As more and more organizations are discovering the use of Big Data analytics, interest in platforms that provide storage, computation, and analytical capabilities is booming. Hadoop caters to this need. Oozie fulfills this necessity of scheduler for a Hadoop job by acting as a cron to better analyze data.

Apache Oozie Essentials is sprinkled with examples to help you take your big data knowledge to the next level. You will discover how to write workflows to run your MapReduce, Pig, Hive, and Sqoop scripts and schedule them to run at a specific time or for a specific business requirement using a Coordinator. You will get to grips with how to embed Spark jobs, which can be used to run your machine learning models on Hadoop.

This book will ensure that you are capable of using Oozie to handle large Hadoop Workflows.

Who this book is written for
If you are an expert Hadoop user who wants to use Apache Oozie to handle workflows efficiently, this book is for you. This book will be handy for anyone who is familiar with Hadoop and wants to automate data and machine learning pipelines.

What you will learn from this book
- Install and configure Oozie from source code on your Hadoop cluster
- Dive into the world of Oozie with Java MapReduce jobs
- Schedule Hive ETL and data ingestion jobs
- Import data from a database through Sqoop jobs in HDFS
- Create and process data pipelines with Pig and Hive scripts as per business requirements.
- Run machine learning Spark jobs on Hadoop
- Create quick Oozie jobs using Hue
- Make the most of Oozie's security capabilities by configuring Oozie’s Security Settings.


Jagat Jasjit Singh

Apache Oozie Essentials

Unleash the power of Apache Oozie to create and manage your Big Data and machine learning pipelines in one go.

Free Sample
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Setting up Oozie'
- A synopsis of the book’s content
- More information on Apache Oozie Essentials
Jagat Jasjit Singh works for one of the largest telecom companies in Melbourne, Australia, as a big data architect. He has a total experience of over 10 years and has been working with the Hadoop ecosystem for more than 5 years. He is skilled in Hadoop, Spark, Oozie, Hive, Pig, Scala, machine learning, HBase, Falcon, Kafka, GraphX, Flume, Knox, Sqoop, Mesos, Marathon, Chronos, Openstack, and Java. He has experience of a variety of Australian and European customer implementations. He actively writes on Big Data and IoT technologies on his personal blog (http://jugnu.life). Jugnu (a Punjabi word) is a firefly that glows at night and illuminates the world with its tiny light. Jagat believes in this same philosophy of sharing knowledge to make the world a better place. You can connect with him on LinkedIn at https://au.linkedin.com/in/jagatsingh.
Preface

With the increasing popularity of Big Data in enterprise, every day more and more workloads are being shifted to Hadoop.

To run those regular processing jobs on Hadoop, we need a scheduler that can act as cron for all data pipelines. Oozie plays this role in the Big Data world.

This book introduces you to the world of Oozie using a step-by-step case study-based approach.

What this book covers

Chapter 1, Setting up Oozie, covers how to install and configure Oozie in Hadoop cluster. We will also learn how to install Oozie from the source code.

Chapter 2, My First Oozie Job, covers running a "Hello World" equivalent first Oozie job. It also introduces the concept of Workflow, Coordinator, and Bundles.

Chapter 3, Oozie Fundamentals, introduces the fundamental concepts of control nodes, expression language, web console, and running Oozie jobs from Hue.

Chapter 4, Running MapReduce Jobs, teaches how to run MapReduce jobs from Oozie and explores the concepts of Coordinators, Datasets, and cron-based frequency schedules.

Chapter 5, Running Pig Jobs, teaches how to run Pig jobs from Oozie. We will also cover the concept of parameterization of Datasets and Coordinator controls.

Chapter 6, Running Hive Jobs, introduces how to run Hive jobs and discusses the concepts of parameterization of Coordinator actions.
Preface

Chapter 7, *Running Sqoop Jobs*, shows how to run Sqoop jobs from Oozie and introduces the concept of HCatalog Datasets and EL functions.

Chapter 8, *Running Spark Jobs*, shows how to run Spark jobs. It also introduces the concept of Bundles and how they are used to group a set of Coordinator jobs.

