**Redis Database**

**What is Redis?**

Redis, which stands for Remote Dictionary Server, is a fast, open source, in-memory, key-value data store.

The project started when Salvatore Sanfilippo, the original developer of Redis, wanted to improve the scalability of his Italian start-up. From there, he developed Redis, which is now used as a database, cache, message broker, and queue.

Redis delivers sub-millisecond response times, enabling millions of requests per second for real-time applications in industries like gaming, ad-tech, financial services, healthcare, and IoT.

Today, Redis is one of the most popular open-source engines today, named the "Most Loved" database by Stack Overflow for five consecutive years. Because of its fast performance, Redis is a popular choice for [caching](https://aws.amazon.com/caching/), session management, gaming, leader boards, real-time analytics, geospatial, ride-hailing, chat/messaging, media streaming, and pub/sub apps.

**Benefits of Redis**

1. **Performance**

* All Redis data resides in memory, which enables low latency and high throughput data access.
* Unlike traditional databases, In-memory data stores don’t require a trip to disk, reducing engine latency to microseconds. Because of this, in-memory data stores can support an order of magnitude more operations and faster response times.
* The result is blazing-fast performance with average read and write operations taking less than a millisecond and support for millions of operations per second.

1. **Flexible data structures**

Unlike other key-value data stores that offer limited data structures, Redis has a vast variety of data structures to meet your application needs. Redis data types include:

* **Strings** – text or binary data up to 512MB in size
* **Lists** – a collection of Strings in the order they were added
* **Sets** – an unordered collection of strings with the ability to intersect, union, and diff other Set types
* **Sorted** **Sets** – Sets ordered by a value
* **Hashes** – a data structure for storing a list of fields and values
* **Bitmaps** – a data type that offers bit level operations
* **Hyper** **Loglogs** – a probabilistic data structure to estimate the unique items in a data set Streams - a log data structure Message queue Geospatial - a longitude-/latitude-based entries Maps, "nearby"

1. **Simplicity and ease-of-use**

* Redis enables you to write traditionally complex code with fewer, simpler lines.
* With Redis, you write fewer lines of code to store, access, and use data in your applications. The difference is that developers who use Redis can use a simple command structure as opposed to the query languages of traditional databases. For example, you can use the Redis hash data structure to move data to a data store with only one line of code.
* A similar task on a data store with no hash data structures would require many lines of code to convert from one format to another. Redis comes with native data structures and many options to manipulate and interact with your data. Over a hundred open-source clients are available for Redis developers.
* Supported languages include Java, Python, PHP, C, C++, C#, JavaScript, Node.js, Ruby, R, Go, and many others.

1. **Replication and persistence**

Redis employs a primary-replica architecture and supports asynchronous replication where data can be replicated to multiple replica servers. This provides improved read performance (as requests can be split among the servers) and faster recovery when the primary server experiences an outage. For persistence, Redis supports point-in-time backups (copying the Redis data set to disk).

1. **High availability and scalability**

Redis offers a primary-replica architecture in a single node primary or a clustered topology. This allows you to build highly available solutions providing consistent performance and reliability. When you need to adjust your cluster size, various options to scale up and scale in or out are also available. This allows your cluster to grow with your demands.

1. **Open Source**

Redis is an open-source project supported by a vibrant community, including AWS. There’s no vendor or technology lock in as Redis is open standards based, supports open data formats, and features a rich set of clients.

**Popular Use Cases**

1. **Caching**

* Redis is a great choice for implementing a highly available in-memory cache to decrease data access latency, increase throughput, and ease the load off your relational or NoSQL database and application.
* Redis can serve frequently requested items at sub-millisecond response times, and enables you to easily scale for higher loads without growing the costlier backend. Database query results caching, persistent session caching, web page caching, and caching of frequently used objects such as images, files, and metadata are all popular examples of caching with Redis.

1. **Chat, messaging, and queues**

Redis supports Pub/Sub with pattern matching and a variety of data structures such as lists, sorted sets, and hashes. This allows Redis to support high performance chat rooms, real-time comment streams, social media feeds and server intercommunication. The Redis List data structure makes it easy to implement a lightweight queue. Lists offer atomic operations as well as blocking capabilities, making them suitable for a variety of applications that require a reliable message broker or a circular list.

1. **Gaming Leaderboards**

Redis is a popular choice among game developers looking to build real-time leaderboards. Simply use the Redis Sorted Set data structure, which provides uniqueness of elements while maintaining the list sorted by users' scores. Creating a real-time ranked list is as easy as updating a user's score each time it changes. You can also use Sorted Sets to handle time series data by using timestamps as the score.

1. **Session Store**

Redis as an in-memory data store with high availability and persistence is a popular choice among application developers to store and manage session data for internet-scale applications. Redis provides the sub-millisecond latency, scale, and resiliency required to manage session data such as user profiles, credentials, session state, and user-specific personalization.

1. **Rich Media Streaming**

Redis offers a fast, in-memory data store to power live streaming use cases. Redis can be used to store metadata about users' profiles and viewing histories, authentication information/tokens for millions of users, and manifest files to enable CDNs to stream videos to millions of mobile and desktop users at a time.

1. **Geospatial**

Redis offers purpose-built in-memory data structures and operators to manage real-time geospatial data at scale and speed. Commands such as GEOADD, GEODIST, GEORADIUS, and GEORADIUSBYMEMBER to store, process, and analyze geospatial data in real-time make geospatial easy and fast with Redis. You can use Redis to add location-based features such as drive time, drive distance, and points of interest to your applications.

1. **Machine Learning**

Modern data-driven applications require machine learning to quickly process a massive volume, variety, and velocity of data and automate decision making. For use cases like fraud detection in gaming and financial services, real-time bidding in ad-tech, and matchmaking in dating and ride sharing, the ability to process live data and make decisions within tens of milliseconds is of utmost importance. Redis gives you a fast in-memory data store to build, train, and deploy machine learning models quickly.

1. **Real-Time Analytics**

Redis can be used with streaming solutions such as Apache Kafka and Amazon Kinesis as an in-memory data store to ingest, process, and analyze real-time data with sub-millisecond latency. Redis is an ideal choice for real-time analytics use cases such as social media analytics, ad targeting, personalization, and IoT.

