Redis Client And Server Set Up (Data Base):

Redis is an open source, advanced **key-value store**. It is often referred to as a **data structure server** since keys can contain strings, hashes,lists, sets and sorted sets.
You can run **atomic operations** on these types, like appending to a string; incrementing the value in a hash; pushing to a list; computing set intersection, union and difference; or getting the member with highest ranking in a sorted set.
In order to achieve its outstanding performance, Redis works with an **in-memory dataset**. Depending on your use case, you can persist it either by dumping the dataset to disk every once in a while, or by appending each command to a log.
Redis also supports trivial-to-setup master-slave replication, with very fast non-blocking first synchronization, auto-reconnection on net split and so forth.
Other features include a simple check-and-set mechanism, pub/sub and configuration settings to make Redis behave like a cache.
You can use Redis from most programming languages out there.
Redis is written in **ANSI C** and works in most POSIX systems like Linux, *BSD, OS X without external dependencies. Linux and OSX are the two operating systems where Redis is developed and more tested, and we **recommend using Linux for deploying**. Redis may work in Solaris-derived systems like SmartOS, but the support is *best effort*. There is no official support for Windows builds, although you may havesome options.

Project History

Redis was originally written in order to help address the scaling needs of http://lloogg.com/, a web analytics startup, by Salvatore (antirez) Sanfilippo. Following its open source debut in March of 2009, Redis quickly gained popularity due to its unique feature set and outstanding performance. VMware now funds ongoing development of the product.

Redis and Memcached Major Feature Comparison

	Redis	Memcached
In-memory	x	x
Virtual-memory	x	
Persist data to disk	x	
Dataset replication	x	
Authentication	x	x
Strong Authentication		x
Simple key-values	x	x
Key enumeration	x	
Data structures	x	
Channel pub/sub	x	
Atomic operations	x	x

Features Unique to Redis

Virtual memory

Explained:

Virtual memory is a method of supplementing the storage available in RAM by mapping additional virtual space to disk based storage.

Example Uses:

- If your working dataset is simply larger than the amount of RAM available to store that dataset.
- If data access is biased to a relatively small set of keys, yet the keys are large enough that they will all not fit into memory simultaneously.

Persist data to disk

Explained:

In memory data may be saved to disk at configurable intervals. These intervals are defined by the number of key changes per second, allowing for a great deal of flexibility if disk performance is a concern.

Example Uses:

- If your application requires a persistent database like data store
- If your application would place too much load on the backend infrastructure should an event occur that would expunge the contents of the in-memory store. I.E. your application requires that its cache be warm following cache system restart.

Replication

Explained:

Redis supports simple master to slave replication. When a relationship is established all data from the master is transferred to the slave. After this is complete all changes to the master replicate to the slave.

Example Uses:

- If your application requires redundancy in the event of hardware failure relating to the primary cache system.
- If you require an off system backup of the data store it is a best practice to perform a backup from the slave system

Key enumeration

Explained:

Most key value stores do not allow you to enumerate the names of the stored keys. In order to access a key you must know the key name. Redis allows you to enumerate the names of the stored keys via a KEYS command and a matching construct.

Example Uses:

- If your application requires that you are able to list and/or act upon a list of present keys.

Data structures

Explained:

Redis supports more data types than simple strings. This allows complex, heavily used data structures to be utilized within the data store in order to take advantage of in-memory performance.

STRING: A simple string

LIST: A list of strings. Stacks and queues can be easily modeled utilizing lists.

SET: An unordered collection of unique items. Union, diff, and intersection operations can be performed on sets. Sets can be utilized to build highly optimized indexes.

SORTED SET: Similar to sets, but each member of the set has an associated floating point score. Sorted sets are essential whenever a range type query is needed. Sorted sets are also an important tool for constructing indexes.

HASH: Redis hashes are similar to Ruby hashes, or Python dictionaries. They are ideal for storing serialized objects.

Channel publish and subscribe

Explained:

Redis allows clients to PUBLISH and SUBSCRIBE to channels. Multiple clients can subscribe to a channel. Another client can then publish to that channel and all the subscribed clients will receive the published value.

Example Uses:

- This functionality can allow you to create a simple high performance shared message bus for a distributed application.

Cloud Security Considerations

Redis was designed to address performance issues encountered with large distributed web applications as such it currently does not support strong authentication or transport layer encryption, as these features would degrade performance.

When using Redis in the cloud it is important that the redis traffic not traverse publicly accessible networks, and that the appropriate precautions are taken to ensure security to the port that Redis is listening on (6379 by default). These precautions would typically take the form of packet filter rule sets and binding Redis to non-publicly facing interfaces on cloud servers.

Who is using Redis?

A significant number of companies are utilizing Redis in order to help them address application-scaling issues. The following is a short list of some of the larger parties.

- Rackspace
- Blizzard Entertainment
- Digg
- Stackoverflow
- Craigslist
- Twitter
- Github

Setting up Enviornment :

The suggested way of installing Redis is compiling it from sources as Redis has no dependencies other than a working GCC compiler and libc. Installing it using the package manager of your Linux distribution is somewhat discouraged as usually the available version is not the latest.

You can either download the latest Redis tar ball from the redis.io web site, or you can alternatively use this special URL that always points to the latest stable Redis version, that is, http://download.redis.io/redis-stable.tar.gz.

In order to compile Redis follow this simple steps:

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make
```

At this point you can try if your build works correctly typing **make test**, but this is an optional step. After the compilation the **src** directory inside the Redis distribution is populated with the different executables that are part of Redis:

- **redis-server** is the Redis Server itself.
- **redis-cli** is the command line interface utility to talk with Redis.
- **redis-benchmark** is used to check Redis performances.
- **redis-check-aof** and **redis-check-dump** are useful in the rare event of corrupted data files.

It is a good idea to copy both the Redis server than the command line interface in proper places using the following commands:

- sudo cp redis-server /usr/local/bin/
- sudo cp redis-cli /usr/local/bin/

In the following documentation I assume that /usr/local/bin is in your PATH environment variable so you can execute both the binaries without specifying the full path.

Starting Redis:

The simplest way to start the Redis server is just executing the **redis-server** binary without any argument.

```
./ redis-server
[28550] 01 Aug 19:29:28 # Warning: no config file specified, using the default config. In order to specify a config file use 'redis-server /path/to/redis.conf'
[28550] 01 Aug 19:29:28 * Server started, Redis version 2.2.12
[28550] 01 Aug 19:29:28 * The server is now ready to accept connections on port 6379
... and so forth …
```

The first thing to do in order to check if Redis is working properly is sending a **PING** command using redis-cli:

```
./redis-cli ping
 pong
```

Running **redis-cli** followed by a command name and its arguments will send this command to the Redis instance running on localhost at port 6379. You can change the host and port used by redis-cli, just try the --help option to check the usage information.
Another interesting way to run redis-cli is without arguments: the program will start into an interactive mode where you can type different commands:

```
./redis-cli
redis 127.0.0.1:6379> ping
PONG
redis 127.0.0.1:6379> set mykey somevalue
OK
redis 127.0.0.1:6379> get mykey
"somevalue"
```

Killing Server :

```
./redis-shutdown
```

Glossary :

- http://yaychris.com/blog/2009/12/redis-part-2.html for more examples and details on Lists in redis.
- https://github.com/andymccurdy/redis-py, https://github.com/fiorix/txredisapi GitHub project on Redis-Python.