On-Premise Database Product Guide

Last generated: March 01, 2018

Table of Contents

Welcome to the Splice Machine On-Premise Database!

Welcome to Splice Machine, the database platform for adaptive applications that manage operational processes. This site contains documentation for our **On-Premise Database**.

If you're not yet familiar with our lineup of products, please visit the Getting Started Page in our web site to learn more about them.

Splice Machine delivers an open-source data platform that incorporates the proven scalability of HBase and the in-memory performance of Apache Spark. The cost-based optimizer uses advanced statistics to choose the best compute engine, storage engine, index access, join order and join algorithm for each task. In this way, Splice Machine can concurrently process transactional and analytical workloads at scale.

You can deploy the Splice Machine *On-Premise Database* on a standalone computer or on your own cluster that is managed by Cloudera, MapR, or Hortonworks. We offer Enterprise, Cluster Community, and Standalone Community editions. You can also access the open source code for our community editions on GitHub.

Getting Started

To get started, read about our products on our web site and decide which edition is the right one for you, then download a version or contact our sales team to start using Splice Machine:

- Learn more about our *Enterprise Edition* features and advantages. To purchase this edition, please contact Splice Machine Sales today.
- Download our free *Cluster Community* edition to install on your own cluster.
- Download our free *Standalone Community* edition to run on your MacOS, Linux, or CentOS computer.

Using our Documentation

Please visit our Documentation Summary page for a quick introduction to navigating and using the components of our documentation system.

Version Information

This site contains the documentation for version 2.5 of Splice Machine.

If you're using an earlier version, you'll find the documentation here:

Version 2.5	Release 2.5 Documentation (Feb 2017)

Splice Machine Requirements

This is an On-Premise-Only topic! [Learn about our products]

This topic summarizes the hardware and software requirements for Splice Machine running on a cluster or on a standalone computer.

Cluster Node Requirements

The following table summarizes the minimum requirements for the nodes in your cluster:

Component	Requirements
Cores	Splice Machine recommends that each node in your cluster have 8-12 hyper-threaded cores (16-32 hyper-threads) for optimum throughput and concurrency.
Memory	We recommend that each machine in your cluster have at least 64 GB of available memory.
Disk Space	Your root drive needs to have at least 100 GB of free space. Splice Machine recommends separate data drives on each cluster node to maintain a separation between the operating system and your database data. You need capacity for a minimum of three times the size of the data you intend to load; the typical recommended configuration is 2 TB or more of attached storage per node. **Your data disks should be set up with a single partition and formatted with an `ext4` file system.**
Hadoop Ecosystem	The table in the next section, Hadoop Ecosystem Requirements, summarizes the specific Hadoop component versions that we support in each of our product releases.
Software Tools and System Settings	The *Linux Configuration* topic in each section of our *Installation Guide* that pertains to your installation summarizes the software tools and system settings required for your cluster machines.

Amazon Web Services (AWS) Requirements

If you're running on AWS, your cluster must meet these minimum requirements:

Component	Requirements
Minimum Cluster Size	The minimum cluster size on AWS is *5 nodes*: • 1 master node • 4 worker nodes
Minimum Node Size	Minimum recommended size of each node is *m4.4xlarge*.
Disk Space	Minimum recommended storage space: • *100GB* EBS root drive • 4 EBS data drives per node Note that the required number of data drives per node depends on your use case.

Hadoop Ecosystem Requirements

The following table summarizes the required Hadoop ecosystem components for your platform:

Hadoop platform	Linux	Hadoop	HBase	ZooKeeper
CDH 5.12.0 and 5.8.3	CentOS/RHEL 6	2.6.0	1.0.0	3.4.5
HDP 2.5.5	CentOS/RHEL 6	2.7.1	1.1.2	3.4.5
MapR 5.2.0	CentOS/RHEL 6	2.7.0	1.1.1	3.4.5

Java JDK Requirements

Splice Machine supports the following versions of the Java JDK:

- Oracle JDK 1.8, update 60 or higher

NOTE: We recommend that you do not use JDK 1.8.0_40

Splice Machine does not test our releases with OpenJDK, so we recommend against using it.

Standalone Version Prerequisites

You can use the standalone version of Splice Machine on MacOS and Linux computers that meet these basic requirements:

Component	Requirements
Operating System	Mac OS X, version 10.8 or later. CentOS 6.4 or equivalent.
CPU	Splice Machine recommends 2 or more multiple-core CPUs.
Memory	At least 16 GB RAM, of which at least 10 GB is available.
Disk Space	At least 100 GB of disk space available for Splice Machine software, plus as much space as will be required for your data; for example, if you have a 1 TB dataset, you need at least 1 TB of available data space.
Software	You must have JDK installed on your computer.

Splice Machine Enterprise Edition Support

The following table summarizes support for the Enterprise Edition of Splice Machine.

This is an On-Premise-Only topic! [Learn about our products]

Support Summary	
Term	1 year
Named Support Contacts	5 contacts
Access Channels	Web and Phone
Support Hours	24x7
Support Response Times	
Severity 1	1 hour
Severity 2	4 hours
Severity 3	1 business day
Severity 4	1 business day
Severity Definitions	
Severity 1	Production down or severely degraded to point of being non-functional
Severity 2	Production functional overall but issues reducing performance or functionality
Severity 3	Development or minor production issue
Severity 4	Question or issue that does not impact production

Release Notes for the Splice Machine On-Premise Product

This is an On-Premise-Only topic! Learn about our products

This topic includes release notes that are specific to the Splice Machine *On-Premise Database* product, in these sections:

- Supported Platforms
- Enterprise-only Features
- Running the Standalone Version

Most of the information about changes in the Splice Machine database that forms the basis of this product are found in the Splice Machine database release notes.

After Updating

After updating to a new release of Splice Machine, you need to update your stored statement metadata by calling these two system procedures:

```
CALL SYSCS_UTIL.SYSCS_UPDATE_METADATA_STORED_STATEMENTS();
CALL SYSCS_UTIL.SYSCS_EMPTY_STATEMENT_CACHE();
```

Supported Platforms

The supported platforms for release 2.7 are:

- Cloudera CDH 5.12.0, 5.8.3
- MapR 5.2.0
- HortonWorks HDP 2.5.5, 2.6.3

Enterprise-only Features

Some features only work on the *Enterprise Edition* of Splice Machine; they **do not** work on the Community Edition of Splice Machine. To obtain a license for the Splice Machine *Enterprise Edition*, please Contact Splice Machine Sales today.

These are the enterprise-only features in our *On-Premise Database*:

- Backup/Restore
- LDAP integration
- Column-level user privileges
- Kerberos enablement

- Encryption at rest

Running the Standalone Version

The supported operating systems for the STANDALONE release of Splice Machine are:

- Mac OS X (10.8 or greater)
- Centos (6.4 or equivalent)

Splice Machine Installation Guide

This is an On-Premise-Only topic! Learn about our products

This *Installation Guide* walks you through installing Splice Machine on your cluster, or on computer if you're using the standalone version.

The fastest way to get started with Splice Machine is to set up our sandbox on the Amazon Web Services (AWS) platform on EC2 instances using `cloud.splicemachine.com`:

See the Installing Splice Machine on Amazon Web Services topic for step-by-step instructions for setting up the Splice Machine sandbox.

If you want to download and install Splice Machine on your cluster or standalone computer, please read the remainder of this page, which includes these sections:

- The Cluster Node Requirements section below details the hardware and ecosystem requirements for installing Splice Machine on a cluster or on a standalone computer.
- The Configure Linux for Splice Machine section specifies the Linux software that Splice Machine requires.
- The Install Splice Machine links to the platform-specific installation and upgrade pages for each version of Splice Machine.

Installing the Splice Machine Sandbox on AWS

The fastest way to get started with Splice Machine is to set up our sandbox on the Amazon Web Services (AWS) platform on EC2 instances using `cloud.splicemachine.com`.

See the Installing Splice Machine on Amazon Web Services topic for step-by-step instructions.

Cluster Node Requirements

The following table summarizes the minimum requirements for the nodes in your cluster:

Component	Requirements
Cores	Splice Machine recommends that each node in your cluster have 8-12 hyper-threaded cores (16-32 hyper-threads) for optimum throughput and concurrency.
Memory	We recommend that each machine in your cluster have at least 64 GB of available memory.

Component	Requirements
Disk Space	Your root drive needs to have at least 100 GB of free space. Splice Machine recommends separate data drives on each cluster node to maintain a separation between the operating system and your database data. You need capacity for a minimum of three times the size of the data you intend to load; the typical recommended configuration is 2 TB or more of attached storage per node. **Your data disks should be set up with a single partition and formatted with an `ext4` file system.**
Hadoop Ecosystem	The table in the next section, Hadoop Ecosystem Requirements, summarizes the specific Hadoop component versions that we support in each of our product releases.
Software Tools and System Settings	The *Linux Configuration* topic in each section of our *Installation Guide* that pertains to your installation summarizes the software tools and system settings required for your cluster machines.

Amazon Web Services (AWS) Requirements

If you're running on AWS, your cluster must meet these minimum requirements:

Component	Requirements
Minimum Cluster Size	The minimum cluster size on AWS is *5 nodes*: • 1 master node • 4 worker nodes
Minimum Node Size	Minimum recommended size of each node is *m4.4xlarge*.
Disk Space	Minimum recommended storage space: • *100GB* EBS root drive • 4 EBS data drives per node Note that the required number of data drives per node depends on your use case.

Hadoop Ecosystem Requirements

The following table summarizes the required Hadoop ecosystem components for your platform:

Hadoop platform	Linux	Hadoop	HBase	ZooKeeper
CDH 5.12.0 and 5.8.3	CentOS/RHEL 6	2.6.0	1.0.0	3.4.5
HDP 2.5.5	CentOS/RHEL 6	2.7.1	1.1.2	3.4.5
MapR 5.2.0	CentOS/RHEL 6	2.7.0	1.1.1	3.4.5

Java JDK Requirements

Splice Machine supports the following versions of the Java JDK:

- Oracle JDK 1.8, update 60 or higher

NOTE: We recommend that you do not use JDK 1.8.0_40

Splice Machine does not test our releases with OpenJDK, so we recommend against using it.

Standalone Version Prerequisites

You can use the standalone version of Splice Machine on MacOS and Linux computers that meet these basic requirements:

Component	Requirements
Operating System	Mac OS X, version 10.8 or later. CentOS 6.4 or equivalent.
CPU	Splice Machine recommends 2 or more multiple-core CPUs.
Memory	At least 16 GB RAM, of which at least 10 GB is available.
Disk Space	At least 100 GB of disk space available for Splice Machine software, plus as much space as will be required for your data; for example, if you have a 1 TB dataset, you need at least 1 TB of available data space.
Software	You must have JDK installed on your computer.

Configure Linux for Splice Machine

The following table summarizes Linux configuration requirements for running Splice Machine on your cluster:

Configuration Step	Description
Configure SSH access:	Configure the user account that you're using for cluster administration for password-free access, to simplify installation.
Configure swappiness:	`echo 'vm.swappiness = 0' >> /etc/sysctl.conf`
If you are using Ubuntu:	`rm /bin/sh ; ln -sf /bin/bash /bin/sh`
If your using CentOS or RHEL:	`sed -i '/requiretty/ s/^/#/' /etc/sudoers`
Required software:	Verify that the following set of software (or packages) is available on each node in your cluster: » `curl` » `Oracle JDK 1.8, update 60 or higher. We recommend against using JDK 1.8.0_40 or OpenJDK.` **NOTE:** `Your platform management software may re-install JDK during its own installation process.` » `nscd` » `ntp` » `openssh`, `openssh-clients`, and `openssh-server` » `patch` » `rlwrap` » `wget`

Configuration Step	Description
Additional required software on CentOS or RHEL	`If you're running on CENTOS or RHEL, you also need to have this software available on each node:` » `ftp` » `EPEL` repository
Services that must be started	You need to make sure that the following services are enabled and started: » `nscd` » `ntpd` (`ntp` package) » `sshd` (`openssh`-server package)
Time zone setting	Make sure all nodes in your cluster are set to the same time zone.

Install Splice Machine

If you've decided to try our sandbox, see the Installing Splice Machine on Amazon Web Services topic for step-by-step instructions for setting up the Splice Machine sandbox.

To install Splice Machine on your cluster or standalone computer, click the link below to see the instructions for your platform; each page walks you through downloading and installing and configuring a specific version of Splice Machine:

Hadoop Platform	Installation Guide
Cloudera-managed cluster	Installing and Configuring Splice Machine for Cloudera Manager
Hortonworks-managed cluster	Installing and Configuring Splice Machine for Hortonworks HDP
MapR-managed cluster	Installing and Configuring Splice Machine for MapR
Standalone version of Splice Machine	Installing the Standalone Version of Splice Machine

For access to the source code for the Community Edition of Splice Machine, visit our open source GitHub repository.

Installing Splice Machine on Amazon Web Services

This is an On-Premise-Only topic! Learn about our products

The fastest way to deploy Splice Machine is on the Amazon Web Services (AWS) platform on EC2 instances using cloud.splicemachine.com.

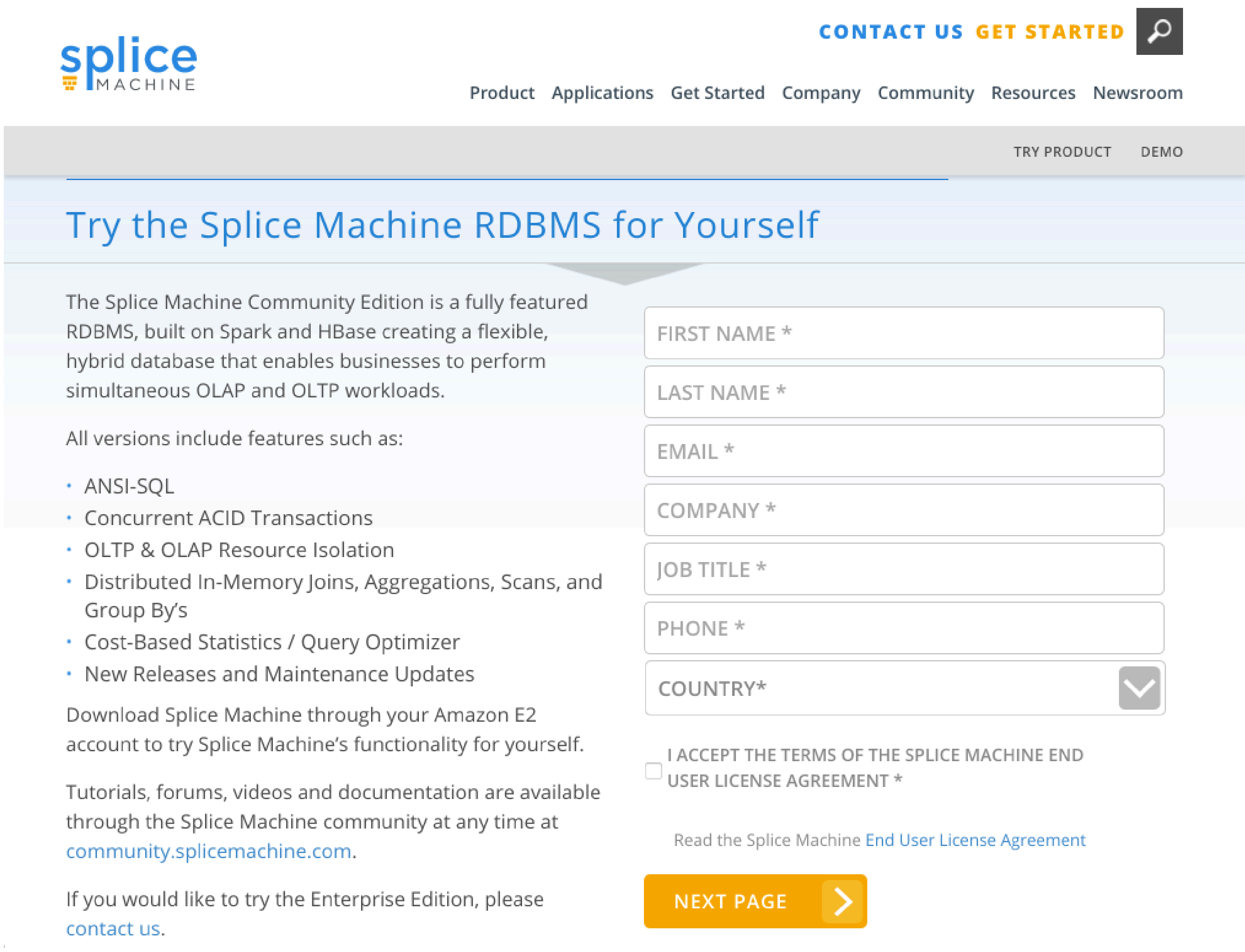
You must have an existing AWS account before running this process.

Getting Started with the SandBox Cluster Generator

To generate your Splice Machine sandbox cluster, point your browser to:

```
www.splicemachine.com/get-started/sandbox-start/
```

Fill in the form that's displayed:


CONTACT US GET STARTED
Product Applications Get Started Company Community Resources Newsroom
TRY PRODUCT DEMO

Try the Splice Machine RDBMS for Yourself

The Splice Machine Community Edition is a fully featured RDBMS, built on Spark and HBase creating a flexible, hybrid database that enables businesses to perform simultaneous OLAP and OLTP workloads.

All versions include features such as:

- ANSI-SQL
- Concurrent ACID Transactions
- OLTP & OLAP Resource Isolation
- Distributed In-Memory Joins, Aggregations, Scans, and Group By's
- Cost-Based Statistics / Query Optimizer
- New Releases and Maintenance Updates

Download Splice Machine through your Amazon E2 account to try Splice Machine's functionality for yourself.

Tutorials, forums, videos and documentation are available through the Splice Machine community at any time at community.splicemachine.com.

If you would like to try the Enterprise Edition, please contact us.

FIRST NAME *

LAST NAME *

EMAIL *

COMPANY *

JOB TITLE *

PHONE *

COUNTRY*

I ACCEPT THE TERMS OF THE SPLICE MACHINE END USER LICENSE AGREEMENT *

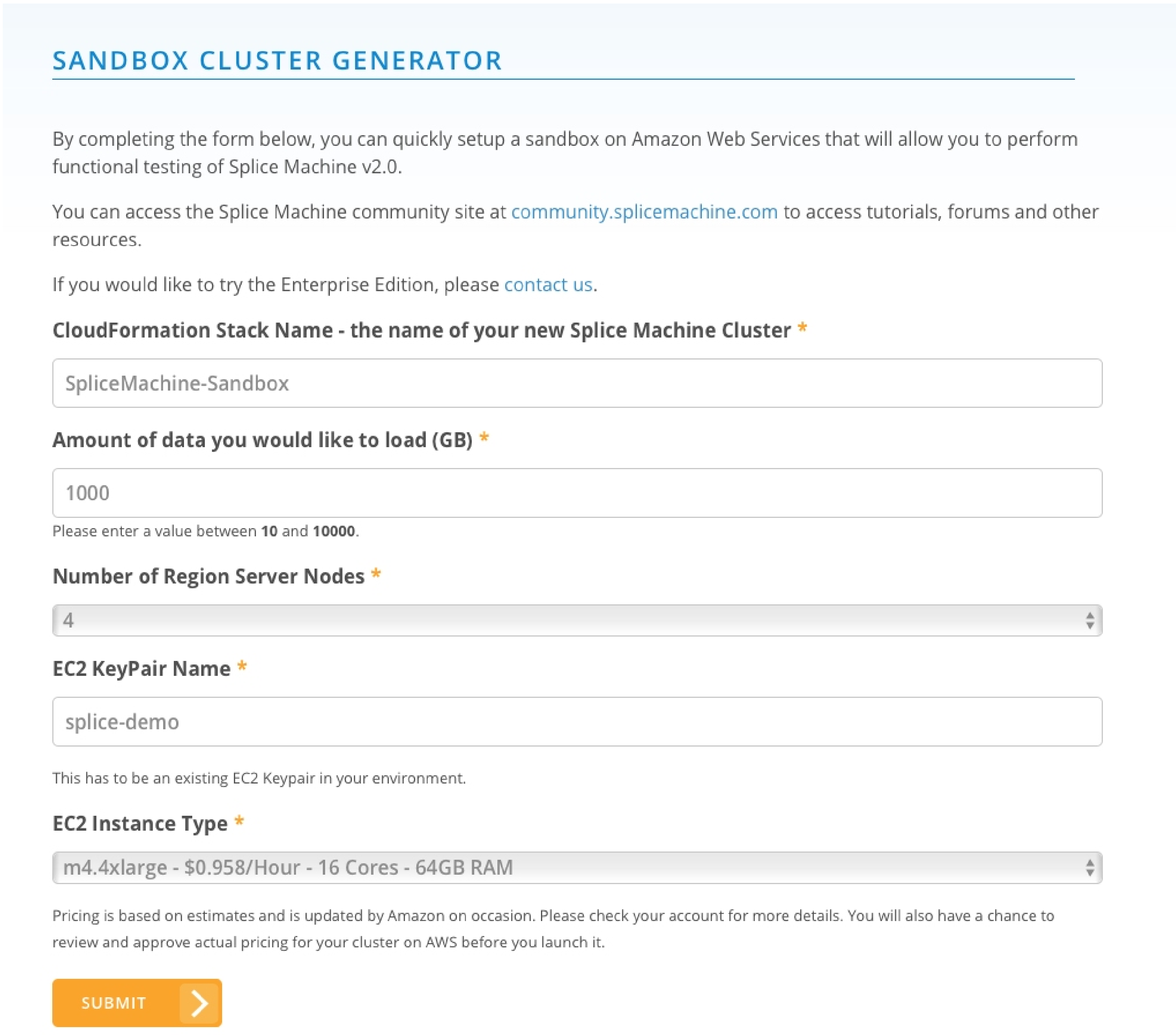
Read the Splice Machine End User License Agreement

NEXT PAGE

Click the `NEXT PAGE` button to proceed and configure your cluster.

Configure Your Cluster

AWS provides numerous choices for configuring your sandbox; Splice Machine recommends a minimum configuration such as the following:



SANDBOX CLUSTER GENERATOR

By completing the form below, you can quickly setup a sandbox on Amazon Web Services that will allow you to perform functional testing of Splice Machine v2.0.

You can access the Splice Machine community site at community.splicemachine.com to access tutorials, forums and other resources.

If you would like to try the Enterprise Edition, please contact us.

CloudFormation Stack Name - the name of your new Splice Machine Cluster *

SpliceMachine-Sandbox

Amount of data you would like to load (GB) *

1000

Please enter a value between **10** and **10000**.

Number of Region Server Nodes *

4

EC2 KeyPair Name *

splice-demo

This has to be an existing EC2 Keypair in your environment.

EC2 Instance Type *

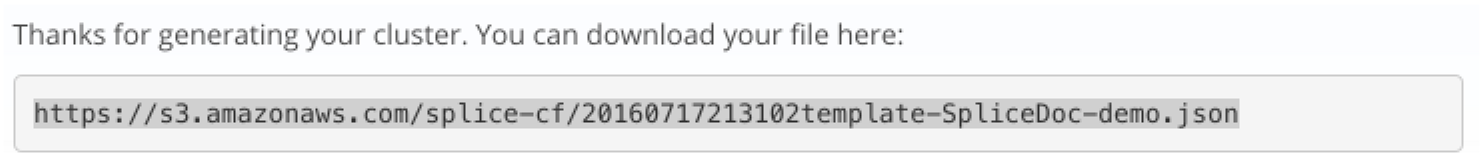
m4.4xlarge - $0.958/Hour - 16 Cores - 64GB RAM

Pricing is based on estimates and is updated by Amazon on occasion. Please check your account for more details. You will also have a chance to review and approve actual pricing for your cluster on AWS before you launch it.

The EC2 KeyPair Name of your pem file, which can be located anywhere on your computer.

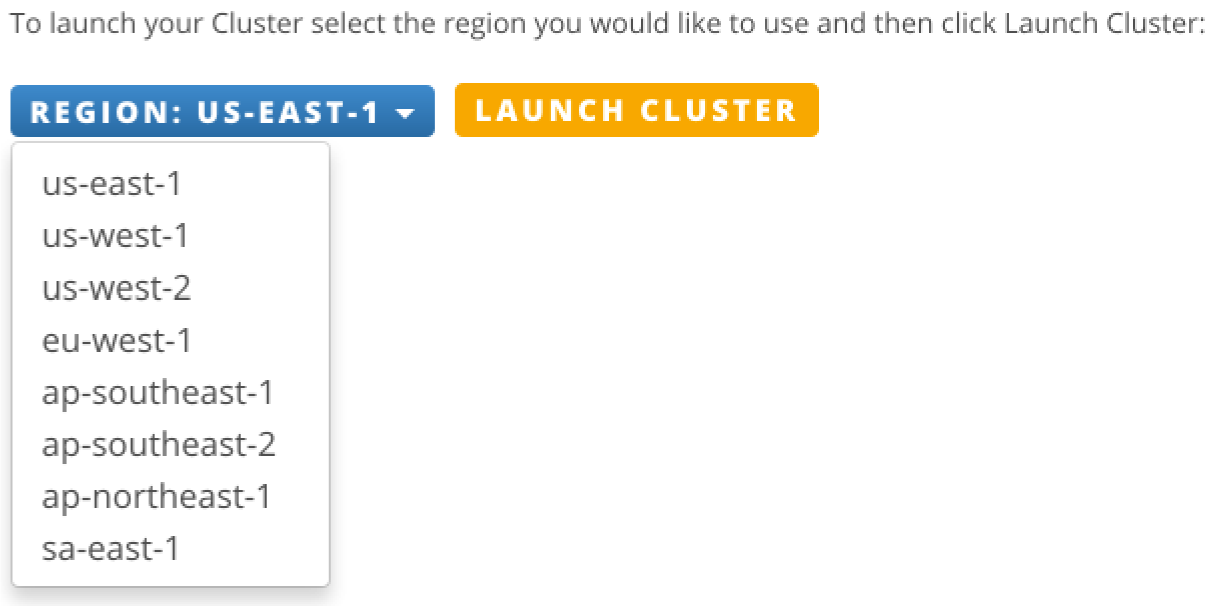
> **NOTE:** When specifying your EC2 KeyPair name, **DO NOT** include any file name suffix. For example, enter `splice-demo`, NOT `splice-demo.pem`.

When you click the `SUBMIT` button, Splice Machine generates an AWS cluster generator template file, which is a `json` file that you can download for safekeeping.


