIONAL**

Home URL: Knowledge Hub Enterprise archive

Create namespace for logging:

```
kubectl create ns logging
```

Generate certificates for hostname "<host>" tls.key, tls.crt and upload in the cluster:

```
kubectl create secret tls logs-tls-cert -n logging --key tls.key --cert tls.crt
```

Configure basic authorization, please set your own username and password:

```
kubectl create secret generic ops-auth -n logging --from-literal=USERNAME='USERNAME' --from-literal=password='PASSWORD'
```
Command:

```
helm upgrade --install --namespace logging -f logging_values.yaml --wait --timeout 600 elk ./helm/charts/elk-*-tgz
```

Define hostname `<host>` values in the yaml file, for example: logs-2.aws.dev-datawatch.com

```
logging_values.yaml

```

```yaml
elasticsearch:
  data:
    persistence:
      size: 120Gi # adjust if needed
  kibana:
    ingress:
      annotations:
        kubernetes.io/ingress.class: nginx
        kubernetes.io/tls-acme: "true"
        nginx.ingress.kubernetes.io/auth-realm: Authentication Required!
        nginx.ingress.kubernetes.io/auth-secret: ops-auth
        nginx.ingress.kubernetes.io/auth-type: basic
      enabled: true
      hosts:
        - `<host>`
      tls:
        - hosts:
          - `<host>`
        secretName: logs-tls-cert
```

Configuring Heapster

**OPTIONAL**


Command:

```
helm upgrade --install --namespace kube-system -f heapster_values.yaml --force --wait --timeout 600 heapster stable/heapster
```
heapster_values.yaml.

```yaml
image:
  repository: k8s.gcr.io/heapster-amd64
  tag: v1.5.3
rbac:
  create: true
resizer:
  enabled: false
```

Configuring Monitoring

**OPTIONAL**

Home URL: Knowledge Hub Enterprise archive

Create namespace for logging:

```
kubectl create ns monitoring
```

Generate tls.key and tls.crt certificates for hostname "<host>" and upload to the cluster:

```
kubectl create secret tls logs-tls-cert -n monitoring --key tls.key --cert tls.crt
```

Configure basic authorization, please set your own username and password:

```
kubectl create secret generic ops-auth -n monitoring --from-literal=username='USERNAME' --from-literal=password='PASSWORD'
```

Command:

```
helm upgrade --install --namespace monitoring -f monitoring_values.yaml --wait --timeout 600 monitoring ./helm/charts/monitoring-*.tgz
```

Define hostname <host>, for example: monitoring-2.aws.dev-altair.com, values in the yaml file

monitoring_values.yaml

```yaml
grafana:
  env:
    GF_SERVER_ROOT_URL: https://<host>
  ingress:
```
Google Compute Engine (GCE)

Set up GCE by using the official documentation.
This section describes how to set up modules from the Knowledge Hub Enterprise installer.

Requirements

- Kubernetes 10.x version
- Helm version >= 2.12
- Configured kubectl for GCE

Configuring helm on the Cluster

Home URL: https://github.com/helm/helm/releases/tag/v2.12.3

Commands:
kubectl apply -f tiller.yaml
helm init --service-account tiller --upgrade -wait
**tiller.yaml**

```yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller-clusterrolebinding
subjects:
  - kind: ServiceAccount
    name: tiller
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: ''
```

**Configuring Nginx-ingress**

**REQUIRED**


Command:

```
helm upgrade --install --namespace kube-system --values nginx_ingress_values.yaml --version 1.0.1 --wait --timeout 600 lb stable/nginx-ingress
```

**nginx_ingress_values.yaml**

```yaml
nameOverride: lb
controller:
  config:
    proxy-body-size: 2048m
publishService:
  enabled: true
  service:
    type: LoadBalancer
```
To get **EXTERNAL-IP**, use the command:

```
kubectl get services lb-controller -n kube-system
```

and update **hosts** file with

```
<EXTERNAL-IP> <ingress_host from values.yaml>
```

Configuring NFS

**OPTIONAL**


Command:

```
helm upgrade --install --namespace kube-system --values nfs_values.yaml --version 0.2.1 --wait --timeout 600 nfs stable/nfs-server-provisioner
```

<table>
<thead>
<tr>
<th><strong>nfs_values.yaml</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>nameOverride: nfs</td>
</tr>
<tr>
<td>persistence:</td>
</tr>
<tr>
<td>enabled: true</td>
</tr>
<tr>
<td>size: 150Gi</td>
</tr>
<tr>
<td>storageClass: &quot;&quot;</td>
</tr>
<tr>
<td>storageClass:</td>
</tr>
<tr>
<td>name: default-nfs</td>
</tr>
<tr>
<td>reclaimPolicy: Retain</td>
</tr>
</tbody>
</table>

Configuring `kubectl` for GCE

**Steps:**

2. Open [https://console.cloud.google.com/home/dashboard](https://console.cloud.google.com/home/dashboard) in the browser and log in to your project.
3. Open the Linux console and run `gcloud auth login`.
4. Copy and open the output link in the browser.
5. Get the verification code and paste it in the console.
6. Get `kubectl` context using `gcloud container clusters get-credentials <cluster name> --zone <zone id> --project <project name>`.
7. Get nodes using `kubectl get nodes` to verify that the correct context is applied.
Knowledge Hub Helm Deployment

Configure Kubernetes using any of the configurations detailed in the previous section. Then, configure and install modules from the Knowledge Hub prerequisites.

Prerequisites

Server:
- Kubernetes 10+ version
- Helm version 2.12+
  
  If you have an incompatible version of the client and helm server, please follow the documentation to upgrade tiller
- 2 ReadWriteMany persistence volumes or storage class with ReadWriteMany option (e.g.,: for Amazon EKS - EFS Provisioner, ... )
- 5+ ReadWriteOnce persistence volumes or storage class with ReadWriteOnce option (e.g.: for Amazon EKS - gp2 provisioner kubernetes.io/aws-ebs, ...)
- Large exports to S3/SMB/File System connections require Kubernetes nodes with a size of at least 200 GB

Client:
- Configured kubectl for cluster
- Helm version 2.12+
- Windows or Linux OS terminal

Notes:
- Download the necessary libraries from the link provided to you by Altair. These libraries must be copied after Knowledge Hub installation to the folder \utils\libs.
- All commands are valid for Windows and Linux environments.
Setup Knowledge Hub

The following instructions describe how to deploy Knowledge Hub Enterprise Server. Note that in these steps, `<knowledge_hub_namespace>` should be replaced with the Knowledge Hub Enterprise Server namespace.

Steps:

1. Create a Knowledge Hub Enterprise namespace.

   ```
kubectl create ns <knowledge_hub_namespace>
   ```

2. Create Docker secret `regsecret` in the Knowledge Hub Enterprise namespace to pull Docker images from the repository:

   ```
   # For release version
   kubectl create secret Docker-registry -n <knowledge_hub_namespace> regsecret --Docker-server=knowledgehubdev.azurecr.io --Docker-username=knowledgehubdev --Docker-password=<Docker registry altair password> --Docker-email=it@custom.com
   ```

3. Configure licensing.

   - **If you are using File Licensing**, deploy the Knowledge Hub Enterprise license in Kubernetes. Run the command from a folder with the `license.lic` file:

     ```
     kubectl create secret generic -n <knowledge_hub_namespace> license --from-file license.lic
     ```

   - **If you are using HWU Licensing**, update `values.yaml` file with the following property:

     ```
     license-api:
     config:
     yaml:
       application:
       license:
       provider: hwu
       hwu:
       host: "<port>@<host>"
     ```

4. Upload certificate for Knowledge Hub Enterprise domain:

   ```
   kubectl create secret tls tls-cert -n <knowledge_hub_namespace> --key tls.key --cert tls.crt
   ```
You can generate a self-signed certificate, but doing so is not recommended for production use.

```bash
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt -subj "/CN=knowledgehub.local"
```

5. Configure `values.yaml` file. The `/knowledgehub/example-values` folder of build artifact contains several sample files, each of which describes a different deployment size:

- `aws-large-values.yaml`
- `aws-medium-values.yaml`
- `aws-small-values.yaml`

Update the following properties in `values.yaml`:

- set `ingress_host` to Knowledge Hub URL;
- set `registry` to registry that will be used to pull images
- set `sharedStorageClass` to storage class name of your cluster

6. To deploy Knowledge Hub Enterprise with custom values, select and edit `<aws-values-yaml>` configuration and execute:

```bash
helm upgrade --install --namespace <knowledge hub namespace> -f <aws-values-yaml> knowledgehub <installer folder>/knowledgehub/knowledgehub-*.tgz
```

where:

- `<installer folder>` is the folder with the unzipped build artifacts
- `<aws-values-yaml>` is one of:
  - `aws-large-values.yaml`
  - `aws-medium-values.yaml`

7. Upload the libraries to the server using the steps outlined [here](#).

8. To check the installation, execute:

```bash
# to get chart status
helm ls -c knowledgehub
# to get statuses of all Knowledge Hub components
helm status knowledgehub
# to get chart values
helm get values knowledgehub
```
Installing JDBC Drivers

To create connections to third-party applications such as Google BigQuery, SQL Server, and Amazon Redshift, the appropriate drivers must be obtained and uploaded to the /libs folder.

Steps:

1. Download the Altair Knowledge Hub Linux (JDBC) drivers from the link provided to you by Altair. The drivers will come in a zipped file.
2. Unzip the file and place its contents in ./utils/libs.
3. Run the script ./bin/utils/linux-config.sh and then choose option 3.
4. Restart the Knowledge Hub services by running the script ./bin/utils/linux-config.sh and then choose option 9.

