Prerequisites

Third party software prerequisites

1. MongoDB schema with initial data

   For the CAN application to come up successfully, it needs some initial data in MongoDB. Usually this is provided by the customer on their premises. However if it is not available, then Avanseus delivery needs to be contacted for an initial data dump that needs to be exported to a MongoDB schema. MongoDB can either be within the Openshift platform or outside. This MongoDB schema must be accessible from the CAN operator pods preferably on port 27017.

2. LDAP with initial user data

   For the CAN application, authentication is made possible using LDAP. Usually customers provide an LDAP connection for integration. However, if LDAP is not made available by the customer, then Avanseus delivery team needs to be requested to get a lightweight LDAP software. One initial user would come with this LDAP. LDAP can either be within the Openshift platform or outside. This LDAP must be accessible from the CAN operator pods preferably on port 1389.
   Note: LDAP Docker image will be provided only when the customer doesn't have their LDAP for integration.

Configuration prerequisites

1. Image pull secrets

   This is an optional step. Please contact Avanseus delivery team if this step is needed. Operators must be given a provision to download the operator image & all other docker images from Avanseus private repository: avanseuscontainer.com.
   Obtain the user credentials from Avanseus Team and create a secret in Openshift platform as follows:

   ```
   oc create secret docker-registry avanseus-docker-registry-secret --docker-server=avanseuscontainer.com
   --docker-username=<username> --docker-password=<password> --docker-email=<email_id>
   ```

   Replace the part enclosed with **<>** with necessary information to create the secret "avanseus-docker-registry-secret".

2. PersistenceVolumeClaim to copy WAR files

By default, the CAN operator does not come with CAN binary. Before deploying, the binaries in WAR format need to be obtained separately and put into a PersistentVolume. Below is a method to do this.

Create a PersistenceVolumeClaim “tmp-data”. Below is an example configuration YAML

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: tmp-data
spec:
  accessModes:
    - ReadWriteOnce
  resources:
   requests:
    storage: 1G
```

Create a dummy application (Apache server) and mount the created PersistenceVolumeClaim. Below is the YAML content

```
apiVersion: v1
kind: Pod
metadata:
 name: pod-httpd
 labels:
   app: apache_webserver
spec:
 containers:
   - name: cntr-httpd
     image: httpd:latest
     ports:
       - containerPort: 80
     volumeMounts:
     - mountPath: /mnt
       name: tmp-data
 volumes:
 - name: tmp-data
   persistentVolumeClaim:
    claimName: tmp-data
```

Change directory to the place where WAR files are available. Copy the WAR files provided by Avanseus into the mount point "/mnt" using the rsync command.

```
oc rsync . pod-httpd:/mnt
```

Once this is done, the dummy application pod can be stopped. Remember that "tmp-data" PersistentVolume must not be deleted as this is used by Operator components to pull the binaries.

3. PersistenceVolumeClaim for CAN application

Create a PersistenceVolumeClaim named “data”. This is needed for the CAN application to write files. YAML configuration is given below.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: data
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5G
```

4. Create Route configuration to access application over web

   Below is the YAML content of file “canroute.yaml” to create a Route for CAN web application.

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    openshift.io/host.generated: "true"
  labels:
    app: can-pod
  name: can
  namespace: default
spec:
  host: can.<ROUTER_DEFAULT_HOSTNAME>
  path: /CAN/
  port:
    targetPort: can-port
  tls:
    insecureEdgeTerminationPolicy: None
    termination: edge
  to:
    kind: Service
    name: can-pod
    weight: 100
  wildcardPolicy: None
```

   Below is the YAML content of file “casroute.yaml” to create a Route for CAS web application.

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    openshift.io/host.generated: "true"
  labels:
    app: can-pod
  name: cas
```

```
  namespace: default
spec:
  host: can.<ROUTER_DEFAULT_HOSTNAME>
  path: /CAS/
  port:
    targetPort: cas-port
  tls:
    insecureEdgeTerminationPolicy: None
    termination: edge
  to:
    kind: Service
    name: can-pod
    weight: 100
  wildcardPolicy: None
```

