Passwords, users, and roles

All Redis Cloud databases require either password-based authentication or role-based access control. Role-based access control allows you to define multiple users with fine-grained authorization features.

Prerequisites

To use role-based access control, your Redis cloud database needs to support Redis version 6.0.0 or later.

The Redis version of a database is displayed in the General section of the Configuration tab of the database detail screen.

Password-based authentication

Password-based authentication is a basic but essential Redis security feature. When you create a Redis Cloud database, your database is given a randomly generated password called the Default user password.

This appears in the Security section of the Configuration tab of the database details screen.
You can see this password on the View Database screen.

Click the icon to see your password and copy it.

You'll need to use this password whenever you connect to your database using a Redis client. For example, in the Redis CLI, you use the AUTH command to provide this password:

```
AUTH 4kTtH2ddXfN2sFmXE6sowOLukx1aJhN8n
```

See your Redis client's documentation for specifics on how to provide your password when connecting.

**Changing the password**

To change your Redis database password:

1. From the View Database screen, click

   ![Edit button](image)

   **View Database**

2. Scroll down to Access Control & Security, and enter the new password:
Role-Based access control

Role-based access control (RBAC) is an access-control mechanism that allows you to define roles with specific sets of permissions. You can then assign users to these roles to provide appropriate levels of access.

RBAC effectively lets you implement the principle of least privilege. For example, you can provide read-only access to an application whose only job is to display Redis data. Similarly, you can prevent new developers from running dangerous administrative commands.

Setting up RBAC

To set up RBAC, first navigate to the Data Access Control screen.

There are three tabs on this screen: Users, Roles, and Redis ACLs.

In the Redis ACLs tab, you define named permissions for specific Redis commands and keys.

In the Roles tab, you create roles. Each role consists of a set of permissions for one or more Redis Cloud databases.

Finally, in the Users tab, you create users, and you assign each user a role.

OSS Redis ACLs vs. Redis Enterprise Cloud RBAC

In open source Redis, you can create users and assign ACLs to them using the ACL command. However, open source Redis does not support generic roles.

In Redis Enterprise Cloud, you configure RBAC using the admin console. As a result, certain open source Redis ACL subcommands are not available in Redis Cloud.

Specifically, Redis Cloud databases block the following ACL subcommands: LOAD, SAVE, SETUSER, DELUSER, GENPASS, and LOG.
Redis Cloud databases allow these ACL subcommands: LIST, USERS, GETUSER, CAT, WHOAMI, and HELP.

In open source Redis, you must explicitly provide access to the MULTI, EXEC, and DISCARD commands. In Redis Cloud, these commands, which are used in transactions, are always permitted. However, the commands run within the transaction block are subject to RBAC permissions.

When you run multi-key commands on multi-slot keys, the return value is failure but the command runs on the keys that are allowed.

### Defining permissions

To define permissions, go to the Redis ACLs tab of the Data Access Control page.

### Data Access Control

<table>
<thead>
<tr>
<th>Name</th>
<th>ACL Rule</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-Access</td>
<td>+@all-*</td>
<td></td>
</tr>
<tr>
<td>Read-Only</td>
<td>+@read-*</td>
<td></td>
</tr>
<tr>
<td>Read-Write</td>
<td>+@all -@dangerous-*</td>
<td></td>
</tr>
<tr>
<td>Read-Only Cache</td>
<td>+@read -cache:*</td>
<td></td>
</tr>
</tbody>
</table>

You define these named permissions using the Redis ACL syntax. This syntax allows you to concisely specify commands, command categories, and keys that should be permitted.

A **command** can be any Redis command. Take the SET command, for example. The Redis ACL rule

```plaintext
+set
```

indicates that the SET command is permitted.

A **command category** is a predefined, named set of commands. For example, the Redis commands that read data are available in the **read** command category. The Redis ACL rule

```plaintext
+@read
```

permits access to all read commands. If you’d like to know which commands are included in the **read** command category, run

```
ACL CAT read
```

from the Redis CLI.

There’s also a syntax for specifying which **keys** can be accessed. For example,

```plaintext
~*
```

indicates that all keys can be accessed. Whereas
indicates that only those keys beginning with the prefix `cache:` can be accessed.

The Redis ACL syntax emphasizes brevity:

- `+` *includes* commands or command categories
- `-` *excludes* commands or command categories
- `@` indicates a command category
- `~` defines a permitted key pattern

**Predefined permissions**

Redis Cloud includes three, predefined permissions:

- Full Access (`+@all ~*`) - All commands are allowed for all keys
- Not Dangerous (`+@all -@dangerous ~*`) - All commands except for the "dangerous" command category are allowed for all keys
- Read Only (`+@read ~*`) - Only the "read" command category is allowed for all keys

**Module command permissions**

Note that you can define permissions for the Redis module commands of any modules that are loaded on the subscription; however, these permissions can only be used for databases that support those modules.

To define database access control, you can either:

- Use the predefined data access roles and add Redis ACLs to them for specific databases.
- Create new data access roles and select the management roles and Redis ACLs that apply to the roles for specific databases.
- Assign roles and Redis ACLs to a database in the access control list section of the database configuration.

**Configuring permissions using Redis ACLs**

To configure a Redis ACL that you can assign to a data access role:

1. In **Data Access Control > Redis ACLs**:
   - To edit an existing Redis ACL: Hover over a Redis ACL and click 🖍️
   - To create a new Redis ACL: Click 💎

2. Enter a descriptive name for the Redis ACL.

3. Enter the **ACL syntax** defining the ACL or click **Rule Builder** to use a form to build the ACL:
   1. For the commands:
      1. Select whether to include or exclude a command or category.
      2. Enter the ACL syntax that defines the commands.
         - To add more command definitions, click 💎
         - All entries in the Commands/Categories column apply to the keys defined in the Keys column.
   2. For the keys, enter the ACL syntax that defines the keys.
To add more key definitions, click +.

3. Click Save Rule.

4. Click Save.

Assigning permissions to a role

To assign Redis ACLs to a data access role:

1. In Data Access Control > Roles:
   - Edit an existing role - Hover over a role and click 
   - Create a new role - Click 

2. In the Redis ACLs section:
   - Edit a Redis ACL association - Hover over a Redis ACL assignment and click 
   - Create a Redis ACL association - Click 

3. Select the databases that the Redis ACL applies to.

4. Select the Redis ACL that defines the access to commands and keys.

5. Click 
   You can click to assign a Redis ACL to another database.

6. Click Save.

Users that are assigned to the role can access the databases according to the Redis ACL definitions.

Assigning a role to a user

To assign a role to a user:

1. In Data Access Control > Users:
   - Edit an existing role - Hover over a role and click 
   - Create a new role - Click 

2. Select a role for the user.
   You can also change the user password.

3. Click 

Updated: January 12, 2022