

The next two properties, `mapreduce.jobhistory.address` and `mapreduce.jobhistory.webapp.address`, may seem similar but have some subtle differences. The `mapreduce.jobhistory.address` property is the host and port where the MapReduce application will send job history via its own internal protocol. The `mapreduce.jobhistory.webapp.address` is where an administrator or a user can view the details of MapReduce jobs that have completed.

Finally, we specify a property for a MapReduce staging directory. When a MapReduce job is submitted to YARN, the MapReduce ApplicationMaster will create temporary data in HDFS for the job and will need a staging area for this data. The property `yarn.app.mapreduce.am.staging-dir` is where we designate such a directory in HDFS (i.e., `/mapred`).

This staging area will also be used by the job history server. The installation script will make sure that the proper permissions and subdirectories are created (i.e., `/mapred/history/done_intermediate`).

yarn-site.xml

The final configuration file is `yarn-site.xml`. The script sets the following values:

```
mapreduce.shuffle --> yarn.nodemanager.aux-services
org.apache.hadoop.mapred.ShuffleHandler --> yarn.nodemanager.aux-
  services.mapreduce.shuffle.class
$yarn_proxy:$YARN_PROXY_PORT --> yarn.web-proxy.address
$rmgr:8030 --> yarn.resourcemanager.scheduler.address
$rmgr:8031 --> yarn.resourcemanager.resource-tracker.address
$rmgr:8032 --> yarn.resourcemanager.address
$rmgr:8033 --> yarn.resourcemanager.admin.address
$rmgr:8088 --> yarn.resourcemanager.webapp.address
```

The `yarn.nodemanager.aux-services` property tells the NodeManager that a MapReduce container will have to do a shuffle from the map tasks to the reduce tasks. Previously, the shuffle step was part of the monolithic MapReduce TaskTracker. With YARN, the shuffle is an auxiliary service and must be set in the configuration file. In addition, the `yarn.nodemanager.aux-services.mapreduce.shuffle.class` property tells YARN which class to use to do the actual shuffle. The class we use for the shuffle handler is `org.apache.hadoop.mapred.ShuffleHandler`. Although it's possible to write your own shuffle handler by extending this class, it is recommended that the default class be used.

The next property is the `yarn.web-proxy.address`. This property is part of the installation process because we decided to run the YARN Proxy Server as a separate process. If we didn't configure it this way, the Proxy Server would run as part of the ResourceManager process. The Proxy Server aims to lessen the possibility of security issues. An ApplicationMaster will send to the ResourceManager a link for the application's web UI but, in reality, this link can point anywhere. The YARN Proxy Server lessens the risk associated with this link, but it doesn't eliminate it.

The remaining settings are the default ResourceManager port addresses.

Start-up Scripts

The Hadoop distribution includes a lot of convenient scripts to start and stop services such as the ResourceManager and NodeManagers. In a production cluster, however, we want the ability to integrate our scripts with the system's services management. On most Linux systems today, that means integrating with the init system.

Instead of relying on the built-in scripts that ship with Hadoop, we provide a set of init scripts that can be placed in `/etc/init.d` and used to start, stop, and monitor the Hadoop services. The files are as follows, with each service being identified by name:

```
hadoop-namenode
hadoop-datanode
hadoop-secondarynamenode
hadoop-resourcemanager
hadoop-nodemanager
hadoop-proxyserver
hadoop-historyserver
```

Of course, not all services run on all nodes; thus the installation script places the correct files on the requisite nodes. Before starting the service, the script also registers each service with `chkconfig`. Once these services are installed, they can be easily managed with commands like the following:

```
# service hadoop-namenode start
# hadoop-resourcemanager status
# hadoop-proxyserver restart
# hadoop-historyserver stop
```

Installing Hadoop with Apache Ambari

A script-based manual Hadoop installation can turn out to be a challenging process as it scales out from tens of nodes to thousands of nodes. Because of this complexity, a tool with the ability to manage installation, configuration, and monitoring of the Hadoop cluster became a much-needed addition to the Hadoop ecosystem. Apache Ambari provides the means to handle these simple, yet highly valuable tasks by utilizing an agent on each node to install required components, change configuration files, and monitor performance or failures of nodes either individually or as an aggregate. Both administrators and developers will find many of the Ambari features useful.