Chapter 9, *Running Oozie in Production*, covers how to package the code for production deployments and how to rerun the jobs that have failed.
Oozie is a workflow scheduler system to run Apache Hadoop jobs. Oozie Workflow jobs are Directed Acyclic Graphs (DAGs) of actions. More information on DAG can be found at https://en.wikipedia.org/wiki/Directed_acyclic_graph. Actions tell what to do in the job. Oozie supports running jobs of various types such as Java, Map-reduce, Pig, Hive, Sqoop, Spark, and Distcp. The output of one action can be consumed by the next action to create a chain sequence.

Oozie has client-server architecture, in which we install the server for storing the jobs and using client we submit our jobs to the server.

In this chapter, we will learn how to install Oozie for learning purpose and in production. For learning purposes, we will build Oozie from the source code, and for production we will use Hadoop distribution by Hortonworks. Throughout the book, we will use Hortonworks single node virtual machine. If you are using a different Hadoop distribution, you should not worry at all. All distribution packages are the same for Oozie software, which is made by the Apache community (http://oozie.apache.org).

After reading this chapter, we will be able to:

- Configure Oozie in Hortonworks distribution using Ambari
- Install Oozie using the source code provided as tar ball by the Apache Oozie website

## Configuring Oozie in Hortonworks distribution

In this section, we will learn how to configure Oozie inside Hortonworks Hadoop distribution using Ambari. We will configure the Oozie server to use a MySQL database instead of the default Derby database to store all job information.
Setting up Oozie

We will use a virtual machine to learn how to configure Oozie in Hortonworks Hadoop distribution. Most of other distributions, such as Cloudera, Pivotal, and so on, have similar steps.

Let's start with the following steps:

1. If you don't have VirtualBox on your machine, then download and install VirtualBox from https://www.virtualbox.org/wiki/Downloads.

2. Download the Hortonworks single node virtual machine from http://hortonworks.com/hdp/downloads/. It will take 1-2 hours depending upon your Internet connection speed.

   ![Import appliance](image)

   It is always good to store the virtual machine images in a common folder. For example, I have folder in my machine such as ~/dev/vm/. It makes virtual machine image management easier.

3. After the download is complete, open the VirtualBox and click on File | Import Appliance:
4. Click on the **Import Appliance** button, browse to the place where you downloaded the virtual machine image, and then click on **Continue**.

5. Wait till the VirtualBox imports the new machine.

6. Once you can see the machine is imported, click on **Start machine** in the virtual machine console.

7. On completion of boot process of the machine, you can log in to the Ambari dashboard by opening the URL `http://127.0.0.1:8080` in your browser.

8. Use the username as well as password as **admin**.

   It will take some time for all services to start up and report their status to Ambari. Once the system has reported the status, all services have a glance at the Ambari console. It is also a good idea to stop the services which we are not using to reduce the load on the system.

9. In the Ambari dashboard, click on the link named **Oozie** on the left side. You can see there are two components for Oozie, **Oozie Server** and **Oozie Client**. Since we are using a single node cluster, we have both the server and client installed on the same machine. In the production environment, you will configure the Oozie server and clients separately on different machines. Using the client, we will submit the jobs to server. Before submitting the job, we will tell where the server is located using the `OOZIE_URL` variable.

   To save time in manually specifying the Oozie server on the client machine every time, you can set the environment variable `OOZIE_URL` in your `bash_profile` or environment file depending on the operating system you use. You should say `export OOZIE_URL=http://oozieserver:11000/oozie;` in this book `oozieserver` will be localhost.
10. Now click on the **Config link** at the top and we will configure the database as MySQL. The Oozie server will use MySQL to store the job information:

![Ambari Oozie configuration](image)

11. You may notice, at this moment, the server has been configured to use a Derby database. Derby is good for playing and testing, but not for running the production server. We will configure it to use a MySQL-based database.