Thanks for generating your cluster. You can download your file here:

```
https://s3.amazonaws.com/splice-cf/20160717213102template-SpliceDoc-demo.json
```

Before launching your cluster, select in which AWS region you want your cluster located:

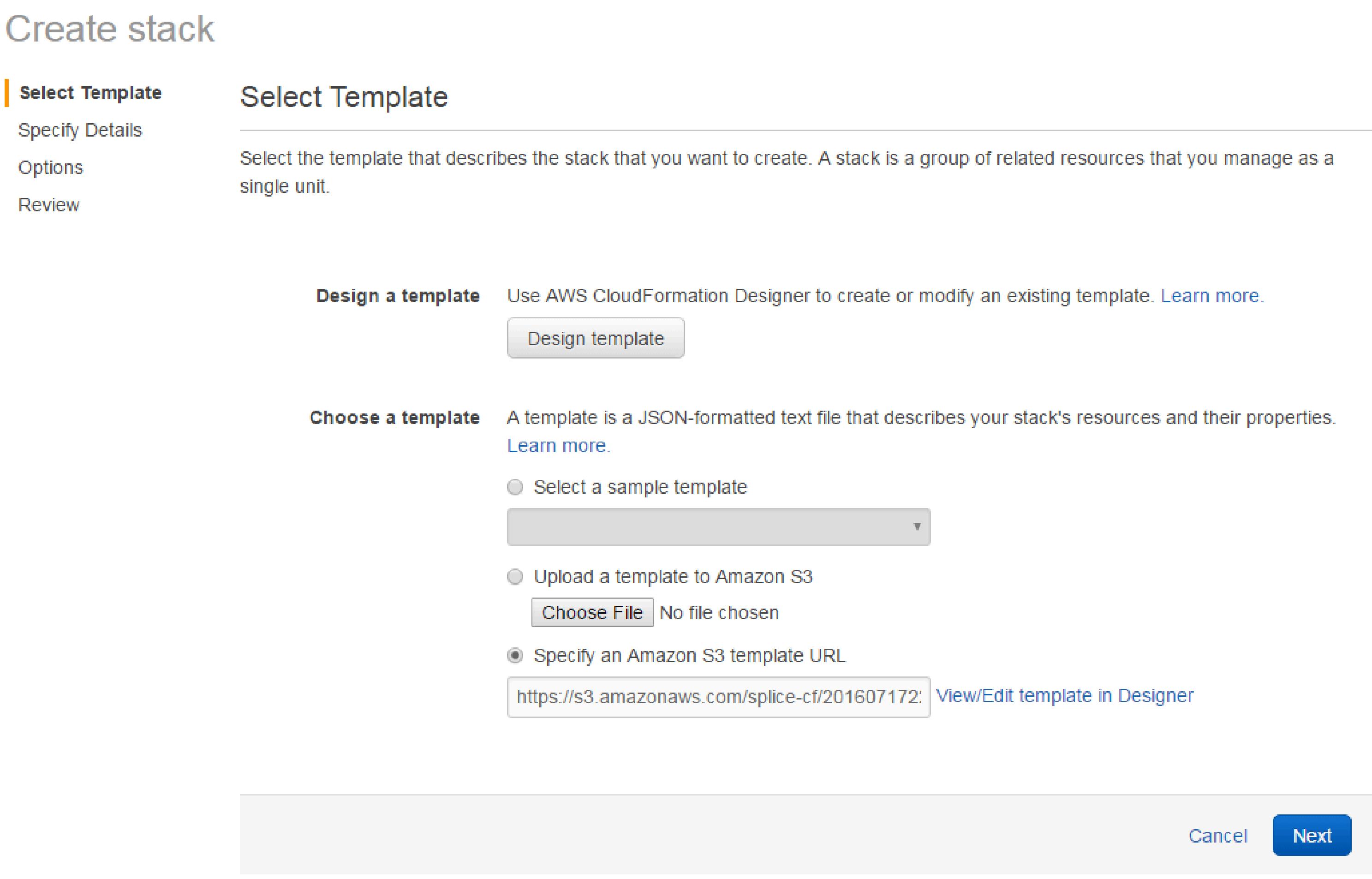

To launch your Cluster select the region you would like to use and then click Launch Cluster:

Click the `LAUNCH CLUSTER` button to proceed to finalizing your sandbox template.

Finalize Your Sandbox Stack Template

Your sandbox stack template file should already be entered into the Amazon S3 template field at the bottom of the screen; if you're using a different template file, you can select that instead; this is **NOT RECOMMENDED**.

Select Your Sandbox Template


Create stack

Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template Use AWS CloudFormation Designer to create or modify an existing template. Learn more.

Choose a template A template is a JSON-formatted text file that describes your stack's resources and their properties. Learn more.

○ Select a sample template

○ Upload a template to Amazon S3

No file chosen

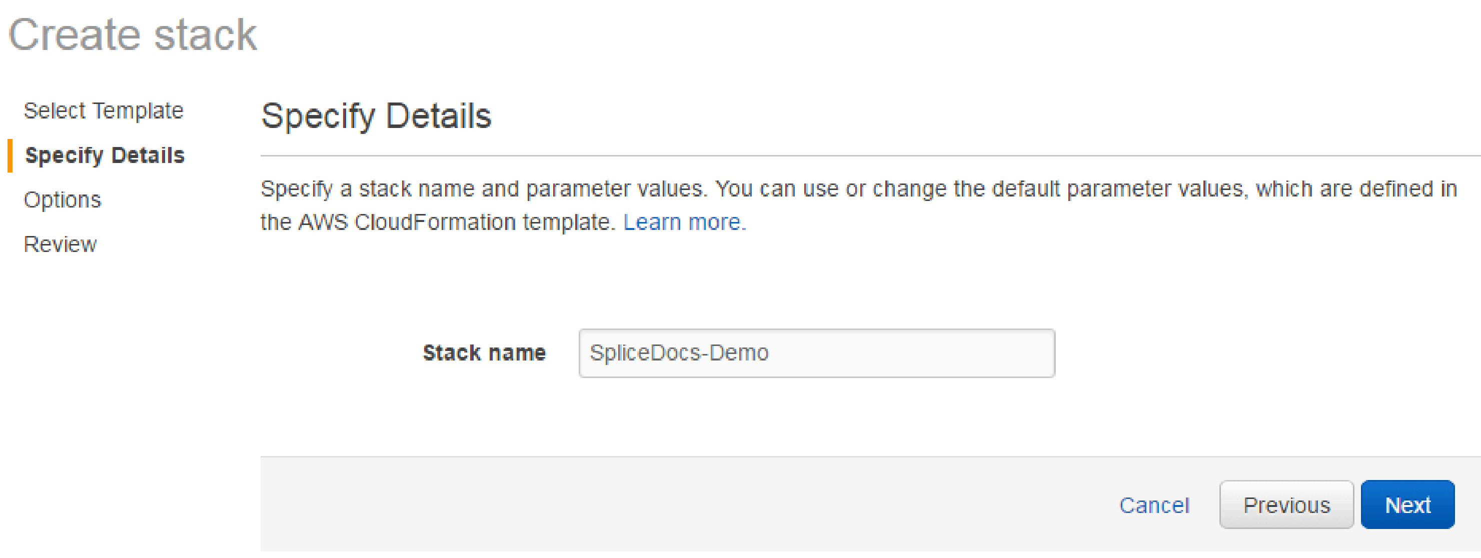
● Specify an Amazon S3 template URL

https://s3.amazonaws.com/splice-cf/201607172

Click the `Next` button to proceed to the *Specify Details* screen.

Rename Your Stack if Desired

You set up a name for your stack at the beginning of this process; however, you can change that here if you want. Then click `Next` to continue on to the *Options* screen.


Create stack

- Select Template
- **Specify Details**
- Options
- Review

Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. Learn more.

Stack name SpliceDocs-Demo

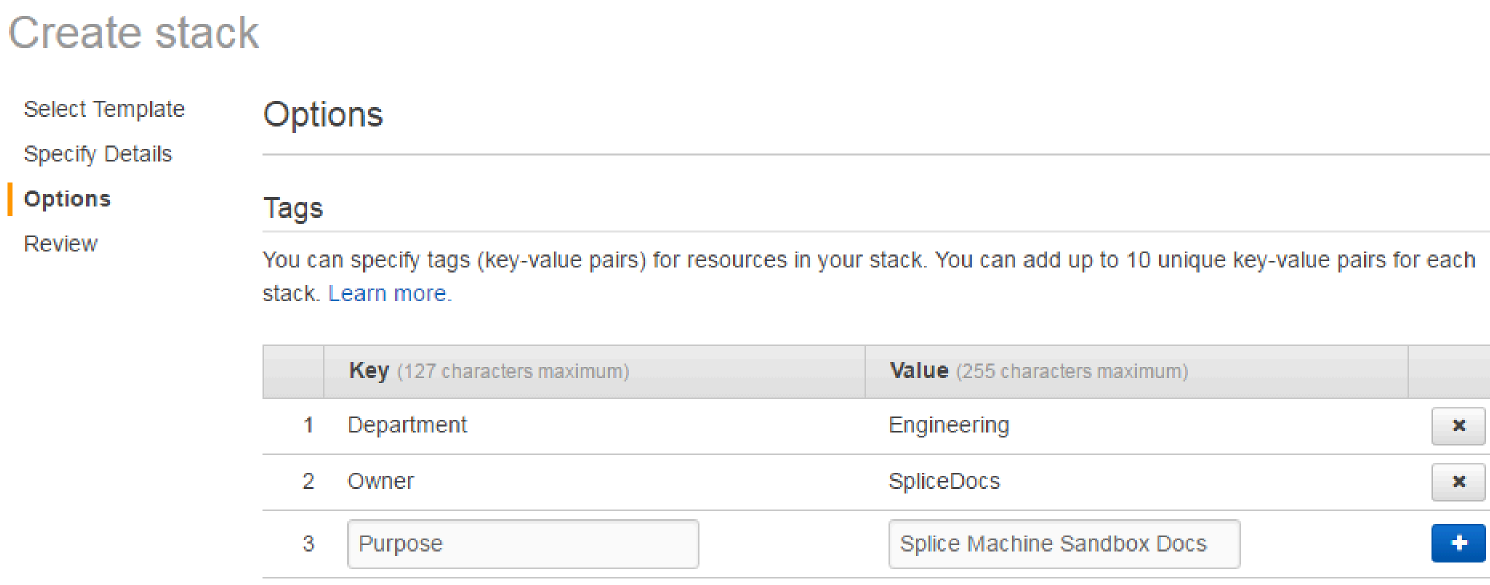
Cancel | Previous | Next

Specify Options for Your Stack

In the *Options* screen, you can add tags and set advanced options for your sandbox stack.

Adding Resource Tags to the Stack, if Desired

You can optionally tag your sandbox stack with whatever key values you want. For example:


Create stack

- Select Template
- Specify Details
- **Options**
- Review

Options

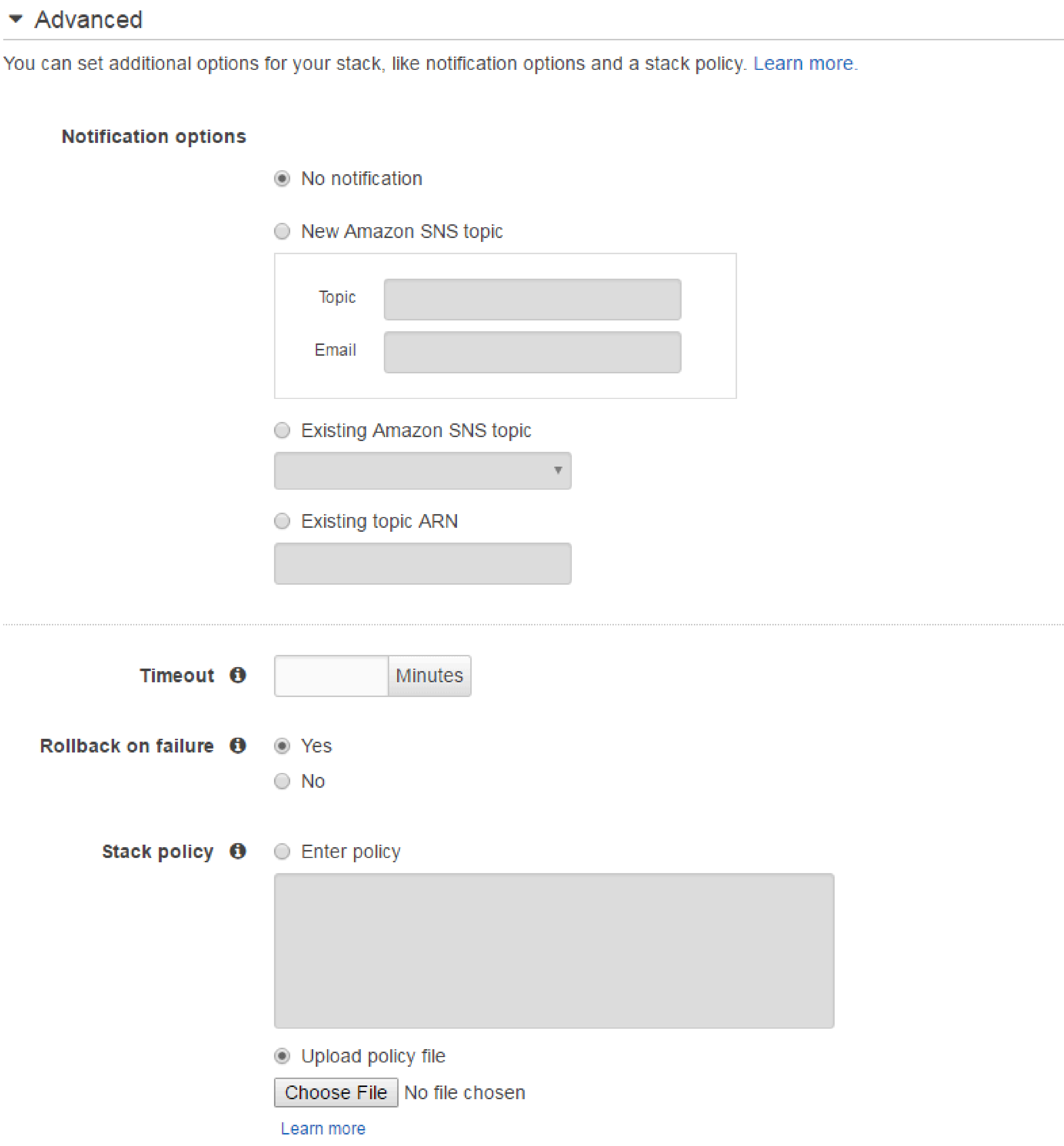
Tags

You can specify tags (key-value pairs) for resources in your stack. You can add up to 10 unique key-value pairs for each stack. Learn more.

	Key (127 characters maximum)	**Value** (255 characters maximum)	
1	Department	Engineering	×
2	Owner	SpliceDocs	×
3	Purpose	Splice Machine Sandbox Docs	+

Setting Advanced Options for Your Stack

You can also configure advanced options for your sandbox stack in the *Options* screen, including notification, failure rollback, and policy options. For more information about these options, click the `Learn more` button.



Advanced

You can set additional options for your stack, like notification options and a stack policy. Learn more.

Notification options

[✓ No notification]

New Amazon SNS topic

Topic

Email

Existing Amazon SNS topic

Existing topic ARN

Timeout Minutes

Rollback on failure [✓ Yes] No

Stack policy Enter policy

[✓ Upload policy file]

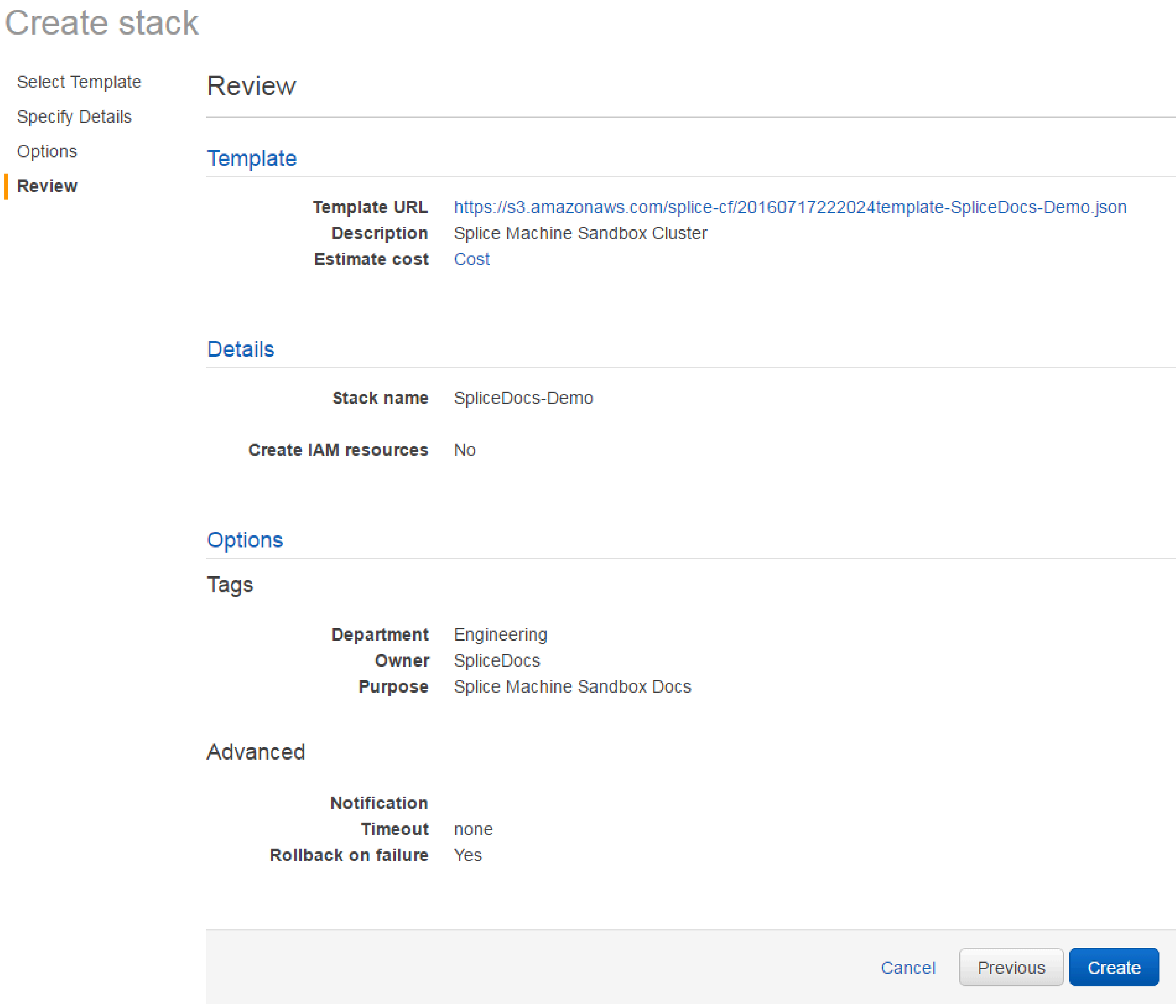
Choose File No file chosen

Learn more

Click the `Next` button at the bottom of the *Options* screen to proceed to the *Review* screen.

Review Your Sandbox Stack Configuration

Finally, review your stack configuration, and then click the `Create` button to start creating your sandbox cluster.



Template

Template URL	https://s3.amazonaws.com/splice-cf/20160717222024template-SpliceDocs-Demo.json
Description	Splice Machine Sandbox Cluster
Estimate cost	Cost

Details

Stack name	SpliceDocs-Demo
Create IAM resources	No

Options

Tags

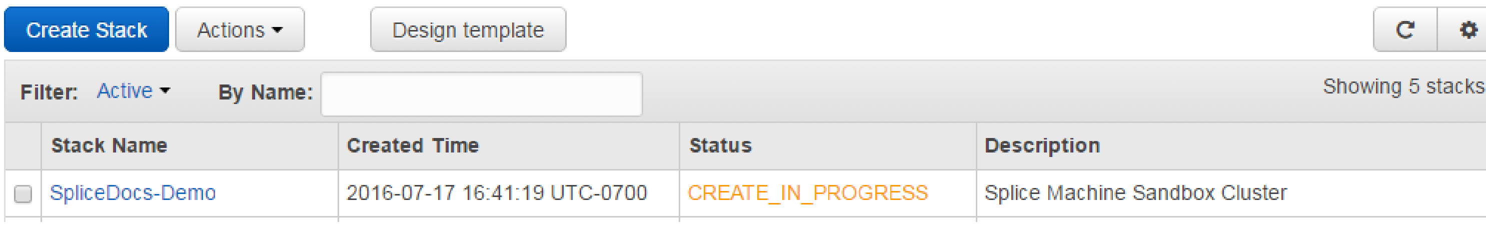
Department	Engineering
Owner	SpliceDocs
Purpose	Splice Machine Sandbox Docs

Advanced

Notification	
Timeout	none
Rollback on failure	Yes

Finish Launching Your Cluster

When AWS starts creating your cluster, you’ll see a progress screen like this:

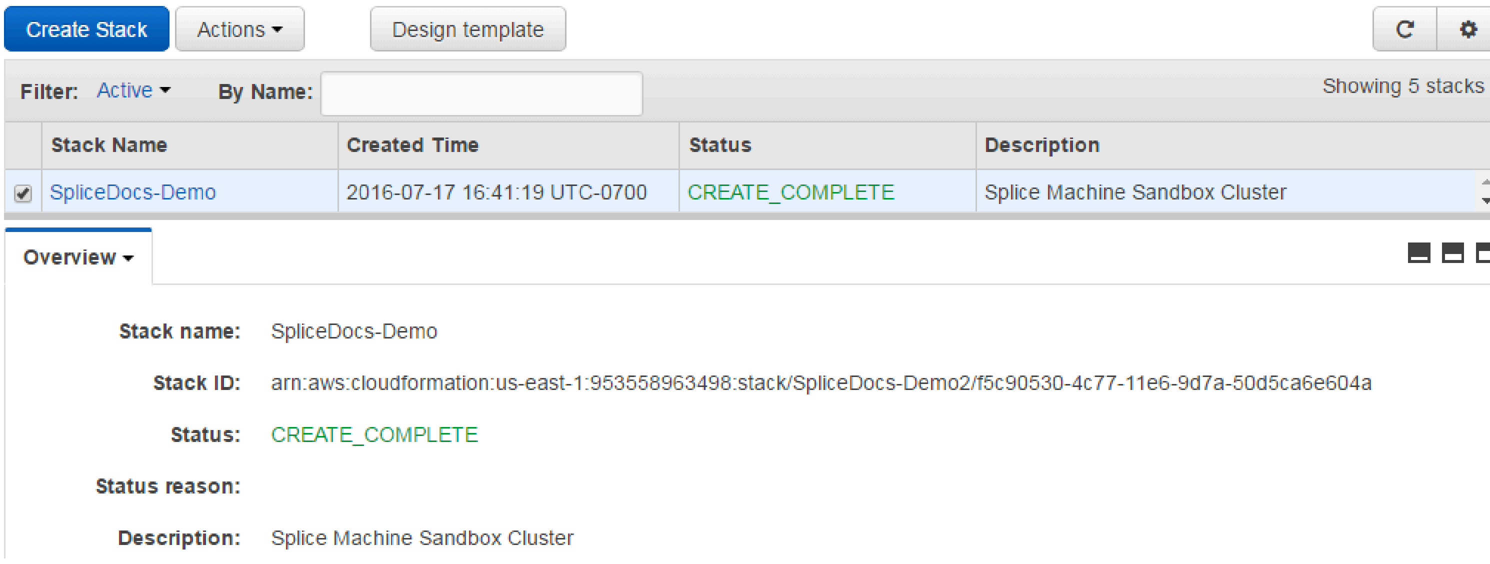


Create Stack | Actions | Design template

Filter: Active | By Name: | Showing 5 stacks

	Stack Name	Created Time	Status	Description
☐	SpliceDocs-Demo	2016-07-17 16:41:19 UTC-0700	CREATE_IN_PROGRESS	Splice Machine Sandbox Cluster

Depending on your configuration, the cluster may take some time to create. When it finishes, you'll see that the status has changed to complete:



Create Stack | Actions | Design template

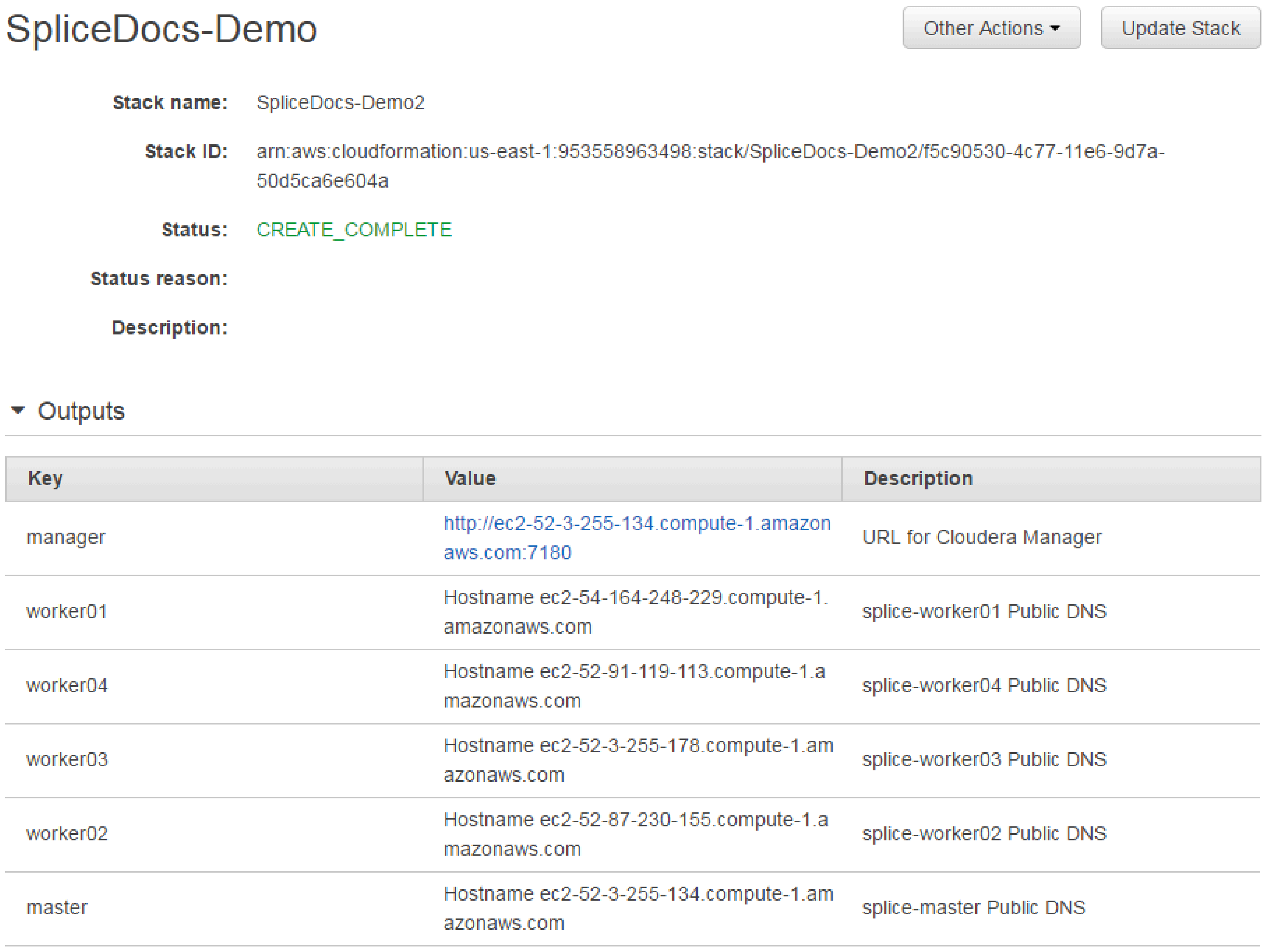
Filter: Active | By Name: | Showing 5 stacks

	Stack Name	Created Time	Status	Description
✓	SpliceDocs-Demo	2016-07-17 16:41:19 UTC-0700	CREATE_COMPLETE	Splice Machine Sandbox Cluster