Installation with Ambari is faster, easier, and less error prone than manually setting up each service's configuration file. As an example, a 12-node cluster install of Hortonworks HDP2.X services (HDFS, MRv2, YARN, Hive, Sqoop, ZooKeeper, HCatalog, Oozie, HBase, and Pig) was accomplished in less than one hour with this tool. Ambari can dramatically cut down on the number of people required to install large clusters and increase the speed with which development environments can be created.



Figure 5.4 YARN metrics dashboard

Configuration files are maintained by an Ambari server acting as the primary interface to make changes to the cluster. Ambari guarantees that the configuration files on all nodes are the same by automatically redistributing them to the nodes every time a service's configuration changes. From an operational perspective, this approach provides peace of mind; you know that the entire cluster—from 10 to 4000-plus nodes—is always in sync. For developers, it allows for rapid performance tuning because the configuration files can be easily manipulated.

Monitoring encompasses the starting and stopping of services, along with reporting on whether a service is up or down, network usage, HDFS, YARN, and a multitude of other load metrics. Ganglia and Nagios report back to the Ambari server monitoring cluster health on issues ranging from utilization of services such as HDFS storage to failures of stack components or entire nodes. Users can also take advantage of the ability to monitor a number of YARN metrics such as cluster memory, total containers, NodeManagers, garbage collection, and JVM metrics. An example of the Ambari dashboard is shown in Figure 5.4.

Performing an Ambari-based Hadoop Installation

Compared to a manual installation of Hadoop 2, when using Ambari there are significantly fewer software requirements and operating system tasks like creating users or

groups and directory structures to perform. To manage the cluster with Ambari, we install two components:

1. The Ambari-Server, which should be its own node separate from the rest of the cluster
2. An Ambari-Agent on each node of the rest of the cluster that requires managing

For the purposes of this installation, we will reference the HDP 2.0 (Hortonworks Data Platform) documentation for the Ambari Automated Install (see <http://docs.hortonworks.com/#2.0> for additional information). In addition, although Ambari may eventually work with other Hadoop installations, we will use the freely available HDP version to ensure a successful installation.

Step 1: Check Requirements

As of this writing, RHEL 5 and 6, CentOS 5 and 6, OEL (Oracle Enterprise Linux) 5 and 6, and SLS 11 are supported by HDP 2, however Ubuntu is not supported at this time. Ensure that each node has yum, rpm, scp, curl, and wget. Additionally, ntpd should be running. Hive/HCatalog, Oozie, and Ambari all require their own internal databases; we can control these databases during installation, but they will need to be installed. If your cluster nodes do not have access to the Internet, you will have to mirror the HDP repository and set up your own local repositories.

Step 2: Install the Ambari Server

Perform the following actions to download the Ambari repository, add it to your existing yum.repos.d on the server node, and install the packages:

```
# wget http://public-repo-1.hortonworks.com/ambari-beta/centos6/1.x/beta/ambari.repo
# cp ambari.repo /etc/yum.repos.d
# yum -y install ambari-server
```

Next we set up the server. At this point, you can decide whether you want to customize the Ambari-Server database; the default is PostgreSQL. You will also be prompted to accept the JDK license unless you specify the --java-home option with the path of the JDK on all nodes in the cluster. For this installation, we will use the default values as signified by the silent install option. If the following command is successful, you should see “Ambari Server ‘setup’ completed successfully.”

```
# ambari-server setup --silent
```

Step 3: Install and Start Ambari Agents

Most of today’s current enterprise security divisions have a hard time accepting some of Hadoop’s more unusual requirements, such as root password-less ssh between all nodes in the cluster. Root password-less ssh is used only to automate installation of the Ambari agents and is not required for day-to-day operation. To stay within many

security guidelines, we will be performing a manual install of the Ambari-Agent on each node in the cluster. Be sure to set up the repository files on each node as described in Step 2. Optionally, you can use `pdsh` and `pdcp` with password-less root to help automate the installation of Ambari agents across the cluster. To install the agent, enter the following for each node:

```
# yum -y install ambari-agent
```

