ZALONI DATA PLATFORM

Pre-requisites 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 Pre-requisites Guide details the various softwares/hardwares that are needed to use the Zaloni data platform. Ensure that you go through this guide before you start the installation process.

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

PLANNING INSTALLATION

Before performing a ZDP installation, it is recommended that you meet the *Hardware Requirements*, *Software Requirements*, and *Environment Requirements* .

1.1 Hardware Requirements

To configure the Zaloni Data Platform in a production environment, the recommendation is to use two or more physical servers with shared storage volumes. Networking should based on the 10 GbE (Gigabit Ethernet) infrastructure. The following are the recommended hardware requirements:

- 16 core 64-bit processor
- 500 GB local storage
- 500 GB shared storage
- 64 GB RAM (This value depends on the Heap size set for the ZDP services (zdp-gateway, zdp-executor, zdp-spark, zdp-mr, zdp-hive, zdp-kibana, zdp-lineage, zdp-dme, zdp-es, zdp-ingestion-warden, zdp-activemq, zdp-logstash). Ensure that the combined Heap size for all the ZDP services is less than this value).
- 4 GB Heap size for all the ZDP services (for Development and POC environments). However, for UAT and Production environments, the Heap size must be increased (based on the load).

Important: These hardware requirements may change based on various factors. For example, load of the transaction.

1.2 Software Requirements

1.2.1 Mandatory Software Requirements

Recommended versions for the following mandatory Hadoop dependencies and the related tasks:

Component	Version	Details
Linux	`6.5+, 7.0`	ZDP supports RHEL or similar (CentOS, Fedora, Amazon Linux). ZDP requires RPM, System V, and bash. ZDP services are packaged as RPM files and System V services. Hence, you must have root or sudo access on the servers where you want to install, configure, start, and stop the ZDP services.
Git	`1.0.0+`	The *zdp-registry* service requires Git to version control configuration for all the other services. Git is marked as a dependency of the *zdp-registry* RPM, and hence, will be installed automatically (if not already installed).
Hadoop distributions	`HDP (2.6.0 to 2.6.4), CDH (5.14.x), EMR (5.11)`	Install and configure Hadoop client on all the nodes where you want to run the *zdp-hive*, *zdp-mr*, and *zdp-spark* services.
Apache Hive™	`1.1 (CDH), 2.1.0 (HDP), 2.3.2 (EMR), 2.1.1 (MAPR)`	Install and configure Hive client on all the nodes where the *zdp-hive*, *zdp-mr*, and *zdp-spark* services are deployed. Set the following parameters to *true* in the *hive-site.xml* file: • *spark.hive.mapred.supports.subdirectories* • *spark.hadoop.mapreduce.input.fileinputformat.input.dir.recursive* • *hive.mapred.supports.subdirectories* • *mapred.input.dir.recursive* For more information on the security restrictions on Hive, refer to the overcome_sec_rest section.
Apache Spark™	`2.2 (CDH/HDP), 2.2.1 (EMR/MAPR)`	Install Spark in the cluster. Spark can be built from source, packaged by the distribution or downloaded from an external repository.
Java Runtime Environment	`1.8.x`	• Install Java 1.8 on all the nodes where you want to run ZDP. • Oracle Java 1.8 must not only be installed on the nodes where ZDP services are deployed, but also on the Hadoop cluster which communicates with ZDP. • To enable Java Cryptography Extensions (JCE), download the the JCE policy from Oracle-JCE, extract the files, and transfer the *local_policy.jar* and *US_export_policy.jar* files to the *${JAVA_HOME}/lib/security* directory. For more information, refer to the *README.txt* packaged with these JAR files. • Ensure that jps is enabled. • Ensure that the *jce.jar* file is located in the *${JAVA_HOME}/lib* directory. If not, you might encounter issues as described in the br_startup_fails section.
MySQL	`5.6.39, 5.7 (High Availability for MySQL cluster)`	• You must have the *CREATE DATABASE* privilege to MySQL in order to set up a database for ZDP to use. The MySQL user assigned to ZDP need all admin privileges within its database to work. • To disable the case-insensitiveness in MySQL table name, set *lower_case_table_names=0* in the MySQL settings file. • Set the *max_connections* variable to *900* or above.

Component	Version	Details
MySQL	`5.6.39, 5.7 (High Availability for MySQL cluster)`	• Set the *query_cache_size* variable to *256M* or above. • Set the *key_buffer_size* variable to *1024M* or above. • Set the *max_allowed_packet* variable to *256M* or above. • For more information on how to install MySQL, refer to the install_mysql section.
Node.js	`4.4.7 to 6.x.x`	Install Node.js from the nodejs package which is available in the standard OS repositories. Install Node.js on all the nodes where you want to run the *zdp-kibana* service. • To install on RHEL, CentOS, or Fedora Linux, run the following command: `curl --silent --location https://rpm.nodesource.com/setup_6.x | bash -` • To install in other Linux environments, see Node.js package manager. Alternatively, you can extract Node.js binaries without installing it. To do so, perform the following steps: 1. Download the appropriate *Nodejs* package by from the Nodejs v6.x repository. 2. Extract the package in a location accessible by the designated ZDP user (*zaloni* by default). 3. If the node binary is added in */usr/bin* or */usr/local/bin*, edit the */etc/sysconfig/zdp-kibana* file and set the variable: *NODE* to the *node* binary (once the *zdp-kibana* service is installed).

1.2.2 Optional Software Requirements

Recommended versions for the following optional Hadoop dependencies and the related tasks:

Component	Version	Details
Apache Flume™	`1.7.0`	Use this component to run streaming ingestion in ZDP. By default, Flume is packages with the ZDP platform and you are not required to install this component separately.
Apache Kafka™	`0.10.0 (CDH), 0.11.0 (CDH/HDP/MAPR/EMR)`	Use this component to run Kafka with streaming transformation.
Apache Ranger™	`1.1 (HDP/EMR)`	For enforcing data access policies on a Hadoop cluster.
Sentry Service	`1.5.1 (CDH)`	For enforcing user privileges to execute HiveQL operations on a location.
Apache Sqoop™	`1.4.6`	Use this to run the **Sqoop Export Action**, **Sqoop Import Action**, and **DB Import Action**.
lftp	`4.x or above`	Use this to run the **File Transfer Action**. For more information on the **File Transfer Action**, refer to the file_transfer section.

1.3 Environment Requirements

1.3.1 Linux Related Requirements

Linux *ulimit* must be set as open files: *65536* & *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.

1.3.2 Network Ports and Access Related Requirements

- An SMTP mail server must be accessible from the nodes where the *zdp-gateway/zdp-executor* services are deployed.
- All the ZDP services must have access to the port where the *zdp-registry* service is installed.
- For a multi-node installation, a shared storage location must be available for sharing data between the different ZDP services. For example, 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.
- For all the micro-services, ensure that the */var/zdp-common* folder exists in all the nodes where the micro-services are installed (similar to *zdp-gateway/zdp-executor*). However, this folder is not required to be a shared location.
- ZDP uses the following network ports by default. These should be preferably available:
  - The *zdp-registry* service uses `8761` for HTTP.
  - The *zdp-gateway* service uses `8080` for HTTP.
  - The *zdp-executor* service uses `8081` for HTTP.
  - The *zdp-mr* service uses `8087` for HTTP.
  - The *zdp-hive* service uses `8090` for HTTP.
  - The *zdp-spark* service uses `8089` HTTP.
  - The *zdp-es* service uses `19200` for HTTP and `19300` for TCP.
  - The *zdp-logstash* service uses `19600` for HTTP and `19500` for TCP.
  - The *zdp-activemq* service uses `61616` for TCP.
  - The *zdp-lineage* service uses `8084` for HTTP and `20000` for TCP.
  - The *zdp-kibana* service uses `5601` for HTTP.
  - The *zdp-ingestion-warden* service uses `8085` for HTTP.
  - The *zdp-dme* service uses `8888` for HTTP.
- To change the default ports for the *zdp-services*, update the following property in the respective */etc/<service_name>/<service_name>.yml* files:

```
server:
  port: 8089
```

- Network interconnectivity is needed between all the nodes where ZDP services are installed. The ports selected during installation are used for inter-process communication. Hence, the various ports defined during ZDP installation must be accessible from all the nodes where ZDP components are deployed.
- Ensure that the *hive-site.xml* file on the node where the *zdp-hive* service is deployed, is consistent with the nodes where the Hive daemons are running.

- If the ZDP components are installed in different nodes/machines, ensure that all such nodes/machines have the same time zone (including MySQL server).

1.4 Hive Action in ZDP

To overcome the security restrictions while executing **Hive Action** in ZDP, perform the following configurations:

1. Add the property: *hive.security.authorization.sqlstd.confwhitelist.append* and the corresponding values in the *hive-site.xml* file. For example, for the Hive version: 1.2.x, set the property as:

```
........
<property>
<name>hive.security.authorization.sqlstd.confwhitelist.append</name>
<value>
mapreduce\.job\.name|hive\.aux\.jars\.path|hive\.exec\.post\.hooks|bedrock\.prop\.
↪exec\.hook\.file|mapred\.job\.queue\.name
</value>
</property>
.........
```

Note:

- **Hive Action** jobs that use custom hiveconf variables must be set in the whitelist above.
- Based on the Hive version, use pipe or comma as the delimiter to separate multiple values for the above parameter.

If the system variable: *HIVE_ACTION_SKIP_ADD_JARS* is set to *true*, you must specify the path for the Hive hook Jar in the *hive.aux.jars.path* parameter in the *hive-site.xml* file. If the Jar path is not specified in *hive.aux.jars.path*, the **Hive Action** does not work. For example:

```
.......
<property>
<name>hive.aux.jars.path</name>
<value>/tmp/auxpath/bedrock-hive-hooks-5.0.2-FINAL.jar</value>
</property>
.......
```

2. Restart HiveServer2.

During query execution, ZDP gets the Jar either through the *ADD JAR* command or through the *hive.aux.jars.path* property defined in the *hive-site.xml* file.