Users seeking to create custom connections using other drivers (i.e., those not currently included in the set of drivers provided by Altair) in Knowledge Hub should follow the steps above to do so. Note that the JDBC versions of these drivers must be used.
Spring Configuration

**Note:** Key properties should be in upper case and use '_-' instead of '.' and '-_' (e.g., `spring.data.cassandra.enabled should be SPRING_DATA_CASSANDRA_ENABLED; application.server.internet-address should be APPLICATION_SERVER_INTERNET_ADDRESS`.

Use **optionalEnv** to set properties:

```java
core-api:
  config:
    optionalEnv:
      CONFIGURATIONPROPERTY_KEY_1: "value"
      CONFIGURATIONPROPERTY_KEY_2: "value"

data-engine-api:
  config:
    optionalEnv:
      CONFIGURATIONPROPERTY_KEY_1: "value"
      CONFIGURATIONPROPERTY_KEY_2: "value"

data-engine-batch:
  worker:
    config:
      optionalEnv:
        CONFIGURATIONPROPERTY_KEY_1: "value"
        CONFIGURATIONPROPERTY_KEY_2: "value"

data-engine-worker:
  config:
    optionalEnv:
      CONFIGURATIONPROPERTY_KEY_1: "value"
      CONFIGURATIONPROPERTY_KEY_2: "value"

machine-learning:
  spark:
    config:
      optionalEnv:
        CONFIGURATIONPROPERTY_KEY_1: "value"
        CONFIGURATIONPROPERTY_KEY_2: "value"
```
## CORE-API PROPERTIES

The following table describes, in detail, the parameters that may be added/modified for this service.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPRING</strong></td>
<td></td>
</tr>
<tr>
<td>SPRING_DATA_CASSANDRA_ENABLED</td>
<td>Accepts the values <strong>true</strong> or <strong>false</strong> Enables (when true) or disables (when false) exports to the Library and exports of pinned data</td>
</tr>
<tr>
<td>SPRING_DATASOURCE_URL</td>
<td>Describes the connection to the Postgres database for the Knowledge Hub service</td>
</tr>
<tr>
<td>SPRING_DATASOURCE_URL_JDBC</td>
<td></td>
</tr>
<tr>
<td>SPRING_DATASOURCE_URL_USERNAME</td>
<td></td>
</tr>
<tr>
<td>SPRING_DATASOURCE_URL_PASSWORD</td>
<td></td>
</tr>
<tr>
<td>SPRING_HTTP_MULTIPART_MAXFILESIZE</td>
<td>Describes the maximum size of files that may be uploaded to the application (e.g., 2000MB)</td>
</tr>
<tr>
<td>SPRING_HTTP_MULTIPART_MAXREQUESTSIZE</td>
<td></td>
</tr>
<tr>
<td><strong>SERVER</strong></td>
<td></td>
</tr>
<tr>
<td>SERVER_PORT</td>
<td>Port on which the application is running</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_ENABLED</td>
<td><strong>true</strong> if HTTPS is enabled</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_ENABLED SERVER_PORT_SSL_KEY_STORE</td>
<td>Certificate settings for HTTPS</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_KEY_STORE_PASSWORD</td>
<td></td>
</tr>
<tr>
<td><strong>APPLICATION</strong></td>
<td></td>
</tr>
<tr>
<td>APPLICATION_SERVERINTERNET_ADDRESS</td>
<td>Describes the redirect URL for login to Salesforce, Google Analytics, Google Adwords (should be identical to the URL specified for ClientId and ClientSecret for these connections), etc.</td>
</tr>
<tr>
<td>APPLICATION_HTTP_CACHE_TIMETOLIVEINDAYS</td>
<td>Describes the amount of time in days that may elapse before a data source’s cache times out</td>
</tr>
<tr>
<td>APPLICATION_DATA_ENGINESTORE_&lt;DESIGN_MODE_LIMIT&gt;</td>
<td>Describes the row limit to be used for data sources in Design Mode; the default value is 10K</td>
</tr>
<tr>
<td>APPLICATION_DATA_ENGINE_SUGGESTION_&lt;PREPARE_CRON&gt;</td>
<td>Describes settings for jobs that calculate suggestions based on data type and content  e.g., 0 */30 * * * * - jobs are run every 30 min</td>
</tr>
<tr>
<td>APPLICATION_DATA_ENGINE_API_URL</td>
<td>URL for internal communication between Knowledge Hub and Knowledge Hub Data Engine services (http://&lt;machine name&gt;:8081)</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>APPLICATION DSL SOURCE CLEANER CRON APPLICATION DSL SOURCE EXPIRATION IN HOURS APPLICATION DSL TEMPORARY ITEM CLEANER CRON APPLICATION DSL TEMPORARY ITEM EXPIRATION IN HOURS</td>
<td>Describes settings for jobs that delete temporary objects</td>
</tr>
<tr>
<td>APPLICATION LICENSE_LOCAL_FILEPATH</td>
<td>Describes the path to the Knowledge Hub license</td>
</tr>
<tr>
<td>APPLICATION IO CONNECTION DISABLED</td>
<td>Specifies which connection types to disable (hide) in Knowledge Hub for all users</td>
</tr>
<tr>
<td>APPLICATION IO APPDATAFOLDER</td>
<td>Describes the path to the application’s internal storage (i.e., File Library; default: <code>${ProgramData}/Datawatch/DNS</code></td>
</tr>
<tr>
<td>APPLICATION IO WRITER COGNOS HTTP CLIENT TIMEOUT</td>
<td>Describes the time in seconds that may elapse before connections to IBM Cognos Analytics time out e.g., 600</td>
</tr>
<tr>
<td>APPLICATION IO READER PREVIEW LIMIT</td>
<td>1000 – row limit for previewing data sources</td>
</tr>
<tr>
<td>APPLICATION IO READER JDBC FETCH SIZE</td>
<td>Describes the number of rows to fetch for a query to a database using JDBC drivers, e.g., 200</td>
</tr>
<tr>
<td>APPLICATION SECURITY AUTHENTICATION XAUTH TOKEN VALIDITY IN SECONDS</td>
<td>Describes how many seconds should elapse before a user times out e.g., 1800</td>
</tr>
<tr>
<td>APPLICATION SECURITY AUTHENTICATION SSO_ENABLED APPLICATION SECURITY AUTHENTICATION SSO AD DOMAIN APPLICATION SECURITY AUTHENTICATION SSO AD SERVER APPLICATION SECURITY AUTHENTICATION SSO SERVICE PRINCIPAL APPLICATION SECURITY AUTHENTICATION SSO KEY TAB LOCATION APPLICATION SECURITY AUTHENTICATION SSO LDAP SEARCH BASE APPLICATION SECURITY AUTHENTICATION SSO LDAP SEARCH FILTER APPLICATION SECURITY AUTHENTICATION SSO REQUEST REGEX APPLICATION SECURITY AUTHENTICATION SSO USER ROLES</td>
<td>Settings for LDAP SSO</td>
</tr>
<tr>
<td>APPLICATION SCHEDULES MONITORING INTERVAL IN MINUTES</td>
<td>Number of minutes that must elapse before the next monitoring operation should be executed in a monitoring schedule</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_FAILED_ATTEMPT_MIN_DELAY_SEC</td>
<td>Delay after the first failed login attempt e.g., 8</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_FAILED_ATTEMPT_MAX_DELAY_SEC</td>
<td>Maximum delay time after a failed login attempt, e.g., 600</td>
</tr>
</tbody>
</table>