**Difference Between Redis and Other Databases**

**1. Difference between Redis and MySQL :**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **REDIS** | **MYSQL** |
| **1.** | It was developed by Redis labs and initially released on May 10, 2009. | It was developed by Oracle and released on May 1995. |
| **2.** | It is written in ANSI and C languages. | It is written in C and C++. |
| **3.** | The primary database model for Redis is Key-Value Store. | The primary database model for MySQL is Relational DBMS. |
| **4.** | It is Data Schema free. | In MySQL Data Schema is fixed. |
| **5.** | It does not support XML data format. | It supports XML data format. |
| **6.** | It supports secondary indexes with RediSearch module only. | It supports secondary indexes without any restrictions. |
| **7.** | It has no Triggers. | It supports Triggers. |
| **8.** | It does not provide the concept of Referential Integrity. Hence, no Foreign Keys. | It provides concept of Referential Integrity and have Foreign keys. |
| **9.** | Server operating systems for Redis are BDS, Linux, OS X and Windows. | Server operating systems for MySQL are FreeBSD, Linux, OS X, Solaris, Windows. |
| **10.** | Famous companies like Twinkl Educational Publishing, Merrill Corporation, ASOS.com Limited, Owler, Inc., etc use Redis. | Famous companies like Airbnb, Pinterest, Slack, Udemy, Twitter, etc uses MySQL. |

**2. Difference between Redis and MongoDB:**

| **S.NO.** | **Redis** | **MongoDB** |
| --- | --- | --- |
| **1.** | It was developed by Redis labs and initially released on May 10, 2009. | It was developed by MongoDB Inc. and initially released on 11 February, 2009. |
| **2.** | Redis is written in ANSI and [C](https://www.geeksforgeeks.org/c-programming-language/) languages. | MongoDB is written in [C++](https://www.geeksforgeeks.org/c-plus-plus/), [Go](https://www.geeksforgeeks.org/golang/), [JavaScript](https://www.geeksforgeeks.org/javascript-tutorial/) and [Python](https://www.geeksforgeeks.org/python-programming-language/) languages. |
| **3.** | The primary database model for Redis is Key-Value Store. | The primary database model for MongoDB is Document Store. |
| **4.** | Redis supported data types are strings, hashes, lists, sets and sorted sets, bit arrays, hyperloglogs and geospatial indexes. Hence, have partial predefined data types. | MongoDB supported data types are string, integer, double, decimal, boolean, date, object\_id, geospatial. Hence, have predefined data types. |
| **5.** | Redis supports secondary indexes with RediSearch module only. | MongoDB supports secondary indexes without any restrictions. |
| **6.** | The Server-side scripting in Redis is through Lua. | The Server-side scripting in MongoDB is through JavaScript. |
| **7.** | Redis supports both Master-Slave Replication and Master-Master Replication. | MongoDB supports only Master-Slave Replication. |
| **8.** | Redis does not support Map Reduce method. | MongoDB supports Map Reduce method. |
| **9.** | Server operating systems for Redis are BDS, Linux, OS X and Windows. | Server operating systems for MongoDB are Solaris, Linux, OS X and Windows. |
| **10.** | Some companies like Twinkl Educational Publishing, Merrill Corporation, ASOS.com Limited, Owler, Inc., etc use Redis. | Some companies like Adobe, Amadeus, Lyft, ViaVarejo, Craftbase, etc use MongoDB. |

**Redis Data Types**

Binary-safe strings.

**Lists**: collections of string elements sorted according to the order of insertion. They are basically linked lists.

**Sets**: collections of unique, unsorted string elements.

**Sorted** **sets**, similar to Sets but where every string element is associated to a floating number value, called score. The elements are always taken sorted by their score, so unlike Sets it is possible to retrieve a range of elements (for example you may ask: give me the top 10, or the bottom 10).

**Hashes**, which are maps composed of fields associated with values. Both the field and the value are strings. This is very similar to Ruby or Python hashes.

**Bit** **arrays** (or simply bitmaps): it is possible, using special commands, to handle String values like an array of bits: you can set and clear individual bits, count all the bits set to 1, find the first set or unset bit, and so forth.

**Hyper** **Loglogs**: this is a probabilistic data structure which is used in order to estimate the cardinality of a set. Don't be scared, it is simpler than it seems... See later in the HyperLogLog section of this tutorial.

**Streams**: append-only collections of map-like entries that provide an abstract log data type. They are covered in depth in the Introduction to Redis Streams.

**Redis Installation**

**Step 1: Turn on Windows Subsystem for Linux**

In Windows 10, Microsoft replaced Command Prompt with PowerShell as the default shell. Open PowerShell as Administrator and run this command to enable Windows Subsystem for Linux (WSL):

>> Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux

Reboot Windows after making the change — note that you only need to do this once.

**Step 2: Launch Microsoft Windows Store**

>> start ms-windows-store:

**Step 3: Install Redis server**

Installing Redis is simple and straightforward. The following example works with Ubuntu (you'll need to wait for initialization and create a login upon first use):

>> sudo apt-add-repository ppa:redislabs/redis

>> sudo apt-get update

>> sudo apt-get upgrade

>> sudo apt-get install redis-server

Please note that the sudo command might or mightn't be required based on the user configuration of your system.

**Step 4: Restart the Redis server**

Restart the Redis server as follows:

>> sudo service redis-server restart

Step 5: Verify if your Redis server is running#

Use the redis-cli command to test connectivity to the Redis database.

>> $ redis-cli

>> 127.0.0.1:6379> set user:1 "Jane"

>> 127.0.0.1:6379> get user:1

"Jane"

Please note: By default, Redis has 0-15 indexes for databases, you can change that number databases NUMBER in redis.conf.