Overview

Stack name:	SpliceDocs-Demo
Stack ID:	arn:aws:cloudformation:us-east-1:953558963498:stack/SpliceDocs-Demo2/f5c90530-4c77-11e6-9d7a-50d5ca6e604a
Status:	CREATE_COMPLETE
Status reason:	
Description:	Splice Machine Sandbox Cluster

After the status changes to complete, you can display your nodes by clicking the stack name link, and then viewing *Outputs*, which will look something like this:



SpliceDocs-Demo

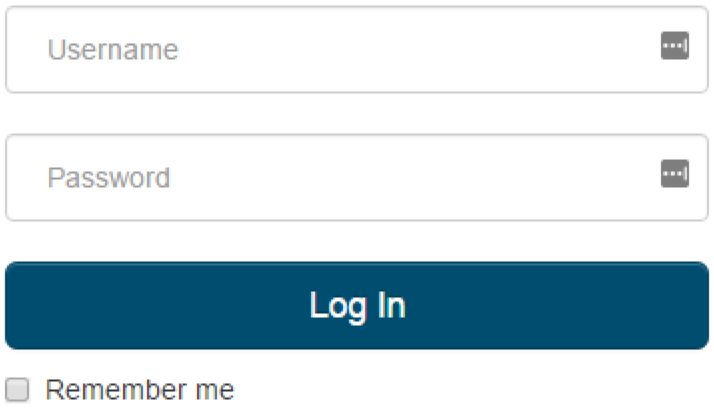
Stack name:	SpliceDocs-Demo2
Stack ID:	arn:aws:cloudformation:us-east-1:953558963498:stack/SpliceDocs-Demo2/f5c90530-4c77-11e6-9d7a-50d5ca6e604a
Status:	CREATE_COMPLETE
Status reason:	
Description:	

Outputs

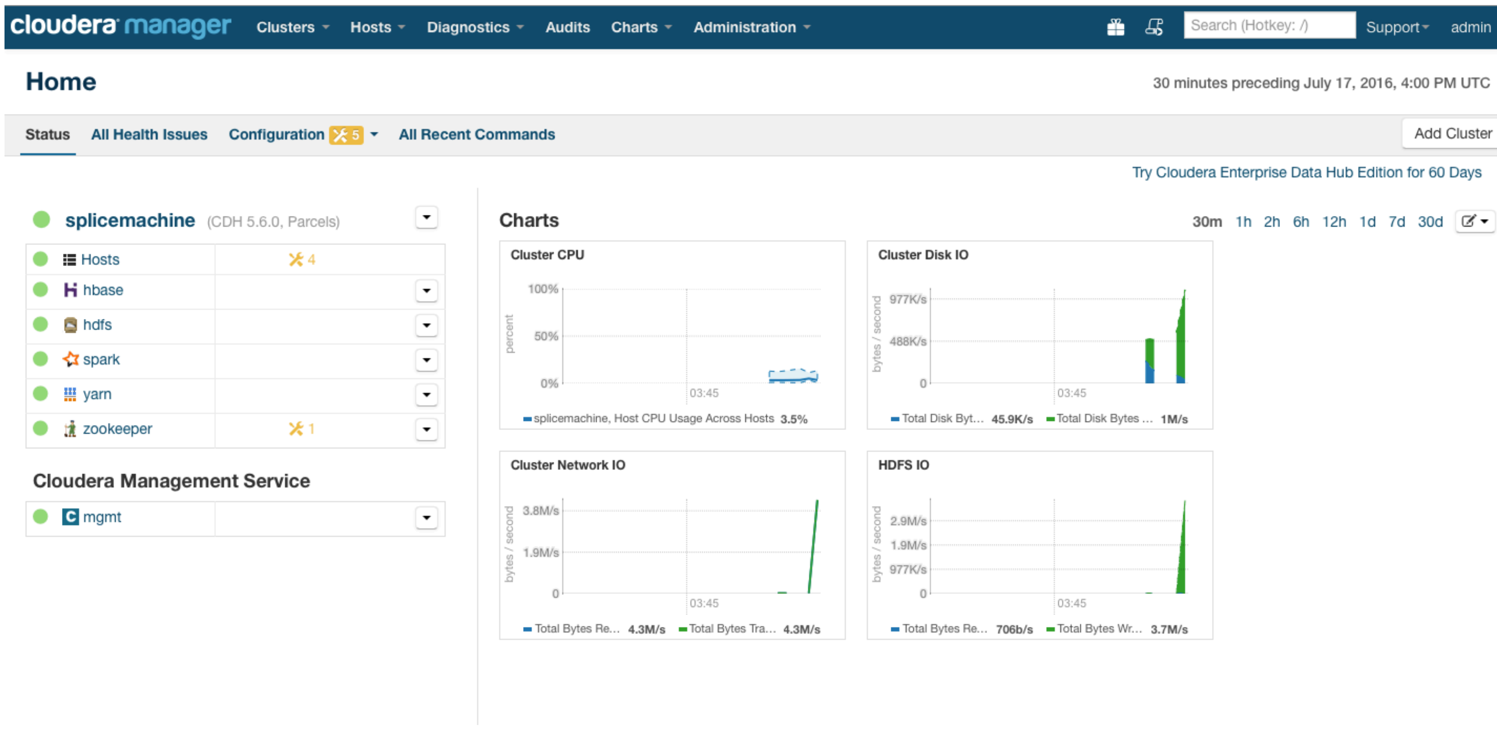
Key	Value	Description
manager	http://ec2-52-3-255-134.compute-1.amazonaws.com:7180	URL for Cloudera Manager
worker01	Hostname ec2-54-164-248-229.compute-1.amazonaws.com	splice-worker01 Public DNS
worker04	Hostname ec2-52-91-119-113.compute-1.amazonaws.com	splice-worker04 Public DNS
worker03	Hostname ec2-52-3-255-178.compute-1.amazonaws.com	splice-worker03 Public DNS
worker02	Hostname ec2-52-87-230-155.compute-1.amazonaws.com	splice-worker02 Public DNS
master	Hostname ec2-52-3-255-134.compute-1.amazonaws.com	splice-master Public DNS

Use Cloudera to Manage your Cluster

You can now log into Cloudera Manager for your cluster by clicking the manager link shown in the *Outputs* screen, which lands you on the login screen:



After logging in, you can use Cloudera to manage your Splice Machine cluster.



Using Splice Machine with Linux

You can now log into one of your sandbox nodes and use Splice Machine from a terminal window on your computer. To do so, `ssh` into the address of the node, which you can see in the Outputs screen from when you created the sandbox, or by viewing your cluster hosts in Cloudera Manager.

Make sure that you include the full name of your EC2 KeyPair when connecting. For example:

```
ssh -i ~/Downloads/splice-demo.pem.txt centos@ec2-54-164-248-229.compute-1.amazonaw
s.com
```

Once you're connected to Splice Machine, you can work with your database. For example:

```
~$ ssh -i ~/Downloads/splice-demo.pem.txt centos@ec2-54-164-248-229.compute-1.amazonaw
s.com
Last login: Sun Jul 17 23:58:17 2016 from 10.250.0.10
[centos@ip-10-250-0-11 ~]$ sqlshell.sh

 ========= rlwrap detected and enabled.  Use up and down arrow keys to scroll through co
mmand line history. ========

Running Splice Machine SQL shell
For help: "splice> help;"
SPLICE* - jdbc:splice://localhost:1527/splicedb
* = current connection
splice> show tables;
TABLE_SCHEM   |TABLE_NAME                |CONGLOM_ID|REMARKS
-----------------------------------------------------------------
SYS           |SYSALIASES                |272       |
SYS           |SYSBACKUP                 |912       |
SYS           |SYSBACKUPFILESET          |1024      |
SYS           |SYSBACKUPITEMS            |1200      |
SYS           |SYSBACKUPJOBS             |1232      |
SYS           |SYSCHECKS                 |288       |
SYS           |SYSCOLPERMS               |688       |
SYS           |SYSCOLUMNS                |80        |
SYS           |SYSCOLUMNSTATS            |1280      |
SYS           |SYSCONGLOMERATES          |48        |
SYS           |SYSCONSTRAINTS            |256       |
SYS           |SYSDEPENDS                |304       |
SYS           |SYSFILES                  |336       |
SYS           |SYSFOREIGNKEYS            |352       |
SYS           |SYSKEYS                   |240       |
SYS           |SYSPERMS                  |864       |
SYS           |SYSPHYSICALSTATS          |1296      |
SYS           |SYSPRIMARYKEYS            |368       |
SYS           |SYSROLES                  |800       |
SYS           |SYSROUTINEPERMS           |704       |
SYS           |SYSSCHEMAS                |32        |
SYS           |SYSSEQUENCES              |816       |
SYS           |SYSSTATEMENTS             |384       |
SYS           |SYSTABLEPERMS             |640       |
SYS           |SYSTABLES                 |64        |
SYS           |SYSTABLESTATS             |1312      |
SYS           |SYSTRIGGERS               |576       |
SYS           |SYSUSERS                  |880       |
SYS           |SYSVIEWS                  |320       |
SYSIBM        |SYSDUMMY1                 |1328      |

30 rows selected
splice>
```

Installing and Configuring Splice Machine for Cloudera Manager

This is an On-Premise-Only topic! Learn about our products

This topic describes installing and configuring Splice Machine on a Cloudera-managed cluster. Follow these steps:

1. Verify Prerequisites
2. Install the Splice Machine Parcel
3. Stop Hadoop Services
4. Make Cluster Modifications for Splice Machine
5. Configure Hadoop Services
6. Make any needed Optional Configuration Modifications
7. Deploy the Client Configuration
8. Restart the Cluster
9. Verify your Splice Machine Installation

Verify Prerequisites

Before starting your Splice Machine installation, please make sure that your cluster contains the prerequisite software components:

- A cluster running Cloudera Data Hub (CDH) with Cloudera Manager (CM)
- HBase installed
- HDFS installed
- YARN installed
- ZooKeeper installed

> **NOTE:** The specific versions of these components that you need depend on your operating environment, and are called out in detail in the Requirements topic of our *Getting Started Guide*.

Install the Splice Machine Parcel

Follow these steps to install CDH, Hadoop, Hadoop services, and Splice Machine on your cluster:

1. **Copy your parcel URL to the clipboard for use in the next step.**
   Which Splice Machine parcel URL you need depends upon which Splice Machine version you're installing and which version of CDH you are using. Here are the URLs for Splice Machine Release 2.7 and 2.5:

CDH Version	Parcel Type	Installer Package Link(s)
5.12.0	EL6	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.12.0/SPLICEMACHINE-2.5.0.1802.cdh5.12.0.p0.540-el6.parcel
	EL7	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.12.0/SPLICEMACHINE-2.5.0.1802.cdh5.12.0.p0.540-el7.parcel
	Precise	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.12.0/SPLICEMACHINE-2.5.0.1802.cdh5.12.0.p0.540-precise.parcel
	SLES11	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.12.0/SPLICEMACHINE-2.5.0.1802.cdh5.12.0.p0.540-sles11.parcel
	Trusty	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.12.0/SPLICEMACHINE-2.5.0.1802.cdh5.12.0.p0.540-trusty.parcel
	Wheezy	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.12.0/SPLICEMACHINE-2.5.0.1802.cdh5.12.0.p0.540-wheezy.parcel
5.8.3	EL6	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.8.3/SPLICEMACHINE-2.5.0.1802.cdh5.8.3.p0.540-el6.parcel
	EL7	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.8.3/SPLICEMACHINE-2.5.0.1802.cdh5.8.3.p0.540-el7.parcel
	Precise	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.8.3/SPLICEMACHINE-2.5.0.1802.cdh5.8.3.p0.540-precise.parcel
	SLES11	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.8.3/SPLICEMACHINE-2.5.0.1802.cdh5.8.3.p0.540-sles11.parcel
	Trusty	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.8.3/SPLICEMACHINE-2.5.0.1802.cdh5.8.3.p0.540-trusty.parcel
	Wheezy	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/parcel/cdh5.8.3/SPLICEMACHINE-2.5.0.1802.cdh5.8.3.p0.540-wheezy.parcel

To be sure that you have the latest URL, please check the Splice Machine Community site or contact your Splice Machine representative.

2. **Add the parcel repository**

   a. Make sure the `Use Parcels (Recommended)` option and the `Matched release` option are both selected.

   b. Click the `Continue` button to land on the *More Options* screen.

   c. Cick the `+` button for the `Remote Parcel Repository URLs` field. Paste your Splice Machine repository URL into this field.

3. **Use Cloudera Manager to install the parcel.**

4. **Verify that the parcel has been distributed and activated.**
   The Splice Machine parcel is identified as `SPLICEMACHINE` in the Cloudera Manager user interface. Make sure that this parcel has been downloaded, distributed, and activated on your cluster.

5. **Restart and redeploy any client changes when Cloudera Manager prompts you.**
   </div>

Stop Hadoop Services

As a first step, we stop cluster services to allow our installer to make changes that require the cluster to be temporarily inactive.

From the Cloudera Manager home screen, click the drop-down arrow next to the cluster on

1. **Select your cluster in Cloudera Manager**
   Click the drop-down arrow next to the name of the cluster on which you are installing Splice Machine.

2. **Stop the cluster**
   Click the `Stop` button.

Make Cluster Modifications for Splice Machine

Splice Machine requires a few modifications at the file system level to work properly on a CDH cluster:

1. **Install updated Java Servlet library:**
   You need to install an updated `javax.servlet-api` library so that Splice Machine can use Spark 2.0.x functionality in YARN.

2. **Remove Spark 1.6.x libraries**
   By default, Splice Machine version uses Spark 2.0. To avoid Spark version mismatches, we strongly recommend that you remove Spark 1.6x libraries from /opt/cloudera/parcels/CDH/jars/; however, if you need to retain Spark 1.6 for other applications, please contact our install team to help with your configuration.

3. **Run our script as *root* user on each node in your cluster to add symbolic links to the Splice Machine uber jar and YARN proxy jar into the YARN directories**
   Issue this command **on each node** in your cluster::
   ```
   sudo /opt/cloudera/parcels/SPLICEMACHINE/scripts/install-splice-symlinks.sh
   ```

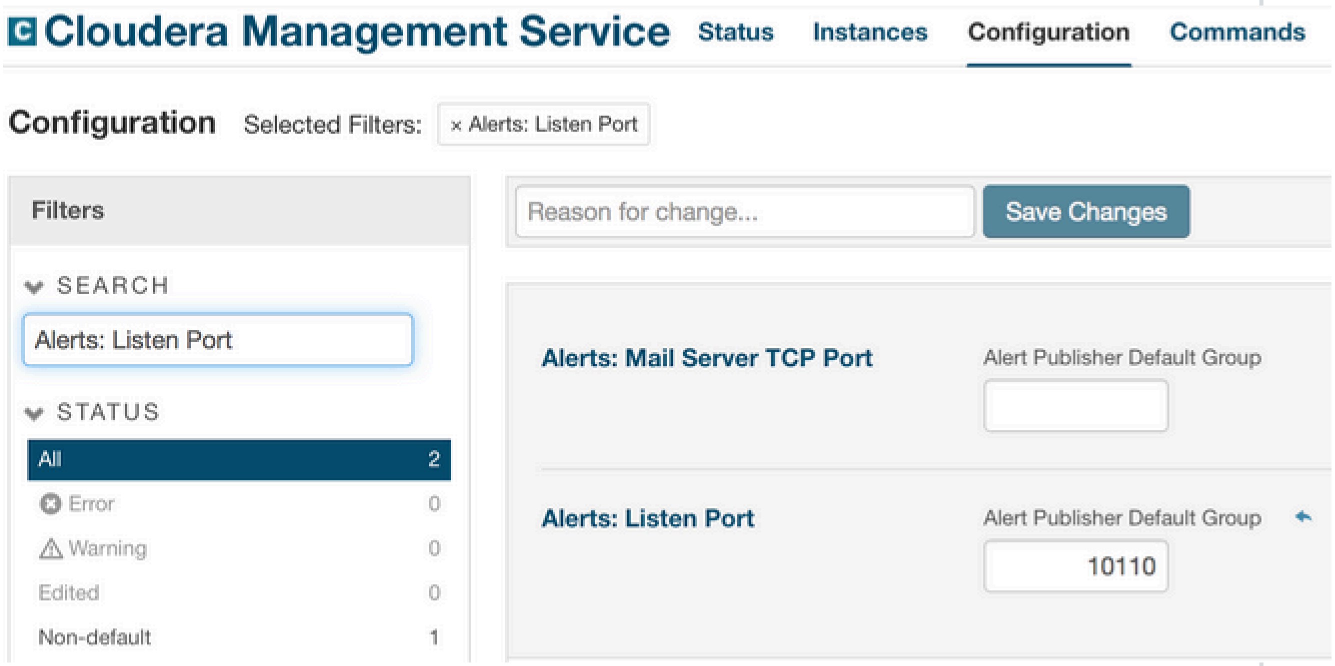
Configure Hadoop Services

Now it's time to make a few modifications in the Hadoop services configurations:

- Configure and Restart the Management Service
- Configure ZooKeeper
- Configure HDFS
- Configure YARN
- Configure HBASE

Configure and Restart the Management Service

1. Select the `Configuration` tab in CM:



Alerts: Mail Server TCP Port — Alert Publisher Default Group

Alerts: Listen Port — Alert Publisher Default Group: 10110

2. Change the value of the Alerts: Listen Port to `10110`.
3. Save changes and restart the Management Service.

Configure ZooKeeper

To edit the ZooKeeper configuration, click `ZooKeeper` in the Cloudera Manager (CM) home screen, then click the `Configuration` tab and follow these steps:

1. **Select the `Service-Wide` category.**
   Make the following changes:

   ```
   Maximum Client Connections = 0 Maximum Session Timeout = 120000
   ```

   Click the `Save Changes` button.

Configure HDFS

To edit the HDFS configuration, click `HDFS` in the Cloudera Manager home screen, then click the `Configuration` tab and make these changes:

1. **Verify that the HDFS data directories for your cluster are set up to use your data disks.**

2. **Change the values of these settings**

Setting	New Value
`Handler Count`	20
`Maximum Number of Transfer Threads`	8192
`NameNodeHandler Count`	64
`NameNode Service Handler Count`	60
`Replication Factor`	2 or 3 *
`Java Heap Size of DataNode in Bytes`	2 GB

3. Click the `Save Changes` button.

Configure YARN

To edit the YARN configuration, click `YARN` in the Cloudera Manager home screen, then click the `Configuration` tab and make these changes:

1. **Verify that the following directories are set up to use your data disks.**

```
NodeManager Local Directories
NameNode Data Directories
HDFS Checkpoint Directories
```

2. **Change the values of these settings**

Setting	New Value
`Heartbeat Interval`	100 ms

`MR Application Classpath`	`$HADOOP_MAPRED_HOME/*`
	`$HADOOP_MAPRED_HOME/lib/*`
	`$MR2_CLASSPATH/opt/cloudera/parcels/SPLICEMACHINE/lib/*`

`YARN Application Classpath`	`$HADOOP_CLIENT_CONF_DIR`
	`$HADOOP_CONF_DIR`
	`$HADOOP_COMMON_HOME/*`
	`$HADOOP_COMMON_HOME/lib/*`
	`$HADOOP_HDFS_HOME/*`
	`$HADOOP_HDFS_HOME/lib/*`
	`$HADOOP_YARN_HOME/*`
	`$HADOOP_YARN_HOME/lib/*`
	`$HADOOP_MAPRED_HOME/*`
	`$HADOOP_MAPRED_HOME/lib/*`
	`$MR2_CLASSPATH`
	`/opt/cloudera/parcels/CDH/lib/hbase/*`
	`/opt/cloudera/parcels/CDH/lib/hbase/lib/*`
	`/opt/cloudera/parcels/SPLICEMACHINE/lib/*`

`Localized Dir Deletion Delay`	86400
`JobHistory Server Max Log Size`	1 GB
`NodeManager Max Log Size`	1 GB
`ResourceManager Max Log Size`	1 GB
`Container Memory`	30 GB (based on node specs)
`Container Memory Maximum`	30 GB (based on node specs)

Setting	New Value
`Container Virtual CPU Cores`	19 (based on node specs)
`Container Virtual CPU Cores Maximum`	19 (Based on node specs)

3. **Add property values**
   You need to add the same two property values to each of four YARN advanced configuration settings.
   Add these properties:

XML Property Name	XML Property Value
`yarn.nodemanager.aux-services.spark_shuffle.class`	org.apache.spark.network.yarn.YarnShuffleService
`yarn.nodemanager.aux-services`	mapreduce_shuffle,spark_shuffle

   To each of these YARN settings:

   - Yarn Service Advanced Configuration Snippet (Safety Valve) for yarn-site.xml
   - Yarn Client Advanced Configuration Snippet (Safety Valve) for yarn-site.xml
   - NodeManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml
   - ResourceManager Advanced Configuration Snippet (Safety Valve) for yarn-site.xml

4. Click the `Save Changes` button.

Configure HBASE

To edit the HBASE configuration, click `HBASE` in the Cloudera Manager home screen, then click the `Configuration` tab and make these changes:

1. **Change the values of these settings**

Setting	New Value
HBase Client Scanner Caching	100 ms
Graceful Shutdown Timeout	30 seconds
HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml	The property list for the Safety Valve snippet is shown below, in Step 2
SplitLog Manager Timeout	5 minutes
Maximum HBase Client Retries	40
RPC Timeout	20 minutes (or 1200000 milliseconds)
HBase Client Pause	90
ZooKeeper Session Timeout	120000
HBase Master Web UI Port	16010
HBase Master Port	16000
Java Configuration Options for HBase Master	The HBase Master Java configuration options list is shown below, in Step 3
HBase Coprocessor Master Classes	com.splicemachine.hbase.SpliceMasterObserver
Java Heap Size of HBase Master in Bytes	5 GB
HStore Compaction Threshold	5

Setting	New Value
`HBase RegionServer Web UI port`	16030
`HStore Blocking Store Files`	20
`Java Configuration Options for HBase RegionServer`	The HBase RegionServerJava configuration options list is shown below, in Step 4
`HBase Memstore Block Multiplier`	4
`Maximum Number of HStoreFiles Compaction`	7
`HBase RegionServer Lease Period`	20 minutes (or 1200000 milliseconds)
`HFile Block Cache Size`	0.25
`Java Heap Size of HBase RegionServer in Bytes`	24 GB
`HBase RegionServer Handler Count`	200
`HBase RegionServer Meta-Handler Count`	200
`HBase Coprocessor Region Classes`	com.splicemachine.hbase.MemstoreAwareObserver com.splicemachine.derby.hbase.SpliceIndexObserver com.splicemachine.derby.hbase.SpliceIndexEndpoint com.splicemachine.hbase.RegionSizeEndpoint com.splicemachine.si.data.hbase.coprocessor.TxnLifecycleEndpoint com.splicemachine.si.data.hbase.coprocessor.SIObserver com.splicemachine.hbase.BackupEndpointObserver
`Maximum number of Write-Ahead Log (WAL) files`	48

Setting	New Value
`RegionServer Small Compactions Thread Count`	4
`HBase RegionServer Port`	16020
`Per-RegionServer Number of WAL Pipelines`	16

2. Set the value of `HBase Service Advanced Configuration Snippet (Safety Valve)` for `hbase-site.xml`:

```
<property><name>dfs.client.read.shortcircuit.buffer.size</name><value>131
072</value></property>
<property><name>hbase.balancer.period</name><value>60000</value></propert
y>
<property><name>hbase.client.ipc.pool.size</name><value>10</value></prope
rty>
<property><name>hbase.client.max.perregion.tasks</name><value>100</valu
e></property>
<property><name>hbase.coprocessor.regionserver.classes</name><value>com.s
plicemachine.hbase.RegionServerLifecycleObserver</value></property><prope
rty><name>hbase.hstore.defaultengine.compactionpolicy.class</name><valu
e>com.splicemachine.compactions.SpliceDefaultCompactionPolicy</value></pr
operty>
<property><name>hbase.hstore.defaultengine.compactor.class</name><value>c
om.splicemachine.compactions.SpliceDefaultCompactor</value></property>
<property><name>hbase.htable.threads.max</name><value>96</value></propert
y>
<property><name>hbase.ipc.warn.response.size</name><value>-1</value></pro
perty>
<property><name>hbase.ipc.warn.response.time</name><value>-1</value></pro
perty>
<property><name>hbase.master.loadbalance.bytable</name><value>true</valu
e></property>
<property><name>hbase.mvcc.impl</name><value>org.apache.hadoop.hbase.regi
onserver.SIMultiVersionConsistencyControl</value></property>
<property><name>hbase.regions.slop</name><value>0.01</value></property>
<property><name>hbase.regionserver.global.memstore.size.lower.limit</nam
e><value>0.9</value></property>
<property><name>hbase.regionserver.global.memstore.size</name><value>0.2
5</value></property>
<property><name>hbase.regionserver.maxlogs</name><value>48</value></prope
rty>
<property><name>hbase.regionserver.wal.enablecompression</name><value>tru
e</value></property>
<property><name>hbase.rowlock.wait.duration</name><value>0</value></prope
rty>
<property><name>hbase.status.multicast.port</name><value>16100</value></p
roperty>
<property><name>hbase.wal.disruptor.batch</name><value>true</value></prop
erty>
<property><name>hbase.wal.provider</name><value>multiwal</value></propert
y>
<property><name>hbase.wal.regiongrouping.numgroups</name><value>16</valu
e></property>
<property><name>hbase.zookeeper.property.tickTime</name><value>6000</valu
e></property>
<property><name>hfile.block.bloom.cacheonwrite</name><value>true</valu
e></property>
```

```
<property><name>io.storefile.bloom.error.rate</name><value>0.005</valu
e></property>
<property><name>splice.client.numConnections</name><value>1</value></prop
erty>
<property><name>splice.client.write.maxDependentWrites</name><value>6000
0</value></property>
<property><name>splice.client.write.maxIndependentWrites</name><value>600
00</value></property>
<property><name>splice.compression</name><value>snappy</value></property>
<property><name>splice.marshal.kryoPoolSize</name><value>1100</value></pr
operty>
<property><name>splice.olap_server.clientWaitTime</name><value>900000</va
lue></property>
<property><name>splice.ring.bufferSize</name><value>131072</value></prope
rty>
<property><name>splice.splitBlockSize</name><value>67108864</value></prop
erty>
<property><name>splice.timestamp_server.clientWaitTime</name><value>12000
0</value></property>
<property><name>splice.txn.activeTxns.cacheSize</name><value>10240</valu
e></property>
<property><name>splice.txn.completedTxns.concurrency</name><value>128</va
lue></property>
<property><name>splice.txn.concurrencyLevel</name><value>4096</value></pr
operty>
<property><name>hbase.hstore.compaction.max.size</name><value>26004684
8</value></property>
<property><name>hbase.hstore.compaction.min.size</name><value>16777216</v
alue></property>
<property><name>hbase.hstore.compaction.min</name><value>5</value></prope
rty>
<property><name>hbase.regionserver.thread.compaction.large</name><valu
e>1</value></property>
<property><name>splice.authentication.native.algorithm</name><value>SHA-5
12</value></property>
<property><name>splice.authentication</name><value>NATIVE</value></proper
ty>
```

