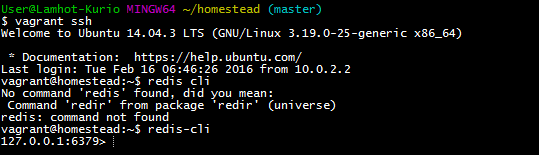
Note for Redis on Laravel

Redis-cli



Hire on

C:\Users\User\redis\vendor\laravel\framework\src\Illuminate\Redis

RedisServiceProvider.php

Change this code:

**public function** register()  
{  
 $this->**app**->singleton(**'redis'**, **function** ($app) {  
 *// return new Database($app['config']['database.redis']);  
 print\_r*($app[**'config'**][**'database.redis'**]);;  
 });  
}

you will now the result:

Result of redis connection:

Array ( [cluster] => [default] => Array ( [host] => 127.0.0.1 [port] => 6379 [database] => 0 ) )

Add this block of code on compser.json

**"predis/predis"**:**"~1.0"**

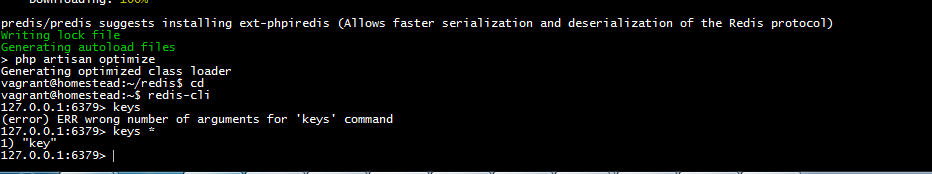
Will show like this :



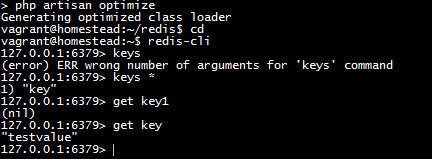
End then try to set something on redis using this command:

Route::*get*(**'/'**, **function** () {  
 $redis = app()->make(**'redis'**);  
 $redis->set(**"key"**, **"testvalue"**);  
 **return** $redis->get(**"key"**);  
});

Try to check on redis



Try to get key:



Sudo apt-get update

Follow this tutorial:

<https://scaleyourcode.com/blog/article/10>

flushdb:hapus database or keys on redis

redis>  LPUSH mylist "world"

(integer) 1

redis>  LPUSH mylist "hello"

(integer) 2

redis>  LRANGE mylist 0 -1

1) "hello"

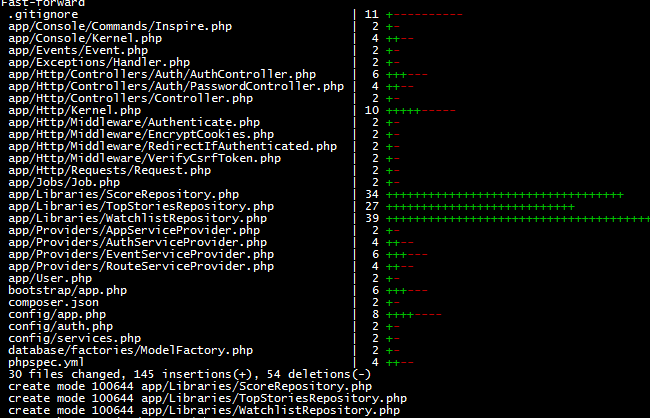
2) "world"

store data in a meaningful way and can make data retrieval faster. Here are some of the datatypes supported by Redis:

* **String:** Similar to Strings in PHP.
* **List:** Similar to a single dimensional array in PHP. You can push, pop, shift and unshift, the elements that are placed in order or insertion FIFO (first in, first out).
* **Hash:** Maps between string fields and string values. They are the perfect data type to represent objects (e.g.: A User with a number of fields like name, surname, and so forth).
* **Set:** Similar to list, except that it has no order and each element may appear only once.
* **Sorted Set:** Similar to Redis Sets with a unique feature of values stored in set. The difference is that each member of a Sorted Set is associated with score, used to order the set from the smallest score to the largest.

Others are bitmaps and hyperloglogs, but they will not be discussed in this article, as they are pretty dense.

List of change



|  |
| --- |
| "psr-4": { |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |
| - "App\\": "app/" |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |
| + "Kurio\\Toronto\\": "app/" |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |
| } |
|  |

|  |
| --- |
|  |

|  |
| --- |
| }, |

Implementt suatu interface

WatchlistRepo

add($watchlist\_key, $url);

{

Watchlist key=topic of artice, 40 group berdasarkan topik

URL:url

}

fetch($watchlist\_key);//get urls from topic

{

}

setUrlTTL($minutes = 180);

{

Every create on e push will be set expired date of each puch to 3 hours

}

remove();//remove expired

remove watchlist

Topstories:

push($axis\_type, $axis\_id, $url);// giving back url top stories

fetch($axis\_type, $axis\_id, $num);// get total set per axis of 70

Score Repository:

push($url, $score);// real time to push value to array of score per url

{

S=[

url: “http://www.someting.com”

Score

{

s1=>1,

s2=>1,

s3=>1,

s4=>1,

s5=>1,

s6=>1

}

Return s;

}

getAverage($url);

{

Get delta

Max= $this->push()

}

remove($url)

{

If score value not suitable remove

}