ZALONI DATA PLATFORM

Installation Guide

Release 5.0.2

Zaloni Inc.

Jan 10, 2019

CONTENTS

The Zaloni Data Platform (ZDP) is a comprehensive, integrated solution that operationalizes data processes along the entire pipeline from data source to data consumer.

The Installation guide details the installation process for the ZDP application. This guide also includes the various troubleshooting scenarios that you may encounter during installation.

Important: The reference links used in this guide may not work. Refer the in-product help files to access the end-to-end documentation for ZDP.

CHAPTER

ONE

INSTALLATION GUIDE

1.1 Introduction

This guide provides instructions to install and configure the Zaloni Data Platform.

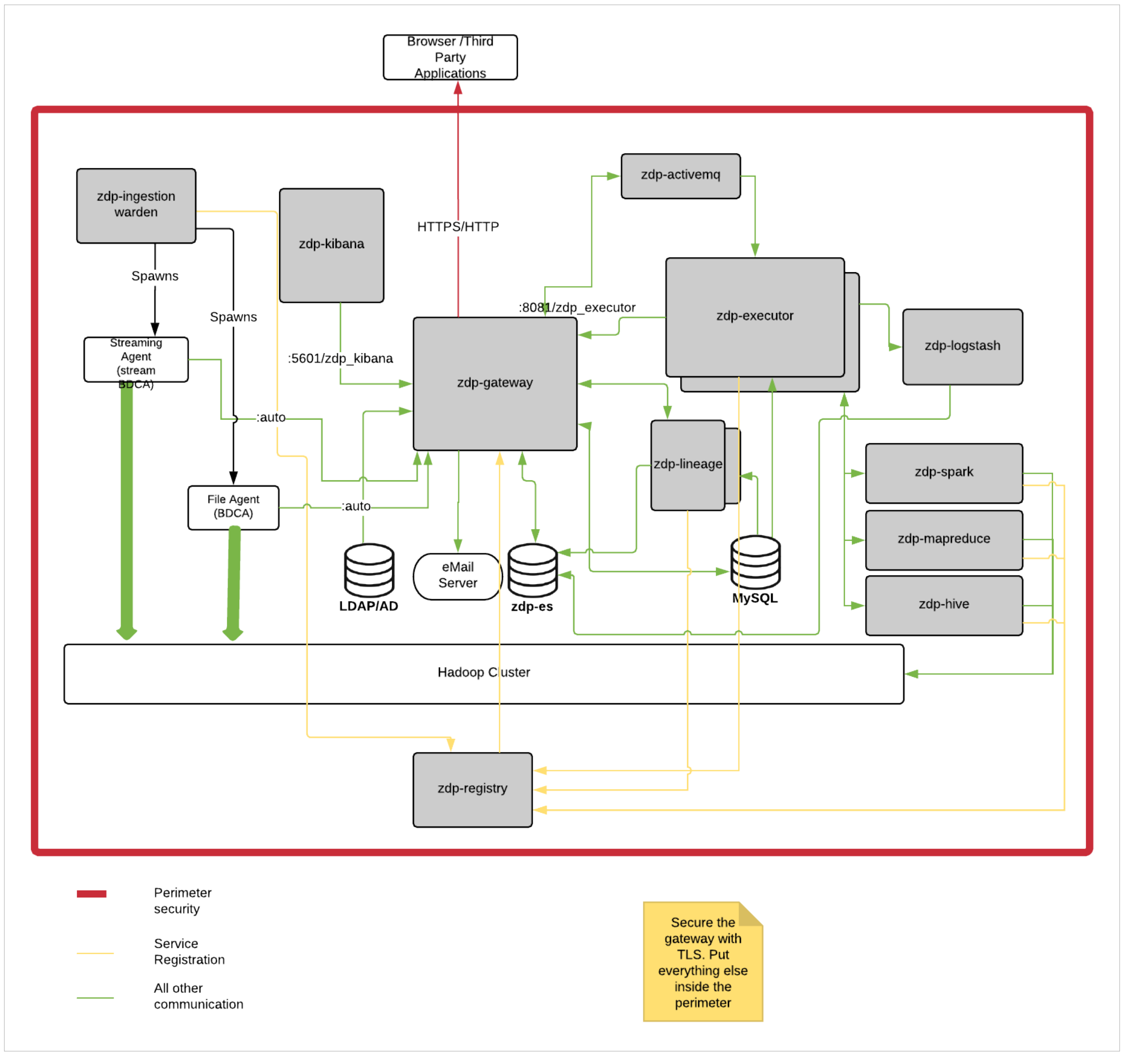
1.2 Installation

1.2.1 Downloading/Extracting ZDP RPMs

To install ZDP, download the following rpm files from the Zaloni Support Portal.

- *zdp-registry_<version>.noarch.rpm*: This rpm is responsible for centralized configuration for the various ZDP processes, such as Elasticsearch, logstash, database, etc.
- *zdp-gateway_<version>.noarch.rpm*: This rpm is responsible for maintaining the ZDP UI and delegating API calls to other services, such as initiating an execution requests, reporting, system configuration, metadata management, etc.
- *zdp-executor_<version>.noarch.rpm*: This rpm is responsible for the execution of the workflows. After a workflow is submitted for execution, the *zdp-executor* executes the workflows based on the priority/capacity.
- *zdp-ingestion-warden_<version>.noarch.rpm*: This rpm is responsible for centralized BDCA configuration, wherein BDCA(s) processes/threads are spawned on the landing zone server where this rpm is installed.
- *zdp-es_<version>.noarch.rpm*: This rpm is responsible for taking data in and storing it in a sophisticated format, optimized for language-based searches. The *zdp-es* service is tightly coupled with the *zdp-logstash* and *zdp-kibana* services. For more information, refer to ELK stack.
- *zdp-logstash_<version>.noarch.rpm*: This rpm is responsible for gathering, enriching, and unifying all of your data regardless of format or schema. The *zdp-logstash* service is tightly coupled with the *zdp-es* and *zdp-kibana* services. For more information, refer to ELK stack.
- *zdp-kibana_<version>.noarch.rpm*: This rpm is responsible for displaying the details stored in *zdp-es* as dashboards. The *zdp-kibana* service is tightly coupled with the *zdp-es* and *zdp-logstash* services. For more information, refer to ELK stack.
- *zdp-activemq_<version>.noarch.rpm*: This rpm acts as a message broker to integrate the *zdp-gateway* and *zdp-executor* services by using queues.
- *zdp-lineage_<version>.noarch.rpm*: This rpm is responsible for tracking workflow level events, such that any change in a file/entity in the ZDP eco-system is captured.

- *zdp-mr_<version>.noarch.rpm*: This rpm runs the MapReduce jobs.
- *zdp-spark_<version>.noarch.rpm*: This rpm helps in the execution of Spark jobs that are executed through Spark-backed workflow actions and transformations.
- *zdp-hive_<version>.noarch.rpm*: This rpm helps in the creation of Hive tables corresponding to the ZDP metadata.
- *zdp-dme_<version>.noarch.rpm*: This rpm is responsible for installing the DME (Data Master Extension) service in the ZDP application. As the DME service is an add-on functionality, you must install this service to use that feature.



1.2.2 Planning the Deployment Topology

Before installing ZDP, please go through the hardware_req, software_req, and env_prereq.

- The rpm files can be installed on a single node or in different nodes. For example, the *zdp-gateway* and *zdp-lineage* services can be installed in two different nodes.
- The *zdp-es*, *zdp-executor*, and *zdp-kibana* services can also be installed in multiple nodes. For example, you may install three *zdp-executor* services on three different nodes.
- You must have the root or sudo access on the servers where you want to install ZDP services.

- The Hadoop client must be installed on all the nodes where the *zdp-hive*, *zdp-spark*, and *zdp-mr* services are installed.
- Ensure that the *zdp-dme* and *zdp-spark* services are in the same cluster. However, these services can be in different nodes.
- The *zaloni* user must have a hadoop home directory. For example, */user/zaloni*.

1.2.3 Installing the ZDP services on a Single Node

To install the ZDP services on a single node, perform the following steps:

1. Access the node where you want to install the rpms.
2. Copy all the rpms to a specific directory within the node.
3. To install all the ZDP services, execute the below command:

   ```
   sudo yum install zdp*.rpm
   ```

   Alternatively, to install individual ZDP services, execute the following command:

   ```
   sudo yum install <rpm name>
   ```

   For example, to install the *zdp-registry* service, run the following command:

   ```
   sudo yum install zdp-registry_*.rpm
   ```

1.2.4 Installing the ZDP services on Multiple Nodes

To install the ZDP services on different nodes, perform the following steps:

1. Access the node where you want to install the particular rpm.
2. Copy the rpm to a specific directory within the node.
3. To install the specific ZDP service, execute the below command:

   ```
   sudo yum install <rpm name>
   ```

   For example, to install the *zdp-registry* service, run the following command:

   ```
   sudo rpm -ivh zdp-registry_<version>.rpm
   ```

1.2.5 RPM Layout and Important Directories

RPMs are installed under the */opt/zdp/* directory (by default). After installation, the following directories are created. You can also inspect these before installation by using the command:

```
rpm -qlp [rpm file name]
```

Directory	Description
/opt/zdp/[service name]	Indicates the location the service binaries are installed.
/opt/zdp/[service name]/bin	Includes the the binary/helper scripts of the service
/etc/[service name]	Includes the post installation configuration scripts of the service.
/var/[service name]/data	Includes any local data created by the service.
/var/[service name]/lib	Includes required libraries for the service.
/var/log/[service name]	Includes the logs for the service.
/var/zdp-common	Includes any common data files shared between services. It is recommended that you mount an NFS shared storage at this point, in a multi-node deployment.
/etc/init.d/[service name]	Indicates the system V script so that you can start, stop, and restart the service.
/etc/hosts	Indicates the IPs and hostnames for all the nodes where the ZDP services are started.

For example, the following components are generated after installation of the *zdp-gateway* and *zdp-dme* services:

Folder structure of zdp-gateway
/opt/zdp/zdp-gateway
/opt/zdp/zdp-gateway/bin
/etc/zdp-gateway/data
/var/log/zdp-gateway
/var/zdp-common
/etc/init.d/zdp-gateway

Folder structure of zdp-dme
/opt/zdp/zdp-dme
/opt/zdp/zdp-dme/bin
/etc/zdp-dme/data
/var/log/zdp-dme
/opt/zdp/zdp-dme/lib

Note: Apart from this, a unix system account called *zaloni* is created when any of the RPMs are installed. The installed services are run by this user by default.

1.3 Post Installation Configuration

1.3.1 Single Node Configuration

Important:

- To encrypt any properties in a yml file, refer to the *Securing Configuration Files* section.

- Once the ZDP services are installed, ensure that you set the Linux *ulimit* as open files: *65536* and max user process: *9000* for the designated ZDP user (*zaloni* by default) in all the nodes where ZDP services are installed. The *zaloni* user is created by the install RPMs if it does not exist.

Once the installation is complete, perform the following post installation configuration steps:

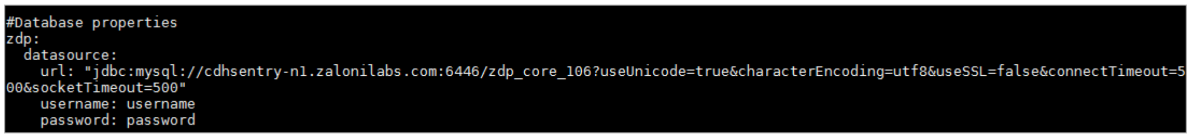
1. To store ZDP's core data (referred to as ZDP core schema), create an empty database in MySQL. For example,

   ```
   mysql -uroot -p -e `CREATE DATABASE zdp_core CHARSET latin1`
   ```

2. To provide the MySQL database URL and credentials, edit the */etc/zdp-registry/application.yml* file and set the following properties:

   ```
   zdp.datasource.url: Indicates the database URL. For example, jdbc:mysql://
   ↪<hostname_of_the_server_where_MySQL_is_installed>:3306/zdp_core?
   ↪autoReconnect=false&useUnicode=true&characterEncoding=utf8&useSSL=false&
   ↪connectTimeout=1000&socketTimeout=1000.
   ```

   For example,

   ```
   #Database properties
   zdp:
     datasource:
       url: "jdbc:mysql://cdhsentry-n1.zalonilabs.com:6446/zdp_core_106?useUnicode=true&characterEncoding=utf8&useSSL=false&connectTimeout=5
   00&socketTimeout=500"
       username: username
       password: password
   ```

   If you want to avail the HA feature for a MySQL cluster where there are single/multiple routers, perform the following tasks:

   (a) Specify this parameter as follows (sample):

   ```
   jdbc:mysql://projb-n4.zalonilabs.com:6446/zdp_core?autoReconnect=false&
   ↪useUnicode=true&characterEncoding=utf8&useSSL=false&connectTimeout=1000&
   ↪socketTimeout=1000

   Here, projb-n4.zalonilabs.com:6446 is the hostname/port for a router in the␣
   ↪MySQL cluster. Note that you can add only one router in this step.
   ```

   (b) Once the *zdp-gateway* service is up, access the *application.yml* file again and update the *zdp.datasource.url* parameter as follows:

   ```
   jdbc:mysql:loadbalance://projb-n4.zalonilabs.com:6446,projb-n3.zalonilabs.
   ↪com:6446,projb-n1.zalonilabs.com:6446,projb-n2.zalonilabs.com:6446/zdp_core?
   ↪autoReconnect=false&useUnicode=true&characterEncoding=utf8&useSSL=false&
   ↪connectTimeout=1000&socketTimeout=1000

    - In this step, you can add the `loadbalance` protocol as well as add the␣
   ↪additional routers in the MySQL cluster.
    - In a multi MySQL router cluster, if you set the parameter: `autoReconnect`␣
   ↪to `true` and one of the MySQL router service is down, the system might␣
   ↪face latency. Hence, it is recommended that you set the parameter:␣
   ↪`autoReconnect` to `false` for `zdp.datasource.url`. Additionally, to avoid␣
   ↪latency due to connection/socket timeout (when one of the MySQL router is␣
   ↪down), set the parameters: `connectTimeout` and `socketTimeout` to specific␣
   ↪values.
    - If one or more of the routers in a multi router MySQL cluster are shut␣
   ↪down, you might observe latency while performing the various operations␣
   ↪involving the cluster. To avoid that issue, you must remove the hostnames/
   ↪ports that correspond to the specific router(s) from the `zdp.datasource.
   ↪url` parameter and restart the `zdp-registry`/`zdp-gateway`/`zdp-executor`␣
   ↪services.
   ```

```
```

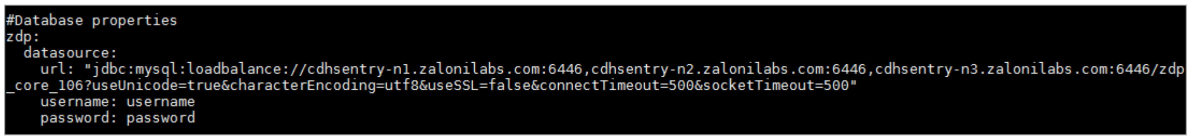
(c) Restart the *zdp-registry*/*zdp-gateway* services and the other applicable services.

Note: For more information on the HA for MySQL, refer to the *High Availability for MySQL* section.

```
zdp.datasource.username: Indicates the database username. For example, root.
```

```
zdp.datasource.password: Indicates the database password. For example, password.
```

For example,



```
#Database properties
zdp:
  datasource:
    url: "jdbc:mysql:loadbalance://cdhsentry-n1.zalonilabs.com:6446,cdhsentry-n2.zalonilabs.com:6446,cdhsentry-n3.zalonilabs.com:6446/zdp
_core_106?useUnicode=true&characterEncoding=utf8&useSSL=false&connectTimeout=500&socketTimeout=500"
    username: username
    password: password
```

3. Configure the *zdp-hive* service:

   - Update the following properties in the */etc/zdp-hive/zdp-hive.yml* file:

```
jdbc.hive.url: Indicates the JDBC connection string for the data store which␣
↪contains metadata. For example, jdbc:hive2://strata-node1.zalonilabs.
↪com:10000/default;principal=hive/strata-node1.zalonilabs.com@ZALONILABS.COM.
```

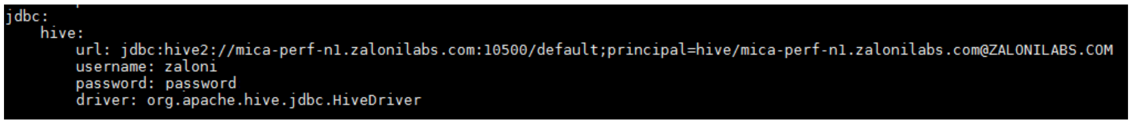
```
jdbc.hive.username: Indicates the JDBC username. For example, username.
```

```
jdbc.hive.password: Indicates the JDBC password. For example, password.
```

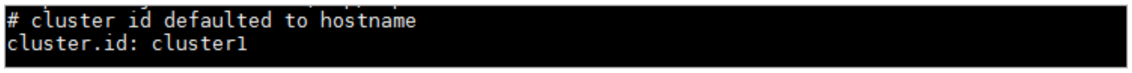
```
jdbc.hive.driver: Indicates the JDBC Hive driver. By default, it is org.
↪apache.hive.jdbc.HiveDriver.
```

```
cluster.id: Indicates the name for the cluster. Default value is the hostname␣
↪of the node. This parameter is mandatory for multi-node setup where this␣
↪string must be updated with a custom value. To use the high availability␣
↪feature where the zdp-hive service is installed in multiple nodes within a␣
↪cluster, you must set the value for this parameter to the same value in all␣
↪the zdp-hive.yml files in all the nodes of the cluster. Note that you must␣
↪not specify the hash (#) special character in the value for the cluster.id␣
↪parameter. For more information refer to the :ref:`diff_nodes_config`␣
↪section.
```

   For example,



```
jdbc:
    hive:
        url: jdbc:hive2://mica-perf-n1.zalonilabs.com:10500/default;principal=hive/mica-perf-n1.zalonilabs.com@ZALONILABS.COM
        username: zaloni
        password: password
        driver: org.apache.hive.jdbc.HiveDriver
```



```
# cluster id defaulted to hostname
cluster.id: cluster1
```

   - Update the following properties in the */etc/sysconfig/zdp-hive* file:

```
hive_lib: Indicates Hive libraries' classpath location. For example, /usr/hdp/
↪2.6.2.14-5/hive/lib/*.jar.
```

```
hcatalog_lib: Indicates the HCatalog libraries' classpath location. For␣
↪example, /usr/hdp/2.6.2.14-5/hive-hcatalog/share/hcatalog/*.jar.
```

```
web_hcatalog_lib: Indicates the Web HCatalog libraries classpath location.␣
↪For example, /usr/hdp/2.6.2.14-5/hive-hcatalog/share/webhcat/java-client/*.
↪jar.
```

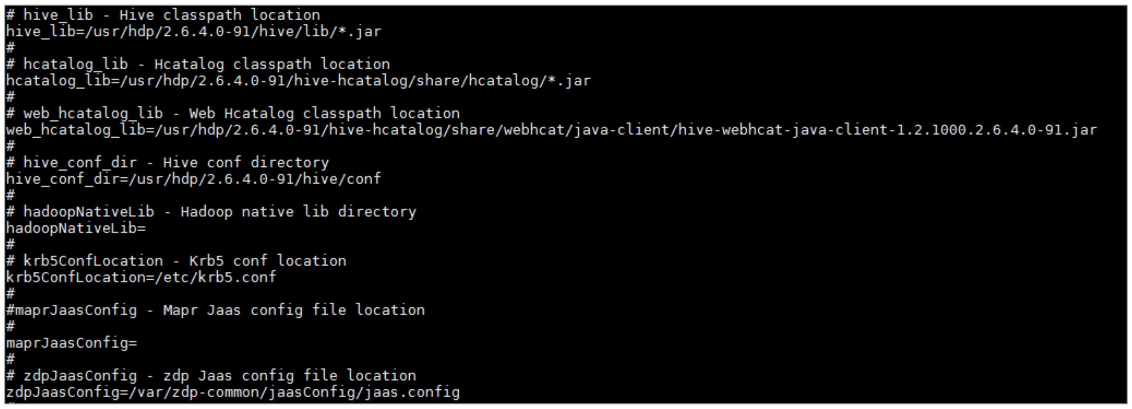
```
hive_conf_dir: Indicates the Hive configuration directory. For example, /etc/
↪hive/2.6.2.14-5/0.
```

```
hadoopNativeLib: Optional. Indicates the location for the Hadoop native␣
↪library. For example, /usr/lib/hadoop/lib/native.
```

```
krb5ConfLocation: Optional. Indicates Krb5 configuration location. For␣
↪example, /etc/krb5.conf.
```

```
zdpJaasConfig: Optional. Indicates the location for the ZDP Jaas␣
↪configuration file. For example, /var/zdp-common/jaasConfig/jaas.config.
```

For example,



```
# hive_lib - Hive classpath location
hive_lib=/usr/hdp/2.6.4.0-91/hive/lib/*.jar
#
# hcatalog_lib - Hcatalog classpath location
hcatalog_lib=/usr/hdp/2.6.4.0-91/hive-hcatalog/share/hcatalog/*.jar
#
# web_hcatalog_lib - Web Hcatalog classpath location
web_hcatalog_lib=/usr/hdp/2.6.4.0-91/hive-hcatalog/share/webhcat/java-client/hive-webhcat-java-client-1.2.1000.2.6.4.0-91.jar
#
# hive_conf_dir - Hive conf directory
hive_conf_dir=/usr/hdp/2.6.4.0-91/hive/conf
#
# hadoopNativeLib - Hadoop native lib directory
hadoopNativeLib=
#
# krb5ConfLocation - Krb5 conf location
krb5ConfLocation=/etc/krb5.conf
#
#maprJaasConfig - Mapr Jaas config file location
#
maprJaasConfig=
#
# zdpJaasConfig - zdp Jaas config file location
zdpJaasConfig=/var/zdp-common/jaasConfig/jaas.config
```

4. Configure the *zdp-mr* service:

   - Update the following properties in the */etc/zdp-mr/zdp-mr.yml* file:

```
jdbc.hive.url: Indicates the JDBC Hive URL. For example, jdbc:hive2://host.
↪zalonilabs.com:10000/default;principal=hive/strata-node1.zalonilabs.
↪com@ZALONILABS.COM.
```

```
jdbc.hive.username: Indicates the JDBC username. For example, username.
```

```
jdbc.hive.password: Indicates the JDBC password. For example, password.
```

```
jdbc.hive.driver: Indicates the JDBC Hive driver. By default, it is org.
↪apache.hive.jdbc.HiveDriver.
```

```
jdbc.spark.url: Indicates the JDBC Spark URL. For example, jdbc:hive2://
↪strata-node1.zalonilabs.com:10016/default;principal=hive/strata-node1.
↪zalonilabs.com@ZALONILABS.COM.
```

```
jdbc.spark.username: Indicates the JDBC Hive username. For example, username.
```

```
jdbc.spark.password: Indicates the JDBC password. For example, password.
```

```
jdbc.spark.driver: Indicates the JDBC Hive driver. By default, it is org.
↪apache.hive.jdbc.HiveDriver.
```

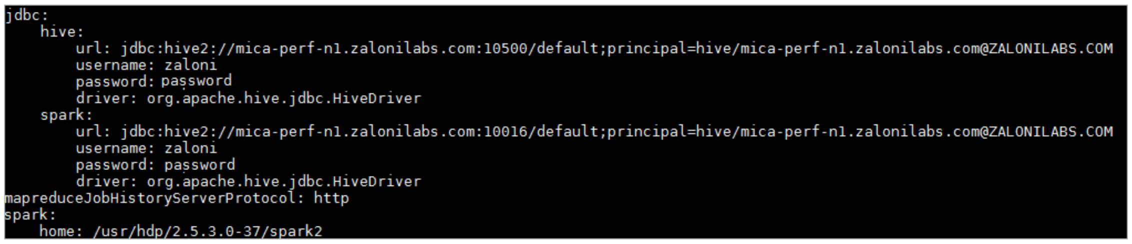
```
mapreduceJobHistoryServerProtocol: Optional. Indicates the job history server␣
↪protocol. Possible values are http or https based on the server␣
↪configuration. Default value is http.
```

```
spark.home: Indicates the Spark home directory classpath. For example, /usr/
↪hdp/current/spark2-client.
```

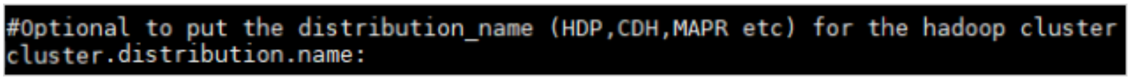
```
cluster.distribution.name: Optional. Indicates the Hadoop distribution name.␣
↪For example, HDP, CDH, MAPR.
```

```
cluster.id: Indicates the name for the cluster. Default value is the hostname␣
↪of the node. This parameter is mandatory for multi-node setup where this␣
↪string must be updated with a custom value. To use the high availability␣
↪feature where the zdp-mr service is installed in multiple nodes within a␣
↪cluster, you must set the value for this parameter to the same value in all␣
↪the zdp-mr.yml files in all the nodes of the cluster. Note that you must␣
↪not specify the hash (#) special character in the value for the cluster.id␣
↪parameter. For more information refer to the :ref:`diff_nodes_config`␣
↪section.
```

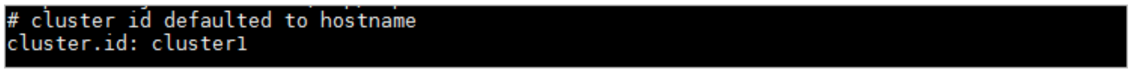
For example,



```
jdbc:
    hive:
        url: jdbc:hive2://mica-perf-n1.zalonilabs.com:10500/default;principal=hive/mica-perf-n1.zalonilabs.com@ZALONILABS.COM
        username: zaloni
        password: password
        driver: org.apache.hive.jdbc.HiveDriver
    spark:
        url: jdbc:hive2://mica-perf-n1.zalonilabs.com:10016/default;principal=hive/mica-perf-n1.zalonilabs.com@ZALONILABS.COM
        username: zaloni
        password: password
        driver: org.apache.hive.jdbc.HiveDriver
mapreduceJobHistoryServerProtocol: http
spark:
    home: /usr/hdp/2.5.3.0-37/spark2
```



```
#Optional to put the distribution_name (HDP,CDH,MAPR etc) for the hadoop cluster
cluster.distribution.name:
```



```
# cluster id defaulted to hostname
cluster.id: cluster1
```

- Update the following properties in the */etc/sysconfig/zdp-mr* file:

```
hadoop_conf_dir: Indicates the Hadoop configuration directory classpath. For␣
↪example, /usr/hdp/2.6.3.0-235/hadoop/conf.
```

```
spark_home: Indicates the Spark home directory classpath. For example, /usr/
↪hdp/current/spark2-client.
```

```
hive_lib: Indicates Hive library classpath. For example, /usr/hdp/current/
↪hive-client/lib/*.jar.
```

```
hive_conf: Indicates Hive configuration directory. For example, /etc/hive/2.6.
↪3.0-235/0.
```

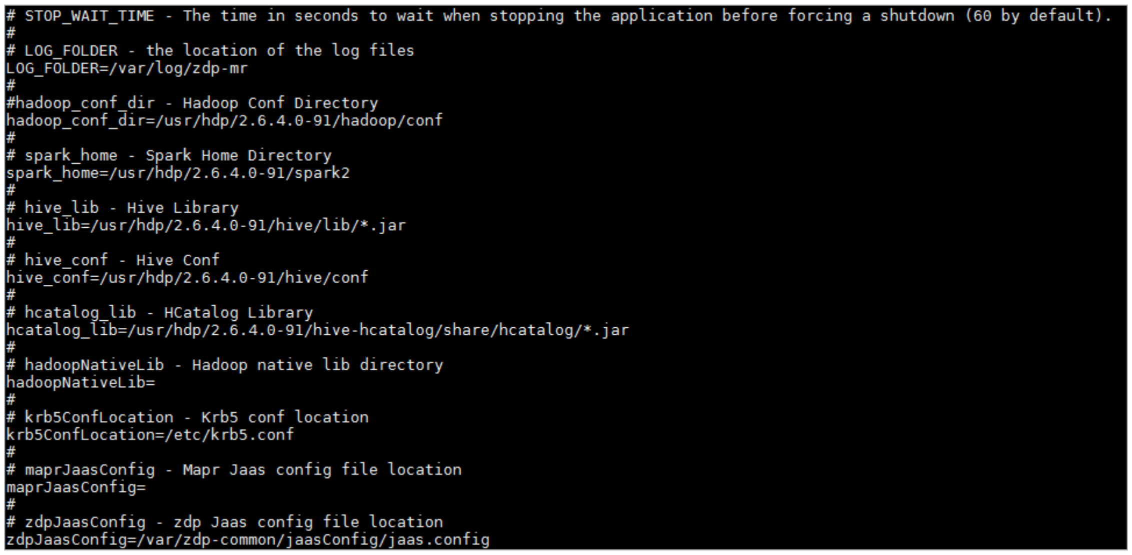
```
hcatalog_lib: Indicates the HCatalog library classpath. For example, /usr/hdp/
↪2.6.3.0-235/hive-hcatalog/share/hcatalog/*.jar.
```

```
hadoopNativeLib: Optional. Indicates the location for the Hadoop native␣
↪library. For example, /usr/lib/hadoop/lib/native.
```

```
krb5ConfLocation: Optional. Indicates the location for the Krb5 configuration␣
↪file. For example, /etc/krb5.conf.
```

```
zdpJaasConfig: Optional. Indicates the location for the ZDP Jaas␣
↪configuration file. For example, /var/zdp-common/jaasConfig/jaas.config.
```

For example,



```
# STOP_WAIT_TIME - The time in seconds to wait when stopping the application before forcing a shutdown (60 by default).
#
# LOG_FOLDER - the location of the log files
LOG_FOLDER=/var/log/zdp-mr
#
#hadoop_conf_dir - Hadoop Conf Directory
hadoop_conf_dir=/usr/hdp/2.6.4.0-91/hadoop/conf
#
# spark_home - Spark Home Directory
spark_home=/usr/hdp/2.6.4.0-91/spark2
#
# hive_lib - Hive Library
hive_lib=/usr/hdp/2.6.4.0-91/hive/lib/*.jar
#
# hive_conf - Hive Conf
hive_conf=/usr/hdp/2.6.4.0-91/hive/conf
#
# hcatalog_lib - HCatalog Library
hcatalog_lib=/usr/hdp/2.6.4.0-91/hive-hcatalog/share/hcatalog/*.jar
#
# hadoopNativeLib - Hadoop native lib directory
hadoopNativeLib=
#
# krb5ConfLocation - Krb5 conf location
krb5ConfLocation=/etc/krb5.conf
#
# maprJaasConfig - Mapr Jaas config file location
maprJaasConfig=
#
# zdpJaasConfig - zdp Jaas config file location
zdpJaasConfig=/var/zdp-common/jaasConfig/jaas.config
```