3. **Set the value of Java Configuration Options for HBase Master**
   If you're using version 2.2 or later of the Spark Shuffle service, set the Java Configuration Options for HBase Master to:

```
-XX:MaxPermSize=512M -XX:+HeapDumpOnOutOfMemoryError -XX:MaxDirectMemoryS
ize=2g -XX:+AlwaysPreTouch -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:C
MSInitiatingOccupancyFraction=70 -XX:+CMSParallelRemarkEnabled -Dcom.su
n.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremot
e.ssl=false -Dcom.sun.management.jmxremote.port=10101 -Dsplice.spark.enab
led=true -Dsplice.spark.app.name=SpliceMachine -Dsplice.spark.master=yar
n-client -Dsplice.spark.logConf=true -Dsplice.spark.yarn.maxAppAttempt
s=1 -Dsplice.spark.driver.maxResultSize=1g -Dsplice.spark.driver.cores=2
-Dsplice.spark.yarn.am.memory=1g -Dsplice.spark.dynamicAllocation.enable
d=true -Dsplice.spark.dynamicAllocation.executorIdleTimeout=120 -Dsplic
e.spark.dynamicAllocation.cachedExecutorIdleTimeout=120 -Dsplice.spark.dy
namicAllocation.minExecutors=0 -Dsplice.spark.dynamicAllocation.maxExecut
ors=12 -Dsplice.spark.io.compression.lz4.blockSize=32k -Dsplice.spark.kry
o.referenceTracking=false -Dsplice.spark.kryo.registrator=com.splicemachi
ne.derby.impl.SpliceSparkKryoRegistrator -Dsplice.spark.kryoserializer.bu
ffer.max=512m -Dsplice.spark.kryoserializer.buffer=4m -Dsplice.spark.loca
lity.wait=100 -Dsplice.spark.memory.fraction=0.5 -Dsplice.spark.schedule
r.mode=FAIR -Dsplice.spark.serializer=org.apache.spark.serializer.KryoSer
ializer -Dsplice.spark.shuffle.compress=false -Dsplice.spark.shuffle.fil
e.buffer=128k -Dsplice.spark.shuffle.service.enabled=true -Dsplice.spar
k.reducer.maxReqSizeShuffleToMem=134217728 -Dsplice.spark.yarn.am.extraLi
braryPath=/opt/cloudera/parcels/CDH/lib/hadoop/lib/native -Dsplice.spar
k.yarn.am.waitTime=10s -Dsplice.spark.yarn.executor.memoryOverhead=2048
-Dsplice.spark.driver.extraJavaOptions=-Dlog4j.configuration=file:/etc/sp
ark/conf/log4j.properties -Dsplice.spark.driver.extraLibraryPath=/opt/clo
udera/parcels/CDH/lib/hadoop/lib/native -Dsplice.spark.driver.extraClassP
ath=/opt/cloudera/parcels/CDH/lib/hbase/conf:/opt/cloudera/parcels/CDH/ja
rs/htrace-core-3.1.0-incubating.jar -Dsplice.spark.executor.extraLibraryP
ath=/opt/cloudera/parcels/CDH/lib/hadoop/lib/native -Dsplice.spark.execut
or.extraClassPath=/opt/cloudera/parcels/CDH/lib/hbase/conf:/opt/cloudera/
parcels/CDH/jars/htrace-core-3.1.0-incubating.jar -Dsplice.spark.ui.retai
nedJobs=100 -Dsplice.spark.ui.retainedStages=100 -Dsplice.spark.worker.u
i.retainedExecutors=100 -Dsplice.spark.worker.ui.retainedDrivers=100 -Dsp
lice.spark.streaming.ui.retainedBatches=100 -Dsplice.spark.executor.core
s=4 -Dsplice.spark.executor.memory=8g -Dspark.compaction.reserved.slot
s=4 -Dsplice.spark.eventLog.enabled=true -Dsplice.spark.eventLog.dir=hdf
s:///user/splice/history -Dsplice.spark.local.dir=/tmp -Dsplice.spark.yar
n.jars=/opt/cloudera/parcels/SPLICEMACHINE/lib/*
```

If you're using a version of the Spark Shuffle service earlier than 2.2, set the Java Configuration Options for HBase Master to this instead:

```
-XX:MaxPermSize=512M -XX:+HeapDumpOnOutOfMemoryError -XX:MaxDirectMemoryS
ize=2g -XX:+AlwaysPreTouch -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:C
MSInitiatingOccupancyFraction=70 -XX:+CMSParallelRemarkEnabled -Dcom.su
n.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremot
e.ssl=false -Dcom.sun.management.jmxremote.port=10101 -Dsplice.spark.enab
led=true -Dsplice.spark.app.name=SpliceMachine -Dsplice.spark.master=yar
n-client -Dsplice.spark.logConf=true -Dsplice.spark.yarn.maxAppAttempt
s=1 -Dsplice.spark.driver.maxResultSize=1g -Dsplice.spark.driver.cores=2
-Dsplice.spark.yarn.am.memory=1g -Dsplice.spark.dynamicAllocation.enable
d=true -Dsplice.spark.dynamicAllocation.executorIdleTimeout=120 -Dsplic
e.spark.dynamicAllocation.cachedExecutorIdleTimeout=120 -Dsplice.spark.dy
namicAllocation.minExecutors=0 -Dsplice.spark.dynamicAllocation.maxExecut
ors=12 -Dsplice.spark.io.compression.lz4.blockSize=32k -Dsplice.spark.kry
o.referenceTracking=false -Dsplice.spark.kryo.registrator=com.splicemachi
ne.derby.impl.SpliceSparkKryoRegistrator -Dsplice.spark.kryoserializer.bu
ffer.max=512m -Dsplice.spark.kryoserializer.buffer=4m -Dsplice.spark.loca
lity.wait=100 -Dsplice.spark.memory.fraction=0.5 -Dsplice.spark.schedule
r.mode=FAIR -Dsplice.spark.serializer=org.apache.spark.serializer.KryoSer
ializer -Dsplice.spark.shuffle.compress=false -Dsplice.spark.shuffle.fil
e.buffer=128k -Dsplice.spark.shuffle.service.enabled=true  -Dsplice.spar
k.yarn.am.extraLibraryPath=/opt/cloudera/parcels/CDH/lib/hadoop/lib/nativ
e -Dsplice.spark.yarn.am.waitTime=10s -Dsplice.spark.yarn.executor.memory
Overhead=2048 -Dsplice.spark.driver.extraJavaOptions=-Dlog4j.configuratio
n=file:/etc/spark/conf/log4j.properties -Dsplice.spark.driver.extraLibrar
yPath=/opt/cloudera/parcels/CDH/lib/hadoop/lib/native -Dsplice.spark.driv
er.extraClassPath=/opt/cloudera/parcels/CDH/lib/hbase/conf:/opt/cloudera/
parcels/CDH/jars/htrace-core-3.1.0-incubating.jar -Dsplice.spark.executo
r.extraLibraryPath=/opt/cloudera/parcels/CDH/lib/hadoop/lib/native -Dspli
ce.spark.executor.extraClassPath=/opt/cloudera/parcels/CDH/lib/hbase/con
f:/opt/cloudera/parcels/CDH/jars/htrace-core-3.1.0-incubating.jar -Dsplic
e.spark.ui.retainedJobs=100 -Dsplice.spark.ui.retainedStages=100 -Dsplic
e.spark.worker.ui.retainedExecutors=100 -Dsplice.spark.worker.ui.retained
Drivers=100 -Dsplice.spark.streaming.ui.retainedBatches=100 -Dsplice.spar
k.executor.cores=4 -Dsplice.spark.executor.memory=8g -Dspark.compaction.r
eserved.slots=4 -Dsplice.spark.eventLog.enabled=true -Dsplice.spark.event
Log.dir=hdfs:///user/splice/history -Dsplice.spark.local.dir=/tmp -Dsplic
e.spark.yarn.jars=/opt/cloudera/parcels/SPLICEMACHINE/lib/*
```

4. Set the value of Java Configuration Options for Region Servers:

```
-XX:+HeapDumpOnOutOfMemoryError -XX:MaxDirectMemorySize=2g -XX:MaxPermSiz
e=512M -XX:+AlwaysPreTouch -XX:+UseG1GC -XX:MaxNewSize=4g -XX:InitiatingH
eapOccupancyPercent=60 -XX:ParallelGCThreads=24 -XX:+ParallelRefProcEnabl
ed -XX:MaxGCPauseMillis=5000 -Dcom.sun.management.jmxremote.authenticat
e=false -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmx
remote.port=10102
```

5. Click the `Save Changes` button.

Optional Configuration Modifications

There are a few configuration modifications you might want to make:

- Modify the Authentication Mechanism if you want to authenticate users with something other than the default *native authentication* mechanism.
- Modify the Log Location if you want your Splice Machine log entries stored somewhere other than in the logs for your region servers.

Modify the Authentication Mechanism

Splice Machine installs with Native authentication configured; native authentication uses the `sys.sysusers` table in the `splice` schema for configuring user names and passwords.

You can disable authentication or change the authentication mechanism that Splice Machine uses to LDAP by following the simple instructions in Configuring Splice Machine Authentication

You can use Cloudera’s Kerberos Wizard to enable Kerberos mode on a CDH5.8.x cluster. If you’re enabling Kerberos, you need to add this option to your HBase Master Java Configuration Options:

```
-Dsplice.spark.hadoop.fs.hdfs.impl.disable.cache=true
```

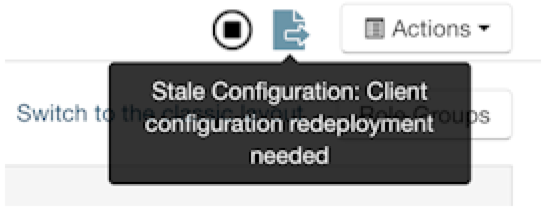
Modify the Log Location

Splice Machine logs all SQL statements by default, storing the log entries in your region server’s logs, as described in our Using Logging topic. You can modify where Splice Machine stroes logs by adding the following snippet to your *RegionServer Logging Advanced Configuration Snippet (Safety Valve)* section of your HBase Configuration:

```
log4j.appender.spliceDerby=org.apache.log4j.FileAppender
log4j.appender.spliceDerby.File=${hbase.log.dir}/splice-derby.log
log4j.appender.spliceDerby.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.spliceDerby.layout.ConversionPattern=%d{EEE MMM d HH:mm:ss,SSS} Threa
d[%t] %m%n

log4j.appender.spliceStatement=org.apache.log4j.FileAppender
log4j.appender.spliceStatement.File=${hbase.log.dir}/splice-statement.log
log4j.appender.spliceStatement.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.spliceStatement.layout.ConversionPattern=%d{EEE MMM d HH:mm:ss,SSS} T
hread[%t] %m%n

log4j.logger.splice-derby=INFO, spliceDerby
log4j.additivity.splice-derby=false

# Uncomment to log statements to a different file:
#log4j.logger.splice-derby.statement=INFO, spliceStatement
# Uncomment to not replicate statements to the spliceDerby file:
#log4j.additivity.splice-derby.statement=false
```

Deploy the Client Configuration

Now that you've updated your configuration information, you need to deploy it throughout your cluster. You should see a small notification in the upper right corner of your screen that looks like this:



Stale Configuration: Client configuration redeployment needed

To deploy your configuration:

1. **Click the notification.**
2. **Click the `Deploy Client Configuration` button.**
3. **When the deployment completes, click the `Finish` button.**

Restart the Cluster

As a first step, we stop the services that we're about to configure from the Cloudera Manager home screen:

1. **Restart ZooKeeper**
   Select `Start` from the `Actions` menu in the upper right corner of the ZooKeeper `Configuration` tab to restart ZooKeeper.

2. **Restart HDFS**
   Click the `HDFS Actions` drop-down arrow associated with (to the right of) HDFS in the cluster summary section of the Cloudera Manager home screen, and then click `Start` to restart HDFS.

   Use your terminal window to create these directories (if they are not already available in HDFS):

   ```
   sudo -iu hdfs hadoop fs -mkdir -p hdfs:///user/hbase hdfs:///user/splice/
   history
   sudo -iu hdfs hadoop fs -chown -R hbase:hbase hdfs:///user/hbase hdf
   s:///user/splice
   sudo -iu hdfs hadoop fs -chmod 1777 hdfs:///user/splice hdfs:///user/spli
   ce/history
   ```

3. **Restart YARN**
   Click the `YARN Actions` drop-down arrow associated with (to the right of) YARN in the cluster summary section of the Cloudera Manager home screen, and then click `Start` to restart YARN.

4. **Restart HBase**
   Click the `HBASE Actions` drop-down arrow associated with (to the right of) HBASE in the cluster summary section of the Cloudera Manager home screen, and then click `Start` to restart HBase.

Verify your Splice Machine Installation

Now start using the Splice Machine command line interpreter, which is referred to as *the splice prompt* or simply `splice>` by launching the `sqlshell.sh` script on any node in your cluster that is running an HBase region server.

NOTE: The command line interpreter defaults to connecting on port `1527` on `localhost`, with username `splice`, and password `admin`. You can override these defaults when starting the interpreter, as described in the Command Line (splice>) Reference topic in our *Developer's Guide*.

Now try entering a few sample commands you can run to verify that everything is working with your Splice Machine installation.

Operation	Command to perform operation
Display tables	`splice> show tables;`

Operation	Command to perform operation
Create a table	`splice> create table test (i int);`
Add data to the table	`splice> insert into test values 1,2,3,4,5;`
Query data in the table	`splice> select * from test;`
Drop the table	`splice> drop table test;`
List available commands	`splice> help;`
Exit the command line interpreter	`splice> exit;`
Make sure you end each command with a semicolon (`;`), followed by the *Enter* key or *Return* key	

See the Command Line (splice>) Reference section of our *Developer's Guide* for information about our commands and command syntax.

</div>

Installing and Configuring Splice Machine for Hortonworks HDP

This is an On-Premise-Only topic! Learn about our products

This topic describes installing and configuring Splice Machine on a Hortonworks Ambari-managed cluster. Follow these steps:

1. Verify Prerequisites
2. Download and Install Splice Machine
3. Stop Hadoop Services
4. Configure Hadoop Services
5. Start Any Additional Services
6. Make any needed Optional Configuration Modifications
7. Verify your Splice Machine Installation

Verify Prerequisites

Before starting your Splice Machine installation, please make sure that your cluster contains the prerequisite software components:

- A cluster running HDP
- Ambari installed and configured for HDP
- HBase installed
- HDFS installed
- YARN installed
- ZooKeeper installed
- Ensure that Phoenix services are **NOT** installed on your cluster, as they interfere with Splice Machine HBase settings.

NOTE: The specific versions of these components that you need depend on your operating environment, and are called out in detail in the Requirements topic of our *Getting Started Guide*.

Download and Install Splice Machine

Perform the following steps **on each node** in your cluster:

1. **Download the installer for your version.**
   Which Splice Machine installer (gzip) package you need depends upon which Splice Machine version you're installing and which version of HDP you are using. Here are the URLs for Splice Machine Release 2.7 and 2.5:

HDP Version	Installer Package Link	
2.5.0	**2.5.5**	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/installer/hdp2.5.5/SPLICEMACHINE-2.5.0.1802.hdp2.5.5.p0.540.tar.gz

   To be sure that you have the latest URL, please check the Splice Machine Community site or contact your Splice Machine representative.

2. **Create the `splice` installation directory:**

```
sudo mkdir -p /opt/splice
```

3. **Download the Splice Machine package into the `splice` directory on the node. For example:**

```
sudo curl '////SPLICEMACHINE-..' -o /opt/splice/SPLICEMACHINE-..
```

4. **Extract the Splice Machine package:**

```
sudo tar -xf SPLICEMACHINE-.. --directory /opt/splice
```

5. **Run our script as *root* user on each node in your cluster to add symbolic links to set up Splice Machine jar script symbolic links**
   Issue this command **on each node** in your cluster:

```
sudo /opt/splice/default/scripts/install-splice-symlinks.sh
```

Stop Hadoop Services

As a first step, we stop cluster services to allow our installer to make changes that require the cluster to be temporarily inactive.

1. **Access the Ambari Home Screen**

2. **Click the `Actions` drop-down in the Ambari *Services* sidebar, and then click the `Stop All` button.**

Configure Hadoop Services

Now it's time to make a few modifications in the Hadoop services configurations:

- Configure and Restart ZooKeeper
- Configure and Restart HDFS
- Configure and Restart YARN
- Configure MapReduce2
- Configure and Restart HBASE

Configure and Restart ZooKeeper

To edit the ZooKeeper configuration, click `ZooKeeper` in the Ambari *Services* sidebar. Then click the `Configs` tab and follow these steps:

1. **Click the *Custom zoo.cfg* drop-down arrow, then click `Add Property` to add the `maxClientCnxns` property and then again to add the `maxSessionTimeout` property, with these values:**

   ```
   maxClientCnxns=0
   maxSessionTimeout=120000
   ```

2. **Click the `Save` button to save your changes. You'll be prompted to optionally add a note such as `Updated ZooKeeper configuration for Splice Machine`. Click `Save` again.**

3. **Select the `Actions` drop-down in the Ambari *Services* sidebar, then click the `Start` action to start ZooKeeper. Wait for the restart to complete.**

Configure and Restart HDFS

To edit the HDFS configuration, click `HDFS` in the Ambari *Services* sidebar. Then click the `Configs` tab and follow these steps:

1. **Edit the HDFS configuration as follows:**

`NameNode Java heap size`	4 GB
`DataNode maximum Java heap size`	2 GB
`Block replication`	2 (for clusters with less than 8 nodes) 3 (for clusters with 8 or more nodes)

2. **Add a new property:**
   Click `Add Property...` under `Custom hdfs-site`, and add the following property:

   ```
   dfs.datanode.handler.count=20
   ```

3. **Save Changes**
   Click the `Save` button to save your changes. You'll be prompted to optionally add a note such as `Updated HDFS configuration for Splice Machine`. Click `Save` again.

4. **Start HDFS**
   After you save your changes, you'll land back on the `HDFS Service Configs` tab in Ambari.

   Click the `Actions` drop-down in the Ambari *Services* sidebar, then click the `Start` action to start HDFS. Wait for the restart to complete.

5. **Create directories for hbase user and the Splice Machine YARN application:**
   Use your terminal window to create these directories:

   ```
   sudo -iu hdfs hadoop fs -mkdir -p hdfs:///user/hbase hdfs:///user/splice/
   history
   sudo -iu hdfs hadoop fs -chown -R hbase:hbase hdfs:///user/hbase hdf
   s:///user/splice
   sudo -iu hdfs hadoop fs -chmod 1777 hdfs:///user/splice hdfs:///user/spli
   ce/history
   ```

Configure and Restart YARN

To edit the YARN configuration, click `YARN` in the Ambari *Services* sidebar. Then click the `Configs` tab and follow these steps:

1. **Update these other configuration values:**

Setting	New Value
`yarn.application.classpath`	`$HADOOP_CONF_DIR,/usr/hdp/current/hadoop-client/*,/usr/hdp/current/hadoop-client/lib/*,/usr/hdp/current/hadoop-hdfs-client/*,/usr/hdp/current/hadoop-hdfs-client/lib/*,/usr/hdp/current/hadoop-yarn-client/*,/usr/hdp/current/hadoop-yarn-client/lib/*,/usr/hdp/current/hadoop-mapreduce-client/*,/usr/hdp/current/hadoop-mapreduce-client/lib/*,/usr/hdp/current/hbase-regionserver/*,/usr/hdp/current/hbase-regionserver/lib/*,/opt/splice/default/lib/*`
`yarn.nodemanager.aux-services.spark2_shuffle.classpath`	/opt/splice/default/lib/*
`yarn.nodemanager.aux-services.spark_shuffle.classpath`	/opt/splice/default/lib/*
`yarn.nodemanager.aux-services.spark2_shuffle.class`	org.apache.spark.network.yarn.YarnShuffleService
`yarn.nodemanager.delete.debug-delay-sec`	86400
`Memory allocated for all YARN containers on a node`	30 GB (based on node specs)
`Minimum Container Size (Memory)`	1 GB (based on node specs)
`Minimum Container Size (Memory)`	30 GB (based on node specs)

2. **Save Changes**
   Click the `Save` button to save your changes. You'll be prompted to optionally add a note such as `Updated YARN configuration for Splice Machine`. Click `Save` again.

3. **Start YARN**
   After you save your changes, you'll land back on the `YARN Service Configs` tab in Ambari.

Open the `Service Actions` drop-down in the upper-right corner and select the `Start` action to start YARN. Wait for the restart to complete.

Configure MapReduce2

Ambari automatically sets these values for you:

- Map Memory
- Reduce Memory
- Sort Allocation Memory
- AppMaster Memory
- MR Map Java Heap Size
- MR Reduce Java Heap Size

You do, however, need to make a few property changes for this service.

To edit the MapReduce2 configuration, click `MapReduce2` in the Ambari *Services* sidebar. Then click the `Configs` tab and follow these steps:

1. **You need to replace `${hdp.version}` with the actual HDP version number you are using in these property values:**

   ```
   mapreduce.admin.map.child.java.opts
   mapreduce.admin.reduce.child.java.opts
   mapreduce.admin.user.env
   mapreduce.application.classpath
   mapreduce.application.framework.path
   yarn.app.mapreduce.am.admin-command-opts
   MR AppMaster Java Heap Size
   ```

   NOTE: An example of an HDP version number that you would substitute for `${hdp.version}` is `2.5.0.0-1245`.

2. **Click the `Save` button to save your changes. You'll be prompted to optionally add a note such as `Updated MapReduce2 configuration for Splice Machine`. Click `Save` again.**

3. **Select the `Actions` drop-down in the Ambari *Services* sidebar, then click the `Start` action to start MapReduce2. Wait for the restart to complete.**

Configure and Restart HBASE

To edit the HBase configuration, click `HBase` in the Ambari *Services* sidebar. Then click the `Configs` tab and follow these steps:

1. **Change the values of these settings**

Setting	New Value
`% of RegionServer Allocated to Write Buffer (hbase.regionserver.global.memstore.size)`	0.25
`HBase RegionServer Maximum Memory (hbase_regionserver_heapsize)`	24 GB
`% of RegionServer Allocated to Read Buffers (hfile.block.cache.size)`	0.25
`HBase Master Maximum Memory (hbase_master_heapsize)`	5 GB
`Number of Handlers per RegionServer (hbase.regionserver.handler.count)`	200
`HBase RegionServer Meta-Handler Count`	200
`HBase RPC Timeout`	1200000 (20 minutes)
`Zookeeper Session Timeout`	120000 (2 minutes)
`hbase.coprocessor.master.classes`	com.splicemachine.hbase.SpliceMasterObserver
`hbase.coprocessor.region.classes`	The value of this property is shown below, in Step 2
`Maximum Store Files before Minor Compaction (hbase.hstore.compactionThreshold)`	5
`Number of Fetched Rows when Scanning from Disk (hbase.client.scanner.caching)`	1000

Setting	New Value
`hstore blocking storefiles (hbase.hstore.blockingStoreFiles)`	20
`Advanced hbase-env`	The value of this property is shown below, in Step 3
`Custom hbase-site`	The value of this is shown below, in Step 4

2. **Set the value of the `hbase.coprocessor.region.classes` property to the following:**

```
com.splicemachine.hbase.MemstoreAwareObserver,com.splicemachine.derby.hba
se.SpliceIndexObserver,com.splicemachine.derby.hbase.SpliceIndexEndpoin
t,com.splicemachine.hbase.RegionSizeEndpoint,com.splicemachine.si.data.hb
ase.coprocessor.TxnLifecycleEndpoint,com.splicemachine.si.data.hbase.copr
ocessor.SIObserver,com.splicemachine.hbase.BackupEndpointObserver
```

3. **Under `Advanced hbase-env`, set the value of `hbase-env template` to the following:**

```
# Set environment variables here.

# The java implementation to use. Java 1.6 required.
export JAVA_HOME={{java64_home}}

# HBase Configuration directory
export HBASE_CONF_DIR=${HBASE_CONF_DIR:-{{hbase_conf_dir}}}

# Extra Java CLASSPATH elements. Optional.
export HBASE_CLASSPATH=${HBASE_CLASSPATH}
# add Splice Machine to the HBase classpath
SPLICELIBDIR="/opt/splice/default/lib"
APPENDSTRING=$(echo $(find ${SPLICELIBDIR} -maxdepth 1 -name \*.jar | sor
t) | sed 's/ /:/g')
export HBASE_CLASSPATH="${HBASE_CLASSPATH}:${APPENDSTRING}"

# The maximum amount of heap to use, in MB. Default is 1000.
# export HBASE_HEAPSIZE=1000

# Extra Java runtime options.
# Below are what we set by default. May only work with SUN JVM.
# For more on why as well as other possible settings,
# see http://wiki.apache.org/hadoop/PerformanceTuning
export SERVER_GC_OPTS="-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateSt
amps -Xloggc:{{log_dir}}/gc.log-`date +'%Y%m%d%H%M'`"
# Uncomment below to enable java garbage collection logging.
# export HBASE_OPTS="$HBASE_OPTS -verbose:gc -XX:+PrintGCDetails -XX:+Pri
ntGCDateStamps -Xloggc:$HBASE_HOME/logs/gc-hbase.log"

# Uncomment and adjust to enable JMX exporting
# See jmxremote.password and jmxremote.access in $JRE_HOME/lib/managemen
t to configure remote password access.
# More details at: http://java.sun.com/javase/6/docs/technotes/guides/man
agement/agent.html
#
# export HBASE_JMX_BASE="-Dcom.sun.management.jmxremote.ssl=false -Dcom.s
un.management.jmxremote.authenticate=false"
# If you want to configure BucketCache, specify '-XX: MaxDirectMemorySiz
e=' with proper direct memory size
# export HBASE_THRIFT_OPTS="$HBASE_JMX_BASE -Dcom.sun.management.jmxremot
e.port=10103"
# export HBASE_ZOOKEEPER_OPTS="$HBASE_JMX_BASE -Dcom.sun.management.jmxre
mote.port=10104"

# File naming hosts on which HRegionServers will run. $HBASE_HOME/conf/re
gionservers by default.
export HBASE_REGIONSERVERS=${HBASE_CONF_DIR}/regionservers
```

```
# Extra ssh options. Empty by default.
# export HBASE_SSH_OPTS="-o ConnectTimeout=1 -o SendEnv=HBASE_CONF_DIR"

# Where log files are stored. $HBASE_HOME/logs by default.
export HBASE_LOG_DIR={{log_dir}}

# A string representing this instance of hbase. $USER by default.
# export HBASE_IDENT_STRING=$USER

# The scheduling priority for daemon processes. See 'man nice'.
# export HBASE_NICENESS=10

# The directory where pid files are stored. /tmp by default.
export HBASE_PID_DIR={{pid_dir}}

# Seconds to sleep between slave commands. Unset by default. This
# can be useful in large clusters, where, e.g., slave rsyncs can
# otherwise arrive faster than the master can service them.
# export HBASE_SLAVE_SLEEP=0.1

# Tell HBase whether it should manage it's own instance of Zookeeper or n
ot.
export HBASE_MANAGES_ZK=false

export HBASE_OPTS="${HBASE_OPTS} -XX:ErrorFile={{log_dir}}/hs_err_pid%p.l
og -Djava.io.tmpdir={{java_io_tmpdir}}"
```