Next, configure the Ambari-Agent ini file with the FQDN of the Ambari-Server. By default, this value is localhost. This task can be done using `sed`.

```
# sed -i 's/hostname=localhost/hostname=yourAmbariServerFQDNhere/g' /etc/ambari-agent/conf/ambari-agent.ini
```

On all nodes, start Ambari with the following command:

```
# ambari-agent start
```

Step 4: Start the Ambari Server

To start the Ambari-Server, enter

```
# ambari-server start
```

Log into the Ambari-Server web console using `http://AmbariServerFQDN:8080`. If everything is working properly, you should see the login screen shown in Figure 5.5. The default login is username = admin and password = admin.

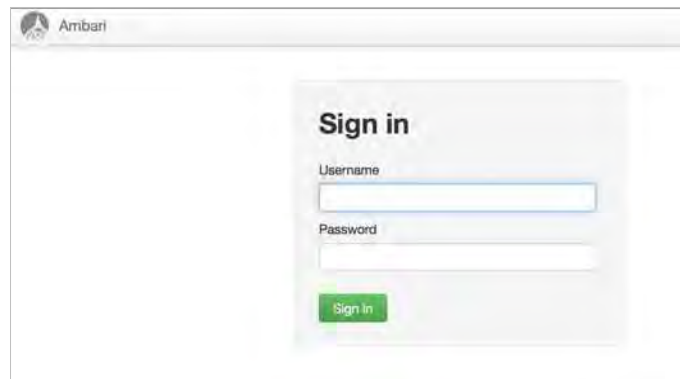


Figure 5.5 Ambari sign-in screen

Step 5: Install an HDP2.X Cluster

The first task is to name your cluster.

1. Enter a name for your cluster and click the green “Next” button (see Figure 5.6).

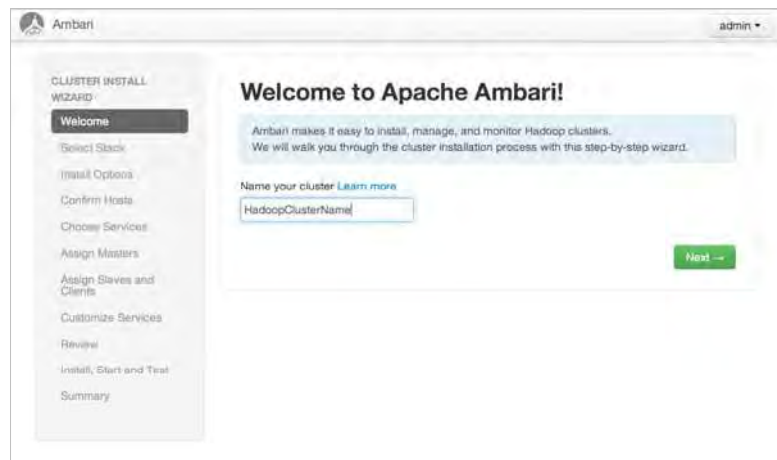


Figure 5.6 Enter a cluster name

2. The next option is to select the version of the HDP software stack. Currently the options include only the recent version of HDP. Select the 2.X stack option and click Next as shown in Figure 5.7.

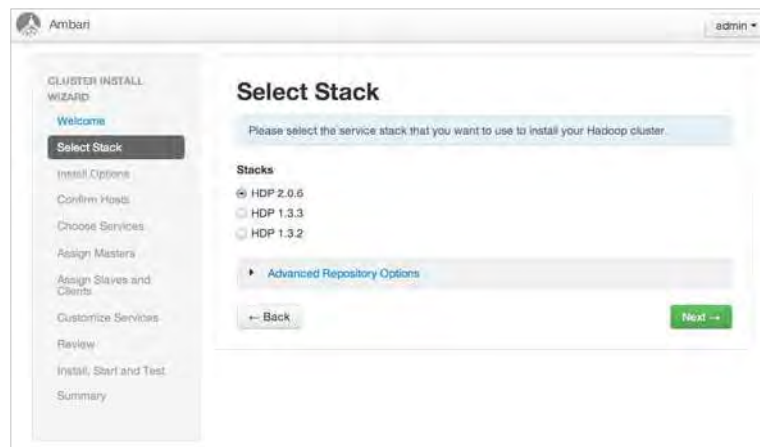


Figure 5.7 Select a Hadoop 2.X software stack

3. Next, Ambari will ask for your list of nodes, one per line, and ask you to select manual or private key registration. In this installation, we are using the manual method (see Figure 5.8).