12. Log in to the virtual machine using SSH as follows:

   ```bash
   $ ssh root@127.0.0.1 -p 2222
   
   The default password is `hadoop`.
   
13. After you log in to the SSH session, log in to MySQL:

   ```bash
   $ mysql -u root
   
   Since this is a test virtual machine, the password is not configured. In production, you will be having password protection.
   
14. At the MySQL prompt, execute the following SQL statements:

   ```sql
   CREATE USER 'oozie'@'%' IDENTIFIED BY 'hadoop';
   CREATE DATABASE oozie;
   GRANT ALL PRIVILEGES ON oozie.* TO 'oozie'@'%' WITH GRANT OPTION;
   ```
16. To make Oozie work with MySQL, we need to get driver for it. Let's download the MySQL JDBC driver from the MySQL JDBC jar download section. Extract the jar to a folder such as /root/mysql inside the virtual machine:

```
$ cd -
$ mkdir mysql
$ cd mysql
$ # Download the MySQL JDBC Driver
$ wget http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.36.tar.gz
$ # Extract tar
$ tar -xvf mysql-connector-java-5.1.36.tar.gz
$ # Tell Ambari that we got new MYSQL JDBC driver which it can use
$ ambari-server setup --jdbc-db=mysql --jdbc-driver=/root/mysql/mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar
```
17. In the Ambari dashboard, configure the MySQL database with the following details:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Name</td>
<td>oozie</td>
</tr>
<tr>
<td>Database Username</td>
<td>oozie</td>
</tr>
<tr>
<td>Database Password</td>
<td>hadoop</td>
</tr>
<tr>
<td>JDBC Driver Class</td>
<td>com.mysql.jdbc.Driver</td>
</tr>
<tr>
<td>JDBC Database URL</td>
<td>jdbc:mysql://localhost:3306/${oozie.db.schema.name}?createDatabaseIfNotExist=true</td>
</tr>
</tbody>
</table>

18. In the Ambari dashboard page, click on Test Connection. If all is good, there should be a green tick. So, we have now configured the Oozie server to use MySQL database instead of Derby.

19. Finally, to confirm that Oozie works properly, in another browser tab open the Oozie dashboard by entering the URL http://127.0.0.1:11000/oozie.

This completes the first section in which we learned how to configure Oozie for Hortonworks Ambari distribution.

**Installing Oozie using tar ball**

In this section, we will learn how to build and install Oozie from the source code. Since Hortonworks virtual machine had already Oozie installed, we did not need to do anything.

Just to learn how to install Oozie from tar ball in this section, we will use a Vagrant-based machine in which we will configure and install the Oozie server.

The summary of the steps we will perform is as follows:

1. Create a test build machine.
2. Download and build the Oozie code to make a WAR file.
3. Download the Oozie third-party dependency jars and libraries.
4. Package the Oozie WAR file and its dependencies into a WAR file.
5. Configure the MySQL database for the Oozie server.
6. Configure the shared library.
7. Start and test the Oozie server.
Just as a heads-up, the vagrant machine needs lot of resources to build the code. So, if you do not have a powerful machine, you can build it directly on your host operating system rather than the virtual machine. I am working on a MacBook Pro, which has a 16 GB RAM. I gave 8 GB to the virtual machine to show how to install Oozie from source.

Creating a test virtual machine
The following are the steps to create a test virtual machine:

1. Download latest Oozie distribution from the Apache Oozie website. Go to the downloads section and download the latest version (4.2.0 at time of writing) in machine where you want to install it.

2. Download and install Vagrant depending upon your operating system:

3. After this, go to the VirtualBox website. Depending on your computer operating system, download and install the VirtualBox.

4. If you already have a test machine that has a Linux-based operating system, then you can skip the Vagrant-based setup and follow the steps for building Oozie from scripts.