1.5 EMR Related Requirements

- If files must be ingested from the *local (EBS or EFS)* file system on an EC2 node to HDFS in an EMR cluster, the EC2 node (where the BDCA will run) must have connectivity to all the nodes in the EMR cluster.

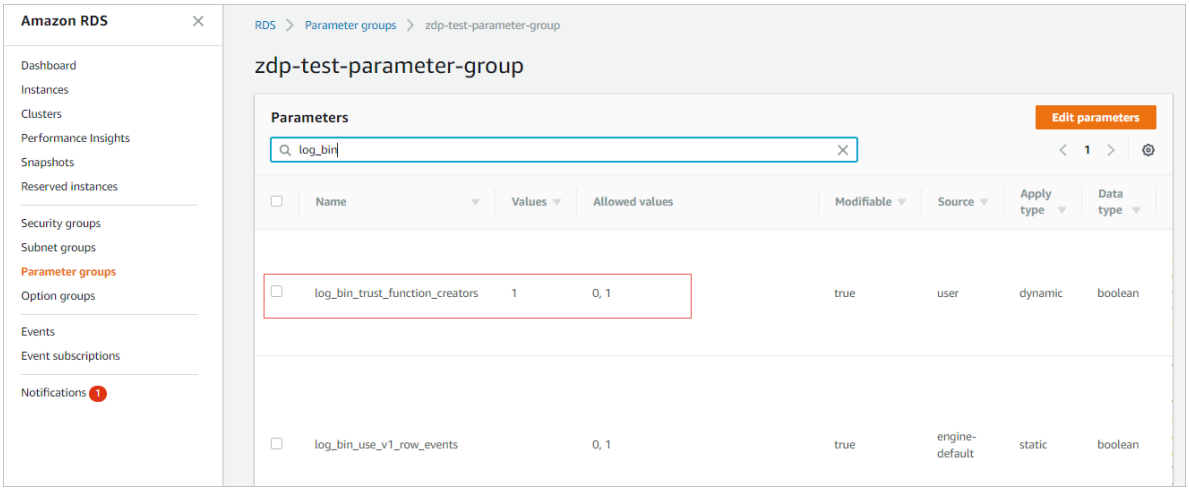
Note: This constraint is also applicable if files need to be ingested from S3 to HDFS in an EMR cluster.

- While using the RDS service in an EMR distribution to create a MySQL database, ensure that you set the *log_bin_trust_function_creators* parameter to 1, as follows (before installing ZDP):

```
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
```