In both the above mentioned YAML files, replace the <ROUTER_DEFAULT_HOSTNAME> with the default router host of the Openshift installation. If "***apps.avanseus1.avanseuscandev.com***" was the router default hostname, then the configuration line in the YAML file would be as shown below:

```
host: can.apps.avanseus1.avanseuscandev.com
```

5. Create ConfigMaps with necessary information

Pods within the operator need configurations to startup the application. These configurations have to be stored in Config maps on Openshift platform. CAN operator comes with 5 applications which would run in 5 different pods:

- CAN (Needs config map)
- CAN worker - 1 (Needs config map)
- CAN worker - 2 (Needs config map)
- CAN worker - 3 (Needs config map)
- VBI

Below are the ConfigMap files that needs to be imported:

`can-master-config.yaml` - Master config file:

```
apiVersion: v1
data:
 catalina.properties: "# Licensed to the Apache Software Foundation (ASF) under one
  or more\r\n# contributor license agreements.  See the NOTICE file distributed
  with\r\n# this work for additional information regarding copyright ownership.\r\n#
  The ASF licenses this file to You under the Apache License, Version 2.0\r\n# (the
  \"License\"); you may not use this file except in compliance with\r\n# the License.
  \ You may obtain a copy of the License at\r\n#\r\n#     http://www.apache.org/licenses/LICENSE-2.0\r\n#\r\n#
  Unless required by applicable law or agreed to in writing, software\r\n# distributed
  under the License is distributed on an \"AS IS\" BASIS,\r\n# WITHOUT WARRANTIES
  OR CONDITIONS OF ANY KIND, either express or implied.\r\n# See the License for
  the specific language governing permissions and\r\n# limitations under the License.\r\n\r\n#\r\n#
  List of comma-separated packages that start with or equal this string\r\n# will
```

```
 cause a security exception to be thrown when\r\n# passed to checkPackageAccess
 unless the\r\n# corresponding RuntimePermission (\"accessClassInPackage.\"+package)
 has\r\n# been
granted.\r\npackage.access=sun.,org.apache.catalina.,org.apache.coyote.,org.apache.jasper.,org.apache.tomcat.\r
\n#\r\n#
 List of comma-separated packages that start with or equal this string\r\n# will
 cause a security exception to be thrown when\r\n# passed to checkPackageDefinition
 unless the\r\n# corresponding RuntimePermission (\"defineClassInPackage.\"+package)
 has\r\n# been granted.\r\n#\r\n# by default, no packages are restricted for definition,
 and none of\r\n# the class loaders supplied with the JDK call
checkPackageDefinition.\r\n#\r\npackage.definition=sun.,java.,org.apache.catalina.,org.apache.coyote.,\\\r\norg.ap
ache.jasper.,org.apache.naming.,org.apache.tomcat.\r\n\r\n#\r\n#\r\n#
 List of comma-separated paths defining the contents of the \"common\"\r\n# classloader.
 Prefixes should be used to define what is the repository type.\r\n# Path may be
 relative to the CATALINA_HOME or CATALINA_BASE path or absolute.\r\n# If left
 as blank,the JVM system loader will be used as Catalina's \"common\"\r\n# loader.\r\n#
 Examples:\r\n#     \"foo\": Add this folder as a class repository\r\n#     \"foo/*.jar\":
 Add all the JARs of the specified folder as class\r\n#                     repositories\r\n#
 \    \"foo/bar.jar\": Add bar.jar as a class repository\r\n#\r\n# Note: Values
 are enclosed in double quotes (\"...\") in case either the\r\n#       ${catalina.base}
 path or the ${catalina.home} path contains a comma.\r\n#       Because double
 quotes are used for quoting, the double quote character\r\n#       may not appear
 in a
path.\r\ncommon.loader=\"${catalina.base}/lib\",\"${catalina.base}/lib/*.jar\",\"${catalina.home}/lib\",\"${catalina.hom
e}/lib/*.jar\"\r\n\r\n#\r\n#
 List of comma-separated paths defining the contents of the \"server\"\r\n# classloader.
 Prefixes should be used to define what is the repository type.\r\n# Path may be
 relative to the CATALINA_HOME or CATALINA_BASE path or absolute.\r\n# If left
 as blank, the \"common\" loader will be used as Catalina's \"server\"\r\n# loader.\r\n#
 Examples:\r\n#     \"foo\": Add this folder as a class repository\r\n#     \"foo/*.jar\":
 Add all the JARs of the specified folder as class\r\n#                     repositories\r\n#
 \    \"foo/bar.jar\": Add bar.jar as a class repository\r\n#\r\n# Note: Values
 may be enclosed in double quotes (\"...\") in case either the\r\n#       ${catalina.base}
 path or the ${catalina.home} path contains a comma.\r\n#       Because double
 quotes are used for quoting, the double quote character\r\n#       may not appear
 in a path.\r\nserver.loader=\r\n\r\n#\r\n# List of comma-separated paths defining
 the contents of the \"shared\"\r\n# classloader. Prefixes should be used to define
 what is the repository type.\r\n# Path may be relative to the CATALINA_BASE path
 or absolute. If left as blank,\r\n# the \"common\" loader will be used as Catalina's
 \"shared\" loader.\r\n# Examples:\r\n#     \"foo\": Add this folder as a class
 repository\r\n#     \"foo/*.jar\": Add all the JARs of the specified folder as
 class\r\n#                     repositories\r\n#     \"foo/bar.jar\": Add bar.jar
 as a class repository\r\n# Please note that for single jars, e.g. bar.jar, you
 need the URL form\r\n# starting with file:.\r\n#\r\n# Note: Values may be enclosed
 in double quotes (\"...\") in case either the\r\n#       ${catalina.base} path
 or the ${catalina.home} path contains a comma.\r\n#       Because double quotes