Ambari admin

CLUSTER INSTALL WIZARD

- Welcome
- Select Stack
- Install Options**
- Confirm Hosts
- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services
- Review
- Install, Start and Test
- Summary

Install Options

Enter the list of hosts to be included in the cluster and provide your SSH key.

Target Hosts

Enter a list of hosts using the Fully Qualified Domain Name (FQDN), one per line. Or use [Pattern Expressions](#)

SetupDemo-ea82128b-285b-4601-b2cb-bc0d3482ad49
SetupDemo-d589f5d-4e0e-48f0-b3b1-a83100033607
SetupDemo-b6125e6a-4d31-445e-964a-

Host Registration Information

☐ Provide your SSH Private Key to automatically register hosts

No file chosen

Choose File

SSH user (root or passwordless sudo account) root

☒ Perform manual registration on hosts and do not use SSH

[Back](#) [Register and Confirm](#)

Figure 5.8 Hadoop install options

The installed Ambari-Agents should register with the Ambari-Server at this point. Any install warnings will also be displayed here, such as `ntpd` not running on the nodes. If there are issues or warnings, the registration window will indicate these as shown in Figure 5.9. Note that the example is installing a “cluster” of one node.

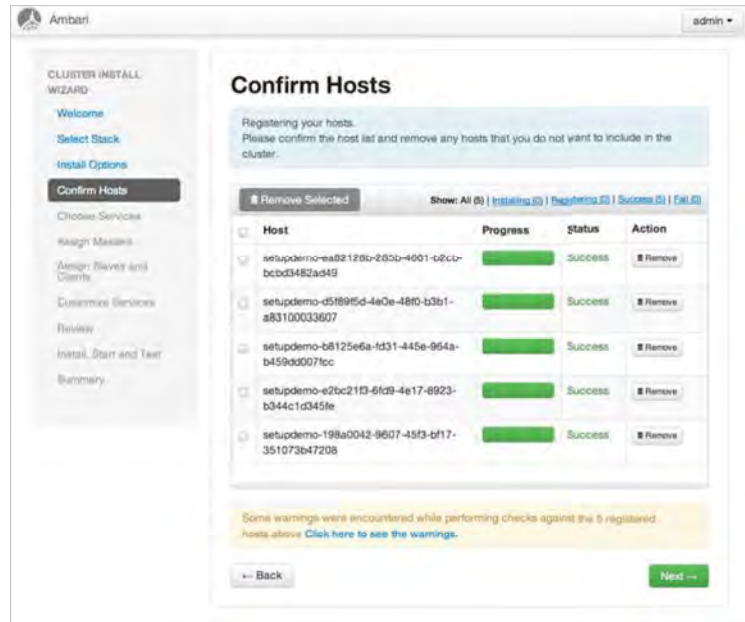


Figure 5.9 Ambari host registration screen

If everything is set correctly, your window should look like Figure 5.10.



Figure 5.10 Successful Ambari host registration screen

4. The next step is to select which components of the HDP2.X stack you want to install. At the very least, you will want to install HDFS, YARN, and MapReduceV2. In Figure 5.11, we will install everything.