- If you’re using version 2.2 or later of the Spark Shuffle service, set these HBase Master option values:

```
export HBASE_MASTER_OPTS=”${HBASE_MASTER_OPTS} -Xms{{master_heapsize}}
-Xmx{{master_heapsize}} ${JDK_DEPENDED_OPTS}
-XX:+HeapDumpOnOutOfMemoryError -XX:MaxDirectMemorySize=2g
-XX:+AlwaysPreTouch -XX:+UseParNewGC -XX:+UseConcMarkSweepGC
-XX:CMSInitiatingOccupancyFraction=70 -XX:+CMSParallelRemarkEnabled
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.port=10101 -Dsplice.spark.enabled=true
-Dsplice.spark.app.name=SpliceMachine -Dsplice.spark.master=yarn-client
-Dsplice.spark.logConf=true -Dsplice.spark.yarn.maxAppAttempts=1
-Dsplice.spark.driver.maxResultSize=1g -Dsplice.spark.driver.cores=2
-Dsplice.spark.yarn.am.memory=1g
-Dsplice.spark.dynamicAllocation.enabled=true
-Dsplice.spark.dynamicAllocation.executorIdleTimeout=120
-Dsplice.spark.dynamicAllocation.cachedExecutorIdleTimeout=120
-Dsplice.spark.dynamicAllocation.minExecutors=0
-Dsplice.spark.dynamicAllocation.maxExecutors=12
-Dsplice.spark.io.compression.lz4.blockSize=32k
-Dsplice.spark.kryo.referenceTracking=false
```

```
-Dsplice.spark.kryo.registrator=com.splicemachine.derby.impl.SpliceSparkKr
yoRegistrator -Dsplice.spark.kryoserializer.buffer.max=512m
-Dsplice.spark.kryoserializer.buffer=4m -Dsplice.spark.locality.wait=100
-Dsplice.spark.memory.fraction=0.5 -Dsplice.spark.scheduler.mode=FAIR
-Dsplice.spark.serializer=org.apache.spark.serializer.KryoSerializer
-Dsplice.spark.shuffle.compress=false
-Dsplice.spark.shuffle.file.buffer=128k
-Dsplice.spark.shuffle.service.enabled=true
-Dsplice.spark.reducer.maxReqSizeShuffleToMem=134217728
-Dsplice.spark.yarn.am.extraLibraryPath=/usr/hdp/current/hadoop-client/
lib/native -Dsplice.spark.yarn.am.waitTime=10s
-Dsplice.spark.yarn.executor.memoryOverhead=2048
-Dsplice.spark.driver.extraJavaOptions=-Dlog4j.configuration=file:/etc/
spark/conf/log4j.properties -Dsplice.spark.driver.extraLibraryPath=/usr/
hdp/current/hadoop-client/lib/native
-Dsplice.spark.driver.extraClassPath=/usr/hdp/current/hbase-regionserver/
conf:/usr/hdp/current/hbase-regionserver/lib/htrace-
core-3.1.0-incubating.jar -Dsplice.spark.executor.extraLibraryPath=/usr/
hdp/current/hadoop-client/lib/native
-Dsplice.spark.executor.extraClassPath=/usr/hdp/current/hbase-
regionserver/conf:/usr/hdp/current/hbase-regionserver/lib/htrace-
core-3.1.0-incubating.jar -Dsplice.spark.ui.retainedJobs=100
-Dsplice.spark.ui.retainedStages=100
-Dsplice.spark.worker.ui.retainedExecutors=100
-Dsplice.spark.worker.ui.retainedDrivers=100
-Dsplice.spark.streaming.ui.retainedBatches=100
-Dsplice.spark.executor.cores=4 -Dsplice.spark.executor.memory=8g
-Dspark.compaction.reserved.slots=4 -Dsplice.spark.eventLog.enabled=true
-Dsplice.spark.eventLog.dir=hdfs:///user/splice/history
-Dsplice.spark.local.dir=/tmp -Dsplice.spark.yarn.jars=/opt/splice/
default/lib/*"
```

- If you're using a version of the Spark Shuffle service earlier than 2.2, set these HBase Master option values instead:

```
export HBASE_MASTER_OPTS="${HBASE_MASTER_OPTS} -Xms{{master_heapsize}}
-Xmx{{master_heapsize}} ${JDK_DEPENDED_OPTS}
-XX:+HeapDumpOnOutOfMemoryError -XX:MaxDirectMemorySize=2g
-XX:+AlwaysPreTouch -XX:+UseParNewGC -XX:+UseConcMarkSweepGC
-XX:CMSInitiatingOccupancyFraction=70 -XX:+CMSParallelRemarkEnabled
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.port=10101 -Dsplice.spark.enabled=true
-Dsplice.spark.app.name=SpliceMachine -Dsplice.spark.master=yarn-client
-Dsplice.spark.logConf=true -Dsplice.spark.yarn.maxAppAttempts=1
-Dsplice.spark.driver.maxResultSize=1g -Dsplice.spark.driver.cores=2
-Dsplice.spark.yarn.am.memory=1g
-Dsplice.spark.dynamicAllocation.enabled=true
```

```
-Dsplice.spark.dynamicAllocation.executorIdleTimeout=120
-Dsplice.spark.dynamicAllocation.cachedExecutorIdleTimeout=120
-Dsplice.spark.dynamicAllocation.minExecutors=0
-Dsplice.spark.dynamicAllocation.maxExecutors=12
-Dsplice.spark.io.compression.lz4.blockSize=32k
-Dsplice.spark.kryo.referenceTracking=false
-Dsplice.spark.kryo.registrator=com.splicemachine.derby.impl.SpliceSparkKr
yoRegistrator -Dsplice.spark.kryoserializer.buffer.max=512m
-Dsplice.spark.kryoserializer.buffer=4m -Dsplice.spark.locality.wait=100
-Dsplice.spark.memory.fraction=0.5 -Dsplice.spark.scheduler.mode=FAIR
-Dsplice.spark.serializer=org.apache.spark.serializer.KryoSerializer
-Dsplice.spark.shuffle.compress=false
-Dsplice.spark.shuffle.file.buffer=128k
-Dsplice.spark.shuffle.service.enabled=true
-Dsplice.spark.yarn.am.extraLibraryPath=/usr/hdp/current/hadoop-client/
lib/native -Dsplice.spark.yarn.am.waitTime=10s
-Dsplice.spark.yarn.executor.memoryOverhead=2048
-Dsplice.spark.driver.extraJavaOptions=-Dlog4j.configuration=file:/etc/
spark/conf/log4j.properties -Dsplice.spark.driver.extraLibraryPath=/usr/
hdp/current/hadoop-client/lib/native
-Dsplice.spark.driver.extraClassPath=/usr/hdp/current/hbase-regionserver/
conf:/usr/hdp/current/hbase-regionserver/lib/htrace-
core-3.1.0-incubating.jar -Dsplice.spark.executor.extraLibraryPath=/usr/
hdp/current/hadoop-client/lib/native
-Dsplice.spark.executor.extraClassPath=/usr/hdp/current/hbase-
regionserver/conf:/usr/hdp/current/hbase-regionserver/lib/htrace-
core-3.1.0-incubating.jar -Dsplice.spark.ui.retainedJobs=100
-Dsplice.spark.ui.retainedStages=100
-Dsplice.spark.worker.ui.retainedExecutors=100
-Dsplice.spark.worker.ui.retainedDrivers=100
-Dsplice.spark.streaming.ui.retainedBatches=100
-Dsplice.spark.executor.cores=4 -Dsplice.spark.executor.memory=8g
-Dspark.compaction.reserved.slots=4 -Dsplice.spark.eventLog.enabled=true
-Dsplice.spark.eventLog.dir=hdfs:///user/splice/history
-Dsplice.spark.local.dir=/tmp -Dsplice.spark.yarn.jars=/opt/splice/
default/lib/*"
```

4. **Finish updating of `hbase-env template` with the following:**

```
export HBASE_REGIONSERVER_OPTS="${HBASE_REGIONSERVER_OPTS} -Xmn{{regionse
rver_xmn_size}} -Xms{{regionserver_heapsize}} -Xmx{{regionserver_heapsiz
e}} ${JDK_DEPENDED_OPTS} -XX:+HeapDumpOnOutOfMemoryError -XX:MaxDirectMem
orySize=2g -XX:+AlwaysPreTouch -XX:+UseG1GC -XX:MaxNewSize=4g -XX:Initiat
ingHeapOccupancyPercent=60 -XX:ParallelGCThreads=24 -XX:+ParallelRefProcE
nabled -XX:MaxGCPauseMillis=5000 -Dcom.sun.management.jmxremote.authentic
ate=false -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.j
mxremote.port=10102"
# HBase off-heap MaxDirectMemorySize
export HBASE_REGIONSERVER_OPTS="$HBASE_REGIONSERVER_OPTS {% if hbase_ma
x_direct_memory_size %} -XX:MaxDirectMemorySize={{hbase_max_direct_memor
y_size}}m {% endif %}"
```

5. **In `Custom hbase-site` property, add the following properties:**

```
dfs.client.read.shortcircuit.buffer.size=131072
hbase.balancer.period=60000
hbase.client.ipc.pool.size=10
hbase.client.max.perregion.tasks=100
hbase.coprocessor.regionserver.classes=com.splicemachine.hbase.RegionServ
erLifecycleObserver
hbase.hstore.compaction.max.size=260046848
hbase.hstore.compaction.min.size=16777216
hbase.hstore.compaction.min=5
hbase.hstore.defaultengine.compactionpolicy.class=com.splicemachine.compa
ctions.SpliceDefaultCompactionPolicy
hbase.hstore.defaultengine.compactor.class=com.splicemachine.compaction
s.SpliceDefaultCompactor
hbase.htable.threads.max=96
hbase.ipc.warn.response.size=-1
hbase.ipc.warn.response.time=-1
hbase.master.loadbalance.bytable=TRUE
hbase.mvcc.impl=org.apache.hadoop.hbase.regionserver.SIMultiVersionConsis
tencyControl
hbase.regions.slop=0.01
hbase.regionserver.global.memstore.size.lower.limit=0.9
hbase.regionserver.lease.period=1200000
hbase.regionserver.maxlogs=48
hbase.regionserver.thread.compaction.large=1
hbase.regionserver.thread.compaction.small=4
hbase.regionserver.wal.enablecompression=TRUE
hbase.rowlock.wait.duration=0
hbase.splitlog.manager.timeout=3000
hbase.status.multicast.port=16100
hbase.wal.disruptor.batch=TRUE
hbase.wal.provider=multiwal
hbase.wal.regiongrouping.numgroups=16
hbase.zookeeper.property.tickTime=6000
hfile.block.bloom.cacheonwrite=TRUE
io.storefile.bloom.error.rate=0.005
splice.authentication.native.algorithm=SHA-512
splice.authentication=NATIVE
splice.client.numConnections=1
splice.client.write.maxDependentWrites=60000
splice.client.write.maxIndependentWrites=60000
splice.compression=snappy
splice.marshal.kryoPoolSize=1100
splice.olap_server.clientWaitTime=900000
splice.ring.bufferSize=131072
splice.splitBlockSize=67108864
splice.timestamp_server.clientWaitTime=120000
splice.txn.activeTxns.cacheSize=10240
splice.txn.completedTxns.concurrency=128
splice.txn.concurrencyLevel=4096
```

6. **Save Changes**
   Click the `Save` button to save your changes. You'll be prompted to optionally add a note such as `Updated HDFS configuration for Splice Machine`. Click `Save` again.

7. **Start HBase**
   After you save your changes, you'll land back on the HBase Service `Configs` tab in Ambari.

   Open the `Service Actions` drop-down in the upper-right corner and select the `Start` action to start HBase. Wait for the restart to complete.

Start any Additional Services

We started this installation by shutting down your cluster services, and then configured and restarted each individual service used by Splice Machine.

If you had any additional services running, such as Ambari Metrics, you need to restart each of those services.

Optional Configuration Modifications

There are a few configuration modifications you might want to make:

- Modify the Authentication Mechanism if you want to authenticate users with something other than the default *native authentication* mechanism.
- Modify the Log Location if you want your Splice Machine log entries stored somewhere other than in the logs for your region servers.
- Adjust the replication factor if you have a small cluster and need to improve resource usage or performance.

Modify the Authentication Mechanism

Splice Machine installs with Native authentication configured; native authentication uses the `sys.sysusers` table in the `splice` schema for configuring user names and passwords.

You can disable authentication or change the authentication mechanism that Splice Machine uses to LDAP by following the simple instructions in Configuring Splice Machine Authentication

If you're using Kerberos, you need to add this option to your HBase Master Java Configuration Options:

```
-Dsplice.spark.hadoop.fs.hdfs.impl.disable.cache=true
```

Modify the Log Location

Splice Machine logs all SQL statements by default, storing the log entries in your region server's logs, as described in our Using Logging topic. You can modify where Splice Machine stores logs by adding the following snippet to your *RegionServer Logging Advanced Configuration Snippet (Safety Valve)* section of your HBase Configuration:

```
log4j.appender.spliceDerby=org.apache.log4j.FileAppender
log4j.appender.spliceDerby.File=${hbase.log.dir}/splice-derby.log
log4j.appender.spliceDerby.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.spliceDerby.layout.ConversionPattern=%d{EEE MMM d HH:mm:ss,SSS} Threa
d[%t] %m%n

log4j.appender.spliceStatement=org.apache.log4j.FileAppender
log4j.appender.spliceStatement.File=${hbase.log.dir}/splice-statement.log
log4j.appender.spliceStatement.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.spliceStatement.layout.ConversionPattern=%d{EEE MMM d HH:mm:ss,SSS} T
hread[%t] %m%n

log4j.logger.splice-derby=INFO, spliceDerby
log4j.additivity.splice-derby=false

# Uncomment to log statements to a different file:
#log4j.logger.splice-derby.statement=INFO, spliceStatement
# Uncomment to not replicate statements to the spliceDerby file:
#log4j.additivity.splice-derby.statement=false
```

Verify your Splice Machine Installation

Now start using the Splice Machine command line interpreter, which is referred to as `the splice prompt` or simply `splice>` by launching the `sqlshell.sh` script on any node in your cluster that is running an HBase region server.

NOTE: The command line interpreter defaults to connecting on port `1527` on `localhost`, with username `splice`, and password `admin`. You can override these defaults when starting the interpreter, as described in the Command Line (splice>) Reference topic in our *Developer's Guide*.

Now try entering a few sample commands you can run to verify that everything is working with your Splice Machine installation.

Operation	Command to perform operation
Display tables	`splice> show tables;`

Operation	Command to perform operation
Create a table	`splice> create table test (i int);`
Add data to the table	`splice> insert into test values 1,2,3,4,5;`
Query data in the table	`splice> select * from test;`
Drop the table	`splice> drop table test;`
Exit the command line interpreter	`splice> exit;`
Make sure you end each command with a semicolon (`;`), followed by the *Enter* key or *Return* key	

See the Command Line (splice>) Reference section of our *Developer's Guide* for information about our commands and command syntax.

Installing and Configuring Splice Machine for MapR

This is an On-Premise-Only topic! Learn about our products

This topic describes installing and configuring Splice Machine on a MapR-managed cluster. Follow these steps:

1. Verify Prerequisites
2. Download and Install Splice Machine
3. Configure Cluster Services
4. Stop Cluster Services
5. Restart Cluster Services
6. Create the Splice Machine Event Log Directory
7. Make any needed Optional Configuration Modifications
8. Verify your Splice Machine Installation

MapR Secure Clusters Only Work with the Enterprise Edition of Splice Machine

MapR secure clusters do not support the Community Edition of Splice Machine. You can check this by:

1. Open the HBase Configuration page
2. Search the XML for the `hbase.security.authentication` setting
3. If the setting value is anything other than `simple`, you need the *Enterprise* edition of Splice Machine.

To read more about the additional features available in the Enterprise Edition, see our Splice Machine Editions page. To obtain a license for the Splice Machine *Enterprise Edition*, **please Contact Splice Machine Sales today.**

Verify Prerequisites

Before starting your Splice Machine installation, please make sure that your cluster contains the prerequisite software components:

- A cluster running MapR with MapR-FS
- HBase installed
- YARN installed

- ZooKeeper installed
- The latest `mapr-patch` should be installed to bring in all MapR-supplied platform patches, before proceeding with your Splice Machine installation.
- The MapR Ecosystem Package (MEP) should be installed on all cluster nodes, before proceeding with your Splice Machine installation.
- Ensure Spark services are **NOT** installed; Splice Machine cannot currently coexist with Spark 1.x on a cluster:
  - If `MEP version 1.x` is in use, you must remove the `Spark 1.x` packages from your cluster, as described below, in Removing Spark Packages from Your Cluster.
  - MEP version 2.0 bundles Spark 2.0, and will not cause conflicts with Splice Machine.

NOTE: The specific versions of these components that you need depend on your operating environment, and are called out in detail in the Requirements topic of our *Getting Started Guide*.

Removing Spark Packages from Your Cluster

To remove Spark packages from your cluster and update your configuration, follow these steps

1. **If you're running a Debian/Ubuntu-based Linux distribution:**

   ```
   dpkg -l | awk '/ii.*mapr-spark/{print $2}' | xargs sudo apt-get purge
   ```

   If you're running on a RHEL/CentOS-based Linux distribution:

   ```
   rpm -qa \*mapr-spark\* | xargs sudo yum -y erase
   ```

2. **Reconfigure node services in your cluster:**

   ```
   sudo /opt/mapr/server/configure.sh -R
   ```

Download and Install Splice Machine

Perform the following steps **on each node** in your cluster:

1. **Download the installer for your version.**
   Which Splice Machine installer (gzip) package you need depends upon which Splice Machine version you're installing and which version of MapR you are using. Here are the URLs for Splice Machine Release 2.7 and 2.5:

MapR Version	Installer Package Link	
2.5.0	5.2.0	https://s3.amazonaws.com/splice-releases/2.5.0.1802/cluster/installer/mapr5.2.0/SPLICEMACHINE-2.5.0.1802.mapr5.2.0.p0.540.tar.gz

To be sure that you have the latest URL, please check the Splice Machine Community site or contact your Splice Machine representative.

2. **Create the `splice` installation directory:**

   ```
   mkdir -p /opt/splice
   ```

3. **Download the Splice Machine package into the `splice` directory on the node. For example:**

   ```
   cd /opt/splicecurl -O '////SPLICEMACHINE-..'
   ```

4. **Extract the Splice Machine package:**

   ```
   tar -xf SPLICEMACHINE-..
   ```

5. **Create a symbolic link. For example:**

   ```
   ln -sf SPLICEMACHINE-.. default
   ```

6. **Run our script as *root* user on each node in your cluster to add symbolic links to the set up the Splice Machine jar and to script symbolic links:**
   Issue this command on each node in your cluster:

   ```
   sudo bash /opt/splice/default/scripts/install-splice-symlinks.sh
   ```

Configure Cluster Services

The scripts used in this section all assume that password-less `ssh` and password-less `sudo` are enabled across all cluster nodes. These scripts are designed to be run on the CLDB node in a cluster with only one CLDB node. If your cluster has multiple CLDB nodes, do not run these script; you need to change configuration settings manually **on each node** in the cluster. To do so, refer to the `*.patch` files in the `/opt/splice/default/conf` directory.

If you're running on a cluster with a single CLDB node, follow these steps:

1. **Tighten the ephemeral port range so HBase doesn't bump into it:**

   ```
   cd /opt/splice/default/scripts
   ./install-sysctl-conf.sh
   ```

2. **Update `hbase-site.xml`:**

   ```
   cd /opt/splice/default/scripts
   ./install-hbase-site-xml.sh
   ```

3. **Update `hbase-env.sh`:**

   ```
   cd /opt/splice/default/scripts
   ./install-hbase-env-sh.sh
   ```

4. **Update `yarn-site.xml`:**

   ```
   cd /opt/splice/default/scripts
   ./install-yarn-site-xml.sh
   ```

5. **Update `warden.conf`:**

   ```
   cd /opt/splice/default/scripts
   ./install-warden-conf.sh
   ```

6. **Update `zoo.cfg`:**

   ```
   cd /opt/splice/default/scripts
   ./install-zookeeper-conf.sh
   ```

Stop Cluster Services

You need to stop all cluster services to continue with installing Splice Machine. Run the following commands as `root` user **on each node** in your cluster

```
sudo service mapr-warden stopsudo service mapr-zookeeper stop
```

Restart Cluster Services

Once you've completed the configuration steps described above, start services on your cluster; run the following commands **on each node** in your cluster:

```
sudo service mapr-zookeeper startsudo service mapr-warden start
```

Create the Splice Machine Event Log Directory

Create the Splice Machine event log directory by executing the following commands on a single cluster node while MapR is up and running:

```
sudo -su mapr hadoop fs -mkdir -p /user/splice/historysudo -su mapr hadoop fs -chow
n -R mapr:mapr /user/splicesudo -su mapr hadoop fs -chmod 1777 /user/splice/history
```

Optional Configuration Modifications

There are a few configuration modifications you might want to make:

- Modify the Authentication Mechanism if you want to authenticate users with something other than the default *native authentication* mechanism.
- Modify the Log Location if you want your Splice Machine log entries stored somewhere other than in the logs for your region servers.
- Adjust the Replication Factor if you have a small cluster and need to improve resource usage or performance.

Modify the Authentication Mechanism

Splice Machine installs with Native authentication configured; native authentication uses the `sys.sysusers` table in the `splice` schema for configuring user names and passwords.

You can disable authentication or change the authentication mechanism that Splice Machine uses to LDAP by following the simple instructions in Command Line (splice>) Reference

Modify the Log Location

Splice Machine logs all SQL statements by default, storing the log entries in your region server's logs, as described in our Using Logging topic. You can modify where Splice Machine stores logs as follows:

1. Append the following configuration information to the `/opt/mapr/hbase/hbase-1.1.1/conf/log4jd/properties` file *on each node* in your cluster:

```
log4j.appender.spliceDerby=org.apache.log4j.FileAppender
log4j.appender.spliceDerby.File=${hbase.log.dir}/splice-derby.log
log4j.appender.spliceDerby.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.spliceDerby.layout.ConversionPattern=%d{EEE MMM d HH:mm:s
s,SSS} Thread[%t] %m%n

log4j.appender.spliceStatement=org.apache.log4j.FileAppender
log4j.appender.spliceStatement.File=${hbase.log.dir}/splice-statement.log
log4j.appender.spliceStatement.layout=org.apache.log4j.EnhancedPatternLay
out
log4j.appender.spliceStatement.layout.ConversionPattern=%d{EEE MMM d HH:m
m:ss,SSS} Thread[%t] %m%n

log4j.logger.splice-derby=INFO, spliceDerby
log4j.additivity.splice-derby=false

# Uncomment to log statements to a different file:
#log4j.logger.splice-derby.statement=INFO, spliceStatement
# Uncomment to not replicate statements to the spliceDerby file:
#log4j.additivity.splice-derby.statement=false
```

2. Use either of these methods to take the log changes live:

   a. Restart the MapR service on each node in your cluster:

   ```
   service mapr-warden restart
   ```

   b. OR, restart the HBase service *on each node* in your cluster by issuing these commands from your Master node:

   ```
   sudo -su mapr maprcli node services -hbmaster restart -nodes <master node>
   sudo -su mapr maprcli node services -hbregionserver restart -nodes <regional node 1> <regional node 2>
   … <regional node n>
   ```

Adjust the Replication Factor

The default namespace replication factor for Splice Machine is 3. If you're running a small cluster you may want to adjust the replication down to improve resource and performance drag. To do so in MapR, use the following command:

```
maprcli volume modify -nsreplication <value>
```

For more information, see the MapR documentation site, doc.mapr.com.

Verify your Splice Machine Installation

Now start using the Splice Machine command line interpreter, which is referred to as *the splice prompt* or simply `splice>` by launching the `sqlshell.sh` script on any node in your cluster that is running an HBase region server.

> **NOTE:** The command line interpreter defaults to connecting on port `1527` on `localhost`, with username `splice`, and password `admin`. You can override these defaults when starting the interpreter, as described in the Command Line (splice>) Reference topic in our *Developer's Guide*.

Now try entering a few sample commands you can run to verify that everything is working with your Splice Machine installation.

Operation	Command to perform operation
Display tables	`splice> show tables;`
Create a table	`splice> create table test (i int);`
Add data to the table	`splice> insert into test values 1,2,3,4,5;`
Query data in the table	`splice> select * from test;`
Drop the table	`splice> drop table test;`
List available commands	`splice> help;`
Exit the command line interpreter	`splice> exit;`
Make sure you end each command with a semicolon (`;`), followed by the *Enter* key or *Return* key	

See the Command Line (splice>) Reference section of our *Developer's Guide* for information about our commands and command syntax.

Installing the Standalone Version of Splice Machine

This is an On-Premise-Only topic! Learn about our products

This topic walks you through downloading, installing, and getting started with using the standalone version of Splice Machine.

About the Standalone Version

The standalone version of Splice Machine runs on a single computer, rather than on a cluster; it is intended to allow you to get your feet wet with Splice Machine. The standalone version installs quickly on your MacOS, Linux, or CentOS computer, and enables rapid access to Splice Machine's capabilities. It's an excellent way to experiment with Splice Machine with a reasonably small amount of data.

Note that many of our customers also use the standalone version of Splice Machine for ongoing development work. For example, if your engineers want to create stored procedures to optimize aspects of your database, they can develop and debug those procedures on the standalone version before deploying them to your cluster.

This topic walks you through getting the standalone version of Splice Machine installed on your computer. Because Splice Machine is a Java application, youll have to meet some prerequisites for the way your computer supports Java. You may already have created these settings for other Java applications, but just to be sure, we list them in this tutorial for each of the supported operating systems.

You can watch the video version of getting Splice Machine installed on your computer, or follow the written instructions.

NOTE: Many other Big Data applications and services like HBase, Kafka, Hadoop, Prometheus, and even parts of the ELK stack are also written (at least partially) in Java, and so they require these extra Java setup steps as well.

Watch the Video Version

This video walks you through setting up `$JAVA_HOME` on MacOS, then configuring installation prerequisites for each of the operating systems below. We'll also do a quick install and sanity check on Splice Machine.

Follow the Written Instructions

This section includes written instructions for configuring your computer for Splice Machine and then installing our standalone version, in these steps:

- Prepare for Installation on Your Computer
- Install Splice Machine

- Start Using Splice Machine

Note that we have also created a short tutorial that shows you how to import the demo data included in our installation. We recommend that you follow the steps in this tutorial to import and then run a few test queries against the demo data to start experiencing the full power of Splice Machine.

Prepare for Installation on Your Computer

This section walks you through preparing your computer for installation of Splice Machine; follow the instructions for your operating system:

- Mac OSX
- Ubuntu Linux
- CentOS/Red Hat Enterprise Linux (RHEL)

Configure Mac OSX for Splice Machine

Follow these steps to prepare your MacOS computer to work with Splice Machine:

1. **If you have Homebrew installed**
   Fire up the `Terminal` app and enter the following commands:

   ```
   $ brew update
   $ brew cask install java
   $ brew install rlwrap
   ```

2. **If you do not have Homebrew installed:**

   a. Download the Java SE Development Kit (JDK 8) from this URL:

      ```
      http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloa
      ds-2133151.html
      ```

   b. Open the downloaded installer and compete the installation wizard to install the JDK.

   c. Download and install `rlWrap`. You can find rlWrap online, along with instructions for installing it using `brew` or another installation method. Some experts advise using Homebrew instead of any other method.

3. **Check that your $JAVA_HOME environment variable is set:**

   a. Edit your `~/.bash_profile` file with your favorite text editor; for example:

      ```
      $ sudo vi ~/.bash_profile
      ```

   b. Add the following `export` command at the bottom of the `.bash_profile` file.

```
export JAVA_HOME=`/usr/libexec/java_home`
```

c. Save the file and close your editor.

d. Run the following command to load the updated profile into your current session:

```
$ source ~/.bash_profile
```

e. To verify that this setting is correct, make sure that the following commands display the same value:

```
$ echo $JAVA_HOME/usr/libexec/java_home
```

Configure Ubuntu Linux for Splice Machine

Follow these steps to prepare your Ubuntu computer to work with Splice Machine:

NOTE: Installing on Linux can be a bit tricky, so please use the `pwd` command to ensure that you are in the directory in which you need to be.