Once ZDP is installed, you can reset the parameter to 0:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 0;
```



1.6 HDInsight Related Requirements

As the HDInsight cluster nodes have Ubuntu OS, *yum* must be available.

1.7 Recommended RDBMS Versions for the DB Import Action

Database	Version
MySQL	5.1.7/5.3/5.6
Oracle	11g/11i
PostgreSQL	8.4
SQL Server	SQL Server 2012
Teradata	15.00.02.08
Netezza	7.2.1.0
SAP HANA	2.00.020.00.1500920972

1.8 Configuring an EC2 Instance

Amazon Elastic Cloud Compute (EC2) allows you to create, launch, and terminate server instances as per your requirement. You can rent virtual computers on which you can run your own computer applications. It supports a scalable deployment of applications. For more information on Amazon EC2, refer to the Amazon EC2.

Using an EC2 machine is similar to using a Linux box. However, if you want to install ZDP in an EC2 instance, ensure that you install/configure AWS/MySQL in the EC2 instance. For more information, refer to the install_mysql sections.

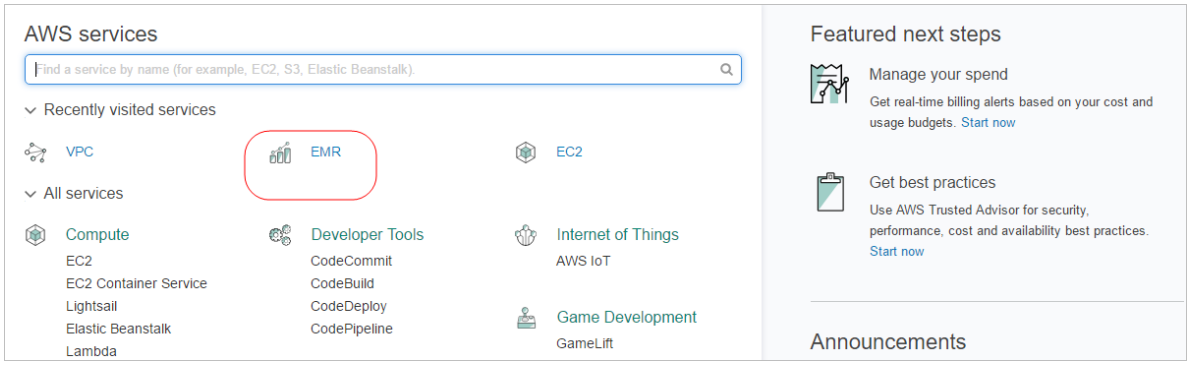
1.9 Configuring an EMR Machine

Amazon Elastic MapReduce (EMR) provides a managed Hadoop framework that can process vast amounts of data across dynamically scalable Amazon EC2 instances. For more information on EMR, refer to Amazon EMR.

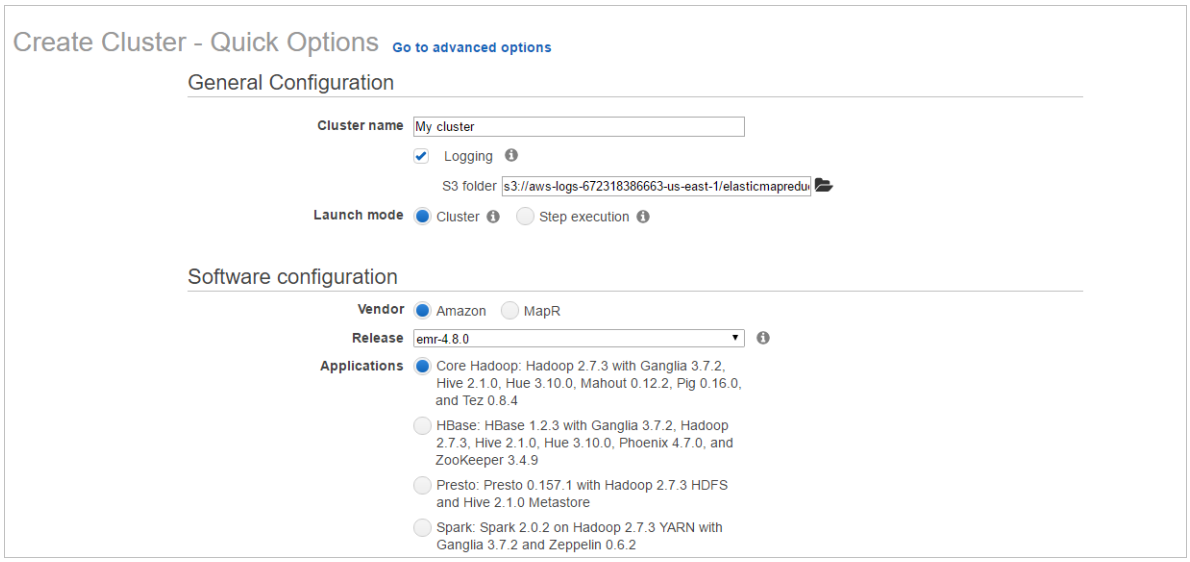
You can set up an EMR machine with various components and configure the details in the EMR master node. If you do so, all the other nodes in the EMR instance. also inherits the configured properties.

To configure an EMR machine, perform the following steps:

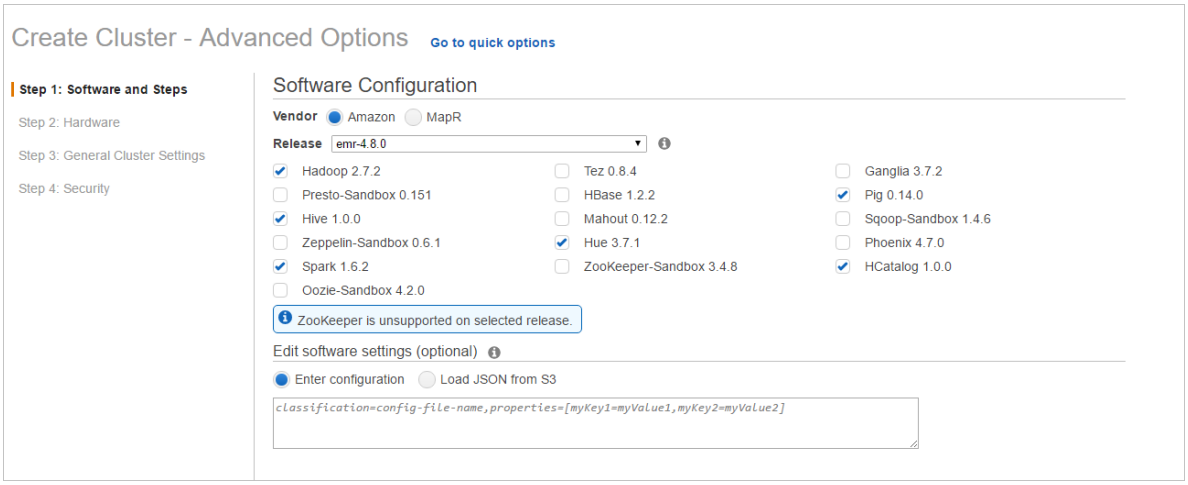
1. Log in to the AWS console.



2. Select **EMR** and click **Create cluster**. The system displays the Configuration page.



3. By default, the system selects an application bundle. To customize your application bundle, click **Go to advanced options** at the top of the page.



4. Select the ZDP-certified EMR release defined in the *Software Requirements* section. The system populates the Amazon-certified versions for all the Hadoop dependencies.
5. Select the specific Hadoop dependencies that must be available in the EMR instance for ZDP to function. For example, Hadoop, Hcatalog, Spark, Hive.
6. Complete the remaining configuration steps. For more information, refer to the EMR documentation section.

