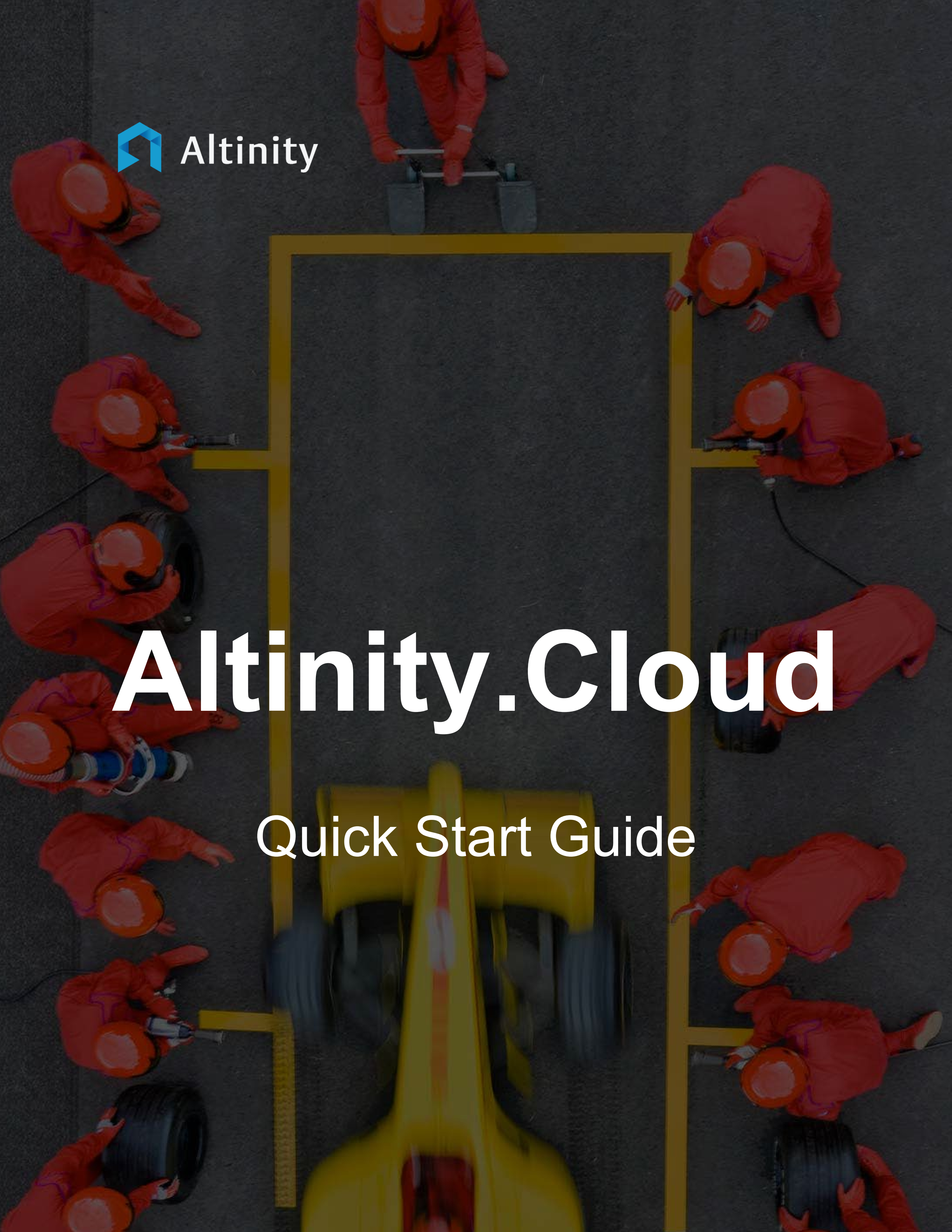


Altinity

Altinity.Cloud

Quick Start Guide

Table of Contents

Overview

Welcome to Altinity.Cloud! Altinity.Cloud is the fastest, easiest way to set up, administer, and use ClickHouse. Your ClickHouse are fully managed so you can focus on your work.

If this is your first time using Altinity.Cloud, this quick start guide will give you the minimum steps to become familiar with the system. When you're ready to dig deeper and use the full power of ClickHouse in your Altinity.Cloud environment, check out our Administrator and Developer Guides for in depth knowledge and best practices.
The Quick Start Guide is focused on these 6 major steps:

1. Create an Account
2. Login to Altinity.Cloud
3. Explore Altinity Cloud Manager
4. Create your First Cluster
5. Your First Queries
6. Connect to Cluster From a Remote Client

Create an Account

To start your Altinity.Cloud journey, the first thing you need is an account. New users can sign up for a test account on the Altinity.Cloud Test Drive page. Enter your contact information, and an Altinity.Cloud rep will get right with you.

Once finished, the Altinity.Cloud team will reach out to you with your login credentials. Altinity.Cloud uses your email address as your username, and you'll be assigned an initial password.

Login to Altinity.Cloud

There are two methods to login:

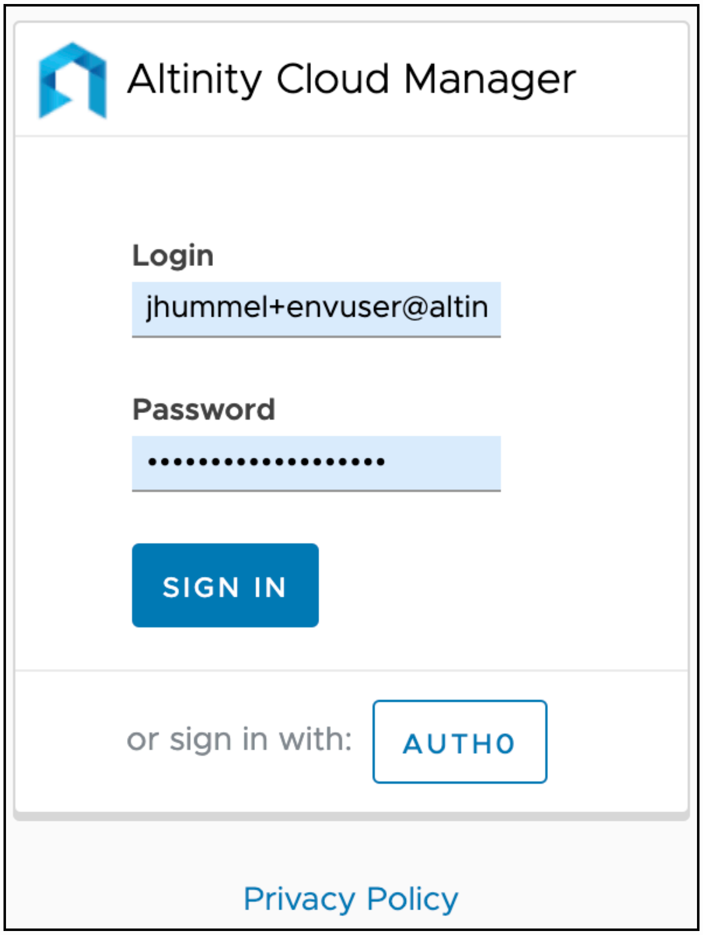
- Login with Username and Password
- Login with Auth0

Login with Username and Password

If you’ve used any web site, you’re likely familiar with this process.

