» mysql_database

The `mysql_database` resource creates and manages a database on a MySQL server.

Caution: The `mysql_database` resource can completely delete your database just as easily as it can create it. To avoid costly accidents, consider setting `prevent_destroy` on your database resources as an extra safety measure.

» Example Usage

```
resource "mysql_database" "app" {
  name = "my_awesome_app"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the database. This must be unique within a given MySQL server and may or may not be case-sensitive depending on the operating system on which the MySQL server is running.
- `default_character_set` - (Optional) The default character set to use when a table is created without specifying an explicit character set. Defaults to "utf8".
- `default_collation` - (Optional) The default collation to use when a table is created without specifying an explicit collation. Defaults to `utf8_general_ci`. Each character set has its own set of collations, so changing the character set requires also changing the collation.

Note that the defaults for character set and collation above do not respect any defaults set on the MySQL server, so that the configuration can be set appropriately even though Terraform cannot see the server-level defaults. If you wish to use the server's defaults you must consult the server's configuration and then set the `default_character_set` and `default_collation` to match.

» Attributes Reference

No further attributes are exported.

» mysql_grant

The `mysql_grant` resource creates and manages privileges given to a user on a MySQL server.

» Example Usage

```
resource "mysql_user" "jdoe" {
  user     = "jdoe"
  host     = "example.com"
  password = "password"
}

resource "mysql_grant" "jdoe" {
  user       = "${mysql_user.jdoe.user}"
  host       = "${mysql_user.jdoe.host}"
  database   = "app"
  privileges = ["SELECT", "UPDATE"]
}
```

» Argument Reference

The following arguments are supported:

- `user` - (Required) The name of the user.
- `host` - (Optional) The source host of the user. Defaults to "localhost".
- `database` - (Required) The database to grant privileges on. At this time, privileges are given to all tables on the database (`mydb.*`).
- `privileges` - (Required) A list of privileges to grant to the user. Refer to a list of privileges (such as here) for applicable privileges.
- `grant` - (Optional) Whether to also give the user privileges to grant the same privileges to other users.

» Attributes Reference

No further attributes are exported.

» mysql_user

The `mysql_user` resource creates and manages a user on a MySQL server.

Note: The password for the user is provided in plain text, and is obscured by an unsalted hash in the state Read more about sensitive data in state. Care is required when using this resource, to avoid disclosing the password.

» Example Usage

```
resource "mysql_user" "jdoe" {
  user               = "jdoe"
  host               = "example.com"
  plaintext_password = "password"
}
```

» Example Usage with an Authentication Plugin

```
resource "mysql_user" "nologin" {
  user               = "nologin"
  host               = "example.com"
  auth_plugin        = "mysql_no_login"
}
```

» Argument Reference

The following arguments are supported:

- `user` - (Required) The name of the user.
- `host` - (Optional) The source host of the user. Defaults to ”localhost”.
- `plaintext_password` - (Optional) The password for the user. This must be provided in plain text, so the data source for it must be secured. An *unsalted* hash of the provided password is stored in state. Conflicts with `auth_plugin`.
- `password` - (Optional) Deprecated alias of `plaintext_password`, whose value is *stored as plaintext in state*. Prefer to use `plaintext_password` instead, which stores the password as an unsalted hash. Conflicts with `auth_plugin`.
- `auth_plugin` - (Optional) Use an authentication plugin to authenticate the user instead of using password authentication. Description of the fields allowed in the block below. Conflicts with `password` and `plaintext_password`.

The `auth_plugin` value supports:

- `AWSAuthenticationPlugin` - Allows the use of IAM authentication with Amazon Aurora. For more details on how to use IAM auth with Aurora, see here.
- `mysql_no_login` - Uses the MySQL No-Login Authentication Plugin. The No-Login Authentication Plugin must be active in MySQL. For more information, see here.

» Attributes Reference

No further attributes are exported.