**DATA-ENGINE-API PROPERTIES**

The following table describes, in detail, the parameters that may be added/modified for this service.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPRING</td>
<td></td>
</tr>
<tr>
<td>SPRING_DATA_CASSANDRA_ENABLED</td>
<td>Accepts the values true or false Enables (when true) or disables (when false) exports to the Library and exports of pinned data</td>
</tr>
<tr>
<td>SPRING_DATASOURCE_URL_JDBC</td>
<td>Describes the connection to the Postgres database for the Knowledge Hub service</td>
</tr>
<tr>
<td>SPRING_DATASOURCE_URL_USERNAME</td>
<td></td>
</tr>
<tr>
<td>SPRING_DATASOURCE_URL_PASSWORD</td>
<td></td>
</tr>
<tr>
<td>LOGGING</td>
<td></td>
</tr>
<tr>
<td>LOGGING_FILE</td>
<td>full path to Data Engine service log file</td>
</tr>
<tr>
<td>LOGBACK</td>
<td></td>
</tr>
<tr>
<td>LOGBACK_LOGLEVEL</td>
<td>Logging level of the Data Engine service log file</td>
</tr>
<tr>
<td>SERVER</td>
<td></td>
</tr>
<tr>
<td>SERVER_PORT</td>
<td>8081 – port on which the Data Engine is running</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_ENABLED</td>
<td>true if HTTPS is used.</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_KEY_STORE</td>
<td>Certificate settings for HTTPS</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_KEY_STORE_PASSWORD</td>
<td></td>
</tr>
<tr>
<td>SERVER_PORT_SSL_KEY_PASSWORD</td>
<td></td>
</tr>
<tr>
<td>APPLICATION</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_DATAENGINE_SUGGESTION_RANK_THRESHOLD</td>
<td>Describes settings for suggestions based on data type and content; shows minimum rank for retrieving and sorting suggestions</td>
</tr>
<tr>
<td>APPLICATION_DATAENGINE_STORE_STATISTICS_AWAIT_TIMEOUT</td>
<td>Time to wait before statistics requests time out e.g., 60s</td>
</tr>
<tr>
<td>APPLICATION_DATAENGINE_STORE_DESIGN_MODE_LIMIT</td>
<td>Describes the row limit to be used for data sources in Design Mode; the default value is 10K</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>APPLICATION_DATA_ENGINE_STORE_GLOBALROWLIMIT</td>
<td>Row limit applied when the Design Mode limit is disabled e.g., 5000</td>
</tr>
<tr>
<td>APPLICATION_DATA_ENGINE_STORE_COLUMN_LIMIT</td>
<td>100 - column limit after Pivot and Transpose.</td>
</tr>
<tr>
<td>APPLICATION_DATA_ENGINE_STORE_DISTINCT_VALUE_LIMIT</td>
<td>250 - number of displayed distinct values limit</td>
</tr>
<tr>
<td>APPLICATION_DATA_ENGINE_STORE_LIMIT_DATA_NODES</td>
<td>Enables or disables limit to count of rows in all data nodes e.g., true (enabled); false (disabled)</td>
</tr>
<tr>
<td>APPLICATION_DATA_ENGINE_STORE_EXPORT_DATA_AWAIT_TIMEOUT_IN_SEC</td>
<td>3600 - export timeout</td>
</tr>
<tr>
<td>APPLICATION_IO_WRITER_COGNOS_HTTP_CLIENT_TIMEOUT</td>
<td>600 - timeout for connection to IBM Cognos Analytics</td>
</tr>
<tr>
<td>APPLICATION_SERVERINTERNET_ADDRESS</td>
<td>Redirect URL for logins to Salesforce, Microsoft SharePoint, Google Analytics, Google BigQuery, Google Adwords, Google Drive (redirect url should be specified for ClientId and ClientSecret for Google connections).</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_XAUTH_SECRET</td>
<td>Security token, should be equal to all other security tokens in all other application config files</td>
</tr>
<tr>
<td>APPLICATION_CORE_API_URL</td>
<td>Address for internal communication with the Knowledge Hub service</td>
</tr>
<tr>
<td>READER</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_TIMEOUT_IN_SEC</td>
<td>Describes the time in seconds that may elapse before connections to JDBC drivers time out e.g., 60</td>
</tr>
<tr>
<td>JDBC</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_*</td>
<td>Configuration settings for JDBC drivers</td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_DEFAULT_LOGINTIMEOUT</td>
<td>Describes the time in seconds that may elapse before connections to JDBC drivers time out after login e.g., 60</td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_DEFAULT_SOCKETTIMEOUT</td>
<td>Describes the time in seconds that may elapse before a socket timeout occurs when using connections to JDBC drivers e.g., 60</td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_CDATA_JDBC_ALL_TIMEOUT</td>
<td>Describes the time in seconds that may elapse before all connections to JDBC drivers time out e.g., 60</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_COM_MYSQL_JDBC_DRIVER_USECURSORFETCH</td>
<td>Settings for mySQL JDBC driver</td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_COM_MYSQL_JDBC_DRIVER_LOGINTIMEOUT</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_COM_MYSQL_JDBC_DRIVER_SOCKETTIMEOUT</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_ORACLE_JDBC_ORACLEDRIVER_ORACLE_NET_</td>
<td>Settings for Oracle JDBC driver</td>
</tr>
<tr>
<td>CONNECT_TIMEOUT</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_ORACLE_JDBC_ORACLEDRIVER_ORACLE_JDBC_</td>
<td></td>
</tr>
<tr>
<td>READTIMEOUT</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_COM_FACEBOOK_PRESTO_JDBC_PRESTODRIVER_</td>
<td>Settings for Presto JDBC driver</td>
</tr>
<tr>
<td>SSL</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_COM_FACEBOOK_PRESTO_JDBC_PRESTODRIVER_</td>
<td></td>
</tr>
<tr>
<td>PASSWORD</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_IO_READER_JDBC_DRIVER_ORG_APACHE_HIVE_JDBC_HIVEDRIVER_</td>
<td>Settings for Apache Hive JDBC driver</td>
</tr>
<tr>
<td>JAVA_SECURITY_KRB5_CONF</td>
<td></td>
</tr>
<tr>
<td>Security.Auth.Login.Config</td>
<td></td>
</tr>
<tr>
<td>READER_PREVIEW_LIMIT</td>
<td>1000 - row limit for preview data sources.</td>
</tr>
<tr>
<td>REPORT</td>
<td></td>
</tr>
<tr>
<td>REPORT_TEXT_VIEW_MAX_CACHE_IN_MB</td>
<td>This option sets the limit in megabytes for storing</td>
</tr>
<tr>
<td></td>
<td>the converted reports.</td>
</tr>
<tr>
<td>REPORT_TEXT_VIEW_MAX_CACHE_COUNT</td>
<td>This option sets the limit in counts for storing</td>
</tr>
<tr>
<td></td>
<td>the converted reports.</td>
</tr>
<tr>
<td>REPORT_TEXT_VIEW_NUMBER_OF_PAGES</td>
<td>If the page number of THE report exceeds this setting,</td>
</tr>
<tr>
<td></td>
<td>then conversion option converts from PDF reports to</td>
</tr>
<tr>
<td></td>
<td>TXT report.</td>
</tr>
</tbody>
</table>
SOCIAL-API PROPERTIES

The following table describes, in detail, the parameters that may be added/modified for this service.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SERVER</strong></td>
<td>A security token that should be identical to all other security token indicated in all other application config files (e.g., application.security.authentication.xauth.secret in other config files)</td>
</tr>
<tr>
<td>SERVER_JWT_SECRET</td>
<td>true if HTTPS is used. Describes parameters for the SSL certificate.</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_ENABLED</td>
<td>false if only a single run is applied</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_KEYSTORE_PASSWORD</td>
<td>allocated memory size for Spark service</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_KEYSTORE_PATH</td>
<td>Number of attempts to connect to the Cassandra database</td>
</tr>
<tr>
<td>SPARK_APP_NAME</td>
<td>Spark application name</td>
</tr>
<tr>
<td>SPARK_APP_SCHEDULE</td>
<td>登入日期</td>
</tr>
<tr>
<td>SPARK_APP_SCHEDULING_ENABLED</td>
<td>Number of second for each attempt to connect to the Cassandra database</td>
</tr>
<tr>
<td>SPARK_DRIVER_MEMORY</td>
<td>Number of attempts to connect to the Cassandra database</td>
</tr>
<tr>
<td>DATABASE</td>
<td>Logging level</td>
</tr>
<tr>
<td>DATABASE_CASSANDRA_CONNECTION_ATTEMPTS_AMOUNT</td>
<td>Full path to the ML and Spark services log file (e.g., C:\Windows\Temp\MonarchSwarm\Logs\ml-app.log)</td>
</tr>
<tr>
<td>DATABASE_CASSANDRA_CONNECTION_WAIT_TIME_SECONDS</td>
<td>Minimum number of actions in sequence to generate suggestions</td>
</tr>
<tr>
<td>LOGGING_LOGLEVEL</td>
<td>Minimum number of actions in sequence to generate suggestions</td>
</tr>
<tr>
<td>LOGGING_LOGFILEPATH</td>
<td>Logging level</td>
</tr>
<tr>
<td>LOGGING_LOGLEVEL</td>
<td>Full path to the ML and Spark services log file (e.g., C:\Windows\Temp\MonarchSwarm\Logs\ml-app.log)</td>
</tr>
<tr>
<td>LOGGING_LOGFILEPATH</td>
<td>Minimum number of actions in sequence to generate suggestions</td>
</tr>
<tr>
<td>LOGGING_LOGFILEPATH</td>
<td>Minimum number of actions in sequence to generate suggestions</td>
</tr>
</tbody>
</table>
LICENSE-API PROPERTIES

The following table describes, in detail, the parameters that may be added/modified for this service.

Note that you can easily switch between license providers:

- **local** – license specified in `APPLICATION_LICENSE_LOCAL_FILEPATH`
- **hwu** – Local core-api HWU License Service provider.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPRING</strong></td>
<td>Name of Spring application (<strong>license-api</strong>)</td>
</tr>
<tr>
<td>SPRING_APPLICATION_NAME</td>
<td>Name of Spring application (license-api)</td>
</tr>
<tr>
<td><strong>SERVER</strong></td>
<td>8085 - Port on which License service is running</td>
</tr>
<tr>
<td>SERVER_PORT</td>
<td>Port on which License service is running</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_ENABLED</td>
<td>True if HTTPS is used</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_KEY_STORE</td>
<td>Certificate settings for HTTPS</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_KEY_STORE_PASSWORD</td>
<td>Certificate settings for HTTPS</td>
</tr>
<tr>
<td>SERVER_PORT_SSL_KEY_PASSWORD</td>
<td>Certificate settings for HTTPS</td>
</tr>
<tr>
<td>SERVER_COMPRESSION_ENABLED</td>
<td>Enable or disable HTTP response compression</td>
</tr>
<tr>
<td>SERVER_COMPRESSION_MIME_TYPES</td>
<td>Content types that are compressed (e.g., text/html, application/json)</td>
</tr>
<tr>
<td><strong>APPLICATION</strong></td>
<td>Type of license provider, can be &quot;local&quot; or &quot;hwu&quot;</td>
</tr>
<tr>
<td>APPLICATION_LICENSE_PROVIDER</td>
<td>Type of license provider, can be &quot;local&quot; or &quot;hwu&quot;</td>
</tr>
<tr>
<td>APPLICATION_LICENSE_LOCAL_FILEPATH</td>
<td>Path to license.lic file</td>
</tr>
<tr>
<td>APPLICATION_LICENSE_HWU_HOST</td>
<td>Altair License Server address. Should be written as &quot;;&lt;port&gt;;&lt;host&gt;&quot;. Note that the URL to the Altair License Server should be set as an environment variable.</td>
</tr>
<tr>
<td>APPLICATION_LICENSE_HWU_CHECKER_CRON</td>
<td>Schedule to execute remote license pool check (e.g., **00/5 * * * ***)</td>
</tr>
<tr>
<td>APPLICATION_LICENSE_HWU_GROUP</td>
<td>Name of group on Altair License Server (e.g., ${COMPUTERNAME}). Note that this property should also be set as an environment variable.</td>
</tr>
<tr>
<td>APPLICATION_LICENSE_HWU_LOG_ENABLED</td>
<td>Enable (true) or disable (false) hwu logging</td>
</tr>
<tr>
<td>APPLICATION_LICENSE_HWU_LOG_LEVEL</td>
<td>Level of hwu logging (e.g., info)</td>
</tr>
<tr>
<td>APPLICATION_LICENSE_HWU_LOG_FACILITY</td>
<td>Type of output (e.g., stderr)</td>
</tr>
</tbody>
</table>
Setting Up LDAP Authentication