To login to Altinity.Cloud with your username and password:

1. Enter your username - in this case your email address - in the field marked **Login**.
2. Enter your **Password**, then click **Sign In**.



Altinity Cloud Manager

Login

jhummel+envuser@altin

Password

•••••••••••••••••••

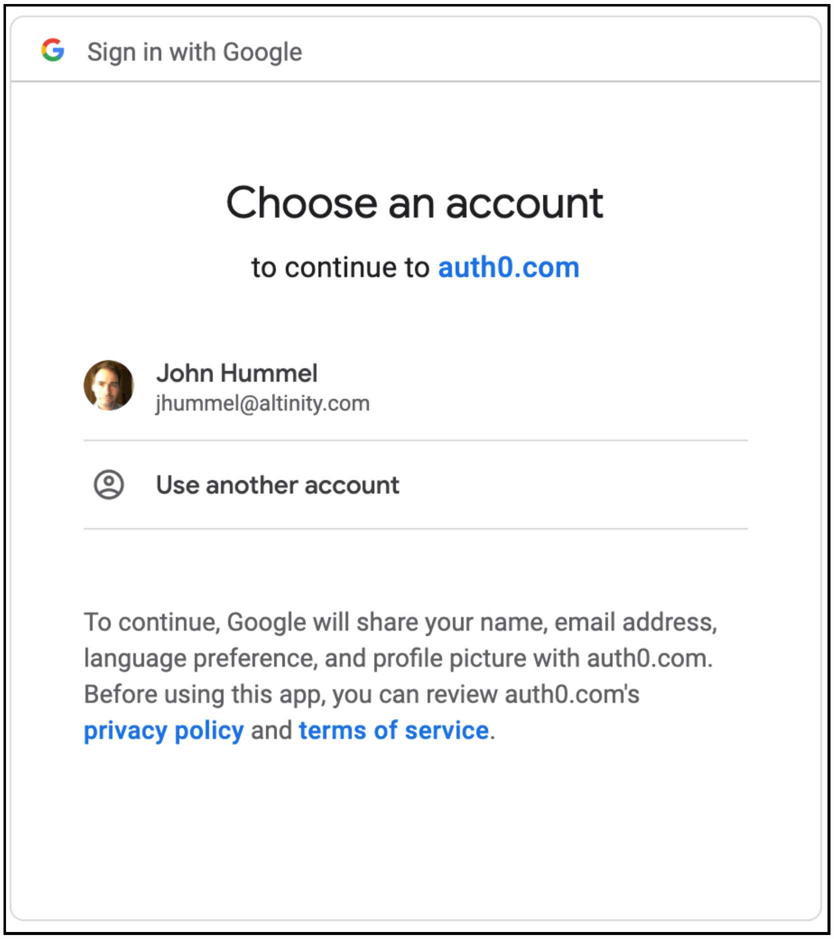
Login with Auth0

Auth0 allows you to authenticate to Altinity.Cloud through a trusted authentication provider, in this case Google. Once set up, you can click **Auth0** to automatically login.

- **Requirements**: In order to use Auth0, you must have a Google account with the same email address that you use for Altinity.Cloud.

To setup authentication with Auth0 for the first time:

1. Access the Altinity.Cloud page.
2. Select **Auth0**.
3. Select the Google account to use for authentication.
   IMPORTANT NOTE: The Google account must have the same email address as your Altinity.Cloud account.



Sign in with Google

Choose an account

to continue to auth0.com

John Hummel
jhummel@altinity.com

Use another account

To continue, Google will share your name, email address, language preference, and profile picture with auth0.com. Before using this app, you can review auth0.com's privacy policy and terms of service.

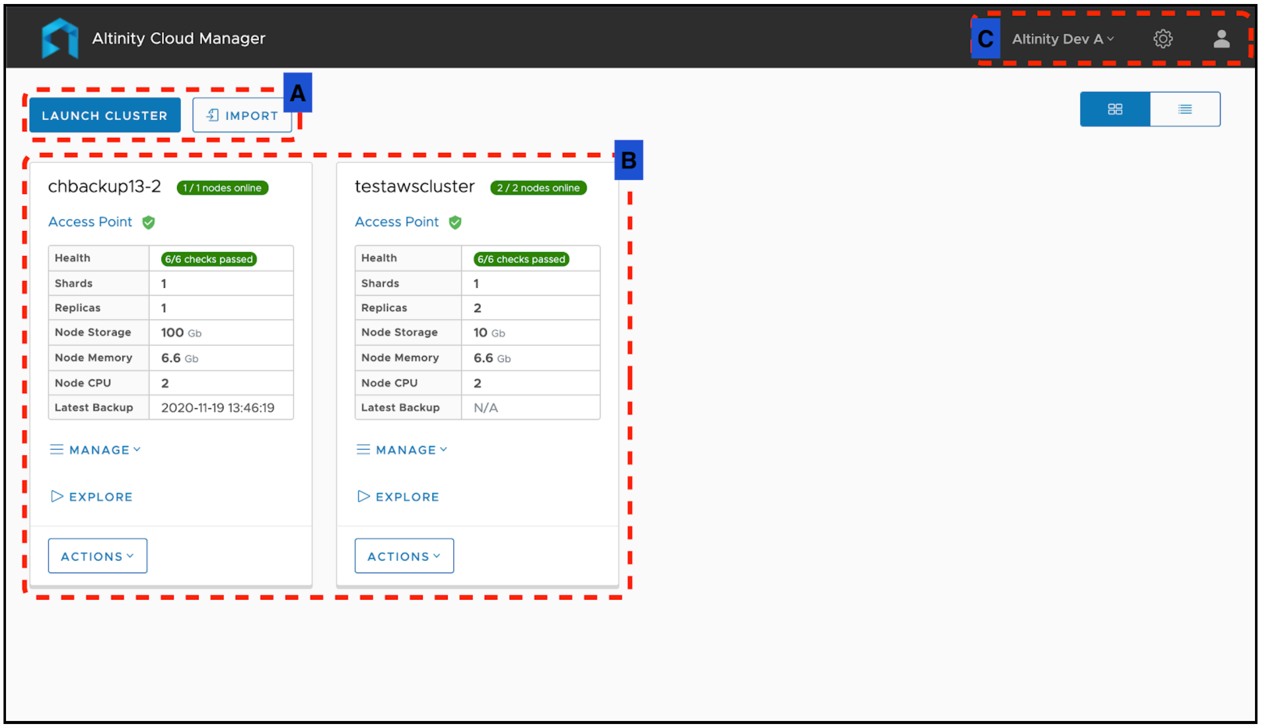
4. Select **Continue with Google**, and you'll be in Altinity.Cloud.