5. Configure the *zdp-spark* service:

   - Update the following properties in the */etc/zdp-spark/zdp-spark.yml* file:

```
jdbc.spark.url: Indicates the JDBC Spark URL. For example, jdbc:hive2://
↪strata-node1.zalonilabs.com:10016/default;principal=hive/strata-node1.
↪zalonilabs.com@ZALONILABS.COM.
```

```
jdbc.spark.username: Indicates the JDBC Hive username. For example, username.
```

```
jdbc.spark.password: Indicates the JDBC Hive password. For example, password.
```

```
jdbc.spark.driver: Indicates the JDBC Hive driver. For example, org.apache.
↪hive.jdbc.HiveDriver.
```

```
transformation.executed-via: Optional. Indicates the Spark commands that must␣
↪be used to execute the Spark jobs. This can be either spark-submit or spark-
↪shell. The default value is spark-shell.
```

```
spark.default: Optional. Indicates the Spark version that must be selected␣
↪(by default). This can be either spark1 or spark2.
```

```
spark.spark1.home: Optional. Indicates the home directory of Spark 1. For␣
↪example, /usr/hdp/2.5.3.0-37/spark.
```

```
spark.spark1.version: Optional. Indicates the Spark version for Spark 1.For␣
↪example, 1.6.
```

```
spark.spark1.master.url: Optional. Indicates the master URL for Spark 1. For␣
↪example, yarn.
```

```
spark.spark1.deploy.mode: Optional. Indicates the deploy mode for Spark 1.␣
↪This can be either client or cluster. For spark-shell, this value can be␣
↪only client while spark-submit accepts both client and cluster as the␣
↪deploy mode."
```

```
spark.spark1.extra.env: Optional. Indicates the environment arguments for␣
↪Spark 1. For example, KAFKA_KERBEROS_PARAMS=-Djava.security.auth.login.
↪config=/usr/hdp/current/kafka-broker/config/kafka_jaas.conf && LOG_DIR=/var/
↪log/kafka.
```

```
spark.spark1.extra.opts: Optional. Indicates the extra arguments for Spark 1.␣
↪For example, --conf spark.driver.extraLibraryPath=/usr/hdp/current/hadoop-
↪client/lib/native:/usr/hdp/current/hadoop-client/lib/native/Linux-amd64-63.
```

```
spark.spark1.shell.directive: Optional. Indicates the directive for spark-
↪shell for Spark 1. For example, SPARK_PRINT_LAUNCH_COMMAND=true.
```

```
spark.spark1.submit.directive: Optional. Indicates the directive for spark-
↪submit for Spark 1. For example, SPARK_PRINT_LAUNCH_COMMAND=true.
```

```
spark.spark1.kafka.version: Optional. Indicates the Kafka version for Spark 1.
↪ For example, 0.9.
```

```
spark.spark2.home: Optional. Indicates the home directory of Spark 2. For␣
↪example, /usr/hdp/2.5.3.0-37/spark2.
```

```
spark.spark2.version: Optional. Indicates the Spark version for Spark 2.For␣
↪example, 2.3.
```

```
spark.spark2.master.url: Optional. Indicates the master URL for Spark 2. For␣
↪example, yarn.
```

```
spark.spark2.deploy.mode: Optional. Indicates the deploy mode for Spark 2.␣
↪This can be either client or cluster. For spark-shell, this value can be␣
↪only client while spark-submit accepts both client and cluster as the␣
↪deploy mode.
```

```
spark.spark2.extra.env: Optional. Indicates the environment arguments for␣
↪Spark 2. For example, KAFKA_KERBEROS_PARAMS=-Djava.security.auth.login.
↪config=/usr/hdp/current/kafka-broker/config/kafka_jaas.conf && LOG_DIR=/var/
↪log/kafka.
```

```
spark.spark2.extra.opts: Optional. Indicates the extra arguments for Spark 2.␣
↪For example, --conf spark.driver.extraLibraryPath=/usr/hdp/current/hadoop-
↪client/lib/native:/usr/hdp/current/hadoop-client/lib/native/Linux-amd64-64.
```

```
spark.spark2.shell.directive: Optional. Indicates the directive for spark-
↪shell for Spark 2. For example, SPARK_PRINT_LAUNCH_COMMAND=true.
```

```
spark.spark2.submit.directive: Optional. Indicates the directive for spark-
↪submit for Spark 2. For example, SPARK_PRINT_LAUNCH_COMMAND=true.
```

```
spark.spark2.kafka.version: Optional. Indicates the Kafka version for Spark 2.
↪ For example, 0.10.
```

```
sparkify.spark.home: Optional. Indicates the home directory of Spark 2.␣
↪Sparkify actions support the `spark-submit` mode only. For example, `/usr/
↪hdp/2.5.3.0-37/spark2`.
```

```
sparkify.spark.master.url: Optional. Indicates the master URL for Spark 2.␣
↪For example, yarn.
```

```
sparkify.spark.deploy.mode: Optional. Indicates the deploy mode for Spark 2.␣
↪This can be either client or cluster.
```

```
sparkify.spark.version: Optional. Indicates the Spark version to be used␣
↪while executing the Sparkified actions. For example, 2.3.
```

```
sparkify.spark.jars: Indicates the three jars (hive-metastore, commons-lang,␣
↪libthrift) that are used in the Sparkified actions. You can find the path␣
↪for those jars with appropriate version under Spark installation. You must␣
↪list the jars in the format: /usr/hdp/2.5.3.0-37/spark2/jars/hive-metastore-
↪1.2.1.spark2.jar,/usr/hdp/2.5.3.0-37/spark2/jars/commons-lang-2.6.jar,/usr/
↪hdp/2.5.3.0-37/spark2/jars/libthrift-0.9.2.jar. This is a mandatory␣
↪parameter for executing Sparkified actions.
```

```
kafka.version: Optional. Indicates the version for the Kafka component. Kafka␣
↪is used during streaming transformation by using load/output controls. For␣
↪example, 0.10.0.
```

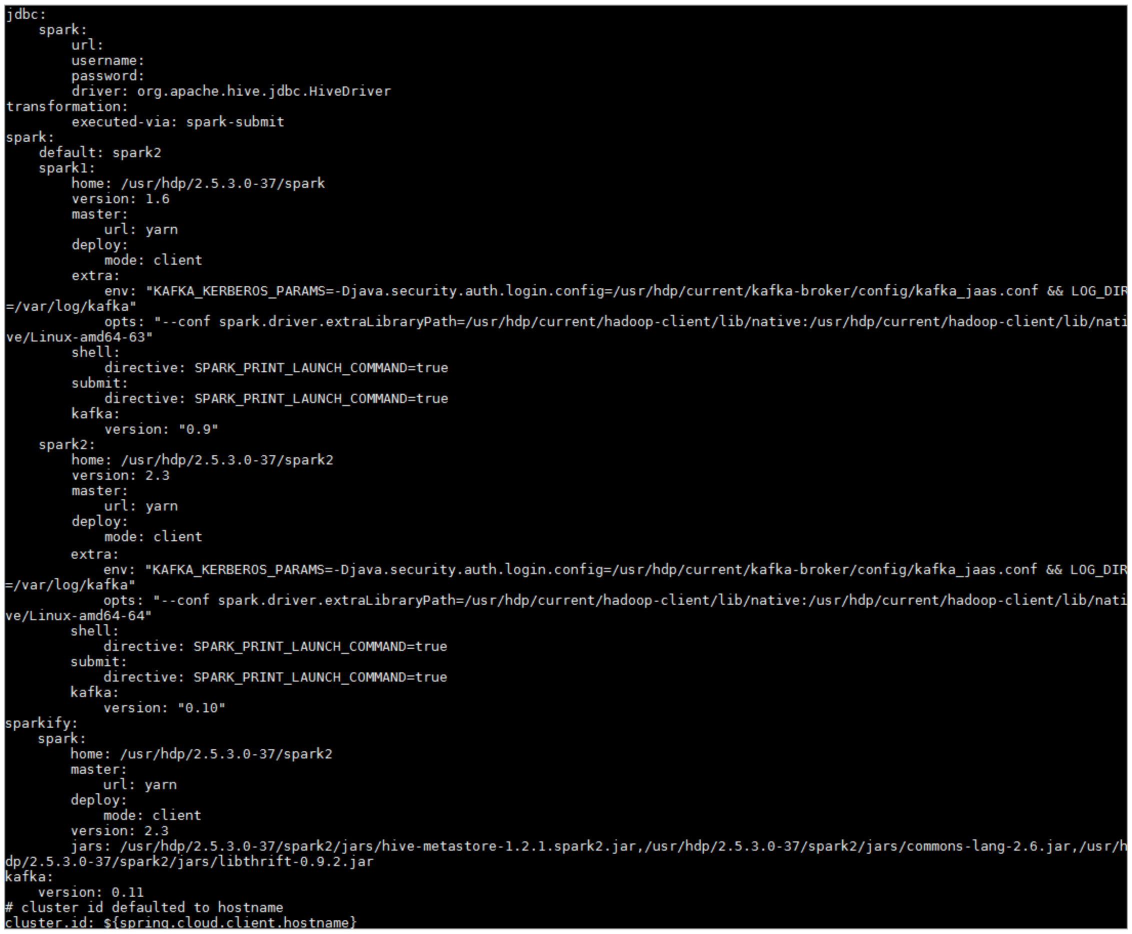
```
blocking.overhead.check.ratio: Optional. Indicates the system should consider␣
↪the already used set of records/blocks for the next set of pairs while␣
↪grouping the very similar records in DME. if the value of the parameter are␣
↪set as 0, records will be considered for next set of pairing. Else the␣
↪records will be filtered out.
```

```
survivorship-rule-pick-non-null-values: Optional. Indicates the system should␣
↪avoid null value in the golden record (after applying any of the␣
↪survivorship rule). Default value is: true.
```

```
survivorship-rules-trusted-values-have-high-priority: Optional. Indicates the␣
↪system should generates golden record from the input record, even if the␣
↪exact trusted value is not matched/available. The system validate the␣
↪trusted value in the order it is configured and pick the record which␣
↪matches trusted value. Default value is: false.
```

```
cluster.id: Indicates the name for the cluster. Default value is the hostname␣
↪of the node. This parameter is mandatory for multi-node setup where this␣
↪string must be updated with custom values. To use the high availability␣
↪feature where the zdp-spark service is installed in multiple nodes within a␣
↪cluster, you must set the value for this parameter to the same value in all␣
↪the zdp-spark.yml files in all the nodes of the cluster. Note that you must␣
↪not specify the hash (#) special character in the value for the cluster.id␣
↪parameter. For more information, refer to the :ref:`diff_nodes_config`␣
↪section.
```

For example,



```
jdbc:
    spark:
        url:
        username:
        password:
        driver: org.apache.hive.jdbc.HiveDriver
transformation:
        executed-via: spark-submit
spark:
    default: spark2
    spark1:
        home: /usr/hdp/2.5.3.0-37/spark
        version: 1.6
        master:
            url: yarn
        deploy:
            mode: client
        extra:
            env: "KAFKA_KERBEROS_PARAMS=-Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/config/kafka_jaas.conf && LOG_DIR
=/var/log/kafka"
            opts: "--conf spark.driver.extraLibraryPath=/usr/hdp/current/hadoop-client/lib/native:/usr/hdp/current/hadoop-client/lib/nati
ve/Linux-amd64-63"
        shell:
            directive: SPARK_PRINT_LAUNCH_COMMAND=true
        submit:
            directive: SPARK_PRINT_LAUNCH_COMMAND=true
        kafka:
            version: "0.9"
    spark2:
        home: /usr/hdp/2.5.3.0-37/spark2
        version: 2.3
        master:
            url: yarn
        deploy:
            mode: client
        extra:
            env: "KAFKA_KERBEROS_PARAMS=-Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/config/kafka_jaas.conf && LOG_DIR
=/var/log/kafka"
            opts: "--conf spark.driver.extraLibraryPath=/usr/hdp/current/hadoop-client/lib/native:/usr/hdp/current/hadoop-client/lib/nati
ve/Linux-amd64-64"
        shell:
            directive: SPARK_PRINT_LAUNCH_COMMAND=true
        submit:
            directive: SPARK_PRINT_LAUNCH_COMMAND=true
        kafka:
            version: "0.10"
sparkify:
    spark:
        home: /usr/hdp/2.5.3.0-37/spark2
        master:
            url: yarn
        deploy:
            mode: client
        version: 2.3
        jars: /usr/hdp/2.5.3.0-37/spark2/jars/hive-metastore-1.2.1.spark2.jar,/usr/hdp/2.5.3.0-37/spark2/jars/commons-lang-2.6.jar,/usr/h
dp/2.5.3.0-37/spark2/jars/libthrift-0.9.2.jar
kafka:
    version: 0.11
# cluster id defaulted to hostname
cluster.id: ${spring.cloud.client.hostname}
```

- Update the following properties in the */etc/sysconfig/zdp-spark* file:

```
hadoop_conf: Indicates the Hadoop configuration directory. For example, /usr/
↪hdp/2.6.3.0-235/hadoop/conf.
```

```
hcatalog_lib: Indicates the HCatalog libraries' classpath location. For␣
↪example, /usr/hdp/2.6.0.3-8/hive-hcatalog/share/hcatalog/*.jar.
```

```
spark_home: Indicates the Spark home directory. For example, /usr/hdp/current/
↪spark2-client.
```

```
binBashPath: Optional. Indicates the path for bin bash. For example, /bin/
↪bash.
```

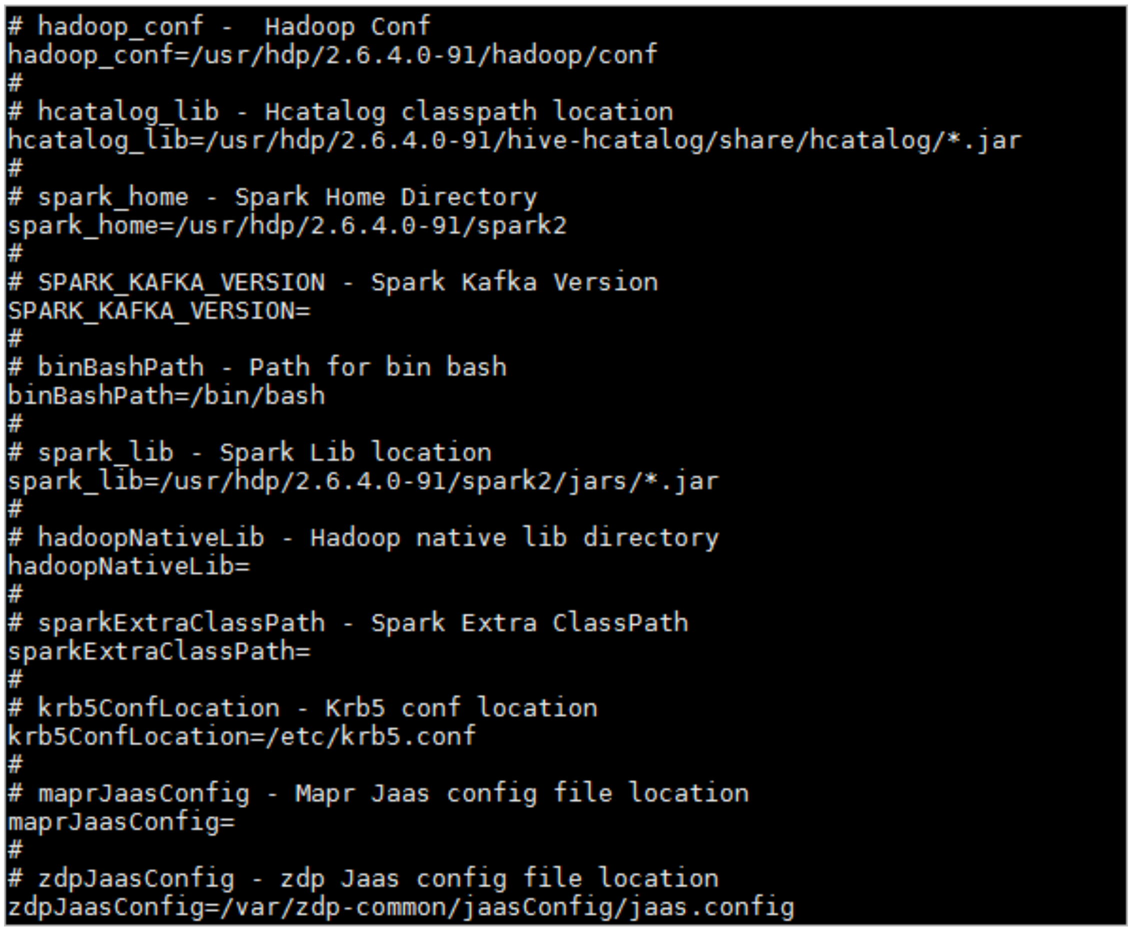
```
```

```
spark_lib: Indicates the location where the Spark libraries are located. For␣
↪example, /usr/hdp/current/spark2-client/jars/*.jar.
```

```
sparkExtraClassPath: Optional. Indicates the Spark extra classpath that can␣
↪contain additional jars, required for running Hadoop jobs. This classpath␣
↪may be required when spark-env.sh or spark-defaults.conf have the classpath␣
↪already defined. This classpath is used by the zdp-spark micro-service.␣
↪This option is mandatory for EMR distribution. For example, /usr/lib/hadoop-
↪lzo/lib/*:/usr/lib/hadoop/hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/
↪share/aws/emr/emrfs/conf:/usr/share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/
↪emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/usr/share/aws/emr/security/
↪lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-spark-client.jar:/
↪usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
↪sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/usr/share/aws/emr/s3select/
↪lib/emr-s3-select-spark-connector.jar.
```

```
zdpJaasConfig: Optional. Indicates the location for the ZDP Jaas␣
↪configuration file. For example, /var/zdp-common/jaasConfig/jaas.config.
```

For example,



```
# hadoop_conf -  Hadoop Conf
hadoop_conf=/usr/hdp/2.6.4.0-91/hadoop/conf
#
# hcatalog_lib - Hcatalog classpath location
hcatalog_lib=/usr/hdp/2.6.4.0-91/hive-hcatalog/share/hcatalog/*.jar
#
# spark_home - Spark Home Directory
spark_home=/usr/hdp/2.6.4.0-91/spark2
#
# SPARK_KAFKA_VERSION - Spark Kafka Version
SPARK_KAFKA_VERSION=
#
# binBashPath - Path for bin bash
binBashPath=/bin/bash
#
# spark_lib - Spark Lib location
spark_lib=/usr/hdp/2.6.4.0-91/spark2/jars/*.jar
#
# hadoopNativeLib - Hadoop native lib directory
hadoopNativeLib=
#
# sparkExtraClassPath - Spark Extra ClassPath
sparkExtraClassPath=
#
# krb5ConfLocation - Krb5 conf location
krb5ConfLocation=/etc/krb5.conf
#
# maprJaasConfig - Mapr Jaas config file location
maprJaasConfig=
#
# zdpJaasConfig - zdp Jaas config file location
zdpJaasConfig=/var/zdp-common/jaasConfig/jaas.config
```

6. Start all the services. For more information, refer to the *Application Start-up and Shut-down* section.
7. Update the following properties in the */etc/zdp-ingestion-warden/zdp-ingestion-warden.yml* file:

```
eureka.instance.metadata-map.service.name: Optional. Indicates the name for the␣
↪landing zone server.
```

```
eureka.instance.metadata-map.flume-ingestion-failed-directory: Indicates the␣
↪failed directory for streaming ingestion. The default value is /var/zdp-
↪ingestion-warden/agents/flume/failed_dir.
```

```
eureka.instance.metadata-map.flume-installation-dir: Optional. Indicates the␣
↪installation directory for streaming ingestion where the BDCA agents will be␣
↪installed. The default value is /var/zdp-ingestion-warden/agents/flume/
↪installation.
```

```
eureka.instance.metadata-map.bdca.enabled: Optional. Indicates whether you want␣
↪to use the same node as the landing zone server for file-based ingestion.
```

```
eureka.instance.metadata-map.flume.enabled: Optional. Indicates whether you want␣
↪to use the same node as the landing zone server for streaming ingestion.
```

```
eureka.instance.metadata-map.warden-group: Optional. Indicates the cluster name␣
↪to identify all the zdp-ingestion-warden services that belong to the same␣
↪cluster.
```

```
eureka.instance.metadata-map.warden-group: If a specific BDCA within a zdp-
↪ingestion-warden service shuts down, you can failover to a different zdp-
↪ingestion-warden service that belongs to the same cluster. The value for the␣
↪parameter is populated from the parameter: warden.group (defined in the zdp-
↪ingestion-warden.yml file).
```

```
warden.group: Optional. Indicates the cluster name to identify all the zdp-
↪ingestion-warden services that belong to the same cluster. If a specific BDCA␣
↪within a zdp-ingestion-warden service shuts down, you can failover to a␣
↪different zdp-ingestion-warden service that belongs to the same cluster.
```

```
flume.enabled: Indicates if the specific zdp-ingestion-warden service can launch␣
↪Flume agents. The default value is true.
```

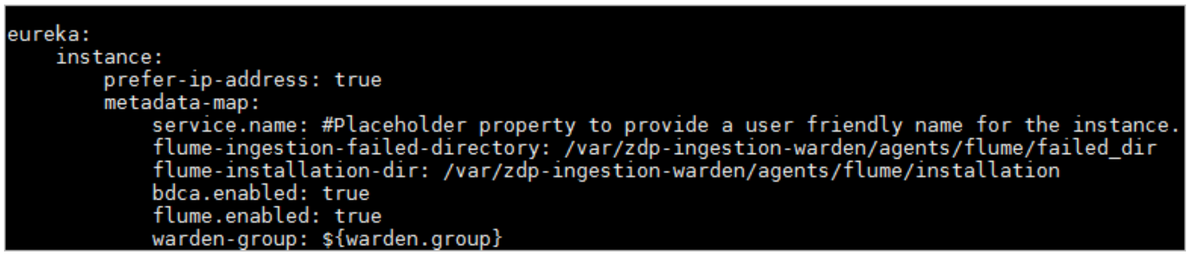
```
flume.limits.hard: Indicates the number of Flume agents that the zdp-ingestion-
↪warden service can launch. It is not recommended to set this value to zero␣
↪(default value) as the service can then launch infinite number of Flume agents␣
↪and that might impact the resource utilization.
```

```
dca.enabled: Indicates if the specific zdp-ingestion-warden service can launch␣
↪BDCA agents. The default value is true.
```

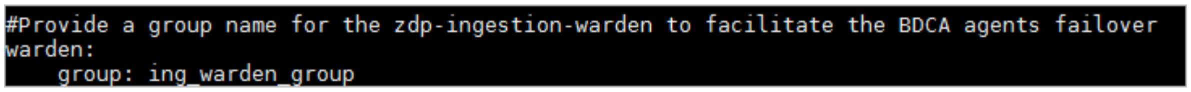
```
dca.limits.hard: Indicates the number of BDCA agents that the zdp-ingestion-
↪warden service can launch. It is not recommended to set this value to zero␣
↪(default value) as the service can then launch infinite number of BDCA agents␣
↪and that might impact the resource utilization.
```

```
application.agent.bdca.execution.plugin: Indicates the path for the plugin that␣
↪defines the logic for switching user for starting/stopping BDCAs. For example, /
↪opt/zdp/zdp-ingestion-warden/bin/plugins/bdca-execution-plugin-sudo.
```

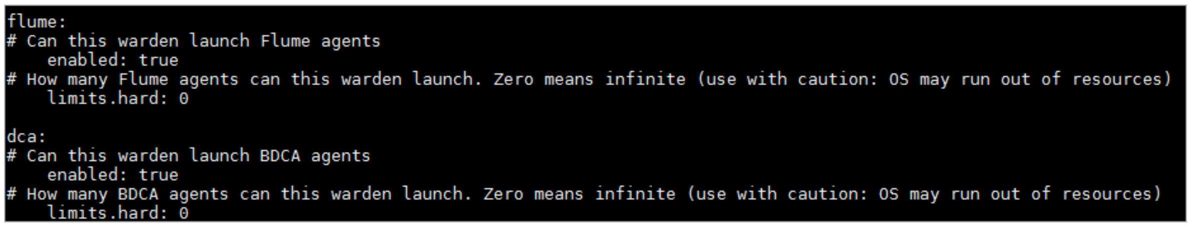
For example,



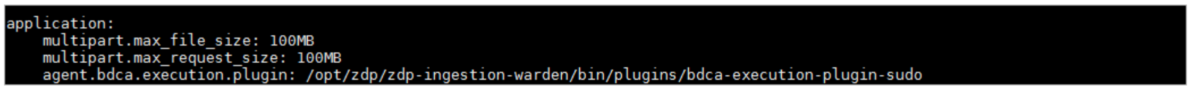
```
eureka:
    instance:
        prefer-ip-address: true
        metadata-map:
            service.name: #Placeholder property to provide a user friendly name for the instance.
            flume-ingestion-failed-directory: /var/zdp-ingestion-warden/agents/flume/failed_dir
            flume-installation-dir: /var/zdp-ingestion-warden/agents/flume/installation
            bdca.enabled: true
            flume.enabled: true
            warden-group: ${warden.group}
```



```
#Provide a group name for the zdp-ingestion-warden to facilitate the BDCA agents failover
warden:
    group: ing_warden_group
```



```
flume:
# Can this warden launch Flume agents
    enabled: true
# How many Flume agents can this warden launch. Zero means infinite (use with caution: OS may run out of resources)
    limits.hard: 0

dca:
# Can this warden launch BDCA agents
    enabled: true
# How many BDCA agents can this warden launch. Zero means infinite (use with caution: OS may run out of resources)
    limits.hard: 0
```



```
application:
    multipart.max_file_size: 100MB
    multipart.max_request_size: 100MB
    agent.bdca.execution.plugin: /opt/zdp/zdp-ingestion-warden/bin/plugins/bdca-execution-plugin-sudo
```

8. (Optional) To use the DME service, configure the following properties and start the *zdp-dme* service:

   • Update the following properties in the */etc/sysconfig/zdp-dme* file.

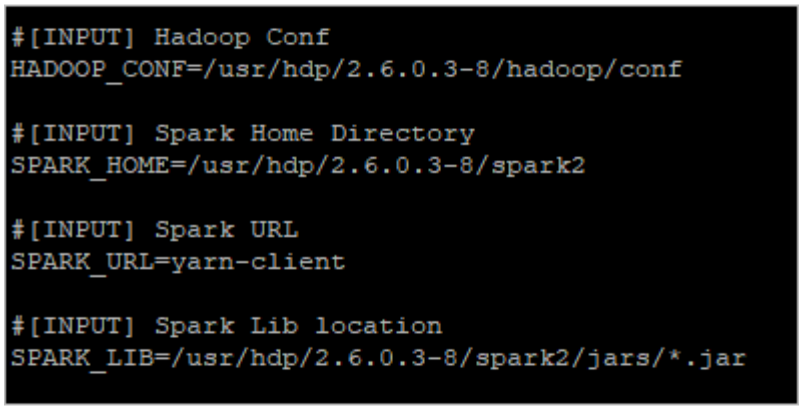
   ```
   HADOOP_CONF: Indicates the Hadoop configuration directory. For example, /usr/
   ↪hdp/2.6.0.3-8/hadoop/conf.
   ```

   ```
   SPARK_HOME: Indicates the Spark home directory. For example, /usr/hdp/2.6.0.3-
   ↪8/spark2.
   ```

   ```
   SPARK_URL: Indicates the deploy mode for Spark. For example, yarn-client.
   ```

   ```
   SPARK_LIB: Optional. Indicates the location of the Spark libraries. For␣
   ↪example, /usr/hdp/current/spark2-client/jars/*.jar.
   ```

   For example,

   ```
   #[INPUT] Hadoop Conf
   HADOOP_CONF=/usr/hdp/2.6.0.3-8/hadoop/conf

   #[INPUT] Spark Home Directory
   SPARK_HOME=/usr/hdp/2.6.0.3-8/spark2

   #[INPUT] Spark URL
   SPARK_URL=yarn-client

   #[INPUT] Spark Lib location
   SPARK_LIB=/usr/hdp/2.6.0.3-8/spark2/jars/*.jar
   ```

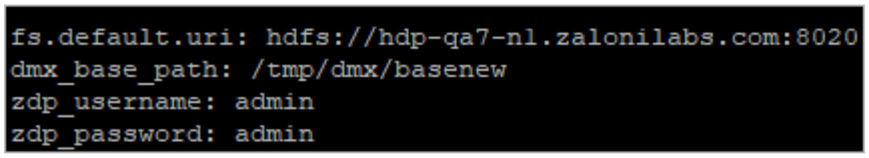
   • Update the following properties in the */etc/zdp-dme/zdp-dme.yml* file.

```
dme_base_path: Indicates the DME base path. For example, /user/zaloni/dmx/
↪basenew.
```

```
zdp_username: Indicates DME username. For example, admin.
```

```
zdp_password: Indicates DME password. For example, admin.
```

For example,



```
fs.default.uri: hdfs://hdp-qa7-n1.zalonilabs.com:8020
dmx_base_path: /tmp/dmx/basenew
zdp_username: admin
zdp_password: admin
```

- Set the *DME_FEATURE_ENABLED* parameter to *true* in system variables. For more information, refer to the configure_system_variable section.

9. Stop and start the *zdp-services*. For more information on starting and stopping the ZDP services, refer to the *Application Start-up and Shut-down* section.

1.3.2 Multi-Node Configuration

Important: To encrypt any properties in a yml file, refer to the *Securing Configuration Files* section.

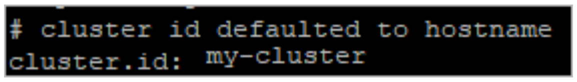
To install the rpms in different nodes, perform the following steps in addition to the steps mentioned in the *Single Node Configuration* section.

Important:

- The ports where the *zdp-gateway* and *zdp-registry* services are hosted must be accessible from all the ports where the other ZDP services (*zdp-logstash*, *zdp-kibana*, etc) are hosted.
- If the *zdp-gateway*, *zdp-executor*, and *zdp-logstash* services are not all co-located, ensure that the */var/zdp-common* location on each node is shared over a network drive.

1. Update the string: *spring.cloud.client.hostname* to a custom value for the *cluster.id* parameter (available in the */etc/zdp-hive/zdp-hive.yml*, */etc/zdp-spark/zdp-spark.yml*, and */etc/zdp-mr/zdp-mr.yml* files). This custom value must be identical in all the applicable yml files. In the below screenshot, the custom value is *my-cluster*.

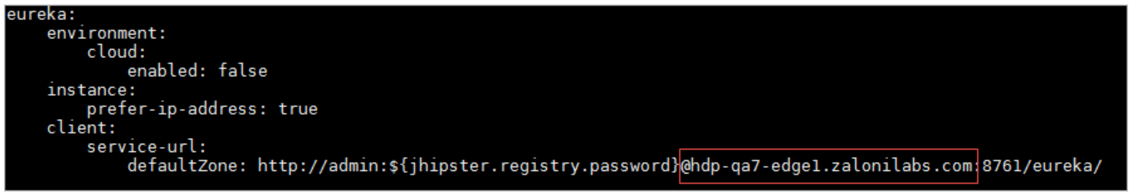
   For example,



```
# cluster id defaulted to hostname
cluster.id: my-cluster
```

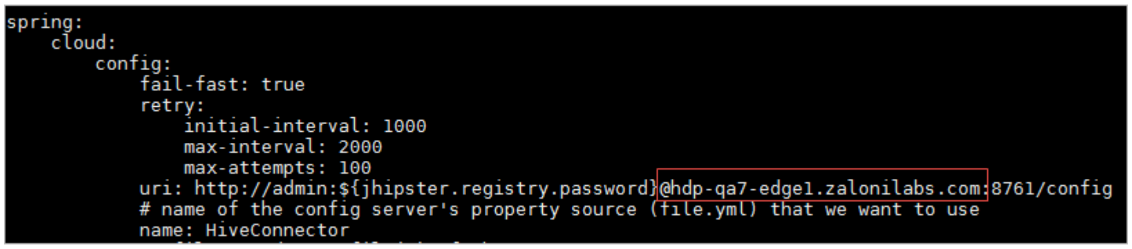
2. Replace *localhost* of the following properties with the hostname of the server where the *zdp-registry* service is installed (Applicable services are: *zdp-spark*, *zdp-executor*, *zdp-mr*, *zdp-ingestion-warden*, *zdp-hive*, *zdp-gateway*, *zdp-lineage*, *zdp-logstash*, *zdp-kibana*, and *zdp-dme* (optional)):

   - The *eureka.client.service.url.defaultZone* property in the */etc/<service_name>/<service_name>.yml* file.