**Step 6: Stop the Redis Server**

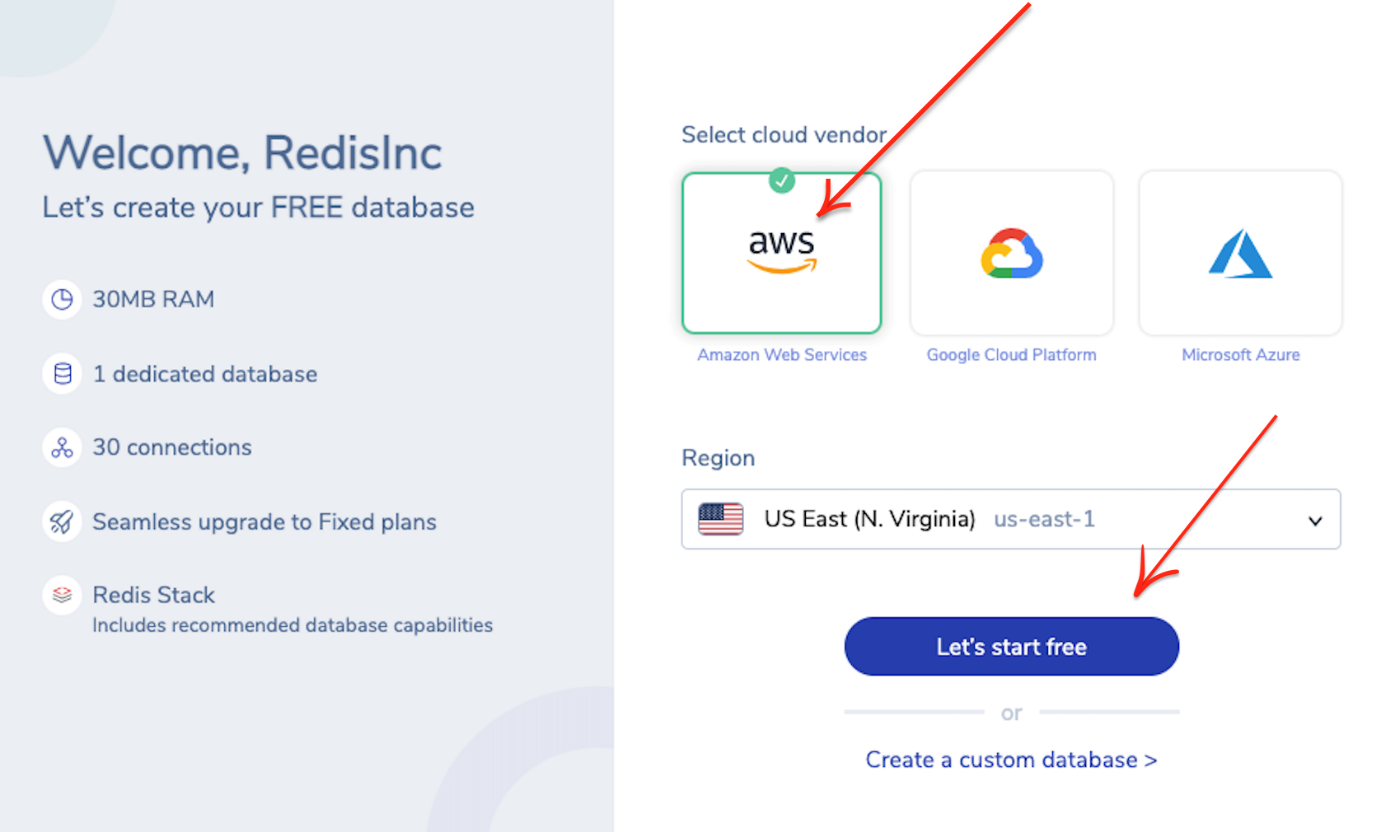
>> sudo service redis-server stop

**Step 1. Create a free Cloud account**

Create your free [Redis Enterprise Cloud account](https://redis.com/try-free/). Once you click on “Get Started”, you will receive an email with a link to activate your account and complete your signup process.

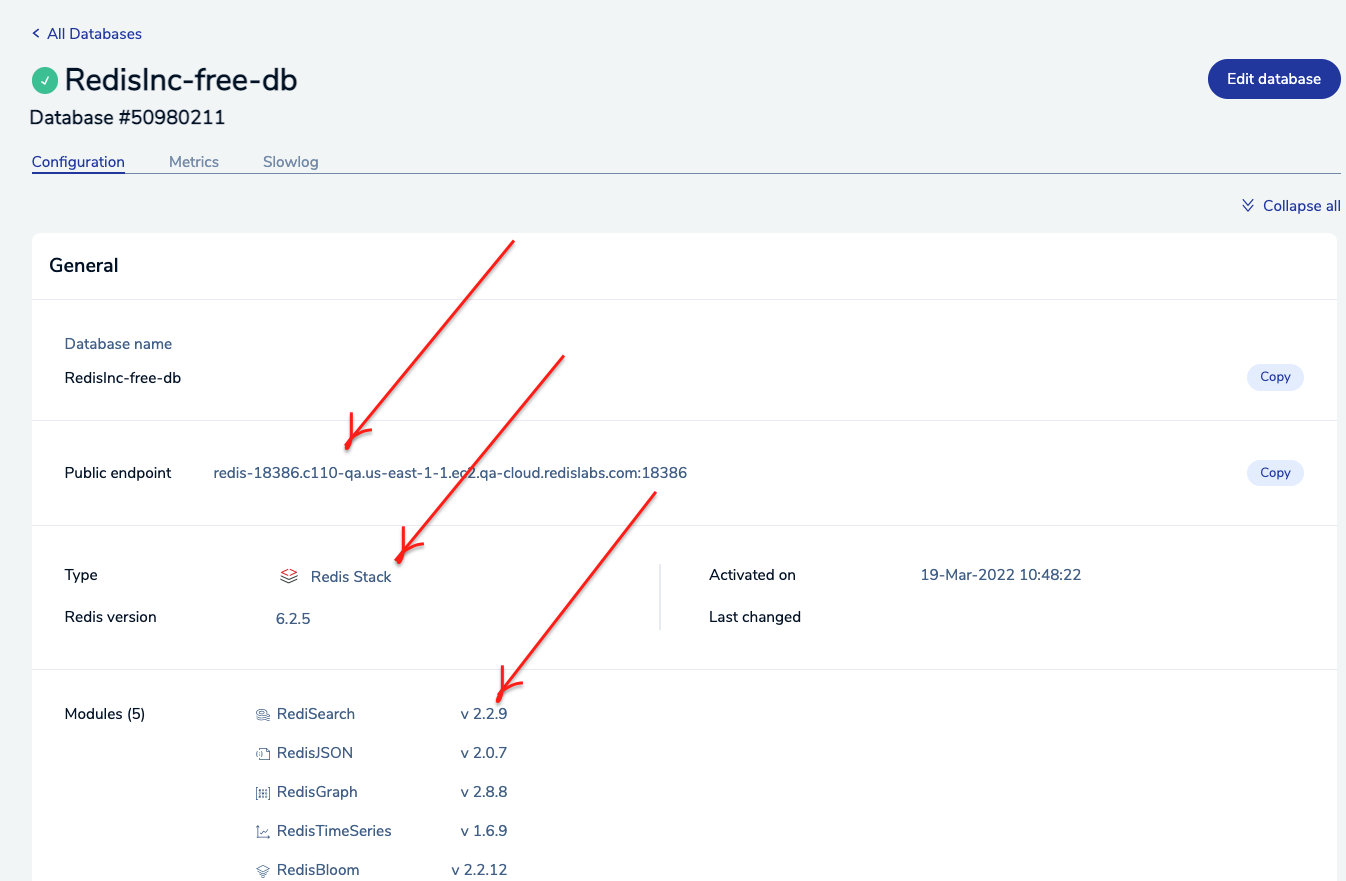
**Step 2. Create a database**

Choose your preferred cloud vendor. Select the region and then click "Let's start free" to create your free database automatically.