1. **Install the Java SE Development Kit**
   Fire up the `Terminal` app and enter the following commands:

```
$ sudo add-apt-repository ppa:webupd8team/java$ sudo apt-get update$ sud
o apt-get install oracle-java8-installer
$ sudo apt install oracle-java8-set-default
```

2. **Set the $JAVA_HOME environment variable is set:**

   a. Find the location of Java on your computer by executing this command:

```
$ sudo update-alternatives --config java
```

   Copy the resulting path to the clipboard.

   b. Use your favorite text editor to open `/etc/environment`:

```
sudo vi /etc/environment
```

   c. Add the following `export` command at the bottom of the `/etc/environment` file.

```
JAVA_HOME="/usr/lib/jvm/java-8-oracle"
```

d. Run the following command to load the updated profile into your current session:

```
source ~/etc/environment
```

e. Run this command to make sure the setting is correct:

```
$ echo $JAVA_HOME
```

3. **Install Additional Libraries:**
   You need to have the following packages installed for Splice Machine to run:

   - curl
   - nscd
   - ntp
   - openssh
   - openssh-clients
   - openssh-server
   - patch
   - rlwrap
   - wget

   You can use the `apt-get` package manager to install each of these packages, using a command line like this:

```
$ sudo apt-get install <packagename>
```

4. **Configure Additional Parameters:**
   You need to complete these steps:

Create symbolic link for YARN	`sudo ln -s /usr/bin/java /bin/java`
Configure swappiness	`$ echo 'vm.swappiness = 0' >> /etc/sysctl.conf`
Create an alias	`$ rm /bin/sh ; ln -sf /bin/bash /bin/sh`

5. **Ensure all necessary services are started:**
   You can start all necessary services as follows:

   ```
   $ sudo service nscd start  && service ntp start  && service ssh start
   ```

   Finally, you can check that the services are running with the following command:

   ```
   $ service <service_name> status
   ```

Configure CentOS/Red Hat Enterprise Linux (RHEL) for Splice Machine

Follow these steps to prepare your RHEL computer to work with Splice Machine:

NOTE: Installing on Linux can be a bit tricky, so please use the `pwd` command to ensure that you are in the directory in which you need to be.

1. **Install the Java SE Development Kit**
   Fire up the `Terminal` app and enter the following commands:

   To install JDK 8, run one of the following sets of commands:

   ```
   $ cd /opt/$ wget --no-cookies --no-check-certificate --header "Cookie: gp
   w_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-c
   ookie" "http://download.oracle.com/otn-pub/java/jdk/8u121-b13/e9e7ea248e2
   c4826b92b3f075a80e441/jdk-8u121-linux-x64.tar.gz"$ tar xzf jdk-8u121-linu
   x-i586.tar.gz
   ```

   or:

   ```
   $ wget http://download.oracle.com/otn-pub/java/jdk/8u121-b13/e9e7ea248e2c
   4826b92b3f075a80e441/jdk-8u121-linux-x64.rpm$ sudo yum localinstall jdk-8
   u121-linux-x64.rpm
   ```

   or:

   ```
   $ sudo rpm -ivh jdk-8u121-linux-x64.rpm
   ```

2. **Set up sudo rights**
   Your User ID must be in the `sudoers` file. To do so, please see the following web page:

```
https://www.digitalocean.com/community/tutorials/how-to-edit-the-sudoer
s-file-on-ubuntu-and-centos
```

3. **Set the $JAVA_HOME environment variable is set:**

   a. Find the path to the java installation on your computer:

   ```
   $ find / -name java
   ```

   b. Configure your `$JAVA_HOME` environment variable to the path you found in the previous step, as follows:

   ```
   $ echo export JAVA_HOME=/opt/jdk1.8.0_121 >/etc/profile.d/javaenv.sh
   ```

   c. Make sure your `$JRE_HOME` variable is set up:

   ```
   $ echo export JRE_HOME=/opt/jdk1.8.0_121/jre >/etc/profile.d/javaen
   v.sh
   ```

   d. Set up your `$PATH` variable for your JVM:

   ```
   $ echo export PATH=$PATH:/opt/jdk1.8.0_121/bin:/opt/jdk1.8.0_121/jr
   e/bin >/etc/profile.d/javaenv.sh
   ```

   e. Confirm the variables are set with the echo command (you may need to reload your terminal first):

   ```
   $ echo $JAVA_HOME
   ```

4. **Install Additional Libraries:**
   You need to have the following packages installed for Splice Machine to run:
   curl, nscd, ntp, openssh, openssh-clients, openssh-server, patch, rlwrap, wget, ftp, nc, EPEL repository

   - curl
   - EPEL repository
   - ftp
   - nc
   - nscd
   - ntp
   - openssh

- openssh-clients
- openssh-server
- patch
- rlwrap
- wget

You can install these libraries as follows:

a. To update CentOS:

```
$ yum update
```

b. To install a binary:

```
$ yum install <packagename>
```

c. To install EPEL repository:

```
$ yum -y install epel-release
```

d. To test if a package is installed:

```
$ yum info <packagename>
```

5. **Configure Additional Parameters:**
Execute this command:

```
$ sed -i '/requiretty/ s/^/#/' /etc/sudoers
```

6. **Ensure all necessary services are started:**
You can start all necessary services as follows:

```
$ /sbin/service nscd start  && /sbin/service ntpd start  && /sbin/servic
e sshd start
```

Finally, you can check that the services are running with the following command:

```
$ chkconfig --list
```

Install Splice Machine

Now that you've got your system configured for Splice Machine, let's download and install the standalone version of Splice Machine, and start using it!

1. **Download the Splice Machine installer.**
   Visit this page: https://www.splicemachine.com/get-started/download/

2. **Copy the downloaded tarball (.gz file) to the directory on your computer in which you want to install Splice Machine**
   You should only install in a directory whose name does not contain spaces, because some scripts will not operate correctly if the working directory has spaces in its name.

3. **Install Splice Machine:**
   Unpack the tarball `gz` file that you downloaded: https://s3.amazonaws.com/splice-releases/2.5.0.1802/standalone/splicemachine-2.5.0.1729.tar.gz

   This creates a `splicemachine` subdirectory and installs Splice Machine software in it.

Start Using Splice Machine

Start Splice Machine on your computer and run a few commands to verify the installation:

1. **Make your install directory the current directory**

   ```
   cd splicemachine
   ```

2. **Run the Splice Machine start-up script**

   ```
   ./bin/start-splice.sh
   ```

   Initialization of the database may take a couple minutes. It is ready for use when you see this message:

   ```
   Splice Server is ready
   ```

3. **Start using the Splice Machine command line interpreter by launching the `sqlshell.sh` script:**

   ```
   ./bin/sqlshell.sh
   ```

   Once you have launched the command line interpreter (the `splice>` prompt), we recommend verifying that all is well by running a few sample commands. First:

   ```
   splice> show tables;
   ```

You'll see the names of the tables that are already defined in the Splice Machine database; namely, the system tables. Once that works, you know Splice Machine is alive and well on your computer, and you can use help to list the available commands:

```
splice> help;
```

When you're ready to exit Splice Machine:

```
splice> exit;
```

NOTE: Make sure you end each command with a semicolon (`;`), followed by the *Enter* key or *Return* key.

Finally

We do hope you found these instructions helpful, whatever your Java service happens to be. Now that you are ready to start working with Splice Machine, youll find great content here:

- Note that we have also created a short tutorial that shows you how to import the demo data included in our installation. We recommend that you follow the steps in this tutorial to import and then run a few test queries against the demo data to start experiencing the full power of Splice Machine.
- Documentation - Start Here: Documentation Home Page
- Tutorials: Tutorials Home Page

We welcome any of your questions or comments on our forum or slack channel within the Splice Machine community. If you have more questions or would like to see a demo of our system, reach out to us at info@splicemachine.com or by filling out this form.

Importing and Querying Demo Data

This is an On-Premise-Only topic! Learn about our products

This topic walks you through importing and querying the demo (sample) data that is packaged with the Splice Machine standalone installer.

You must install the standalone version of Splice Machine on your computer before following the steps in this tutorial.

The Splice Machine installer package includes demo data that you can import into your database, so you can get used to working with your new database. We recommend that you follow the steps in this topic to import this demo data, and then run a few test queries against it to verify your installation.

About the Demo Data

The demo data included in your installer package requires about 30 MB in compressed format. Importing the demo data creates three tables, each of which contains one million records:

Table	Description
`T_HEADER`	Standard *headers* from a transaction system
`T_DETAIL`	Standard *detail* records from a transaction system
`CUSTOMERS`	A list of target *customers*

Follow the Written Instructions

Import the Data

Follow these steps to import the demo data into your Splice Machine database:

1. **Start the command line interpreter**
   You can use the Splice Machine command line interpreter (CLI), or `splice>` prompt, to work directly with your database. If you're using the cluster version of Splice Machine, you can access the `splice>` prompt by entering this shell command on any node on which it is available:

```
./sqlshell.sh
```

If you're using the standalone version of Splice Machine, use these steps to access the `splice>` prompt:

```
cd <your.splicemachine-directory>
./bin/sqlshell.sh
```

2. **Modify the script that loads the data to use your path:**
   Before running the `loadall.sql` script, you must change the file path used in the script.

   There are calls to `SYSCS_UTIL.IMPORT_DATA` near the bottom of the script. Change the file path parameter in each of these calls to use the absolute path to your Splice Machine `demodata` directory:

```
call SYSCS_UTIL.IMPORT_DATA('SPLICE', 'T_HEADER',  null, '<yourPath>/demo
data/data/theader.csv', ...;call SYSCS_UTIL.IMPORT_DATA('SPLICE', 'T_DETA
IL',  null, '<yourPath>/demodata/data/tdetail.csv', ...;call SYSCS_UTIL.I
MPORT_DATA('SPLICE', 'CUSTOMERS', null, '<yourPath>/demodata/data/custome
rs.csv', ...;
```

   Make sure you use the absolute (versus relative) path. For example:

```
call SYSCS_UTIL.IMPORT_DATA('SPLICE', 'T_HEADER',  null, '/Users/myName/m
ySplice/demodata/data/theader.csv', ...;call SYSCS_UTIL.IMPORT_DATA('SPLI
CE', 'T_DETAIL',  null, '/Users/myName/mySplice/demodata/data/tdetail.cs
v', ...;call SYSCS_UTIL.IMPORT_DATA('SPLICE', 'CUSTOMERS', null, '/Users/
myName/mySplice/demodata/data/customers.csv',...;
```

3. **Run the modify script to loads the data:**
   From the `splice>` prompt, *run* the file that will load the data, using single quotes around the path/filename (and remember to include the semicolon at the end):

```
splice> run 'demodata/sql/loadall.sql';
```

4. **Wait for the script to finish**
   If your database is not currently running, start it up and launch the command line interpreter (`splice>` prompt) by issuing this command in your terminal window:

```
./bin/sqlshell.sh
```

   The loading process can take several minutes: the `loadall.sql` file creates the schema, loads the data, and creates indexes for the tables.

   NOTE: While the database is running, logging information is written to the `splice.log` file, which is found in the `splicemachine` directory.

   When you again see the `splice>` prompt, the demo data is ready to use. We recommend running the sample queries in the next section to get a feel for using Splice Machine and the demo data.

Run Sample Queries

After you have imported the demo data, you can use the `splice>` command line interpreter to run the sample queries on this page to get some experience with using Splice Machine.

You can simply copy the select command from each of the samples below to your clipboard. Then paste from the clipboard at the `splice>` prompt and press the *Enter* key or *Return* key to submit the query.

Example of Selecting a Subset

You can use the following query to select the customer IDs from a subset of the transaction detail (`T_DETAIL`) table, based on transaction date and category ID.

```
select customer_master_id
   from T_DETAIL d
   where TRANSACTION_DT >= DATE('2010-01-01')
      and TRANSACTION_DT <= DATE('2013-12-31')
      AND ORIGINAL_SKU_CATEGORY_ID >= 44427
      and original_sku_category_id <= 44431;
```

Example of Selecting With a Join

You can use the following to query a join of the `T_HEADER` and `CUSTOMERS` tables.

```
select t.transaction_header_key, t.transaction_dt, t.store_nbr,
       t.geocapture_flg, t.exchange_rate_percent    from T_HEADER t, CUSTOMERS c   w
here c.customer_master_id=t.customer_master_id   and t.customer_master_id > 14000
   and t.customer_master_id < 15000;
```

Troubleshooting Transaction Exceptions on MacOS

If you're running transactions in the standalone version of Splice Machine on MacOS, you may run into an exception caused by the clock having moved backwards. This happens only rarely, and is due to the fact that OS X has its own time-maintenance daemon that can (rarely) cause the clock to move backwards, which causes a transaction exception.

When this happens, you'll see an exception messages like the following:

```
SQLSTATE: XJ001Java exception: 'java.io.IOException: java.lang.IllegalStateExceptio
n: Unable to obtain timestamp, clock moved backwards
```

To correct the problem, simply re-run the query or statement that generated the exception.

Configuring Splice Machine Authentication

This is an On-Premise-Only topic! Learn about our products

This topic describes the mechanisms you can use in Splice Machine to authenticate users and how to configure the mechanism you choose to use, in these sections:

- Supported Authentication Mechanisms
- Configuring Authentication

Supported Authentication Mechanisms

You can use one of the following authentication mechanisms, each of which is described below the table:

Authentication Mechanism	Description
`None`	Any user ID and password combination is allowed to connect to database.
`Native`	User IDs in a database table are validated against the corresponding, encrypted password. This is the default authentication setting for Splice Machine installations.
`LDAP`	User IDs are validated against an existing LDAP service. **ENTERPRISE ONLY: This feature is available only for the Splice Machine Enterprise version of our On-Premise Database product.** You cannot use this feature with the Community editions of Splice Machine. For a list of the additional features available in the Enterprise edition, see our Splice Machine Editions page. To obtain a license for the Splice Machine *Enterprise Edition*, **please Contact Splice Machine Sales today.**

Configuring Authentication

You configure Splice Machine authentication by adding or updating properties in your HBase configuration file; this is typically done during installation of Splice Machine, but you can modify your settings whenever you want. This section contains the following subsections:

- Locating Your Configuration File
- Disabling Authentication
- Using Native Authentication
- Using LDAP Authentication

Locating Your Configuration File

The following table specifies the platform-specific location of the configuration you need to update when changing your Splice Machine authentication properties:

Platform	Configuration file to modify with your authentication properties
CDH	hbase-site.xml
HDP	Select the Custom HBase Configs option from the HBase configuration tab.
MapR	hbase-site.xml
Standalone version	splicemachine/lib/splice-site.xml

Configure your authentication settings by adding or modifying properties in the configuration file.

Disabling Authentication

If you want to disable authentication for your Splice Machine database, you can set the authentication property to `NONE`.

NOTE: Splice Machine **strongly encourages you to not use** an open database for production databases!

You can configure an open database that allows any user to authenticate against the database by setting your authentication properties as follows:

```
<property>
   <name>splice.authentication</name>
   <value>NONE</value>
</property>
```

Using Native Authentication

Native authentication is the default mechanism for Splice Machine; you don't need to modify your configuration if you wish to use it. Native authentication uses the `sys.sysusers` table in the `splice` schema for configuring user names and passwords.

The default native authentication property settings are:

```
<property>
    <name>splice.authentication</name>
    <value>NATIVE</value>
</property>
<property>
    <name>splice.authentication.native.algorithm</name>
    <value>SHA-512</value>
</property>
```

You can use `MD5`, `SHA-256`, or `SHA-512` for the value of the `native.algorithm` property; `SHA-512` is the default value.

Switching to Native Authentication

If you are switching your authentication from to `Native` authentication from another mechanism (including `NONE`), there's one additional step you need to take: you must re-initialize the credentials database (`SYSUSERS` table), by adding the following property setting to your configuration file:

```
<property>
    <name>splice.authentication.native.create.credentials.database</name>
    <value>true</value>
</property>
```

Using LDAP Authentication

LDAP authentication in Splice Machine uses an external LDAP server.

LDAP authentication is available only with a Splice Machine Enterprise license; you cannot use LDAP authentication with the Community version of Splice Machine.

To obtain a license for the Splice Machine Enterprise Edition, **please Contact Splice Machine Sales today.**

To use LDAP with Splice Machine, you must:

- Contact us to obtain a license key from Splice Machine.
- Enable Enterprise features by adding your Splice Machine license key to your HBase configuration file as the value of the `splicemachine.enterprise.key` property, as shown below.
- Make sure that a user with name `splice` has been created in the LDAP server.

» Add the Splice Machine LDAP properties in your HBase configuration file, along with the license key property. Note that you may need to set `splice.authentication` properties in both service and client HBase configuration files:

LDAP Property Settings

These are the property settings you need to configure:

```
<property>
   <name>splicemachine.enterprise.key</name>
   <value><your-Splice-Machine-license-key></value>
</property>
<property>
   <name>splice.authentication</name>
   <value>LDAP</value>
</property>
<property>
   <name>splice.authentication.ldap.server</name>
   <value><ldap://servername-ldap.yourcompany.com:389></value>
</property>
<property>
   <name>splice.authentication.ldap.searchAuthDN</name>
   <value><cn=commonName,ou=Users,dc=yourcompany,dc=com></value>
</property>
<property>
   <name>splice.authentication.ldap.searchAuthPW</name>
   <value><yourpassword</span></value>
</property>
<property>
   <name>splice.authentication.ldap.searchBase</name>
   <value>ou=Users,dc=yourcompany,dc=com</value>
</property>
<property>
   <name>splice.authentication.ldap.searchFilter</name>
   <value>&lt;(&amp;(objectClass=*)(uid=%USERNAME%))&gt;</value>
</property>
```

Notes about the LDAP property values:

- Specify the location of your external LDAP server host in the splice.authentication.ldap.server property on port `389`.
- The `ldap.searchAuthDN` property is the security principal:
  - This is used to create the initial LDAP context (aka its connection to a specific DN).
  - It must have the authority to search the user space for user DNs.
  - The `cn=` is the *common name* of the security principal.
- The `ldap.searchAuthPW` property specifies password Splice Machine should use to perform the DN search.
- The `ldap.searchBase` property specifies the root DN of the point in your hierarchy from which to begin a guest or anonymous search for the user's DN.

- The `ldap.searchFilter` property specifies the search filter to use to determine what constitutes a user while searching for a user DN

Connecting with JDBC and LDAP

You can then use our JDBC driver to connect to your database with LDAP authentication, using a connection string similar to this:

```
jdbc:splice://localhost:1527/splicedb;user=yourName;password=yourPswd
```

Configuring SSL/TLS Authentication

This is an On-Premise-Only topic! Learn about our products

This topic describes how to configure SSL/TLS on your cluster to support secure JDBC connections to your Splice Machine database.

By default, JDBC connections to your database are not encrypted. You can configure SSL/TLS authentication for two different encryption modes:

- *Basic SSL/TLS Encryption* encrypts the data sent back and forth between a client and the server.
- *SSL/TLS Encryption with Peer Authentication* encrypts the data sent between client and server, and adds a layer of authentication known as peer authentication, which uses trusted certificates to authenticate the sender and/or receiver.

  The term *peer* is used in this context to refer to the other side of a server-client communication: the client is the server's peer, and the server is the client's peer.You can set up peer authentication on a server, a client, or both.

The remainder of this topic shows you how to configure this authentication, in these sections:

- Generating Certificates walks you through generating the certificates.
- Importing Certificates describes how to import certificates to clients and servers.
- Updating Configuration Options shows you how to update your server's configuration options and then restart your server.
- Restarting your Server describes how to restart the server so that your updated security options take effect.
- Connecting Securely From a Client walks you through connecting securely from various clients.

NOTE: We use `highlighted text` in this section to display sample names that you should replace with your own names.

Generating Certificates

This section shows you how to generate the required, trusted certificates on your server and client(s).

To configure your database for SSL/TLS authentication, you'll need to use the *keytool* application that is included with the JDK; this tool is documented here: http://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html.

Generate a Server Certificate

Use the *keytool* to generate a key-pair in the keystore. Here's an example of interacting with the keytool:

```
% keytool -genkey -alias MyServerName -keystore ~/vault/ServerKeyStore
Enter keystore password: myPassword
Re-enter new password: myPassword
What is your first and last name?
   [Unknown]: John Doe
What is the name of your organizational unit?
   [Unknown]: TechPubs
What is the name of your organization?
   [Unknown]: MyCompany
What is the name of your City or Locality?
   [Unknown]: San Francisco
What is the name of your State or Province?
   [Unknown]: CA
What is the two-letter country code for this unit?
   [Unknown]: US
IS CN=John Doe, OU=TechPubs, O=MyCompany, L=San Francisco, ST=CA, C=US correct?
   [no]: yes

Enter key password for <MyServerName>
            (RETURN if same as keystore password): myPassword
```

Now issue this *keytool* command to generate the certificate from the key you just created:

```
% keytool -export -alias MyServerName \
> -keystore ~/value/ServerKeyStore -rfc -file ServerCertificate \
> -storepass myPassword
Certificate stored in file <ServerCertificate>
% ls -ltr
total 8
-rw-rw-r-- 1 myName myGroup 1295 Aug 3 23:15 ServerKeyStore
-rw-rw-r-- 1 myName myGroup 1181 Aug 3 23:23 ServerCertificate
%
```

Generate Client Certificates

Now use the *keytool* to generate a client key-pair in the keystore; for example:

```
% keytool -genkey -alias MyClientName -keystore ~/vault/ClientKeyStore
```

Respond to the questions in the same way as you did when generting the server key-pair.

And then generate the client certificate:

```
% keytool -export -alias MyClientName \
> -keystore ~/value/ClientKeyStore -rfc -file ClientCertificate \
> -storepass myPassword
Certificate stored in file <ClientCertificate>
% ls -ltr
total 8
-rw-rw-r-- 1 myName myGroup 1295 Aug 3 23:15 ClientKeyStore
-rw-rw-r-- 1 myName myGroup 1181 Aug 3 23:23 ClientCertificate
%
```

Importing Certificates

After you've generated your certificates, it's time to:

- Import the server certificate to the client keystore.
- Import the client certificate to the server keystore.
- Secure and deploy the keystore and trust store.

Import Server Certificate to Client Keystore

You need to copy (`scp`) the server certificate to the client, and then use a *keytool* command like this to import the certificate:

```
% keytool -import -alias favoriteServerCertificate \
 -file ServerCertificate -keystore ~/vault/ClientTrustStore \
 -storepass secretClientTrustStorePassword
Owner: CN=John Doe, OU=TechPubs, O=MyCompany, L=San Francisco, ST=CA, C=US
Issuer: CN=John Doe, OU=TechPubs, O=MyCompany, L=San Francisco, ST=CA, C=US
Serial number: 24a2c7f7
Valid from: Thu Aug 03 23:15:52 UTC 2017 until: Wed Nov 01 23:15:52 UTC 2017
Certificate fingerprints:
         MD5:  5B:86:E9:C9:DF:E1:34:41:C8:B0:55:DE:C6:34:8A:21
         SHA1: 17:C8:43:FE:9C:FF:0C:4A:B2:51:36:0C:79:16:EB:73:24:C3:5A:83
         SHA256: 8D:55:1A:10:37:39:21:14:8E:21:3A:10:78:A1:C7:25:5F:9C:A7:8D:4E:3F:87:4
0:A0:ED:70:BE:EC:0F:7A:D9
         Signature algorithm name: SHA1withDSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 8B 71 1E 04 E7 E4 84 E6   35 B3 6B EB B5 92 1A 35  .q......5.k....5
0010: 5E FD B1 40                                        ^..@
]
]
```

Import Client Certificates to Server Keystore

Next, copy (`scp`) the client certificate to the region server and use *keytool* to import the certificate:

```
% keytool -import -alias Client_1_Certificate \
 -file ClientCertificate -keystore ~/vault/ServerTrustStore \
 -storepass secretServerTrustStorePassword
Owner: CN=John Doe, OU=TechPubs, O=MyCompany, L=San Francisco, ST=CA, C=US
Issuer: CN=John Doe, OU=TechPubs, O=MyCompany, L=San Francisco, ST=CA, C=US
Serial number: 7507d351
Valid from: Thu Aug 03 23:25:19 UTC 2017 until: Wed Nov 01 23:25:19 UTC 2017
Certificate fingerprints:
         MD5:  19:46:4C:D5:6C:A5:40:AC:6C:F6:2C:DC:7B:86:C5:45
         SHA1: BB:D7:9C:5E:5A:EB:E3:D1:F0:22:49:47:D4:C5:31:24:1E:06:0F:AE
         SHA256: 88:8E:6E:97:ED:78:B1:AE:5E:65:09:30:C2:E8:AF:B3:DD:40:5A:7B:19:97:ED:0
4:E0:A3:82:66:E9:A4:3E:2A
         Signature algorithm name: SHA1withDSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 59 37 E4 92 34 0A A2 45   93 E6 45 3A AF 57 77 E8  Y7..4..E..E:.Ww.
0010: E6 B9 24 08                                        ..$.
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
%
```

Secure and Deploy the Keystore and Trust Store

Once you've imported your certificates, you should copy the `vault` directory to a location that is directly accessible from HBase. We recommend copying it to a directory such as `/etc/vault` or `/hbase/vault`.

You **MUST** deploy the vault directory to the same location on each region server, so that all region servers can access it during server startup.

Updating Configuration Options

Now that you've got your certificates all set up, you need to modify a few configuration options and restart the server to take your new security options live. On a CDH cluster, you need to update the region server Java options, which you'll find in the Admin console:

```
CDH->HBase->Configuration->Java Configuration Options for HBase Region Server
```

- If you’re using basic SSL/TLS (without peer authentication), add this property:

```
-Dderby.drda.sslMode=basic
```

- If you’re using full SSL/TLS (with peer authentication), add these properties:

```
-Dderby.drda.sslMode=peerAuthentication
-Djavax.net.ssl.keyStore=/tmp/vault/ServerKeyStore
-Djavax.net.ssl.keyStorePassword=myPassword
-Djavax.net.ssl.trustStore=/tmp/vault/ServerTrustStore
-Djavax.net.ssl.trustStorePassword=secretServerTrustStorePassword
```

Rebooting Your Cluster

Once you’ve updated your configuration, restart HBase to make the changes effective in your cluster. After HBase restart, you can verify that the server started in secure mode by examining the logs: If you look at /var/log/hbase/splice.log, you should see a message similar to this:

```
Mon AUG 28 04:52:03 UTC 2017 : Splice Machine Network Server - 10.9.2.2 - (1) started an
d ready to accept SSL connections on port 1527
```

Connecting Securely From a Client

You can now connect securely to your database. This section provides several examples:

- Running the splice> Command Line Securely
- Running a JDBC Client App Securely
- Adding New Client Nodes
- Connecting Securely From a Third Party Client

Running the splice> Command Line Securely

To run the splice> command line securely, you need to export several `env` variables before starting `sqlshell`. You can then issue a `connect` command that specifies the type of security, as shown below.

