Creating a Database

KeyDB is a high-performance fork of Redis that offers multithreading, active-active replication, and enhanced memory management. Setting up KeyDB correctly is essential for achieving low-latency performance and ensuring durability in modern applications. This guide walks through various methods to run and connect to KeyDB: using the KeyDB CLI, running inside Docker containers, and integrating with scripting workflows. It also outlines best practices to follow during configuration and operation.

Creating Using keydb-cli

KeyDB provides a built-in command-line interface tool called keydb-cli. It allows direct interaction with a KeyDB server and supports both local and remote connections. All standard Redis-compatible commands can be executed through this tool, along with extended functionality supported by KeyDB.

Connect to KeyDB:

If you have a local KeyDB instance running, either from a package manager or inside Docker, you can start the CLI with no extra arguments:

```
keydb-cli
```

To connect to a remote KeyDB instance, provide the host, port, and authentication details if configured:

```
keydb-cli -h <host> -p <port> -a <password>
```

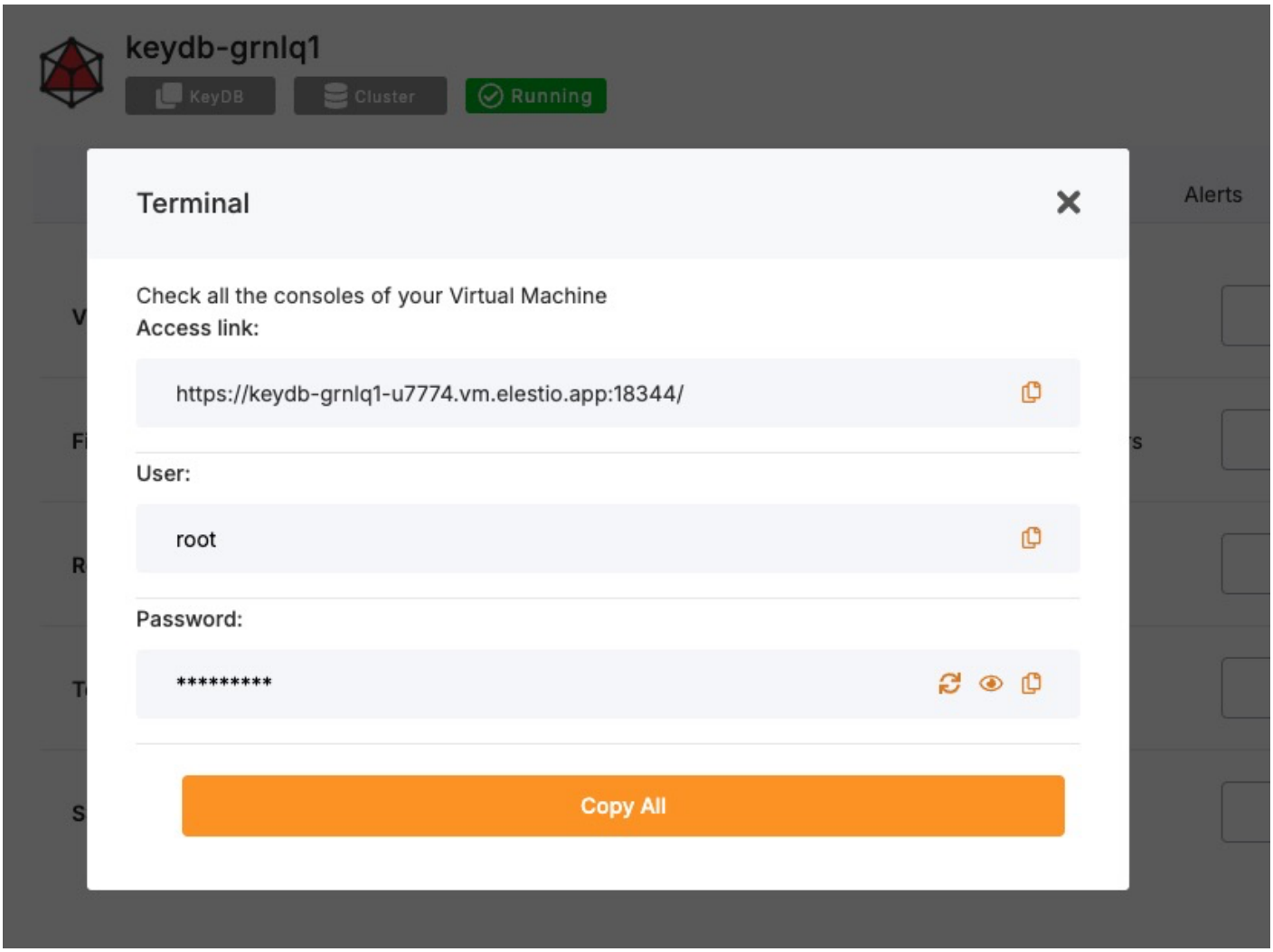
After executing the command, you will be placed in the KeyDB shell, where you can interactively issue commands.