The following steps describe how to implement LDAP authentication in Knowledge Hub.

Steps:

1. Install Knowledge Hub with default parameters.
2. Login to the Kubernetes dashboard and update the core-api configuration in ConfigMaps.

```json
{"data":{
    "APPLICATION_SECURITY_AUTHENTICATION_PROVIDER": "ldap <available values: basic, ldap, oauth2>",
    "APPLICATION_SECURITY_AUTHENTICATION_USERSPROVISIONED": "false",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ACTIVE_DIRECTORY": "true",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_DOMAIN": "<domain>",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SERVER": "ldap://<full computer name of LDAP server>.<domain name>/",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_MANAGED": "<LDAP admin user>",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_MANAGEDN": "<password of admin user>",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTE_MAPPING_COMMONNAME": "cn",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTE_MAPPING_EMAIL": "mail",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTE_MAPPING_FIRSTNAME": "givenname",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTE_MAPPING_GROUPS": "memberOf",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTE_MAPPING_LASTNAME": "sn",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTE_MAPPING_LOGIN": "userPrincipalName",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTE_MAPPING_PHONENUMBER": "telephonenumber",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_CUSTOMATTRIBUTES": "displayName, distinguishedName, name, objectCategory, objectClass, primaryGroupID, sAMAccountName, sAMAccountType, servicePrincipalName",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SEARCHBASE=DC": "<domain component 1>,DC=<domain component 2>",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SEARCHFILTER": "(| (userPrincipalName={0}) (sAMAccountName={0}))",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_GROUP_MAPPING": "true",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLE_MAPPING": "true",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_GROUP_MAPPING_1": "GroupName1, GroupName2",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLE_MAPPING_1": "GroupName1, GroupName2",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLE_MAPPING_2": "GroupName1, GroupName2",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_USER_ROLES": "2",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ADMIN_USERS": "<super_admin user1>, <super_admin user2>"
}}
```
For example, if the full computer name of the Knowledge Hub server is **WIN-SWARMSERVER**, the LDAP server is **WIN-LDAPSERVER**, and the domain name is **altair.com**:

```
{
    "data": {
        "APPLICATION_SECURITY_AUTHENTICATION_PROVIDER": "ldap",
        "APPLICATION_SECURITY_AUTHENTICATION_USERSPROVISIONED": "false",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ACTIVEDIRECTORY": "true",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_DOMAIN": "altair.com",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SERVER": "ldap://WIN-LDAPSERVER.altair.com/",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_MANAGEDDN": "swarmadmin@altair.com",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_MANAGEPASSWORD": "Passw0rd#",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_COMMONNAME": "cn",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_EMAIL": "mail",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_FIRSTNAME": "givenname",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_GROUPS": "memberOf",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_LASTNAME": "sn",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_LOGIN": "userPrincipalName",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_PHONENUMBER": "telephonenumber",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_CUSTOMATTRIBUTES": "displayName, distinguishedName, name, objectCategory, objectClass, primaryGroupID, sAMAccountName, sAMAccountType, servicePrincipalName",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SEARCHBASE=DC": "altair,DC=com",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SEARCHFILTER": "((userPrincipalName={0}) (sAMAccountName={0}))",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_GROUPMAPPING": "true",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLEMAPPING": "true",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLESMAP_1": "Accounting, Finance",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLESMAP_2": "Sales, Marketing",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_USERROLES": "3",
        "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ADMINUSERS": "bmatthews@altair.com, mbarney@altair.com"
    }
}
```

3. Restart the core-api service.
Setting Up SSO Authentication

The following steps describe how to implement SSO authentication in Knowledge Hub.

Steps:
1. Install Knowledge Hub with default parameters.
2. Generate a linuxsso.keytab
   For example, if the full computer name of the Knowledge Hub server is WIN-SWARMSERVER and the domain name is altair.com, run the following script in Powershell:
   ```
   setspn -A HTTP/WIN-SWARMSERVER.altair.com knhubenterprise
   ktpass /out c:\temp\linuxsso.keytab /mapuser
   knhubenterprise@ALTAIR.COM /princ HTTP/WIN-SWARMSERVER@ALTAIR.COM
   /pass Password# /ptype KRB5_NT_PRINCIPAL /crypto All
   ```
3. Configure the krb5.conf file.
   ```
   [libdefaults]
   default_realm = <DOMAIN NAME>
   default_keytab_name = <path to keytab file>
   forwardable=true
   dns_lookup_realm = true
   rdns = false
   dns_lookup_kdc = true
   
   [realms]
   <DOMAIN NAME> = {
     kdc = <name of domain controller>.<domain>:88
     admin_server = <name of domain controller>.<domain>:88
   }
   
   [domain_realm]
   <domain> = <DOMAIN>
   .<domain> = <DOMAIN>
   
   [appdefaults]
   kinit = {
     renewable = true
     forwardable= true
   }
   ```
For example, if the full computer name of the domain controller is **WIN-LDAPSERVER** and the domain name is **altair.com**:

```plaintext
[libdefaults]
default_realm = ALTAIR.COM
default_keytab_name = /libs/linuxsso.keytab
forwardable=true
dns_lookup_realm = true
dns = false
dns_lookup_kdc = true

[realms]
ALTAIR.COM = {
    kdc = WIN-LDAPSERVER.altair.com:88
    admin_server = WIN-LDAPSERVER.altair.com:88
}

[domain_realm]
altair.com = ALTAIR.COM
.altair.com = ALTAIR.COM

[appdefaults]
kinit = {
    renewable = true
    forwardable= true
}
```

4. Add files **krb5.conf** and **linuxsso.keytab** to the `./libs` folder on the Knowledge Hub server by using the `utils` script with the option "upload libraries from ./libs folder" (#3).

5. Login to the Kubernetes dashboard and update the core-api configuration in ConfigMaps.
"data": {
    "JAVA_OPTS": "-Djava.security.krb5.conf=/libs/krb5.conf -Dsun.security.krb5.debug=true",
    "APPLICATION_SECURITY_AUTHENTICATION_PROVIDER": "ldap",
    "APPLICATION_SECURITY_AUTHENTICATION_USERSPROVISIONED": "false",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SSO_ENABLED": "true",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SSO_SERVICE_PRINCIPAL": "HTTP/<full computer name of Knowledge Hub server>/@<DOMAIN NAME>",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SSO_KEYTABLOCATION": "/libs/linuxsso.keytab",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SSO_REQUESTREGEX": "/^api/+/ldap_sso",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_LOGIN": "userPrincipalName",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_FIRST_NAME": "givenname",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_LAST_NAME": "sn",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_COMMON_NAME": "cn",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_EMAIL": "mail",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_PHONE_NUMBER": "telephoneNumber",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_GROUPS": "memberOf",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_CUSTOMATTRIBUTES": "displayName, distinguishedName, name, objectCategory, objectClass, primaryGroupID, sAMAccountName, sAMAccountType, servicePrincipalName",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ACTIVEDIRECTORY": "true",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_DOMAIN": "<domain name>",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SERVER": "ldap://<full computer name of LDAP server>.<domain name>/",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_MANAGEDN": "<LDAP admin user>",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_MANAGEPASSWORD": "<password of admin user>",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SEARCHBASE": "DC=<domain component 1>,DC=<domain component 2>",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SEARCHFILTER": " ((userPrincipalName={0}) (sAMAccountName={0}))",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_GROUPMAPPING": "true",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLEMAPPING": "true",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLESMAP_1": "GroupName1, GroupName2",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLESMAP_2": "GroupName1, GroupName2",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_USER_ROLES": "2",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ADMIN_USERS": "<super_admin user1>, <super_admin user2>"
}
For example, if the full computer name of the Knowledge Hub server is WIN-SWARMSERVER, the LDAP server is WIN-LDAPSERVER, and the domain name is altair.com:

```json
{...
  "data": {
    "JAVA_OPTS": "-Djava.security.krb5.conf=/libs/krb5.conf -Dsun.security.krb5.debug=true",
    "APPLICATION_SECURITY_AUTHENTICATION_PROVIDER": "ldap",
    "APPLICATION_SECURITY_AUTHENTICATION_USERSPROVISIONED": "false",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SSO_ENABLED": "true",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SSO_SERVICEPRINCIPAL": "HTTP/\WIN-SWARMSERVER.altair.com/@ALTAIR.COM",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SSO_KEYTABLOCATION": "/libs/linuxsso.keytab",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SSO_REQUESTREGEX": "^/api.+/ldap_sso",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUITEMAPPING_LOGIN": "userPrincipalName",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUITEMAPPING_FIRST_NAME": "givenname",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUITEMAPPING_LAST_NAME": "sn",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUITEMAPPING_COMMON_NAME": "cn",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUITEMAPPING_EMAIL": "mail",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUITEMAPPING_PHONE_NUMBER": "telephonenumber",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUITEMAPPING_GROUPS": "memberOf",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_CUSTOMATTRIBUTES": "displayName, distinguishedName, name, objectCategory, objectClass, primaryGroupID, sAMAccountName, sAMAccountType, servicePrincipalName",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_ACTIVEDIRECTORY": "true",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_DOMAIN": "altair.com",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SERVER": "ldap://\WIN-LDAPSERVER.altair.com/",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_MANAGEDN": "sam@altair.com",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_MANAGEPASSWORD": "#Passw0rd#",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SEARCHBASE": "DC=altair,DC=com",
    "APPLICATION_SECURITY_AUTHENTICATION_LDAP_SEARCHFILTER": "( (userPrincipalName={0}) {sAMAccountName={0}})"
  }
}
```
Each of the properties added to the core-api configuration file is described as follows:

