GROWEXX

Introduction to Redis with PHP

Prepared By:- Rupesh Mishra
Technical Lead

TECH
POWERS
GROWTH

Table of Contents

What is Redis?

Definition: Redis is an in-memory key-value store, often used as a cache and message broker.

Key Features:

- Extremely fast read/write operations
- Supports various data types: strings, lists, sets, hashes, sorted sets, etc.
- Persistent storage options (RDB, AOF)
- Pub/Sub messaging model

Why Use Redis?

Performance Boost:
Redis can significantly improve performance for read-heavy applications.

Caching:
Store frequently accessed data like user sessions, API responses, and database queries.

Pub/Sub:
Enable real-time message passing between systems.

Scalability:
Redis is highly scalable, making it suitable for large-scale applications.

Installing Redis on PHP

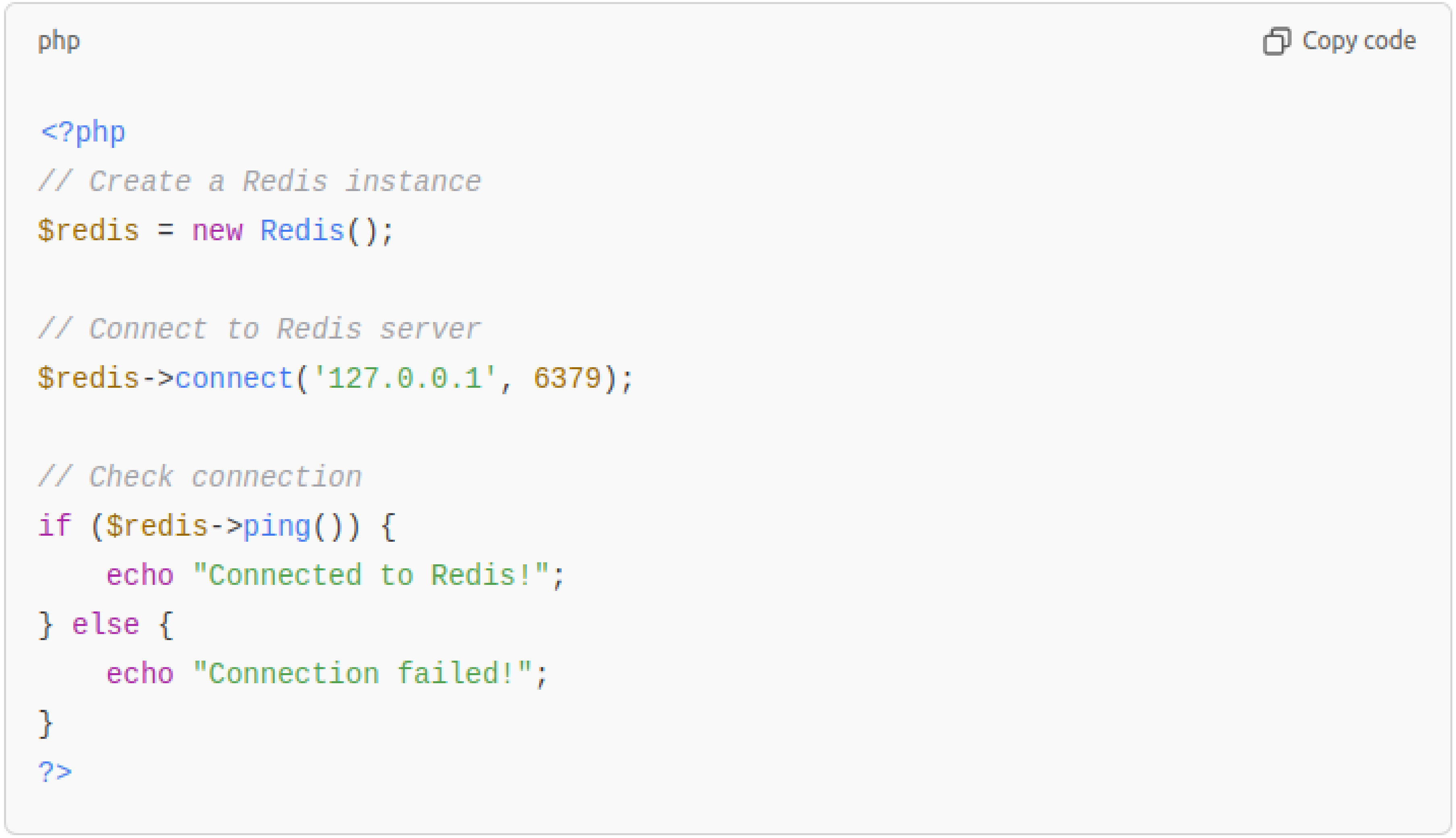
1. Install Redis Server:

- Linux:
  sudo apt-get install redis-server
  or
  brew install redis (on macOS)
- Windows:
  Download Redis for Windows from Microsoft's GitHub repository.

2. Install PHP Redis Extension:

- Using PECL (preferred):
  pecl install redis
- Enable the extension:
  Add extension=redis.so (for Linux/macOS) or extension=php_redis.dll (for Windows) in php.ini.

Connecting PHP to Redis



```php
<?php
// Create a Redis instance
$redis = new Redis();

// Connect to Redis server
$redis->connect('127.0.0.1', 6379);

// Check connection
if ($redis->ping()) {
    echo "Connected to Redis!";
} else {
    echo "Connection failed!";
}
?>
```

Common Redis Commands in PHP

- **Set a Value:**

```
$redis->set('key', 'value');
```

- **Get a Value:**
-
```
$value = $redis->get('key');
echo $value;
```

- **Check if Key Exists:**

```
$exists = $redis->exists('key');
```

- **Delete a Key:**

```
$redis->del('key');
```

- **Increment a Value:**

```
$redis->incr('counter');
```

Using Redis for Caching

Cache API Response:

```
$cache_key = 'api_response';

// Check if response is already cached
$cached_response = $redis->get($cache_key);
if ($cached_response) {
    echo "Cache hit: " . $cached_response;
} else {
    // Fetch from API and cache it
    $api_response = file_get_contents('https://api.example.com/data');
    $redis->setex($cache_key, 3600, $api_response); // Cache for 1 hour
    echo "Cache miss: " . $api_response;
}
```

Advanced Redis Operations

Hash Data Structures:

```
$redis->hSet('user:1', 'name', 'John');
$redis->hGet('user:1', 'name');
```

Lists (Queues):

```
$redis->lPush('queue', 'task1');
 $redis->rPop('queue');
```

Pub/Sub (Publish and Subscribe):

```
// Publisher
$redis->publish('channel', 'Hello World!');

 // Subscriber
 $redis->subscribe(['channel'], function($message) {
       echo "Received: $message";
   });
```

Error Handling & Best Practices

Error Handling:

```
try {
    $redis->connect('127.0.0.1', 6379);
} catch (RedisException $e) {
    echo "Redis connection failed: " . $e->getMessage();
}
```

Best Practices:

- Use Redis for frequently accessed data.
- Avoid overusing Redis for large datasets or complex operations.
- Use connection pooling for production environments.
- Use Redis' expiration feature to avoid stale cache.

Conclusion

- Redis is an efficient tool to improve performance, cache data, and support real-time messaging.
- Integrating Redis with PHP is straightforward and offers various benefits like reduced database load, faster responses, and better scalability.

Thank You