Running KeyDB Using Docker

KeyDB can be containerized using Docker to ensure consistent environments across local development, testing, and production systems. This is a convenient way to isolate dependencies and manage deployment configurations.

Access Elestio Terminal

If you are using Elestio to host your KeyDB service, log in to the Elestio dashboard. Navigate to your KeyDB instance, then open **Tools > Terminal**. This will provide a browser-based shell within the server environment that has access to your containerized services.



Once inside the terminal, switch to the application directory:

```
cd /opt/app/
```

Access the KeyDB Container Shell

Elestio services use Docker Compose for container orchestration. To enter the KeyDB container and interact with its runtime environment, use the following command:

```
docker-compose exec keydb bash
```

This starts a bash session inside the running KeyDB container.

Access KeyDB CLI from Within the Container

The keydb-cli tool is available within the container and can be used to run commands directly against the KeyDB server. If authentication is required, supply the password using the -a flag:

```
keydb-cli -a <password>
```

You'll now be connected to the KeyDB instance running inside the container.

Test Connectivity

To confirm the KeyDB instance is functional, run a test by setting and retrieving a key:

```
set testkey "Hello KeyDB"
get testkey
```

Expected output:

```
"Hello KeyDB"
```

This confirms that read/write operations are working correctly inside the containerized KeyDB environment.

Connecting Using keydb-cli in Scripts

The keydb-cli command can also be used non-interactively, which is useful for shell scripts, cron jobs, or CI/CD workflows that require interaction with the KeyDB server.

To set a key via a script:

```
keydb-cli -h <host> -p <port> -a <password> SET example_key "example_value"
```

This will set the specified key in a single command without launching the interactive shell.

Best Practices for Setting Up KeyDB

Use Meaningful Key Naming Conventions

To ensure readability and manageability, adopt consistent naming conventions. Use namespaces separated by colons to logically group related keys:

```
user:1001:profile
session:2025:token
```

This simplifies debugging, metric tracking, and migration efforts.

Follow Consistent Data Structures

KeyDB supports Redis-compatible data structures including strings, hashes, sets, sorted sets, lists, and streams. Always choose the most efficient type based on access patterns and data lifecycle. For example, hashes are ideal for storing grouped attributes, while sets work well for unique lists.

Inconsistent structure usage can lead to inefficient memory use and unexpected command behavior.

Enable Authentication and TLS

Security should not be overlooked in production systems. Always configure a strong password using the requirepass directive in keydb.conf. Additionally, enable TLS for encrypted traffic if the database is accessible over the internet or across networks.

Example keydb.conf settings:

```
requirepass strong_secure_password
tls-port 6379
tls-cert-file /etc/ssl/certs/cert.pem
tls-key-file /etc/ssl/private/key.pem
```

These settings help secure both access and data transmission.

Configure Persistence Options

KeyDB supports both Redis-style persistence mechanisms: RDB snapshots and AOF logging. These ensure data durability in the event of process restarts or hardware failure.

Recommended settings in keydb.conf:

```
save 900 1
appendonly yes
appendfsync everysec
```

Use AOF for greater durability, RDB for faster restarts, or both for a balanced setup.

Monitor and Tune Performance

Monitor performance using built-in KeyDB commands like INFO, MONITOR, and SLOWLOG. These provide insights into memory usage, command execution times, and system health. You can also integrate external monitoring tools like Prometheus, RedisInsight, or Grafana for real-time visualization.

Proper monitoring allows you to proactively tune memory limits, max clients, and replication settings.

Common Issues and Their Solutions

Issue	Cause	Solution
NOAUTH Authentication required	Connecting to an instance that requires a password without supplying one	Use the -a flag or send the AUTH command before other commands
ERR Client sent AUTH, but no password is set	Authentication is attempted on a server that does not require it	Remove the -a option or check the requirepass directive
Cannot connect to KeyDB on ‘localhost’	The server is not running or bound to another address/port	Check service status and inspect keydb.conf and Docker port mappings
Docker KeyDB container refuses connections	Network misconfiguration or the container is still initializing	Use docker-compose logs keydb and verify exposed ports
Data not persisted after restart	Persistence settings are disabled	Enable RDB and/or AOF in the configuration file

Revision #1
Created 2025-06-25 06:57:03 UTC
Updated 2025-06-25 07:03:52 UTC