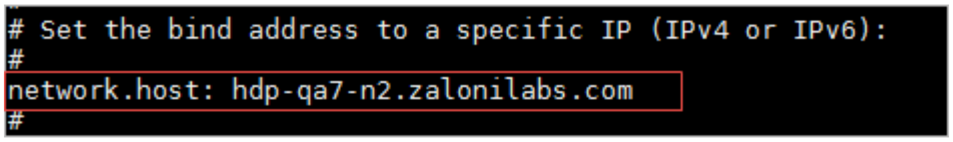
```
eureka:
    environment:
        cloud:
            enabled: false
    instance:
        prefer-ip-address: true
    client:
        service-url:
            defaultZone: http://admin:${jhipster.registry.password}@hdp-qa7-edge1.zalonilabs.com:8761/eureka/
```

- The *spring.cloud.config.uri* property in the */etc/<service-name>/bootstrap-prod.yml* file. (This file is not applicable for *zdp-kibana* service)



```
spring:
    cloud:
        config:
            fail-fast: true
            retry:
                initial-interval: 1000
                max-interval: 2000
                max-attempts: 100
            uri: http://admin:${jhipster.registry.password}@hdp-qa7-edge1.zalonilabs.com:8761/config
            # name of the config server's property source (file.yml) that we want to use
            name: HiveConnector
```

3. In the *network.host* parameter in the */etc/zdp-es/elasticsearch.yml* file, update the hostname for the server where the *zdp-es* service is installed.



```
# Set the bind address to a specific IP (IPv4 or IPv6):
#
network.host: hdp-qa7-n2.zalonilabs.com
#
```

4. Update the following properties in the */etc/zdp-registry/application.yml* file:

```
activemq.broker.url: Specify the URI that allows you to run a configured broker␣
↪by using a single URI for all the configuration.
```

```
activemq.broker.sslurl: Specify the secured URI that allows you to run a␣
↪Configured broker by using a single URI for all the configuration.
```

```
activemq.username: Specify the username to access the broker.
```

```
activemq.password: Specify the password to access the broker.
```

```
zdp.datasource.url: Specify the URL to access the MySQL database. Replace␣
↪localhost with the hostname of the server where MySQL is installed.
```

```
zdp.datasource.username: Specify the username to access the MySQL database.
```

```
zdp.datasource.password: Specify the password to access the MySQL database.
```

```
elasticsearch.cluster.name: Optional. Specify the cluster name for the server␣
↪where the zdp-es service is run.
```

```
elasticsearch.connection.tcp: Indicates the comma-separated hostname:port to␣
↪connect to multiple nodes where the zdp-es services are installed. For example,␣
↪node1.example.com:19300,node2.example.com:19300.
```

```
elasticsearch.connection.uri: Indicates the comma-separated protocol://
↪hostname:port to connect to multiple nodes where the zdp-es services are␣
↪installed. If protocol is not defined, the value is assumed to be http://. For␣
↪example, http://node1.example.com:19200,http://node2.example.com:19200.
```

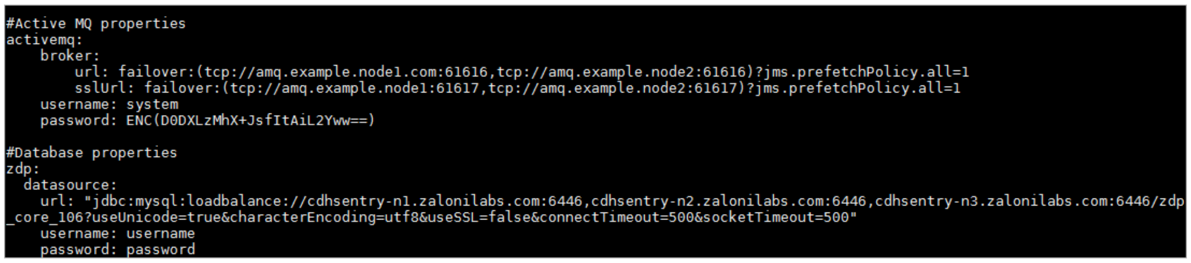
```
elasticsearch.connection.mode.node.port: Indicates the port where the zdp-es␣
↪service is installed. For example, 19300.
```

```
logstash.host: Specify the hostname for the server where the zdp-logstash service␣
↪is run.
```

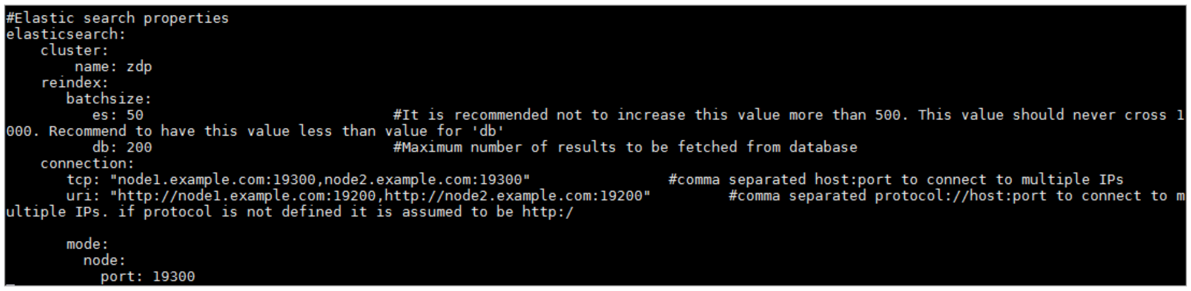
```
logstash.http.port: Optional. Specify the http port for the server where the zdp-
↪logstash service is run.
```

```
logstash.tcp.port: Optional. Specify the tcp port for the server where the zdp-
↪logstash service is run.
```

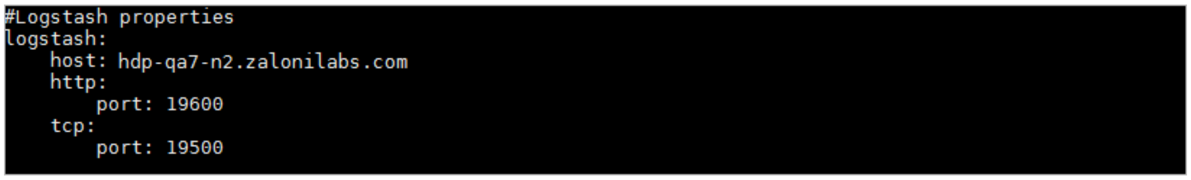
For example,



```
#Active MQ properties
activemq:
    broker:
        url: failover:(tcp://amq.example.node1.com:61616,tcp://amq.example.node2:61616)?jms.prefetchPolicy.all=1
        sslUrl: failover:(tcp://amq.example.node1:61617,tcp://amq.example.node2:61617)?jms.prefetchPolicy.all=1
    username: system
    password: ENC(D0DXLzMhX+JsfItAiL2Yww==)

#Database properties
zdp:
  datasource:
    url: "jdbc:mysql:loadbalance://cdhsentry-n1.zalonilabs.com:6446,cdhsentry-n2.zalonilabs.com:6446,cdhsentry-n3.zalonilabs.com:6446/zdp
_core_106?useUnicode=true&characterEncoding=utf8&useSSL=false&connectTimeout=500&socketTimeout=500"
    username: username
    password: password
```



```
#Elastic search properties
elasticsearch:
    cluster:
        name: zdp
    reindex:
      batchsize:
         es: 50                            #It is recommended not to increase this value more than 500. This value should never cross 1
000. Recommend to have this value less than value for 'db'
         db: 200                           #Maximum number of results to be fetched from database
    connection:
      tcp: "node1.example.com:19300,node2.example.com:19300"                #comma separated host:port to connect to multiple IPs
      uri: "http://node1.example.com:19200,http://node2.example.com:19200"         #comma separated protocol://host:port to connect to m
ultiple IPs. if protocol is not defined it is assumed to be http:/

      mode:
        node:
          port: 19300
```



```
#Logstash properties
logstash:
    host: hdp-qa7-n2.zalonilabs.com
    http:
        port: 19600
    tcp:
        port: 19500
```

Note: If you update the properties, such as *cluster.name*, *http.port*, and *tcp.port* in the */etc/<service-name>/<service-name>.yml* file, you must update those in the */etc/zdp-registry/application.yml* file.

5. Restart the services. For more information on starting the ZDP services, refer to the *Application Start-up and Shut-down* section.

Note: If you make any changes in the *zdp-services* files, restart all the services.

1.3.3 Cloud Specific Configuration

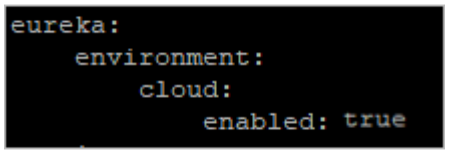
After the installation of ZDP in the cloud environment, perform the following post installation configurations:

- To use a public IP as the ZDP URL, configure the IP in *BEDROCK_URL* under **Administration** > **System Configuration Management**. In the EMR/Azure environment, ZDP cannot populate the public IP for this property and instead only displays the local IP.

  Note: This configuration is applicable to any cloud installation.

- To register public IP and hostname of the cluster microservices (*zdp-mr*, *zdp-spark*, *zdp-hive*, and *zdp-dme*) in the *zdp-registry* service, set the *eureka.environment.cloud.enabled* property as *true* (default value is *false*) in the */etc/zdp-<cluster_microservices>/zdp-<cluster_microservices>.yml* files.

  For example,

```
eureka:
    environment:
        cloud:
            enabled: true
```

  Note: This configuration is applicable to any cloud installation. The default value: *false* will work if the services can communicate over the private IP (as per the network setup).

- To successfully ingest file from an edge node to the EMR cluster, ensure the connectivity between the EMR nodes and the EC2 edge node. The EC2 edge node and the EMR cluster nodes must be connected to the same LAN or VLAN. The HDFS architecture expects that the edge node has access to all the nodes (via IP addresses) that are part of the HDFS cluster. If an EC2 node is considered as an edge node and you want to access an EMR cluster, you must ensure that all the nodes that are part of the EMR cluster are accessible from the EC2 edge node.

  Note: This configuration is applicable to EMR distribution and AWS.

- To avoid getting permission denied exception in the Yarn log location, access the *yarn-site.xml* file and set the parameter: *yarn.nodemanager.remote-app-log-dir* to a HDFS path with the 777 permission. Alternatively, you can set the same key in either of the **MR/Spark Configurations**, **Spark Configurations**, or **MR Configurations** field along with the path value, while defining individual workflow actions that generate Yarn logs. For example, *conf yarn.nodemanager.remote-app-log-dir=/tmp/zdp-temp*. However, ensure that the specified path has the 777 permission.

  Note: This configuration is applicable to any cloud installation.

- To link the EMR cluster to the ZDP platform, click **AWS Link** under **Administration> Cluster Connection Management**. On clicking this option, the system navigates to the Amazon AWS website where you can create/connect clusters.

  Note: This configuration is applicable to EMR distribution.

1.3.4 JVM Argument Configuration

To configure the JVM arguments (such as -Xmx, -Xms) for the various ZDP services (*zdp-gateway*, *zdp-executor*, etc), perform the following steps:

1. Access the following file for a specific service:

```
/etc/sysconfig/zdp-<daemon>

For example,

For the zdp-gateway service: /etc/sysconfig/zdp-gateway
```

2. Set the *JAVA_OPTS* parameter to the various arguments as follows:

```
JAVA_OPTS="<JVM arguments>"

For example,

JAVA_OPTS="-Xmx2000m -Xms256m"
```

1.3.4.1 JVM argument in MapR environment

To perform ingestion in a MapR environment, add the following JVM argument in the */opt/zdp/zdp-ingestion-warden/bin/agents/bdca/dca-launch.sh* file:

```
"-Djava.security.auth.login.config=/opt/mapr/conf/mapr.login.conf -Dhadoop.
↪login=hybrid"
```

1.3.5 Managing Memory Utilization for Multiple Workflows

To optimize the disk utilization and to periodically clean up the H2 database (that stores data that is generated during workflow execution), the system cleans the existing H2 database entries from the previous executed workflows. If the status of a executed workflow is *COMPLETED*, *ABANDONED*, or *FAILED*, the system cleans up the details for such a workflow, based on the values set for the following properties in the */opt/zdp/zdp-executor/config/zdp-executor.yml* file:

```
spring.batch.history.cleanup.scheduled.cronExpression: Allows you to set the trigger␣
↪time for cleanup. Default value is 0 55 23 * * ? to indicate that the cleanup␣
↪activity must be performed at 11:55 PM daily.
```

```
spring.batch.history.cleanup.enabled: Allows you to enable periodic cleanup of the H2␣
↪database. This property is set to true (by default). Additionally, during the␣
↪startup of the zdp-executor, the system cleans up the entire H2 database.
```

Note: The cleanup activity can take place only if the *zdp-executor* service is up.

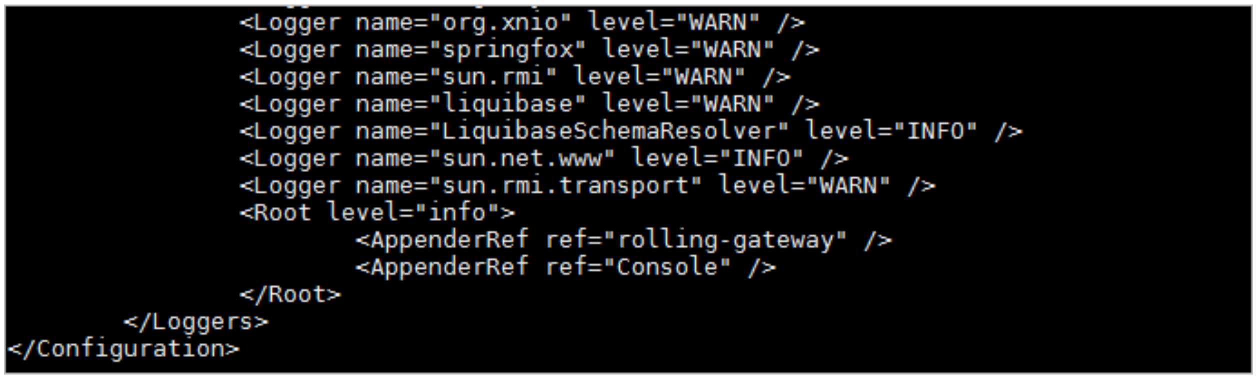
1.3.6 Logging Settings for the ZDP Services

The logging settings for the ZDP services (such as *zdp-gateway/zdp-executor/zdp-ingestion-warden*) are stored in the */etc/zdp-<service_name>/log4j2.xml* file. log4j2 is the logging engine. By default, log4j2 is configured to report *INFO* level messages. This can be changed by modifying the log4j2 Root Level property. The possible logging levels are:

- *DEBUG*
- *INFO*

- *ERROR*
- *WARN*

Excerpt of log4j2.xml.properties:



```
                <Logger name="org.xnio" level="WARN" />
                <Logger name="springfox" level="WARN" />
                <Logger name="sun.rmi" level="WARN" />
                <Logger name="liquibase" level="WARN" />
                <Logger name="LiquibaseSchemaResolver" level="INFO" />
                <Logger name="sun.net.www" level="INFO" />
                <Logger name="sun.rmi.transport" level="WARN" />
                <Root level="info">
                        <AppenderRef ref="rolling-gateway" />
                        <AppenderRef ref="Console" />
                </Root>
        </Loggers>
</Configuration>
```

To change the logging level, perform the following steps:

1. Update the Root Level property in the *log4j2.xml* file.
2. Restart the specific service.

Note: After the successful installation of the ZDP, edit the values of the required keys available in the system_view section. For example, for the key *BEDROCK_HDFS_LIB_DIR*, edit the parameter as: */tmp/hdfs-jars*

1.4 High Availability for MySQL

1.4.1 Overview

Zaloni Data Platform (ZDP) application accomplishes high availability (HA) at the database storage layer. HA for MySQL is handled by leveraging the MySQL InnoDB cluster feature.

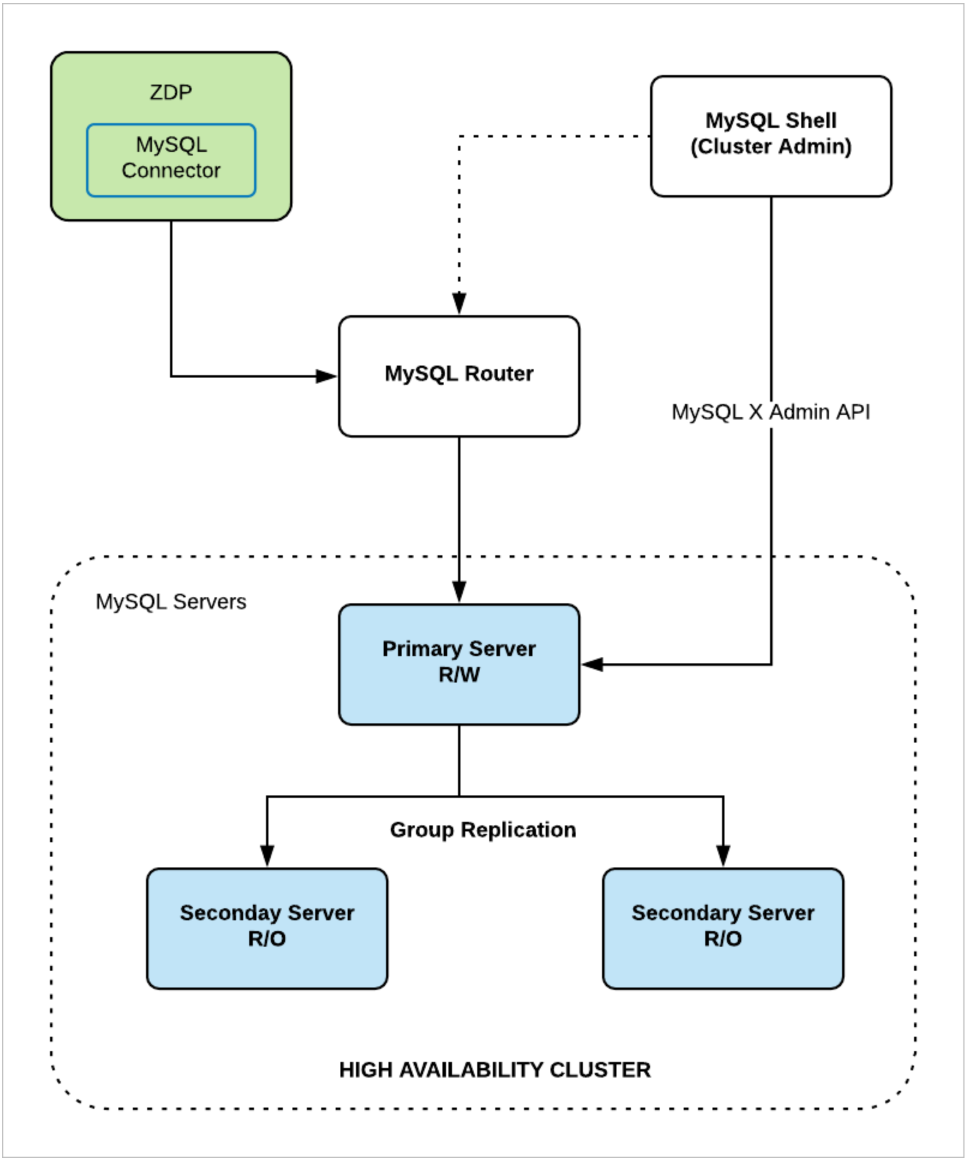
MySQL InnoDB cluster delivers an integrated, native, HA solution for your databases. MySQL InnoDB cluster consists of the following components:

- **MySQL Server** with group replication to replicate data to all nodes of the cluster.
- **MySQL Router** to ensure client requests are load balanced and routed to the correct servers in case of any database failures. MySQL router configures itself automatically based on the cluster deployment and transparently routes ZDP application requests to the appropriate primary server instance. If server instance failure occurs, MySQL router detects those failures and delegates client requests to a healthy primary server instance. If the primary server fails, one of the secondary server automatically becomes the primary server. The default is to have one primary server, but multiple primary servers can be configured.
- **MySQL Shell** to create and administer InnoDB clusters with at least three nodes.

With regards to data replication, MySQL HA relies on group replication and this is transparent to the user. If you use ZDP with the HA feature for MySQL, note that the primary keys in ZDP (for example, workflow ID) gets incremented by the default value of 7 (configurable).

Note: For more information, refer to HA for MySQL.

The following diagram shows an overview of the HA MySQL InnoDB cluster and ZDP integration.



1.4.2 Pre-requisites

- MySQL 5.7
- CentOS 7
- Python and wget:

  If python and wget is not installed in all the nodes of the MySQl cluster, run the below command:

  ```
  #yum install python wget
  ```

- Update the */etc/hosts* file in all the nodes of the MySQL cluster:

  Sample update:

```
10.11.13.186    rvsql-node1.zalonilabs.com
10.11.13.182    rvsql-node2.zalonilabs.com
10.11.13.183    rvsql-node3.zalonilabs.com
```

- Disable SELinux and firewall
  - Access the */etc/sysconfig/config* file, as follows:

    ```
    vi /etc/selinux/config
    ```

  - Set the *SELINUX* parameter to *disabled*.

    ```
    SELINUX=disabled
    ```

  - Disable the firewall, as follows:

    ```
    systemctl stop firewalld;systemctl disable firewalld
    ```

Note: For more information on disabling SELinux, refer to Disabling SELinux.

1.4.3 Installation and Configuration

To use the HA feature for MySQL, you must install/configure various MySQL components. To do so, perform the following steps for the respective MySQL components:

1.4.3.1 MySQL Server

Important: You must be a root user to perform the below tasks.

1. To download the MySQL repository, run the following command:

   ```
   #wget https://repo.mysql.com//mysql57-community-release-el7-10.noarch.rpm
   ```

2. To install the MySQL repository, run the following command:

   ```
   #yum install mysql57-community-release-el7-10.noarch.rpm
   ```

1.4.3.2 MySQL Shell

Important: You must be a root user to perform the below tasks.

1. To install MySQL server/shell, run the following command on all the nodes of the MySQL cluster:

   ```
   #yum install mysql-server mysql-shell
   ```

2. To start the MySQL service, run the following command on all the nodes of the MySQL cluster:

```
#service mysqld start;chkconfig mysqld on
```

3. To fetch the temporary root password, run the following command on all the nodes of the MySQL cluster:

```
#grep 'A temporary password is generated for root@localhost' /var/log/mysqld.log␣
↪|tail -1
```

4. To reset the new MySQL password of the root user, run the following command on all the nodes of the MySQL cluster:

```
#mysql_secure_installation
```

5. To create *zaloni* user, run the following commands on all the nodes of the MySQL cluster:

```
[root@rvsql-node1 lib]# mysql -p
mysql> create user 'zaloni'@'%' identified by 'password';
```

6. To provide all the privileges to the *zaloni* user, run the following commands on all the nodes of the MySQL cluster:

```
mysql> grant all privileges on *.* to 'zaloni'@'%' with grant option;
mysql> flush privileges;
```

7. To configure the local instance calling the following function as show below in all the nodes of the MySQL cluster, run the following command:

```
mysqlsh --uri root@localhost:3306
MySQL  localhost:3306 ssl  JS > dba.configureLocalInstance('root@localhost:3306')
Please provide the password for 'root@localhost:3306': ***********
```

The system displays the following output:

```
Configuring local MySQL instance listening at port 3306 for use in an InnoDB␣
↪cluster...
This instance reports its own address as rvsql-node1.zalonilabs.com
Clients and other cluster members will communicate with it through this address␣
↪by default. If this is not correct, the report_host MySQL system variable␣
↪should be changed.

WARNING: User 'root' can only connect from localhost.
If you need to manage this instance while connected from other hosts, new␣
↪account(s) with the proper source address specification must be created.

1) Create remotely usable account for 'root' with same grants and password
2) Create a new admin account for InnoDB cluster with minimal required grants
3) Ignore and continue
4) Cancel
```

Specify appropriate values for the system-prompted tasks, as follows:

```
Please select an option [1]: 2
Please provide an account name (e.g: icroot@%) to have it created with the␣
↪necessary privileges or leave empty and press Enter to cancel.
Account Name: zaloni
```

The system displays the following output:

```
User 'zaloni'@'%' already exists and will not be created.

Some configuration options need to be fixed:
+----------------------------------+---------------+----------------+-------------
↪------------------------------------+
| Variable                         | Current Value | Required Value | Note        ␣
↪                                    |
+----------------------------------+---------------+----------------+-------------
↪------------------------------------+
| binlog_checksum                  | CRC32         | NONE           | Update the␣
↪server variable                      |
| enforce_gtid_consistency         | OFF           | ON             | Update read-
↪only variable and restart the server |
| gtid_mode                        | OFF           | ON             | Update read-
↪only variable and restart the server |
| log_bin                          | 0             | 1              | Update read-
↪only variable and restart the server |
| log_slave_updates                | 0             | ON             | Update read-
↪only variable and restart the server |
| master_info_repository           | FILE          | TABLE          | Update read-
↪only variable and restart the server |
| relay_log_info_repository        | FILE          | TABLE          | Update read-
↪only variable and restart the server |
| server_id                        | 0             | <unique ID>    | Update read-
↪only variable and restart the server |
| transaction_write_set_extraction | OFF           | XXHASH64       | Update read-
↪only variable and restart the server |
+----------------------------------+---------------+----------------+-------------
↪------------------------------------+

The following variable needs to be changed, but cannot be done dynamically: 'log_
↪bin'

Detecting the configuration file...
Found configuration file at standard location: /etc/my.cnf
```

Specify appropriate values for the system-prompted tasks, as follows:

```
Do you want to modify this file? [y/N]: y
Do you want to perform the required configuration changes? [y/n]: y
```

The system displays the following output:

```
Configuring instance...
The instance 'localhost:3306' was configured for cluster usage.
MySQL server needs to be restarted for configuration changes to take effect.
```

Restart the MySQL service in all the nodes within the MySQL cluster, as follows:

```
sudo service mysqld stop/start
```

8. To check if MySQL shell is installed, run the MySQL shell command in all the nodes of the MySQL cluster, as follows:

```
[root@rvsql-node1 ~]# mysqlsh
```

The system displays the following output:

```
MySQL Shell 8.0.11

Copyright (c) 2016, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.␣
↪Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
```

Specify appropriate values for the system-prompted tasks, as follows:

```
MySQL  JS > dba.checkInstanceConfiguration('zaloni@rvsql-node1.zalonilabs.com:3306
↪');
Please provide the password for 'zaloni@rvsql-node1.zalonilabs.com:3306':␣
↪***********
```

The system displays the following output:

```
Validating local MySQL instance listening at port 3306 for use in an InnoDB␣
↪cluster...

This instance reports its own address as rvsql-node1.zalonilabs.com
Clients and other cluster members will communicate with it through this address␣
↪by default. If this is not correct, the report_host MySQL system variable␣
↪should be changed.

Checking whether existing tables comply with Group Replication requirements...
No incompatible tables detected

Checking instance configuration...
Note: verifyMyCnf option was not given so only dynamic configuration will be␣
↪verified.
Instance configuration is compatible with InnoDB cluster

The instance 'rvsql-node1.zalonilabs.com:3306' is valid for InnoDB cluster usage.

{
 "status": "ok"
}
```

9. To create MySQL cluster, run the following command:

```
MySQL  rvsql-node1.zalonilabs.com:3306 ssl  JS > var cluster = dba.createCluster(
↪'zalonicluster',{memberSslMode:'DISABLED'});
```

The system displays the following output:

```
A new InnoDB cluster will be created on instance 'zaloni@rvsql-node1.zalonilabs.
↪com:3306'.

Validating instance at rvsql-node1.zalonilabs.com:3306...

This instance reports its own address as rvsql-node1.zalonilabs.com

Instance configuration is suitable.
Creating InnoDB cluster 'zalonicluster' on 'zaloni@rvsql-node1.zalonilabs.com:3306
↪'...
```

```
WARNING: On instance 'rvsql-node1.zalonilabs.com:3306' membership change cannot␣
↪be persisted since MySQL version 5.7.22 does not support the SET PERSIST␣
↪command (MySQL version >= 8.0.5 required). Please use the <Dba>.
↪configureLocalInstance command locally to persist the changes.
Adding Seed Instance...

Cluster successfully created. Use Cluster.addInstance() to add MySQL instances.
At least 3 instances are needed for the cluster to be able to withstand up to one␣
↪server failure.
```

While executing the above command, you might see the following error:

```
Dba.createCluster: Dba.createCluster: An open session is required to perform this␣
↪operation. (RuntimeError)
```

To overcome such an error, run the following command:

```
\connect --mc zaloni@rvsql-node1.zalonilabs.com:3306
```

10. To add the second node in the cluster, run the following command:

```
MySQL  rvsql-node1.zalonilabs.com:3306 ssl  JS > cluster.addInstance(
↪'zaloni@rvsql-node2.zalonilabs.com:3306',{memberSslMode:'DISABLED'})
```

The system displays the following output:

```
A new instance will be added to the InnoDB cluster. Depending on the amount of␣
↪data on the cluster this might take from a few seconds to several hours.
```

Specify appropriate values for the system-prompted tasks, as follows:

```
Please provide the password for 'zaloni@rvsql-node2.zalonilabs.com:3306':␣
↪***********
```

The system displays the following output:

```
Adding instance to the cluster ...

Validating instance at rvsql-node2.zalonilabs.com:3306...

This instance reports its own address as rvsql-node2.zalonilabs.com

Instance configuration is suitable.
WARNING: On instance 'rvsql-node2.zalonilabs.com:3306' membership change cannot␣
↪be persisted since MySQL version 5.7.22 does not support the SET PERSIST␣
↪command (MySQL version >= 8.0.5 required). Please use the <Dba>.
↪configureLocalInstance command locally to persist the changes.
WARNING: On instance 'rvsql-node1.zalonilabs.com:3306' membership change cannot␣
↪be persisted since MySQL version 5.7.22 does not support the SET PERSIST␣
↪command (MySQL version >= 8.0.5 required). Please use the <Dba>.
↪configureLocalInstance command locally to persist the changes.
The instance 'zaloni@rvsql-node2.zalonilabs.com:3306' was successfully added to␣
↪the cluster.
```

11. To add the third node in the cluster, run the following command:

```
MySQL  rvsql-node1.zalonilabs.com:3306 ssl  JS > cluster.addInstance(
↪'zaloni@rvsql-node3.zalonilabs.com:3306',{memberSslMode:'DISABLED'})
```

The system displays the following output:

```
A new instance will be added to the InnoDB cluster. Depending on the amount of␣
↪data on the cluster this might take from a few seconds to several hours.
```

Specify appropriate values for the system-prompted tasks, as follows:

```
Please provide the password for 'zaloni@rvsql-node3.zalonilabs.com:3306':␣
↪***********
```

The system displays the following output:

```
Adding instance to the cluster ...

Validating instance at rvsql-node3.zalonilabs.com:3306...

This instance reports its own address as rvsql-node3.zalonilabs.com

Instance configuration is suitable.
WARNING: On instance 'rvsql-node3.zalonilabs.com:3306' membership change cannot␣
↪be persisted since MySQL version 5.7.22 does not support the SET PERSIST␣
↪command (MySQL version >= 8.0.5 required). Please use the <Dba>.
↪configureLocalInstance command locally to persist the changes.
WARNING: On instance 'rvsql-node1.zalonilabs.com:3306' membership change cannot␣
↪be persisted since MySQL version 5.7.22 does not support the SET PERSIST␣
↪command (MySQL version >= 8.0.5 required). Please use the <Dba>.
↪configureLocalInstance command locally to persist the changes.
The instance 'zaloni@rvsql-node3.zalonilabs.com:3306' was successfully added to␣
↪the cluster.
```

12. To check the cluster status, run the following command:

```
MySQL  rvsql-node1.zalonilabs.com:3306 ssl  JS > cluster.status()
```

The system displays the following output:

```
{
 "clusterName": "zalonicluster",
 "defaultReplicaSet": {
  "name": "default",
  "primary": "rvsql-node1.zalonilabs.com:3306",
  "ssl": "DISABLED",
  "status": "OK",
  "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
  "topology": {
      "rvsql-node1.zalonilabs.com:3306": {
          "address": "rvsql-node1.zalonilabs.com:3306",
          "mode": "R/W",
          "readReplicas": {},
          "role": "HA",
          "status": "ONLINE"
      },
      "rvsql-node2.zalonilabs.com:3306": {
          "address": "rvsql-node2.zalonilabs.com:3306",
          "mode": "R/O",
          "readReplicas": {},
          "role": "HA",
          "status": "ONLINE"
      },
      "rvsql-node3.zalonilabs.com:3306": {
          "address": "rvsql-node3.zalonilabs.com:3306",
          "mode": "R/O",
          "readReplicas": {},
          "role": "HA",
          "status": "ONLINE"
      }
  }
 },
  "groupInformationSourceMember": "mysql://zaloni@rvsql-node1.zalonilabs.com:3306"
 }
```

13. If you exit from the terminal and want to check the cluster status again, run the following command:

```
#mysqlsh --uri zaloni@rvsql-node1.zalonilabs.com:3306
MySQL  rvsql-node1.zalonilabs.com:3306 ssl  JS > var cluster = dba.getCluster(
↪'zalonicluster');
MySQL  rvsql-node1.zalonilabs.com:3306 ssl  JS > cluster.status();
```

1.4.3.3 MySQL Router

Important: You must be a mysql user to perform the below tasks.