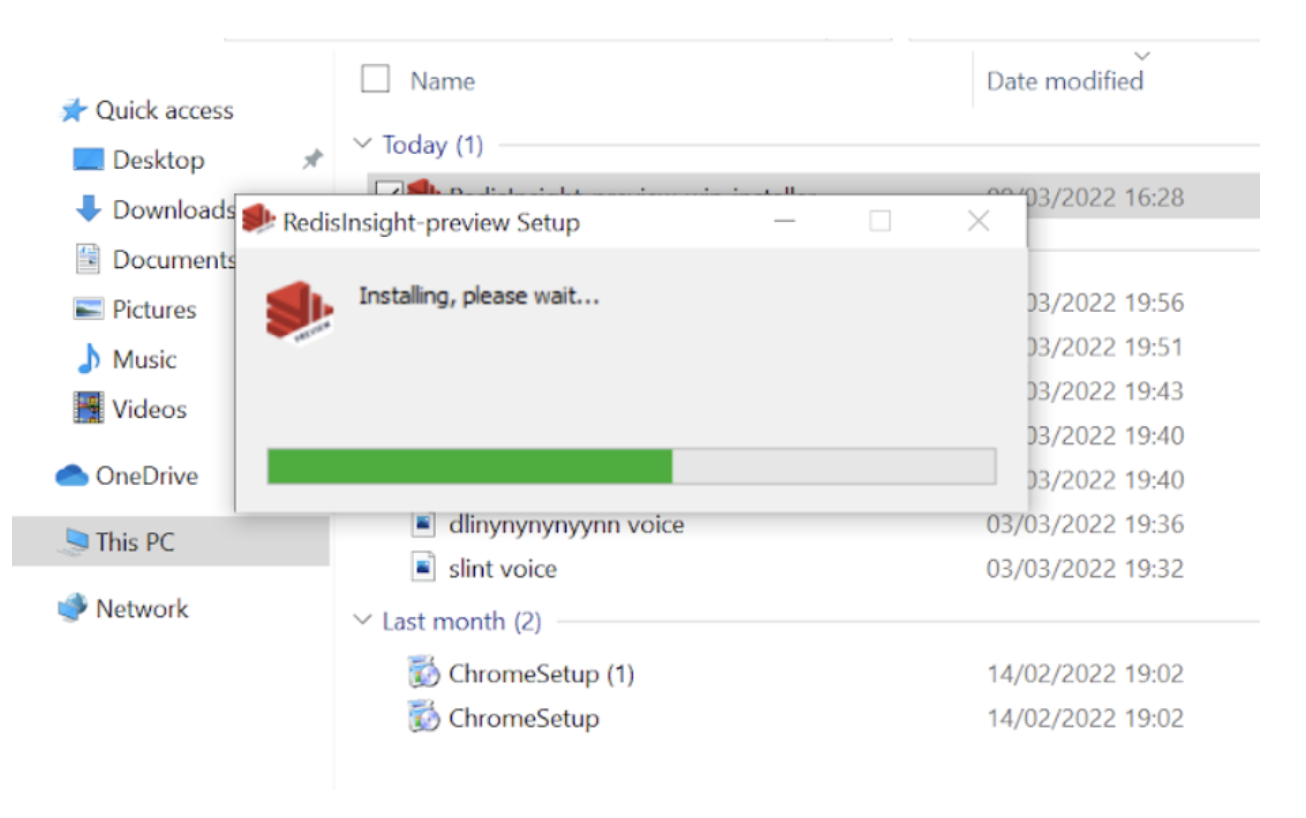
**Step 3. Verify the database details**

You will be provided with Public endpoint URL and "Redis Stack" as the type of database with the list of modules that comes by default.

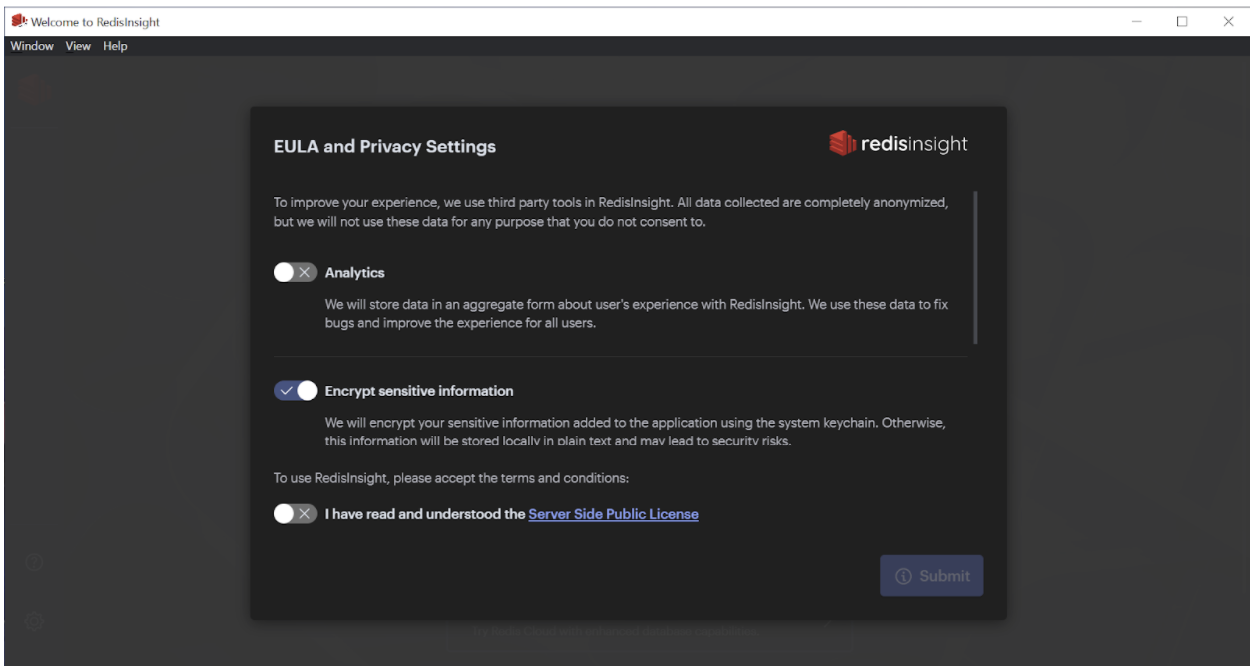


**Step 4. Install RedisInsight**

Click on the RedisInsight executable (.exe file) and install it in your system.

