

[Releases](#)[Community](#)[Code](#)[Developers](#)[Feedback](#)[中文版](#)

Version (0.5.0-
incubating)

- Getting Started
- Quickstart**
- Use Cases
- Talks & Powered By
- Comparison
- Documentation

Quickstart

This guide provides a quick peek at Hudi's capabilities using spark-shell. Using Spark datasources, we will walk through code snippets that allows you to insert and update a Hudi dataset of default storage type: [Copy on Write](#). After each write operation we will also show how to read the data both snapshot and incrementally.

Setup spark-shell

Hudi works with Spark-2.x versions. You can follow instructions [here](#) for setting up spark.

From the extracted directory run spark-shell with Hudi as:

```
bin/spark-shell --packages org.apache.hudi:hudi-spark-bundle:0.5.0-incubating --conf 'spark.serializer=org.apache.spark.serializer.KryoSerializer'
```

Setup table name, base path and a data generator to generate records for this guide.

```
import org.apache.hudi.QuickstartUtils._
import scala.collection.JavaConversions._
import org.apache.spark.sql.SaveMode._
import org.apache.hudi.DataSourceReadOptions._
import org.apache.hudi.DataSourceWriteOptions._
import org.apache.hudi.config.HoodieWriteConfig._

val tableName = "hudi_cow_table"
val basePath = "file:///tmp/hudi_cow_table"
val dataGen = new DataGenerator
```

The [DataGenerator](#) can generate sample inserts and updates based on the the sample trip schema [here](#)

Insert data

Generate some new trips, load them into a DataFrame and write the DataFrame into the Hudi dataset as below.

- Setup spark-shell
- Insert data
- Query data
- Update data
- Incremental query
- Point in time query
- Where to go from here?



Version (0.5.0-incubating)

- Getting Started ▾
- Documentation ▾

Docker Demo

A Demo using docker containers

Lets use a real world example to see how hudi works end to end. For this purpose, a self contained data infrastructure is brought up in a local docker cluster within your computer.

The steps have been tested on a Mac laptop

Prerequisites

- Docker Setup : For Mac, Please follow the steps as defined in [<https://docs.docker.com/v17.12/docker-for-mac/install/>]. For running Spark-SQL queries, please ensure atleast 6 GB and 4 CPUs are allocated to Docker (See Docker -> Preferences -> Advanced). Otherwise, spark-SQL queries could be killed because of memory issues.
- kafkacat : A command-line utility to publish/consume from kafka topics. Use `brew install kafkacat` to install kafkacat
- /etc/hosts : The demo references many services running in container by the hostname. Add the following settings to /etc/hosts

```
127.0.0.1 adhoc-1
127.0.0.1 adhoc-2
127.0.0.1 namenode
127.0.0.1 datanode1
127.0.0.1 hiveserver
127.0.0.1 hivemetastore
127.0.0.1 kafkabroker
127.0.0.1 sparkmaster
127.0.0.1 zookeeper
```

Also, this has not been tested on some environments like Docker on Windows.

Setting up Docker Cluster

Build Hudi

The first step is to build hudi

- A Demo using docker containers
 - Prerequisites
 - Setting up Docker Cluster
 - Build Hudi
 - Bringing up Demo Cluster
 - Demo
 - Step 1 : Publish the first batch to Kafka
 - Step 2: Incrementally ingest data from Kafka topic
 - Step 3: Sync with Hive
 - Step 4 (a): Run Hive Queries
 - Step 4 (b): Run Spark-SQL Queries
 - Step 4 (c): Run Presto Queries
 - Step 5: Upload second batch to Kafka and run DeltaStreamer to ingest
 - Step 6(a): Run Hive Queries
 - Step 6(b): Run Spark SQL Queries
 - Step 6(c): Run Presto Queries
 - Step 7 : Incremental Query for COPY-ON-WRITE Table
 - Step 8: Schedule and Run Compaction for Merge-On-Read dataset
 - Step 9: Run Hive Queries including incremental queries
 - Testing Hudi in Local Docker environment
 - Building Local Docker Containers:



Version (0.5.0-
incubating)

- Getting Started ▾
- Documentation ▾

Docker Demo

A Demo using docker containers

Lets use a real world example to see how hudi works end to end. For this purpose, a self contained data infrastructure is brought up in a local docker cluster within your computer.