1. To install MySQL router in a node within the MYSQL cluster, run the following command in the specific node:

```
#yum install mysql-router
```

2. To create the required directories and provide appropriate permission in the node, run the following command:

```
#mkdir -p /opt/mysql/router;chown -R mysql:mysql /opt/mysql/router
```

3. To establish connection between the MySQL router node and the other nodes within the MySQl cluster (that hosts the primary/secondary MySQL servers), run the following command in the router node:

```
[root@rvsql-node1 ~]# su - mysql
-bash-4.2$ mysqlrouter --bootstrap zaloni@rvsql-node1.zalonilabs.com:3306 --
↪directory /opt/mysql/router
Please enter MySQL password for zaloni:
```

The system displays the following output:

```
Reconfiguring MySQL Router instance at '/opt/mysql/router'...
MySQL Router  has now been configured for the InnoDB cluster 'zalonicluster'.

The following connection information can be used to connect to the cluster.

Classic MySQL protocol connections to cluster 'zalonicluster':
- Read/Write Connections: localhost:6446
- Read/Only Connections: localhost:6447
X protocol connections to cluster 'zalonicluster':
- Read/Write Connections: localhost:64460
- Read/Only Connections: localhost:64470
```

If the router node needs to establish connection with multiple primary/secondary MySQL servers, specify the comma-separated addresses for all such servers. For example, run the following command:

```
-bash-4.2$ mysqlrouter --bootstrap zaloni@projb-n1.zalonilabs.com:3306,projb-n3.
↪zalonilabs.com:3306,projb-n2.zalonilabs.com:3306,projb-n4.zalonilabs.com:3306 --
↪directory /opt/mysql/router
```

4. To update and start the MySQL router service, run the following command:

```
sudo mysqlrouter --bootstrap zaloni@projb-n[1,2,3,4].zalonilabs.com:3306 --
↪user=mysqlrouter --directory /etc/mysqlrouter/ --force

sudo service mysqlrouter start
```

5. To keep consistency on all the nodes across the cluster, run the following command in all the nodes:

```
[root@rvsql-node1 lib]# mysqlsh
```

The system displays the following output:

```
MySQL Shell 8.0.11

Copyright (c) 2016, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.␣
↪Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
```

Specify appropriate values for the system-prompted tasks, as follows:

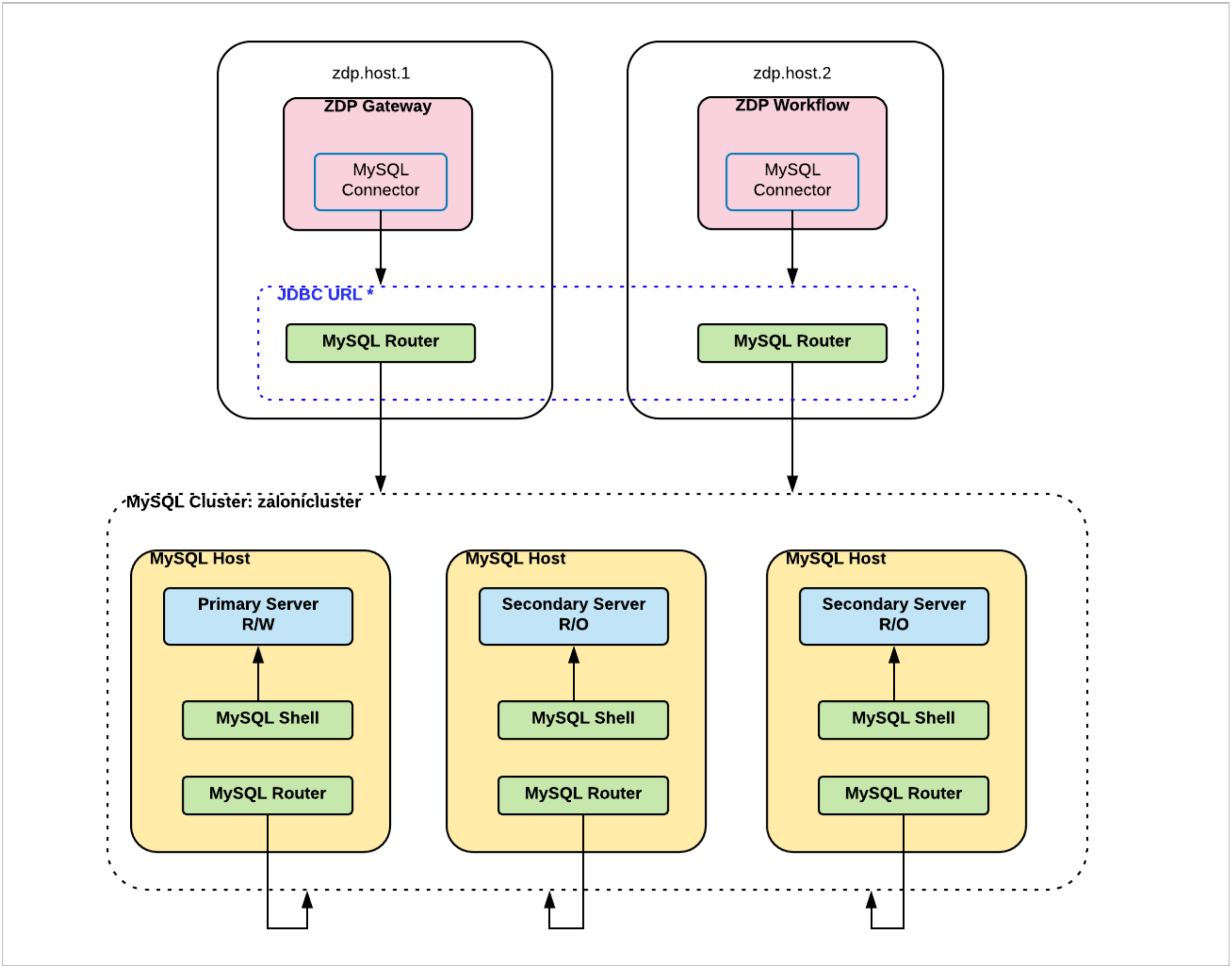
```
MySQL  JS > dba.configureLocalInstance('zaloni@rvsql-node1.zalonilabs.com:3306');
Please provide the password for 'zaloni@rvsql-node1.zalonilabs.com:3306':␣
↪***********
```

The system displays the following output:

```
The instance 'rvsql-node1.zalonilabs.com:3306' belongs to an InnoDB cluster.

Detecting the configuration file...
Found configuration file at standard location: /etc/my.cnf
Do you want to modify this file? [y/N]: y
Persisting the cluster settings...
The instance 'rvsql-node1.zalonilabs.com:3306' was configured for use in an␣
↪InnoDB cluster.

The instance cluster settings were successfully persisted.
```

1.4.4 Recommended Cluster Configuration

There are many ways to deploy a MySQL InnoDB cluster. Zaloni recommends the following cluster deployment:



ZDP Client

- For better performance, it is recommended that MySQL routers are installed in all the ZDP client hosts (for example, *zdp-gateway*, *zdp-executor*) as recommended by MySQL. However, this is not mandatory.
- For a cluster with single *zdp-gateway* and *zdp-executor* services, set the *LimitNOFILE* parameter to *1500* in the */lib/systemd/system/mysqlrouter.service* file. However, you must set a higher value while using multiple *zdp-gateway* and *zdp-executor* services.
- For a cluster with single *zdp-gateway* and *zdp-executor* services, set the *max_connections* parameter to *1500* (in the */etc/mysqlrouter.conf* file for MySQL router and in the */etc/my.cnf* file for MySQL cluster instances). However, if there are multiple *zdp-gateway/zdp-executor* services, you must reconfigure the value for this parameter.
- ZDP clients must point to the multi-host JDBC URL that references to the list of MySQL routers. This JDBC URL is configured in the *application.yml* file within the *zdp-registry* service. For more information, refer to the *Single Node Configuration* section.

  MySQL connector handles the load balancing and redirects to a healthy MySQL router endpoint. If any of the client hosts are not available, ZDP clients can point to any MySQL router installed in the MySQL cluster.
- ZDP does not recommend you to use the HA for MySQL feature with multi-primary servers (MySQL instances with group replication). This is because multi-level database relationships with *CASCADE* foreign keys are not allowed. For more information, refer to the MySQL limitations.

MySQL Cluster

- The recommended number of nodes is three. Having a three node cluster allows a single node failure. You can increase the number of nodes to accommodate more node failures.
- Single primary server is recommended. Multi primary servers may be used but performance risks are introduced.
- All cluster nodes must have MySQL shell. This may allow only the users with the required permissions to log in to the cluster and administer the same.
- (Optional) All cluster nodes must have MySQL routers. These processes are on standby mode if all the MySQL routers in the ZDP client host fail.
- The routers must allow read/write operation from ZDP. This is required to perform all the necessary ZDP operations in the MySQL database via the routers.

Important:

- If the primary MySQL server is down (in a MySQL cluster with one primary server), you cannot perform write operation in the MySQL database till a new primary server is elected.
- If a router is shut down in a MySQL cluster (with multiple routers), you cannot perform write operation in the MySQL database till a new router is brought up again.

1.4.4.1 Working with the InnoDB cluster

Working with InnoDB cluster

1.4.5 References

- MySQL 5.7 InnoDB Cluster User Guide
- MySQL 5.7 Group Replication
- InnoDB Tutorial
- For troubleshooting tips, refer to the *Error in MySQL cluster and ZDP connection* section.

1.5 High Availability for Elasticsearch

To achieve high availability for Elasticsearch, multiple Elasticsearch nodes must be clustered together where a single node behaves as the master node and the other nodes become the data nodes. Multiple replica of data is distributed in these nodes. Hence, if a node becomes unavailable, the data can be fetched from the other nodes. When a new node is created, Elasticsearch schema and data is copied to that node instantly. Also, if the master node becomes unavailable, one of the data nodes behave as the master node to avoid any downtime. Each node has information about the other nodes.

Important: Kibana does not support HA for Elasticsearch. Hence, if a Elasticsearch node (that is part of the Elasticsearch cluster) becomes unavailable, you may not be able to view the details in the **Dashboard** tab.

To configure HA for the *zdp-es* service, perform the following steps:

1. Create an Elasticsearch cluster that consists of multiple Elasticsearch nodes. To do so, configure the following properties in the */etc/zdp-es/elasticsearch.yml* files for all the nodes that are part of the Elasticsearch cluster:

```
cluster.name: Optional. Indicates the name of the Elasticsearch cluster that␣
↪includes all the data nodes and the master node. You must specify an identical␣
↪value for this parameter in all the elasticsearch.yml files. Although this␣
↪property is optional, it is recommended that you update a custom value (default␣
↪value is zdp). For example, my_es_cluster.
```

```
node.name: Optional. Indicates the name for the specific node. You must specify a␣
↪unique name for each node. Although this property is optional, it is␣
↪recommended that you specify a custom value. The system auto-generates the␣
↪value (if not specified). For example, my_es_node1.
```

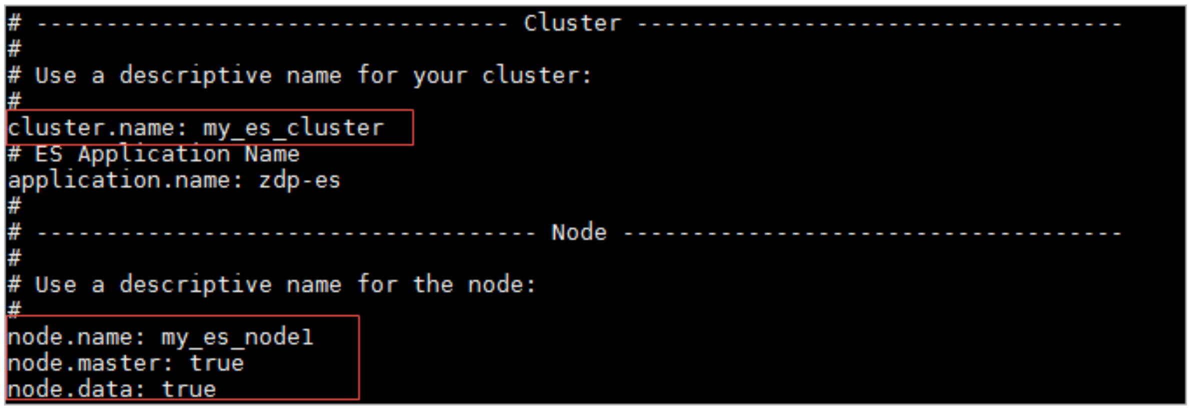
```
node.master: Optional. Set this parameter to true to indicate that the node can␣
↪become a master node. If this parameter is set to true in multiple nodes, the␣
↪node that starts first, becomes the master node. If the master node is␣
↪unavailable, any other node (with this parameter set to true) can become the␣
↪master node. If you set both the parameters: node.data and node.master to true,␣
↪the same node can act as the master node, while also storing data. However, if␣
↪you set node.master to true and node.data to false, the master node does not␣
↪store data. Possible values are true (default value) and false.
```

```
node.data: Optional. Set this parameter to true to indicate that the node can␣
↪store data. If you set both the parameters: node.data and node.master to true,␣
↪the same node can act as the master node, while also storing data. However, if␣
↪you set node.master to true and node.data to false, the master node does not␣
↪store data. Possible values are true (default value) and false.
```

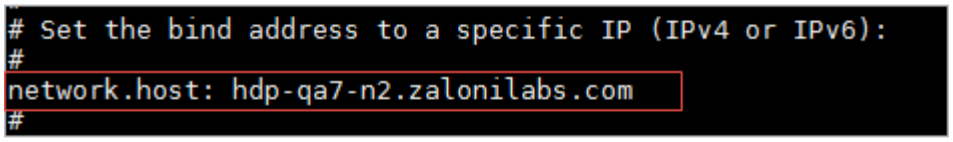
```
network.host: Optional. Indicates the  hostname for the server where the zdp-es␣
↪service is installed. Although this property is optional, it is recommended␣
↪that you update a custom value. The default value is 0.0.0.0 that indicates␣
↪that the zdp-es service listens to all the network interfaces attached to this␣
↪node. For example, hdp-qa7-n2-zalonilabs.com.
```

```
discovery.zen.ping.unicast.hosts: Indicates all the nodes (in an array) that␣
↪contribute to the Elasticsearch cluster.
```

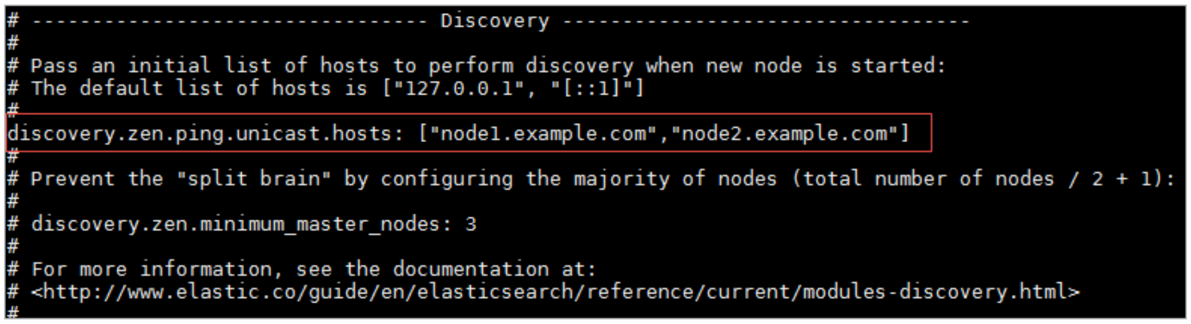
For example,



```
# ---------------------------------- Cluster -----------------------------------
#
# Use a descriptive name for your cluster:
#
cluster.name: my_es_cluster
# ES Application Name
application.name: zdp-es
#
# ------------------------------------ Node ------------------------------------
#
# Use a descriptive name for the node:
#
node.name: my_es_node1
node.master: true
node.data: true
```



```
# Set the bind address to a specific IP (IPv4 or IPv6):
#
network.host: hdp-qa7-n2.zalonilabs.com
#
```



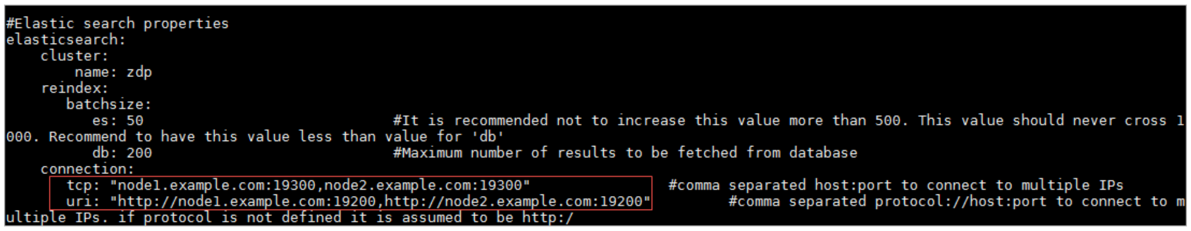
```
# ---------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when new node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
discovery.zen.ping.unicast.hosts: ["node1.example.com","node2.example.com"]
#
# Prevent the "split brain" by configuring the majority of nodes (total number of nodes / 2 + 1):
#
# discovery.zen.minimum_master_nodes: 3
#
# For more information, see the documentation at:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/modules-discovery.html>
#
```

2. Specify multiple nodes (that are part of the Elasticsearch cluster) in the */etc/zdp-registry/application.yml* file, as follows:

```
elasticsearch.connection.tcp: Indicates the comma-separated hostname:port to␣
↪connect to multiple nodes where the zdp-es services are installed. For example,␣
↪node1.example.com:19300,node2.example.com:19300.
```

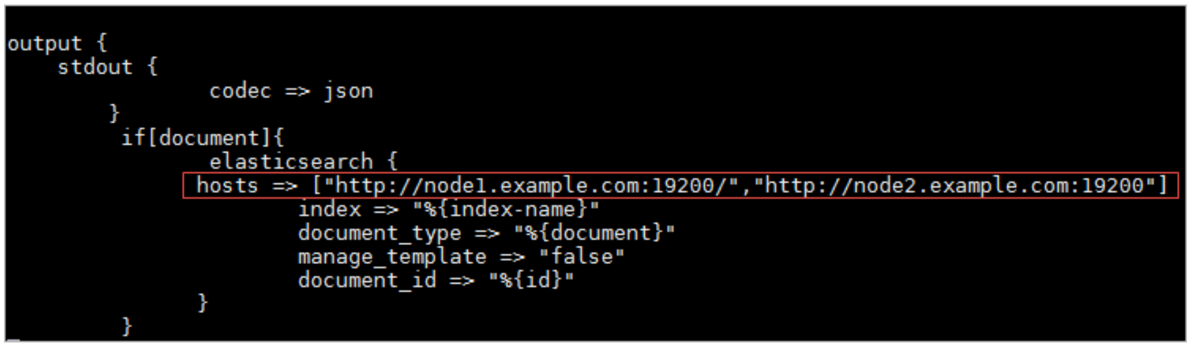
```
elasticsearch.connection.uri: Indicates the comma-separated protocol://
↪hostname:port to connect to multiple nodes where the zdp-es services are␣
↪installed. If protocol is not defined, the value is assumed to be http://. For␣
↪example, http://node1.example.com:19200,http://node2.example.com:19200.
```

For example,



```
#Elastic search properties
elasticsearch:
    cluster:
        name: zdp
    reindex:
       batchsize:
          es: 50                            #It is recommended not to increase this value more than 500. This value should never cross 1
000. Recommend to have this value less than value for 'db'
          db: 200                           #Maximum number of results to be fetched from database
    connection:
       tcp: "node1.example.com:19300,node2.example.com:19300"    #comma separated host:port to connect to multiple IPs
       uri: "http://node1.example.com:19200,http://node2.example.com:19200"        #comma separated protocol://host:port to connect to m
ultiple IPs. if protocol is not defined it is assumed to be http:/
```

3. In the */etc/zdp-logstash/conf.d/logstash-socket.conf* file, update the parameter: *hosts* to include all the Elasticsearch nodes that are already defined in the application.yml file. For example, *[“http://node1.example.com:19200/”,”http://node2.example.com:19200”]*.



```
output {
    stdout {
                codec => json
        }
         if[document]{
                elasticsearch {
                hosts => ["http://node1.example.com:19200/","http://node2.example.com:19200"]
                        index => "%{index-name}"
                        document_type => "%{document}"
                        manage_template => "false"
                        document_id => "%{id}"
               }
         }
```

4. Start the *zdp-es* service in all the nodes.

5. Start the *zdp-gateway* service so that indices get created in the Elasticsearch cluster. For more information, refer to the *Recreating Indices in Elasticsearch* section.

1.6 High Availability for ActiveMQ

To achieve high availability for ActiveMQ, multiple ActiveMQ brokers must be clustered together. The messages from the *zdp-gateway* services are sent to one of the brokers, which are then forwarded to the *zdp-executor* service for workflow execution. If the *zdp-executor* services connected to that cluster are unavailable, one of the brokers stores the messages temporarily and routes the messages to a *zdp-executor* service (whenever the service becomes available). If one of the brokers become unavailable, the messages from *zdp-gateway* are sent to an active broker.

To use HA for ActiveMQ, perform the following steps:

1. To set up a cluster of multiple brokers (so that individual brokers are aware of the presence of other brokers within the same cluster and can communicate with one another), add the following properties in the */etc/zdp-activemq/activemq.xml* file (located within the <broker> and </broker> tags):

   ```
   userName: Specify the username to access the nodes where the ActiveMQ brokers are␣
   ↪installed.
   ```

   ```
   password: Specify the password to access the nodes where the ActiveMQ brokers are␣
   ↪installed.
   ```

   ```
   uri: Specify the comma-separated values for all the nodes where the ActiveMQ␣
   ↪brokers are installed.
   ```

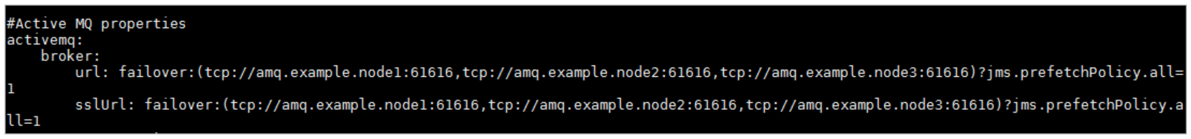
   For example,

   ```
   <broker xmlns="http://activemq.apache.org/schema/core" brokerName="localhost"␣
   ↪dataDirectory="${activemq.data}">
       <!--
        <networkConnectors>
               <networkConnector
                     userName="system"
                     password="manager"
              uri="static:(tcp://amq.example.node1:61616,tcp://amq.example.
   ↪node2:61616,tcp://amq.example.node3:61616)"/>
    </networkConnectors>
        -->
   </broker>
   ```

2. To enable failover among the brokers, update the */etc/zdp-registry/application.yml* file, as follows:

   ```
   activemq.broker.url: Specify the hostnames:port for all the nodes where the␣
   ↪ActiveMQ brokers are installed. For example, failover:(tcp://amq.example.
   ↪node1:61616,tcp://amq.example.node2:61616,tcp://amq.example.node3:61616)?jms.
   ↪prefetchPolicy.all=1.
   ```

   ```
   activemq.broker.sslUrl: Specify the hostname:port for all the nodes where the␣
   ↪ActiveMQ brokers are installed. For example, failover:(tcp://amq.example.
   ↪node1:61617,tcp://amq.example.node2:61617,tcp://amq.example.node3:61617)?jms.
   ↪prefetchPolicy.all=1.
   ```



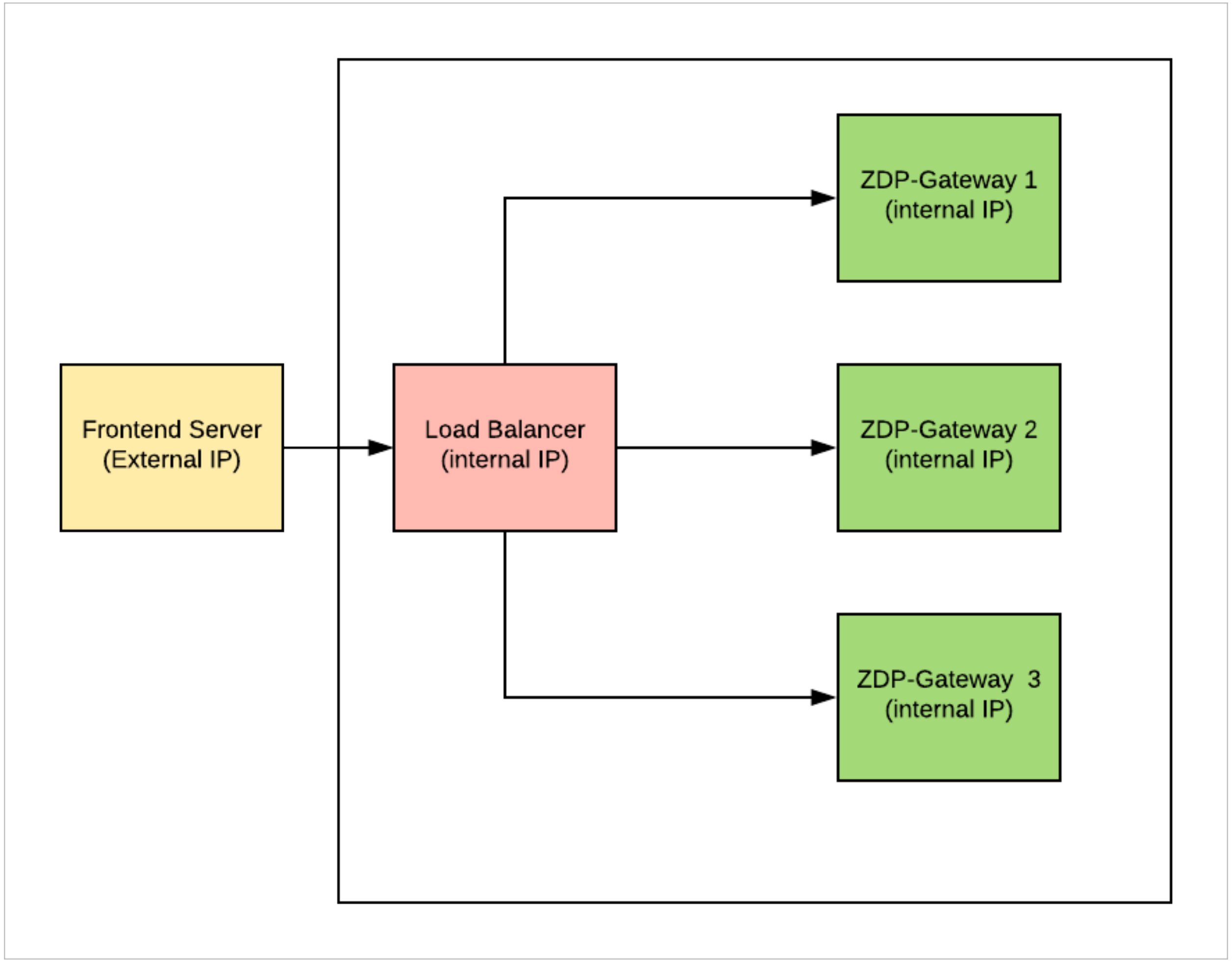
```
#Active MQ properties
activemq:
    broker:
        url: failover:(tcp://amq.example.node1:61616,tcp://amq.example.node2:61616,tcp://amq.example.node3:61616)?jms.prefetchPolicy.all=
1
        sslUrl: failover:(tcp://amq.example.node1:61616,tcp://amq.example.node2:61616,tcp://amq.example.node3:61616)?jms.prefetchPolicy.a
ll=1
```

1.7 High Availability for ZDP-Gateway

ZDP allows you to use multiple *zdp-gateway* services by using load balancers. A load balancer distributes the application traffic across a cluster of servers, thereby increasing the availability of web applications. A load balancer accepts incoming network and application traffic and distributes the traffic across multiple servers. By balancing the traffic across multiple servers, a load balancer reduces individual server load and prevents a server from becoming a single point of failure.

Load balancers enable you to scale out applications. As new servers are added to the resource pool, the load balancer immediately begin sending traffic to the new server. When one application server becomes unavailable, the load balancer directs all new application requests to other available servers in the pool.

Important: If you use a load balancer to load balance the traffic between multiple *zdp-gateway* services, you must access the ZDP UI by using the URL of the load balancer and not the individual *zdp-gateway* services. If you access the *zdp-gateway* service, the system does not load balance the traffic.


Frontend Server (External IP)
Load Balancer (internal IP)
ZDP-Gateway 1 (internal IP)
ZDP-Gateway 2 (internal IP)
ZDP-Gateway 3 (internal IP)

1.7.1 Pre-requisites

- Multiple nodes where the *zdp-gateway* service is installed.
- A load balancer. For example, HAProxy, Nginx, etc.

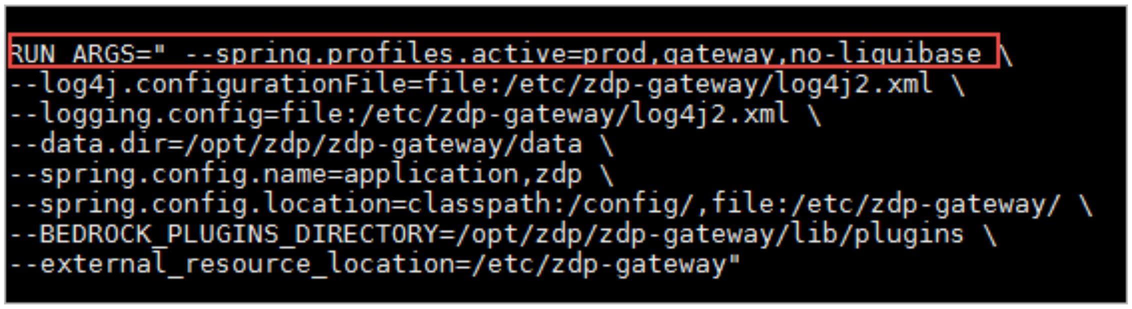
1.7.2 Installation and Configuration

1.7.2.1 ZDP-Gateway

Access the various nodes where you want to install the *zdp-gateway* service, install the service, and start the service. Note that you must not start multiple *zdp-gateway* services simultaneously (during initial startup).

To use multiple *zdp-gateway* services, perform the following steps:

1. Access a node where you want to install the *zdp-gateway* service, install the service, and start the service.
2. Access the other nodes where the subsequent *zdp-gateway* services need to be installed.
   (a) In all such nodes, install the *zdp-gateway* services and access the */etc/zdp-gateway/zdp-gateway-<version>.conf* files:



```
RUN_ARGS=" --spring.profiles.active=prod,gateway,no-liquibase \
--log4j.configurationFile=file:/etc/zdp-gateway/log4j2.xml \
--logging.config=file:/etc/zdp-gateway/log4j2.xml \
--data.dir=/opt/zdp/zdp-gateway/data \
--spring.config.name=application,zdp \
--spring.config.location=classpath:/config/,file:/etc/zdp-gateway/ \
--BEDROCK_PLUGINS_DIRECTORY=/opt/zdp/zdp-gateway/lib/plugins \
--external_resource_location=/etc/zdp-gateway"
```

   (b) To disable the liquibase configuration, pass the value *no-liquibase* for the argument: *spring.profiles.active*.
   (c) Start all the services.

Important: You must not start multiple *zdp-gateway* services simultaneously (during initial startup).

1.7.2.2 Load Balancer - HAProxy

HA for the *zdp-gateway* service can be achieved by using any load balancers. Zaloni recommends HAProxy as it has been well-tested for high availability of ZDP Gateways. HAProxy offers high availability, load balancing, and proxying for TCP and HTTP-based applications.

To use HAProxy as a load balancer for the *zdp-gateway* service, perform the following steps:

1. To install HAProxy, execute the following command:

```
yum -y install haproxy
```

2. To configure the various nodes, access the */etc/haproxy/haproxy.cfg* file as follows:

```
vi /etc/haproxy/haproxy.cfg
```

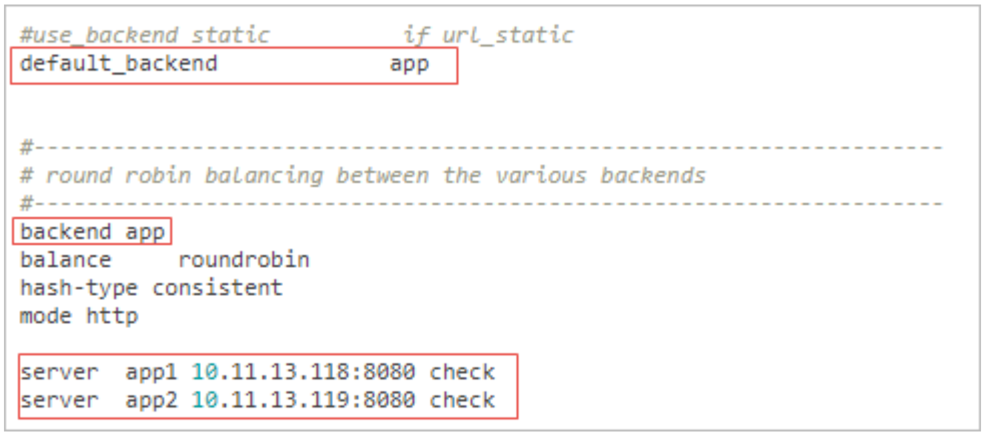
3. Update the configuration file as follows:

```
#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
# to have these messages end up in /var/log/haproxy.log you will
# need to:
```

```
#
# 1) configure syslog to accept network log events.  This is done
#    by adding the '-r' option to the SYSLOGD_OPTIONS in
#    /etc/sysconfig/syslog
#
# 2) configure local2 events to go to the /var/log/haproxy.log
#   file. A line like the following can be added to
#   /etc/sysconfig/syslog
#
#    local2.*                       /var/log/haproxy.log
#
#log         10.11.13.118

chroot      /var/lib/haproxy
pidfile     /var/run/haproxy.pid
maxconn     4000
user        haproxy
group       haproxy
daemon

# turn on stats unix socket
stats socket /var/lib/haproxy/stats

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
mode                    http
log                     global
option                  httplog
option                  dontlognull
option http-server-close
option forwardfor       except 127.0.0.0/8
option                  redispatch
retries                 3
timeout http-request    10s
timeout queue           1m
timeout connect         10s
timeout client          1h
timeout server          1h
timeout http-keep-alive 10s
timeout check           10s
maxconn                 3000

#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend  main
bind *:90
option http-server-close
acl url_static       path_beg       -i /static /images /javascript /stylesheets
acl url_static       path_end       -i .jpg .gif .png .css .js

#use_backend static          if url_static
default_backend             app
```

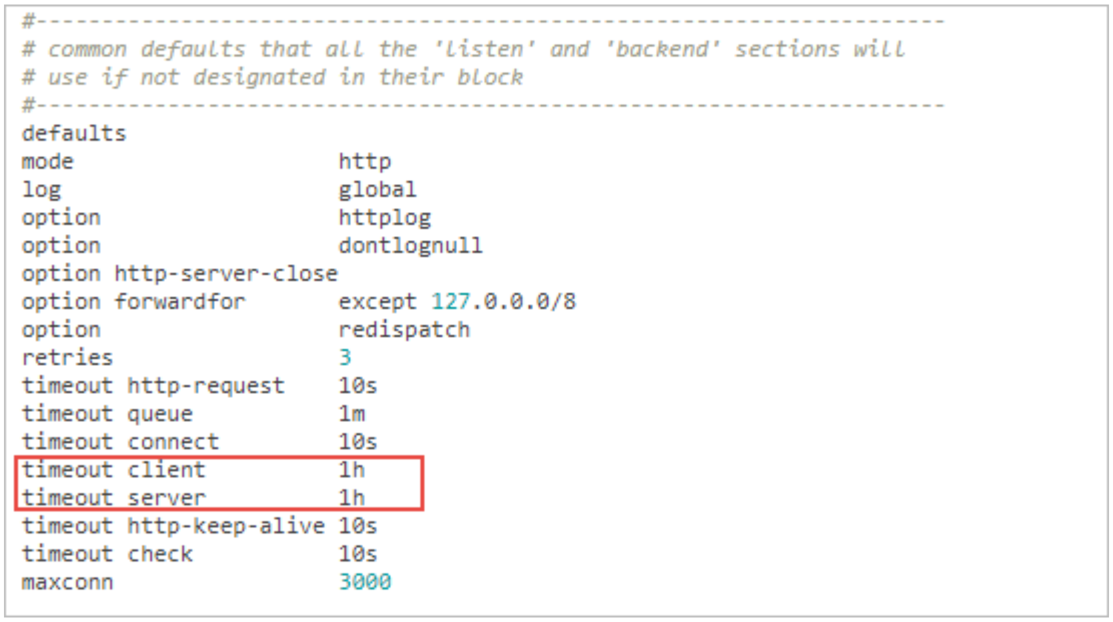
```
#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend app
balance     roundrobin
hash-type consistent
mode http

server  app1 10.11.13.118:8080 check
server  app2 10.11.13.119:8080 check
```

Important: Set the *default_end* parameter to a specific value (warden), call the specific value, and define numerous instances for the *zdp-gateway* services.



```
#use_backend static          if url_static
default_backend             app


#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend app
balance     roundrobin
hash-type consistent
mode http

server  app1 10.11.13.118:8080 check
server  app2 10.11.13.119:8080 check
```

Important: Update the values for the *timeout client* and *timeout server* parameters to 1h.



```
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
mode                    http
log                     global
option                  httplog
option                  dontlognull
option http-server-close
option forwardfor       except 127.0.0.0/8
option                  redispatch
retries                 3
timeout http-request    10s
timeout queue           1m
timeout connect         10s
timeout client          1h
timeout server          1h
timeout http-keep-alive 10s
timeout check           10s
maxconn                 3000
```

1.7.3 Recommended load balancers

Any load balancer.