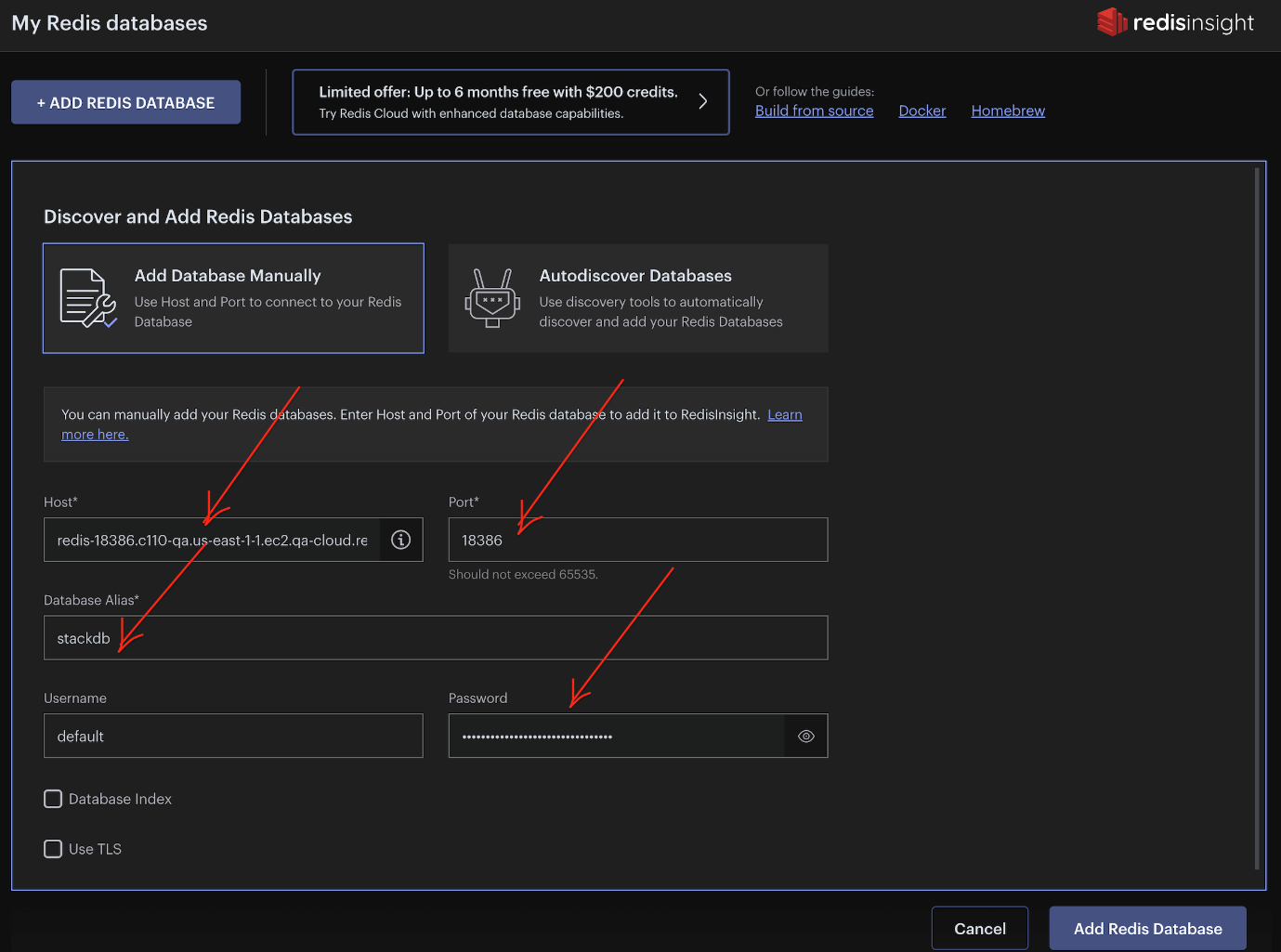


Once the RedisInsight software is installed, click on its icon to open the RedisInsight application. It will display the End-User License Agreement and Privacy Settings. Enable Analytics and Encrypt sensitive information as per your preference.

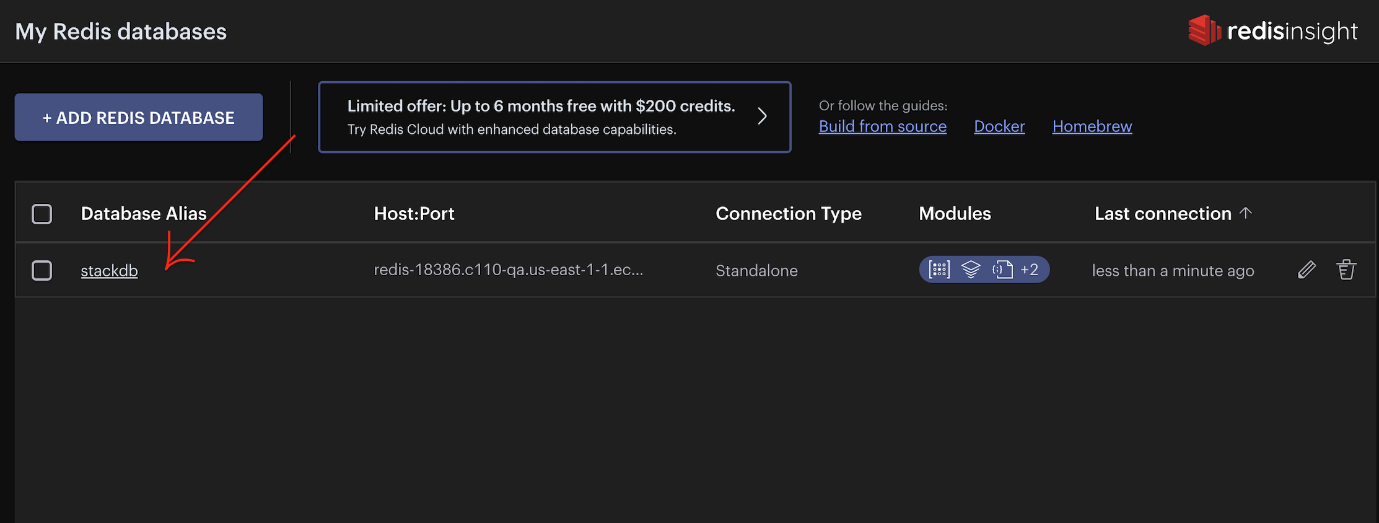


**Step 5. Connect to the Redis Database**

Enter the requested details, including Host (endpoint), Port, and Alias in the form, as shown below. You can use "default" as the username for now. Then click “ADD REDIS DATABASE”.

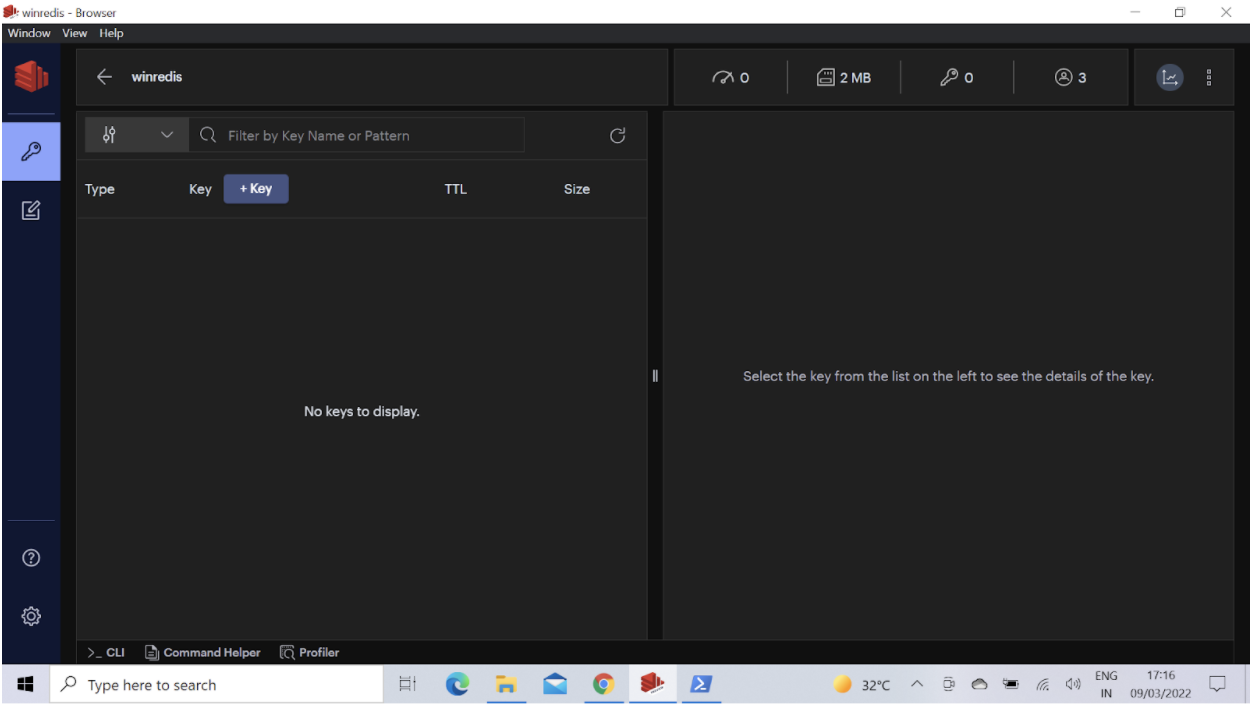


Once added, you will see the database name listed as shown below:



**Step 6. Use "Browser Tool"**

Click on the "Key" icon on the left sidebar to open up the browser tool.