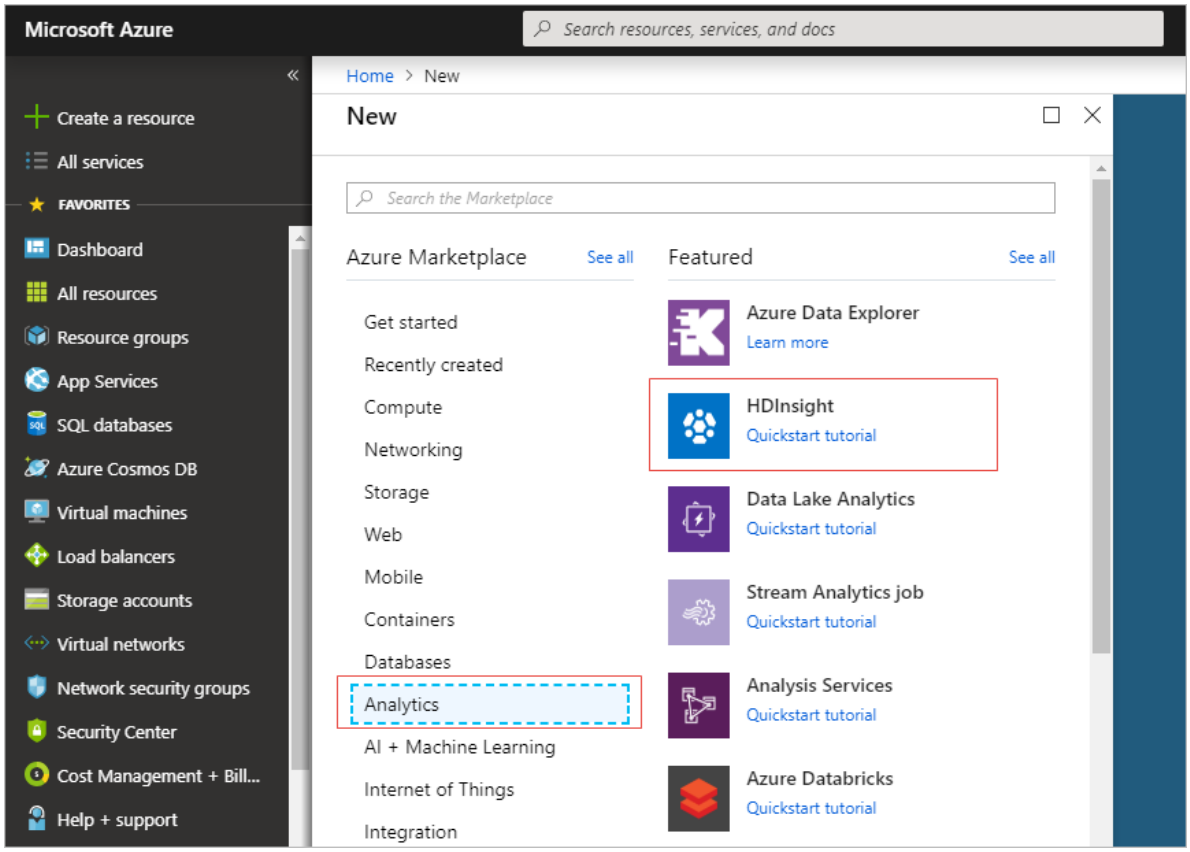
Note: It is recommended that you do not enable the **EMRFS Consistent View** check box on an EMR cluster. If you enable this check box, the behavior of HDFS is modified and standard ACLs are not used. Hence, you cannot assign ownership or group permissions on directories and files. If you enable this check box and submit a workflow action, access validation cannot take place.

1.10 Configuring Azure HDInsight cluster

To manage the data lake, an Azure HDInsight cluster can be used as the underlying layer, in conjunction with ZDP. Such a cluster can contain several components, such as Hadoop, Spark, Kafka, etc. Such a cluster consists of several nodes. For more information, refer to HDInsight cluster.

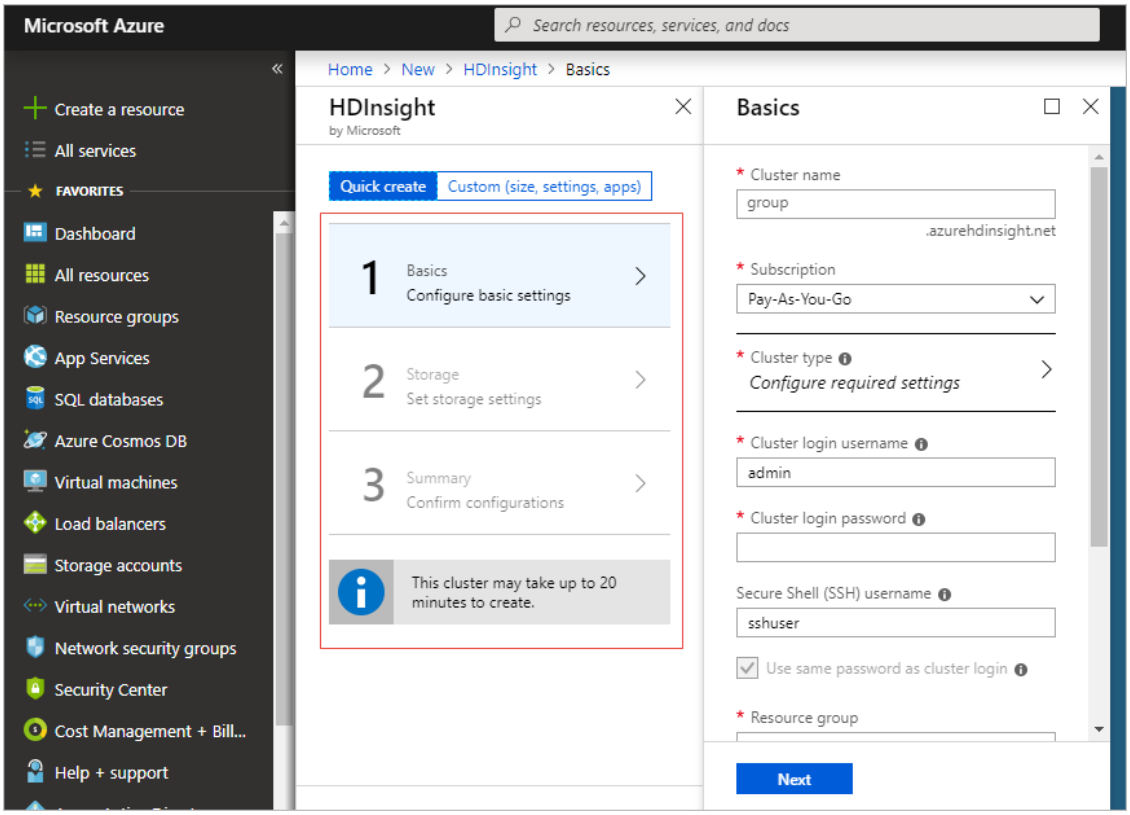
To create a HDInsight cluster in Azure, perform the following steps:

1. Log in to portal.azure.com.
2. Go to **Analytics > HDInsight**.

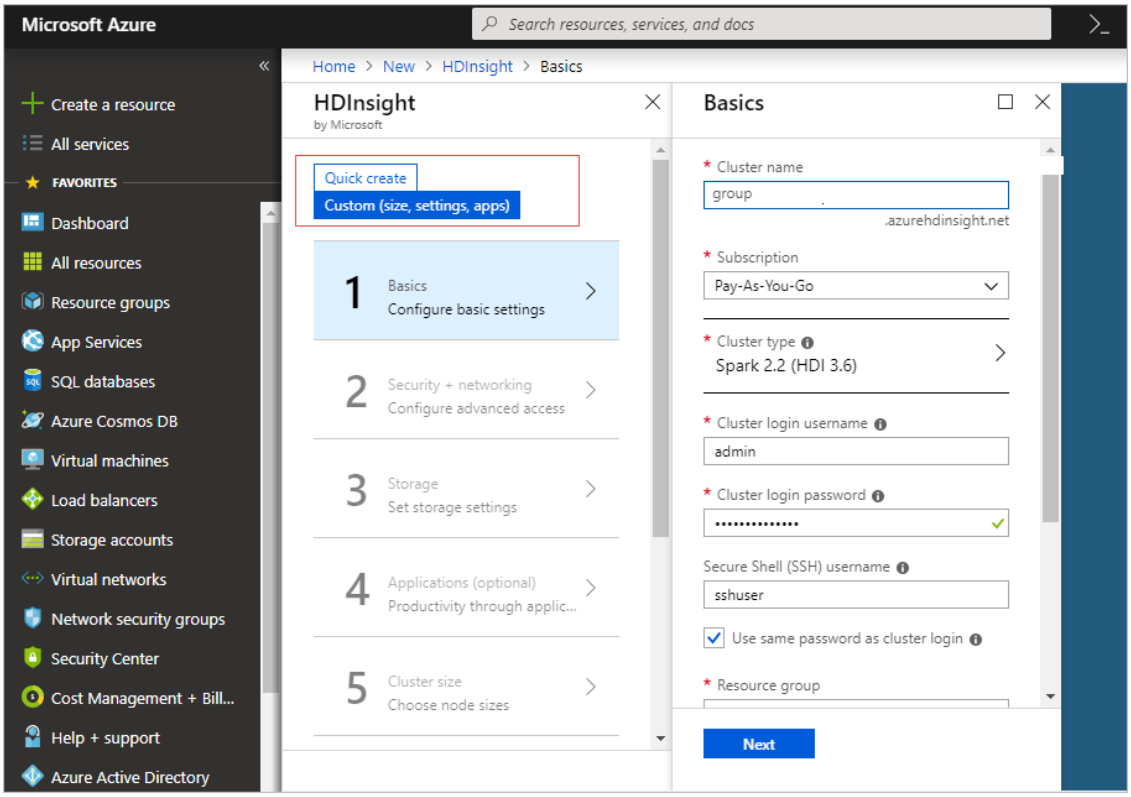


The system displays the following two options to create the HDInsight cluster:

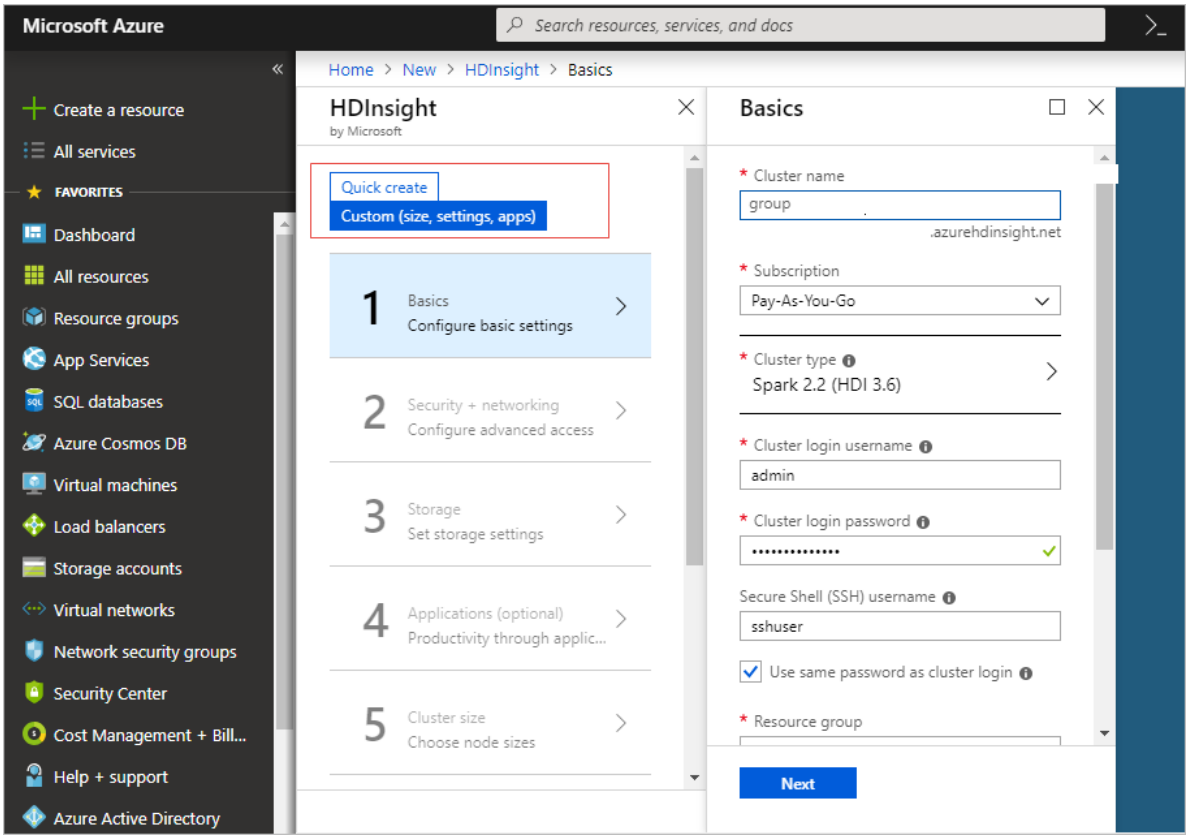
- *Quick create*: This menu item contains limited configuration options to configure the HDInsight cluster.



- *Custom*. This menu item contains detailed configuration options to configure the HDInsight cluster.

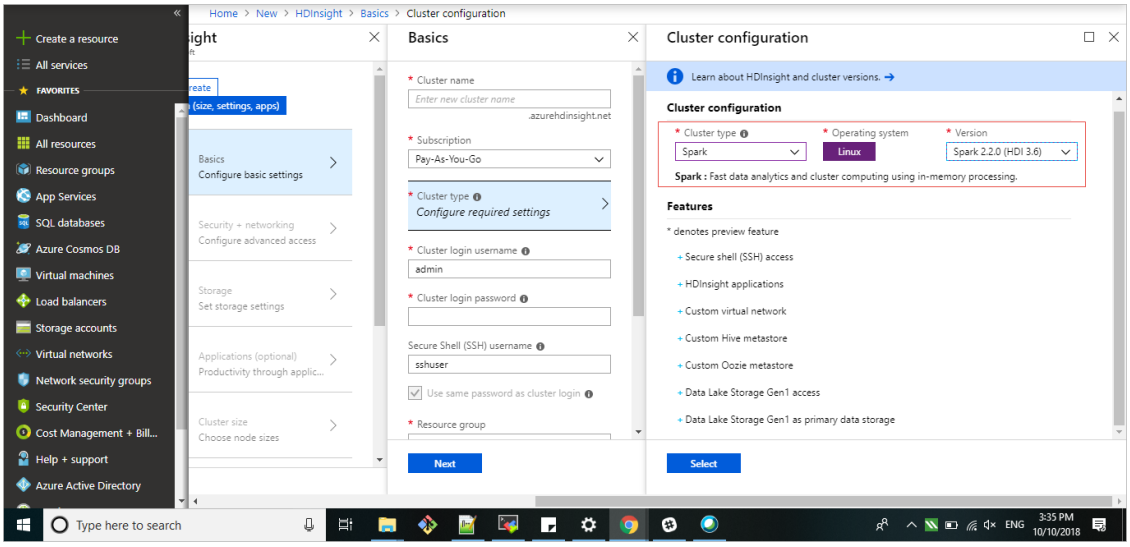


3. Under the **Basic** tab, perform the following steps:



(a) In **Cluster name**, specify a name for the HDInsight cluster that you want to create.

(b) In **Cluster type**, select Spark, the operating system, and the Spark version. Once you select Spark, the dependent components, such as Hadoop are automatically added to the environment.



(c) In **Cluster login username**, specify the default user to access the Azure portal.

(d) In **Cluster login password**, specify the password for default user account, to access the Azure portal.

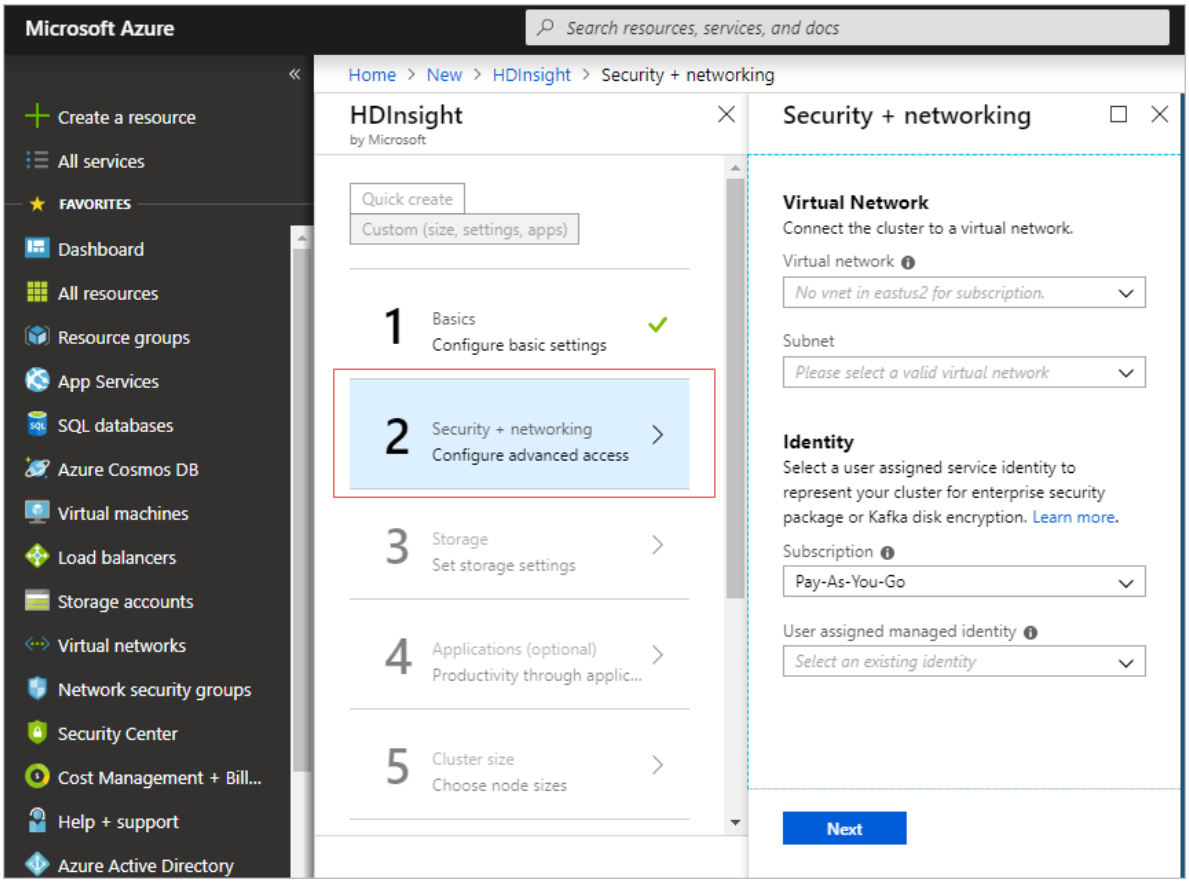
(e) In **Secure Shell username**, specify the username for the user that can SSH to the cluster by using CLI.

(f) To use the same password for both SSH as well as accessing the Azure portal, select the **Use same pass-**

word as cluster login option.