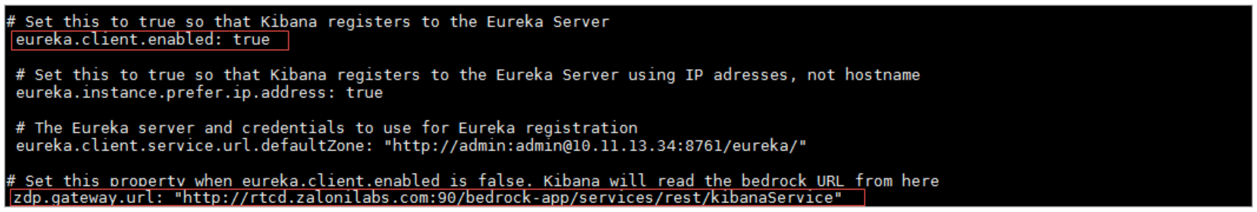
Note: Zaloni recommends HAProxy as it has been well-tested for high availability of ZDP Gateways.

1.7.4 High availability for Kibana with Gateway

You can use Kibana with multiple *zdp-gateway* services, such that if one of the *zdp-gateway* service becomes unavailable, one of the others becomes available and Kibana works in sync with the Gateway.

To leverage that feature, perform the following steps:

- In the */etc/zdp-kibana/kibana.yml* file, set the value for the *eureka.client.enabled* parameter as *true*. It is recommended that you always set this value as *true* in order to get the *zdp-kibana* service registered with the *zdp-registry* service. However, this value can be set as *false* if you do not want Kibana to register itself to the *zdp-registry* service.
- In the */etc/zdp-kibana/kibana.yml* file, set the value for the *zdp.gateway.url* parameter as the URL of the load balancer. In a non-HA environment, you can set the value as the URL of the *zdp-gateway* service.

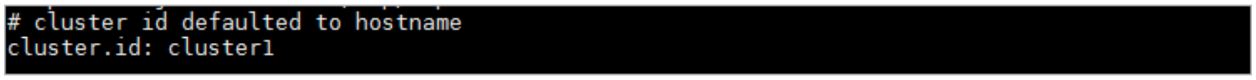


```
# Set this to true so that Kibana registers to the Eureka Server
eureka.client.enabled: true

 # Set this to true so that Kibana registers to the Eureka Server using IP adresses, not hostname
 eureka.instance.prefer.ip.address: true

 # The Eureka server and credentials to use for Eureka registration
 eureka.client.service.url.defaultZone: "http://admin:admin@10.11.13.34:8761/eureka/"

# Set this property when eureka.client.enabled is false. Kibana will read the bedrock URL from here
zdp.gateway.url: "http://rtcd.zalonilabs.com:90/bedrock-app/services/rest/kibanaService"
```

1.8 High Availability for Cluster Microservices

ZDP allows you to configure multiple cluster microservices that belong to the same cluster. If you do so, you can ensure that there is at least one active cluster microservice in a cluster. For example, if there are three *zdp-hive* services within a cluster and one of these services go down, there are two other services that can perform the same tasks. This is also applicable for *zdp-mr* and *zdp-spark* services.

To enable high availability for the cluster microservices (*zdp-spark/zdp-hive/zdp-mr*), ensure that you set the parameter: *cluster.id* (in the */etc/zdp-<cluster_microservices>/zdp-<cluster_microservices>.yml* files) to the same value in all the instances where the specific cluster microservice is installed. For example, there are three nodes that are part of a cluster and you have installed the *zdp-mr* service in all these nodes. To use the high availability feature for the *zdp-mr* service, you must access all the nodes and set the *cluster.id* parameter (in all the *zdp-mr.yml* files) to the same value. Note that you must not specify the hash (#) special character in the value for the *cluster.id* parameter.



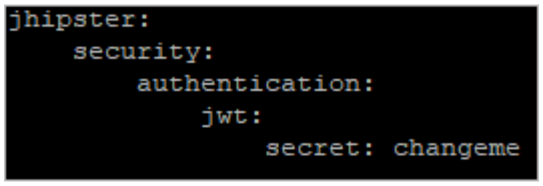
```
# cluster id defaulted to hostname
cluster.id: cluster1
```

1.9 Securing Configuration Files

By default, certain sensitive values (such as connection passwords) are either hashed or encrypted in the database. However, you can obfuscate sensitive information in the various yml files of the ZDP services.

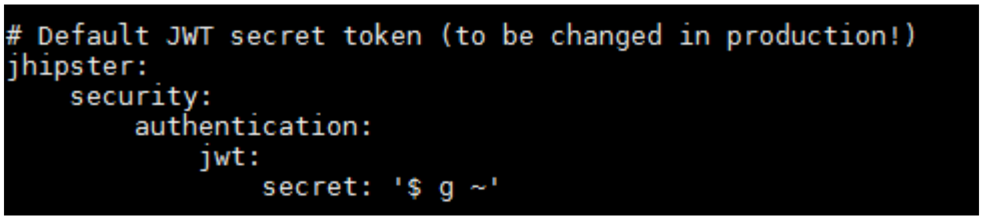
For example, if you want to encrypt MySQL database password that contains special characters, perform the following steps:

1. Update the value for the *jhipster.security.authentication.jwt.secret* parameter in the */etc/zdp-registry/application.yml* to a new value (for example, $ g ~). The default value is *changeme*.



```
jhipster:
    security:
        authentication:
            jwt:
                secret: changeme
```

Once you update the value for the secret parameter, the screenshot appears as:



```
# Default JWT secret token (to be changed in production!)
jhipster:
    security:
        authentication:
            jwt:
                secret: '$ g ~'
```

Important: If the secret parameter value contains special characters, ensure that you enqoute the value in single quotes or escape the same with backslash (\) as per the yaml conventions at YAML conventions.

2. Run the command:

```
sudo /opt/zdp/zdp-registry/bin/encrypt.sh  input='<MySQL password that needs to␣
↪be encrypted>' password='$ g ~'
```

For example,

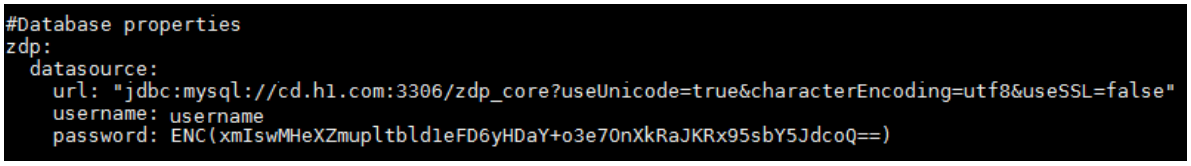
```
sudo /opt/zdp/zdp-registry/bin/encrypt.sh input='a$ $/^&a' password='$ g ~'
```

Here, `a$ $/^&a` is the MySQL database password that needs to be encrypted.

The generated output is:

```
----ENVIRONMENT-----------------
Runtime: Oracle Corporation Java HotSpot(TM) 64-Bit Server VM 25.131-b11
----ARGUMENTS-------------------

input: 'a$ $/^&a'
password: '$ g ~'

----OUTPUT----------------------
xmIswMHeXZmupltbld1eFD6yHDaY+o3e7OnXkRaJKRx95sbY5JdcoQ==
```

3. Copy the output and set the values in the */etc/zdp-registry/application.yml* file for the MySQL credentials:



```
#Database properties
zdp:
  datasource:
    url: "jdbc:mysql://cd.h1.com:3306/zdp_core?useUnicode=true&characterEncoding=utf8&useSSL=false"
    username: username
    password: ENC(xmIswMHeXZmupltbld1eFD6yHDaY+o3e7OnXkRaJKRx95sbY5JdcoQ==)
```

Note: *ENC()* is used to wrap the encrypted passwords. You can mix plain and encrypted passwords in your yml files. Encrypting yml files is not supported by Elasticsearch, Kibana, and Logstash services.

4. Restart the services. For more information on starting the ZDP services, refer to the *Application Start-up and Shut-down* section.

   Similarly, you can encrypt password from other yml files, such as *zdp-gateway*, *zdp-executor*, *zdp-spark*, *zdp-mr*, *zdp-hive*, *zdp-lineage*, and *zdp-ingestion-warden*. However, you cannot encrypt passwords in specific yml files, such as bootstrap*.yml (within any ZDP service)/zdp-registry.yml files.

5. By default, ZDP rpm configuration files are installed at /etc/zdp* with a file permission of 644. To increase the security, you can change file permission to 400 by using the following command:

```
find /etc/zdp* -type f -not -path '*/\.*'-exec chmod 400 -- {} +
```

1.10 ZDP over HTTPS

This section is mandatory when you want to configure ZDP over HTTPS (HTTP over SSL). This protocol encrypts the communication between client and server. ZDP supports HTTPS for the following services:

- *zdp-registry*
- *zdp-gateway*
- *zdp-executor*
- *zdp-ingestion-warden*
- *zdp-spark*
- *zdp-mr*
- *zdp-hive*
- *zdp-lineage*
- *zdp-dme*
- *zdp-activemq*
- *zdp-kibana*

Important: To enable HTTPS for a specific service, place the keystore/truststore files in a common location that is accessible to the service user. For example, zaloni. However, even if you do not place these in a common location, ensure that the service user has access to such locations. You can have a common truststore file for all the nodes where the ZDP services are deployed (in case of multi node deployment). This truststore file contains the certificates issued by a CA. However, in case you have multiple truststore files (separate truststore files for individual nodes), ensure that you add all the certificates of all the nodes in all such truststore files. This is required as all the ZDP services must be able to trust every other ZDP services. Additionally, it is advisable to have individual keystore files for all the separate nodes (where the ZDP services run).

Note: While using the high availability feature for the zdp-gateway service in the HTTPS mode, you can set up HTTPS for both the load balancer (for example, HAProxy) node as well as the node where the zdp-gateway service runs. If you want to do so, the HAProxy node must have the CA trust chain of the node where the zdp-gateway service runs.

To enable HTTPS on the ZDP services, perform the following steps:

1. Add/update the following properties in the */etc/zdp-{service_name}/zdp-{service_name}.yml* file (here, {service_name} can be *zdp-registry/zdp-gateway/zdp-executor/zdp-ingestion-warden/zdp-spark/zdp-mr/zdp-hive/zdp-lineage/zdp-dme*):

```
eureka.instance.nonSecurePortEnabled: Set this property to false to disable non␣
↪secure port for the specific ZDP service.
```

```
eureka.instance.securePortEnabled: Set this property to true to enable secure␣
↪port for the specific ZDP service.
```

```
eureka.instance.prefer-ip-address: Set this property to false to enable HTTPS for␣
↪the specific ZDP service.
```

```
eureka.instance.securePort: Set the port to access the HTTPS-enabled specific ZDP␣
↪service.
```

```
eureka.instance.hostname: Set the hostname to access the HTTPS-enabled specific␣
↪ZDP service. You can also find the hostname for a specific service in the /etc/
↪hosts location.
```

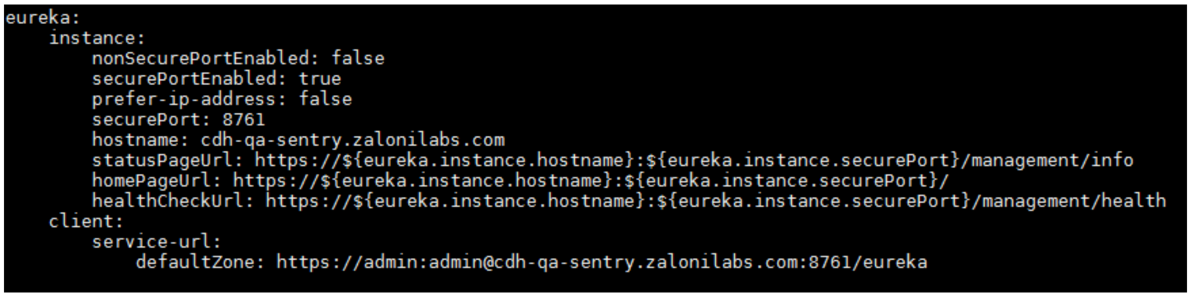
```
eureka.instance.statusPageUrl: Set the URL for the status page. Ensure that the␣
↪protocol is https. For example: https://${eureka.instance.hostname}:${eureka.
↪instance.securePort}/management/info.
```

```
eureka.instance.homePageUrl: Set the URL for the home page. Ensure that the␣
↪protocol is https. For example, https://${eureka.instance.hostname}:${eureka.
↪instance.securePort}/.
```

```
eureka.instance.healthCheckUrl: Set the URL to check the health status. Ensure␣
↪that the protocol is https. For example, https://${eureka.instance.hostname}:$
↪{eureka.instance.securePort}/management/health.
```

```
eureka.client.service-url.defaultZone: Set the protocol to https, the username/
↪password to access the zdp-registry service, and the hostname/port for the zdp-
↪registry service.
```

   For example,



```
eureka:
    instance:
        nonSecurePortEnabled: false
        securePortEnabled: true
        prefer-ip-address: false
        securePort: 8761
        hostname: cdh-qa-sentry.zalonilabs.com
        statusPageUrl: https://${eureka.instance.hostname}:${eureka.instance.securePort}/management/info
        homePageUrl: https://${eureka.instance.hostname}:${eureka.instance.securePort}/
        healthCheckUrl: https://${eureka.instance.hostname}:${eureka.instance.securePort}/management/health
    client:
        service-url:
            defaultZone: https://admin:admin@cdh-qa-sentry.zalonilabs.com:8761/eureka
```

2. Define the values for the following properties in the */etc/zdp-{service_name}/zdp-{service_name}.yml* file (here,

{service_name} can be *zdp-registry/zdp-gateway/zdp-executor/zdp-ingestion-warden/zdp-spark/zdp-mr/zdp-hive/zdp-lineage/zdp-dme*):

```
server.ssl.key-store: Specify the path to the keystore file for the specific ZDP␣
↪service. Ensure that the zaloni service user has full access to this path. A␣
↪sample value is /opt/zdp/certs/zdp-keystore.jks.
```

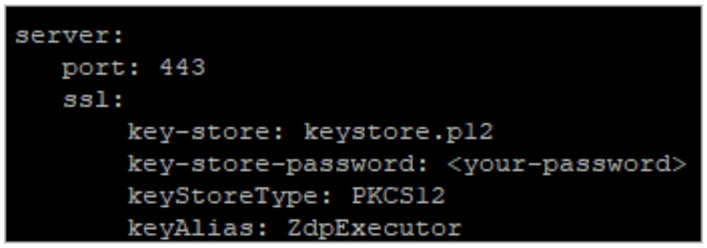
```
server.ssl.key-store-password: Specify the password to access the keystore file␣
↪for the specific ZDP service. For example, password.
```

```
server.ssl.keyStoreType: Specify the keystore type for the specific ZDP service.␣
↪For example, PKCS12 or JKS.
```

```
server.ssl.keyAlias: Specify the alias name for the key. This value is case␣
↪sensitive. Ensure that the value is identical to the one defined in keystore.␣
↪Example value: zdp-server.
```

```
server.ssl.enabled: Set this parameter to true to run the ZDP services in the␣
↪HTTPS mode.
```

For example,



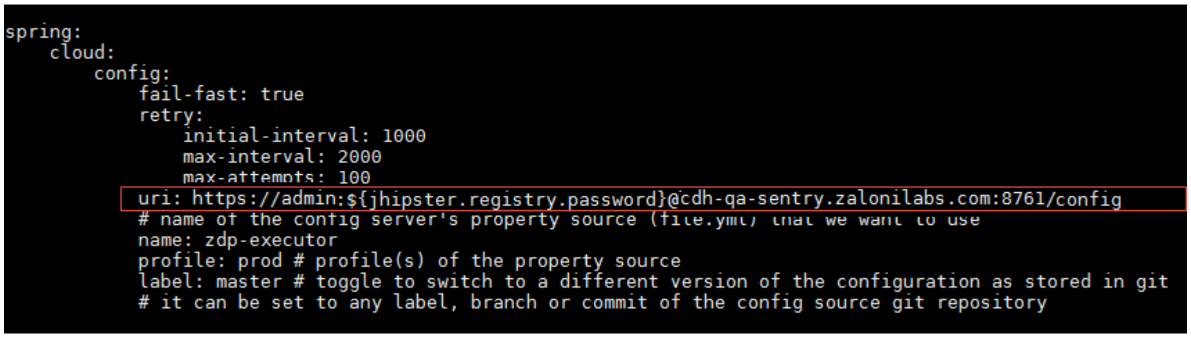
```
server:
   port: 443
   ssl:
       key-store: keystore.p12
       key-store-password: <your-password>
       keyStoreType: PKCS12
       keyAlias: ZdpExecutor
```

3. For the above listed services, import their respective custom truststore to the Java cacerts location by using the following command:

```
    sudo keytool -importkeystore -srckeystore  <location of the custom truststore>
↪   -destkeystore  <location of the Java truststore>

For example,

    sudo keytool -importkeystore -srckeystore  /opt/zdp/certs/zdp-truststore.jks ␣
↪-destkeystore  /usr/java/jdk1.8.0_131/jre/lib/security/cacerts
```

4. In the */etc/zdp-{service_name}/bootstrap-prod.yml*, update the *spring.cloud.config.uri* parameter to indicate the protocol as https and the hostname/port for the *zdp-registry* service. Here, applicable services are *zdp-executor/zdp-mr/zdp-spark/zdp-hive/zdp-dme/zdp-ingestion-warden/zdp-lineage/zdp-gateway*:

   For example,



```
spring:
    cloud:
        config:
            fail-fast: true
            retry:
                initial-interval: 1000
                max-interval: 2000
                max-attempts: 100
            uri: https://admin:${jhipster.registry.password}@cdh-qa-sentry.zalonilabs.com:8761/config
            # name of the config server's property source (file.yml) that we want to use
            name: zdp-executor
            profile: prod # profile(s) of the property source
            label: master # toggle to switch to a different version of the configuration as stored in git
            # it can be set to any label, branch or commit of the config source git repository
```

5. For the *zdp-kibana* service, add the following properties in the */etc/zdp-kibana/kibana.yml*:

```
server.ssl.enabled: Set this parameter to true to indicate that SSL is enabled on the node where the zdp-kibana service is installed. If you do so, the communication between Kibana and the browser is SSL enabled.
```

```
server.ssl.cert: Set the path to access the certificate files for securing the node where the zdp-kibana service is installed. A sample value is /opt/zdp/certs/zdp-keystore.crt.
```

```
server.ssl.key: Set the path to access the key files for securing the node where the zdp-kibana service is installed. A sample value is `/opt/zdp/certs/zdp-keystore.key.
```

```
elasticsearch.ssl.cert: Set the path to access the certificate files for securing the node where the zdp-gateway service is installed. A sample value is `/opt/zdp/certs/zdp-beta-n4.zalonilabs.com.crt.
```

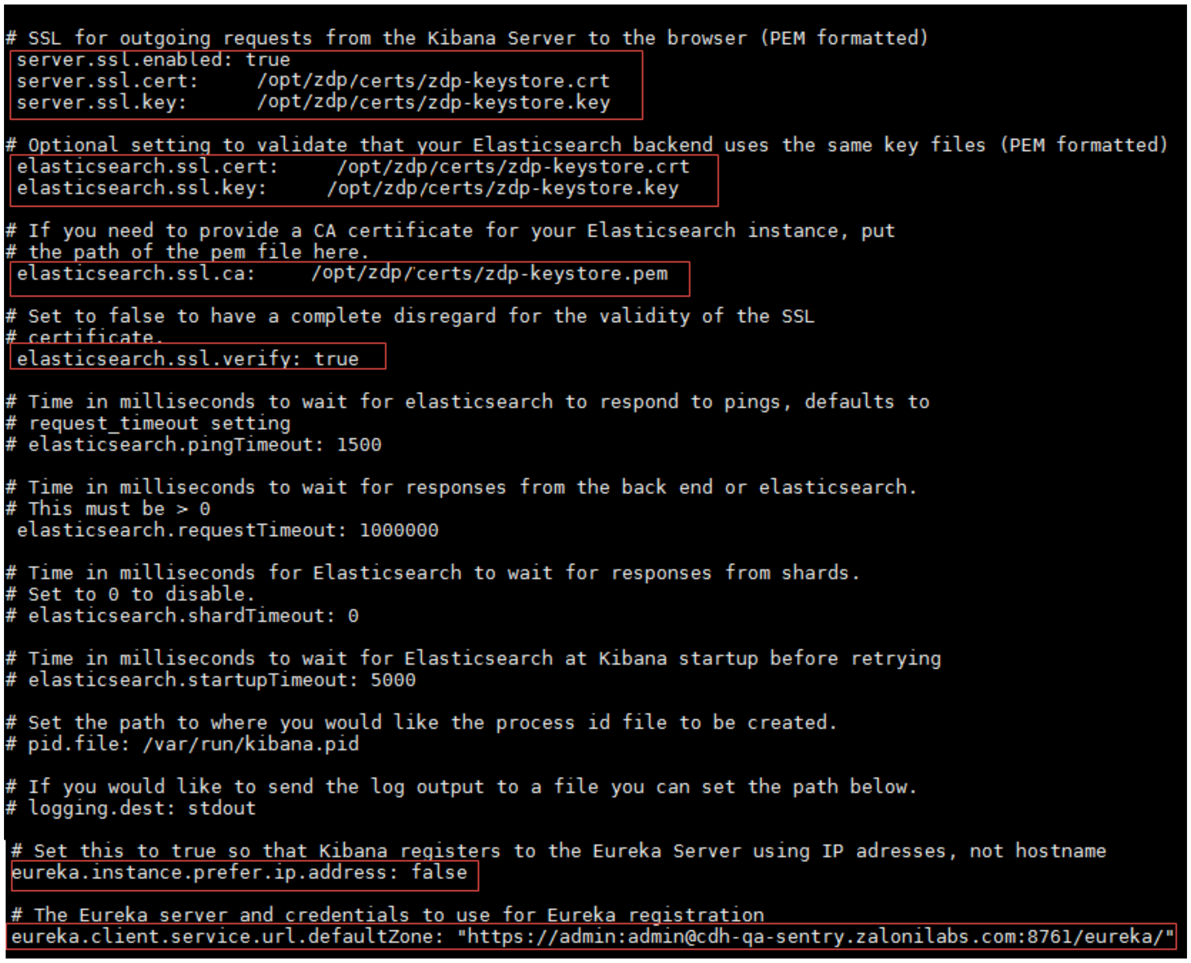
```
elasticsearch.ssl.key: Set the path to access the key files for securing the node where the zdp-gateway service is installed. A sample value is /opt/zdp/certs/zdp-beta-n4.zalonilabs.com.key.
```

```
elasticsearch.ssl.ca: Set the path to access the PEM formatted for securing the node where the zdp-gateway service is installed. A sample value is /opt/zdp/certs/zdp-keystore.pem.
```

```
elasticsearch.ssl.verify: Set this parameter to true to indicate that SSL is enabled for communication with Kibana.
```

```
eureka.instance.prefer.ip.address: Set this parameter to false to enable HTTPS.
```

```
eureka.client.service.url.defaultZone: Set the protocol to https, the username/password to access the zdp-registry service, and the hostname/port for the zdp-registry service.
```



```
# SSL for outgoing requests from the Kibana Server to the browser (PEM formatted)
 server.ssl.enabled: true
 server.ssl.cert:     /opt/zdp/certs/zdp-keystore.crt
 server.ssl.key:      /opt/zdp/certs/zdp-keystore.key

# Optional setting to validate that your Elasticsearch backend uses the same key files (PEM formatted)
 elasticsearch.ssl.cert:     /opt/zdp/certs/zdp-keystore.crt
 elasticsearch.ssl.key:     /opt/zdp/certs/zdp-keystore.key

# If you need to provide a CA certificate for your Elasticsearch instance, put
# the path of the pem file here.
 elasticsearch.ssl.ca:     /opt/zdp/certs/zdp-keystore.pem

# Set to false to have a complete disregard for the validity of the SSL
# certificate.
 elasticsearch.ssl.verify: true

# Time in milliseconds to wait for elasticsearch to respond to pings, defaults to
# request_timeout setting
# elasticsearch.pingTimeout: 1500

# Time in milliseconds to wait for responses from the back end or elasticsearch.
# This must be > 0
 elasticsearch.requestTimeout: 1000000

# Time in milliseconds for Elasticsearch to wait for responses from shards.
# Set to 0 to disable.
# elasticsearch.shardTimeout: 0

# Time in milliseconds to wait for Elasticsearch at Kibana startup before retrying
# elasticsearch.startupTimeout: 5000

# Set the path to where you would like the process id file to be created.
# pid.file: /var/run/kibana.pid

# If you would like to send the log output to a file you can set the path below.
# logging.dest: stdout

 # Set this to true so that Kibana registers to the Eureka Server using IP adresses, not hostname
 eureka.instance.prefer.ip.address: false

 # The Eureka server and credentials to use for Eureka registration
 eureka.client.service.url.defaultZone: "https://admin:admin@cdh-qa-sentry.zalonilabs.com:8761/eureka/"
```

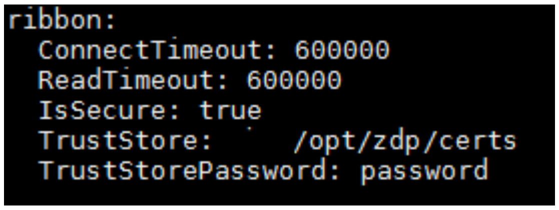
6. For the *zdp-kibana* service, add the following properties in the */etc/zdp-gateway/zdp-gateway.yml* file:

```
ribbon.IsSecure: Specify the parameter to true to indicate that the connection␣
↪with Kibana is secured (HTTPS-enabled).
```

```
ribbon.TrustStore: Specify the location where the truststore is placed. This is␣
↪the truststore of the node where the zdp-gateway service is running. This␣
↪truststore must contain a trust chain so that the zdp-gateway service trusts␣
↪the zdp-kibana service. Hence, the trust chain from the CA (that was used to␣
↪sign Kibana's TLS certificate) must be installed on the zdp-gateway node.
```

```
ribbon.TrustStorePassword: Specify the password for the truststore file.
```

For example,



```
ribbon:
  ConnectTimeout: 600000
  ReadTimeout: 600000
  IsSecure: true
  TrustStore:     /opt/zdp/certs
  TrustStorePassword: password
```

7. Add/update the following parameters in the */etc/zdp-registry/application.yml* file:

```
activemq.broker.url: For the ActiveMQ broker, specify the protocol as SSL and the␣
↪IP name for the node where the zdp-activemq service is installed.
```

```
activemq.broker.sslUrl: For the ActiveMQ broker, specify the protocol as SSL and␣
↪the IP name for the node where the zdp-activemq service is installed.
```

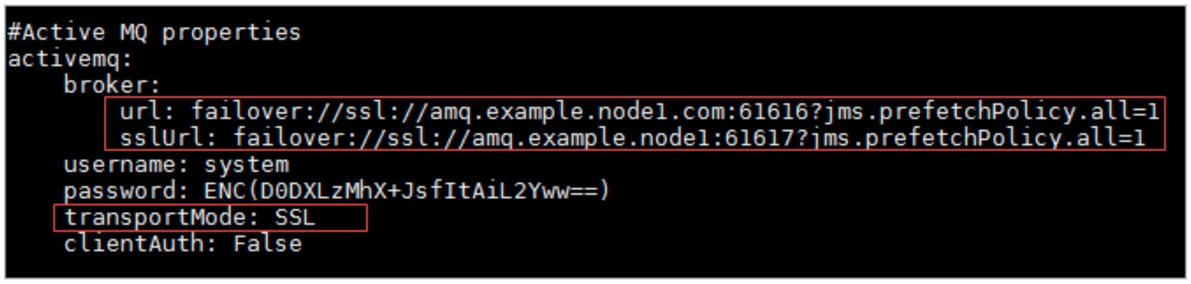
```
activemq.transportMode: Specify the transport mode as SSL.
```

```
elasticsearch.connection.tcp: Indicates the comma-separated hostname:port to␣
↪connect to multiple nodes where the zdp-es services are installed. For example,␣
↪node1.example.com:19300,node2.example.com:19300.
```

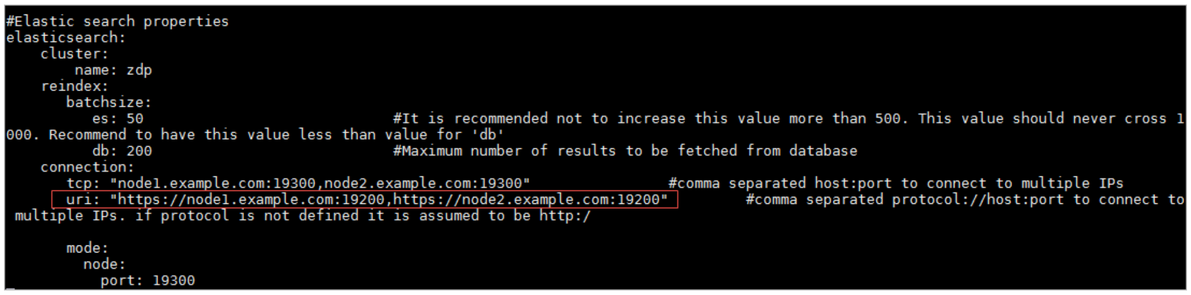
```
elasticsearch.connection.uri: Indicates the comma-separated protocol://
↪hostname:port to connect to multiple nodes where the zdp-es services are␣
↪installed. For HTTPS, the protocol must be set as https. If protocol is not␣
↪defined, the value is assumed to be http://. For example, http://node1.example.
↪com:19200,http://node2.example.com:19200.
```

```
logstash.host: Specify the hostname for the node where the zdp-logstash service␣
↪is installed.
```