6. Create a folder in your system called dev, or any suitable location where we can clone code. We will call the dev/apache_oozie_essentials location as <BOOK_CODE_HOME> in this book. The following are the commands to do this:

   $ git clone https://github.com/jagatsingh/apache_oozie_essentials.git
   $ cd <BOOK_CODE_HOME>
   $ cd learn_oozie/ch01
   $ # Let's start the virtual machine
   $ vagrant up

7. Wait for some time till our new test machine comes up.

   Here is what Vagrant does behind the scene:
   - Gets the image of the Centos 6.5 operating system
   - Installs JDK, MySQL, Git, and Maven

8. All the preceding steps are being done by the provider script, which is shown as follows:

   $ sudo yum install -y java-1.7.0-openjdk mysql-server git unzip zip apache-maven telnet
   $ cp /vagrant/files/maven/settings.xml /etc/maven/
   $ sudo service mysqld start
9. When the machine starts off completely, you will see something, as shown in the following figure:

![Vagrant up finish](image)

If you need a quick tutorial on how Vagrant works, then read the documentation on Vagrant at https://docs.vagrantup.com/v2/.

10. Now we can log in to the virtual machine by using the command `vagrant ssh`. This command should be executed from the folder `ch01`.

11. Inside the Vagrant virtual machine, `mount/vagrant` is same as the `ch01` folder, placed at `<BOOK_CODE_HOME>/learn_oozie/`, from where we started the Vagrant.

```bash
$ cd /vagrant

$ ls
```
Setting up Oozie

Building Oozie source code

Let’s build Oozie from the source code. We will download the latest Oozie distribution and build it. All of these steps are present in the script `build_oozie.sh` placed at `cat/vagrant/scripts/`.

The contents of the script which we will run is as follows:

```
# Download and make Oozie distribution
$ cd ~/
$ mkdir {oozie_build,oozie_install,hadoop_install}
$ cd oozie_build
$ wget http://apache.mirror.digitalpacific.com.au/oozie/4.2.0/oozie-4.2.0.tar.gz
$ tar -xvf oozie-4.2.0.tar.gz
$ cd oozie-4.2.0
$ bin/mkdistro.sh -DskipTests -P hadoop-2
```

Summary of the build script

In the `oozie_build` directory, we will build Oozie. In the `oozie_install` directory, we will install Oozie. In the `hadoop_install` directory, we will download Hadoop distribution and copy few jars needed for Oozie to run. You can also download the jars from your own hadoop cluster.

Let’s run the command to start the Oozie build. It will take some time to download all the dependencies and build the source code:

```
$ /vagrant/scripts/build_oozie.sh
```

If you already have a Maven repository on your host machine and want to to avoid downloading maven artifacts again, then look at the Maven settings file. I have configured (and commented) it to use my MacBook home maven as I already had all the artifacts there. You can uncomment that if you want to do something similar.
Codehaus Maven move

Codehaus no longer serves up Maven repositories, we need to configure Maven to download those dependencies from a different location. If you look at `/etc/maven/settings.xml`, which came with this machine, it has already been modified. You can see the details about it on the Codehaus website at http://www.codehaus.org/mechanics/maven/.

On a successful build, you should see something like the following screenshot:

```
Chapter 1

Codehaus Maven move

On a successful build, you should see something like the following screenshot:

```

Download dependency jars

To run Oozie properly, the Oozie WAR file needs to have some dependencies packaged with it. Some of them are Hadoop, MySQL JDBC driver, Ext-js, and so on. The MySQL JDBC driver is used by the server database, and Ext-js is used by the Oozie web console.

We will copy all of them in to one folder, `libext`, and then use the `oozie-setup.sh` command to build the WAR file.
Setting up Oozie

Let's download the Hadoop jars from your cluster or by executing the following steps:

```
$ cd ~/hadoop_install
$ wget https://archive.apache.org/dist/hadoop/common/hadoop-2.4.0/hadoop-2.4.0.tar.gz
$ tar -xvf hadoop-2.4.0.tar.gz
```

Now we should have Hadoop extracted to the folder `~/hadoop_install`

The preceding steps can be executed in one go using the following command:

`/vagrant/scripts/download_hadoop_jars.sh`

Preparing to create a WAR file

To create the WAR file, we need to copy the Oozie distro built earlier and combine it with the jars for Hadoop, the MySQL JDBC driver, and the Ext-js library.