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_PROVIDER</td>
<td>Use ldap for LDAP/SSO; may also be basic for basic or oauth2 for OAuth2 authentication</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_USERSPROVISIONED</td>
<td>Enables (true) or disables (false) explicit provisioning. If explicit provisioning is disabled, the system creates Knowledge Hub users automatically. When set to true, users must be created manually</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_DEFAULT_PASSWORD</td>
<td>The default password for new users created through LDAP and added multiple users</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_SSO_ENABLED</td>
<td>true to enable SSO; false if using LDAP authentication</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_SSO_SERVICE_PRINCIPAL</td>
<td>Full computer name of the Knowledge Hub server in the form HTTP/&lt;COMPUTER NAME&gt;.&lt;domain&gt;/@&lt;DOMAIN&gt; (e.g., HTTP/WIN-SWARMSERVER.altair.com@ALTAIR.COM)</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_SSO_KEYTABLOCATION</td>
<td>Path to keytab file (e.g., /libs/linuxss0.keytab)</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_SSO_REQUESTREGEX</td>
<td>^/api/.+/ldap_sso - Setting for SSO</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_ATTRIBUTEMAPPING_</td>
<td>Attributes used to add users by LDAP query</td>
</tr>
<tr>
<td>LOGIN: USERPRINCIPALNAME</td>
<td></td>
</tr>
<tr>
<td>FIRST_NAME: GIVENNAME</td>
<td></td>
</tr>
<tr>
<td>LAST_NAME: SN</td>
<td></td>
</tr>
<tr>
<td>COMMON_NAME: CN</td>
<td></td>
</tr>
<tr>
<td>EMAIL: MAIL</td>
<td></td>
</tr>
<tr>
<td>PHONE_NUMBER: TELEPHONENUMBER</td>
<td></td>
</tr>
<tr>
<td>GROUPS: MEMBEROF</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP_QUERY_CUSTOMATTRIBUTES</td>
<td></td>
</tr>
<tr>
<td>DISPLAYNAME</td>
<td></td>
</tr>
<tr>
<td>DISTINGUISHEDNAME</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td></td>
</tr>
<tr>
<td>OBJECTCATEGORY</td>
<td></td>
</tr>
<tr>
<td>OBJECTCLASS</td>
<td></td>
</tr>
<tr>
<td>PRIMARYGROUPID</td>
<td></td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SAMACCOUNTNAME</td>
<td></td>
</tr>
<tr>
<td>SAMACCOUNTTYPE</td>
<td></td>
</tr>
<tr>
<td>SERVICEPRINCIPALNAME</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.ACTIVEDIRECTORY</td>
<td>true when AD is used</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.DOMAIN</td>
<td>Domain name</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.DOMAINUSERS</td>
<td>Allow LDAP authentication for any of two forests in one domain. The default value for this setting is false. To authenticate users from just one domain via LDAP, set this property to true and then set the correct domain in the property APPLICATION_SECURITY_AUTHENTICATION_LDAP.DOMAIN</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.SERVER</td>
<td>Full computer name of LDAP server</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.MANAGEDN</td>
<td></td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.MANAGEPASSWORD</td>
<td>User name and password to use to connect to LDAP server when SSO.ENABLED = false and LDAP.ACTIVEDIRECTORY = false. If SSO.ENABLED = true and LDAP.ACTIVEDIRECTORY = true, these properties may be omitted from the config file. These credentials are also used to add multiple users to Knowledge Hub using LDAP query</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.SEARCHBASE</td>
<td>Domain name components (e.g., DC=altair,DC=com if domain is altair.com)</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.SEARCHFILTER</td>
<td>Filter used to search for LDAP users</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.USER_ROLES</td>
<td>User role(s) for SSO users</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.ADMIN_USERS</td>
<td>List of users automatically created with the Super Administrator role in Knowledge Hub (if USERS.PROVISIONED = false). When this list is provided, there is no need to login as an administrator and create the first LDAP user.</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.ROLEMAPPING</td>
<td>true to enable role-mapping in Knowledge Hub; false to disable</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.GROUPMAPPING</td>
<td>true to enable group-mapping in Knowledge Hub; false to disable</td>
</tr>
<tr>
<td>APPLICATION_SECURITY_AUTHENTICATION_LDAP.ROLESMAP</td>
<td>Mapping of Knowledge Hub roles to LDAP groups</td>
</tr>
</tbody>
</table>
Notes:

- The property `APPLICATION_SECURITY_AUTHENTICATION_LDAP_SEARCH_FILTER` uses the format "username@domain".

- If a user does not specify the domain in the login form, the value in `APPLICATION_SECURITY_AUTHENTICATION_LDAP_DOMAIN` will be used as the domain.

- LDAP search attributes should have values in "username@domain" format.

- If the property `USERSPROVISIONED` is set to `TRUE`, and the user is not included in the `ADMIN_USERS` list, an error (i.e., "Users %user_login% does not exist") is returned when the user logs into the application via SSO. In this case, the user must be manually added through the User Management page (via LDAP) of Knowledge Hub.

- If the property `USERSPROVISIONED` is set to `FALSE`, and the user exists in Active Directory, a new user is created upon login to Knowledge Hub via SSO. This user’s profile will include a login, last name, and first name, and s/he will have the role(s) specified in `USER_ROLES`.

- If the user exists in Active Directory, and the new user is included in the `ADMIN_USERS` list, the user can log into Knowledge Hub via SSO and this user will have the role Super Administrator regardless if the property `USERSPROVISIONED` is set to `TRUE` or `FALSE`.

To enable role/group mapping, set the following properties:

- `APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLEMAPPING` – `true` to enable role mapping; `false` otherwise

- `APPLICATION_SECURITY_AUTHENTICATION_LDAP_GROUPMAPPING` - `true` to enable group mapping; `false` otherwise

- `APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLESMAP_%role_id1%`:
  
  "%GroupName1, GroupName2%" – Mapping of first Knowledge Hub role to LDAP groups

- `APPLICATION_SECURITY_AUTHENTICATION_LDAP_ROLESMAP_%role_id2%`:

  "%GroupName1, GroupName2%" – Mapping of second Knowledge Hub role to LDAP groups

6. Restart the core-api service.
Deleting Knowledge Hub
Enterprise

Deleting Knowledge Hub Enterprise Server

To delete the Knowledge Hub Enterprise helm chart and configuration, run:

```
helm del --purge <Knowledge Hub release name>
```

To get `<Knowledge Hub release name>`, run `helm ls` and find the Knowledge Hub Enterprise release name.

This command does not delete Persistent Volume Claims and Persistent Volumes. You can install Knowledge Hub Enterprise with helm once more without data loss.

To delete Knowledge Hub Enterprise Server completely, delete the Kubernetes namespace with the command:

```
kubectl delete ns <Knowledge Hub namespaces>
```

To get a list of available namespaces, run `kubectl get ns`.

Deleting Modules

DELETING THE TRACER (JAEGER)

1. Delete the jaeger-operator helm chart:

```
$ helm del jaeger-operator --purge
```
2. Delete the Jaeger components:

```
$ kubectl delete customresourcedefinition
jaegers.io.jaegertracing
```

**DELETING THE LOGGER (ELK STACK)**

To delete the logger, run:

```
helm del --purge <elk release name>
```

and then delete Kubernetes namespace: `kubectl delete ns <logging namespace>`.

**REMOVING MONITORING (GRAFANA)**

To remove monitoring, run:

```
helm del --purge <monitoring release name>
```

and then delete the Kubernetes namespace: `kubectl delete ns <monitoring namespace>`.