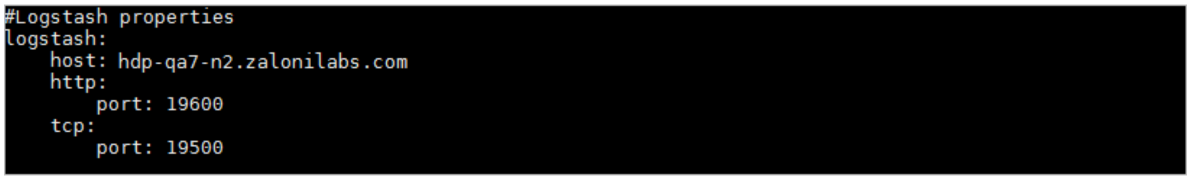
For example,



```
#Active MQ properties
activemq:
    broker:
        url: failover://ssl://amq.example.node1.com:61616?jms.prefetchPolicy.all=1
        sslUrl: failover://ssl://amq.example.node1:61617?jms.prefetchPolicy.all=1
    username: system
    password: ENC(D0DXLzMhX+JsfItAiL2Yww==)
    transportMode: SSL
    clientAuth: False
```



```
#Elastic search properties
elasticsearch:
    cluster:
        name: zdp
    reindex:
       batchsize:
          es: 50                           #It is recommended not to increase this value more than 500. This value should never cross 1
000. Recommend to have this value less than value for 'db'
          db: 200                          #Maximum number of results to be fetched from database
    connection:
       tcp: "node1.example.com:19300,node2.example.com:19300"                #comma separated host:port to connect to multiple IPs
       uri: "https://node1.example.com:19200,https://node2.example.com:19200"          #comma separated protocol://host:port to connect to
 multiple IPs. if protocol is not defined it is assumed to be http:/

       mode:
         node:
           port: 19300
```

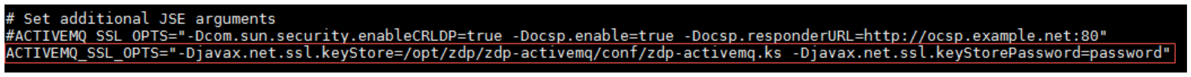


```
#Logstash properties
logstash:
    host: hdp-qa7-n2.zalonilabs.com
    http:
        port: 19600
    tcp:
        port: 19500
```

8. Place the keystore file (containing public-private key pair of the server running the ActiveMQ broker) in the */opt/zdp/zdp-activemq/conf* folder and update the name/keystore password in the */opt/zdp/zdp-activemq/bin/env* file:

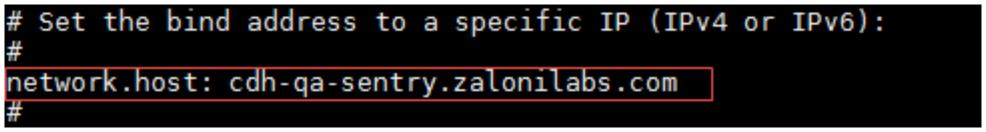
```
ACTIVEMQ_SSL_OPTS="-Djavax.net.ssl.keyStore=/opt/zdp/zdp-activemq/conf/zdp-
↪activemq.ks -Djavax.net.ssl.keyStorePassword=password"
```

For example,



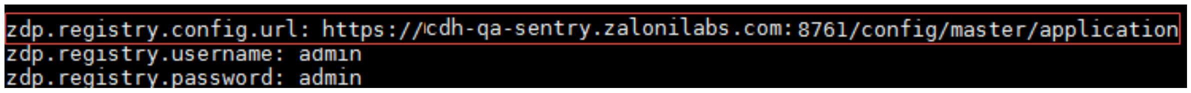
```
# Set additional JSE arguments
#ACTIVEMQ_SSL_OPTS="-Dcom.sun.security.enableCRLDP=true -Docsp.enable=true -Docsp.responderURL=http://ocsp.example.net:80"
ACTIVEMQ_SSL_OPTS="-Djavax.net.ssl.keyStore=/opt/zdp/zdp-activemq/conf/zdp-activemq.ks -Djavax.net.ssl.keyStorePassword=password"
```

9. For the *network.host* parameter in the */etc/zdp-es/elasticsearch.yml* file, specify the hostname for the node where the *zdp-es* service is installed.



```
# Set the bind address to a specific IP (IPv4 or IPv6):
#
network.host: cdh-qa-sentry.zalonilabs.com
#
```

10. In the */etc/zdp-logstash/zdp-logstash.yml* file, specify the protocol as htpps and the hostname/port for the *zdp-registry* service.



```
zdp.registry.config.url: https://cdh-qa-sentry.zalonilabs.com:8761/config/master/application
zdp.registry.username: admin
zdp.registry.password: admin
```

11. To verify that HTTPS is enabled in *https://<hostname>:<port>*, restart the *zdp-gateway* service.

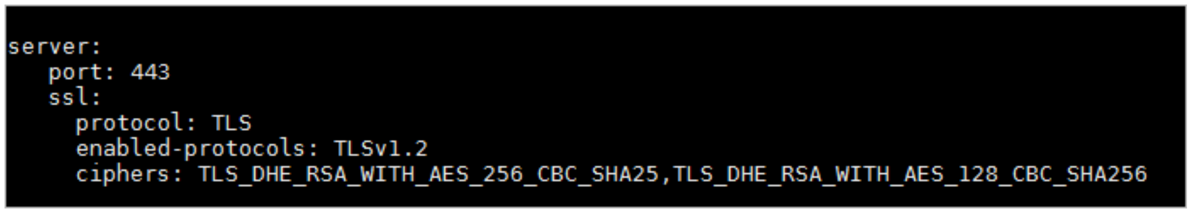
1.10.1 Support for TLSv1.2

By default, ZDP is supported with TLSv1.2 and a standard cipher. It is not recommended to change the default TLS version. However, if you want to modify the default values, perform the following steps:

- For the specific ZDP services (such as *zdp-gateway/zdp-registry/zdp-ingestion-warden/zdp-executor/zdp-mr/zdp-hive/zdp-spark/zdp-dme/zdp-lineage*), access the respective *zdp-{service_name}.yml* file and add the following properties:

Properties	Description
`server.ssl.protocol`	Indicate the SSL protocol. For example, TLS or SSL.
`server.ssl.enabled-protocols`	Indicate the version for the SSL protocol. For example, TLSv1.0.
`server.ssl.ciphers:`	Indicate the comma-separated list of custom ciphers. You must specify only the ciphers that are supported by the specified SSL protocol. For example, *TLS_DHE_RSA_WITH_AES_256_CBC_SHA25,TLS_DHE_RSA_WITH_AES_128_CBC_SHA256*.

For example,



```
server:
   port: 443
   ssl:
     protocol: TLS
     enabled-protocols: TLSv1.2
     ciphers: TLS_DHE_RSA_WITH_AES_256_CBC_SHA25,TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
```

- In the */etc/sysconfig/zdp-{service_name}* file (applicable services are *zdp-gateway/zdp-registry/zdp-ingestion-warden/zdp-executor/zdp-mr/zdp-hive/zdp-spark/zdp-dme/zdp-lineage*), add the following JVM property (if not already present):

```
JAVA_OPTS="$JAVA_OPTS -Djdk.tls.rejectClientInitiatedRenegotiation=true"
```

1.11 Configuring ZDP with Kerberos

Kerberos is a network authentication protocol that provides authentication for client/server applications by using the secret-key cryptography. By using Kerberos, a client can prove its' identity to a server (and vice versa) across an insecure network connection and encrypt all of their communications to assure privacy and data integrity.

You can use the Kerberos platform with ZDP. To do so, you must fulfill the following prerequisites:

- Configure Hadoop cluster for Kerberos.
- You work in an Active Directory Kerberos, Linux-based MIT Kerberos, KDC. For more information, refer to the Active Directory Kerberos and Linux-based MIT Kerberos respectively.
- Create the Kerberos principal for the ZDP service user and generate the keytab file. Copy the keytab file to the node where ZDP runs.
- Ensure that you have the Kerberos principal name for Hive's service account to connect through JDBC/Beeline. The Hive principal is used in cluster connection.
- Install kstart Linux package in the ZDP edge node.
- The service user (*zaloni*) must have Keytab location, Kerberos principal, KRB5 conf location.

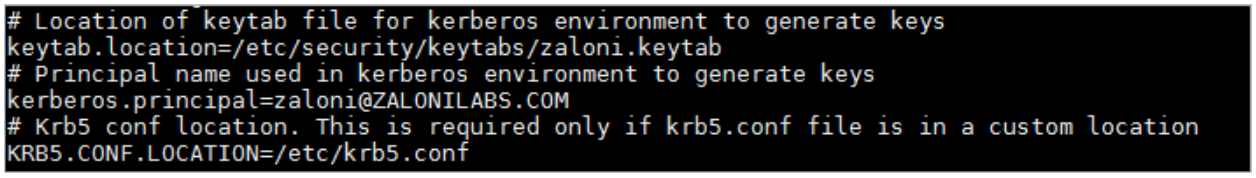
To use ZDP in the Kerberised mode, set the same values for the following properties in the */opt/zdp/zdp-ingestion-warden/config/agents/bdca/dca.properties*, */opt/zdp/zdp-hive/config/zdp-hive.yml*, */opt/zdp/zdp-mr/config/zdp-mr.yml*, */opt/zdp/zdp-spark/config/zdp-spark.yml*, and */opt/zdp/zdp-dme/config/zdp-dme.yml* files:

```
keytab.location: Indicate the location of the keytab file for the Kerberos␣
↪environment to generate keys.
```

```
kerberos.principal: Indicate the principal name used in the Kerberos environment to␣
↪generate keys.
```

```
KRB5.CONF.LOCATION: Indicate the Krb5 conf location. This is required only if krb5.
↪conf file is in a custom location.
```

For example,



```
# Location of keytab file for kerberos environment to generate keys
keytab.location=/etc/security/keytabs/zaloni.keytab
# Principal name used in kerberos environment to generate keys
kerberos.principal=zaloni@ZALONILABS.COM
# Krb5 conf location. This is required only if krb5.conf file is in a custom location
KRB5.CONF.LOCATION=/etc/krb5.conf
```

Note: To create a Kerberized cluster, refer to the configuration guide of your Hadoop distribution.

1.11.1 Accessing Hive CLI/Beeline on a Kerberized cluster

In a secured cluster, the the JDBC connection URL needs an extra argument to connect to the Hive server. Hive's service principal (not the end-user's principal) must be present in the URL. In the cluster configuration page, you must append the Hive principal to the Hive Server Jdbc URL, in the format: *jdbc:hive2://<hs2 server>:<port>/default;principal=<Hive_principal_name>@<realm_name>*.

Note: The *realm_name* used here is case sensitive and it must be always in capital letter. For example, cluster.zalonioinlabs.com@ZALONILABS.COM.

The end user (zaloni) also needs a valid ticket.

```
beeline> !connect jdbc:hive2://localhost:10000/default;principal=hive/cdh54-x.
↪zalonioinlabs.com@ZALONILABS.COM org.apache.hive.jdbc.HiveDriver Connecting to .....
↪ Enter password for ...: **** Connected to: Apache Hive (version 0.13.1-cdh5.3.6)
```

If the end user does not have a valid ticket:

```
beeline> !connect jdbc:hive2://10.11.12.201:10000/default;principal=hive/cdh54-guru1.
↪zalonilabs.com@ZALONILABS.COM org.apache.hive.jdbc.HiveDriver scan complete in 10ms␣
↪Connecting to ..... Enter password for ...: **** ERROR transport.TSaslTransport:␣
↪SASL negotiation failure javax.security.sasl.SaslException: GSS initiate failed␣
↪[Caused by GSSException: No valid credentials provided (Mechanism level: Failed to␣
↪find any Kerberos tgt)] at com.sun.security.sasl.gsskerb.GssKrb5Client.
↪evaluateChallenge(GssKrb5Client.java:212) .....
```

For Hive to be able to impersonate users (including ZDP) on a secured cluster, authorized users and hosts have to be whitelisted in the *core-site.xml* and *hive-site.xml* files. You will need the following:

```
hadoop.proxyuser.hive.groups = *
hadoop.proxyuser.hive.hosts = *
```

1.12 Configuring ZDP with Ranger

By using Apache Ranger, you can easily manage policies for access to files, folders, databases, tables, or column. These policies can be set for individual LDAP/AD users (or groups). The ZDP Cluster Security Manager synchronizes Role Based Access Control (RBAC) policies on cluster security managers, such as Apache Ranger. Apache Ranger protects resources, such as HDFS paths and Hive tables in a data lake.

If the Ranger feature is enabled, ZDP automatically creates access policies on Ranger for the entities and file patterns. The policies control which users and groups can access the various cluster resources and the access permissions are based on project role permissions that the users and groups have in the project.

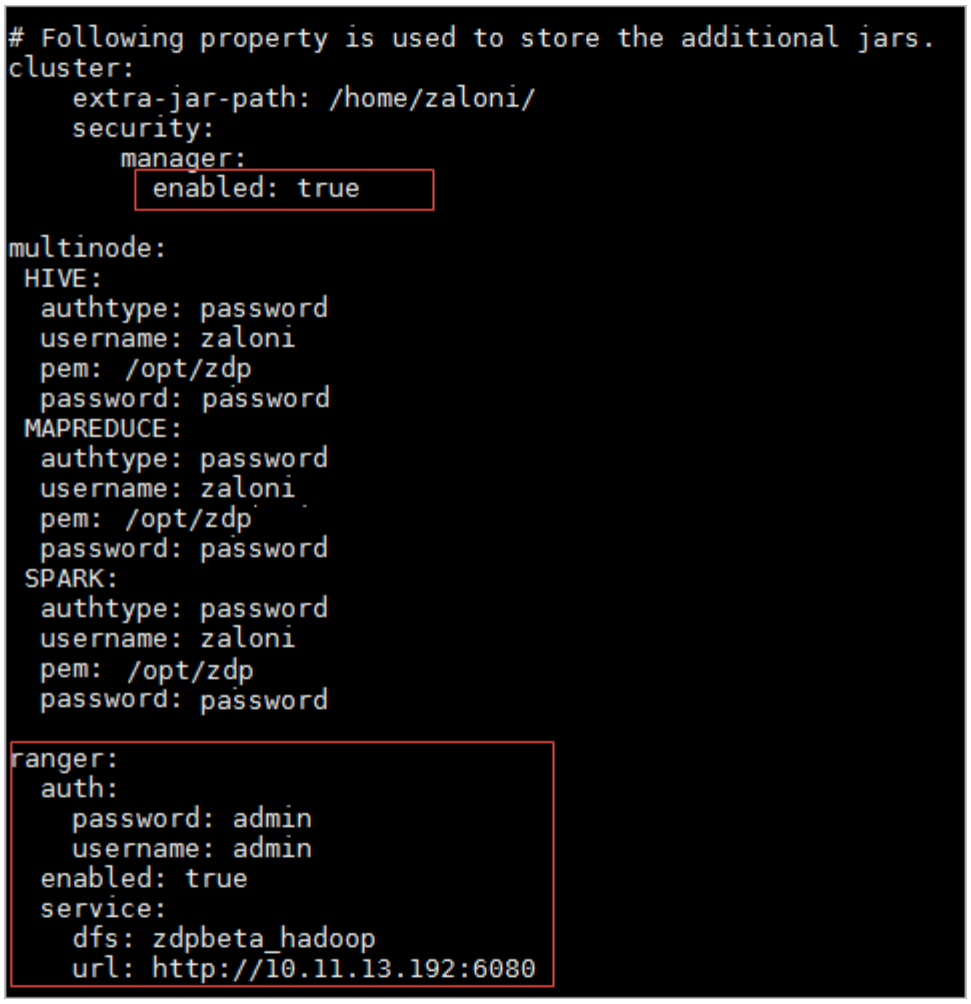
Note:

- To associate a user with a project in Ranger-configured ZDP, that user must exist in Ranger.
- If you use ZDP with Ranger, ingestion will fail if you do not have permission in the HDFS destination path and the system cannot create table in Hive if you do not have access to the Hive databases.
- The */user/zaloni/bedrock-lib* folder must be configured in Ranger with appropriate permissions for all users.
- For more information on using Bedrock with Ranger, refer to the bedrock_ranger section.

1.12.1 Enabling the Ranger feature in ZDP

To enable Ranger in ZDP, perform the following steps:

1. Add/update the following properties in the */etc/zdp-gateway/zdp-gateway.yml* file.



```
# Following property is used to store the additional jars.
cluster:
    extra-jar-path: /home/zaloni/
    security:
       manager:
         enabled: true

multinode:
 HIVE:
  authtype: password
  username: zaloni
  pem: /opt/zdp
  password: password
 MAPREDUCE:
  authtype: password
  username: zaloni
  pem: /opt/zdp
  password: password
 SPARK:
  authtype: password
  username: zaloni
  pem: /opt/zdp
  password: password

ranger:
  auth:
    password: admin
    username: admin
  enabled: true
  service:
    dfs: zdpbeta_hadoop
    url: http://10.11.13.192:6080
```

2. Ensure that the values for the *cluster.security.manager.enabled* and *ranger.enabled* parameters are set to *true*.
3. Restart the *zdp-gateway* service. This automatically creates policies for all the existing projects. For more information, refer to the *Application Start-up and Shut-down* section.
4. If the sync does not happen at boot time, call the ReST API to manually sync the policies. For more information, refer to the sync_project_ranger section.

Note: For any issues encountered with Ranger, you can refer to the ranger logs (*cluster-secmgr.log*) located under the */var/log/zdp-gateway/* directory.

1.12.2 Working with Ranger Hive plugin

ZDP does not create Hive policies in Ranger. If the cluster has Ranger plugin for Hive enabled, the Hive policies on Ranger does not reflect the project and role permissions. Thus, before using ZDP with Apache Ranger, perform the following steps for the Hive-plugin, based on your environment.

1.12.2.1 Disabling Hive plugin

Disable Hive plugin so that all data security is handled at the HDFS layer. By using such a setup, the effective security policies remain the same; as the security checks are performed when the query job reads/writes HDFS data.

Note: The responses on failed permission checks arrive when the job goes to the HDFS layer instead of arriving from the Hive query layer.

1.12.2.2 Creating a custom Hive policy in Ranger

This configuration can be set if you do not want Hive plugin to be disabled.

1. Create a custom Hive policy in Ranger.
2. Provide the necessary permissions to the LDAP/AD users (or groups) in ZDP across the required Hive databases (and tables) used in ZDP.

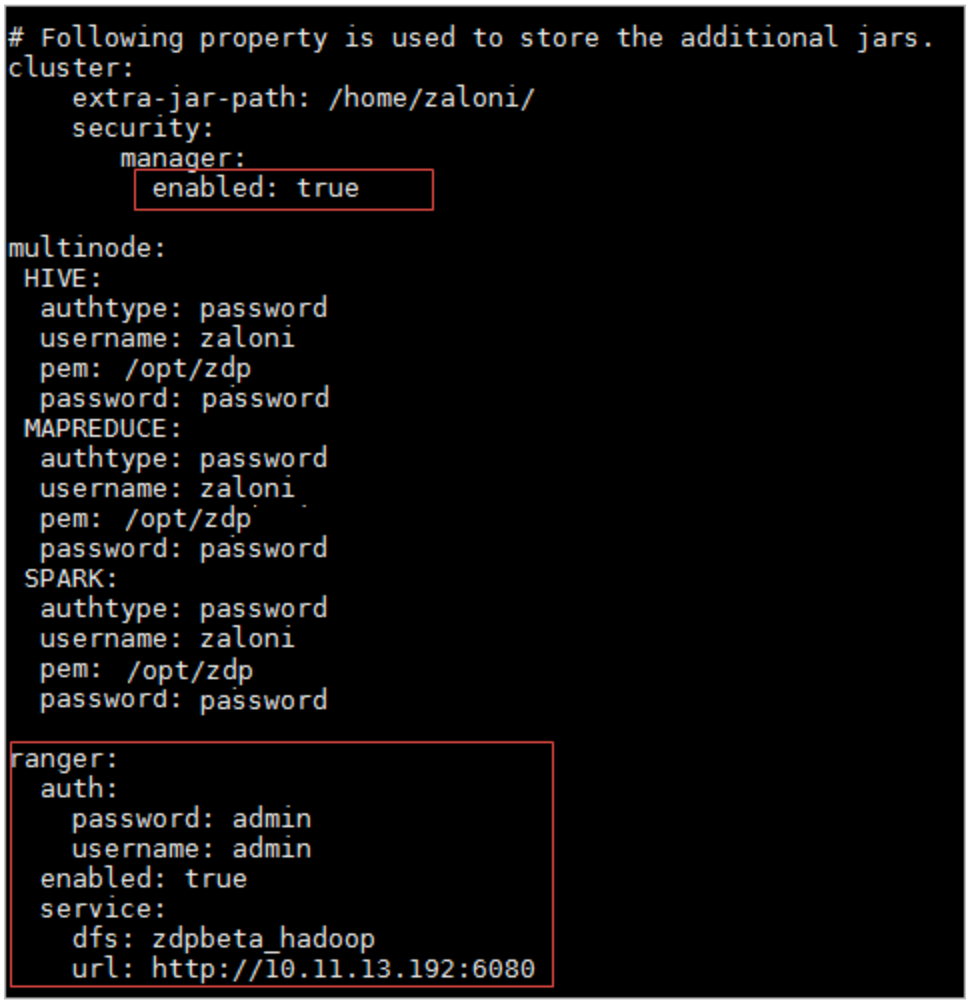
By doing so, you can ensure that:

- The effective security policies are enforced at the Hive layer.
- The responses on the failed permission checks arrive from Hive and before the job is submitted.

Note: ZDP prefers to handle Hive plugin dependency with Apache Ranger by using the step defined in the *Disabling Hive plugin* section.

1.12.3 Ranger Configuration

To configure Ranger and integrate with ZDP, the following properties must be defined in the */etc/zdp-gateway/zdp-gateway.yml* file:



```
# Following property is used to store the additional jars.
cluster:
    extra-jar-path: /home/zaloni/
    security:
       manager:
         enabled: true

multinode:
 HIVE:
  authtype: password
  username: zaloni
  pem: /opt/zdp
  password: password
 MAPREDUCE:
  authtype: password
  username: zaloni
  pem: /opt/zdp
  password: password
 SPARK:
  authtype: password
  username: zaloni
  pem: /opt/zdp
  password: password

ranger:
  auth:
    password: admin
    username: admin
  enabled: true
  service:
    dfs: zdpbeta_hadoop
    url: http://10.11.13.192:6080
```

```
cluster.security.manager.enabled: Indicates that the cluster security manager is␣
↪enabled or not. To use Ranger, the value for this parameter must be set to true.
```

```
ranger.auth.password: Indicates the password for the Ranger user. The recommended␣
↪good practice is to encrypt the value.
```

```
ranger.auth.username: Indicates the username to log into the Ranger UI. This user␣
↪must have the permissions to manage policies in Ranger.
```

```
ranger.enabled: Indicates if the Ranger integration must be triggered or not. To use␣
↪Ranger, the value for this parameter must be set to true. The default value is␣
↪false.
```

```
ranger.service.dfs: Indicates the Ranger service name under which ZDP must create␣
↪policies. You must provide the correct service name. If the value for this property␣
↪is incorrect, policy creation in ZDP fails. The error can be monitored in the zdp-
↪gateway logs.
```

```
ranger.service.url: Indicates the URL of the Ranger instance. If Ranger is installed␣
↪in the same node as the zdp-registry service, keep the value for the url parameter␣
↪as localhost.
```

1.12.4 Configuration Matrix

Based on the various values, set for the cluster.security.manager.enabled and ranger.enabled parameters, the system performs the following operations:

cluster.security.manager.enabled	ranger.enabled	Behavior
true	true	Policies are generated and pushed to Ranger.
true	false	Policies are generated and not pushed to Ranger.
false	true	Policies are neither generated nor pushed to Ranger.
false	false	Policies are neither generated nor pushed to Ranger

1.13 Overcoming Security Restrictions

The subsequent sub-sections provide the steps to overcome security restrictions to prevent workflow execution failures in ZDP. Such configurations may differ based on the environment set within the enterprise, such as the Hadoop distribution and various Hadoop utilities.

1.13.1 Configuring the HDFS-Sentry Plugin

For Sentry-enabled environment where the underlying distribution is CDH, the following steps must be performed before executing any workflow in ZDP.

1. Log into the Cloudera Manager as an administrator.
2. Follow the configuration steps as explained in the Cloudera documentation.

The above configurations are required to provide necessary permissions to the designated ZDP user (user that deployed ZDP), to prevent workflows execution failures.

Note: The following is one of the known limitation (related to permissions) within a Sentry enabled environment. All operations such as queries and commands (either via jdbc connection or by Metastore API) are performed by the Hive Metastore user (for example - hive).

1.13.2 For Parquet Entity Action in ZDP

Sentry (Cloudera's security manager) does not allow you to add jars. Hence, you must set the *HIVE_ACTION_SKIP_ADD_JARS* system variable to *true*. For more information, refer to the configure_system_variable section.

Additionally, you must add the *parquet-hive-bundle-1.4.0.jar* (present in the microservices) in the *hive.aux.jars.path* (library directory of Hive).

Also, you must grant all the permissions where this jar is located to the role that you use. Hence, you must give all the necessary permissions in the `file:///usr/lib/hive/lib/parquet-hive-bundle-1.4.0.jar` and `hdfs://cdhsentry.n1.zalonilabs.com:8020/user/bedrock/bedrock-lib/metadata/parquet-hive-bundle-1.4.0.jar` locations to the role and set the *AUX_CLASSPATH* parameter in the *hive-site.xml* file in the Cloudera manager to */usr/lib/hive/lib/parquet-hive-bundle-1.4.0.jar*.

1.13.3 For Token Masking Action in ZDP

To execute the **Token Masking Action** successfully on a Sentry-enabled environment, perform the following steps:

1. Access the ZDP 5.0.2 specific JARs (`bedrock-masking-udf-5.0.2-FINAL.jar` and `bedrock-tokenization-udf-5.0.2-FINAL.jar`), located (by default) in the */opt/zdp/zdp-hive/lib/token-mask* path of the node where the *zdp-hive* service is deployed.
2. Copy the ZDP-5.0.2 JARs to the node where the Hive server runs and place it in a specific path.
3. Add this path against the *hive.aux.jars.path* parameter in the *hive-site.xml* file.
4. Grant all the permissions where these JARs are located to the role that you use.

   For example,

   ```
   GRANT ALL ON URI 'file:///usr/lib/hive/lib/<my.jar>' TO ROLE <example_role>;
   ```

5. Give read-write-execute permission to the hive user in the output path.

For any further assistance related to ZDP, contact support.zaloni.com.

1.13.4 For Ranger Key Management Service

When impersonation is enabled and you are using Ranger KMS (Ranger Key Management Service) for HDFS encryption, ensure to add and configure the following properties in the *kms-site.xml* file:

```
hadoop.kms.proxyuser.bedrock.groups=*
hadoop.kms.proxyuser.hive.groups=*
hadoop.kms.proxyuser.bedrock.hosts=*
hadoop.kms.proxyuser.hive.hosts=*
hadoop.kms.proxyuser.bedrock.users=*
hadoop.kms.proxyuser.hive.users=*
```

1.14 Firewall

If all the rpms are installed on the same node, open the ports that are being used by the *zdp-gateway* service. For distributed systems, all the daemons need to communicate with each other over the listed ports. In such a case, set up perimeter security so that user can access only the *zdp-gateway* service. Ports used by different services are listed in the Nodes_and_ports section.

1.15 Application Start-up and Shut-down

1.15.1 Starting the ZDP services

To start the ZDP services, run the following command:

```
sudo systemctl start zdp-registry zdp-es zdp-gateway zdp-activemq zdp-logstash zdp-
↪lineage zdp-kibana zdp-ingestion-warden zdp-executor zdp-hive zdp-mr zdp-spark zdp-
↪dme
```

Alternatively, you can start the ZDP services by following the below order:

```
sudo service zdp-registry start
sudo service zdp-es start
sudo service zdp-gateway start
sudo service zdp-activemq start
sudo service zdp-logstash start
sudo service zdp-lineage start
sudo service zdp-kibana start
sudo service zdp-ingestion-warden start
sudo service zdp-executor start
sudo service zdp-hive start
sudo service zdp-mr start
sudo service zdp-spark start
sudo service zdp-dme start
```

If you want to run the services as a different user, such as `bedrock`, perform the following steps:

1. Stop all the ZDP services.
2. If the new user is *bedrock*, run the following command on all the nodes where the ZDP services are installed:

   ```
   chown -R bedrock:bedrock {/opt/zdp,/var/zdp*,/var/log/zdp*}
   ```

3. Edit the service files at */etc/systemd/system/zdp*.service* and change instances of the `User=zaloni` property, as follows:

   ```
   User=bedrock
   ```

4. Start all the services.

1.15.2 Stopping the ZDP services

To stop the ZDP services, use either of the following commands:

```
sudo systemctl stop <service_name>

sudo service <service_name> stop
```

For example,

```
sudo systemctl stop zdp-registry

sudo service zdp-registry stop
```

1.15.3 Restarting the ZDP services

To restart the ZDP services, use either of the following commands:

```
sudo systemctl restart <service_name>

sudo service <service_name> restart
```

For example,

```
sudo systemctl restart zdp-registry

sudo service zdp-registry restart
```

```
```

1.15.4 Seeking help for the ZDP services

While facing any issues with the services, you can seek help.

To view the available help options, use either of the following commands:

```
sudo systemctl help <service_name>

sudo service <service_name> help
```

For example,

```
sudo systemctl help zdp-registry

sudo service zdp-registry help
```

1.15.5 Uninstalling RPMs

To uninstall a service, run the following command:

```
sudo rpm -e <service name>
```

For example,

```
sudo rpm -e zdp-registry
```

Note: Uninstalling does not delete config files, logs, data files, and the MySQL database. These need to be deleted manually.

To upgrade the rpms, run the following command:

```
sudo rpm -U <service name>
```

For example,

```
sudo rpm -U zdp-registry
```

1.16 ZDP Integration

1.16.1 SMTP Integration

To integrate ZDP with an SMTP server, update the following property under **Administration > System Configuration Management**:

Properties	Description
`emailserver`	Represents the email server; the value should be specified as [host]:[port];host is the email server's hostname or IP; port is the port on which the email server is listening; for example, 10.0.0.1:25.

1.16.2 Hive Integration

By default, the **Hive Action** will work in the embedded mode utilizing the configuration of the Hive client.

To integrate ZDP with Hive Server or Hive Server 2, update the following parameters in the */etc/zdp-mr/zdp-mr.yml* and */etc/zdp-hive/zdp-hive.yml* files:

Properties	Description
`jdbc.hive.driver`	Represents the Hive JDBC driver. • For Hive Server, this will be: org.apache.hadoop.hive.jdbc.HiveDriver. • For Hive Server 2, this will be: org.apache.hive.jdbc.HiveDriver.
`jdbc.hive.username`	Indicates the Hive user in case Hive authentication is enabled.
`jdbc.hive.password`	Indicates the password for the Hive user to connect to the Hive server.

1.16.3 Spark Integration

ZDP integrates with Apache Spark^{TM} through the Spark-backed workflow actions and transformations, by using Spark Home directory. You can update the Spark home in the *spark_home* parameter in the */etc/sysconfig/zdp-mr*, */etc/sysconfig/zdp-spark*, */etc/sysconfig/zdp-dme* (optionally).

The Spark Home directory has all the Spark libraries and configuration files. ZDP uses the *bedrock-transformation-spark<SPARK_MAJOR_VERSION>.jar* file, located under the *opt/zdp/zdp-executor/transformation-spark* directory, to execute **Transformation Action**. Additionally, *applicationContext.xml* file and config has Spark configuration properties. All jars uploaded for User Defined Functions are stored in UDF_JARS located under the shared directory: */opt/zdp/zdp-executor/transformation-spark*.

1.17 Validating the Installation

To validate the ZDP installation, you must ensure that the primary services are running.

1.17.1 List Running ZDP Services

The instructions described in this section will guide you to check the status of the ZDP services.

- To check the status of ZDP services running on a node, execute the following command:

```
sudo service <service name> status
```

- Alternatively, log in to the ZDP UI, navigate to **Administration** > **Services Monitor**, and verify the status of the appropriate services.

1.18 Monitoring Logs

ZDP services logs are located under the directory: */var/log/<service name>*. To view the logs specific for a service, access the node where the service is installed and view the logs under */var/log/<service_name>*.