(g) In **Resource group**, select a resource group that manages Azure resources for your projects. If there is no existing resource group, access the **Resource group** tab on the left pane and create one.

4. Under the **Security + networking** tab, perform the following steps:

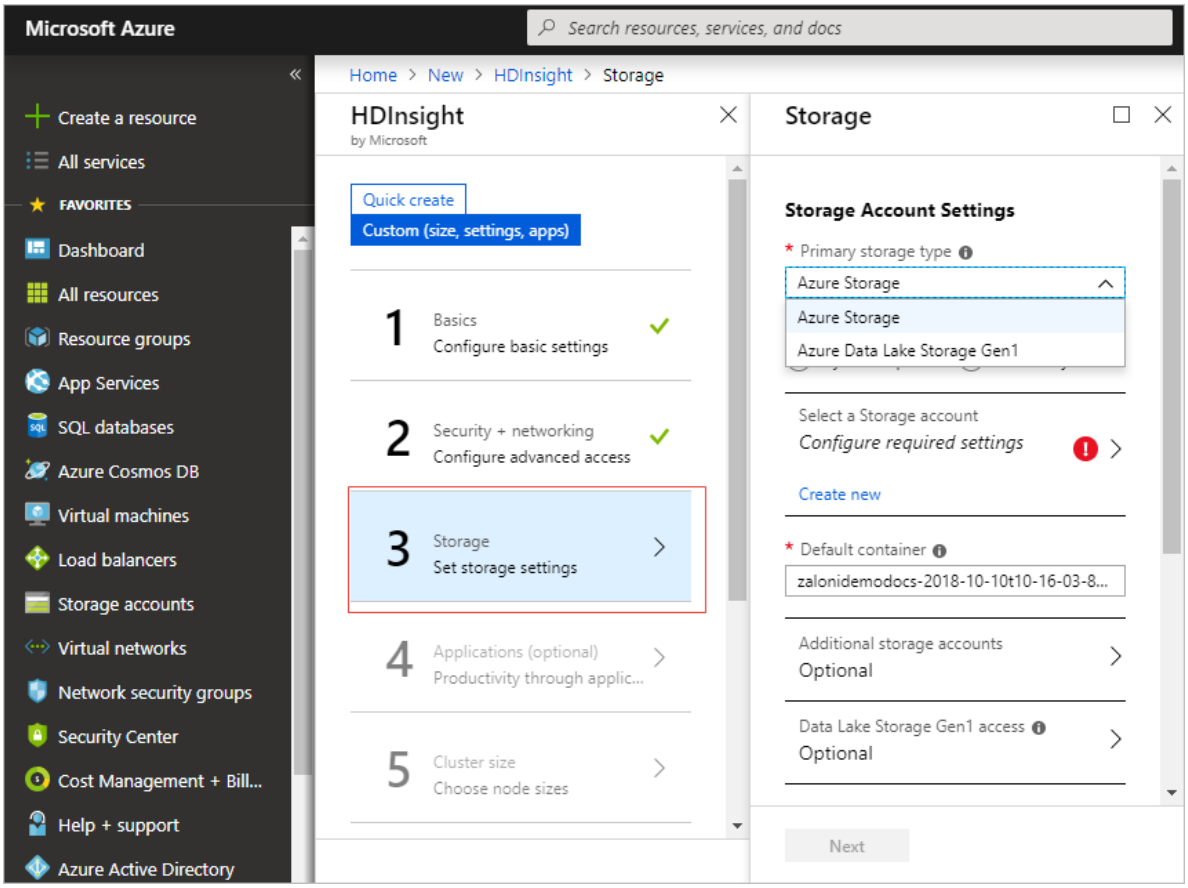


(a) In **Virtual network**, select the name of your virtual network. If you do not specify any option or there is no existing virtual network, the system creates one at runtime. Alternatively, access the **Virtual networks** tab on the left pane and create one.

(b) In **Subnet**, select the name of your subnet. If you do not specify any option or there is no existing subnet, the system creates one at runtime.

(c) In **Subscription**, select a subscription model for your cluster. For example, *Pay-As-You-Go*.

5. Under the **Storage** tab, perform the following steps:



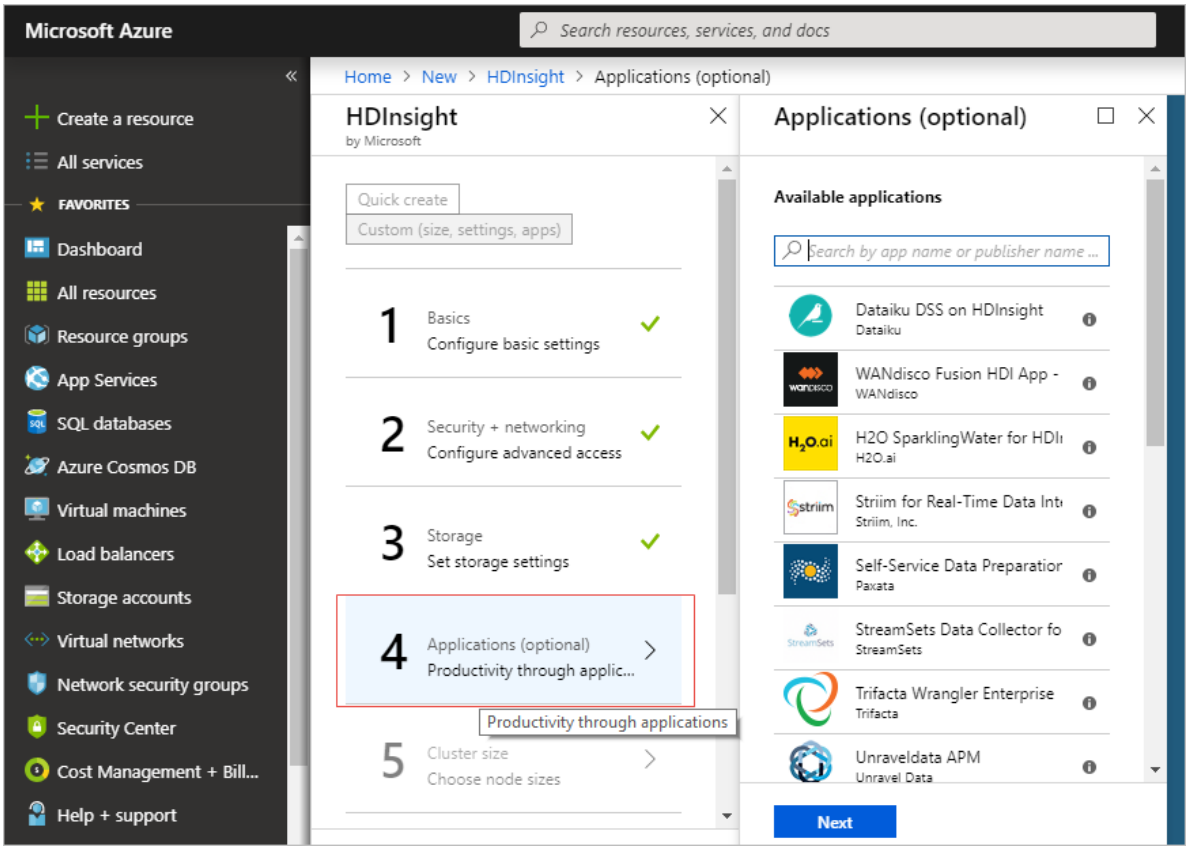
(a) In **Primary storage type**, select either *Azure Storage* or *Azure Data Lake Storage Gen1*. *Azure Storage* indicates the Azure Blob storage, while *Azure Data Lake Storage Gen1* indicates ADLS. For more information on using either of these mechanisms with ZDP, refer to the *Using ZDP with HDInsight on ADLS and Azure Blob Storage* section.

(b) Select a storage account. If there is no existing storage account, create an account.

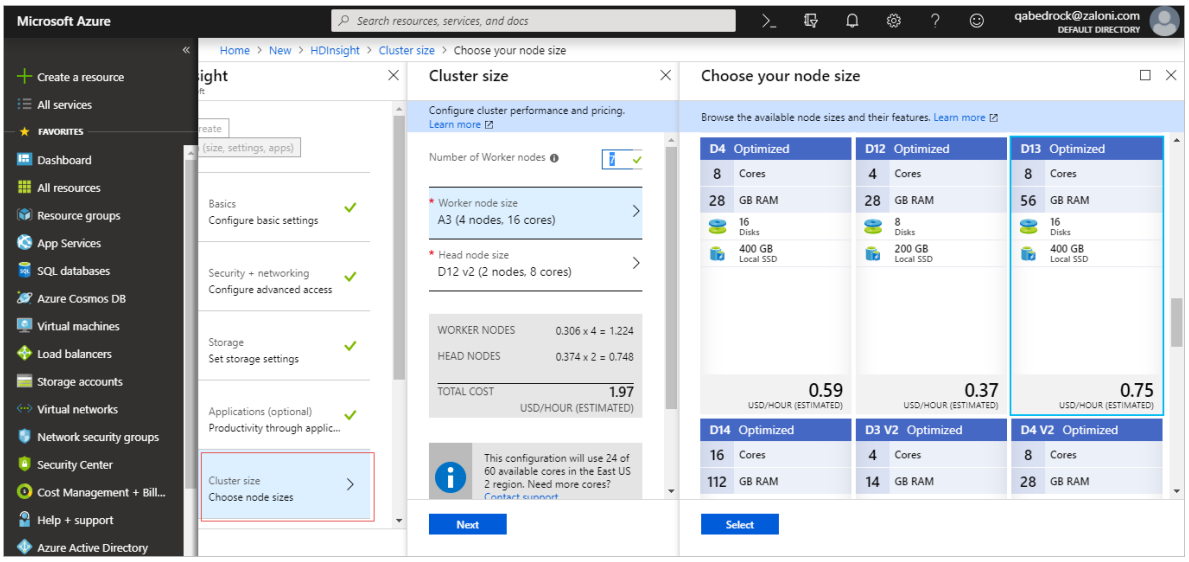
(c) In **Default container**, select the root directory for the cluster storage. Note that the default container contains application and system logs. Ensure that you retrieve the logs before deleting the container.

(d) Configure the additional properties, as appropriate.

6. Under the **Applications** tab, select all the third-party applications that you want to use in the cluster. All the supported applications are listed. ZDP does not have any specific recommendations for using any third-party application.



7. Under the **Cluster size** tab, perform the following steps:



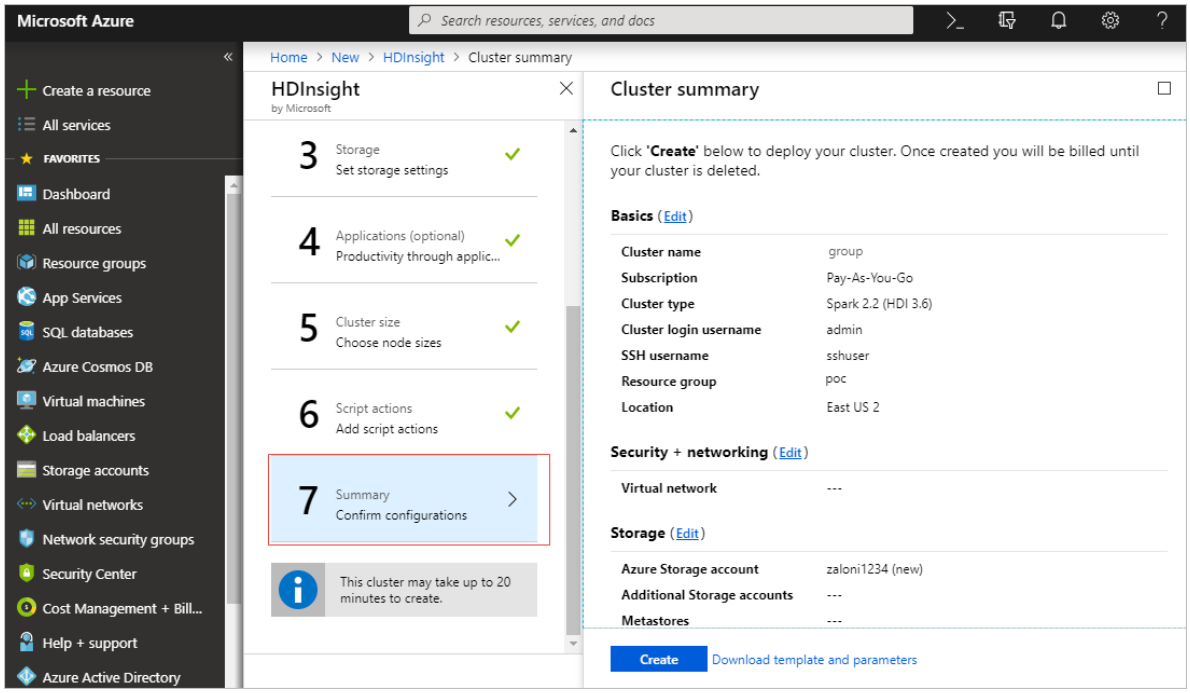
(a) Specify the number of worker nodes. Worker nodes are the data nodes. Based on the usage of your cluster, define the number of worker nodes to be present in the cluster.

(b) From the pre-defined list of available worker node configurations, select an option that best suits your business needs. Based on your selection of the worker node configuration, the system displays the amount incurred (based on the usage). Note that you must have the same configuration for all the worker nodes in the same cluster.