After you've completed the Auth0 setup process, you can login to Altinity.Cloud by selecting **Auth0** from the login page.

Explore Altinity Cloud Manager

Once you've logged in to Altinity.Cloud, let's take a moment and familiarize ourselves with the environment. The default page is the **Altinity Cloud Manager**.

The Altinity Cloud Manager page is separated into the following sections:



chbackup13-2

Health	6/6 checks passed
Shards	1
Replicas	1
Node Storage	100 Gb
Node Memory	6.6 Gb
Node CPU	2
Latest Backup	2020-11-19 13:46:19

testawscluster

Health	6/6 checks passed
Shards	1
Replicas	2
Node Storage	10 Gb
Node Memory	6.6 Gb
Node CPU	2
Latest Backup	N/A

- **A**: Cluster Creation: Clusters can be created from scratch with **Launch Cluster**, or the structure and settings configured with JSON through **Import**.
- **B**: Clusters: Each cluster associated with your Altinity.Cloud account is listed in either tile format, or as a short list. They'll display a short summary of their health and performance. By selecting a cluster, you can view the full details.
- **C**: User Management: Change which organization you are currently working in and update your Account Settings.

Create Your First Cluster

Time to make your first cluster! For this example, we're creating a minimally sized cluster, but you can upgrade your cluster later to make it the exact size you need for your ClickHouse needs. For more information, see the Administrator Guide.

To create your first cluster:

1. From the Altinity Cloud Manager page, select **Launch Cluster**. This starts the **Cluster Launch Wizard**.
2. The first page is **Resources Configuration**, where we set the name, size and authentication for the new cluster. When finished, click **Next**. Use the following settings:

Setting	Value
Name	Whatever name you like. Cluster names will be used to create the DNS name of the cluster. Therefore, cluster names must follow DNS name restrictions (letters, numbers, and dashes allowed, periods and special characters are not). Cluster names must be **lower case**.
Node Type	Select **m5.large**. This is the size of the node. This selection gives us a cluster with 2 CPUs and 6.5 Gb RAM. Recall that we can upgrade this cluster later if you want.
Node Storage	Set to **30 Gb**. The size of each Cluster node in Gb (gigabytes). Each node will have the same storage area.

Number of Volumes	Set to **1**. Network storage can be split into separate volumes. Use more volumes to increase query performance.
Volume Type	Select **gp2 (Not Encrypted)**. Volumes can be either encrypted or unencrypted, depending on your security requirements.
Number of Shards	Set to **1**. The shard represents a set of nodes. Shards can then be replicated to provide increased availability and recovery.
ClickHouse Version	Select **the most recent Altinity Stable Release**. Your ClickHouse cluster can use the version that best meets your needs. Note that **all nodes will run the same ClickHouse version**.
ClickHouse User Name	Autoset to **admin**. The default administrative user.
ClickHouse User Password and **Confirm Password**	Set to your security requirements. Both the ClickHouse User Password and Confirm Password must match.

3. The next page is High Availability Configuration. This is where you can set your replication, Zookeeper, and backup options. Use the following values for your first cluster, then click **Next** to continue:

Setting	Value
Data Replication	Set to **Enabled**.

	Data replication duplicates data across replica clusters for increased performance and availability.
Number of Replicas	Set to **2**. Only required if **Data Replication** is **Enabled**. Sets the number of replicas for each cluster shard.
Zookeeper Configuration	Select **Single Node** for development and test clusters. Select **Three Nodes** for production usage. • **IMPORTANT NOTE**: This setting cannot be changed later. Apache Zookeeper manages synchronization between the clusters.
Zookeeper Node Type	**Default** is selected by default.
Node Placement	Select **Separate Nodes**.
Enable Backups	Set to **Enabled**.