**Utilities Configuration**

To configure utilities for Knowledge Hub Enterprise Server, run `utils.sh <knowledge
hub helm release> <knowledge hub helm release> from the ./utils/ directory
and select:

- 1 - Show libraries in the cluster /libs folder
- 2 - download libraries to the './libs' folder - Download libraries on the local machine in the ./utils/libs from cluster /libs folder
- 3 - Upload libraries from the local machine folder ./utils/libs into the cluster /libs folder. After execution, all services must be restarted to apply changes.
- 4 - Removes libraries from the cluster /libs folder. After execution, all services must be restarted to apply changes.
- 5 - Get default tokens for the Knowledge Hub namespace
6 - Update your Knowledge Hub license file. Before execution, copy a new version of license.lic to the folder ./utils/

7 - Restart all services of the Knowledge Hub

8 - Restore Knowledge Hub databases from the '.utils/backup/<date>' folder

9 – Create backup Knowledge Hub database in '.utils/backup/<date>'

10 - Generate cipher keys for Knowledge Hub chart and patch the application with it (required)

11 - Exit the utils menu.
Elastic Log Export/Import for Knowledge Hub Enterprise

Exporting Elasticsearch to AWS S3

Elasticsearch logs can be exported to an AWS S3 bucket in JSON format.

Steps:
1. Create an S3 bucket for backup, e.g., `elasticsearch-dump-bucket`.
2. The file `elasticsearch-export-job.yaml` is necessary to export logs. Configure `elasticsearch-export-job.yaml` to specify S3 details (e.g., bucket, access key, secret access key) and query date.

   ```yaml
   - name: BUCKET
     value: "elasticsearch-dump-bucket"
   - name: ACCESS_KEY
     value: "<ACCESS_KEY>"
   - name: SECRET_ACCESS_KEY
     value: "<SECRET_ACCESS_KEY>"
   ```

   By default, logs will be exported for the last day. To change the date, add the following properties:

   ```yaml
   - name: FROM_DATE
     value: "2019-05-15"
   - name: TO_DATE
     value: "2019-05-15"
   ```

3. Deploy and run the following k8s job.

   ```bash
   kubectl apply -n logger -f elasticsearch-export-job.yaml
   ```
Importing Logs in k8s ELK

Steps:

1. Configure k8s port forwarding for the elk-elasticsearch-client service.

   ```bash
   kubectl port-forward -n logging service/elk-elasticsearch-client 9202:9200
   ```

2. Install elasticdump from the link [https://www.npmjs.com/package/elasticdump](https://www.npmjs.com/package/elasticdump)


   ```bash
   ```
Migrating (Upgrading) Windows Installations to a Cluster

Backing Up/Restoring Knowledge Hub

BACKING UP WINDOWS INSTALLATIONS

To backup Windows installations, the following required variables must be set:

- JAVA_HOME - Location of java installation
- PG_DUMP - Location of PostgreSQL pg_dump executable
- NODETOOL - Location of Cassandra nodetool executable
- CASSANDRA_DATA - Location of Cassandra data folder

Run the following commands from ./single-server/bin/utils/:

```
<table>
<thead>
<tr>
<th>Windows Backup Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET JAVA_HOME=C:\Program Files\Java&lt;jdk\jre folder&gt;</td>
</tr>
<tr>
<td>SET PG_DUMP=&quot;C:\Program Files\PostgreSQL\11\bin\pg_dump.exe&quot;</td>
</tr>
<tr>
<td>SET NODETOOL=&quot;C:\Program Files\Datawatch Monarch Swarm\cassandra\bin\nodetool.bat&quot;</td>
</tr>
<tr>
<td>SET CASSANDRA_DATA=&quot;C:\Program Files\Datawatch Monarch Swarm\cassandra\data\data&quot;</td>
</tr>
<tr>
<td>windows-backup.bat</td>
</tr>
</tbody>
</table>
```

Note that the parameters provided in the example above should be changed according to actual file locations on the given system.
If the error "C:\Program' is not recognized as an internal or external command, operable program or batch file" appears, change the first command to "SET JAVA_HOME=C:\Program Files\Java\<jdk/jre folder>".

The components will be backed up in ./single-server/bin/utils/backup/<date_time> (e.g., ./single-server/bin/utils/backup/2019-03-06_07-48-51).

After successful backup, the following files are stored in the backup folder:
dataengineapi_db.gz, newserver_db.gz, fs-file_library.tar, fs-libs.tar, newserver_keyspace.tar.gz, and datawatch_keyspace.tar.gz.

BACKING UP LINUX SINGLE-SERVER INSTALLATIONS

Knowledge Hub Single Server supports the backup and restoration of the following components.

- social-db – the Cassandra databases
- meta-db – the PostgreSQL databases
- file-system – the file-libraries and libs docker volumes

To backup Knowledge Hub Single Server, run the following command from ./bin/utils/.

./linux-backup.sh

The components will be backed up in ./bin/utils/backup/<date_time> (e.g.: ./bin/utils/backup/2019-03-06_07-48-51).

After successful backup, the following files are stored in the backup folder:
dataengineapi_db.gz, newserver_db.gz, fs-file_library.tar, fs-libs.tar, newserver_keyspace.tar.gz, and datawatch_keyspace.tar.gz.

RESTORING BACKED UP INSTALLATIONS VIA KUBERNETES

To restore backups of Linux Single Server or Windows installations, configure the backup_folder_path variable in <swarm-enterprise>/conf/swarm.yaml and then run <swarm-enterprise>/bin/utils.sh with option 8.

<table>
<thead>
<tr>
<th>Windows Backup</th>
</tr>
</thead>
<tbody>
<tr>
<td>./utils.sh # option 8</td>
</tr>
</tbody>
</table>

Logins to some connections (e.g., Google Drive) may need to be refreshed after restoring Knowledge Hub Enterprise Server.
Migrating Cipher Keys from Windows to Enterprise Server Installations

Copy values from `application-prod.yml` into helm values using following mapping matrix:

<table>
<thead>
<tr>
<th>application-prod.yml</th>
<th>values.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>application.security.authentication.xauth.secret</td>
<td>global.config.authentication.xauth.secret</td>
</tr>
<tr>
<td>application.security.cipher.keyPair.privateKey</td>
<td>global.config.cipher.keyPair.privateKey</td>
</tr>
<tr>
<td>application.security.cipher.keyPair.publicKey</td>
<td>global.config.cipher.keyPair.publicKey</td>
</tr>
</tbody>
</table>

Example of helm values:

```yaml
global:
  config:
    cipher:
      keyPair:
        privateKey: <PUT_KEY_HERE>
        publicKey: <PUT_KEY_HERE>
    authentication:
      xauth:
        secret: <PUT_KEY_HERE>
```
YAML File Templates

To use the following templates, copy their contents to a code editor, modify the necessary properties, save using the given filename, and then apply as given in the instructions.

**eks-vpc.yaml**

```yaml
---
AWSTemplateFormatVersion: '2010-09-09'
Description: VPC with two public and private subnets
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label: default: VPC configuration
        Parameters:
          - VPCName
          - VpcCIDR
      - Label: default: Public subnets
        Parameters:
          - PublicSubnet1CIDR
          - PublicSubnet2CIDR
      - Label: default: Private subnets
        Parameters:
          - PrivateSubnet1CIDR
          - PrivateSubnet2CIDR

Parameters:

VPCName:
  Description: Name of the VPC
  Type: String
  Default: EKS-VPC

VpcCIDR:
  Description: IP range (CIDR notation) for this VPC
```
Type: String
Default: 172.20.0.0/16

PublicSubnet1CIDR:
Description: IP range (CIDR notation) for the public subnet in the first az
Type: String
Default: 172.20.0.0/22

PublicSubnet2CIDR:
Description: IP range (CIDR notation) for the public subnet in the second az
Type: String
Default: 172.20.4.0/22

PrivateSubnet1CIDR:
Description: IP range (CIDR notation) for the private subnet in the first az
Type: String
Default: 172.20.32.0/19

PrivateSubnet2CIDR:
Description: IP range (CIDR notation) for the private subnet in the second az
Type: String
Default: 172.20.64.0/19

Resources:

VPC:
Type: AWS::EC2::VPC
Properties:
  CidrBlock: !Ref VpcCIDR
  Tags:
    - Key: Name
      Value: !Ref VPCName

InternetGateway:
Type: AWS::EC2::InternetGateway
Properties:
  Tags:
    - Key: Name
      Value: !Ref VPCName
InternetGatewayAttachment:
   Type: AWS::EC2::VPCGatewayAttachment
   Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

PublicSubnet1:
   Type: AWS::EC2::Subnet
   Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs '']
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
         - Key: Name
           Value: !Sub ${VPCName}-public-${AWS::Region}a

PublicSubnet2:
   Type: AWS::EC2::Subnet
   Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs '']
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
         - Key: Name
           Value: !Sub ${VPCName}-public-${AWS::Region}b

PublicRouteTable:
   Type: AWS::EC2::RouteTable
   Properties:
      VpcId: !Ref VPC
      Tags:
         - Key: Name
           Value: !Sub ${VPCName}-public

DefaultPublicRoute:
   Type: AWS::EC2::Route
   DependsOn: InternetGatewayAttachment
   Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
PublicSubnet1RouteTableAssociation:
   Type: AWS::EC2::SubnetRouteTableAssociation
   Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:
   Type: AWS::EC2::SubnetRouteTableAssociation
   Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2

PrivateSubnet1:
   Type: AWS::EC2::Subnet
   Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs '']
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
   Tags:
      - Key: Name
        Value: !Sub ${VPCName}-private-${AWS::Region}a

PrivateSubnet2:
   Type: AWS::EC2::Subnet
   Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs '']
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
   Tags:
      - Key: Name
        Value: !Sub ${VPCName}-private-${AWS::Region}b

PrivateRouteTable1:
   Type: AWS::EC2::RouteTable
   Properties:
      VpcId: !Ref VPC
   Tags:
      - Key: Name
        Value: !Sub ${VPCName}-private-${AWS::Region}a