 are used for quoting, the double quote character\r\n#       may not appear in
 a path.\r\nshared.loader=\r\n\r\n# Default list of JAR files that should not be
 scanned using the JarScanner\r\n# functionality. This is typically used to scan
 JARs for configuration\r\n# information. JARs that do not contain such information
 may be excluded from\r\n# the scan to speed up the scanning process. This is the
 default list. JARs on\r\n# this list are excluded from all scans. The list must
 be a comma separated list\r\n# of JAR file names.\r\n# The list of JARs to skip
 may be over-ridden at a Context level for individual\r\n# scan types by configuring
 a JarScanner with a nested JarScanFilter.\r\n# The JARs listed below include:\r\n#
 - Tomcat Bootstrap JARs\r\n# - Tomcat API JARs\r\n# - Catalina JARs\r\n# - Jasper
 JARs\r\n# - Tomcat JARs\r\n# - Common non-Tomcat JARs\r\n# - Test JARs (JUnit,
```

```
  Cobertura and
dependencies)\r\ntomcat.util.scan.StandardJarScanFilter.jarsToSkip=\\\r\nbootstrap.jar,commons-daemon.jar,tomc
at-juli.jar,\\\r\nannotations-api.jar,el-api.jar,jsp-api.jar,servlet-api.jar,websocket-api.jar,\\\r\njaspic-api.jar,\\\r\ncatalin
a.jar,catalina-ant.jar,catalina-ha.jar,catalina-storeconfig.jar,\\\r\ncatalina-tribes.jar,\\\r\njasper.jar,jasper-el.jar,ecj-*.ja
r,\\\r\ntomcat-api.jar,tomcat-util.jar,tomcat-util-scan.jar,tomcat-coyote.jar,\\\r\ntomcat-dbcp.jar,tomcat-jni.jar,tomcat-
websocket.jar,\\\r\ntomcat-i18n-en.jar,tomcat-i18n-es.jar,tomcat-i18n-fr.jar,tomcat-i18n-ja.jar,\\\r\ntomcat-juli-adapte
rs.jar,catalina-jmx-remote.jar,catalina-ws.jar,\\\r\ntomcat-jdbc.jar,\\\r\ntools.jar,\\\r\ncommons-beanutils*.jar,common
s-codec*.jar,commons-collections*.jar,\\\r\ncommons-dbcp*.jar,commons-digester*.jar,commons-fileupload*.jar,\\\r\
ncommons-httpclient*.jar,commons-io*.jar,commons-lang*.jar,commons-logging*.jar,\\\r\ncommons-math*.jar,com
mons-pool*.jar,\\\r\njstl.jar,taglibs-standard-spec-*.jar,\\\r\ngeronimo-spec-jaxrpc*.jar,wsdl4j*.jar,\\\r\nant.jar,ant-junit
*.jar,aspectj*.jar,jmx.jar,h2*.jar,hibernate*.jar,httpclient*.jar,\\\r\njmx-tools.jar,jta*.jar,log4j*.jar,mail*.jar,slf4j*.jar,\\\r\n
xercesImpl.jar,xmlParserAPIs.jar,xml-apis.jar,\\\r\njunit.jar,junit-*.jar,hamcrest-*.jar,easymock-*.jar,cglib-*.jar,\\\r\nob
jenesis-*.jar,ant-launcher.jar,\\\r\ncobertura-*.jar,asm-*.jar,dom4j-*.jar,icu4j-*.jar,jaxen-*.jar,jdom-*.jar,\\\r\njetty-*.jar,
oro-*.jar,servlet-api-*.jar,tagsoup-*.jar,xmlParserAPIs-*.jar,\\\r\nxom-*.jar\r\n\r\n#
  Default list of JAR files that should be scanned that overrides the default\r\n#
  jarsToSkip list above. This is typically used to include a specific JAR that\r\n#
  has been excluded by a broad file name pattern in the jarsToSkip list.\r\n# The
  list of JARs to scan may be over-ridden at a Context level for individual\r\n#
  scan types by configuring a JarScanner with a nested
JarScanFilter.\r\ntomcat.util.scan.StandardJarScanFilter.jarsToScan=\\\r\nlog4j-web*.jar,log4j-taglib*.jar,log4javasc
ript*.jar,slf4j-taglib*.jar\r\n\r\n#
  String cache
configuration.\r\ntomcat.util.buf.StringCache.byte.enabled=true\r\n#tomcat.util.buf.StringCache.char.enabled=true\r
\n#tomcat.util.buf.StringCache.trainThreshold=500000\r\n#tomcat.util.buf.StringCache.cacheSize=5000\r\n\r\n#Ke
y\r\nPASSWORD_KEY=<AVANSEUS_PASSWORD_KEY>\r\n"
config.properties: |
  #Domain details
  avanseus.app.cas.domain=<APP_DOMAIN>
  avanseus.app.can.domain=<APP_DOMAIN>

  avanseus.vbi.ip=<VBI_IP>
  avanseus.vbi.port=30004

  #Domain protocol
  avanseus.protocol=https

  #Host details
  avanseus.app.cas.host=127.0.0.1:2003
  avanseus.app.can.host=127.0.0.1:2000

  #MongoDb configuration
  avanseus.mongodb.ip=<MONGODB_IP>
  avanseus.mongodb.port=<MONGODB_PORT>
  avanseus.mongodb.username=<MONGODB_USERNAME>
  avanseus.mongodb.password=<MONGODB_PASSWORD>
  avanseus.mongodb.dbName=<MONGODB_SCHEMA_NAME>

  avanseus.mongodb.admin.username=admin
  avanseus.mongodb.admin.password=<MONGODB_ADMIN_PASSWORD>
  avanseus.mongodb.admin.dbName=admin

  #log configuration
  avanseus.log.path=/data/workspace/logs/
  avanseus.log.thread.poolsize=20

  #LDAP server configuration
  avanseus.ldap.organisation=<LDAP_ORG>
```

```
  avanseus.ldap.domain=<LDAP_DOMAIN>
  avanseus.ldap.url1=ldap://<LDAP_URL1>:<LDAP_PORT1>
  avanseus.ldap.url2=ldap://<LDAP_URL2>:<LDAP_PORT2>
  avanseus.ldap.userDn=cn=Directory Manager
  avanseus.ldap.password=<LDAP_PASSWORD>

  #NAS path
  avanseus.app.nas.path=/data/workspace/nasmount/

  #Email Config
  avanseus.email.fromId=<EMAIL_FROM_ID>
  avanseus.email.pwd=<EMAIL_PASSWORD>
  avanseus.email.host=<EMAIL_SERVER>
  avanseus.email.fromName=<EMAIL_SENDER_NAME>
  avanseus.email.port=<EMAIL_PORT>

  #Cluster node configuration
  avanseus.predictionNode.name=nodeA
 setenv.sh: |
  export JAVA_OPTS="$JAVA_OPTS -Dapplication.nasmount=/data/workspace/nasmount/
-Dmongo.host=<MONGODB_IP> -Dmongo.port=<MONGODB_PORT>  -Djava.security.manager
-Djava.security.policy=${CATALINA_BASE}/conf/security.policy"
  export TOMCAT_OPTS="$TOMCAT_OPTS -Djava.security.debug=access,failure"
kind: ConfigMap
metadata:
 creationTimestamp: null
 managedFields:
 - apiVersion: v1
   fieldsType: FieldsV1
   fieldsV1:
     f:data:
       .: {}
       f:catalina.properties: {}
       f:config.properties: {}
       f:setenv.sh: {}
   manager: oc
   operation: Update
   time: "2020-09-23T07:08:13Z"
 name: can-master-config
 selfLink: /api/v1/namespaces/default/configmaps/can-master-config
```