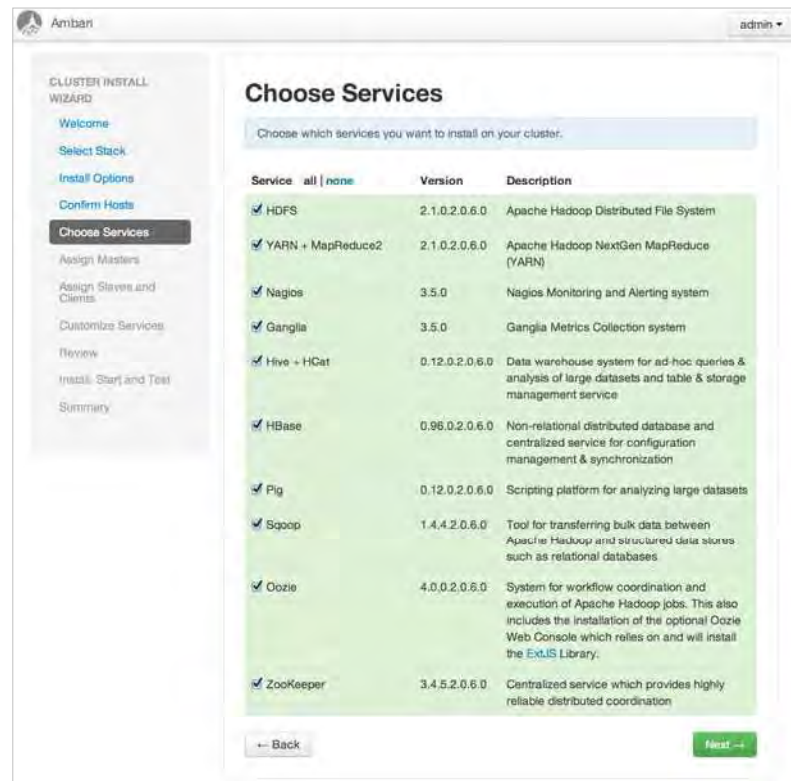


Figure 5.11 Ambari services selection

- The next step is to assign functions to the various nodes in your cluster (see Figure 5.12). The Assign Masters window allows for the selection of master nodes—that is, NameNode, ResourceManager, HBaseMaster, Hive Server, Oozie Server, etc. All nodes that have registered with the Ambari-Server will be available in the drop-down selection box. Remember that the ResourceManager has replaced the JobTracker from Hadoop version 1 and in a multi-node installation should always be given its own dedicated node.

The screenshot shows the 'Assign Masters' window in the Ambari web console. On the left is a sidebar for the 'CLUSTER INSTALL WIZARD' with steps: Welcome, Select Stack, Install Options, Confirm Hosts, Choose Services, **Assign Masters**, Assign Slaves and Clients, Customize Services, Review, Install, Start and Test, and Summary. The main area is titled 'Assign Masters' and contains a list of services with their assigned hosts and available roles.

Assign master components to hosts you want to run them on.
 * HiveServer2, Hive Metastore, and WebHCat Server will be hosted on the same server.

Service	Assigned Host	Available Roles
NameNode	setupdemo-198a0	NameNode, Hadoop Server
SNameNode	setupdemo-b8125	Ganglia Server, HBase Master
History Server	setupdemo-b8125	ZooKeeper
ResourceManager	setupdemo-b8125	
Nagios Server	setupdemo-198a0	NameNode, History Server
Ganglia Server	setupdemo-198a0	ResourceManager, HiveServer2
HiveServer2	setupdemo-b8125	Hive Metastore, WebHCat Server
Hive Metastore	setupdemo-b8125e6a-fd31-445e	Oozie Server, ZooKeeper
WebHCat Server	setupdemo-b8125e6a-fd31-445e	
HBase Master	setupdemo-198a0	
Oozie Server	setupdemo-b8125	
ZooKeeper	setupdemo-198a0	
ZooKeeper	setupdemo-b8125	
ZooKeeper	setupdemo-d5f89f	

At the bottom, there is a 'Back' button and a 'Next' button. A yellow warning box states: '2 hosts not running master services'.

Figure 5.12 Ambari host assignment

6. In this step, you assign NodeManagers (which run YARN containers), RegionServers, and DataNodes (HDFS). Remember that the NodeManager has replaced the TaskTracker from Hadoop version 1, so you should always co-locate one of these node managers with a DataNode to ensure that local data is available for YARN containers. The selection window is shown in Figure 5.13. Again, this example has only a single node.