The steps have been tested on a Mac laptop

Prerequisites

- Docker Setup : For Mac, Please follow the steps as defined in [https://docs.docker.com/v17.12/docker-for-mac/install/]. For running Spark-SQL queries, please ensure atleast 6 GB and 4 CPUs are allocated to Docker (See Docker -> Preferences -> Advanced). Otherwise, spark-SQL queries could be killed because of memory issues.
- kafkacat : A command-line utility to publish/consume from kafka topics. Use `brew install kafkacat` to install kafkacat
- /etc/hosts : The demo references many services running in container by the hostname. Add the following settings to /etc/hosts

```
127.0.0.1 adhoc-1
127.0.0.1 adhoc-2
127.0.0.1 namenode
127.0.0.1 datanode1
127.0.0.1 hiveserver
127.0.0.1 hivemetastore
127.0.0.1 kafkabroker
127.0.0.1 sparkmaster
127.0.0.1 zookeeper
```

Also, this has not been tested on some environments like Docker on Windows.

Setting up Docker Cluster

Build Hudi

The first step is to build hudi

step-5-upload-second-batch-to-kafka-and-run-deltastreamer-to-ingest

A Demo using docker containers

Prerequisites

Setting up Docker Cluster

Build Hudi

Bringing up Demo Cluster

Demo

Step 1 : Publish the first batch to Kafka

Step 2: Incrementally ingest data from Kafka topic

Step 3: Sync with Hive

Step 4 (a): Run Hive Queries

Step 4 (b): Run Spark-SQL Queries

Step 4 (c): Run Presto Queries

Step 5: Upload second batch to Kafka and run DeltaStreamer to ingest

Step 6(a): Run Hive Queries

Step 6(b): Run Spark SQL Queries

Step 6(c): Run Presto Queries

Step 7 : Incremental Query for COPY-ON-WRITE Table

Step 8: Schedule and Run Compaction for Merge-On-Read dataset

Step 9: Run Hive Queries including incremental queries

Testing Hudi in Local Docker environment

Building Local Docker Containers:



Version (0.5.0-incubating)

- Getting Started
- Documentation

A Demo using docker containers

Lets use a real world example to see how hudi works end to end. For this purpose, a self contained data infrastructure is brought up in a local docker cluster within your computer.

The steps have been tested on a Mac laptop

Prerequisites

- Docker Setup** : For Mac, Please follow the steps as defined in [https://docs.docker.com/v17.12/docker-for-mac/install/]. For running Spark-SQL queries, please ensure atleast 6 GB and 4 CPUs are allocated to Docker (See Docker -> Preferences -> Advanced). Otherwise, spark-SQL queries could be killed because of memory issues.
- kafkacat** : A command-line utility to publish/consume from kafka topics. Use `brew install kafkacat` to install kafkacat
- /etc/hosts** : The demo references many services running in container by the hostname. Add the following settings to /etc/hosts

```
127.0.0.1 adhoc-1
127.0.0.1 adhoc-2
127.0.0.1 namenode
127.0.0.1 datanode1
127.0.0.1 hiveserver
127.0.0.1 hivemetastore
127.0.0.1 kafkabroker
127.0.0.1 sparkmaster
127.0.0.1 zookeeper
```

Also, this has not been tested on some environments like Docker on Windows.

Setting up Docker Cluster

Build Hudi

The first step is to build hudi

- A Demo using docker containers
 - Prerequisites
 - Setting up Docker Cluster
 - Build Hudi
 - Bringing up Demo Cluster
 - Demo
 - Step 1 : Publish the first batch to Kafka
 - Step 2: Incrementally ingest data from Kafka topic
 - Step 3: Sync with Hive
 - Step 4 (a): Run Hive Queries**
 - Step 4 (b): Run Spark-SQL Queries
 - Step 4 (c): Run Presto Queries
 - Step 5: Upload second batch to Kafka and run DeltaStreamer to ingest
 - Step 6(a): Run Hive Queries
 - Step 6(b): Run Spark SQL Queries
 - Step 6(c): Run Presto Queries
 - Step 7 : Incremental Query for COPY-ON-WRITE Table
 - Step 8: Schedule and Run Compaction for Merge-On-Read dataset
 - Step 9: Run Hive Queries including incremental queries