`can-slave-config-node-b.yaml` - NODE-B config file:

```
apiVersion: v1
data:
 config.properties: |
  #Domain details
  avanseus.app.cas.domain=<APP_DOMAIN>
  avanseus.app.can.domain=<APP_DOMAIN>

  avanseus.vbi.ip=<VBI_IP>
  avanseus.vbi.port=30004

  #Domain protocol
  avanseus.protocol=https
```

```
  #Host details
  avanseus.app.cas.host=127.0.0.1:2003
  avanseus.app.can.host=127.0.0.1:2000

  #MongoDb configuration
  avanseus.mongodb.ip=<MONGODB_IP>
  avanseus.mongodb.port=<MONGODB_PORT>
  avanseus.mongodb.username=<MONGODB_USERNAME>
  avanseus.mongodb.password=<MONGODB_PASSWORD>
  avanseus.mongodb.dbName=<MONGODB_SCHEMA_NAME>

  avanseus.mongodb.admin.username=admin
  avanseus.mongodb.admin.password=<MONGODB_ADMIN_PASSWORD>
  avanseus.mongodb.admin.dbName=admin

  #log configuration
  avanseus.log.path=/data/workspace/logs/
  avanseus.log.thread.poolsize=20

  #LDAP server configuration
  avanseus.ldap.organisation=<LDAP_ORG>
  avanseus.ldap.domain=<LDAP_DOMAIN>
  avanseus.ldap.url1=ldap://<LDAP_URL1>:<LDAP_PORT1>
  avanseus.ldap.url2=ldap://<LDAP_URL2>:<LDAP_PORT2>
  avanseus.ldap.userDn=cn=Directory Manager
  avanseus.ldap.password=<LDAP_PASSWORD>

  #NAS path
  avanseus.app.nas.path=/data/workspace/nasmount/

  #Email Config
  avanseus.email.fromId=<EMAIL_FROM_ID>
  avanseus.email.pwd=<EMAIL_PASSWORD>
  avanseus.email.host=<EMAIL_SERVER>
  avanseus.email.fromName=<EMAIL_SENDER_NAME>
  avanseus.email.port=<EMAIL_PORT>

  #Cluster node configuration
  avanseus.predictionNode.name=nodeB
kind: ConfigMap
metadata:
 creationTimestamp: null
 managedFields:
 - apiVersion: v1
   fieldsType: FieldsV1
   fieldsV1:
     f:data:
       .: {}
       f:config.properties: {}
   manager: oc
   operation: Update
   time: "2020-10-11T10:32:15Z"
 name: can-slave-config-node-b
 selfLink: /api/v1/namespaces/default/configmaps/can-slave-config-node-b
```