Ambari admin

CLUSTER INSTALL WIZARD

- Welcome
- Select Stack
- Install Options
- Confirm Hosts
- Choose Services
- Assign Masters
- Assign Slaves and Clients**
- Customize Services
- Review
- Install, Start and Test
- Summary

Assign Slaves and Clients

Assign slave and client components to hosts you want to run them on. Hosts that are assigned master components are shown with a red dot. 'Client' will install HDFS Client, MapReduce 2 Client, YARN Client, Hive Client, HCat Client, HBase Client, Pig, Sqoop, Oozie Client and ZooKeeper Client.

Host	☒ DataNode	☒ NodeManager	☒ RegionServer
setupdemo-ea62128b-285b-4601-b2cb-bcbd3482ad49	☒	☒	☒
setupdemo-df89f5d-4e0e-48f0-b3b1-a83100033607	☐	☐	☐
setupdemo-b8125e6a-fd31-445e-964a-b459dd007fcc	☐	☐	☐
setupdemo-e2bc21f3-6fd9-4e17-8923-b344c1d345fe	☒	☒	☒
setupdemo-198a0042-9607-45f3-bf17-381073b47208	☐	☐	☐

Host:

[< Back](#) [Next >](#)

Figure 5.13 Ambari slave and client component assignment

7. The next set of screens allows you to define any initial parameter changes and usernames for the services selected for installation (i.e., Hive, Oozie, and Nagios). Users are required to set up the database passwords and alert reporting email before continuing. The Hive database setup is pictured in Figure 5.14.

Figure 5.14 Ambari customized services window

8. The final step before installing Hadoop is a review of your configuration. Figure 5.15 summarizes the actions that are about to take place. Be sure to double-check all settings before you commit to an install.

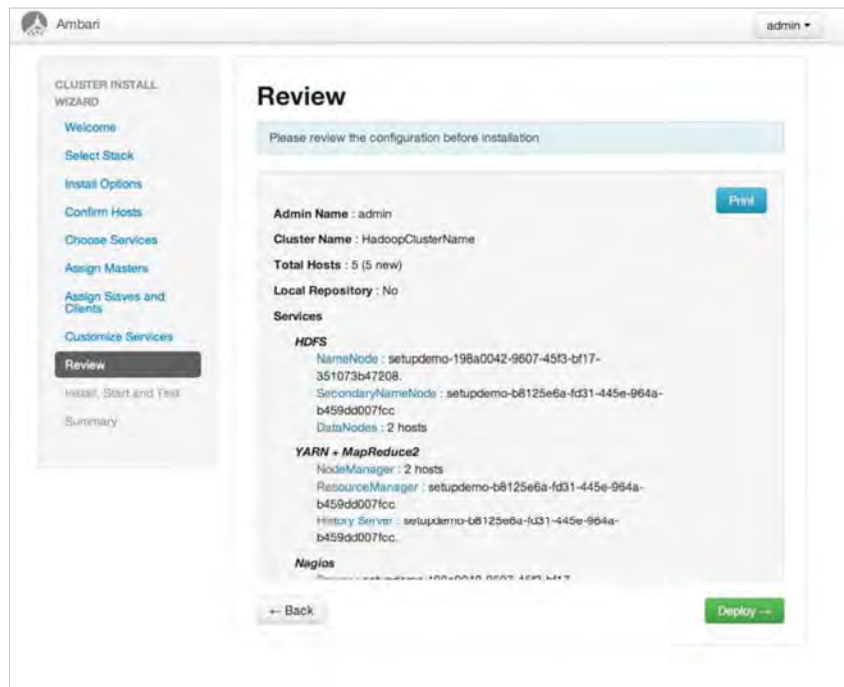


Figure 5.15 Ambari final review window

9. During the installation step shown in Figure 5.16, the cluster is actually provisioned with the various software packages. By clicking on the node name, you can drill down into the installation status of every component. Should the installation encounter any errors on specific nodes, these errors will be highlighted on this screen.