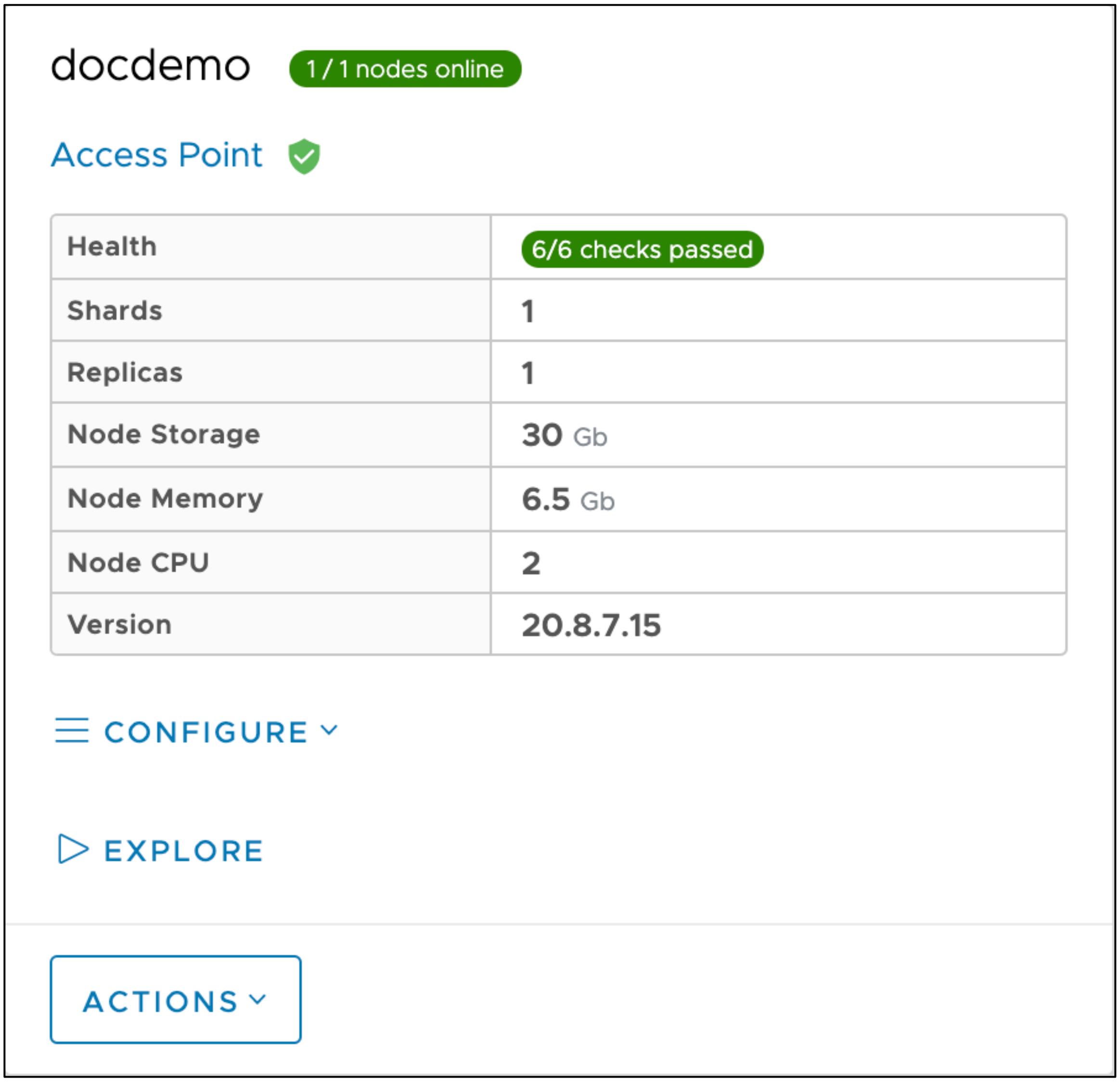
4. The **Connection Configuration** page determines how to communicate with your new cluster. Set the following values, then select **Next** to continue:

Setting	Value
DNS Name	This is automatically set based on your cluster name.
Use TLS	Set to **Enabled**.

	When enabled, communications with your cluster are encrypted with TLS.
Load Balancer Type	Select **Altinity Edge Ingress**. • **IMPORTANT NOTE**: This setting requires clients to support SNI (Server Name Indication). This will require the most current ClickHouse client and Python clickhouse-driver. This setting cannot be changed after the cluster is created.

5. Last page! **Review & Launch** lets you double check your settings. When you're ready, click **Launch**.

It will take a few minutes before your new cluster is ready, so grab your coffee or tea or other hydrating beverage. When it's complete, you'll see your new cluster with all nodes online and all health checks passed:



docdemo 1 / 1 nodes online

Access Point

Health	6/6 checks passed
Shards	1
Replicas	1
Node Storage	30 Gb
Node Memory	6.5 Gb
Node CPU	2
Version	20.8.7.15

Your First Queries

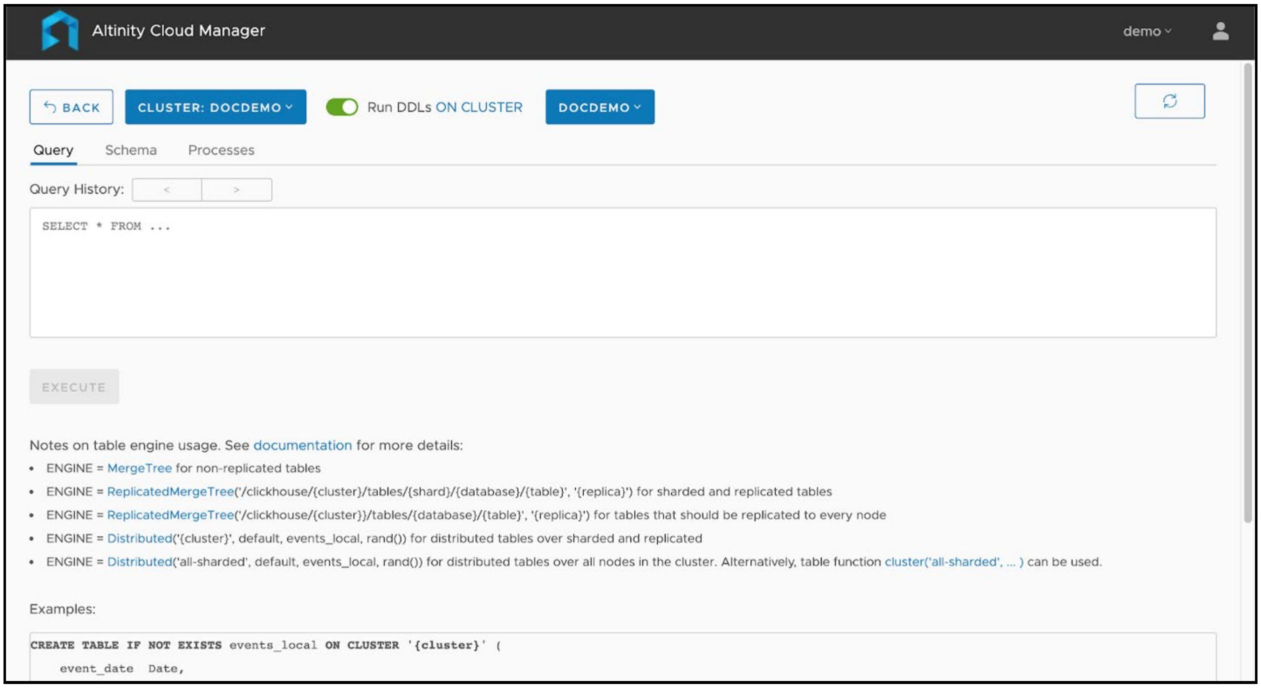
Once your cluster is created, time to create some tables and do some queries.

For those experienced with ClickHouse, this will be very familiar. For people who are new to ClickHouse, creating tables is very similar to other SQL based databases, with some extra syntax that defines the type of table we're making. This is part of what gives ClickHouse its speed. For complete details on ClickHouse commands, see the ClickHouse SQL Reference Guide.