`can-slave-config-node-c.yaml` - Node-C config file

```
apiVersion: v1
data:
 config.properties: |
  #Domain details
  avanseus.app.cas.domain=<APP_DOMAIN>
  avanseus.app.can.domain=<APP_DOMAIN>

  avanseus.vbi.ip=<VBI_IP>
  avanseus.vbi.port=30004

  #Domain protocol
  avanseus.protocol=https

  #Host details
  avanseus.app.cas.host=127.0.0.1:2003
  avanseus.app.can.host=127.0.0.1:2000

  #MongoDb configuration
  avanseus.mongodb.ip=<MONGODB_IP>
  avanseus.mongodb.port=<MONGODB_PORT>
  avanseus.mongodb.username=<MONGODB_USERNAME>
  avanseus.mongodb.password=<MONGODB_PASSWORD>
  avanseus.mongodb.dbName=<MONGODB_SCHEMA_NAME>

  avanseus.mongodb.admin.username=admin
  avanseus.mongodb.admin.password=<MONGODB_ADMIN_PASSWORD>
  avanseus.mongodb.admin.dbName=admin

  #log configuration
  avanseus.log.path=/data/workspace/logs/
  avanseus.log.thread.poolsize=20

  #LDAP server configuration
  avanseus.ldap.organisation=<LDAP_ORG>
  avanseus.ldap.domain=<LDAP_DOMAIN>
  avanseus.ldap.url1=ldap://<LDAP_URL1>:<LDAP_PORT1>
  avanseus.ldap.url2=ldap://<LDAP_URL2>:<LDAP_PORT2>
  avanseus.ldap.userDn=cn=Directory Manager
  avanseus.ldap.password=<LDAP_PASSWORD>

  #NAS path
  avanseus.app.nas.path=/data/workspace/nasmount/

  #Email Config
  avanseus.email.fromId=<EMAIL_FROM_ID>
  avanseus.email.pwd=<EMAIL_PASSWORD>
  avanseus.email.host=<EMAIL_SERVER>
  avanseus.email.fromName=<EMAIL_SENDER_NAME>
  avanseus.email.port=<EMAIL_PORT>

  #Cluster node configuration
  avanseus.predictionNode.name=nodeC
kind: ConfigMap
metadata:
 creationTimestamp: null
```

```
managedFields:
- apiVersion: v1
  fieldsType: FieldsV1
  fieldsV1:
    f:data:
      .: {}
      f:config.properties: {}
  manager: oc
  operation: Update
  time: "2020-10-11T10:32:15Z"
name: can-slave-config-node-c
selfLink: /api/v1/namespaces/default/configmaps/can-slave-config-node-c
```