First export the environmental variables that specify your key stores:

```
export CLIENT_SSL_KEYSTORE=/home/splice/vault/ClientKeyStore
export CLIENT_SSL_KEYSTOREPASSWD=myPassword
export CLIENT_SSL_TRUSTSTORE=/home/splice/vault/ClientTrustStore
export CLIENT_SSL_TRUSTSTOREPASSWD=secretClientTrustStorePassword
```

Then, start the `splice>` command line:

```
% ./sqlshell.sh
```

The `sqlshell` command issues a default (no security) connection command. To connect securely, add an `ssl=` option to the connect command. You use different `connect` commands for each of the three security modes:

Security Mode	Connect Command
None	`connect 'jdbc:splice://x.x.x.xxx:1527/splicedb;user=splice;password=admin';`
Basic SSL	`connect 'jdbc:splice://x.x.x.xxx:1527/splicedb;user=splice;password=admin;ssl=basic';`
SSL w/Peer Authentication	`connect 'jdbc:splice://x.x.x.xxx:1527/splicedb;user=splice;password=admin;ssl=peerAuthentication';`

Running a JDBC Client App Securely

To use a secured connection with a JDBC client app, you need to specify a connection string that includes the `ssl` option. If you don't specify this option, the default JDBC connection is unsecured, as shown in the *Connect Command* table in the previous section.

Here's a sample declaration for a peer authenticated connection to a Splice Machine database:

```
String dbUrl = "jdbc:splice://1.2.3.456:1527/splicedb;user=splice;password=admin;ssl=pee
rAuthentication";
```

We can create a Java program that includes that declaration and then compile it into `SampleJDBC.java` with a command like this:

```
+ javac -classpath ".:./db-client-2.6.0.1729-SNAPSHOT.jar" ./SampleJDBC.java
```

We can then use a command like this to execute and JDBC app with the correct SSL keystore and truststore properties:

```
% java -classpath .:./db-client-2.6.0.1729-SNAPSHOT.jar
-Djavax.net.ssl.keyStore=/home/splice/vault/ClientKeyStore
-Djavax.net.ssl.keyStorePassword=myPassword
-Djavax.net.ssl.trustStore=/home/splice/vault/ClientTrustStore
-Djavax.netDjavax.net.ssl.trustStore.ssl.trustStorePassword=secretClientTrustStorePasswo
rd SampleJDBC
```

Adding New Client Nodes

Whenever you connect a new client node to a server, you need to perform a few steps to enable SSL/TLS on the new node:

- Generate a new client certificate.
- Import the new client certificate into the server's keystore.
- Import the server certificate into the new client's keystore.
- Restart the server.

Finally, you need to set these env variables:

```
export CLIENT_SSL_KEYSTORE=/home/splice/vault/ClientKeyStore
export CLIENT_SSL_KEYSTOREPASSWD=myPassword
export CLIENT_SSL_TRUSTSTORE=/home/splice/vault/ClientTrustStore
export CLIENT_SSL_TRUSTSTOREPASSWD=secretClientTrustStorePassword
```

Connecting Securely From a Third Party Client

This section describes what you need to do to connect securely to your Splice Machine from a third party client. We use Zeppelin as an example; other clients will have similar requirements. For Zeppelin, follow these steps

1. Navigate to and edit the `bin/interpreter.sh` file in the `Zeppelin` installation directory.
2. Find the `JAVA_INTP_OPTS` property definition.
3. Append the following SSL properties onto that definition:

   ```
   JAVA_INTP_OPTS+="
   -Dzeppelin.log.file=${ZEPPELIN_LOGFILE} \
   -Djavax.net.ssl.keyStore=${CLIENT_SSL_KEYSTORE} \
   -Djavax.net.ssl.keyStorePassword=${CLIENT_SSL_KEYSTOREPASSWD} \
   -Djavax.net.ssl.trustStore=${CLIENT_SSL_TRUSTSTORE} \
   -Djavax.netDjavax.net.ssl.trustStore.ssl.trustStorePassword=${CLIENT_SSL_TRUSTSTORE
   PASSWD} "
   ```

4. Make sure that you have exported the SSL keystore and truststore env variables:

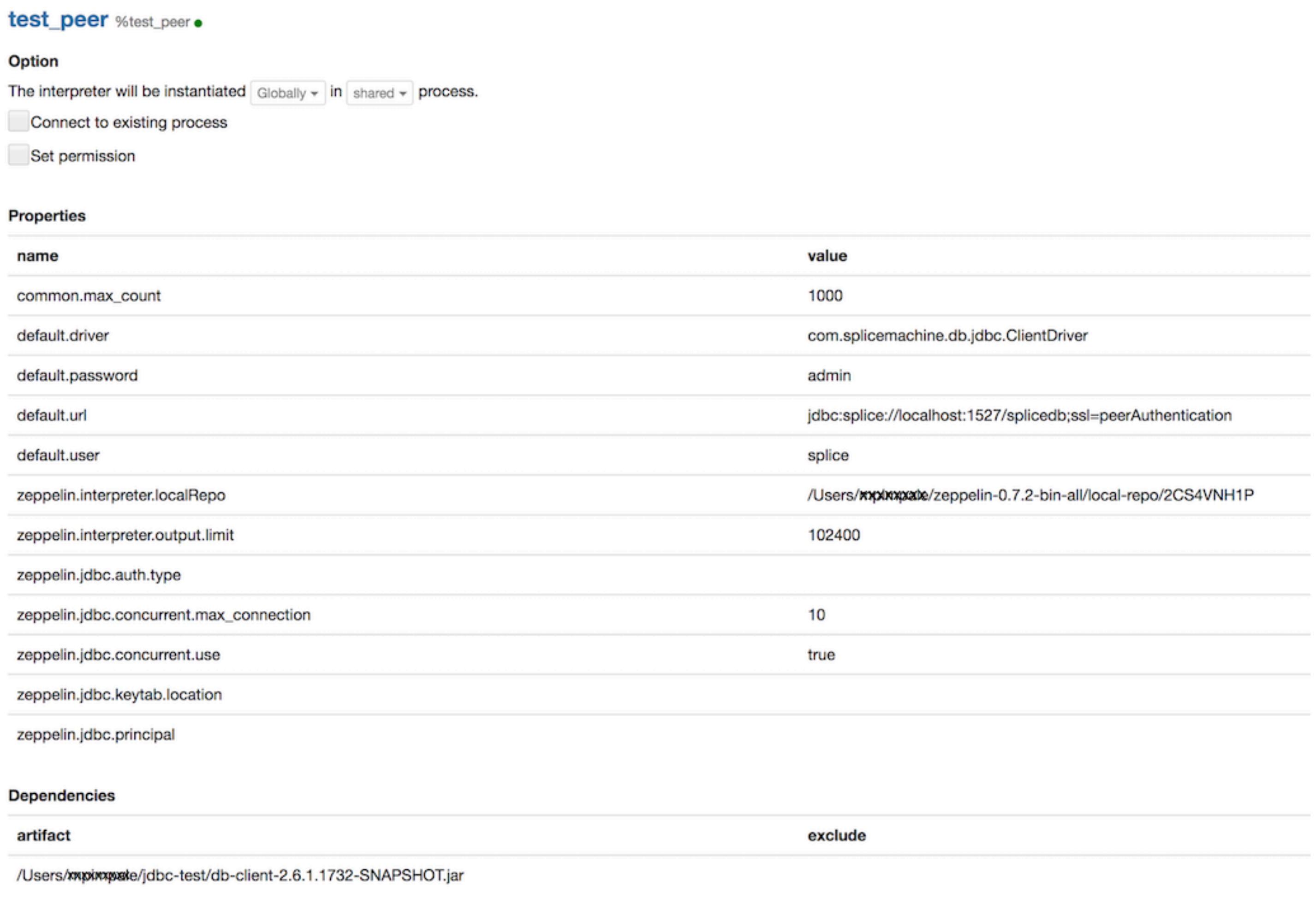
   ```
   export CLIENT_SSL_KEYSTORE=/home/splice/vault/ClientKeyStore
   export CLIENT_SSL_KEYSTOREPASSWD=myPassword
   export CLIENT_SSL_TRUSTSTORE=/home/splice/vault/ClientTrustStore
   export CLIENT_SSL_TRUSTSTOREPASSWD=secretClientTrustStorePassword
   ```

5. Restart Zeppelin:

   ```
   % zeppelin-daemon.sh start
   ```

6. Create a new JDBC Interpreter

   Navigate to the Zeppelin interface URL:, then click **Interpreter->+Create** to create a new interpreter. The image below shows sample settings for the new interpreter:



test_peer %test_peer ●

Option

The interpreter will be instantiated Globally ▾ in shared ▾ process.

☐ Connect to existing process

☐ Set permission

Properties

name	value
common.max_count	1000
default.driver	com.splicemachine.db.jdbc.ClientDriver
default.password	admin
default.url	jdbc:splice://localhost:1527/splicedb;ssl=peerAuthentication
default.user	splice
zeppelin.interpreter.localRepo	/Users/[redacted]/zeppelin-0.7.2-bin-all/local-repo/2CS4VNH1P
zeppelin.interpreter.output.limit	102400
zeppelin.jdbc.auth.type	
zeppelin.jdbc.concurrent.max_connection	10
zeppelin.jdbc.concurrent.use	true
zeppelin.jdbc.keytab.location	
zeppelin.jdbc.principal	

Dependencies

artifact	exclude
/Users/[redacted]/jdbc-test/db-client-2.6.1.1732-SNAPSHOT.jar	

Be sure to provide the correct JDBC driver loaction in the artifact dependencies section.

7. Save the new interpreter.
8. Create a new Note with the new interpreter.

Splice Machine Administrator’s Guide

This is an On-Premise-Only topic! Learn about our products

This *Administrator’s Guide* describes the administrative tasks associated with configuring and maintaining your Splice Machine software, in the following topics:

Topic	Description
Starting Your Database	How to start or restart your database.
Backing Up and Restoring	How to back up and restore your Splice Machine database.
Cleaning Your Database	How to clean your Splice Machine database.
Shutting Down Your Database	How to shut down your Splice Machine database.
Accessing Derby Properties	How to enable and disable Derby features by setting Derby properties.
Upgrading to the Enterprise Edition of Splice Machine	Describes how to upgrade from the *Community Edition* of Splice Machine to the *Enterprise Edition*.

For access to the source code for the Community Edition of Splice Machine, visit our open source GitHub repository.

For a summary of all Splice Machine documentation, see the Documentation Summary topic.

Starting Your Database

This is an On-Premise-Only topic! Learn about our products

- Starting Your Splice Machine Database on a Cloudera-Managed Cluster
- Starting Your Splice Machine Database on a Hortonworks HDP-Managed Cluster
- Starting Your Splice Machine Database on a MapR-Managed Cluster
- Starting Your Splice Machine Database on a Standalone installation

Starting Your Splice Machine Database on a Cloudera-Managed Cluster

Use the Cloudera Manager to start HBase:

1. **Navigate to the *Services->All Services* screen in *Cloudera Manager*, and select this action to start *HBase*:**

```
Actions -> Start
```

Starting Your Splice Machine Database on a Hortonworks HDP–Managed Cluster

Use the Ambari dashboard to start Splice Machine:

1. **Log in to the Ambari Dashboard by pointing your browser to the publicly visible `<hostName>` for your master node that is hosting Ambari Server:**

```
http://<hostName>:8080/
```

2. **Start cluster services:**

```
Action -> Start All
```

Starting Your Splice Machine Database on a MapR-Managed Cluster

To start Splice Machine, use the MapR Command System (MCS):

1. **Navigate to the node that is running the `mapr-webserver` service.**
2. **Log into `https://<hostIPAddr>:8443`, substituting the correct `<hostIPAddr>` value. The login credentials are the ones you used when you added the `mapr` user during installation.**
3. **Start your HBase master**
4. **Start all of your Region Servers**

Starting Your Database in the Standalone Version

Follow these steps to start your database if you're using the Standalone version of Splice Machine:

1. **Change directory to your install directory:**
   ```
   cd splicemachine
   ```
2. **Run the Splice Machine start-up script:**
   ```
   $ ./bin/start-splice.sh
   ```

Restarting Splice Machine

If you're running the standalone version of Splice Machine on a personal computer that goes into sleep mode, you need to stop and then restart Splice Machine when you wake the computer back up. For example, if you're running Splice Machine on a laptop, and close your laptop, then you'll need to stop and restart Splice Machine when you reopen your laptop.

To stop and restart Splice Machine, follow these steps:

1. **Make sure that you have quit the `splice>` command line interpreter:**
   ```
   splice> quit;
   ```
2. **Change directory to your install directory:**
   ```
   cd splicemachine
   ```
3. **Run the Splice Machine shut-down script:**

```
$ ./bin/stop-splice.sh
```

4. **Run the Splice Machine start-up**

   ```
   $ ./bin/start-splice.sh
   ```

Backing Up and Restoring Your Database

ENTERPRISE ONLY: This feature is available only for the Splice Machine Enterprise version of our On-Premise Database product.

You cannot use this feature with the Community editions of Splice Machine. For a list of the additional features available in the Enterprise edition, see our Splice Machine Editions page.

To obtain a license for the Splice Machine *Enterprise Edition*, **please Contact Splice Machine Sales today.**

Splice Machine provides built-in system procedures that make it easy to back up and restore your entire database. You can:

- create full and incremental backups to run immediately
- schedule daily full or incremental backups
- restore your database from a backup
- manage your backups
- access logs of your backups

The rest of this topic will help you with working with your backups, in these sections:

- About Splice Machine Backups
- Using the Backup Operations
- Backing Up to Cloud Storage

Backup Resource Allocation

Splice Machine backup jobs use a Map Reduce job to copy HFiles; this process may hang up if the resources required for the Map Reduce job are not available from Yarn. See the Backup Resource Allocation section of our *Troubleshooting Guide* for specific information about allocation of resources.

About Splice Machine Backups

Splice Machine supports: both full and incremental backups:

- A *full backup* backs up all of the files/blocks that constitute your database.
- An *incremental backup* only stores database files/blocks that have changed since a previous backup.

Because backups can consume a lot of disk space, most customers define a backup strategy that blends their needs for security, recover-ability, and space restrictions. Since incremental backups require a lot less space than do full backups, and allow for faster recovery of data, many customers schedule daily, incremental backups.

NOTE: Splice Machine automatically detects when it is the first run of an incremental backup and performs a one-time full backup; subsequent runs will only back up changed files/blocks.

Backup IDs, Backup Jobs, and Backup Tables

Splice Machine uses *backup IDs* to identify a specific full or incremental *backup* that is stored on a file system, and *backup job IDs* to identify each scheduled *backup job*. Information about backups and backup jobs is stored in these system tables:

System Table	Contains Information About
`SYS.SYSBACKUP`	Each database backup; you can query this table to find the ID of and details about a backup that was run at a specific time.
`SYS.SYSBACKUPITEMS`	Each item (table) in a backup.
`SYS.SYSBACKUPJOBS`	Each backup job that has been run for the database.

Temporary Tables and Backups

There's a subtle issue with performing a backup when you're using a temporary table in your session: although the temporary table is (correctly) not backed up, the temporary table's entry in the system tables will be backed up. When the backup is restored, the table entries will be restored, but the temporary table will be missing.

There's a simple workaround:

1. Exit your current session, which will automatically delete the temporary table and its system table entries.
2. Start a new session (reconnect to your database).
3. Start your backup job.

Using the Backup Operations

This section summarizes and provides examples of using the Splice Machine backup operations:

- Scheduling a Daily Backup Job
- Running an Immediate Backup
- Restoring Your Database From a Previous Backup
- Reviewing Backup Information

- Canceling a Scheduled Backup Job
- Canceling a Backup That's In Progress
- Deleting a Backup
- Deleting Outdated Backups

You must make sure that the directory to which you are backing up or from which data is being restored is accessible to the HBase user who is initiating the restore. Make sure the directory permissions are set correctly on the backup directory.

> **NOTE:** Note that you can store your backups in a cloud-based storage service such as AWS; for more information, see the Backing Up to Cloud Storage section below.

Scheduling a Daily Backup Job

Use the `SYSCS_UTIL.SYSCS_SCHEDULE_DAILY_BACKUP` system procedure to schedule a job that performs a daily incremental or full backup of your database:

```
SYSCS_UTIL.SYSCS_SCHEDULE_DAILY_BACKUP( backupDir, backupType, startHour );
```

backupDir

A `VARCHAR` value that specifies the path to the directory in which you want the backup stored.

Note that this directory can be cloud-based, as described in the Backing Up to Cloud Services section below.

backupType

A `VARCHAR(30)` value that specifies the type of backup that you want performed; one of the following values: `full` or `incremental`. The first run of an incremental backup is always a full backup.

startHour

Specifies the hour (`0-23`) in GMT at which you want the backup to run each day. A value less than `0` or greater than `23` produces an error and the backup is not scheduled.

Example 1: Set up a full backup to run every day at 3am:

To run a full backup every night at 3am:

```
call SYSCS_UTIL.SYSCS_SCHEDULE_DAILY_BACKUP('/home/backup', 'full', 3);
```

Example 2: Set up an incremental backup to run every day at noon:

To run an incremental backup every day at noon.

```
call SYSCS_UTIL.SYSCS_BACKUP_DATABASE('/home/backup', 'incremental', 12);
```

Running an Immediate Backup

Use the `SYSCS_UTIL.SYSCS_BACKUP_DATABASE` system procedure to immediately run a full or incremental backup.

```
SYSCS_UTIL.SYSCS_BACKUP_DATABASE( backupDir, backupType );
```

backupDir

A `VARCHAR` value that specifies the path to the directory in which you want the backup stored.

Note that this directory can be cloud-based, as described in the Backing Up to Cloud Services section below.

backupType

A `VARCHAR(30)` value that specifies the type of backup that you want performed; use.one of the following values: `full` or `incremental`.

Example 1: Execute a full backup now

To execute a backup right now:

```
call SYSCS_UTIL.SYSCS_BACKUP_DATABASE('/home/backup', 'full');
```

Example 2: Execute an incremental backup now:

This call will run an incremental backup right now. Splice Machine checks the `SYSBACKUP` system table to determine if there already is a backup for the system; if not, Splice Machine will perform a full backup, and subsequent backups will be incremental. The backup data is stored in the specified directory.

```
call SYSCS_UTIL.SYSCS_BACKUP_DATABASE('/home/backup', 'incremental');
```

Restoring Your Database From a Previous Backup

You can restore your database from a previous backup, use the `SYSCS_UTIL.SYSCS_RESTORE_DATABASE` system procedure.

To restore your database from a previous backup, use the `SYSCS_UTIL.SYSCS_RESTORE_DATABASE` system procedure:

```
SYSCS_UTIL.SYSCS_RESTORE_DATABASE(backupDir, backupId);
```

backupDir

A `VARCHAR` value that specifies the path to the directory in which the backup isstored.

You can find the *backupId* you want to use by querying the `SYSBACKUP` System Table. See the Reviewing Backup Information section below for more information.

backupId

A `BIGINT` value that specifies which backup you want to use to restore your database.

There are several important things to know about restoring your database from a previous backup:

- Restoring a database **wipes out your database** and replaces it with what had been previously backed up.
- You **cannot use your cluster** while restoring your database.
- You **must reboot your database** after the restore is complete by first Starting Your Database.

Example: Restore the database from a local, full backup

This example restores your database from the backup stored in the `/home/backup` directory that has `backupId=1273`:

```
call SYSCS_UTIL.SYSCS_RESTORE_DATABASE('/home/backup', 1273);
```

Reviewing Backups

Splice Machine stores information about your backups and scheduled backup jobs in system tables that you can query, and stores a backup log file in the directory to which a backup is written when it runs.

Backup Job Information

Information about your scheduled backup jobs is stored in the `SYSBACKUPJOBS` system table:

```
splice> select * from SYS.SYSBACKUPJOBS;
JOB_ID         |FILESYSTEM        |TYPE            |HOUR_OF_DAY|BEGIN_TIMESTAMP
-----------------------------------------------------------------------------------
--
22275          |/data/backup/0101 |FULL            |22          |2015-04-03 18:43:42.6
31
```

You can query this table to find a job ID, if you need to cancel a scheduled backup.

Backup Information

Information about each backup of your database is stored in the `SYSBACKUP` system table, including the ID assigned to the backup and its location. You can query this table to find the ID of a specific backup, which you need if you want to restore your database from it, or to delete it:

```
splice> select * from SYS.SYSBACKUP;
BACKUP_ID |BEGIN_TIMESTAMP          |END_TIMESTAMP            |STATUS |FILESYSTE
M         |SCOPE |INCR&|INCREMENTAL_PARENT_&|BACKUP_ITEM
------------------------------------------------------------------------------------
------------------------------------------------------
22275     |2015-04-03 18:40:56.877 |2015-04-03 18:43:42.631 |S      |/data/backup/01
01 |D     |false|-1       |15
21428     |2015-04-03 18:30:55.964 |2015-04-03 18:33:49.494 |S      |/data/backup/01
01 |D     |false|-1       |15
20793     |2015-04-03 18:23:53.574 |2015-04-03 18:27:07.07  |S      |/data/backup/01
01 |D     |false|-1       |87
```

Backup Log Files

When you run a backup, a log file is created or updated in the directory in which the backup is stored. This log file is named `backupStatus.log`, and is stored in plain text, human-readable format. Here’s a sample snippet from a log file:

```
Expected time for backup ~12 hours, expected finish at 15:30 on April 8, 2015
5 objects of 833 objects backed up..
6 objects of 833 objects backed up

Finished with Success. Total time taken for backup was 11 hours 32 minutes.
```

Canceling a Scheduled Backup

You can cancel a daily backup with the `SYSCS_UTIL.SYSCS_CANCEL_DAILY_BACKUP` system procedure:

```
SYSCS_UTIL.SYSCS_CANCEL_DAILY_BACKUP( jobId );
```

jobId

A `BIGINT` value that specifies which scheduled backup job you want to cancel.

You can find the *jobId* you want to cancel by querying the `SYSBACKUPJOBS` system table.

NOTE: Once you cancel a daily backup, it will no longer be scheduled to run.

Example: Cancel a daily backup

This example cancels the backup that has `jobId=1273`:

```
call SYSCS_UTIL.SYSCS_CANCEL_DAILY_BACKUP(1273);
```

Canceling a Backup That’s In Progress

You can call the `SYSCS_UTIL.SYSCS_CANCEL_BACKUP` system procedure to cancel a backup that is currently running:

```
SYSCS_UTIL.SYSCS_CANCEL_BACKUP(  );
```

Example: Cancel a running backup

This example cancels the Splice Machine backup job that is currently running.

```
call SYSCS_UTIL.SYSCS_CANCEL_BACKUP();
```

Deleting a Backup

Use the `SYSCS_UTIL.SYSCS_DELETE_BACKUP` system procedure to delete a single backup:

```
SYSCS_UTIL.SYSCS_DELETE_BACKUP( backupId );
```

backupId

A `BIGINT` value that specifies which backup you want to delete.

You can find the *backupId* you want to delete by querying the `SYSBACKUP` system table,

Example: Delete a backup

This example deletes the backup that has `backupId=1273`:

```
call SYSCS_UTIL.SYSCS_DELETE_BACKUP(1273);
```

Deleting Outdated Backups

Use the `SYSCS_UTIL.SYSCS_DELETE_OLD_BACKUPS` system procedure to delete all backups that are older than a certain number of days.

```
SYSCS_UTIL.SYSCS_DELETE_OLD_BACKUPS( backupWindow );
```

backupWindow

An `INT` value that specifies the number of days of backups that you want retained. Any backups created more than `backupWindow` days ago are deleted.

Example: Delete all backups more than a week old

This example deletes all backups that are more than a week old:

```
call SYSCS_UTIL.SYSCS_DELETE_OLD_BACKUPS(7);
```

Backing Up to Cloud Storage - AWS

You can specify cloud-based directories as destinations for your backups. This section describes how to set up credentials to allow Splice Machine to create and manage backups on AWS.

You need to enable backups by storing your AWS Access Key ID and Secret Access Key values in your cluster's HDFS core-site.xml file: how you set up your credentials depends on the Hadoop platform you are using; see the section below for your platform:

> **IMPORTANT:** You must have access to the S3 bucket to which you are backing up your database. The instructions below give general guidelines; however, S3 access differs in every deployment. For more information, see these sites:
>
> - http://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html
> - https://wiki.apache.org/hadoop/AmazonS3

- Enabling backups on CDH
- Enabling backups on HDP
- Enabling backups on MapR

Enabling Splice Machine Backups on CDH

You can use Cloudera Manager to configure properties to enable Splice Machine backups; follow these steps:

1. **Navigate to the Cloudera Manager home screen.**
2. **Stop both HBase and HDFS:**
   - Click the `HBase Actions` drop-down arrow associated with (to the right of) `HBase` in the cluster summary section of the home screen, and then click `Stop` to stop HBase.
   - Click the `HDFS Actions` drop-down arrow associated with (to the right of) and then click `Stop` to stop HDFS.
3. **Click `HDFS` in the Cloudera Manager home screen, then click the `Configuration` tab, and in category, click `Advanced`. Then set these property values in the `Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml` field:**

   ```
   fs.s3.awsAccessKeyId        = <Your AWS Access Key>
   fs.s3.awsSecretAccessKey    = <Your AWS Access Secret Key>
   ```

4. **Restart both services:**
   - Click the `HDFS Actions` drop-down arrow associated with (to the right of) HDFS in the cluster summary

section of the Cloudera Manager home screen, and then click `Start` to restart HDFS.

- Navigate to the *HBase Status* tab in Cloudera Manager. Then, using the `Actions` drop-down in the upper-right corner, click `Start` to create a start HBase.

Enabling Splice Machine Backups on HDP

You can use the Ambari dashboard to configure these properties. Follow these steps:

1. **Navigate to the HDFS `Configs` screen.**
2. **Select the `Services` tab at the top of the Ambari dashboard screen, then stop both HBase and HDFS:**
   - Click `HBase` in the left pane of the screen, then click `Service Actions->Stop` in the upper-right portion of the Ambari screen.
   - Click `HDFS` in the left pane of the screen, the click `Service Actions->Stop`.
3. **Select `Custom core-site` and add these properties:**

   ```
   fs.s3.awsAccessKeyId        = <Your AWS Access Key>
   fs.s3.awsSecretAccessKey    = <Your AWS Secret Access Key>
   ```

4. **Restart both services:**
   - Click `HDFS` in the left pane of the screen, the click `Service Actions->Restart All`.
   - Click `HBase` in the left pane of the screen, the click `Service Actions->Restart All`.