Cluster Explore

The Cluster Explore page allows you to run queries, view the schema, and check on processes for your cluster. It’s a quick way of getting into your ClickHouse database, run commands and view your schema straight from your web browser. We’ll be using this to generate our first tables and input some data.

To access Cluster Explore for your cluster, just click **Explore** for the specific cluster to manage.



```
SELECT * FROM ...
```

Notes on table engine usage. See documentation for more details:

- ENGINE = MergeTree for non-replicated tables
- ENGINE = ReplicatedMergeTree('/clickhouse/{cluster}/tables/{shard}/{database}/{table}', '{replica}') for sharded and replicated tables
- ENGINE = ReplicatedMergeTree('/clickhouse/{cluster}}/tables/{database}/{table}', '{replica}') for tables that should be replicated to every node
- ENGINE = Distributed('{cluster}', default, events_local, rand()) for distributed tables over sharded and replicated
- ENGINE = Distributed('all-sharded', default, events_local, rand()) for distributed tables over all nodes in the cluster. Alternatively, table function cluster('all-sharded', ...) can be used.

Examples:

```
CREATE TABLE IF NOT EXISTS events_local ON CLUSTER '{cluster}' (
    event_date  Date,
```

For our example, we’re going to create two tables:

- **events_local**: This table will use the ReplicatedMergeTree table engine. If you don’t know about table engines, don’t worry about that for now. See the ClickHouse Engines page for complete information.
- **events**: This table will be distributed on your cluster with the Distributed table engine.

In our examples, we'll be using macro variables - these are placed between curly brackets and let us use the same SQL commands on different clusters and environments without having to fill in every detail. Any time you see an entry like `{cluster}` or `{shard}` you should recognize those as a macro variable.

The commands below will create these tables into the **default** database on your cluster.

Create Tables

To create your first tables:

1. From the **Altinity Cloud Manager** select **Explore** for the cluster to manage**.**
2. Select **Query**.
3. For our first table, copy and paste the following into the **Query** window, then select **Execute**.

```
CREATE TABLE IF NOT EXISTS events_local ON CLUSTER '{cluster}' (
    event_date  Date,
    event_type  Int32,
    article_id  Int32,
    title       String
) ENGINE =
ReplicatedMergeTree('/clickhouse/{cluster}/tables/{shard}/{database}/{table}',
'{replica}')
    PARTITION BY toYYYYMM(event_date)
    ORDER BY (event_type, article_id);
```

You should see the following under **Execute** confirming the commands ran, just replacing **docdemo** with your cluster:

```
docdemo.demo.beta.altinity.cloud:8443 (query time: 0.342s)
chi-docdemo-docdemo-0-0     9000  0          1     0
chi-docdemo-docdemo-0-1     9000  0          0     0
```

5. Now let’s create our second table. Back in the **Query** window, enter the following and select **Execute**:

```
CREATE TABLE events ON CLUSTER '{cluster}' AS events_local
    ENGINE = Distributed('{cluster}', default, events_local, rand())
```

Once again, you should see confirmation under **Execute**:

```
docdemo.demo.beta.altinity.cloud:8443 (query time: 0.162s)
chi-docdemo-docdemo-0-0     9000  0          1     0
chi-docdemo-docdemo-0-1     9000  0          0     0
```

5. Now that we have some tables, let’s not leave them empty. Inserting data into a ClickHouse table is very similar to most SQL systems. Let’s Insert our data, then do a quick Select on it. Enter the following Insert command into **Query**, then select **Execute**:

```
INSERT INTO events VALUES(today(), 1, 13, 'Example');
```

You’ll see the results confirmed under **Execute**, just like before.

```
OK.
```

Then enter the following **Select** command and select **Execute** again:

```
SELECT * FROM events;
```

The results will look like the following:

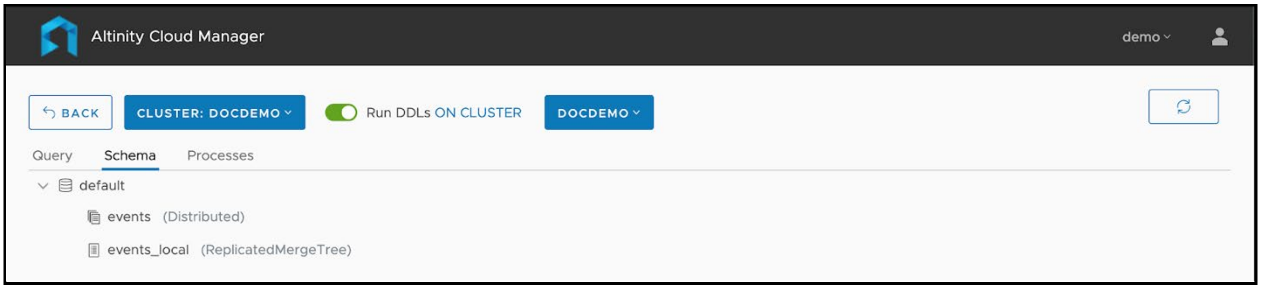
```
docdemo.demo.beta.altinity.cloud:8443 (query time: 0.018s)
┌─event_date─┬─event_type─┬─article_id─┬─title───┐
│ 2020-11-19 │          1 │         13 │ Example │
└────────────┴────────────┴────────────┴─────────┘
```

View Schema

The Database Schema shows a graphical view of your cluster's database, the tables in it, and their structure.

To view your Schema:

1. From the **Altinity Cloud Manager** select **Explore** for the cluster to manage**.**
2. Select **Schema.**



You can expand the databases to display the tables in each database, or select the table to view its details, schema, and some sample rows.