`can-slave-config-node-d.yaml` - Node-D config file

```
apiVersion: v1
data:
 config.properties: |
  #Domain details
  avanseus.app.cas.domain=<APP_DOMAIN>
  avanseus.app.can.domain=<APP_DOMAIN>

  avanseus.vbi.ip=<VBI_IP>
  avanseus.vbi.port=30004

  #Domain protocol
  avanseus.protocol=https

  #Host details
  avanseus.app.cas.host=127.0.0.1:2003
  avanseus.app.can.host=127.0.0.1:2000

  #MongoDb configuration
  avanseus.mongodb.ip=<MONGODB_IP>
  avanseus.mongodb.port=<MONGODB_PORT>
  avanseus.mongodb.username=<MONGODB_USERNAME>
  avanseus.mongodb.password=<MONGODB_PASSWORD>
  avanseus.mongodb.dbName=<MONGODB_SCHEMA_NAME>

  avanseus.mongodb.admin.username=admin
  avanseus.mongodb.admin.password=<MONGODB_ADMIN_PASSWORD>
  avanseus.mongodb.admin.dbName=admin

  #log configuration
  avanseus.log.path=/data/workspace/logs/
  avanseus.log.thread.poolsize=20

  #LDAP server configuration
  avanseus.ldap.organisation=<LDAP_ORG>
  avanseus.ldap.domain=<LDAP_DOMAIN>
  avanseus.ldap.url1=ldap://<LDAP_URL1>:<LDAP_PORT1>
  avanseus.ldap.url2=ldap://<LDAP_URL2>:<LDAP_PORT2>
  avanseus.ldap.userDn=cn=Directory Manager
  avanseus.ldap.password=<LDAP_PASSWORD>

  #NAS path
```

```
  avanseus.app.nas.path=/data/workspace/nasmount/

  #Email Config
  avanseus.email.fromId=<EMAIL_FROM_ID>
  avanseus.email.pwd=<EMAIL_PASSWORD>
  avanseus.email.host=<EMAIL_SERVER>
  avanseus.email.fromName=<EMAIL_SENDER_NAME>
  avanseus.email.port=<EMAIL_PORT>

  #Cluster node configuration
  avanseus.predictionNode.name=nodeD
kind: ConfigMap
metadata:
 creationTimestamp: null
 managedFields:
 - apiVersion: v1
   fieldsType: FieldsV1
   fieldsV1:
     f:data:
       .: {}
       f:config.properties: {}
   manager: oc
   operation: Update
   time: "2020-10-11T10:32:15Z"
 name: can-slave-config-node-d
 selfLink: /api/v1/namespaces/default/configmaps/can-slave-config-node-d
```