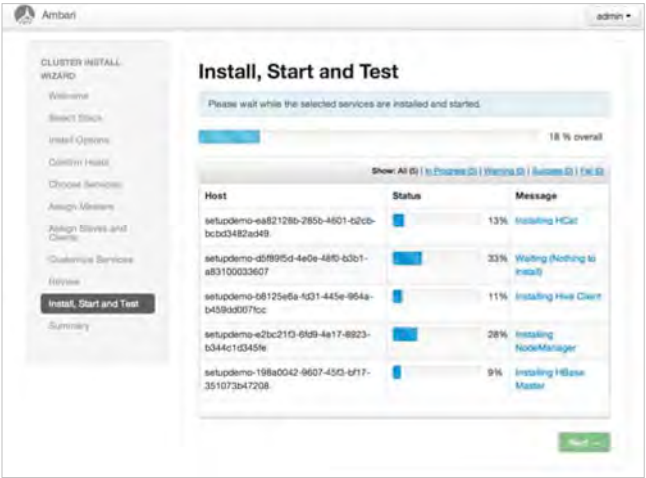


Figure 5.16 Ambari deployment process

10. Once the installation of the nodes is complete, a summary window similar to Figure 5.17 will be displayed. The screen indicates which tasks were completed and identifies any preliminary testing of cluster services.

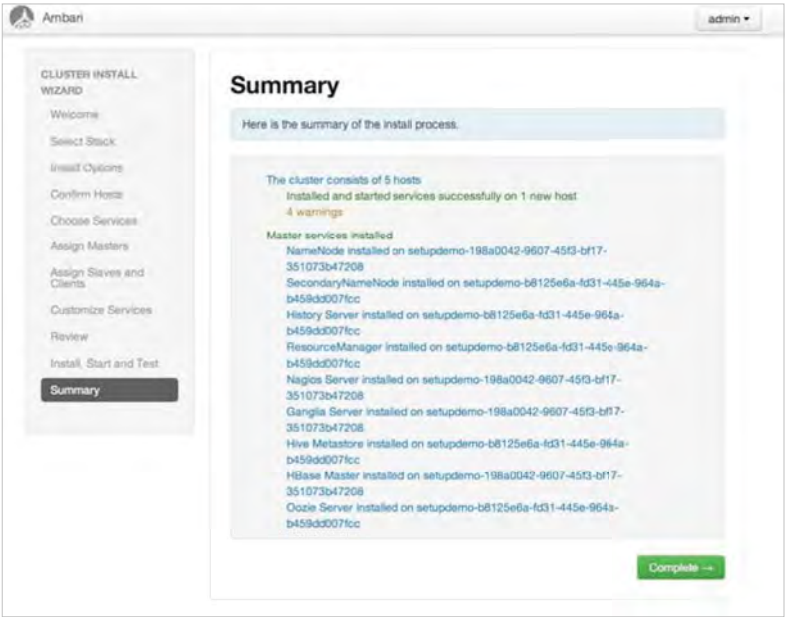


Figure 5.17 Ambari summary window

Congratulations! You have just completed installing HDP2.X with Ambari. Consult the online Ambari documentation (<http://docs.hortonworks.com/#2>) for further details on the installation process.

If you are using Hive and have a FQDN longer than 60 characters (as is common in some cloud deployments), please note that this can cause authentication issues with the MySQL database that Hive installs by default with Ambari. To work around this issue, start the MySQL database with the “--skip-name-resolve” option to disable FQDN resolution and authenticate based only on IP number.

Wrap-up

It is possible to perform an automated script-based install of moderate to very large clusters. The use of parallel distributed shell and copy commands (`pdsh` and `pdcp`, respectively) makes a fully remote installation on any number of nodes possible. The script-based install process is designed to be flexible, and users can easily modify it for their specific needs on Red Hat (and derivative)–based distributions.

In addition to the install script, some useful functions for creating and changing the Hadoop XML property files are made available to users. To aid with start-up and shut-down of Hadoop services, the scripted install also provides SysV init scripts for Red Hat–based systems.

Finally, a graphical install process using Apache Ambari was described in this chapter. With Ambari, the entire Hadoop installation process can be automated with a powerful point-and-click interface. As we will see in Chapter 6, “Apache Hadoop YARN Administration,” Ambari can also be used for administration purposes.

Installing Hadoop 2 YARN from scratch is also easy. The single-machine installation outlined in Chapter 2, “Apache Hadoop YARN Install Quick Start,” can be used as a guide. Again, in custom scenarios, `pdsh` and `pdcp` can be very valuable.