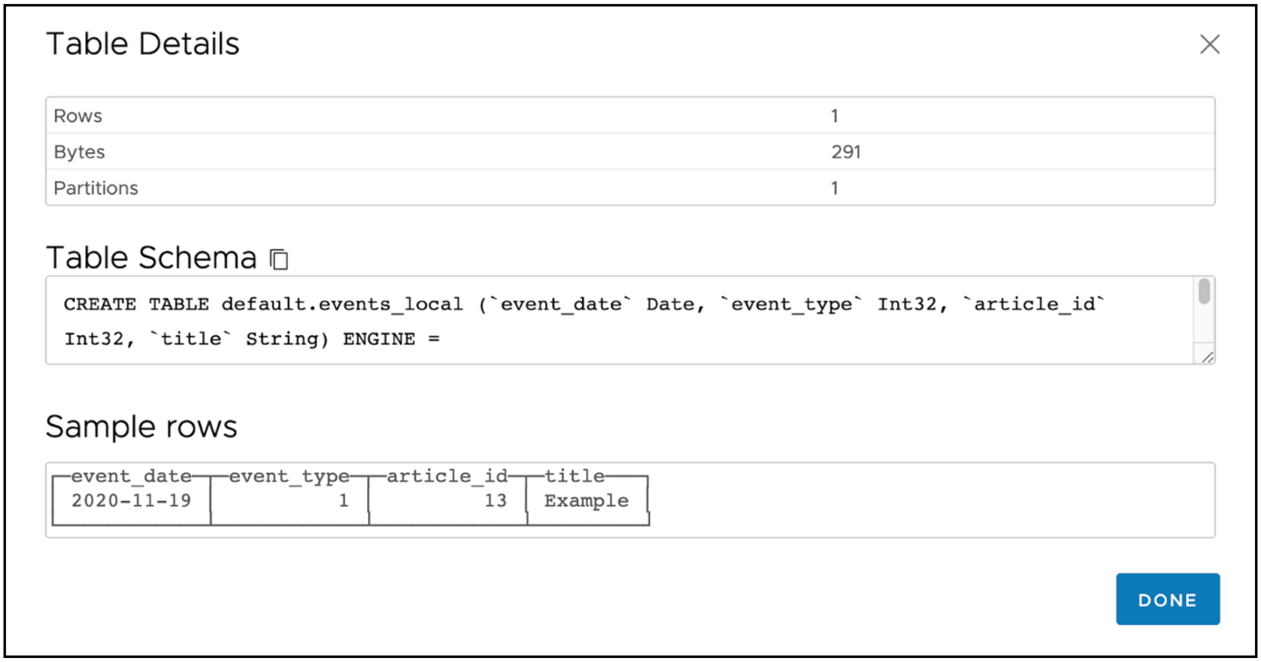


Table Details

Rows	1
Bytes	291
Partitions	1

Table Schema

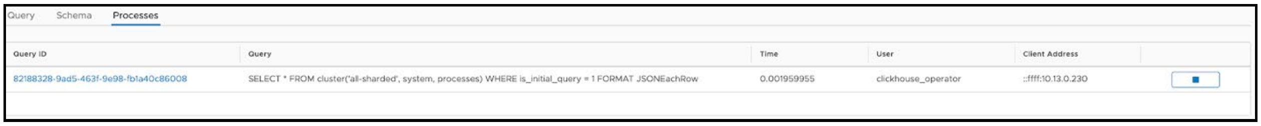
```
CREATE TABLE default.events_local (`event_date` Date, `event_type` Int32, `article_id`
Int32, `title` String) ENGINE =
```

Sample rows

```
┌─event_date─┬─event_type─┬─article_id─┬─title───┐
│ 2020-11-19 │          1 │         13 │ Example │
└────────────┴────────────┴────────────┴─────────┘
```

View Processes

To view current actions running on your cluster select **Processes**. This displays what processes are currently running, what user account they are running under, and allows you to view more details regarding the process.



Query ID	Query	Time	User	Client Address	
82188328-9ad5-463f-9e98-fb1a40c86008	SELECT * FROM cluster('all-sharded', system, processes) WHERE is_initial_query = 1 FORMAT JSONEachRow	0.001959955	clickhouse_operator	::ffff:10.13.0.230	■

Connect to Cluster From a Remote Client

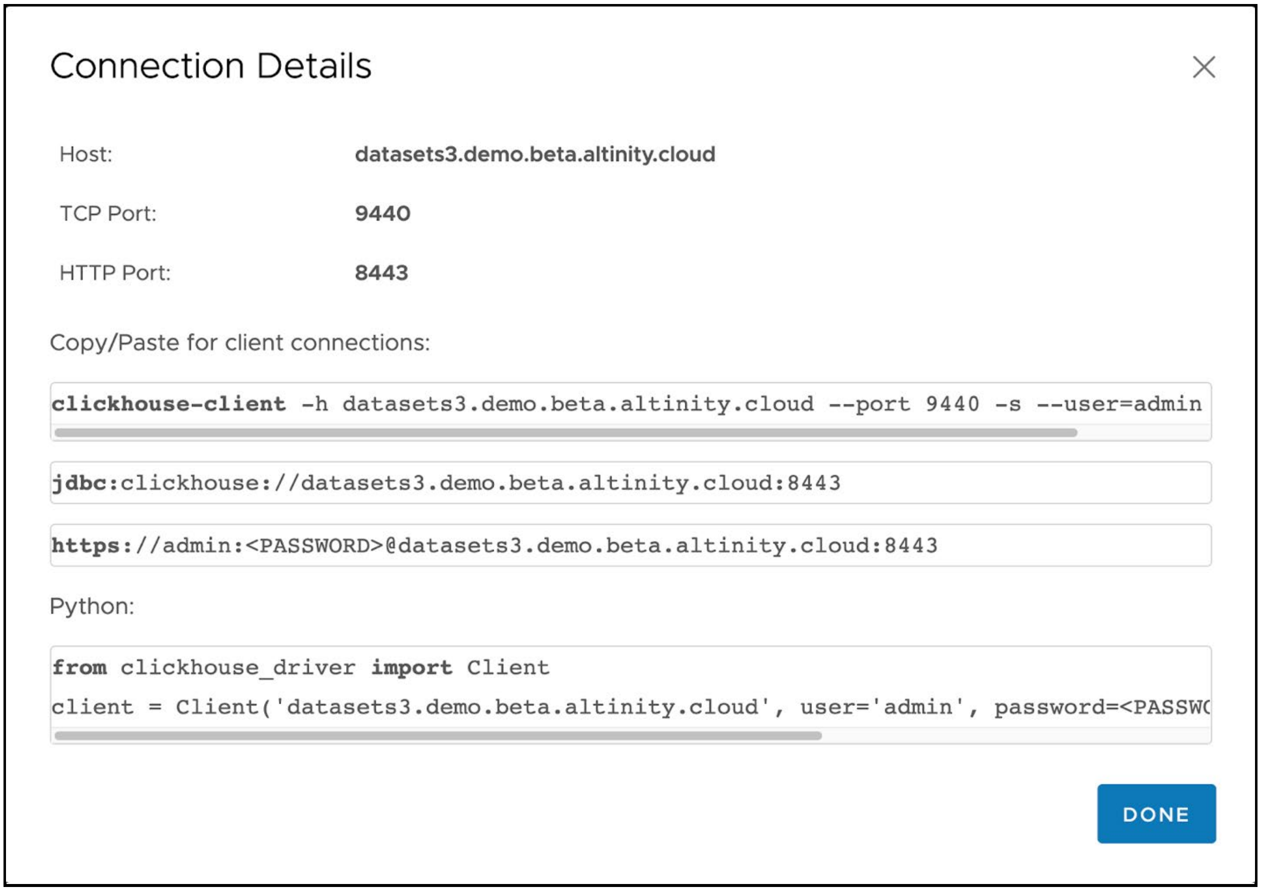
Now that we've shown how to create a cluster and use ClickHouse SQL queries into your new cluster, let's connect to it remotely.