In the above mentioned `YAML` files replace the text enclosed in `<>` with necessary information. Below are their details:

- `<AVANSEUS_PASSWORD_KEY>` : Replace this with the password key to be used. Request the Avanseus delivery team for this. This is very specific for a given DB dump.
- `<APP_DOMAIN>` : Replace this with the application domain name. This must be updated with the host name configured in the Route configuration.
- `<VBI_IP>` : Replace this with the VBI process IP. This must be updated with the Openshift cluster's Master node IP.
- `<MONGODB_IP>` : Replace this with MongoDB IP which is reachable from the pods.
- `<MONGODB_PORT>` : Replace this with MongoDB Port which is reachable from the pods.
- `<MONGODB_USERNAME>` : Replace this with MongoDB schema read/write username.
- `<MONGODB_PASSWORD>` : Replace this with above username's password (Encrypted). The encryption technique is defined in the Password encryption document. Please refer to that.
- `<MONGODB_SCHEMA_NAME>` : Replace this with DB schema name.
- `<MONGODB_ADMIN_PASSWORD>` : Replace this with MongoDB admin password (Encrypted). The encryption technique is defined in the Password encryption document. Please refer to that..
- `<LDAP_URL1>` : Replace this with LDAP URL 1. LDAP IP reachable from pods.
- `<LDAP_URL2>` : Replace this with LDAP URL 2. Same as above if the customer does not provide failover LDAP URL.
- `<LDAP_PORT1>` : Replace this with LDAP Port 1. LDAP port reachable from pods.

- `<LDAP_PORT2>` : Replace this with LDAP Port 2. Same as above if the customer does not provide failover LDAP instance.
- `<LDAP_PASSWORD>` : Replace this with LDAP password (Encrypted). The encryption technique is defined here in the Password encryption document. Please refer to that.
- `<LDAP_ORG>` : Replace this with LDAP organisation. Customers usually provide this or contact Avanseus support team for this.
- `<LDAP_DOMAIN>` : Replace this with LDAP domain. Customers usually provide this or contact Avanseus support team for this.
- `<EMAIL_FROM_ID>` : Replace this with Email ID of sender.
- `<EMAIL_SENDER_NAME>` : Replace this with Email sender's name.
- `<EMAIL_SERVER>` : Replace this with Email server host or domain.
- `<EMAIL_PORT>` : Replace this with Email server port.
- `<EMAIL_PASSWORD>` : Replace this with Email sender's password.

The last 5 entries must be filled up with relevant SMTP server or relay server information which is reachable from the Openshift cluster.

Installation

Operator would be available in the OperatorHub. Below is a link to the video which shows how to install the CAN operator.

https://avanseuscandev.com/releases/CAN/5.0/Videos/can-operator-openshift-installation-guide.mp4

Post installation

After installation, CAN application would be available on following URL:

https://can.<ROUTER_DEFAULT_HOSTNAME>/CAN/

If CAN tool was integrated with LDAP on customer premise, then the user credentials would be known.
If CAN tool was integrated with Avanseus's light weight LDAP, then request the Avanseus delivery team for credentials.