

Installing and Updating an Extension

Redis supports **modules** to extend core functionality with new data types, commands, or algorithms. These modules behave like plugins in other systems and are loaded at server startup. Examples include [RedisBloom](#), [RedisTimeSeries](#), [RedisJSON](#), and [RedisSearch](#).

In Elestio-hosted Redis instances or any Docker Compose-based setup, modules can be loaded by specifying them in the service configuration. This guide walks through how to install, load, and manage Redis modules using Docker Compose, along with common issues and best practices.

Installing and Enabling Redis Modules

Redis modules are typically compiled as shared object (.so) files and must be loaded at server startup using the `--loadmodule` option. These module files are mounted into the container and referenced from within the container's file system. To use a module like RedisBloom in a Docker Compose setup:

Update docker-compose.yml

Mount the module file into the container and load it:

```
services:
  redis:
    image: redis/redis-stack-server:latest
    volumes:
      - ./modules/redisbloom.so:/data/redisbloom.so
    command: ["redis-server", "--loadmodule", "/data/redisbloom.so"]
    ports:
      - "6379:6379"
```

Here:

- `./modules/redisbloom.so` is the local path on your host machine.
- `/data/redisbloom.so` is the path inside the container.

Make sure the .so file exists in the specified directory before running Docker Compose.

Restart the Redis Service

After updating the Compose file, restart the service:

```
docker-compose down  
docker-compose up -d
```

This will reload Redis with the specified module.

Verify the Module is Loaded

Once Redis is running, connect to it using redis-cli:

```
docker-compose exec redis redis-cli -a <yourPassword>
```

Run the following command:

```
MODULE LIST
```

Expected output:

```
1) 1) "name"  
   2) "bf"  
   3) "ver"  
   4) (integer) 20207
```

This confirms the module (in this case, bf for RedisBloom) is loaded and active.

Checking Module Availability & Compatibility

Redis modules must match the Redis server version and platform. You can verify compatibility through the module's documentation or by testing it in a local development setup before using it in production.

To inspect module-related details:

```
INFO MODULES
```

To verify the correct Redis image is being used:

```
docker-compose exec redis redis-server --version
```

If a module fails to load, check the container logs:

```
docker-compose logs redis
```

This often reveals missing paths or compatibility issues.

Updating or Unloading Modules

Unlike MySQL, Redis does **not** support dynamic unloading of modules once loaded. To update or remove a module:

1. **Stop the container:**

```
docker-compose down
```

2. **Edit docker-compose.yml:**

- Change the .so file path if updating the module.
- Remove the `--loadmodule` line if unloading the module.

3. **Restart the container:**

```
docker-compose up -d
```

Always test updated modules in staging before applying to production.

Troubleshooting Common Module Issues

| Issue | Cause | Resolution |
|-------------------------------|--|--|
| Redis fails to start | Incorrect module path or incompatible binary | Check docker-compose logs redis and verify the .so path and architecture |
| MODULE command not recognized | Using a Redis image without module support | Use an image like redis/redis-stack-server which supports modules |

| Issue | Cause | Resolution |
|---|--|--|
| "Can't open .so file" | Volume not mounted or permission denied | Ensure the .so file exists locally and is readable by Docker |
| Module not appearing in MODULE LIST | Module failed to load silently | Double-check command and container logs |
| Commands from the module not recognized | Module not loaded properly or incompatible | Validate Redis version and module compatibility |

Security Considerations

Redis modules execute native code with the same privileges as Redis itself. Only load trusted, vetted modules from official sources. Avoid uploading or executing arbitrary .so files from unknown authors. In multi-tenant or exposed environments, module misuse could lead to instability or security risks. Ensure the redis user inside the container has limited privileges, and module directories have appropriate permissions.