If you remember from the previous Ambari Oozie configuration, we used MySQL as our database and configured it using the `ambari-setup` command. We will take a similar approach for the MySQL JDBC driver jar, which we are providing by merging it with the Oozie WAR file.

Let's prepare the Oozie distro using the following commands:

```
# Prepare to make Oozie war file
$ cd ~/oozie_install
$ cp ~/oozie_build/oozie-4.2.0/distro/target/oozie-4.2.0-distro.tar.gz ~/oozie_install
$ tar -xvf oozie-4.2.0-distro.tar.gz
$ cd oozie-4.2.0
$ # Removing hsql jar as they cause class conflict
$ rm lib/hsqldb-1.8.0.10.jar
```

Download the MySQL jar using the following commands:

```
# Collect all external jar files
$ mkdir libext
$ wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.36.tar.gz --no-check-certificate
$ tar -xvf mysql-connector-java-5.1.36.tar.gz
$ # Copy MySQL JDBC Driver
$ cp mysql-connector-java-5.1.36/*.jar libext/
```
Merge the Hadoop jars and the ext-js library using the following commands:

```bash
$ cd libext
$ wget http://dev.sencha.com/deploy/ext-2.2.zip
$ # Collect hadoop related jars
$ shopt -s globstar
$ /bin/cp -rf ~/hadoop_install/hadoop-2.4.0/share/**/*.jar ~/oozie_install/oozie-4.2.0/libext
$ # Removing source jars to reduce size
$ rm -rf *sources*
$ rm -rf *jasper*
```

All of the preceding steps can be executed in one go using the following command:

```bash
/vagrant/scripts/war_file_preparation.sh
```

After successful execution, go to `/home/vagrant/oozie_install/oozie-4.2.0/libext` and see that we now have jars placed in the folder.

**Create a WAR file**

Now we need to package the `oozie-distro` and jars that we copied in to the `libext` folder as a single packaged WAR file. This WAR file will be deployed in tomcat by going to the folder `/home/vagrant/oozie_install/oozie-4.2.0` and executing the following command:

```bash
bin/oozie-setup.sh prepare-war
```

The command completes with a WAR file being created in the folder, as shown in the following screenshot:

```
INFO: Adding extension: /home/vagrant/oozie_install/oozie-4.2.0/libext/toocat-cynto.jar
INFO: Adding extension: /home/vagrant/oozie_install/oozie-4.2.0/libext/toocat-dcpp.jar
INFO: Adding extension: /home/vagrant/oozie_install/oozie-4.2.0/libexec/toocat-ibn-es.jar
INFO: Adding extension: /home/vagrant/oozie_install/oozie-4.2.0/libexec/toocat-ibn-fr.jar
INFO: Adding extension: /home/vagrant/oozie_install/oozie-4.2.0/libexec/toocat-ibn-ja.jar
INFO: Adding extension: /home/vagrant/oozie_install/oozie-4.2.0/libexec/xmlenc-1.2.jar
INFO: Adding extension: /home/vagrant/oozie_install/oozie-4.2.0/libexec/xmlenc-1.9.jar
INFO: Adding extension: /home/vagrant/oozie_install/oozie-4.2.0/libexec/zookeeper-3.4.5.jar

New Oozie WAR file with added 'ExtJS library, JARs' at /home/vagrant/oozie_install/oozie-4.2.0/oozie-server/webapps/oozie.war

INFO: Oozie is ready to be started
[vagrant@localhost oozie-4.2.0]$ pwd
/home/vagrant/oozie_install/oozie-4.2.0
[vagrant@localhost oozie-4.2.0]$ bin/oozie-setup.sh prepare-war
```

Prepare a WAR file
Exercise: Execute bin/oozie-setup.sh help and read all the commands possible with the setup command.

Configure Oozie MySQL database

If you remember, we configured Ambari Oozie to use MySQL database for Oozie. We will do the same for this instance of the Oozie server.

At the Mysql prompt, execute the following:

```$ mysql -u root
CREATE USER 'oozie'@'%' IDENTIFIED BY 'hadoop';
CREATE DATABASE oozie;
GRANT ALL PRIVILEGES ON oozie.* TO 'oozie'@'%' WITH GRANT OPTION;
```

This will create the Oozie database, which will be used by the server.