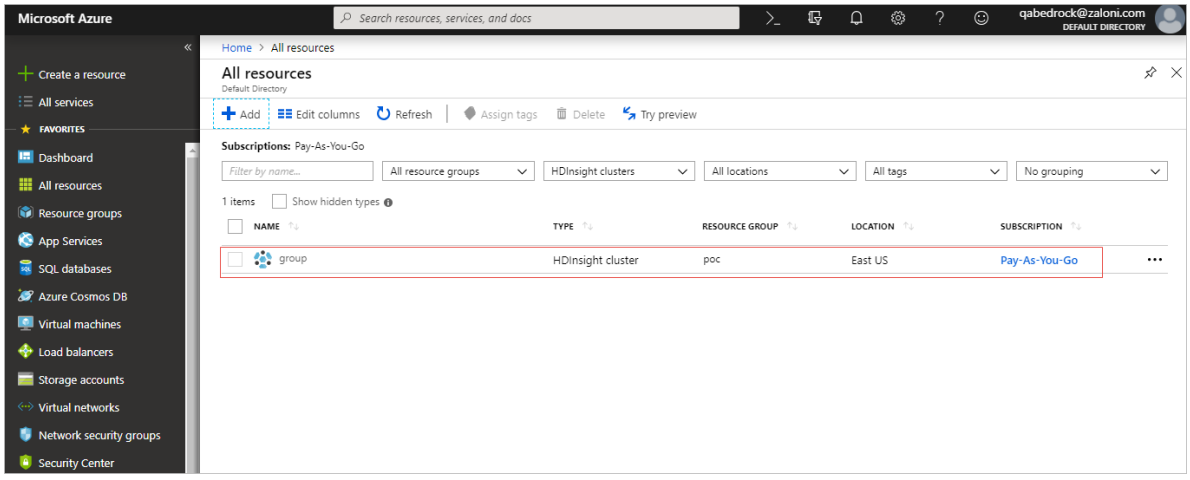
(c) From the pre-defined list of available head node configurations, select an option that best suits your business needs. Based on your selection of the head node configuration, the system displays the amount incurred (based on the usage). Any HDInsight cluster has two head nodes and it is not configurable.

8. Under the **Script actions** tab, specify any custom scripts (for example, Python/Shell) that you want to run on the cluster components (for example, worker node, head node, Zookeeper, etc). ZDP does not have any specific recommendations for using scripts.

9. Under the **Summary** tab, verify if all the displayed details are accurate.



10. Click **Create** to create the cluster. The newly created cluster is displayed in the HDInsight clusters listing page.



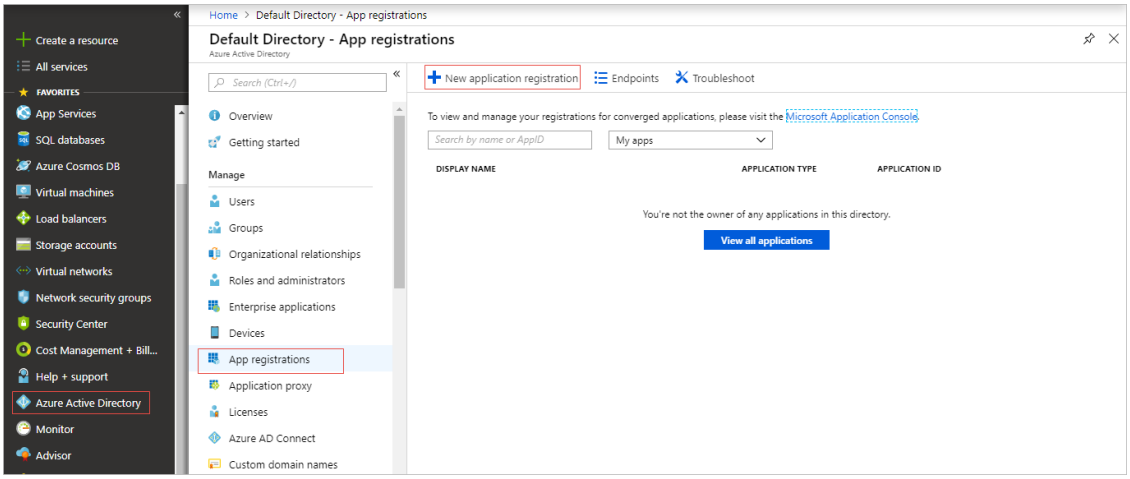
1.10.1 Using ZDP with HDInsight on ADLS and Azure Blob Storage

To use ZDP with HDInsight on ADLS, perform the following steps:

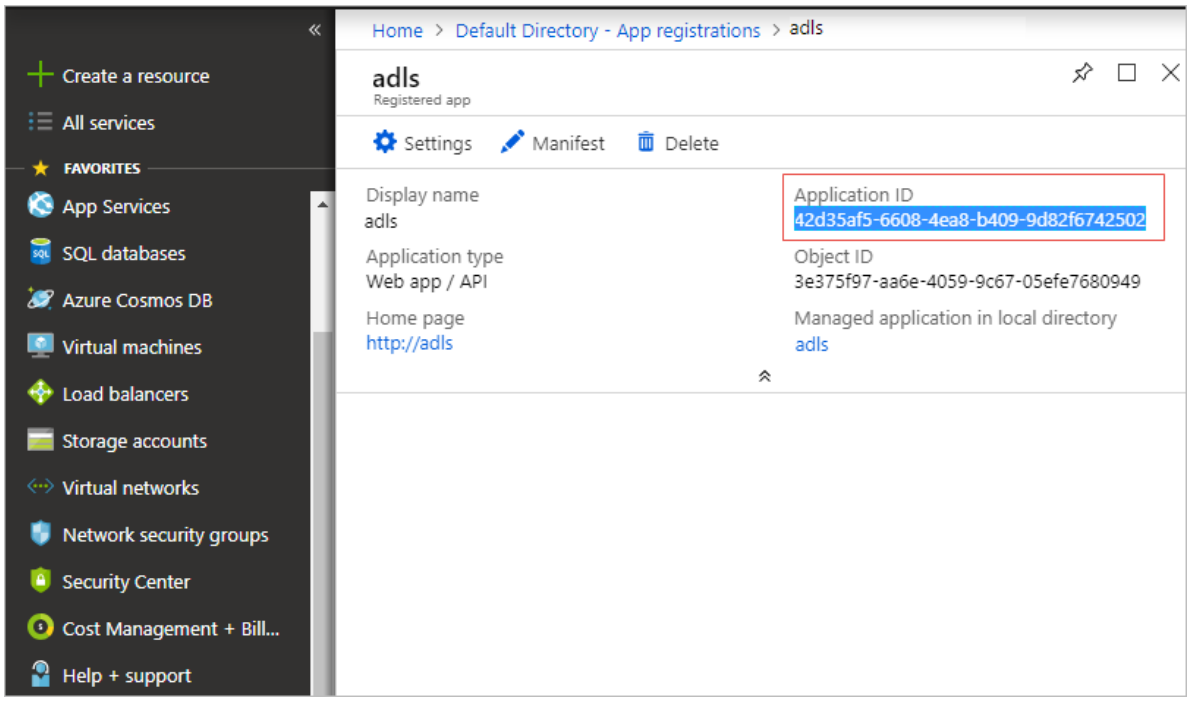
1. To let ZDP connect with ADLS, create an application in the Azure console. To do so, perform the following steps:

   (a) Log in to portal.azure.com.

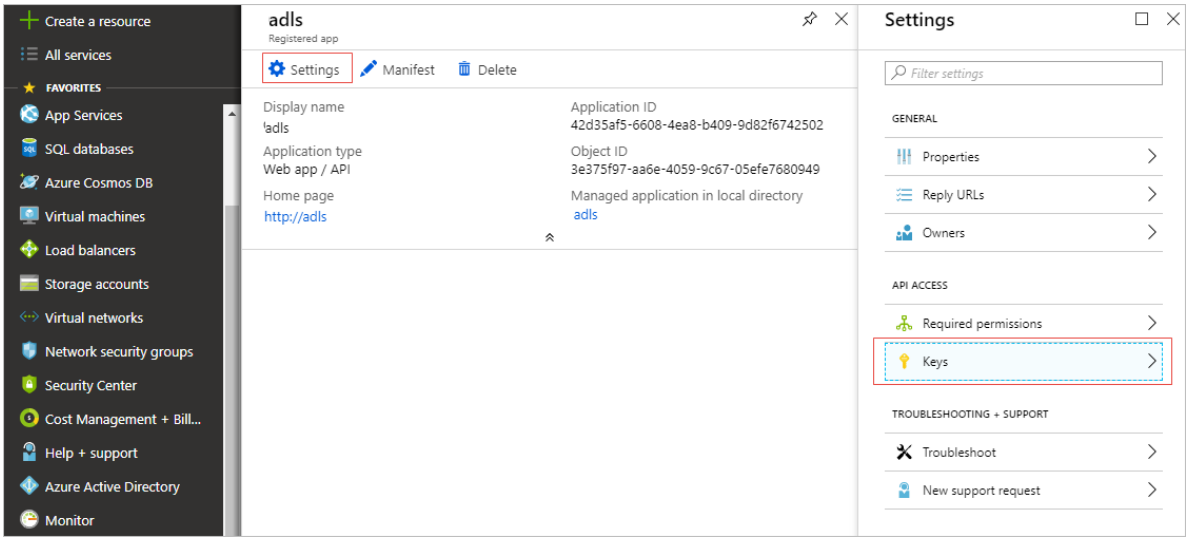
   (b) Navigate to **Azure Active Directory > App registration** and click **New application registration** to create an application.



2. Once you have created the application, access the application to view the details. Copy the value for the **Application ID** property and set the same for the *dfs.adls.oauth2.client.id* parameter (while creating the HDFS connection in ZDP).



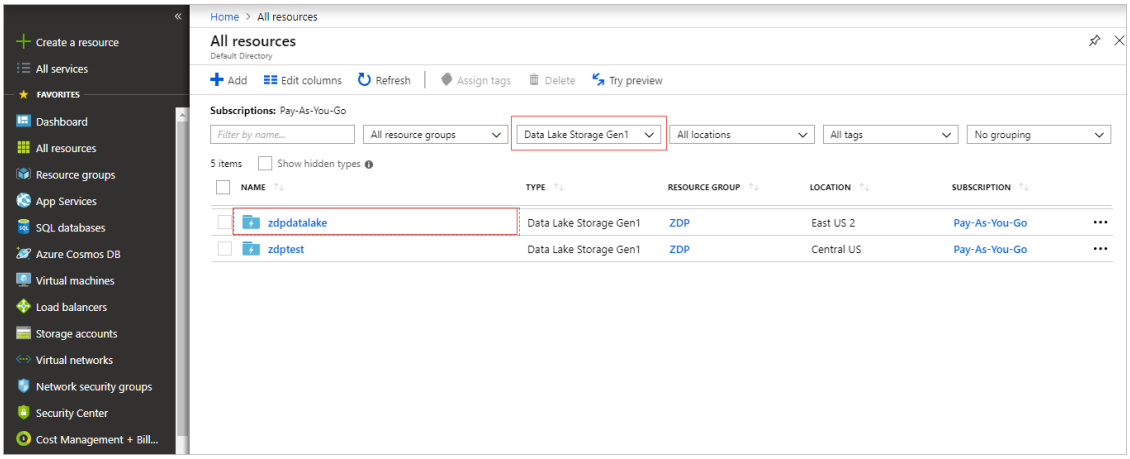
3. Go to **Settings > Keys**.



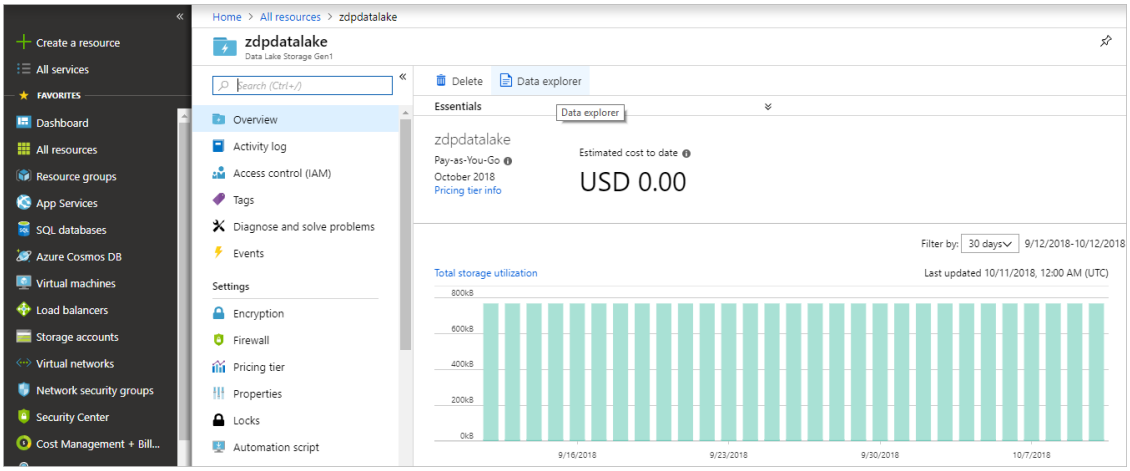
4. Specify a name for the key and select the expiry duration for the key. Once you save the key, the system displays the key. Note that you must copy the key for all future reference (as the key cannot be viewed once you navigate away from the window). You must specify the same value for the *dfs.adls.oauth2.credential* parameter (while creating the HDFS connection in ZDP).

5. To register the application with the ADLS resource, perform the following steps:

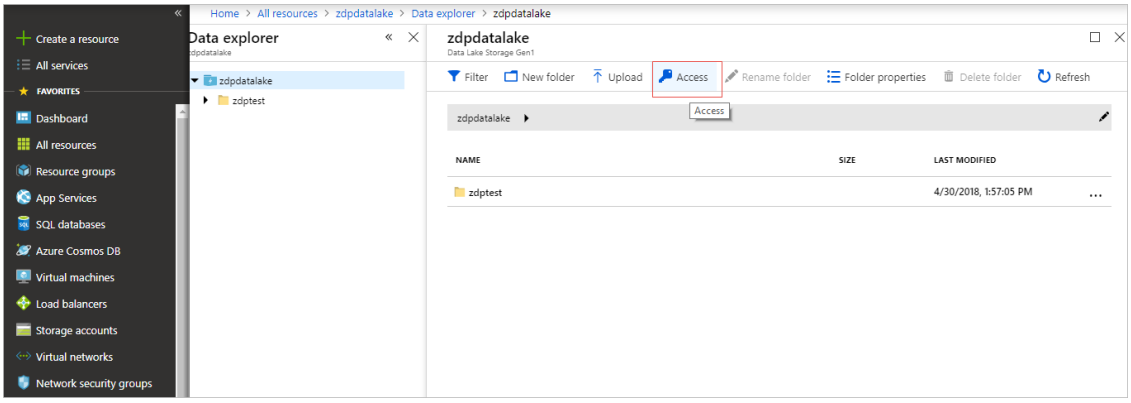
   (a) Access the ADLS resource.



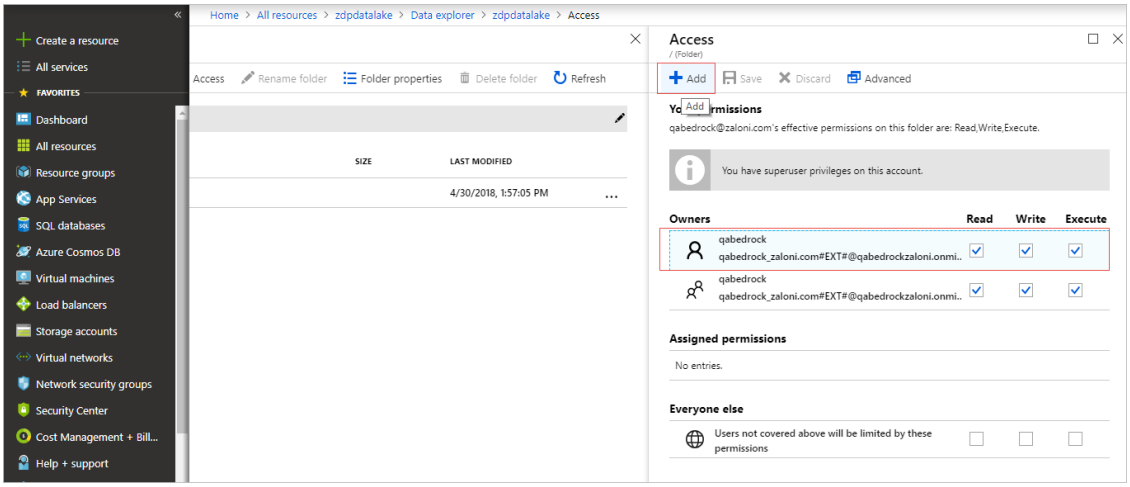
   (b) Go to **Data explorer**.

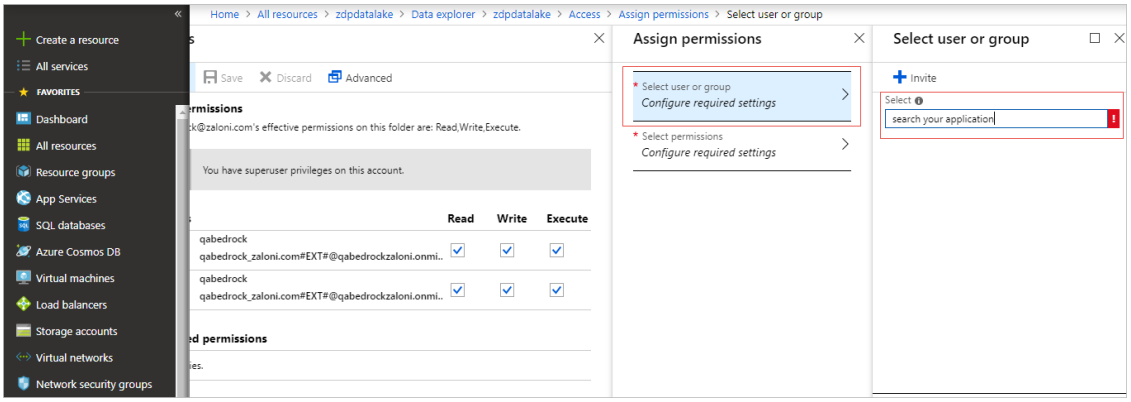


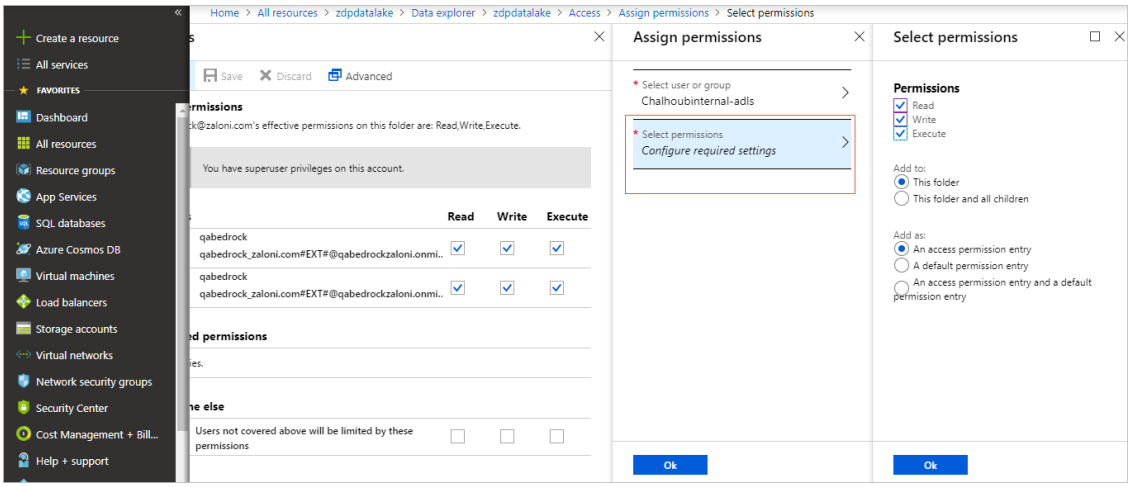
(c) Go to **Access**.



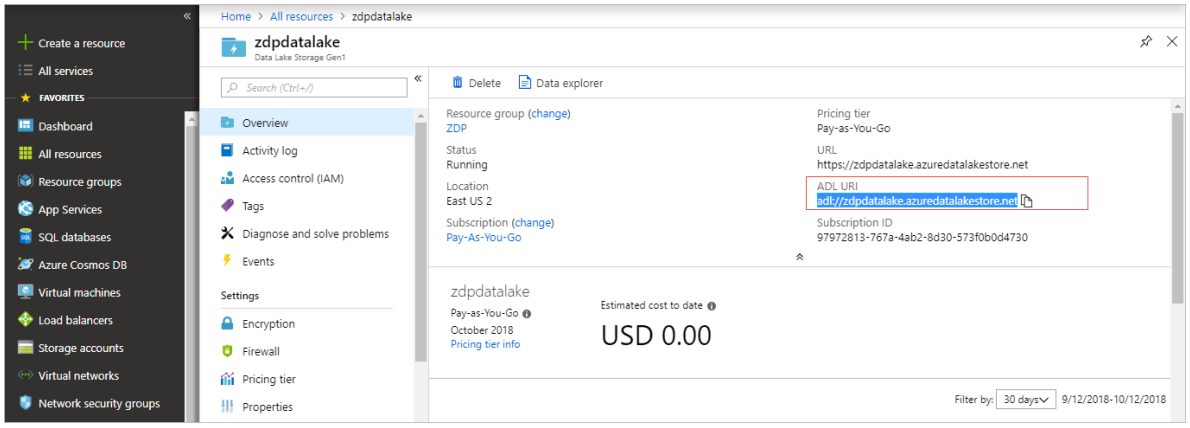
(d) To associate the application with the ADLS resource, click Add and define the owners and permissions for the application. By doing so, you can ensure that ZDP can perform read-write operation on ADLS through the application.







6. Access the overview page for the ADLS and copy the **ADL URL**. You must set the same value for the *File System URI* parameter (while creating the HDFS connection in ZDP).



7. In the ZDP UI, navigate to **Settings > Connections** and create a HDFS connection.

   (a) Specify the value for the **File System URI**. This can be retrieved from the **ADL URL** parameter (available in the ADLS overview page).

   (b) Specify the following values for keys and values:

Property Key	Sample Property Value/Description
`fs.adl.impl`	org.apache.hadoop.fs.adl.AdlFileSystem
`fs.AbstractFileSystem.adl.impl`	org.apache.hadoop.fs.adl.Adl
`dfs.adls.oauth2.access.token.provider.type`	ClientCredential
`dfs.adls.oauth2.refresh.url`	https://login.microsoftonline.com/4d5a9e74-a977-4ab4-a8b3-11f3e790c933/oauth2/token
`dfs.adls.oauth2.client.id`	Set the value for the Application ID available in the Azure portal. Refer to the above steps for more information.
`dfs.adls.oauth2.credential`	Set the value for the key generated in the Azure portal. Refer to the above steps for more information.

8. Define a *BEDROCK* level namespace variable and set the value as the value defined for the **ADL URL** parameter (available in the ADLS overview page).
9. (To perform ingestion) While creating a file pattern, prepend the namespace variable (created above) to the **Destination Path** property. For example, *${BEDROCK.ADLS_HOME_DIRECTORY}/Output_path*.
10. While creating entities in ZDP, prepend the namespace variable (created above) to the **Location** parameter.

To use ZDP with HDInsight on Azure Blob storage, perform the following steps:

1. Access the *core-site.xml* file for your distribution and locate the value for the *fs.azure.account.key.saazcdhdata.blob.core.windows.net* parameter.