For the following, we're going to be using the clickhouse-client program, but the same process will help you gain access from your favorite client.

Full instructions for installing ClickHouse can be found on the ClickHouse Installation page. We'll keep this simple and assume you're using a Linux environment like Ubuntu. For this example, we set up a virtual machine running Ubuntu 20.04.

First, we need to know our connection details for our Altinity.Cloud ClickHouse cluster. To view your connection details:

1. From the Altinity Cloud Manager, select **Access Point** for the cluster to manage.



Connection Details

Host: datasets3.demo.beta.altinity.cloud

TCP Port: 9440

HTTP Port: 8443

Copy/Paste for client connections:

```
clickhouse-client -h datasets3.demo.beta.altinity.cloud --port 9440 -s --user=admin
```

```
jdbc:clickhouse://datasets3.demo.beta.altinity.cloud:8443
```

```
https://admin:<PASSWORD>@datasets3.demo.beta.altinity.cloud:8443
```

Python:

```
from clickhouse_driver import Client
client = Client('datasets3.demo.beta.altinity.cloud', user='admin', password=<PASSW
```

2. From here, you can copy and paste the settings for the ClickHouse client. For example:

```
clickhouse-client -h yourdataset.yourcluster.altinity.cloud --port
9440 -s --user=admin --password
```

ClickHouse Client

The ClickHouse Client is a command line based program that will be familiar to SQL database users. With it you can connect to ClickHouse from your command line, issue commands and other related tasks.

Setting Up ClickHouse Client in Linux

If you’ve already set up ClickHouse client, then you can skip this step.

For those who need quick instructions on how to install ClickHouse client in their deb based Linux environment (like Ubuntu), use the following:

- **IMPORTANT NOTE**: As of this document’s publication, version 20.13 and above of the ClickHouse client is required to connect with the **SNI enabled clusters**. These instructions use the testing version of that client. An updated official stable build is expected to be released soon.

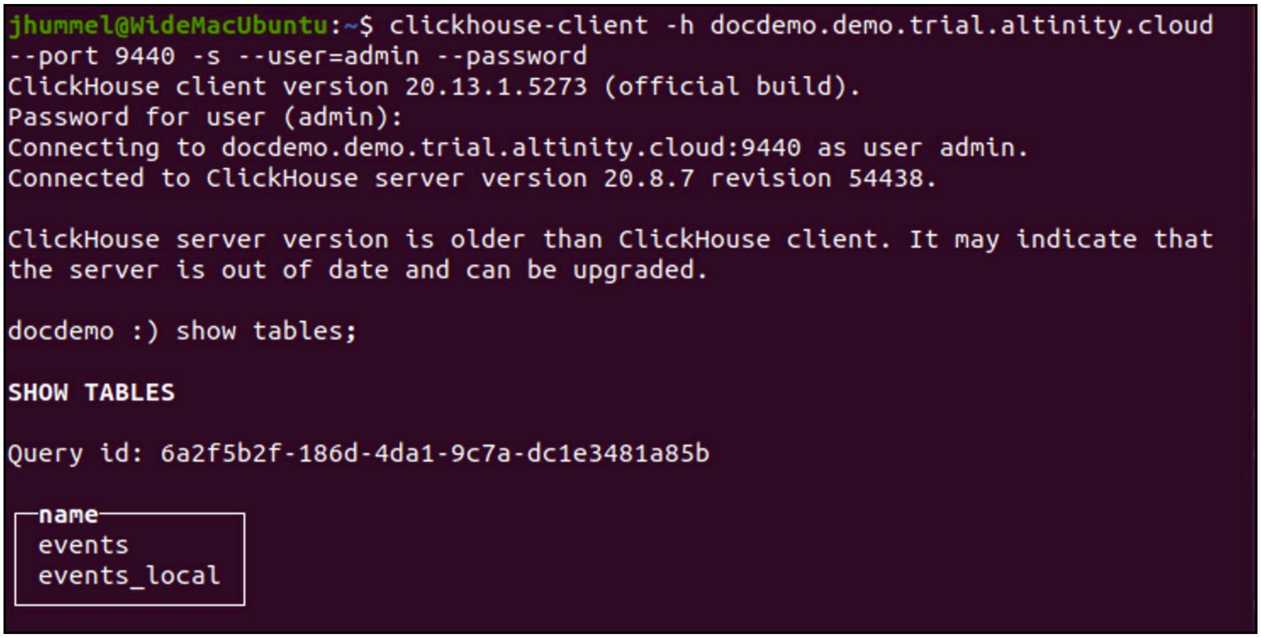
```
sudo apt-get install apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
E0C56BD4

echo "deb https://repo.clickhouse.tech/deb/testing/ main/" | sudo tee
\
    /etc/apt/sources.list.d/clickhouse.list
sudo apt-get update

sudo apt-get install -y clickhouse-client
```

Connect With ClickHouse Client

If your ClickHouse client is ready, then you can copy and paste your connection settings into your favorite terminal program, and you'll be connected.



```
jhummel@WideMacUbuntu:~$ clickhouse-client -h docdemo.demo.trial.altinity.cloud
--port 9440 -s --user=admin --password
ClickHouse client version 20.13.1.5273 (official build).
Password for user (admin):
Connecting to docdemo.demo.trial.altinity.cloud:9440 as user admin.
Connected to ClickHouse server version 20.8.7 revision 54438.

ClickHouse server version is older than ClickHouse client. It may indicate that
the server is out of date and can be upgraded.

docdemo :) show tables;

SHOW TABLES

Query id: 6a2f5b2f-186d-4da1-9c7a-dc1e3481a85b

┌─name─────────┐
│ events       │
│ events_local │
└──────────────┘
```

ClickHouse Python Driver

For users who prefer Python, they can use the **clickhouse-driver** to connect through Python. These instructions are very minimal, and are intended just to get you working in Python with your new Altinity.Cloud cluster.