**Step 5. Overview of User database keys**

Let us import a user database (6k keys). This dataset contains users stored as Redis Hashes.

**Users**

The user hashes contain the following fields:

* user:id : The key of the hash.
* first\_name : First Name.
* last\_name : Last name.
* email : email address.
* gender : Gender (male/female).
* ip\_address : IP address.
* country : Country Name.
* country\_code : Country Code.
* city : City of the user.
* longitude : Longitude of the user.
* latitude : Latitude of the user.
* last\_login : Epoch time of the last login.

**Step 6. Clone the repository**

Open up the CLI terminal and run the following commands:

git clone https://github.com/redis-developer/redis-datasets

cd redis-datasets/user-database

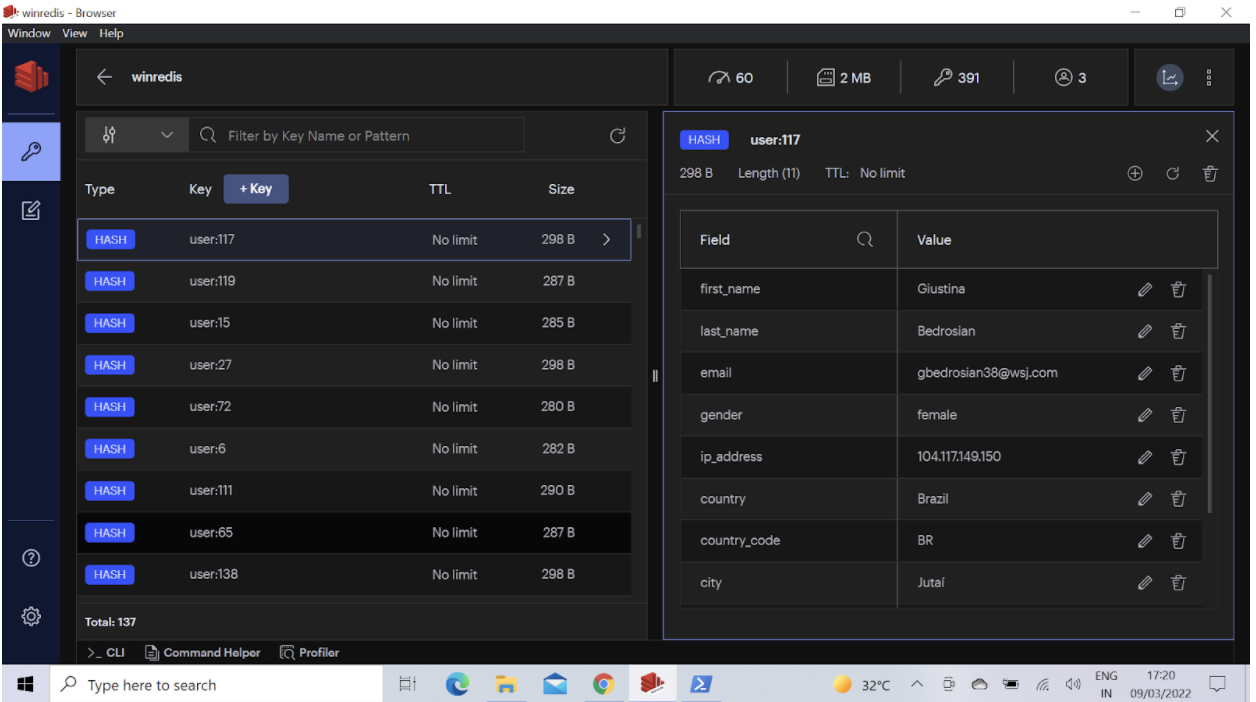
### Step 7. Import the user database keys

Open up the CLI terminal and run the following command.

NOTE: You will need a hostname, port and password to run this for a cloud database.

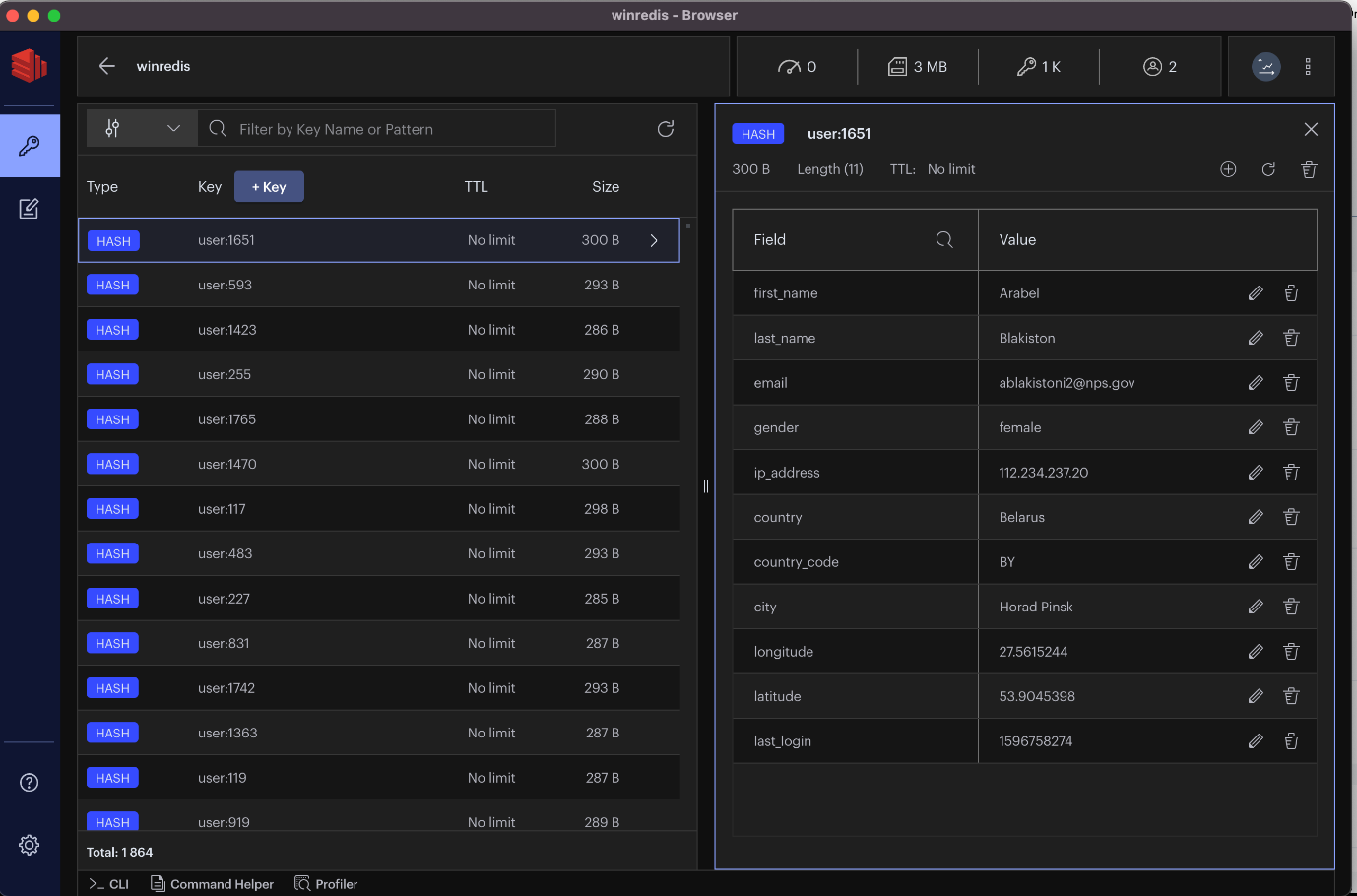
redis-cli -h redis-18386.c110-qa.us-east-1-1.ec2.cloud.redislabs.com -p 18386 -a <enter your password> < ./import\_users.redis