Go to /home/vagrant/oozie_install/oozie-4.2.0/conf and open the oozie-site.xml file. In this file, all the Oozie settings are declared. All the Oozie configuration properties and their default values are defined in the oozie-default.xml file.

Oozie resolves configuration property values in the following order.

If a Java System property is defined, it uses its value, else if the Oozie configuration file (oozie-site.xml) contains the property, it uses its value, else it uses the default value documented in the oozie-default.xml file.

Oozie does not use the oozie-default.xml file found in the conf/ directory. It is there for reference purposes only.

Let's edit the oozie-site.xml and configure the database details. You can use the vi editor or copy the settings from the already created file using the following command:

```$ cp /vagrant/files/oozie/oozie-site.xml /home/vagrant/oozie_install/oozie-4.2.0/conf/
```

If you want to edit it manually, then add the following code:

```xml
<property>
  <name>oozie.service.JPAService.jdbc.driver</name>
  <value>com.mysql.jdbc.Driver</value>
</property>
```
<description>JDBC driver class</description>
</property>

<property>
  <name>oozie.service.JPAService.jdbc.url</name>
  <value>jdbc:mysql://localhost:3306/${oozie.db.schema.name}?createDatabaseIfNotExists=true</value>
  <description>JDBC URL</description>
</property>

<property>
  <name>oozie.service.JPAService.jdbc.username</name>
  <value>oozie</value>
  <description>DB user name</description>
</property>

<property>
  <name>oozie.service.JPAService.jdbc.password</name>
  <value>hadoop</value>
  <description>DB user password</description>
</property>

Exercise: Execute `bin/ooziedb.sh` help and read all the commands possible with the `setup` command.

Let's create the database tables in our newly created database using the following command:

```
bin/ooziedb.sh create -sqlfile oozie.sql -run
```

The following screenshot shows the output generated:

```
[vagrant@localhost oozie-4.2.0]$ bin/ooziedb.sh create -run
  setting CATALINA_OPTS="-Xmx1024m"

Validate DB Connection
DONE
DB schema does not exist
Check O0ZIE_SYS table does not exist
DONE
Create SQL schema
DONE
Create O0ZIE_SYS table
DONE

Oozie DB has been created for Oozie version '4.2.0'

The SQL commands have been written to: /tmp/ooziedb-6751686195906463768.sql
```

Database creation success
Setting up Oozie

Configure the shared library

We just need to tell Oozie about the shared libraries before starting the Oozie server. The Oozie sharelib.tar.gz file bundled with the distribution contains the necessary files to run Oozie Map-reduce streaming, Pig, Hive, Sqoop, Hcatalog, and Distcp actions.

Let's execute the following command:

```
bin/oozie-setup.sh sharelib create -fs oozie-sharelib-4.2.0.tar.gz
```

The following screenshot shows the output generated:

```
Create a shared library
```

Start server testing and verification

The following command is used to start the server:

```
bin/oozied.sh start
```

**Exercise:** Execute `bin/oozied.sh` help and read all the commands possible with the `setup` command.

The command, on successful completion, will not print any error message. We can check the status of Oozie server using the following command:

```
bin/oozie admin -oozie http://localhost:11000/oozie -status
```

The output should be:

```
system mode: NORMAL
```

We can also check the Oozie web console by opening the URL `http://localhost:11000/oozie`.
Summary
We started this chapter with the configuration of Oozie inside the Hortonworks virtual machine. We learned how to configure the database for Oozie. Then we started building Oozie from the source code. We packaged the WAR file and also configured the MySQL database.

This completes the installation for the Oozie server.

In the next chapter, we will run our first Oozie job. We will learn how to run Hadoop filesystem commands in Oozie. We will also install Hue and create our Workflow using the editor provided by it.
Where to buy this book

You can buy Apache Oozie Essentials from the Packt Publishing website. Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

[Click here](http://www.PacktPub.com) for ordering and shipping details.