2. Use the decrypt.sh script to decrypt the value set for the *fs.azure.account.key.saazcdhdata.blob.core.windows.net* parameter. The location for the decrypt.sh script is defined as the value for the *fs.azure.shellkeyprovider.script* property (in the core-site.xml file).
3. Access the ZDP UI, navigate to **Settings > Connections** and create a HDFS connection.
   (a) Specify the value for the **File System URI**. This can be retrieved from the **ADL URL** parameter (available in the ADLS overview page).
   (b) Specify the following values for keys and values:

Property Key	Sample Property Value/Description
`fs.adl.impl`	org.apache.hadoop.fs.adl.AdlFileSystem
`fs.AbstractFileSystem.adl.impl`	org.apache.hadoop.fs.adl.Adl
`dfs.adls.oauth2.access.token.provider.type`	ClientCredential
`dfs.adls.oauth2.refresh.url`	https://login.microsoftonline.com/4d5a9e74-a977-4ab4-a8b3-11f3e790c933/oauth2/token
`fs.azure.account.key.saazcdhdata.blob.core.windows.net`	Set the decrypted value, generated by executing the decrypt.sh script. Refer to the above steps for more information.

1.11 Running Transformations in Kerberized environment

Important: This is applicable for any features that use *SparkSession*. For example, DME, transformation, provisioning, etc. Here, Spark deploy mode is *cluster*.

1. If you intend to run transformations in a Kerberized environment, add the following configuration property in the *${SPARK_HOME}/conf/spark-defaults.conf* file.

Key	Value
spark.driver.extraJavaOptions	*-Djavax.security.auth.useSubjectCredsOnly=false*
spark.yarn.dist.files	*/etc/hive/conf/hive-site.xml,/etc/hive/conf/hive-env.sh*

2. Add the same properties in the */etc/zdp-spark/zdp-spark.yml* file:

```
spark2:
  home: /opt/cloudera/parcels/SPARK2-2.2.0.cloudera3-1.cdh5.13.3.p0.556753/lib/
↪spark2
  version: 2.2
  master:
      url: yarn
  deploy:
      mode: cluster
  extra:
      env:
      opts: --conf "spark.yarn.dist.files=/etc/hive/conf/hive-site.xml,/etc/hive/
↪conf/hive-env.sh" --conf "spark.driver.extraJavaOptions=-Djavax.security.auth.
↪useSubjectCredsOnly=false"
```

1.12 Running Complex Transformations

To run complex transformations, add the following configuration property in the *${SPARK_HOME}/conf/spark-defaults.conf* file.

Key	Value
spark.sql.crossJoin.enabled	*true*