Refresh the key’s view by clicking as shown below:



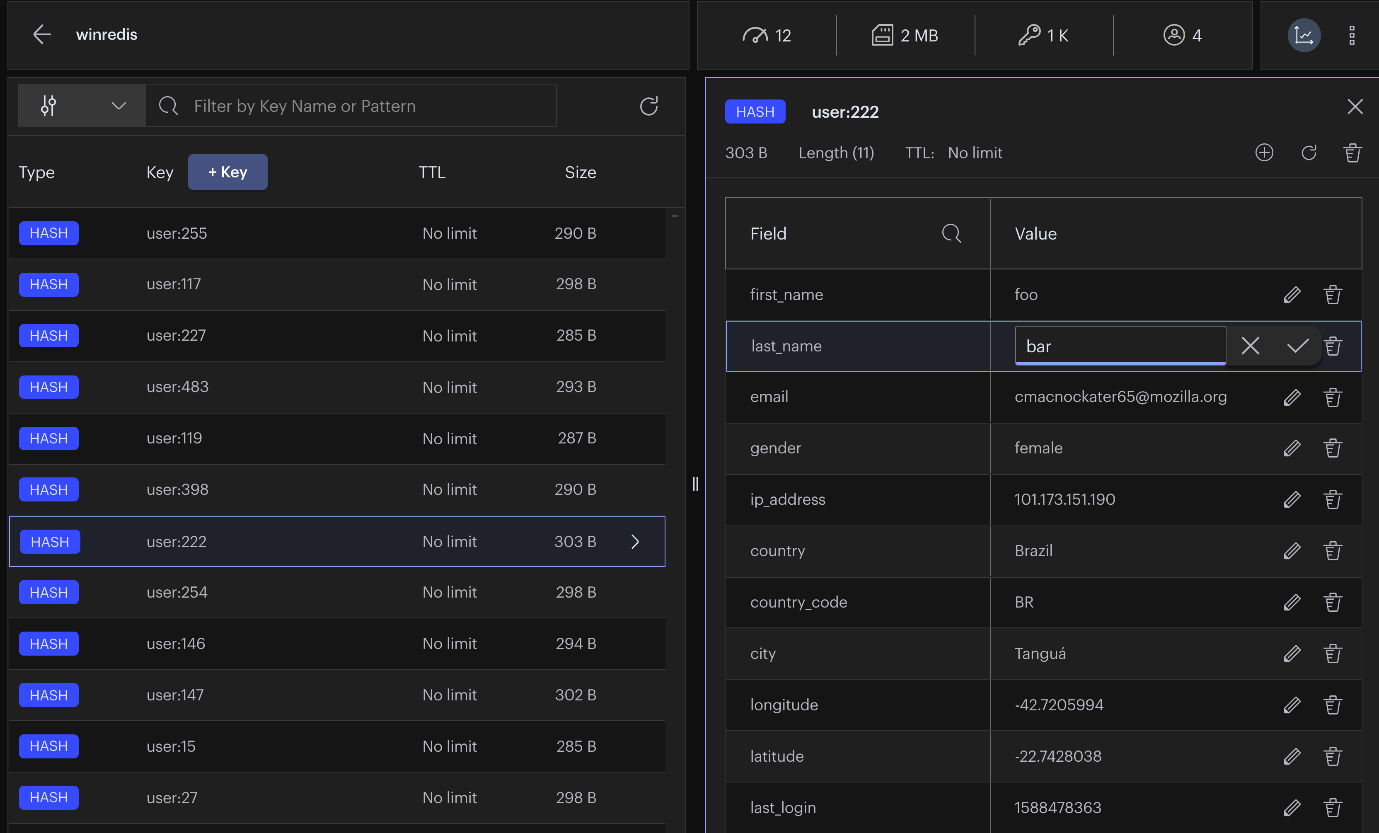
You can get a real-time view of the data in your Redis database as shown below:

Select any key in the keys view and the key's value gets displayed in the right hand side that includes fields and values.



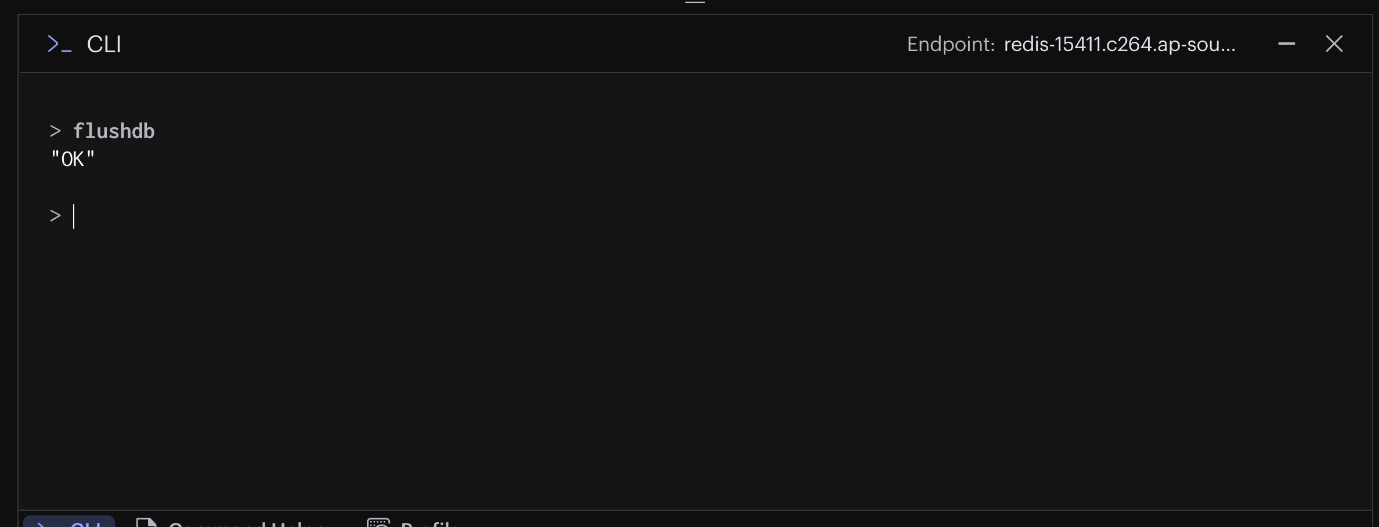
**Step 8. Modify a key**

The Redis Insight browser tool allows you to modify the data instantly. Select any key and change the values as shown in the following screenshot