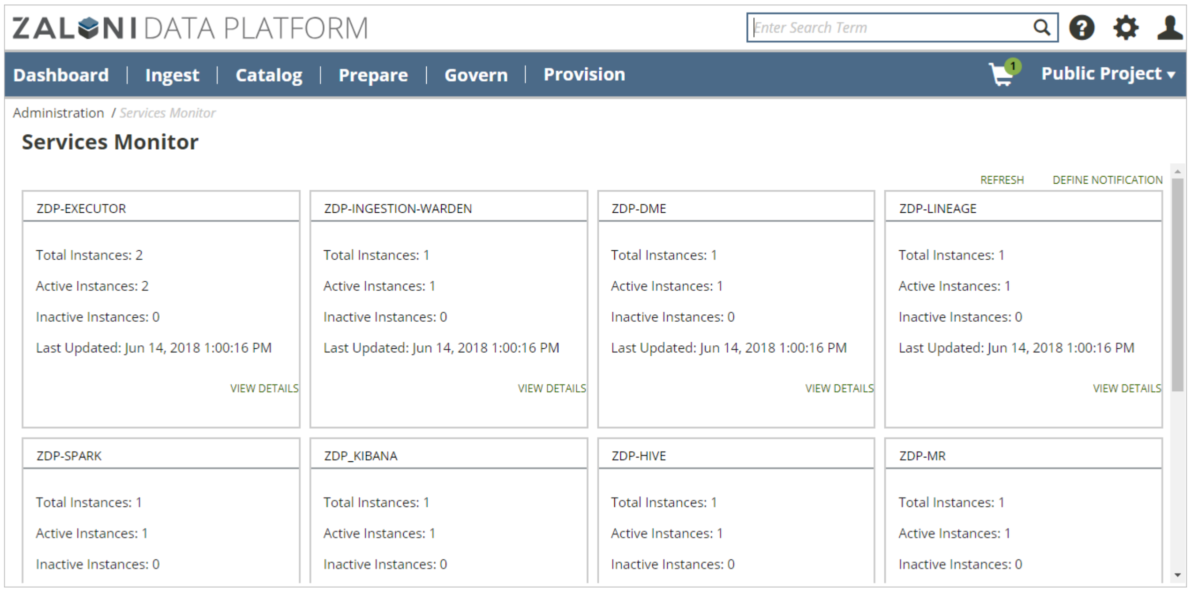
For example, *zdp-gateway* logs are available at:

```
/var/log/zdp-gateway/zdp-gateway.log
```

Note: For the *zdp-ingestion-warden* service, ZDP generates separate log files for individual BDCAs per user that launches the BDCA agent. For example, */var/log/zdp-ingestion-warden/{user that runs the BDCA}/bdca-{lzID}-{ID number}.log*. For example, */var/log/zdp-ingestion-warden/john/bdca-lzID-1.log*.

1.18.1 Monitoring Services

1. Log in to the ZDP application as admin in the browser: *http://<ip>:8080*. This is the IP for the node where the *zdp-gateway* service starts. For example, *http://10.11.13.14:8080/zdp/*.
2. To view the services, such as *zdp-executor*, *zdp-ingestion-warden*, navigate to **Administration > Services Monitor**.



Services Monitor

ZDP-EXECUTOR
Total Instances: 2
Active Instances: 2
Inactive Instances: 0
Last Updated: Jun 14, 2018 1:00:16 PM

ZDP-INGESTION-WARDEN
Total Instances: 1
Active Instances: 1
Inactive Instances: 0
Last Updated: Jun 14, 2018 1:00:16 PM

ZDP-DME
Total Instances: 1
Active Instances: 1
Inactive Instances: 0
Last Updated: Jun 14, 2018 1:00:16 PM

ZDP-LINEAGE
Total Instances: 1
Active Instances: 1
Inactive Instances: 0
Last Updated: Jun 14, 2018 1:00:16 PM

ZDP-SPARK
Total Instances: 1
Active Instances: 1
Inactive Instances: 0

ZDP_KIBANA
Total Instances: 1
Active Instances: 1
Inactive Instances: 0

ZDP-HIVE
Total Instances: 1
Active Instances: 1
Inactive Instances: 0

ZDP-MR
Total Instances: 1
Active Instances: 1
Inactive Instances: 0

3. For specific services, click **VIEW DETAILS**.

1.19 Migrating/Upgrading to ZDP

This section provides steps to execute migration from an older version of ZDP or from Bedrock, an older product from Zaloni.

1.19.1 Upgrading from Earlier Versions of ZDP

If you are upgrading from any earlier version of ZDP, perform the following steps:

1. Shut down all the ZDP services on all the nodes by using the following command:

```
sudo systemctl stop zdp-[servicename]
```

or

```
sudo service zdp-[servicename] stop
```

2. Optionally, take a backup of the following:
   - `/etc/zdp-*` configuration files
   - mysql db
   - data at `/var/zdp*`
3. Upgrade all or minimum required ZDP rpms by using either of the following commands:

   ```
   sudo rpm --upgrade zdp*.rpm
   ```

   or

   ```
   rpm -Uvh zdp-<service>-5.0.2_<version>.noarch.rpm
   ```

4. Compare and update (as applicable) the new configuration files with the older counterparts, manually. The installation program preserves the existing file and renames the new file with the `.rpmnew` extension.

   For example, for *etc/zdp-gateway/zdp-gateway.yml* file:

   The system saves the latest *etc/zdp-gateway/zdp-gateway.yml.* file as *etc/zdp-gateway/zdp-gateway.yml.rpmnew*.

   (a) List the rpmnew files by using the following command:

   ```
   ls /etc/zdp-gateway/*.rpmnew
   ```

   (b) Find the difference between the older and the latest files by using the following command:

   ```
   diff -urN /etc/zdp-gateway/zdp-gateway.yml /etc/zdp-gateway/zdp-gateway.yml.
   ↪rpmnew
   ```

   (c) Add the difference to the existing file: */etc/zdp-gateway/zdp-gateway.yml*.

5. If your driver JARs are placed in the */opt/zdp/zdp-mr/lib/dbimport/sqoop/drivers/<database_type>_<database_version>* location (for executing **DB Import Action**), ensure that you place all your driver JARs in the location: */opt/zdp/zdp-drivers/<database_type>_<database_version>*. Ensure that the */opt/zdp/zdp-drivers* location has read-write-execute permission.
6. Start all the ZDP services. To start/stop services, refer to the *Application Start-up and Shut-down* section and verify the application startup and shutdown.
7. Call the reindex_artifacts API and ensure that you set the value of the *indexAliases* parameter to *globalsearch*.
8. (If you are using the impersonation feature), ensure that you manually enable the following permissions for any project role (custom or pre-defined):
   - *Allow Impersonation* permission in order to let that user be impersonated by another user (that is mapped to a role with the *Impersonate Other Users* permission). The *Allow Impersonation* permission can be selected only for a project level role and not a global role.
   - *Impersonate Other Users* in order to impersonate users (that are mapped to a role with the *Allow Impersonation* permission). The *Impersonate Other Users* permission can be selected only for a project level role and not a global role.
   - For more information, refer to the adding_role section.
9. (In a Sentry-enabled environment) Ensure that you copy the ZDP-5.0.2 specific JARs (for example, bedrock-hive-hooks-5.0.2-FINAL.jar, bedrock-masking-udf-5.0.2-FINAL.jar, bedrock-tokenization-udf-5.0.2-FINAL.jar, parquet-hive-bundle-1.4.0.jar) to the node where the Hive server runs, place it in a specific path,

and add this path against the hive.aux.jars.path parameter in the hive-site.xml file. Additionally, remove the JARs corresponding to earlier ZDP versions. For more information, refer to the *Overcoming Security Restrictions* section.

1.19.2 Migrating from Bedrock

Important: This is applicable only for Bedrock 4.4.1/4.5.0 to ZDP 5.0.2 migration.

1.19.2.1 Bedrock to ZDP Changes

Apart from the new user-facing features, ZDP offers the following advantages over Bedrock, specifically geared towards installation, configuration, and maintenance:

Advantages	Description
Easy Installation	The Bedrock installer is replaced by 13 RPMs. This helps you to install individual services on different nodes or on the same node (depending on your requirements).
Aggregated Properties	In ZDP, all the properties are aggregated into a single YAML file, whereas the configuration properties were located in multiple files in Bedrock.
Centralized Properties	The common configuration that needs to be shared/consumed by multiple services are defined within a centralized YAML file, defined under the *zdp-registry* service.

Following are the Bedrock components and their corresponding ZDP 5.0.2 service names:

Bedrock Component Name	Bedrock Installation Location	ZDP Service Name	ZDP Installation Location
bedrock-app	<BR_HOME>/bedrock-app	zdp-gateway	/opt/zdp/zdp-gateway
bedrock-workflow-executor	<BR_HOME>/bedrock-workflow-executor	zdp-executor	/opt/zdp/zdp-executor
apache-activemq-5.13.0	<BR_HOME>/apache-activemq-5.13.0	zdp-activemq	/opt/zdp/zdp-activemq
elasticsearch-2.4.1	<BR_HOME>/elasticsearch-2.4.1	zdp-es	/opt/zdp/zdp-es
logstash-5.4.0	<BR_HOME>/logstash-5.4.0	zdp-logstash	/opt/zdp/zdp-logstash
rl-meta-1.0-release.jar	<BR_HOME>/dmxservice	zdp-dme	/opt/zdp/zdp-dme
spark-executor-microservices.jar	<cluster_deploy_location>/microservice/spark-executor	zdp-spark	/opt/zdp/zdp-spark
mapreduce-executor-microservices.jar	<cluster_deploy_location>/microservice/mapreduce-executor	zdp-mr	/opt/zdp/zdp-mr
hcatalog-microservices.jar	<cluster_deploy_location>/microservice/hive-connector	zdp-hive	/opt/zdp/zdp-hive
data-lineage-trackers	<BR_HOME>/<port>/jars	zdp-lineage	/opt/zdp/zdp-lineage

The following services are introduced in ZDP 5.0.x:

Services	Description
zdp-registry.rpm	This rpm is responsible for centralized configuration for the various ZDP processes, such as Elasticsearch, logstash, database, etc.
zdp-ingestion-warden.rpm	This rpm is responsible for centralized BDCA configuration, wherein BDCA(s) processes/threads are spawned on the landing zone server where this rpm is installed.
zdp-kibana.rpm	This rpm is responsible for displaying the details stored in *zdp-es* as dashboards. The *zdp-kibana* service is tightly coupled with the *zdp-es* and *zdp-logstash* services. For more information, refer to ELK stack.

1.19.2.2 Using the ZDP Upgrade Package

To upgrade from Bedrock 4.4.1/4.5.0 to ZDP 5.0.2, you must leverage the ZDP upgrade package: *zdp-migrator*. This package performs the automated migration activity and manual activities:

Services	Description
Database migration	The metadata (entities, workflows, transformation) created by using Bedrock 4.4.1/4.5.0, needs to be available after migration.
Configuration properties migration	The properties used to configure Bedrock 4.4.1/4.5.0, need to be migrated to ZDP 5.0.2. During this migration, properties are read and aggregated from the multiple *.properties* files of Bedrock 4.4.1/4.5.0 and written into the ZDP yml files.
Elasticsearch migration	Existing Elasticsearch data needs to be migrated to the upgraded *zdp-es* service.
Service user definition	The ZDP service needs to start with the same user that was used to run the Bedrock components.

1.19.2.3 Planning the ZDP migration

Before starting the migration, perform the followings tasks:

- Take a full backup in the form of a DB dump of the existing database used in the Bedrock 4.5.0/4.4.1.
- If you have an older version of Bedrock, upgrade to version 4.4.1 or 4.5.0. For more information, refer to the Installation guides for the respective versions (available at Zaloni Support Portal).
- Ensure that all the new RPMs for ZDP-5.0.2 are installed in the same manner architecturally, as their corresponding services were installed in Bedrock-4.4.1/4.5.0. For example, (in Bedrock 4.4.1) if you have installed workflow executor and BDCA in a single remote node, you must install the *zdp-executor* and *zdp-ingestion-warden* services in a single remote node in ZDP-5.0.2.
- Ensure that ZDP uses different port for Elasticsearch. Ports used in the previous version of ZDP are not automatically migrated. For default ports of ZDP, refer to the Nodes_and_ports section.
- Ensure that the remote *zdp-services* are migrated separately.
- Ensure that zdp-registry and bedrock-app are installed in the same node. Additionally, install nodeJS in this node.
- For remotely installed zdp-es service, ensure that you install the zdp-es RPM in the same node as the zdp-registry service. Once the migration process is complete, you can uninstall this service.
- The ZDP migrator script tries to locate the existing installed Bedrock components within the Bedrock Home location. The script lists down the folders available inside the Bedrock Home location. Hence, if any of the Bedrock components (from the below table) are not in the running state, you must either delete the respective

folder or transfer the folder to another location. By doing so, you can ensure that the migrator script does not detect that component for migration. For example, in Bedrock 4.4.1 multinode setup, there exist a bedrock-workflow-executor folder inside the Bedrock Home location. However, workflow executor is not in the running state in that node. In that case, (before running the migrator script) you must remove/transfer the bedrock-workflow-executor folder. The following table lists the Bedrock components along with the respective folders:

Bedrock Component	Folder
BDCA	bdca
workflow-executor	bedrock-workflow-executor
Lineage	data-lineage-trackers
ActiveMQ	apache-activemq-<version>
Kibana	kibana
Logstash	Logstash-<version>
Elasticsearch	elasticsearch-<version>
bedrock-app	bedrock-app

1.19.2.4 Running the ZDP Migration Tool

To migrate from Bedrock 4.4.1/4.5.0 to ZDP 5.0.2, execute the script which is packaged in the form of a tarball: *zdp-migrator-5.0.2-*.tar*.

To extract the tarball, perform the following steps:

1. Download the *zdp-migrator-5.0.2-*.tar* file from the Zaloni Support Portal.
2. Copy the downloaded tar file to any desired location on a node.
3. To extract the tarball, execute the below command:

```
tar -xvf [tar name]

For example, tar -xvf  zdp-migrator-5.0.2-*.tar
```

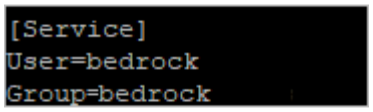
4. The following files and directories are present within the zdp-migrator-5.0.2-* folder:

Services	Description
bin	Includes all the binaries and scripts required for the migration process. This folder includes the following sub-directories: • *es_migration*: Contains scripts related to Elasticsearch data migration. It includes: *zdp-artifacts-syncup.sh* and *zdp-migrate-es-data.sh*. • *zdp-migrator.sh*: Contains scripts related to database, configuration properties migration. • *dme_data_migration*: Contains scripts related to DME database migration. It includes: *dme_tables_data_migration.sql*.
config	Includes the various configuration files for the migrator. • *log4j2.xml*: Contains Log4j configuration file for logs configuration. • *migrate-input.yml*: Contains the input YAML file required for migration.
lib	Includes the various library files for the migrator. • *zdp-migrator-5.0.2-*.jar*: Contains the executable jar required to execute migration tasks.
logs	Includes the log files generated during migration: • es_dump_migrator.log (Elasticsearch log) • migrator.log (database and properties migration log).

5. Change the owner of ZDP files to match the owner of bedrock-app, by running the following command:

```
sudo chown -R bedrock:bedrock {/opt/zdp,/var/zdp*,/var/log/zdp*}
```

6. Ensure that the service *user* and *group* configured in */etc/systemd/system/zdp-{service-name}.service* files, matches the owner of Bedrock. For example, bedrock-app was executed as the *bedrock* user.



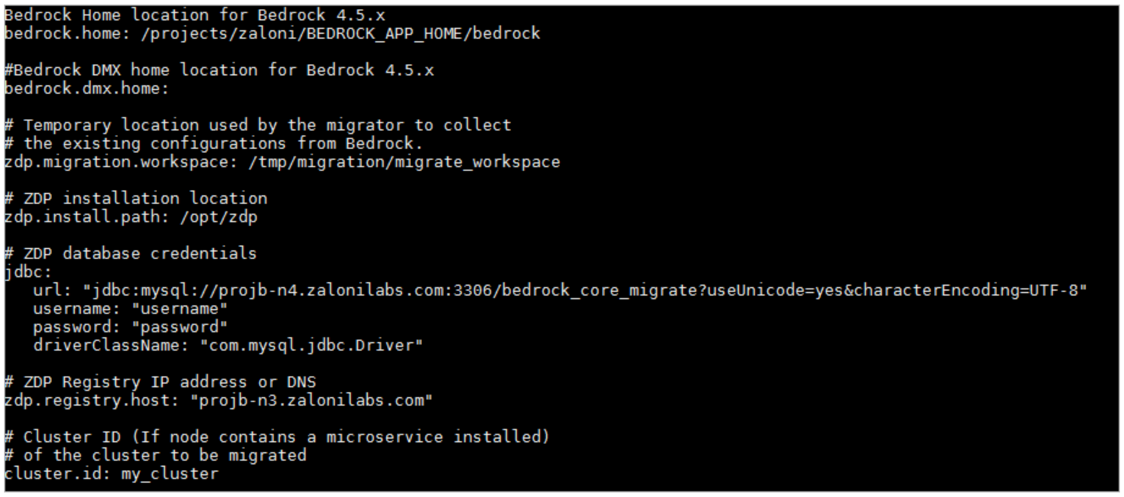
```
[Service]
User=bedrock
Group=bedrock
```

7. Configure the properties available in the *../config/migrate-input.yml* file:

Bedrock Component Name	Bedrock Installation Location
`bedrock.home`	Includes the home directory for the existing Bedrock 4.4.1/4.5.0 installation. For example, */projects/zaloni/BEDROCK_APP_HOME/bedrock*.
`bedrock.dmx.home`	(Optional) Includes the location where Bedrock 4.5.0 DME is installed. For example, */home/bedrock/BR45/dmx/dmx_1.0*.
`zdp.migration.workspace`	Includes the temporary workspace where files generated during migration, are written. For example, */tmp/migration/migrate_workspace*. The following directories are found here: • *br_config_aggregated*: Includes the configuration information for all the Bedrock resources that is used by the migrator to update the ZDP yml/config files. • *installation_notes*: Includes installation note that is generated by the migrator. It contains information about remote BDCA, workflow executor, Flume, and remote cluster. This information is displayed post successful migration to ZDP 5.0.2 to inform the user about the nodes where migrator needs to be run.
`jdbc.url`	Includes the JDBC URL for the Bedrock 4.4.1/4.5.0 database instance. This URL must be accessible from remote location. For example, *jdbc:mysql://projb-n4.zalonilabs.com:3306/bedrock_core_migrate?useUnicode=yes&characterEncoding=UTF-8*.
`jdbc.username`	Includes the JDBC user name for the Bedrock 4.4.1/4.5.0 database instance. This user must have remote access. For example, *username*.
`jdbc.password`	Includes password for the JDBC user. For example, *password*.
`zdp.registry.host`	Includes the hostname/IP of the node where the *zdp-registry* service is installed. For example, *projb-n3.zalonilabs.com*.
`cluster.id`	Includes the cluster id where the cluster microservices are installed. If the node from where the migration is performed, does not contain any cluster micro-services (such as *zdp-mr/zdp-hive/zdp-spark/zdp-dme*), you are not required to define this property. For example, *my_cluster*.

```
jdbc.driverClassName: Includes the driver class for the JDBC connection. For␣
↪example, com.mysql.jdbc.Driver.
```

For example,



```
Bedrock Home location for Bedrock 4.5.x
bedrock.home: /projects/zaloni/BEDROCK_APP_HOME/bedrock

#Bedrock DMX home location for Bedrock 4.5.x
bedrock.dmx.home:

# Temporary location used by the migrator to collect
# the existing configurations from Bedrock.
zdp.migration.workspace: /tmp/migration/migrate_workspace

# ZDP installation location
zdp.install.path: /opt/zdp

# ZDP database credentials
jdbc:
   url: "jdbc:mysql://projb-n4.zalonilabs.com:3306/bedrock_core_migrate?useUnicode=yes&characterEncoding=UTF-8"
   username: "username"
   password: "password"
   driverClassName: "com.mysql.jdbc.Driver"

# ZDP Registry IP address or DNS
zdp.registry.host: "projb-n3.zalonilabs.com"

# Cluster ID (If node contains a microservice installed)
# of the cluster to be migrated
cluster.id: my_cluster
```

1.19.2.5 Executing Migration

By executing the script: */zdp-migrator-5.0.2-*/bin/zdp-migrator.sh*, you can migrate database, configuration properties, and Elasticsearch from Bedrock 4.4.1/4.5.0 to ZDP 5.0.2:

To do so, perform the following steps:

1. Stop all Bedrock services except Bedrock-elasticsearch.
2. Log in as a sudo user where migration needs to be executed.
3. Execute the script: *zdp-migrator.sh*, as follows:

   ```
   bedrock> sudo sh zdp-migrator.sh CORE
   ```

   Important: Ensure that you run the above command on all the nodes where the ZDP services are installed. However, you must not run this command in the node where the zdp-es service is installed (in case zdp-es is installed remotely in an isolated manner).

   Note: If not specified, ZDP assumes the parameter: *CORE* (to migrate database and configuration properties) as the default value. The other possible parameter can be *ES* (can be set to migrate Elasticsearch details).

4. For remotely installed Elasticsearch, manually update the properties available in the /opt/zdp/zdp-es/elasticsearch.yml file.
5. To avoid getting duplicate entries for the zdp-executor service in the MySQl database (or under **Administration > Services Monitor**), ensure that you perform either of the following steps:
   - Update the value for the property: server.port (in the /etc/zdp-executor/zdp-executor.yml file) to match with the port where the Bedrock specific workflow executor was installed.
   - Access the new MySQL database and remove the duplicate entries for workflow executor (manually). You must delete the Bedrock specific workflow-executor details.
6. Start the *zdp-registry*, *zdp-es*, and *zdp-gateway* services. To start/stop services, refer to the *Application Start-up and Shut-down* section.

Important: For remotely installed zdp-es service, ensure that you install the zdp-es RPM in the same node as the zdp-registry service. Once the migration process is complete, you can uninstall this service.

7. To migrate Elasticsearch details, execute the script: zdp-migrator.sh, as follows:

```
bedrock> sudo sh zdp-migrator.sh ES
```

Important: Ensure that you run the script from the node where the zdp-registry/bedrock-app are installed. Additionally, nodeJS must be installed in this node.

8. To synchronize the Elasticsearch data after migration, execute the *zdp-artifacts-syncup.sh* under the */zdp-migrator-5.0.2-*/bin/es_migration* folder, as follows:

```
sudo sh zdp-artifacts-syncup.sh <Specify the protocol here. Sample values are␣
↪http or https. If you do not specify any value, the default value assumed is␣
↪http> <Specify the hostname for the zdp-gateway service. If you do not specify␣
↪any value, the system assumes this value as localhost.> <Specify the port for␣
↪the zdp-gateway service. If you do not specify any value, the system assumes␣
↪this value as 8080.>
```

```
For example,

sudo sh zdp-artifacts-syncup.sh http projb-n4.zalonilabs.com 8089
```

1.19.2.6 Migrating DME Data

To perform the DME data migration, execute the following steps:

1. Execute the steps mentioned in the *Executing Migration* section.
2. Start the *zdp-gateway* service. To start/stop services, refer to the *Application Start-up and Shut-down* section.
3. Execute the script: */zdp-migrator-5.0.2-*/bin/dme_data_migration/dme_tables_data_migration.sql* on the ZDP database.
4. Start the *zdp-dme* service.

1.19.2.7 Migrating to ZDP with Clustered MySQL

To migrate from Bedrock 4.4.1/4.5.0 to ZDP 5.0.2 with clustered MySQL setup, migrate to a non-clustered MySQL setup and perform the following steps:

1. To push the clustered MySQL changes on a ZDP 5.0.2 setup, start the *zdp-gateway* service after the migration.
2. Take the database dump of Bedrock 4.4.1/4.5.0 after the *zdp-gateway* service starts.
3. Import the database dump on a clustered MySQL setup.
4. Modify the */etc/zdp-registry/application.yml* to point to the clustered MySQL instance.
5. Restart the *zdp-registry* and *zdp-gateway* services.

1.19.2.8 Validating the Migration (Post migration checks)

After migration is complete, verify the migration log available in the location mentioned for the parameter: *zdp.migration.workspace*. Additionally, ensure that you observe the following points:

- Logs generated by *bedrock-app*, workflow, and any other Bedrock modules is intact.
- Kibana dashboard information is intact.
- Ingestion history and workflow ingestion history is intact.
- Clone, read, update, and delete operations on existing entity is successful.
- Schedule, rules, rule sets, data quality reports are available.
- Database is not corrupt.
- Data models are retained.
- After DME data is migrated, DME sample data models are retained.
- Once migration is complete, set the cluster (the default cluster in Bedrock 4.5.0/4.4.1) as the default cluster in ZDP 5.0.2, manually.
- If you want to execute **Token Masking Action** in Mapreduce engine, set the value for the parameter: **TOKEN-MASK_EXECUTION_ENGINE** to *MR* (under **Administration > System Configuration Management**).
- Manually populate the values for the parameters that are newly introduced in ZDP 5.0.2 and were not present in Bedrock 4.5.0/4.4.1.
- Ensure that the Elasticsearch indices are properly migrated. If you want to reindex data in ZDP 5.0.2, perform the steps in the *Recreating Indices in Elasticsearch* section.
- Update the Kerberos configurations (done in cluster configuration in Bedrock 4.4.1/4.5.0) in ZDP 5.0.2. For more information, refer to the *Configuring ZDP with Kerberos* section.
- In the */etc/zdp-spark/zdp-spark.yml* file, ensure that you manually update the following properties:
  - `sparkify.spark.home`
  - `sparkify.spark.version`
  - `sparkify.spark.jars`
  - `spark.spark1.home`
  - `spark.spark1.version`
  - `spark.spark1.extra.env`
  - `spark.spark1.extra.opts`
  - `spark.spark1.kafka.version`
  - `spark.spark2.home`
  - `spark.spark2.version`
  - `spark.spark2.extra.env`
  - `spark.spark2.extra.opts`
  - `spark.spark2.kafka.version`
- For Bedrock 4.4.1/4.5.0 to ZDP 5.0.2 migration, ensure that you manually enable the following permissions for any project role (custom or pre-defined):

- *Allow Impersonation* permission in order to let that user be impersonated by another user (that is mapped to a role with the *Impersonate Other Users* permission). The *Allow Impersonation* permission can be selected only for a project level role and not a global role.
- *Impersonate Other Users* in order to impersonate users (that are mapped to a role with the *Allow Impersonation* permission). The *Impersonate Other Users* permission can be selected only for a project level role and not a global role.
- For more information, refer to the adding_role section.

• (In a Sentry-enabled environment) Ensure that you copy the ZDP-5.0.2 specific JARs (for example, bedrock-hive-hooks-5.0.2-FINAL.jar, bedrock-masking-udf-5.0.2-FINAL.jar, bedrock-tokenization-udf-5.0.2-FINAL.jar, parquet-hive-bundle-1.4.0.jar) to the node where the Hive server runs, place it in a specific path, and add this path against the hive.aux.jars.path parameter in the hive-site.xml file. Additionally, remove the JARs corresponding to earlier Bedrock versions. For more information, refer to the *Overcoming Security Restrictions* section.

1.20 Recreating Indices in Elasticsearch

While migrating data from an earlier version of ZDP, indices may need to be migrated. In such cases, the administrator should reindex ZDP indices. This feature may also be useful if the *zdp-es* service is ever unavailable and you want to sync indices with data present in the database.

To start a reindex operation, invoke the reindex_artifacts API.

1.21 References

1.21.1 Installing MySQL

ZDP uses MySQL as its data store. Install MySQL by following the MySQL Reference Manual for your Linux distribution (http://dev.mysql.com/doc/).

Once installed, set MySQL to start automatically depending on what the service is called.

```
root>chkconfigmysqld on
```

Or

```
root>chkconfigmysql on
```

1.21.2 MapR Setting for Spark-submit and Spark-Shell Actions

For MapR environment, ensure that the following property is set (as key-value pair) in *spark-defaults.conf* file for successful execution of Spark-related workflow actions, specifically. The *spark-defaults.conf* file is located under *$SPARK_HOME/conf* directory.

Key	Value
spark.sql.hive.metastore.sharedPrefixes	com.mysql.jdbc,org.postgresql,com.microsoft.sqlserver,oracle.jdbc, com.mapr.fs.shim.LibraryLoader,com.mapr.security.JNISecurity, com.mapr.fs.jniInstalling MySQL

1.21.3 Sample Default Paths of Hive/Spark Libraries in HDP

In the HDP distribution, follow the table to learn about the default path for Hive and Spark libraries:

Parameter	Default Path in HDP
`hcatalog_lib`	usr/hdp/hive-hcatalog/share/hcatalog/<sample>.jar
`web_hcatalog_lib`	/usr/hdp/hive-hcatalog/share/webhcat/java-client/<sample>.jar
`hive_conf_dir`	/etc/hive/conf
`krb5ConfLocation`	/etc/krb5.conf
`hadoop_conf`	/usr/hdp/hadoop/conf
`spark_home`	/usr/lib/SPARK
`hive_lib`	/usr/lib/hive/lib/<sample>.jar

For more information about the default paths for other distribution, refer to the respective product documentation.

CHAPTER

TWO

TROUBLESHOOTING GUIDE

The subsequent sections provide you the steps to troubleshoot certain environment-specific issues related to ZDP.

To troubleshoot such issues, the user is expected to be an enterprise level administrator (having necessary permissions). However, some issues can also be addressed by ZDP level administrators.

2.1 Executor Down

The *zdp-executor* can be down due to various environment issues. One reason can be insufficient executor memory. In such a case, check machine memory and resolve. Workflows do not get executed if executor(s) are down.

To troubleshoot, perform the following steps:

1. To check the status of the *zdp-executor* service, run either of the following commands:

   ```
   sudo systemctl status zdp-executor
   ```

   ```
   sudo service zdp-executor status
   ```

2. If the *zdp-executor* service is down, run either of the following commands to start the service:

   ```
   sudo systemctl restart zdp-executor
   ```

   ```
   sudo service zdp-executor restart
   ```

Note: Check the executor log to gather more information.

2.2 Workflows in the Queue

If the workflows triggered from ZDP pile up in the **QUEUED** state for considerable period of time and do not switch to the **RUNNING** state, perform the following steps:

- To check the status of the *zdp-activemq* service, run either of the following commands:

  ```
  sudo systemctl status zdp-activemq
  ```

  ```
  sudo service zdp-activemq status
  ```

  If the browser-page is non-responsive, *zdp-activemq* services are down and must be restarted.

- To restart the *zdp-activemq* service, run either of the following commands:

  ```
  sudo systemctl restart zdp-activemq
  ```

  ```
  sudo service zdp-activemq restart
  ```

Note: Once the *zdp-activemq* service is up, the queued workflows are not immediately addressed but are resolved within a particular duration automatically.

2.3 Ingestion-warden Down

The *zdp-ingestion-warden* service can be down if the landing zone server is down or if there are configuration issues related to the source directory(s) (or file pattern associated). In such a case, files may not get ingested. Check the *zdp-ingestion-warden* logs first if the service does not respond.

If **zdp-ingestion-warden** is down, perform the following:

1. To restart the *zdp-ingestion-warden* service, run either of the following commands:

   ```
   sudo systemctl restart zdp-ingestion-warden
   ```

   ```
   sudo service zdp-ingestion-warden restart
   ```

2. If no slave is launched by the *zdp-ingestion-warden* service, ensure that the corresponding source directory is a valid path and physically exists in the server.

2.4 ZDP start-up fails

There can be a scenario when the ZDP application and the executor(s) fail to start even after running the start-up script for the *zdp-gateway* and the *zdp-executor*. This can be observed after an installation/upgrade displaying the following error in the server log file (such as *localhost.$DATE.log*):

```
SEVERE: Exception sending context initialized event to listener instance of class org.
↪springframework.web.context.ContextLoaderListener
org.jasypt.exceptions.EncryptionOperationNotPossibleException
```

To overcome such a scenario, perform the following steps:

1. Ensure that *jce.jar* is located in *$JAVA_HOME/lib/* directory.
2. Perform ZDP upgrade (or install) using assistance from Zaloni Support.

2.5 Reports not Generating Metrics

Scenario 1:

If the ZDP reporting module does not generate any metrics, perform the following:

1. To check the status of the *zdp-es* service, run either of the following commands:

```
sudo systemctl status zdp-es
```

```
sudo service zdp-es status
```

2. If the *zdp-es* service is down, run either of the following commands to start the service:

```
sudo systemctl restart zdp-es
```

```
sudo service zdp-es restart
```

Scenario 2:

If ZDP is freshly installed and no reports are generated, ensure that the Elasticsearch index is properly set. If yes, repeat the steps described under Scenario 1.

Scenario 3:

If the *zdp-es* service is active and metrics do not get generated, perform the following:

1. To check the status of the *zdp-logstash* service, run either of the following commands:

```
sudo systemctl status zdp-logstash
```

```
sudo service zdp-logstash status
```

2. If the *zdp-logstash* service is down, run either of the following commands to start the service:

```
sudo systemctl start zdp-logstash
```

```
sudo service zdp-logstash start
```

Scenario 4:

If you still encounter issues, check if the the job history server is up.

1. Type the job history server IP in your browser address bar in the format: `<name_node>:<job_history_server_port>`.
2. If the job history server is down, ask your enterprise level administrator to restart.