PrivateRouteTable2:
   Type: AWS::EC2::RouteTable
Properties:
  VpcId: !Ref VPC
  Tags:
  - Key: Name
    Value: !Sub ${VPCName}-private-${AWS::Region}b

NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NatGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NatGateway1:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway1EIP.AllocationId
    SubnetId: !Ref PublicSubnet1
    Tags:
      - Key: Name
        Value: !Sub ${VPCName}-${AWS::Region}a

NatGateway2:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway2EIP.AllocationId
    SubnetId: !Ref PublicSubnet2
    Tags:
      - Key: Name
        Value: !Sub ${VPCName}-${AWS::Region}b

DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1

DefaultPrivateRoute2:
Type: AWS::EC2::Route
Properties:
  RouteTableId: !Ref PrivateRouteTable2
  DestinationCidrBlock: 0.0.0.0/0
  NatGatewayId: !Ref NatGateway2

PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2

ClusterControlPlaneSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: !Sub ${VPCName}-cluster-sg
    GroupDescription: Cluster communication with worker nodes
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${VPCName}-cluster-sg

Outputs:

VpcId:
  Description: VPC ID
  Value: !Ref VPC

PublicSubnets:
  Description: A list of the public subnets
  Value: !Join [',', [!Ref PublicSubnet1, !Ref PublicSubnet2]]

PrivateSubnets:
  Description: A list of the private subnets
  Value: !Join [',', [!Ref PrivateSubnet1, !Ref PrivateSubnet2]]
Description: Security group for the cluster control plane
Value: !Join [',', [!Ref ClusterControlPlaneSecurityGroup]]
Type: AWS::EC2::VPC::Id

Subnets:
Description: List of private subnets where workers can be created
Type: List[AWS::EC2::Subnet::Id]

ClusterName:
Description: The cluster name provided when the cluster was created. If it is incorrect, nodes will not be able to join the cluster.
Type: String

ClusterControlPlaneSecurityGroup:
Description: The security group of the cluster control plane.
Type: AWS::EC2::SecurityGroup::Id

NodeImageId:
Type: AWS::EC2::Image::Id
Description: EKS AMI id for the node instances, to find for your region perform command `aws ec2 describe-images --owners 602401143452 --filters "Name=name,Values=amazon-eks-node-1.10-v*" --query 'sort_by(Images, &CreationDate)[].ImageId'

NodeInstanceType:
Description: EC2 instance type for the node instances
Type: String
Default: m5.2xlarge
AllowedValues:
- t2.large
- t2.xlarge
- t2.2xlarge
- t3.large
- t3.xlarge
- t3.2xlarge
- m3.medium
- m3.large
- m3.xlarge
- m3.2xlarge
- m4.large
- m4.xlarge
- m4.2xlarge
- m4.4xlarge
- m4.10xlarge
- m5.large
- m5.xlarge
- m5.2xlarge
- m5.4xlarge
- m5.12xlarge
- m5.24xlarge
- c4.large
- c4.xlarge
- c4.2xlarge
- c4.4xlarge
- c4.8xlarge
- c5.1xlarge
- c5.xlarge
- c5.2xlarge
- c5.4xlarge
- c5.9xlarge
- c5.10xlarge
- i3.large
- i3.xlarge
- i3.2xlarge
- i3.4xlarge
- i3.8xlarge
- i3.16xlarge
- r3.large
- r3.xlarge
- r3.2xlarge
- r3.4xlarge
- r3.8xlarge
- r4.large
- r4.xlarge
- r4.2xlarge
- r4.4xlarge
- r4.8xlarge
- r4.16xlarge
- x1.16xlarge
- x1.32xlarge
- p2.large
- p2.xlarge
- p2.8xlarge
- p2.16xlarge
- p3.2xlarge
- p3.8xlarge
- p3.16xlarge
- r5.large
- r5.xlarge
- r5.2xlarge
- r5.4xlarge
- r5.12xlarge
- r5.24xlarge
- r5d.large
- r5d.xlarge
- r5d.2xlarge
- r5d.4xlarge
- r5d.12xlarge
- r5d.24xlarge
- z1d.large
- z1d.xlarge
- z1d.2xlarge
- z1d.3xlarge
- z1d.6xlarge
- z1d.12xlarge

ConstraintDescription: Must be a valid EC2 instance type

NodeVolumeSize:
  Type: Number
  Description: Node volume size
  Default: 50

BootstrapArguments:
  Description: Additional boot arguments to pass to bootstrap script. See files/bootstrap.sh in https://github.com/awslabs/amazon-eks-ami
  Default: --kuberet-exra-args --node-labels=node-role.kubernetes.io/node=true
  Type: String

KeyName:
  Description: The EC2 Key Pair to allow SSH access to the instances
  Type: AWS::EC2::KeyPair::KeyPair

NodeAutoScalingGroupMinSize:
  Type: Number
  Description: Minimum size of Node Group ASG.
  Default: 0

NodeAutoScalingGroupDesiredCapacity:
  Type: Number
  Description: Desired capacity of Node Group ASG.
  Default: 1

NodeAutoScalingGroupMaxSize:
  Type: Number
Description: Maximum size of Node Group ASG. Set to at least 1 greater than NodeAutoScalingGroupDesiredCapacity.
Default: 5

Resources:

NodeInstanceRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Sub ${ClusterName}-nodes-role
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Effect: Allow
        Principal:
          Service:
            - ec2.amazonaws.com
        Action:
          - sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
      - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
      - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
    Path: /

ExternalDnsPolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: !Sub ${ClusterName}-external-dns
    Roles:
      - !Ref NodeInstanceRole
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Effect: Allow
        Action:
          - route53:ListHostedZones
          - route53:ListResourceRecordSets
          - route53:ChangeResourceRecordSets
        Resource: '*'

ClusterAutoscalerPolicy:
  Type: AWS::IAM::Policy
Properties:
  PolicyName: !Sub ${ClusterName}-cluster-autocscaler
  Roles:
    - !Ref NodeInstanceRole
PolicyDocument:
  Version: '2012-10-17'
  Statement:
    - Sid: EksWorkerAutoscalingListAll
      Effect: Allow
      Action:
        - autoscaling:DescribeAutoScalingGroups
        - autoscaling:DescribeAutoScalingInstances
        - autoscaling:DescribeLaunchConfigurations
        - autoscaling:DescribeTags
      Resource: '*'
    - Sid: EksWorkerAutoscalingManageOwned
      Effect: Allow
      Action:
        - autoscaling:SetDesiredCapacity
        - autoscaling:TerminateInstanceInAutoScalingGroup
        - autoscaling:UpdateAutoScalingGroup
      Resource: '*'
      Condition:
        StringEquals:
          autoscaling:ResourceTag/kubernetes.io/cluster: !Ref ClusterName
          autoscaling:ResourceTag/k8s.io/cluster-autoscaler/enabled: 'true'

NodeInstanceProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    InstanceProfileName: !Sub ${ClusterName}-nodes-profile
    Roles:
      - !Ref NodeInstanceRole
    Path: /

NodeSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: !Sub ${ClusterName}-nodes-sg
    GroupDescription: Security group for all nodes in the cluster
    VpcId: !Ref VpcId
    Tags:
      - Key: Name
Value: !Sub ${ClusterName}-nodes-sg
- Key: !Sub kubernetes.io/cluster/${ClusterName}
  Value: owned

NodeSecurityGroupIngress:
  Type: AWS::EC2::SecurityGroupIngress
  DependsOn: NodeSecurityGroup
  Properties:
    GroupId: !Ref NodeSecurityGroup
    SourceSecurityGroupId: !Ref NodeSecurityGroup
    Description: Allow node to communicate with each other
    IpProtocol: '-1'
    FromPort: 0
    ToPort: 65535

NodeSecurityGroupFromControlPlaneIngress:
  Type: AWS::EC2::SecurityGroupIngress
  DependsOn: NodeSecurityGroup
  Properties:
    GroupId: !Ref NodeSecurityGroup
    SourceSecurityGroupId: !Ref ClusterControlPlaneSecurityGroup
    Description: Allow worker Kubelets and pods to receive communication from the cluster control plane
    IpProtocol: tcp
    FromPort: 1025
    ToPort: 65535

ControlPlaneEgressToNodeSecurityGroup:
  Type: AWS::EC2::SecurityGroupEgress
  DependsOn: NodeSecurityGroup
  Properties:
    GroupId: !Ref ClusterControlPlaneSecurityGroup
    DestinationSecurityGroupId: !Ref NodeSecurityGroup
    Description: Allow the cluster control plane to communicate with worker Kubelet and pods
    IpProtocol: tcp
    FromPort: 1025
    ToPort: 65535

NodeSecurityGroupFromControlPlaneOn443Ingress:
  Type: AWS::EC2::SecurityGroupIngress
  DependsOn: NodeSecurityGroup
  Properties:
    GroupId: !Ref NodeSecurityGroup
    SourceSecurityGroupId: !Ref ClusterControlPlaneSecurityGroupOn443
Description: Allow pods running extension API servers on port 443 to receive communication from cluster control plane
  IpProtocol: tcp
  FromPort: 443
  ToPort: 443

ControlPlaneEgressToNodeSecurityGroupOn443:
  Type: AWS::EC2::SecurityGroupEgress
  DependsOn: NodeSecurityGroup
  Properties:
    GroupId: !Ref ClusterControlPlaneSecurityGroup
    DestinationSecurityGroupId: !Ref NodeSecurityGroup
    Description: Allow the cluster control plane to communicate with pods running extension API servers on port 443
    IpProtocol: tcp
    FromPort: 443
    ToPort: 443

