

# Installing and Updating an Extension

KeyDB supports Redis-compatible modules to extend core database functionality with custom data types, specialized algorithms, and advanced operations. These modules are compiled as shared object (.so) files and must be loaded at server startup. Examples include RedisBloom, RedisJSON, and RedisTimeSeries all of which are supported in KeyDB just as in Redis.

In Elestio-hosted KeyDB instances or any Docker Compose-based setup, modules can be mounted and loaded via configuration in `docker-compose.yml`. This guide outlines how to install, load, and manage KeyDB modules using Docker Compose, including verification steps, update methods, and best practices.

## Installing and Enabling KeyDB Modules

Modules in KeyDB must be loaded at server startup using the `--loadmodule` directive. These are .so binaries that are typically mounted into the container from the host file system. The process is nearly identical to Redis module integration.

### Update `docker-compose.yml`

To use a module such as RedisBloom in a KeyDB Docker setup, mount the module file and add the `--loadmodule` directive to the container command.

```
services:
  keydb:
    image: eqalpha/keydb:latest
    volumes:
      - ./modules/redisbloom.so:/data/redisbloom.so
    command: ["keydb-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 where the module will be accessible inside the container.

Ensure that the .so file exists locally before running the container.

## Restart the KeyDB Service

After updating the Docker Compose configuration, apply changes by restarting the container:

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

This reloads KeyDB and ensures the module is initialized during startup.

## Verify the Module is Loaded

Once KeyDB is running, connect to the containerized service:

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

Run the following command to check for loaded modules:

```
MODULE LIST
```

Expected output (for RedisBloom):

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

This confirms that the module (in this case, bf for Bloom filters) has been loaded successfully.

# Checking Module Availability & Compatibility

KeyDB modules must match the container's runtime architecture and the KeyDB version. Many Redis modules work out-of-the-box with KeyDB, but always check the official documentation or test in a controlled environment first.

To inspect module metadata and compatibility:

```
INFO MODULES
```

To confirm the current KeyDB version and platform:

```
docker-compose exec keydb keydb-server --version
```

If a module fails to load, check container logs for detailed error output:

```
docker-compose logs keydb
```

Most load failures are caused by missing binaries, unsupported formats, or incorrect file paths.

## Updating or Unloading Modules

KeyDB does not support dynamic unloading of modules while the server is running. To update or remove a module, the server must be stopped and restarted with the revised configuration.

Stop the container:

```
docker-compose down
```

Edit docker-compose.yml as needed:

- Update the .so path to reference the new module version.
- Remove the --loadmodule line to disable the module entirely.

Start the container again:

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

Always test updated modules in staging before deploying to production environments.

## Troubleshooting Common Module Issues

Issue	Cause	Resolution
KeyDB fails to start	Invalid module path or incompatible binary	Check docker-compose logs keydb and verify path and architecture
MODULE command not recognized	Image does not include module support	Use an image like eqalpha/keydb or eqalpha/keydb:alpine

Issue	Cause	Resolution
"Can't open .so file" error	Volume not mounted or file permission denied	Confirm that the .so file exists and has readable permissions
Module not listed in MODULE LIST	Silent module load failure	Review container logs and validate command syntax
Module commands not recognized	Module did not load correctly	Ensure Redis version and module binary compatibility

# Security Considerations

Modules execute native code within the KeyDB process and inherit its permissions. As such, only load trusted .so files compiled from official or reviewed source code. Avoid uploading or using third-party binaries without auditing. In Elestio-managed or containerized environments, use Docker's file and user isolation to reduce risk:

- Set read-only permissions on mounted .so files.
- Use non-root users inside containers when possible.
- Monitor module behavior with SLOWLOG, INFO, and command auditing.

Improperly configured or malicious modules can cause crashes, memory leaks, or worse. Treat modules as privileged extensions and keep them versioned and tested across environments.

---

Revision #1

Created 2025-06-25 07:13:08 UTC by kaiwalya

Updated 2025-06-25 08:29:36 UTC by kaiwalya