**Step 9. Cleaning up**

Run the following command to clean up all the Redis keys:



**Redis Cheat Sheet**

**The basic syntax of Redis client.**

$redis-cli

**Run Commands on the Remote Server**

You need to connect to the server by the same client **redis-cli**

$ redis-cli -h host -p port -a password

**Redis: Keys Cheat Sheet**

**Del Command**

redis 127.0.0.1:6379> DEL KEY\_NAME

**Dump Command**

redis 127.0.0.1:6379> DUMP KEY\_NAME

**Exists Command**

redis 127.0.0.1:6379> EXISTS KEY\_NAME

**Expire Command**

redis 127.0.0.1:6379> Expire KEY\_NAME TIME\_IN\_SECONDS

**Expireat Command**

redis 127.0.0.1:6379> Expireat KEY\_NAME TIME\_IN\_UNIX\_TIMESTAMP

**Pexpire Command**

redis 127.0.0.1:6379> PEXPIRE KEY\_NAME TIME\_IN\_MILLISECONDS

**Pexpireat Command**

redis 127.0.0.1:6379> PEXPIREAT KEY\_NAME TIME\_IN\_MILLISECONDS\_IN\_UNIX\_TIMESTAMP

**Keys Command**

redis 127.0.0.1:6379> KEYS PATTERN

**Move Command**

redis 127.0.0.1:6379> MOVE KEY\_NAME DESTINATION\_DATABASE

**Persist Command**

redis 127.0.0.1:6379> PERSIST KEY\_NAME

**PTTL Command**

redis 127.0.0.1:6379> SET tutorialname redis OK

**TTL Command**

redis 127.0.0.1:6379> TTL KEY\_NAME

**Random key Command**

redis 127.0.0.1:6379> RANDOMKEY

**Rename Command**

redis 127.0.0.1:6379> RENAME OLD\_KEY\_NAME NEW\_KEY\_NAME

**Renamenx Command**

redis 127.0.0.1:6379> RENAMENX OLD\_KEY\_NAME NEW\_KEY\_NAME

**Type Command**

redis 127.0.0.1:6379> TYPE KEY\_NAME

[**Redis: Strings Cheat Sheet**](https://simplecheatsheet.com/redis-strings/)

**Set Command**

redis 127.0.0.1:6379> SET KEY\_NAME VALUE

**Get Command**

redis 127.0.0.1:6379> GET KEY\_NAME

**Getrange Command**

redis 127.0.0.1:6379> GETRANGE KEY\_NAME start end

**Getset Command**

redis 127.0.0.1:6379> GETSET KEY\_NAME VALUE

**Getbit Command**

redis 127.0.0.1:6379> GETBIT KEY\_NAME OFFSET

**Mget Command**

redis 127.0.0.1:6379> MGET KEY1 KEY2 .. KEYN

**Setbit Command**

redis 127.0.0.1:6379> SETBIT KEY\_NAME OFFSET

**Setex Command**

redis 127.0.0.1:6379> SETEX KEY\_NAME TIMEOUT VALUE

**Setnx Command**

redis 127.0.0.1:6379> SETNX KEY\_NAME VALUE

**Setrange Command**

redis 127.0.0.1:6379> SETRANGE KEY\_NAME OFFSET VALUE

**Strlen Command**

redis 127.0.0.1:6379> STRLEN KEY\_NAME

**Mset Command**

redis 127.0.0.1:6379> MSET key1 value1 key2 value2 .. keyN valueN

**Msetnx Command**

redis 127.0.0.1:6379> MSETNX key1 value1 key2 value2 .. keyN valueN

**Psetex Command**

redis 127.0.0.1:6379> PSETEX key1 EXPIRY\_IN\_MILLISECONDS value1

**Incr Command**

redis 127.0.0.1:6379> INCR KEY\_NAME

**Incrby Command**

redis 127.0.0.1:6379> INCRBY KEY\_NAME INCR\_AMOUNT

**Incrbyfloat Command**

redis 127.0.0.1:6379> INCRBYFLOAT KEY\_NAME INCR\_AMOUNT

**Decr Command**

redis 127.0.0.1:6379> DECR KEY\_NAME

**Decrby Command**

redis 127.0.0.1:6379> DECRBY KEY\_NAME DECREMENT\_AMOUNT

**Append Command**

redis 127.0.0.1:6379> APPEND KEY\_NAME NEW\_VALUE

[**Redis: Hashes Cheat Sheet**](https://simplecheatsheet.com/redis-hashes/)

**Hdel Command**

redis 127.0.0.1:6379> HDEL KEY\_NAME FIELD1.. FIELDN

**Hexists Command**

redis 127.0.0.1:6379> HEXISTS KEY\_NAME FIELD\_NAME

**Hget Command**

redis 127.0.0.1:6379> HGET KEY\_NAME FIELD\_NAME

**Hgetall Command**

redis 127.0.0.1:6379> HGETALL KEY\_NAME

**Hincrby Command**

redis 127.0.0.1:6379> HINCRBY KEY\_NAME FIELD\_NAME INCR\_BY\_NUMBER

**Hincrbyfloat Command**

redis 127.0.0.1:6379> HINCRBYFLOAT KEY\_NAME FIELD\_NAME INCR\_BY\_NUMBER

**Hkeys Command**

redis 127.0.0.1:6379> HKEYS KEY\_NAME FIELD\_NAME INCR\_BY\_NUMBER

**Hlen Command**

redis 127.0.0.1:6379> HLEN KEY\_NAME

**Hmget Command**

redis 127.0.0.1:6379> HMGET KEY\_NAME FIELD1...FIELDN

**Hset Command**

redis 127.0.0.1:6379> HSET KEY\_NAME FIELD VALUE

**Hsetnx Command**

redis 127.0.0.1:6379> HSETNX KEY\_NAME FIELD VALUE

**Hvals Command**

redis 127.0.0.1:6379> HVALS KEY\_NAME FIELD VALUE

**Redis with Python**

1. **Installation**

To install Redis in python

**>> pip install redis**

1. **Connection to Redis in Python**

>> redisConnectionPool = redis.ConnectionPool(host='localhost', port=6379, db=0)

>> redisConn = redis.Redis(connection\_pool=redisConnectionPool)

1. **Connection to MySQL Server in Python**

>>sqlConn = connection.connect(host="localhost", database = "beneficiary\_db", user="root", passwd="")

1. **Reading Local MySQL Data**

>> query = "Select \* from beneficiaries"

>> df = sql.read\_sql(query, sqlConn)

1. **MySQL to Redis**

>>def MySQLtoRedis(keyValue, dataframe):

df = pa.serialize(dataframe).to\_buffer().to\_pybytes()

result = redisConn.set(keyValue,df)

if result == True:

print('Upload Done')

>>MySQLtoRedis('this\_is\_key', df)

1. **Redis to MySQL**

>>def RedistoMySQL(keyValue):

data = redisConn.get(keyValue)

try:

return pa.deserialize(data)

except:

print("No data")

>>LoadfromRedis('this\_is\_key')