ClusterControlPlaneSecurityGroupIngress:
  Type: AWS::EC2::SecurityGroupIngress
  DependsOn: NodeSecurityGroup
  Properties:
    GroupId: !Ref ClusterControlPlaneSecurityGroup
    SourceSecurityGroupId: !Ref NodeSecurityGroup
    Description: Allow pods to communicate with the cluster API server
    IpProtocol: tcp
    ToPort: 443
    FromPort: 443

NodeLaunchConfig:
  Type: AWS::AutoScaling::LaunchConfiguration
  Properties:
    LaunchConfigurationName: !Sub ${ClusterName}-nodes
    IamInstanceProfile: !Ref NodeInstanceProfile
    ImageId: !Ref NodeImageId
    InstanceType: !Ref NodeInstanceType
    KeyName: !Ref KeyName
    SecurityGroups:
      - !Ref NodeSecurityGroup
    BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: !Ref NodeVolumeSize
          VolumeType: gp2
DeleteOnTermination: true

UserData:
Fn::Base64:
!Sub |
#!/bin/bash
set -o xtrace
/etc/eks/bootstrap.sh ${ClusterName} ${BootstrapArguments}

NodeGroup1:
Type: AWS::AutoScaling::AutoScalingGroup
UpdatePolicy:
  AutoScalingScheduledAction:
    IgnoreUnmodifiedGroupSizeProperties: true
  AutoScalingRollingUpdate:
    MinInstancesInService: !Ref NodeAutoScalingGroupDesiredCapacity
    MaxBatchSize: !Ref NodeAutoScalingGroupMaxSize
    PauseTime: PT15M
    SuspendProcesses:
      - AlarmNotification
      - ReplaceUnhealthy
      - ScheduledActions
      - Terminate
    WaitOnResourceSignals: false
Properties:
  AutoScalingGroupName: !Sub ${ClusterName}-nodes-a
  LaunchConfigurationName: !Ref NodeLaunchConfig
  DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
  MinSize: !Ref NodeAutoScalingGroupMinSize
  MaxSize: !Ref NodeAutoScalingGroupMaxSize
  HealthCheckGracePeriod: 300
  HealthCheckType: EC2
  VPCZoneIdentifier: [!Select [0, !Ref Subnets]]
  TerminationPolicies: ['OldestInstance']
Tags:
  - Key: Name
    Value: !Sub ${ClusterName}-nodes-a
    PropagateAtLaunch: true
  - Key: kubernetes.io/cluster
    Value: !Ref ClusterName
    PropagateAtLaunch: true
  - Key: kubernetes.io/cluster/${ClusterName}
    Value: owned
    PropagateAtLaunch: true
- Key: k8s.io/cluster-autoscaler/enabled
  PropagateAtLaunch: false
  Value: true

NodeGroup2:
  Type: AWS::AutoScaling::AutoScalingGroup
  UpdatePolicy:
    AutoScalingScheduledAction:
      IgnoreUnmodifiedGroupSizeProperties: true
    AutoScalingRollingUpdate:
      MinInstancesInService: !Ref
      NodeAutoScalingGroupDesiredCapacity
    MaxBatchSize: !Ref NodeAutoScalingGroupMaxSize
    PauseTime: PT15M
    SuspendProcesses:
      - AlarmNotification
      - ReplaceUnhealthy
      - ScheduledActions
      - Terminate
    WaitOnResourceSignals: false
  Properties:
    AutoScalingGroupName: !Sub ${ClusterName}-nodes
    LaunchConfigurationName: !Ref NodeLaunchConfig
    DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
    MinSize: !Ref NodeAutoScalingGroupMinSize
    MaxSize: !Ref NodeAutoScalingGroupMaxSize
    HealthCheckGracePeriod: 300
    HealthCheckType: EC2
    VPCZoneIdentifier: ![Select [1, !Ref Subnets]]
    TerminationPolicies: ['OldestInstance']
  Tags:
    - Key: Name
      Value: !Sub ${ClusterName}-nodes
      PropagateAtLaunch: true
    - Key: kubernetes.io/cluster
      Value: !Ref ClusterName
      PropagateAtLaunch: true
    - Key: !Sub kubernetes.io/cluster/${ClusterName}
      Value: owned
      PropagateAtLaunch: false
      Value: true
    - Key: k8s.io/cluster-autoscaler/enabled
      PropagateAtLaunch: false
      Value: true
Outputs:

NodeInstanceRole:
  Description: The node instance role
  Value: !GetAtt NodeInstanceRole.Arn

NodeSecurityGroup:
  Description: The security group for the node group
  Value: !Ref NodeSecurityGroup

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Setup simple VPN to access private subnets
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
        default: VPN Network Configuration
          Parameters:
            - VpcId
            - PublicSubnet
      - Label:
        default: EKS Worker Security Configuration
          Parameters:
            - NodeSecurityGroup
      - Label:
        default: VPN Instance Configuration
          Parameters:
            - VpnImageId
            - VpnInstanceType
            - KeyName

Parameters:

VpcId:
  Description: The VPC of EKS
  Type: AWS::EC2::VPC::Id
```
PublisSubnet:
  Description: Public subnets where to deploy VPN server
  Type: AWS::EC2::Subnet::Id

NodeSecurityGroup:
  Description: The security group of the cluster control plane.
  Type: AWS::EC2::SecurityGroup::Id

VpnImageId:
  Type: AWS::EC2::Image::Id
  Description: EKS AMI id for the node instances, to find for your region perform command `aws ec2 describe-images --owners 679593333241 --filters "Name=name,Values=SoftEther L2TP/IPsec VPN*" --query 'sort_by(Images, &CreationDate)[].ImageId'`

VpnInstanceType:
  Description: EC2 instance type for the VPN instance
  Type: String
  Default: t2.small
  AllowedValues:
    - t2.small
    - t2.medium
    - t2.large
    - m5.large
    - c4.large
  ConstraintDescription: Must be a valid EC2 instance type

KeyName:
  Description: The EC2 Key Pair to allow SSH access to the instances
  Type: AWS::EC2::KeyPair::KeyName

Resources:

VpnSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    VpcId: !Ref VpcId
    GroupName: vpn-to-eks-nodes
    GroupDescription: VPN security group
    SecurityGroupIngress:
      - CidrIp: 0.0.0.0/0
        IpProtocol: tcp
        FromPort: 22
ToPort: 22
- CidrIp: 0.0.0.0/0
  IpProtocol: udp
  FromPort: 500
  ToPort: 500
- CidrIp: 0.0.0.0/0
  IpProtocol: udp
  FromPort: 1701
  ToPort: 1701
- CidrIp: 0.0.0.0/0
  IpProtocol: udp
  FromPort: 4500
  ToPort: 4500

SecurityGroupEgress:
- CidrIp: 0.0.0.0/0
  IpProtocol: ^-1^'

Tags:
- Key: Name
  Value: vpn-to-eks-nodes

NodeSecurityGroupIngress:
  Type: AWS::EC2::SecurityGroupIngress
  DependsOn: VPNSecurityGroup
  Properties:
    Description: Allow access to nodes from VPN
    GroupId: !Ref NodeSecurityGroup
    SourceSecurityGroupId: !Ref VPNSecurityGroup
    IpProtocol: ^-1^'

VpnInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref VpnImageId
    InstanceType: !Ref VpnInstanceType
    KeyName: !Ref KeyName
    SecurityGroupIds:
      - !Ref VPNSecurityGroup
    SubnetId: !Ref PublisSubnet
  Tags:
    - Key: Name
      Value: vpn-to-eks-nodes

VpnEIP:
  Type: AWS::EC2::EIP
DependsOn: VpnInstance
Properties:
  InstanceId: !Ref VpnInstance

Outputs:

VpnEndpoint:
  Description: VPN IP address
  Value: !Ref VpnEIP

VpnUser:
  Description: VPN User name
  Value: vpn

VpnPass:
  Description: Password for VPN
  Value: !Ref VpnInstance

elasticsearch-export-job.yaml

apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: elasticsearch-export-job
spec:
  schedule: "0 6 * * *"
jobTemplate:
  spec:
    backoffLimit: 5
template:
  spec:
    restartPolicy: Never
    containers:
      - name: elasticsearch-dump
        image: taskrabbit/elasticsearch-dump:v4.3.0
        env:
          - name: ELASTIC_HOST
            value: "elk-elasticsearch-client:9200"
          - name: BUCKET
            value: "elasticsearch-dump-bucket"
          - name: ACCESS_KEY
value: "<ACCESS_KEY>"
- name: SECRET_ACCESS_KEY
  value: "<SECRET_ACCESS_KEY>"
command: ["/bin/sh","-c"]
args:
  - FROM_DATE=${FROM_DATE:-$(date -d@"$(date +%s-86400)" '+%Y-%m-%d') } && TO_DATE=${TO_DATE:-$(date '+%Y-%m-%d')} "elasticdump --input=http://$(ELASTIC_HOST)/logstash-* --searchBody="{"query":{"range":{"@timestamp":{"gte":"${FROM_DATE}","lte":"${TO_DATE}"}}}" --s3Bucket "$(BUCKET)" --s3AccessKeyId "$(ACCESS_KEY)" --s3SecretAccessKey "$(SECRET_ACCESS_KEY)" --s3RecordKey "elasticsearch-backup-${FROM_DATE}-${TO_DATE}.json" --s3Compress"