Enabling Splice Machine Backups on MapR

To enable Amazon S3 access on a MapR cluster, you must stop services, change the configuration files on each node, and then restart services. Follow these steps:

1. **Stop all MapR services by stopping the warden service on each host:**

   ```
   sudo service mapr-warden stop
   ```

2. **You need to edit two files on each MapR-FS fileserver and HBase RegionServer in your cluster to allow hosts access to Amazon S3. You need to provide the fs.s3 access key ID and secret in each of these files:**

- `/opt/mapr/hadoop/hadoop-2.x.x/etc/hadoop/core-site.xml` for *Hadoop/MapReduce/ YARN 2.x* site configuration
- `/opt/mapr/hadoop/hadoop-0.x.x/conf/core-site.xml` for *Hadoop/MapReduce 0.x/1.x* site configuration

If both *MapReduce v1* and *YARN/MapReduce 2* are installed on the MapR compute hosts, the newer *hadoop-2.x.x* version of the file will be canonical, and the older *hadoop-0.x.x* file symbolically linked to it. You can check this using the following `ls` and `file` commands:

```
$ ls -al /opt/mapr/hadoop/hadoop-0*/conf/core-site.xml /opt/mapr/hadoop/h
adoop-2*/etc/hadoop/core-site.xml
lrwxrwxrwx 1 mapr root  54 Apr 24 11:01 /opt/mapr/hadoop/hadoop-0.20.2/co
nf/core-site.xml -> /opt/mapr/hadoop/hadoop-2.5.1/etc/hadoop/core-site.xm
l
-rw-r--r-- 1 mapr root 775 Apr 24 12:50 /opt/mapr/hadoop/hadoop-2.5.1/et
c/hadoop/core-site.xml
```

```
$ file /opt/mapr/hadoop/hadoop-0*/conf/core-site.xml /opt/mapr/hadoop/had
oop-2*/etc/hadoop/core-site.xml
/opt/mapr/hadoop/hadoop-0.20.2/conf/core-site.xml:      symbolic link to
`/opt/mapr/hadoop/hadoop-2.5.1/etc/hadoop/core-site.xml'
/opt/mapr/hadoop/hadoop-2.5.1/etc/hadoop/core-site.xml: XML  document tex
t
```

3. **Add your access key ID and secret key in each file by adding the following properties between the `<configuration>` and `</configuration>` tags:**

```
<!-- AWS s3://bucket/... block-based access -->
<property>
<name>fs.s3.awsAccessKeyId</name>
<value>_AWS_ACCESS_KEY_ID_</value>
</property>
<property>
<name>fs.s3.awsSecretAccessKey</name>
<value>_AWS_SECRET_ACCESS_KEY_</value>
</property>
<!-- AWS s3n://bucket/... filesystem-like access -->
<property>
<name>fs.s3n.awsAccessKeyId</name>
<value>_AWS_ACCESS_KEY_ID_</value>
</property>
<property>
<name>fs.s3n.awsSecretAccessKey</name>
<value>_AWS_SECRET_ACCESS_KEY_</value>
</property>
```

4. **Use the `hadoop` command to view your configuration changes:**

   ```
   $ hadoop conf | grep fs\\.s3 | grep -i access | sort -u
   <property><name>fs.s3.awsAccessKeyId</name><value>_AWS_ACCESS_KEY_ID_</va
   lue><source>core-site.xml</source></property>
   <property><name>fs.s3.awsSecretAccessKey</name><value>_AWS_SECRET_ACCES
   S_KEY_</value><source>core-site.xml</source></property>
   <property><name>fs.s3n.awsAccessKeyId</name><value>_AWS_ACCESS_KEY_ID_</v
   alue><source>core-site.xml</source></property>
   <property><name>fs.s3n.awsSecretAccessKey</name><value>_AWS_SECRET_ACCES
   S_KEY_</value><source>core-site.xml</source></property>
   ```

5. **You can also verify that access is correctly configured with the `hadoop` command to list the contents of an existing bucket. For example:**

   ```
   sudo -iu mapr hadoop fs -ls s3n://yourbucketname/
   ```

6. **Finally, restart MapR services on each node via MapR's warden::**

   ```
   sudo service mapr-warden start
   ```

See Also

- `SYSCS_UTIL.SYSCS_BACKUP_DATABASE`
- `SYSCS_UTIL.SYSCS_CANCEL_BACKUP`
- `SYSCS_UTIL.SYSCS_CANCEL_DAILY_BACKUP`
- `SYSCS_UTIL.SYSCS_DELETE_BACKUP`
- `SYSCS_UTIL.SYSCS_DELETE_OLD_BACKUPS`
- `SYSCS_UTIL.SYSCS_RESTORE_DATABASE`
- `SYSCS_UTIL.SYSCS_SCHEDULE_DAILY_BACKUP`
- `SYSBACKUP` system table
- `SYSBACKUPITEMS` system table
- `SYSBACKUPJOBS` system table

Cleaning Your Database

This is an On-Premise-Only topic! Learn about our products

Cleaning your database essentially wipes out any user-defined tables, indexes, and related items.

This is a destructive process and should only be used by an administrator. Following the steps in this topic destroys your database data. And if you have non-SpliceMachine data stored in HBase, you must exercise additional caution to not destroy that data. Please follow the steps for your platform carefully.

You need to follow different steps, depending on which version of Splice Machine you are using:

- Cleaning Your Splice Machine Database on a Cloudera-Managed Cluster
- Cleaning Your Splice Machine Database on a Hortonworks HDP-Managed Cluster
- Cleaning Your Splice Machine Database on a MapR-Managed Cluster
- Cleaning Your Splice Machine Database on a Standalone installation

Cleaning Your Splice Machine Database on a Cloudera-Managed Cluster

Follow these steps to clean your database if you're using the Cloudera-managed cluster version of Splice Machine:

This is a destructive process and should only be used by an administrator!

1. **Shut down HBase and HDFS**
   Navigate to the `Services->All Services` screen in Cloudera Manager, and select these actions to stop *HBase* and *HDFS*:

   ```
   hbase -> Actions -> Stop
   hdfs1 -> Actions -> Stop
   zookeeper1 -> Actions -> Stop
   ```

2. **Use the Zookeeper client to clean things:**
   Restart ZooKeeper in the `Services->All Services` screen:

   ```
   zookeeper1 -> Actions -> Start
   ```

   Log in to the machine running Zookeeper on your cluster and start up a command-line (terminal) window.

Run the `zookeeper-client` command. At the prompt, run the following commands:

```
rmr /splice
rmr /hbase
quit
```

3. **Start HDFS**
   Navigate to the `Services->All Services` screen in Cloudera Manager, and restart *HDFS*:

   ```
   hdfs1 -> Actions -> Start
   ```

4. **Clean up HBase**
   Use the following shell command to delete the existing `/hbase` directory. You can run this command on any Data Node:

   ```
   sudo -su hdfs hadoop fs -rm -r /hbase
   ```

   If you are logged in as root, use this command instead:

   ```
   sudo -u hdfs hadoop fs -rm -r /hbase
   ```

   NOTE: If the machine running Cloudera Manager is not part of the cluster, **do not** run the command on that machine

5. **Create a new HBase directory:**
   Navigate to the `HBase` screen in Cloudera Manager, and create a new `/hbase` directory by selecting:

   ```
   Actions -> Create Root Directory
   ```

6. **Restart HBase**
   Now restart HBase from the same `Home->Services->hbase1` screen in Cloudera Manager, using this action:

   ```
   Actions -> Start
   ```

Cleaning Your Splice Machine Database on a Hortonworks HDP-Managed Cluster

Follow these steps to clean (or *flatten*) your database if you're using Splice Machine on an Ambari-managed Hortonworks Cluster:

This is a destructive process and should only be used by an administrator!

1. **Shut down HBase and HDFS**
   Log in to the Ambari Dashboard by pointing your browser to the publicly visible `<hostName>` for your master node that is hosting Ambari Server:

   ```
   http://<hostName>:8080/
   ```

   Select these actions to stop HBase and HDFS: :

   ```
   Services->HBase->Service Actions->Stop
   Services->HDFS->Service Actions->Stop
   ```

2. **Use the Zookeeper client to clean things**
   Log in to the node running Zookeeper on your cluster and start up a command-line (terminal) window.

   Run the `zookeeper-client` command. At the prompt, run the following commands:

   ```
   rmr /splice
   rmr /hbase
   quit
   ```

3. **Restart HDFS**
   Use the Ambari Dashboard to restart HDFS:

   ```
   Services->HDFS->Service Actions->Start
   ```

4. **Re-create the required directory structure**
   You need to SSH into a node that is running the HDFS Client and re-create the directory structure that Splice Machine expects by issuing these commands:

   Run the `zookeeper-client` command. At the prompt, run the following commands:

   ```
   sudo -su hdfs hadoop fs -rm -r /apps/hbase
   sudo -su hdfs hadoop fs -mkdir /apps/hbase
   sudo -su hdfs hadoop fs -mkdir /apps/hbase/data
   sudo -su hdfs hadoop fs -chown hbase:hdfs /apps/hbase
   sudo -su hdfs hadoop fs -chown hbase:hdfs /apps/hbase/data
   ```

5. **Restart HBase**
   Use the Ambari Dashboard to restart HBase:

   ```
   Services->HBase->Service Actions->Start
   ```

Cleaning Your Splice Machine Database on a MapR-Managed Cluster

Follow the steps below to clean (flatten) your database on your MapR cluster. You must be logged in as the cluster administrator (typically `clusteradmin` or `ec2-user`) to run each step. Unless otherwise specified, run each of these steps **on your cluster control node**; some steps, as indicated, must be run on each node in your cluster.

This is a destructive process and should only be used by an administrator!

1. **Stop the HBase RegionServers and Master:**
   Use the following command **on your control node** to stop HBase on your cluster:

   ```
   ~/splice-installer-mapr4.0/stop-hbase.sh
   ```

2. **Remove old data from HDFS:**
   Ignore any error messages you may see when you run this command:

   ```
   sudo -iu mapr hadoop fs -rm -r -f 'maprfs:///hbase/*'
   ```

3. **Stop MapR warden services:**
   Run the following command **on each node** in your cluster:

   ```
   sudo service mapr-warden stop
   ```

4. **Launch the ZooKeeper command line shell:**
   Note that the exact path may vary with different MapR versions

   ```
   /opt/mapr/zookeeper/zookeeper-3.4.5/bin/zkCli.sh
   ```

5. **Connect to the local ZooKeeper instance:**
   When the ZooKeeper command shell prompts you, enter the `connect` command shown here:

   ```
   Connecting to localhost:2181
   Welcome to ZooKeeper!
   JLine support is enabled
   [zk: localhost:2181(CONNECTING) 0] connect localhost:5181
   ```

6. **Complete the connection:**
   Press `Enter` again to display the connected prompt

   ```
   [zk: localhost:5181(CONNECTED) 1]
   ```

7. **Clear old ZooKeeper data:**
   Enter the following commands to clear ZooKeeper data and then exit the command shell:

```
rmr /splice
rmr /hbase
quit
```

8. **Restart MapR warden services on all nodes:**
   Run the following command **on each node** in your cluster:

   ```
   sudo service mapr-warden start
   ```

   Once you do so, your cluster will re-create the Splice Machine schema, and the command line interface will once again be available after a minute or so.

9. **Restart HBase**
   Run this command to restart hbase:

   ```
   ~/splice-installer-mapr4.0/start-hbase.sh
   ```

Cleaning Your Database in the Standalone Version

Follow these steps to clean your database if you're using the Standalone version of Splice Machine:

1. **Make sure that you have quit the `splice>` command line interpreter:**

   ```
   splice> quit;
   ```

2. **Change directory to your install directory:**

   ```
   cd splicemachine
   ```

3. **Run the following scripts:**

   ```
   $ ./bin/stop-splice.sh
   $ ./bin/clean.sh
   $ ./bin/start-splice.sh
   ```

Shutting Down Your Database

This is an On-Premise-Only topic! Learn about our products

This topic describes how to shut down your Splice Machine database. You need to follow different steps, depending on which version of Splice Machine you are using:

- Shutting Down Your Splice Machine Database on a Cloudera-Managed Cluster
- Shutting Down Your Splice Machine Database on a Hortonworks HDP-Managed Cluster
- Shutting Down Your Splice Machine Database on a MapR-Managed Cluster
- Shutting Down Your Splice Machine Database on a Standalone installation

Shutting Down Your Splice Machine Database on a Cloudera-Managed Cluster

Use the Cloudera Manager to either shut down HBase or to shut down the entire cluster, whichever is appropriate for your situation.

1. **Navigate to the *Services->All Services* screen in *Cloudera Manager*, and select this action to stop *HBase*:**

   ```
   hbase -> Actions -> Stop
   ```

2. **If you also want to shut down the entire cluster, select this action in the same screen to stop*HDFS*:**

   ```
   hdfs1 -> Actions -> Stop
   ```

Shutting Down Your Splice Machine Database on a Hortonworks HDP-Managed Cluster

Use the Ambari dashboard to shut down Splice Machine:

1. **Log in to the Ambari Dashboard by pointing your browser to the publicly visible `<hostName>` for your master node that is hosting Ambari Server:**

   ```
   http://<hostName>:8080/
   ```

2. **Shut down cluster services by selecting:**

   ```
   Action -> Stop All
   ```

Shutting Down Your Splice Machine Database on a MapR-Managed Cluster

To shut down Splice Machine, use the MapR Command System (MCS) to stop HBase.

1. **Navigate to the node that is running the `mapr-webserver` service.**
2. **Log into `https://<hostIPAddr>:8443`, substituting the correct `<hostIPAddr>` value.**
3. **Stop hbase.**

Shutting Down Your Database in the Standalone Version

Follow these steps to shut down your database if you're using the Standalone version of Splice Machine:

1. **Make sure that you have quit the `splice>` command line interpreter:**

   ```
   splice> quit;
   ```

2. **Change directory to your install directory:**

   ```
   cd splicemachine
   ```

3. **Run the following scripts:**

   ```
   $ ./bin/stop-splice.sh
   ```

This stops the Splice Machine database and the Splice Machine Administrative Console.

Derby Property Access

This is an On-Premise-Only topic! Learn about our products

This topic describes how to enable and disable Derby features by setting Derby properties, which are divided into categories, as shown in the following table.

NOTE: When a property value is used, the property is searched for in the order shown in the table; in other words, the property is searched for in the JVM properties; if not found there, it is searched for in the Service properties, then in the Database properties, and finally in the App properties.

Property Type	Description

JVM (System)	These properties can be as command line arguments to JVM:
	» For Maven, include the command line arguments as an `<argument>` element in the `pom.xml` file. For example:
	`<argument>-Dderby.language.logStatementText=true</argument>`
	» For shell scripts, you may need to manually add the argument to the java executable command line. For example:
	`java -Dderby.language.logStatementText=true ...`
	System properties can also be set manually using the `System.setProperty(key, value)` function.

Property Type	Description
Service	Service properties are a special type of database property that is required to boot the database; as such, these properties cannot be stored in the database. Instead, they are stored outside the database, as follows: » Derby stores service properties in the `service.properties file` » Splice Machine stores service properties in a Zookeeper element. You can temporarily change service properties with the `SYSCS_UTIL.SYSCS_SET_GLOBAL_DATABASE_PROPERTY` built-in system procedure. These changes will be lost when the server is next restarted. To make a permanent change in a Derby service property, modify the value in the `service.properties` file and then restart the server to apply the changes. To make a permanent change in a Splice Machine service property, modify the value in ZooKeeper and then restart the server to apply the changes.
Database	Splice Machine database properties are saved in a hidden HBASE table with `CONGLOMERATEID=16`.
App (Derby/ Splice)	App properties for both Derby and Splice Machine are saved to the `derby.properties` file in the Derby/Splice home directory. App properties can also be saved to these HBASE XML configuration files: » `hbase-default.xml` » `hbase-site.xml` » `splice-site.xml` Note that the XML files must reside in the `CLASSPATH`.

Site File Example

The following is an example of a `splice-site.xml` file:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
   <configuration>
       <property>
           <name>splice.debug.logStatementContext</name>
           <value>true</value>
           <description>Property  to enable logging of all statements.</description>
           </property>
          <property>
          <name>splice.debug.david</name>
          <value>true</value>
          <description>Property  to test something.</description>
          </property>
   </configuration>
```

Displaying Derby/Splice Properties

You can display the Derby/Splice properties with the `SYSCS_UTIL.SYSCS_GET_ALL_PROPERTIES` system procedure, which displays information like this:

```
splice> call SYSCS_UTIL.SYSCS_GET_ALL_PROPERTIES();

KEY                                       |VALUE            |TYPE
----------------------------------------------------------------------
derby.authentication.builtin.algorithm    |SHA-256          |DATABASE
derby.user.BROWSE                         |Browse           |APP
derby.engineType                          |2                |SERVICE
derby.david.foo                           |WINTERS          |APP
derby.connection.requireAuthentication    |false            |JVM
derby.locks.escalationThreshold           |500              |SERVICE
derby.database.defaultConnectionMode      |fullAccess       |SERVICE
derby.database.propertiesOnly             |false            |SERVICE
derby.database.collation                  |UCS_BASIC        |DATABASE
derby.language.logStatementText           |false            |JVM
derby.storage.propertiesId                |16               |SERVICE
derby.language.logQueryPlan               |true             |JVM
splice.updateSystemProcs                  |false            |JVM
derby.storage.rowLocking                  |false            |SERVICE
```

See Also

- `SYSCS_UTIL.SYSCS_SET_GLOBAL_DATABASE_PROPERTY`
- The *Derby Properties Guide* on the Apache Derby documentation site.

Enabling Enterprise Features in Splice Machine

This is an On-Premise-Only topic! Learn about our products

There are two mechanisms for enabling enterprise features in Splice Machine:

- To access enterprise features such as Backup and Restore, you can simply call the `SYSCS_UTIL.SYSCS_ENABLE_ENTERPRISE` built-in system procedure with the license key you obtain from Splice Machine, as described in the next section.
- To unlock Splice Machine Enterprise features that require configuration changes in your HBase settings, such as Kerberos and LDAP, you need to add one or more properties to your configuration file, as described in Using Configuration Properties to Upgrade to the Enterprise, below.

Using `SYSCS_UTIL.SYSCS_ENABLE_ENTERPRISE` to Upgrade

If you want to use Enterprise features, such as Backup and Restore, that do not need system properties updated, you can call the `SYSCS_UTIL.SYSCS_ENABLE_ENTERPRISE` system procedure to unlock those features. You only need to do this once:

1. Obtain your Splice Machine Enterprise Edition license key.
2. Enter this command on the `splice>` command line:

   ```
   splice> CALL SYSCS_UTIL.SYSCS_ENABLE_ENTERPRISE('<yourLicenseKey>');Statemen
   t executed.
   ```

   If you enter an invalid license key, you'll see an error message:

   ```
   splice> CALL SYSCS_UTIL.SYSCS_ENABLE_ENTERPRISE ('<bogus-license>');
   Error
   -------------------------------
   ERROR XSRSE: Unable to enable the enterprise Manager. Enterprise services ar
   e disabled. Contact your Splice Machine representative to enable.
   ```

Using Configuration Properties to Upgrade to the Enterprise

If your site uses Kerberos or LDAP, you need to update to the Enterprise version of Splice Machine by modifying your cluster's HBase configuration, and then restart Splice Machine. Follow these steps:

1. Obtain your Splice Machine Enterprise Edition license key.
2. Edit the `hbase-site.xml` configuration file, adding this property:

   ```
   <property>   <name>splicemachine.enterprise.key</name>   <value><your-Spl
   ice-Machine-license-key></value></property>
   ```

3. If you're using or switching from another authentication mechanism to LDAP, also add the LDAP properties to your `hbase-site.xml` file, as described in the Splice Machine Authentication and Authorization topic.
4. If you're using Kerberos, add this to your HBase Master Java Configuration Options:

   ```
   -Dsplice.spark.hadoop.fs.hdfs.impl.disable.cache=true
   ```

5. Restart Splice Machine, by first Starting Your Database.

Splice Machine Troubleshooting and Best Practices

This is an On-Premise-Only topic! Learn about our products

This topic provides troubleshooting guidance for these issues that you may encounter with your Splice Machine database:

- Restarting Splice Machine after an HMaster Failure
- Updating Stored Query Plans after a Splice Machine Update
- Increasing Parallelism for Spark Shuffles
- Force Compaction to Run Locally
- Kerberos Configuration Option
- Resource Allocation for Backup Jobs
- Bulk Import of Very Large Datasets with Spark 2.2

Restarting Splice Machine After HMaster Failure

If you run Splice Machine without redundant HMasters, and you lose your HMaster, follow these steps to restart Splice Machine:

1. Restart the HMaster node
2. Restart every HRegion Server node

Updating Stored Query Plans after a Splice Machine Update

When you install a new version of your Splice Machine software, you need to make these two calls:

```
CALL SYSCS_UTIL.SYSCS_UPDATE_METADATA_STORED_STATEMENTS();
CALL SYSCS_UTIL.SYSCS_EMPTY_STATEMENT_CACHE();
```

These calls will update the stored metadata query plans and purge the statement cache, which is required because the query plan APIs have changed. This is true for both minor (patch) releases and major new releases.

Increasing Parallelism for Spark Shuffles

You can adjust the minimum parallelism for Spark shuffles by adjusting the value of the `splice.olap.shuffle.partitions` configuration option.

This option is similar to the `spark.sql.shuffle.partitions` option, which configures the number of partitions to use when shuffling data for joins or aggregations; however, the `spark.sql.shuffle.partitions` option is set to allow a lower number of partitions than is optimal for certain operations.

Specifically, increasing the number of shuffle partitions with the `splice.olap.shuffle.partitions` option is useful when performing operations on small tables that generate large, intermediate datasets; additional, but smaller sized partitions allows us to operate with better parallelism.

The default value of `splice.olap.shuffle.partitions` is `200`.

Force Compaction to Run on Local Region Server

Splice Machine attempts to run database compaction jobs on an executor that is co-located with the serving Region Server; if it cannot find a local executor after a period of time, Splice Machine uses whatever executor Spark executor it can get; to force use of a local executor, you can adjust the `splice.spark.dynamicAllocation.minExecutors` configuration option.

To do so:

- Set the value of `splice.spark.dynamicAllocation.minExecutors` to the number of Region Servers in your cluster
- Set the value of `splice.spark.dynamicAllocation.maxExecutors` to equal to or greater than that number. Adjust these setting in the `Java Config Options` section of your HBase Master configuration.

The default option settings are:

```
-Dsplice.spark.dynamicAllocation.minExecutors=0
-Dsplice.spark.dynamicAllocation.maxExecutors=12
```

For a cluster with 20 Region Servers, you would set these to:

```
-Dsplice.spark.dynamicAllocation.minExecutors=20
-Dsplice.spark.dynamicAllocation.maxExecutors=20
```

Kerberos Configuration Option

If you're using Kerberos, you need to add this option to your HBase Master Java Configuration Options:

```
-Dsplice.spark.hadoop.fs.hdfs.impl.disable.cache=true
```

Resource Management for Backup Jobs

Splice Machine backup jobs use a Map Reduce job to copy HFiles; this process may hang up if the resources required for the Map Reduce job are not available from Yarn. To make sure the resources are available, follow these three configuration steps:

1. Configure minimum executors for Splice Spark
2. Verify that adequate vcores are available for Map Reduce tasks
3. Verify that adequate memory is available for Map Reduce tasks

Configure the minimum number of executors allocated to Splice Spark

You need to make sure that both of the following configuration settings relationships hold true.

```
(splice.spark.dynamicAllocation.minExecutors + 1) < (yarn.nodemanager.resource.cpu-v
cores * number_of_nodes)
```

```
(splice.spark.dynamicAllocation.minExecutors * (splice.spark.yarn.executor.memoryOve
rhead+splice.spark.executor.memory) + splice.spark.yarn.am.memory) < (yarn.nodemanag
er.resource.memory-mb * number_of_nodes)
```

The actual `minExecutors` allocated to Splice Spark may be less than specified in `splice.spark.dynamicAllocation.minExecutors` because of memory constraints in the container. Once Splice Spark is launched, Yarn will allocate the actual `minExecutor` value and memory to Splice Spark. You need to verify that enough vcores and memory remain available for Map Reduce tasks.

Verify that adequate vcores are available

The Map Reduce application master requires the following number of vcores:

```
yarn.app.mapreduce.am.resource.cpu-vcores * splice.backup.parallesim
```

There must be at least this many additional vcores available to execute Map Reduce tasks:

```
max{mapreduce.map.cpu.vcores,mapreduce.reduce.cpu.vcores}
```

Thus, the total number of vcores that must be available for Map Reduce jobs is:

```
yarn.app.mapreduce.am.resource.cpu-vcores * splice.backup.parallesim + max{mapreduc
e.map.cpu.vcores,mapreduce.reduce.cpu.vcores}
```

Verify that adequate memory is available

The Map Reduce application master requires this much memory:

```
yarn.scheduler.minimum-allocation-mb * splice.backup.parallesim
```

There must be at least this much memory available to execute Map Reduce tasks:

```
yarn.scheduler.minimum-allocation-mb
```

Thus, the total number of memory that must be available for Map Reduce jobs is:

```
yarn.scheduler.minimum-allocation-mb * (splice.backup.parallesim+1)
```

Bulk Import of Very Large Datasets with Spark 2.2

When using Splice Machine with Spark 2.2 with Cloudera, bulk import of very large datasets can fail due to direct memory usage. Use the following settings to resolve this issue:

Update Shuffle-to-Mem Setting

Modify the following setting in the Cloudera Manager's *Java Configuration Options for HBase Master*:

```
-Dsplice.spark.reducer.maxReqSizeShuffleToMem=134217728
```

Update the YARN User Classpath

Modify the following settings in the Cloudera Manager's *YARN (MR2 Included) Service Environment Advanced Configuration Snippet (Safety Valve)*:

```
YARN_USER_CLASSPATH=/opt/cloudera/parcels/SPARK2/lib/spark2/yarn/spark-2.2.0.clouder
a1-yarn-shuffle.jar:/opt/cloudera/parcels/SPARK2/lib/spark2/jars/scala-library-2.1
1.8.jar
YARN_USER_CLASSPATH_FIRST=true
```

About our Configuration Options

This is an On-Premise-Only topic! Learn about our products

The following table provides summary information about (some of) the configuration options used by Splice Machine.

Property	Typical value	Description
`splice.authentication`	`NATIVE`	This is documented in our Configuring Authentication topic.
`splice.authentication.native.algorithm`	SHA-512	This is documented in our Configuring Authentication topic.
`splice.client.write.maxDependentWrites`	`60000`	A form of write throttling that controls the maximum number of concurrent dependent writes from a single process. Dependent writes are writes to a table with indexes and generate more independent writes (writes to the indexes themselves). They are segregated so we can guarantee progress by reserving some IPC threads for independent writes.
`splice.client.write.maxIndependentWrites`	`60000`	A form of write throttling that controls the maximum number of concurrent independent writes from a single process.
`splice.compression`	`snappy`	The type of compression to use when compressing Splice Tables. This is set the same HBase sets table compression, and has the same codecs available to it (GZIP,Snappy, or LZO depending on what is installed).
`splice.debug.logStatementContext`	`false`	Whether to log all statements. Note that this is costly in OLTP workloads.
`splice.marshal.kryoPoolSize`	`1100`	The maximum number of Kryo objects to pool for reuse. This setting should only be adjusted if there are an extremely large number of operations allowed on the system.
`splice.olap_server.clientWaitTime`	`900000`	The number of milliseconds the OLAP client should wait for a result.

Property	Typical value	Description
`splice.root.path`	`/splice`	Zookeper root node for Splice Machine. All Zookeeper nodes used by Splice Machine will hang from this root node.
`splice.splitBlockSize`	`67108864`	Default size for Spark partitions when reading data from HBase. We sub-split each HBase region into several partitions targeting that size, but it rarely matches exactly. This is because it depend on a number of factors, including the number of storefiles in use by a given HBase region.
`splice.timestamp_server.clientWaitTime`	`120000`	The number of milliseconds the timestamp client should wait for a result.
`splice.txn.completedTxns.concurrency`	`128`	Concurrency level for the completed transactions cache. Specifies the estimated number of concurrently updating threads.
`splice.txn.concurrencyLevel`	`4096`	Expected concurrent updates to a transaction region. Increasing it increases memory consumption, decreasing it decreases concurrency on transaction operations. A reasonable default is the number of ipc threads configured for this system.