2.6 Lineage Graphs not Generated

Lineage may not be generated in ZDP due to the following reasons:

- **Scenario 1**:

  The *zdp-lineage* service may be down. To rectify this issue, perform the following steps:

  1. To check the status of the *zdp-lineage* service, run either of the following commands:

  ```
  sudo systemctl status zdp-lineage
  ```

  ```
  sudo service zdp-lineage status
  ```

  2. If the *zdp-lineage* service is down, run either of the following commands to start the service:

```
sudo systemctl start zdp-lineage
```

```
sudo service zdp-lineage start
```

- **Scenario 2**:

  The *zdp-es* service might be down. To fix this issue, perform the following steps:

  1. To check the status of the *zdp-es* service, run either of the following commands:

  ```
  sudo systemctl status zdp-es
  ```

  ```
  sudo service zdp-es status
  ```

  2. If the *zdp-es* service is down, run either of the following commands to start the service:

  ```
  sudo systemctl start zdp-es
  ```

  ```
  sudo service zdp-es start
  ```

2.7 MapR Considerations

- To execute the **Parquet Entity Action** in a *MapR VM*, add the *tmpDirPrefix* property to *HiveEntityFormatConverter* in the *jobContext.xml* file and set the value of the property as *maprfs:///*.
- If **Sqoop Import Action** fails in the *MapR* environment, add (or check) for the following properties in the *yarn-site.xml* file:

  1. Specify the *zookeeper* address.

  ```
  <property>
         <name>yarn.resourcemanager.zk-address</name>
         <value>maprdemo:5181</value>
  </property>
  ```

  2. Add the *automatic-failover* property (if not specified) and set the value to *true*.

  ```
  <property>
   <name>yarn.resourcemanager.ha.automatic-failover.enabled</name>
   <value>true</value>
  </property>
  ```

2.8 Encountering Toast Errors

ZDP toast notifies environment related issues (if any) as a pop-up message with a red background. A few examples are listed below:

- *An internal error has occurred. Please contact your system administrator.*

  This message is displayed in an error toast while logging in ZDP, indicating that the *zdp-gateway* service is unable to communicate with the *zdp-ingestion-warden* service and encountering error.

- *Failed*

  If the *zdp-es* service is down, error toasts displaying *Failed* constantly pops up. This is a rare scenario as ZDP always expects the *zdp-es* service to be up.

For more information, refer to the *Reports not Generating Metrics* section.

2.9 Transformations fail with impersonation

Your transformations fail in an impersonation-enabled ZDP instance, thereby displaying the following error in the log:

```
org.apache.hadoop.security.AccessControlException: Permission denied: user=john,␣
↪access=WRITE, inode="/user/john/.sparkStaging/application_1487241589577_0004
↪":hdfs:hdfs:drwxr-xr-x

       at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
↪check(FSPermissionChecker.java:319)
       at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
↪check(FSPermissionChecker.java:292)
       at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
↪checkPermission(FSPermissionChecker.java:213)
```

Such error can be encountered, if the user triggering the workflow (or logged-in user) do not own (or have) a hdfs home directory (such as */user/<username>*).

To troubleshoot such an issue, perform the following steps:

1. Provide the path against the following properties in the *mapred-site.xml* file located under *$HADOOP_HOME/conf* location.

   An excerpt from a *mapred-site.xml* file is displayed here.

   ```
   <property>
     <name>yarn.app.mapreduce.am.staging-dir</name>
     <value>/tmp/staging</value>
   </property>

   <property>
     <name>mapreduce.jobtracker.staging.root.dir</name>
     <value>/tmp/mapred/staging</value>
   </property>
   ```

   Ensure that the path provided for these properties is accesible to all users triggering workflows in ZDP.

2. Using */tmp* for such properties may not be a secured approach, as */tmp* is ideally accessible to all users. So, when a spark job processes over secured files, there might be a possibility of certain output files being stored (or left over) within such location after execution.

To follow an alternative approach overcoming such security risks, refer to the section transformation_action section.

2.10 Connect LDAP server with SSL

If you want to import LDAP users (and groups) from a *LDAP* server with SSL enabled, you might encounter a pop-up error while trying to connect (or testing the configuration). Such error can indicate that the ZDP is unable to connect to the LDAP server as the *SSL* certificate (installed in the *LDAP* server) is missing in the *Java CA Certificates Store* within the *Java* package (used for ZDP).

To troubleshoot such an issue, perform the following steps:

1. Import the *SSL* certificate from the *LDAP* Server and store in an accesible location (within the server where the *zdp-gateway* service is running). For example, you can store a SSL certificate: *ldap-ssl.cer* in a location: */home/bedrock/ssl* within the server where the *zdp-gateway* service is installed.
2. In the server terminal, check the Java package path (used for ZDP) in the node. For example */usr/java/jdk1.7.0_67*.
3. Within the Java package, check for the location where the *cacerts* file is located. For example, */usr/java/jdk1.7.0_67/jre/lib/security/cacerts*.
4. Run the following command to install (or add) the *SSL* certificate (imported from the *LDAP* server) into the *Trust CA* for *JAVA*.

   ```
   Command -
   keytool -importcert -noprompt -storepass changeit -file /<absolute_path_of_SSL_
   ↪Certificate> -alias <SSL_Certificate_Alias_Name> -keystore <java_package_
   ↪location>/jre/lib/security/cacerts

   Example - keytool -importcert -noprompt -storepass changeit -file /home/bedrock/
   ↪ssl/ldap-ssl.cer -alias ldapserverssl -keystore
   ```

   The system displays a confirmation message (in the terminal) on successful installation of the *SSL* certificate.
5. Restart the *zdp-gateway* service by using the command provided in the *Application Start-up and Shut-down* section.
6. Once the *zdp-gateway* service restarts, navigate to **Administration** > **Users** > **LDAP CONFIGURATION** and enable *SSL* by using the toggle-button.
7. Provide the SSL port (default *636*) and update the applicable *LDAP* parameters (if any).
8. Click **TEST CONFIGURATION** to check if ZDP can connect to the LDAP server by using the current configuration properties.

For more information on LDAP properties, refer to the configure_ldap section.

2.11 Ranger policies not created

If Ranger policies are not being created and the *zdp-gateway* logs do not reflect the exact issues, you can access the Ranger logs available in the instance where Ranger is installed. For example, *xa_portal.log* in the *$RANGER_HOME/var/log/ranger/admin* location.

2.12 400 Bad Request in the Gateway log

If the value set for *ranger.service.dfs* property in the *ranger.properties* file is incorrect, the entities/file patterns get created in ZDP. However, the Ranger policies are not created for the specific project(s).

In such cases, the *zdp-gateway* service logs an *ERROR* with a message, as follows:

```
2017-01-16T13:36:14,783 [http-nio-9091-exec-9] ERROR - com.zaloni.bedrock.security.
↪mgr.ranger.ClusterSecurityManagerRangerImpl- Error creating policy with name BR (
↪<BEDROCK_HOST_URL>) # 10 (Owned)
org.springframework.web.client.HttpClientErrorException: 400 Bad Request
```

To troubleshoot such an issue, update the correct value for the *ranger.service.dfs* property in the *ranger.properties* file located under the *$ZDP_HOME/Gateway/config/* directory.

2.13 Permission issue while creating entity(s)

In case the entity creation operation is failing, check the permissions for the ZDP service user and the impersonated (Entity "Owner") user on the table location.

Note that the entity creation operation is not dependent on Ranger integration. The Ranger policy for the entity is created only after the entity creation is successful.

2.14 Restore ZDP generated Ranger polices

In case a ZDP generated Ranger policy is modified manually, you can restore:

1. When the policy for the project is synced for the next time. The sync is based on the tri_ggers.
2. Alternately, the changes can also be overwritten by calling the sync_project_ranger service.

2.15 Connectivity between an EC2 node and EMR cluster

Ideally, to ensure connectivity between the *EMR HDFS* nodes and the *EC2* edge node, the *EC2* edge node and the *EMR* cluster nodes must be connected to the same LAN or VLAN.

To successfully ingest file from an edge node to *HDFS* cluster, the *HDFS* architecture expects that the edge node has access to all the nodes (via IP addresses) that are part of the HDFS cluster. If an *EC2* node is considered as an edge node and you want to access an *EMR* cluster, you must ensure that all the nodes that are part of the EMR cluster are accessible from the EC2 edge node.

2.16 Exception in micro-service logs

While saving (or updating) entity in the *EMR* cluster, the micro-service logs might display a *WARN* message:

```
2017-01-25T10:36:46,959 [http-nio-9719-exec-2] WARN  - com.zaloni.bedrock.hive.
↪hcatalog.service.HCatalogClient- Seems like an old version of hive. Retrying with␣
↪old method signature.
```

However, this exception does not restrict entities from getting created (or updated) in ZDP.

Refer to the following points to know more about such scenario.

- **Possible detection**: Such log message can be observed when your environment is set up with *EMR 4.8.x* and *Hive 1.0.0* (or older versions of Hive). This exception can be ignored.
- **Overcoming such scenario**: To prevent such exception from the logs, upgrade Hive to the ZDP recommended version as described in the software_req section.

2.17 Landing Zone directories, created by using s3a, do not work

If you get the `IllegalArgumentException` exception, set the value for the *fs.s3a.threads.max* key to a value greater than the one set for the *fs.s3a.threads.core* key. You can set these values under **Administration** > **Connections** > **ADD**. For more information, refer to the s3a_connection section.

2.18 Ingestion using S3 connection failing in HDP

Ingestion might fail while using a *S3* connection (already defined in ZDP). In such a case, the following error is displayed in the *zdp-ingestion-warden* logs:

```
...
     java.lang.NumberFormatException: For input string: "100M"
     at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)␣
↪~[?:1.8.0_112]
...
```

To overcome such an issue, perform the following steps:

1. Navigate to **Administration** > **Connections** > **ADD** (for existing *S3* connection) in the ZDP UI.
2. Add the following **File System Connection Property Key** and the value based on the requirement.

Connection Property	Value
`fs.s3a.multipart.size`	Example value - *104857600* **Note:** The value must be provided in *bytes* as ZDP (with *Hadoop 2.7.x*) does not support scaling numeric values (using *K,M,G,T,P*), such as *100M*, *10G*, or *1T* .

3. Save the changes for the S3 connection and restart the corresponding the *zdp-ingestion-warden* service (if inactive) to perform ingestion successfully.

For more information on ZDP connections, refer to the def_con_view and s3a_connection sections.

2.19 Errors in DB Import Action

In **DB Import Action**, you may encounter the following errors:

- **java.sql.SQLSyntaxErrorException: ORA-00933: SQL command not properly ended**:

  Specify the option *– –schema your_schema* in **Sqoop additional options**.

- **ERROR tool.ImportTool: Encountered IOException running import job: java.io.IOException: Generating splits for a textual index column allowed only in case of "-Dorg.apache.sqoop.splitter.allow_text_splitter=true"**:

  This error occurs when your primary key or the key on which split is done by Sqoop is of *text* data type *(varchar, char, clob, text)* and your environment is HDP 2.5.

  To avoid getting this error, you must set the key: *GENERIC_OPTIONS* as *-Dorg.apache.sqoop.splitter.allow_text_splitter=true* in **DB import additional options**.

- **Caused by: java.sql.SQLException: [Teradata Database] [TeraJDBC 15.00.00.35] [Error 3707] [SQLState 42000] Syntax error, expected something like '(' between the 'title' keyword and ',':**

  This error occurs while importing tables from the Teradata database when the table contains a column with reserved keyword. The Teradata JDBC driver cannot handle such a scenario and hence, the Sqoop import fails.

  To avoid that, you must remove the tables using the exclude regex option as the tables cannot be imported by using Sqoop.

- **ERROR orm.ClassWriter: No Java type for SQL type 1266 for column SOME_COLUMN_NAME**:

  This error occurs while importing tables from the POSTGRESQL database that has columns with the *JSON* and *TIME* data types.

  To avoid getting this error, you can specify the option: *–map-column-java SOME_COLUMN_NAME=String* in **Sqoop additional options**, which instructs Sqoop to use the *String* data type for writing the column data.

  Note: Do not use this option to import multiple tables as the column name would not be found for other tables. Hence, you must use it to import only the specific table which has that specific column.

- **Missing resource associations for artifact identifiers** (Entity creation fails with this error message):

  This error occurs if you run the DB Import workflow with some connection/options in Project A and create the same workflow in Project B with different name. Thus, ZDP tries to create another entity (under Project B) with the same name and properties as those created by the workflow in Project A.

  To avoid getting this error, you must define a unique **Source Platform** value in the **Db Import Action** (for both the projects) in order to create entities with unique names.

- **org.apache.avro.reflect.ReflectData.addLogicalTypeConversion(Lorg/apache/avro/Conversion;)V**:

  To avoid getting this error, you must set the key: *GENERIC_OPTIONS* as *-Dmapreduce.job.user.classpath.first=true* in **DB import additional options**. For example, *GENERIC_OPTIONS = -Dmapreduce.job.user.classpath.first=true*.

2.20 Duplicate entries are created in the Hive table while importing tables from any database by using DB Import Action

This error occurs if the primary index or the column on which Sqoop performs split, is of the *text* data type *(varchar, char, text, etc)*. This is a Sqoop bug: Sqoop-2664.

To avoid getting this error, you can use the *–split-by <column>* option on a different column for that table in **Sqoop additional options**. For more information, refer to Stackoverflow.

2.21 Defining column names for ORACLE database

Oracle tables can have duplicate columns as the column names are case sensitive. Hence, you can define two columns, such as *COLUMN_1* and *ColuMN_1*. However, entities of such tables cannot be created in ZDP and Hive as column names are case insensitive and ZDP/Hive would consider these columns as duplicate. Hence, you must define unique column names for a certain table.

2.22 Null values in Hive for Boolean values in the SAP_HANA database

This issue occurs while importing tables from the SAP_HANA database that has columns with the *BOOLEAN* data type. Such values are ingested as null values in Hive. Sqoop cannot ingest the Boolean values as *true/false* but only as *1/0*. SAP_HANA uses a generic JDBC connector, which cannot map the Boolean values in Hive properly.

To avoid this error, add the *–map-column-java* argument in **Sqoop additional options**, defining the field name which has the Boolean value. For example, if the field name is *Boolvalue*, use the argument as: *–map-column-java BOOLVALUE=Boolean*.

2.23 Timeout issue in Data Inventory Action

If you get timeout issue while executing the **Data Inventory Action**, set the key and value as *mapreduce.task.timeout* and *0* respectively under **MR/Spark Configurations**. For more information, refer to the data_inventory_action section.

2.24 Yarn Logs Aggregation Error

To avoid getting Yarn logs aggregation error, set the following properties in the *yarn-site.xml* file:

- *yarn.log-aggregation-enable* to *true*
- *yarn.log-aggregation.retain-seconds* to a custom value
- *yarn.nodemanager.log-aggregation.compression-type* to a custom value
- *yarn.log.server.url* to a custom value
- *yarn.log.server.web-service.url* to a custom value

2.25 Data Quality on the Decimal Data Type

To run the **Data Quality Action** on entities with the *DECIMAL* data type fields (in the MapR distribution), add the following:

- In *hive-site.xml*, set:

```
<property>
  <name>hive.aux.jars.path</name>
  <value>file:///share/auxlib/hive-hcatalog-core-2.1.0.jar (sample value)</value>
  <description>Path to Jar file</description>
</property>
```

- In the *hive-env.sh* file under *$HIVE_HOME/conf*, set:

```
export HIVE_AUX_JARS_PATH=/home/mapr/auxlib/hive-hcatalog-core-2.1.0.jar
```

Note: This is applicable for Hive-2.1 in a MapR environment.

2.26 Describing a JSON Entity in Hive in a MapR Distribution

To define a JSON entity in Hive in a MapR distribution, add the following:

- In *hive-site.xml*, set:

```
<property>
  <name>hive.aux.jars.path</name>
  <value>file:///share/auxlib/hive-hcatalog-core-2.1.0.jar (sample value)</value>
  <description>Path to Jar file</description>
</property>
```

- In the *hive-env.sh* file under *$HIVE_HOME/conf*, set:

```
export HIVE_AUX_JARS_PATH=/home/mapr/auxlib/hive-hcatalog-core-2.1.0.jar
```

Note: This is applicable for Hive-2.1 in a MapR environment.

2.27 How to avoid getting the java.io.IOException exception during entity synchronization for large dataset during cluster connection

If the entity-Hive synchronization fails for more than *15000* entities where each entity has *100* fields, you can ensure batch-wise synchronization during cluster connection. To do so, set the parameter: *metadata.synchronization.batchsize* (under */etc/zdp-gateway/zdp-gateway.yml*) to an appropriate value. The default value for the key is *100*.

2.28 How to avoid getting the permission denied exception in the Yarn log location in an EMR environment

While executing workflows that run MapReduce or Hadoop jobs in an EMR environment, you might get the permission denied error. This occurs if there is no appropriate permission in the location where Yarn logs are generated.

To avoid getting this error, access the *yarn-site.xml* file and set the parameter: *yarn.nodemanager.remote-app-log-dir* to a HDFS path with the 777 permission.

Alternatively, you can set the same key in either of the **MR/Spark Configurations**, **Spark Configurations**, or **MR Configurations** field along with the path value, while defining individual workflow actions that generate *Yarn* logs. For example, *conf yarn.nodemanager.remote-app-log-dir=/tmp/zdp-temp*. However, ensure that the specified path has the 777 permission.

2.29 How to run DME in EMR environment

To run DME in an *EMR* environment, set the following property of */etc/zdp-dme/zdp-dme.yml* file.

Properties	Description
eureka.environment.cloud.enabled	Set the value as *true*. Default value is *false*

2.30 Error in MySQL cluster and ZDP connection

While you use the high availability feature for MySQl cluster, you might encounter the following issues:

- If you use multiple routers in a MySQL cluster, perform the write operation on the MySQL database by executing a workflow, and one of the routers go down, the write operation fails with the following exception:

```
{
     "responseMessage":"Could not open JDBC Connection for transaction; nested␣
↪exception is com.mysql.jdbc.exceptions.jdbc4.CommunicationsException:␣
↪Communications link failure",
     "restUri":"/bedrock-app/services/rest/workflows/3/execute",
     "result":null,
     "page":null
    }
```

- If you use multiple routers in a MySQL cluster, perform the write operation on the MySQL database by creating entity, and one of the routers go down, the write operation fails with the following exception:

```
{
    "timestamp":1530786075970,
    "status":500,
    "error":"Internal Server Error",
    "exception":"org.springframework.transaction.CannotCreateTransactionException
↪",
    "message":"Could not open JDBC Connection for transaction; nested exception␣
↪is com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link␣
↪failure\n\nThe last packet successfully received from the server was 380␣
↪milliseconds ago.  The last packet sent successfully to the server was 1␣
↪milliseconds ago.","path":"/bedrock-app/services/rest/projects/1/entities"
    }
```

- If you use multiple routers in a MySQL cluster, perform the read/write operation on the MySQL database by logging into the ZDP UI, and one of the routers go down, the write operation fails with the following exception:

```
{
    "timestamp":1530785961317,
    "status":500,
    "error":"Internal Server Error",
    "exception":"org.springframework.transaction.CannotCreateTransactionException
↪",
    "message":"Could not open JDBC Connection for transaction; nested exception␣
↪is com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link␣
↪failure\n\nThe last packet successfully received from the server was 354␣
↪milliseconds ago.  The last packet sent successfully to the server was 0␣
↪milliseconds ago.","path":"/bedrock-app/services/rest/admin/getUserRole"
    }
```

To troubleshoot the above scenarios, you can ensure that the specific router is up before performing any read/write operation in the MySQL cluster. However, after some duration, the write operation works successfully (even if one of the routers is down).

- If the MySQL service in the primary MySQL server (the server with read-write permission) is down and various ZDP operations (creating entity, executing workflow, logging in to the ZDP application) are performed simultaneously, the system displays the following exception:

```
java.net.SocketException: Socket closed
    at java.net.SocketInputStream.socketRead0(Native Method)
    at java.net.SocketInputStream.socketRead(SocketInputStream.java:116)
    at java.net.SocketInputStream.read(SocketInputStream.java:170)
    at java.net.SocketInputStream.read(SocketInputStream.java:141)
    at org.apache.http.impl.io.AbstractSessionInputBuffer.
↪fillBuffer(AbstractSessionInputBuffer.java:161)
    at org.apache.http.impl.io.SocketInputBuffer.fillBuffer(SocketInputBuffer.
↪java:82)
    at org.apache.http.impl.io.AbstractSessionInputBuffer.
↪readLine(AbstractSessionInputBuffer.java:278)
    at org.apache.http.impl.conn.DefaultHttpResponseParser.
↪parseHead(DefaultHttpResponseParser.java:138)
    at org.apache.http.impl.conn.DefaultHttpResponseParser.
↪parseHead(DefaultHttpResponseParser.java:56)
    at org.apache.http.impl.io.AbstractMessageParser.parse(AbstractMessageParser.
↪java:259)
    at org.apache.http.impl.AbstractHttpClientConnection.
↪receiveResponseHeader(AbstractHttpClientConnection.java:286)
    at org.apache.http.impl.conn.DefaultClientConnection.
↪receiveResponseHeader(DefaultClientConnection.java:257)
    at org.apache.jmeter.protocol.http.sampler.hc.ManagedClientConnectionImpl.
↪receiveResponseHeader(ManagedClientConnectionImpl.java:199)
    at org.apache.jmeter.protocol.http.sampler.MeasuringConnectionManager
↪$MeasuredConnection.receiveResponseHeader(MeasuringConnectionManager.java:212)
    at org.apache.http.protocol.HttpRequestExecutor.
↪doReceiveResponse(HttpRequestExecutor.java:273)
    at org.apache.http.protocol.HttpRequestExecutor.execute(HttpRequestExecutor.
↪java:125)
    at org.apache.http.impl.client.DefaultRequestDirector.
↪tryExecute(DefaultRequestDirector.java:684)
    at org.apache.http.impl.client.DefaultRequestDirector.
↪execute(DefaultRequestDirector.java:486)
    at org.apache.http.impl.client.AbstractHttpClient.
↪doExecute(AbstractHttpClient.java:835)
    at org.apache.http.impl.client.CloseableHttpClient.
↪execute(CloseableHttpClient.java:83)
    at org.apache.jmeter.protocol.http.sampler.HTTPHC4Impl.
↪executeRequest(HTTPHC4Impl.java:697)
    at org.apache.jmeter.protocol.http.sampler.HTTPHC4Impl.sample(HTTPHC4Impl.
↪java:455)
    at org.apache.jmeter.protocol.http.sampler.HTTPSamplerProxy.
↪sample(HTTPSamplerProxy.java:74)
    at org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.
↪sample(HTTPSamplerBase.java:1189)
    at org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.
↪sample(HTTPSamplerBase.java:1178)
    at org.apache.jmeter.threads.JMeterThread.executeSamplePackage(JMeterThread.
↪java:490)
    at org.apache.jmeter.threads.JMeterThread.processSampler(JMeterThread.
↪java:416)
```

```
    at org.apache.jmeter.threads.JMeterThread.run(JMeterThread.java:250)
    at java.lang.Thread.run(Thread.java:745)
```

To troubleshoot this issue, ensure that MySQL service in the primary MySQL server is up.

2.31 How to run Spark transformations in an Azure-based HDInsight cluster

- Add the following paths in the */etc/zdp-spark/zdp-spark.conf* file:

```
sparkExtraClassPath=/etc/hadoop/conf/:/usr/lib/hdinsight-datalake/*:/usr/hdp/2.6.
↪3.40-13/hadoop/lib/hadoop-lzo-0.6.0.2.6.3.40-13.jar:/usr/hdp/current/hadoop-
↪client/hadoop-azure.jar:/usr/hdp/current/hadoop-client/lib/azure-storage-5.4.0.
↪jar:/usr/lib/hdinsight-logging/mdsdclient-1.0.jar:/usr/lib/hdinsight-logging/
↪microsoft-log4j-etwappender-1.0.jar:/usr/lib/hdinsight-logging/json-simple-1.1.
↪jar
```

- Add the following paths in the *spark-env.sh* file under the *${SPARK_HOME}/conf* location:

```
SPARK_DIST_CLASSPATH=$SPARK_DIST_CLASSPATH:/usr/hdp/current/spark2-client/jars/*:/
↪usr/lib/hdinsight-datalake/*:/usr/hdp/current/spark2-client/conf:/usr/hdp/2.6.3.
↪40-13/hadoop/lib/hadoop-lzo-0.6.0.2.6.3.40-13.jar:/usr/hdp/current/hadoop-
↪client/hadoop-azure.jar:/usr/hdp/current/hadoop-client/lib/azure-storage-5.4.0.
↪jar:/usr/lib/hdinsight-logging/mdsdclient-1.0.jar:/usr/lib/hdinsight-logging/
↪microsoft-log4j-etwappender-1.0.jar:/usr/lib/hdinsight-logging/json-simple-1.1.
↪jar
```

2.32 How to perform provisioning for entities with the JSON data format in CDH/EMR

To do so in CDH, you must add the *hive-hcatlog-core.jar* in the *spark-env.sh* file through the Cloudera manager.

```
export SPARK_DIST_CLASSPATH=${SPARK_DIST_CLASSPATH}:/opt/cloudera/parcels/CDH-5.12.1-
↪1.cdh5.12.1.p0.3/lib/hive-hcatalog/share/hcatalog/hive-hcatalog-core.jar
```

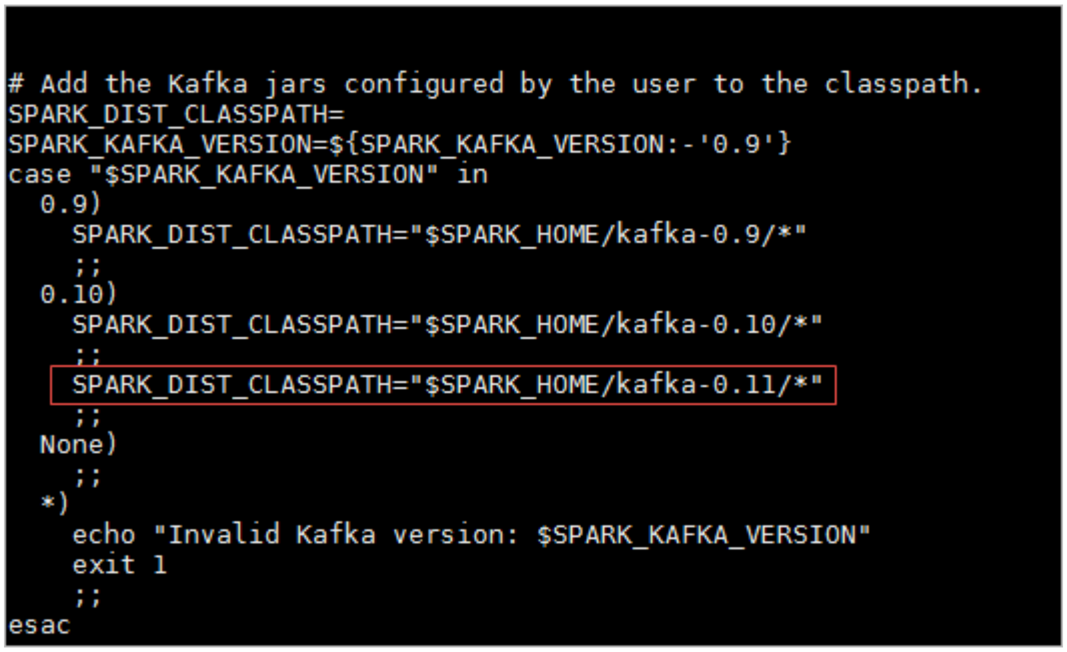
To do so in EMR, you must place the *hive-hcatlog-core.jar* in the location where the Spark Jars are located. For example, */usr/lib/spark/jars/*.

2.33 How to use Kafka 0.11.0 in a CDH environment

If you use Kafka 0.11.0 and higher in a CDH environment, you must manually define the Kafka properties in ${SPARK_HOME}.

To do so, perform the following steps:

1. Create the folder: *kafka-0.11.0* under $SPARK_HOME and move the Kafka-0.11.0 specific Jars from the Kafka installation directory to this folder.
2. Add the Kafka-0.11.0 Jars' path to the classpath in the *${SPARK_HOME}/conf/spark-env.sh* file, as follows:

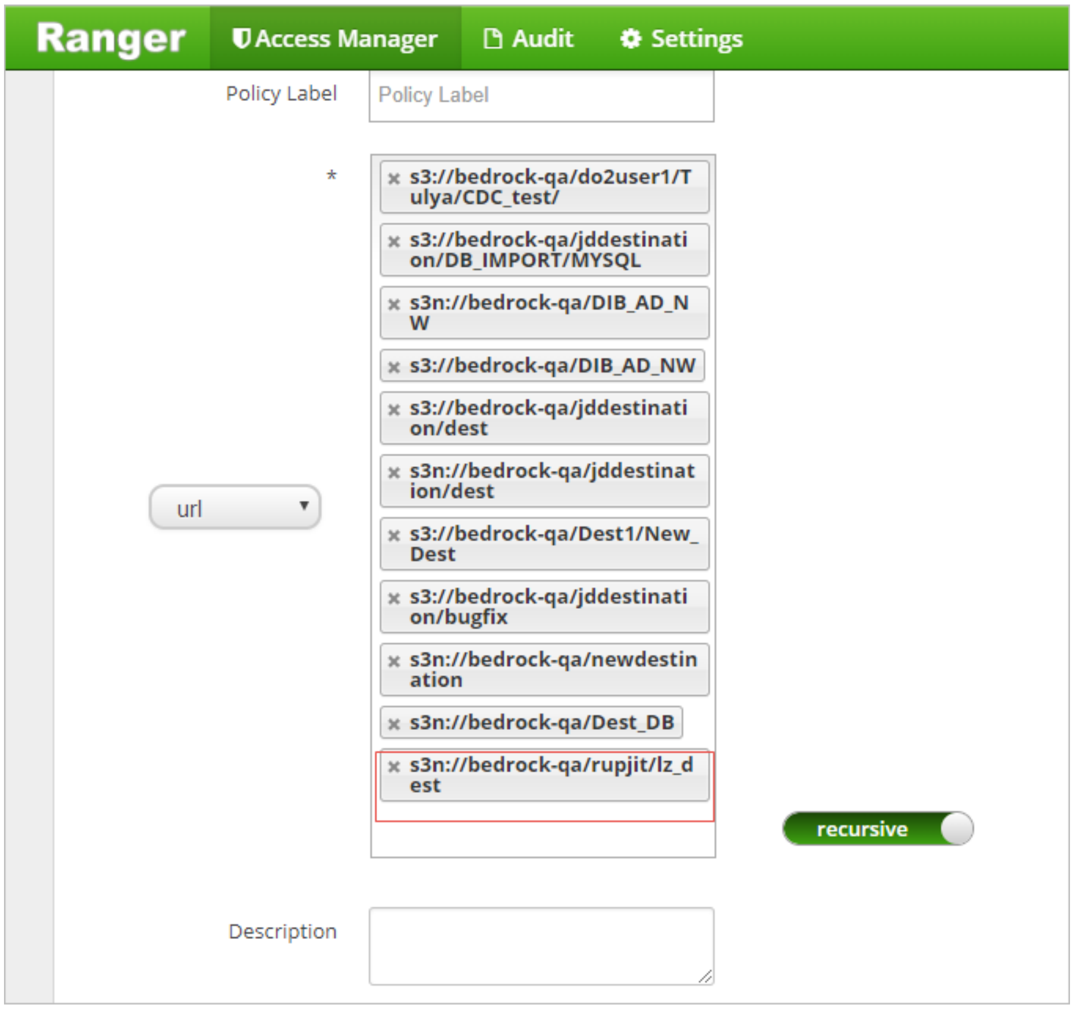


```
# Add the Kafka jars configured by the user to the classpath.
SPARK_DIST_CLASSPATH=
SPARK_KAFKA_VERSION=${SPARK_KAFKA_VERSION:-'0.9'}
case "$SPARK_KAFKA_VERSION" in
  0.9)
    SPARK_DIST_CLASSPATH="$SPARK_HOME/kafka-0.9/*"
    ;;
  0.10)
    SPARK_DIST_CLASSPATH="$SPARK_HOME/kafka-0.10/*"
    ;;
    SPARK_DIST_CLASSPATH="$SPARK_HOME/kafka-0.11/*"
    ;;
  None)
    ;;
  *)
    echo "Invalid Kafka version: $SPARK_KAFKA_VERSION"
    exit 1
    ;;
esac
```

2.34 How to successfully execute DB Import Action in an EMR environment with S3 connections

To successfully execute **DB Import Action** in an EMR environment with S3 connections, perform the following steps:

- While providing S3 as the output path, ensure that you specify the protocol as well in the format: **s3n://<bucket_name>/<folder_path>**. For example, *s3n://bedrock-qa/rupjit/lz_dest*.
- In Ranger, define a policy where the same output path is added.



s3://bedrock-qa/do2user1/Tulya/CDC_test/
s3://bedrock-qa/jddestination/DB_IMPORT/MYSQL
s3n://bedrock-qa/DIB_AD_NW
s3://bedrock-qa/DIB_AD_NW
s3://bedrock-qa/jddestination/dest
s3n://bedrock-qa/jddestination/dest
s3://bedrock-qa/Dest1/New_Dest
s3://bedrock-qa/jddestination/bugfix
s3n://bedrock-qa/newdestination
s3n://bedrock-qa/Dest_DB
s3n://bedrock-qa/rupjit/lz_dest

2.35 Executing Entity Data Quality Action in EMR environment

To execute the **Entity Data Quality Action** in an EMR environment, ensure that you grant read-write access for all the AD users to the *hdfs:///var/log/spark/apps* path (defined in the *spark-defaults.conf* file).

Important: For any further assistance related to ZDP, contact support.zaloni.com.