These instructions are in the `bash` shell and require the following be installed in your environment:

- `Python 3.7` and above
- The Python module `venv`
- `git`

- **IMPORTANT NOTE**: We will be using clickhouse-driver v0.1.6 development build, which has support for Server Name Indication (SNI) when using TLS communications.

To connect with the Python clickhouse-driver:

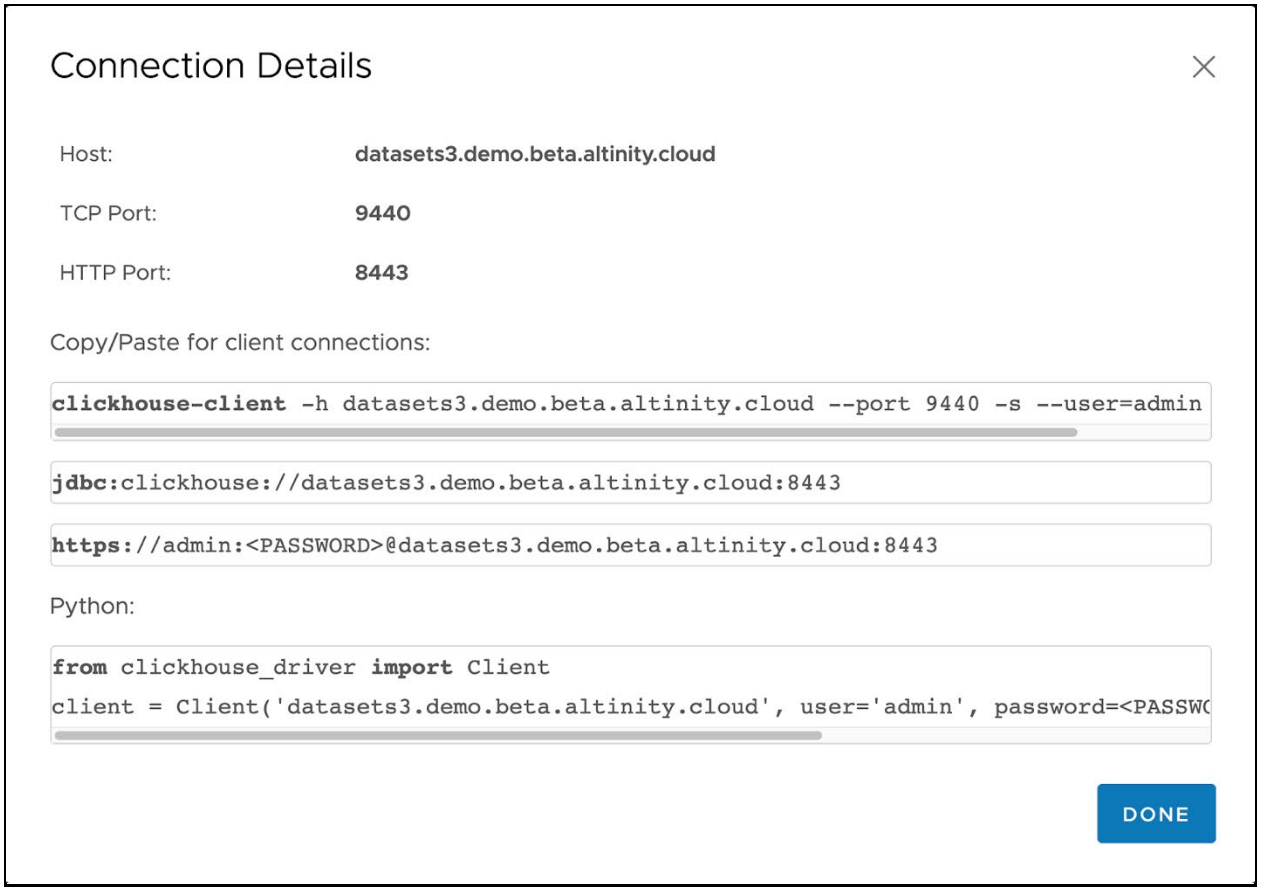
1. **(Optional)** Setup your local environment. For example:

```
python3 -m venv test
. test/bin/activate
```

2. Install the driver with pip3:

```
pip3 install git+https://github.com/mymarilyn/clickhouse-
driver@master#egg=clickhouse-driver
```

3. Start Python.
4. Add the client and connection details. Cluster Explore provides the necessary information to link directly to your cluster.



Connection Details

Host: datasets3.demo.beta.altinity.cloud

TCP Port: 9440

HTTP Port: 8443

Copy/Paste for client connections:

```
clickhouse-client -h datasets3.demo.beta.altinity.cloud --port 9440 -s --user=admin
```

```
jdbc:clickhouse://datasets3.demo.beta.altinity.cloud:8443
```

```
https://admin:<PASSWORD>@datasets3.demo.beta.altinity.cloud:8443
```

Python:

```
from clickhouse_driver import Client
client = Client('datasets3.demo.beta.altinity.cloud', user='admin', password=<PASSW
```

Import the clickhouse_driver client and enter the connection settings:

```
from clickhouse_driver import Client
client = Client('<HOSTNAME>', user='admin',
password=<PASSWORD>, port=9440, secure='y', verify=False)
```

5. Run client.execute and submit your query. Let's just look at the tables from within Python:

```
>>> client.execute('SHOW TABLES in default')
[('events',), ('events_local',)]
```

6. You can perform selects and inserts as you need. For example, continuing from our Your First Queries using **Cluster Explore** above:

```
>>> result = client.execute('SELECT * FROM default.events')
>>> print(result)
[(datetime.date(2020, 11, 23), 1, 13, 'Example')]
```

For more information see the article Clickhouse And Python: Getting To Know The Clickhouse-driver Client.

Conclusion

There are several ways of running ClickHouse to take advantage of the robust features and speed in your big data applications. Altinity.Cloud makes it easy to start up a cluster the way you need, manage it, and connect to it so you can go from concept to execution in the fastest way possible.

If you have any questions, please feel free to Contact Us at any time.

FAQS

When using Launch Cluster, I can't Click Next or complete the process

Make sure that all of your settings are filled in. Some common gotchas are:

- Make sure that the ClickHouse User Password field has been entered and confirmed.
- The cluster name should be in lower case. Cluster names must be compliant with DNS naming standards.