
Ansible Role: MySQL



Installs and configures MySQL or MariaDB server on RHEL/CentOS or Debian/Ubuntu servers.

Requirements

No special requirements; note that this role requires root access, so either run it in a playbook with a global `become: yes`, or invoke the role in your playbook like:

```
1 - hosts: database
2   roles:
3     - role: geerlingguy.mysql
4       become: yes
```

Role Variables

Available variables are listed below, along with default values (see `defaults/main.yml`):

```
1 mysql_user_home: /root
2 mysql_user_name: root
3 mysql_user_password: root
```

The home directory inside which Python MySQL settings will be stored, which Ansible will use when connecting to MySQL. This should be the home directory of the user which runs this Ansible role. The `mysql_user_name` and `mysql_user_password` can be set if you are running this role under a non-root user account and want to set a non-root user.

```
1 mysql_root_home: /root
2 mysql_root_username: root
3 mysql_root_password: root
```

The MySQL root user account details.

```
1 mysql_root_password_update: false
```

Whether to force update the MySQL root user's password. By default, this role will only change the root user's password when MySQL is first configured. You can force an update by setting this to `yes`.

Note: If you get an error like `ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)` after a failed or interrupted playbook run, this usually means the root password wasn't originally updated to begin with. Try ei-

ther removing the `.my.cnf` file inside the configured `mysql_user_home` or updating it and setting `password=''` (the insecure default password). Run the playbook again, with `mysql_root_password_update` set to `yes`, and the setup should complete.

Note: If you get an error like `ERROR 1698 (28000): Access denied for user 'root'@'localhost' (using password: YES)` when trying to log in from the CLI you might need to run as root or sudoer.

```
1 mysql_enabled_on_startup: true
```

Whether MySQL should be enabled on startup.

```
1 mysql_config_file: *default value depends on OS*
2 mysql_config_include_dir: *default value depends on OS*
```

The main `my.cnf` configuration file and include directory.

```
1 overwrite_global_mycnf: true
```

Whether the global `my.cnf` should be overwritten each time this role is run. Setting this to `no` tells Ansible to only create the `my.cnf` file if it doesn't exist. This should be left at its default value (`yes`) if you'd like to use this role's variables to configure MySQL.

```
1 mysql_config_include_files: []
```

A list of files that should override the default global `my.cnf`. Each item in the array requires a “src” parameter which is a path to a file. An optional “force” parameter can force the file to be updated each time ansible runs.

```
1 mysql_databases: []
```

The MySQL databases to create. A database has the values `name`, `encoding` (defaults to `utf8`), `collation` (defaults to `utf8_general_ci`) and `replicate` (defaults to `1`, only used if replication is configured). The formats of these are the same as in the `mysql_db` module.

You can also delete a database (or ensure it's not on the server) by setting `state` to `absent` (defaults to `present`).

```
1 mysql_users: []
```

The MySQL users and their privileges. A user has the values:

- `name`
- `host` (defaults to `localhost`)

- `password` (can be plaintext or encrypted—if encrypted, set `encrypted: yes`)
- `encrypted` (defaults to `no`)
- `priv` (defaults to `*.*:USAGE`)
- `append_privs` (defaults to `no`)
- `state` (defaults to `present`)

The formats of these are the same as in the `mysql_user` module.

```
1 mysql_packages:
2   - mysql
3   - mysql-server
```

(OS-specific, RedHat/CentOS defaults listed here) Packages to be installed. In some situations, you may need to add additional packages, like `mysql-devel`.

```
1 mysql_enablerepo: ""
```

(RedHat/CentOS only) If you have enabled any additional repositories (might I suggest `geerlingguy.repo-epel` or `geerlingguy.repo-remi`), those repositories can be listed under this variable (e.g. `remi`, `epel`). This can be handy, as an example, if you want to install later versions of MySQL.

```
1 mysql_python_package_debian: python3-mysqldb
```

(Ubuntu/Debian only) If you need to explicitly override the MySQL Python package, you can set it here. Set this to `python-mysqldb` if using older distributions running Python 2.

```
1 mysql_port: "3306"
2 mysql_bind_address: '0.0.0.0'
3 mysql_datadir: /var/lib/mysql
4 mysql_socket: *default value depends on OS*
5 mysql_pid_file: *default value depends on OS*
```

Default MySQL connection configuration.

```
1 mysql_log_file_group: mysql *adm on Debian*
2 mysql_log: ""
3 mysql_log_error: *default value depends on OS*
4 mysql_syslog_tag: *default value depends on OS*
5 ``yaml
6
7 MySQL logging configuration. Setting `mysql_log` (the general query log
  ) or `mysql_log_error` to `syslog` will make MySQL log to syslog
  using the `mysql_syslog_tag`.
8
9 ``yaml
10 mysql_slow_query_log_enabled: false
11 mysql_slow_query_log_file: *default value depends on OS*
12 mysql_slow_query_time: 2
```

Slow query log settings. Note that the log file will be created by this role, but if you're running on a server with SELinux or AppArmor, you may need to add this path to the allowed paths for MySQL, or disable the mysql profile. For example, on Debian/Ubuntu, you can run `sudo ln -s /etc/apparmor.d/usr.sbin.mysqld /etc/apparmor.d/disable/usr.sbin.mysqld && sudo service apparmor restart`.

```
1 mysql_key_buffer_size: "256M"
2 mysql_max_allowed_packet: "64M"
3 mysql_table_open_cache: "256"
4 ...
```

The rest of the settings in `defaults/main.yml` control MySQL's memory usage and some other common settings. The default values are tuned for a server where MySQL can consume 512 MB RAM, so you should consider adjusting them to suit your particular server better.

```
1 mysql_server_id: "1"
2 mysql_max_binlog_size: "100M"
3 mysql_binlog_format: "ROW"
4 mysql_expire_logs_days: "10"
5 mysql_replication_role: ''
6 mysql_replication_master: ''
7 mysql_replication_user: {}
```

Replication settings. Set `mysql_server_id` and `mysql_replication_role` by server (e.g. the master would be ID 1, with the `mysql_replication_role` of `master`, and the slave would be ID 2, with the `mysql_replication_role` of `slave`). The `mysql_replication_user` uses the same keys as individual list items in `mysql_users`, and is created on master servers, and used to replicate on all the slaves.

`mysql_replication_master` needs to resolve to an IP or a hostname which is accessible to the Slaves (this could be a `/etc/hosts` injection or some other means), otherwise the slaves cannot communicate to the master.

If the replication master has different IP addresses where you are running ansible and where the mysql replica is running, you can *optionally* specify a `mysql_replication_master_inventory_host` to access the machine (e.g. you run ansible on your local machine, but the mysql master and replica need to communicate on a different network)

```
1 mysql_hide_passwords: false
```

Do you need to hide tasks' output which contain passwords during the execution ?

Later versions of MySQL on CentOS 7

If you want to install MySQL from the official repository instead of installing the system default MariaDB equivalents, you can add the following `pre_tasks` task in your playbook:

```
1  pre_tasks:
2    - name: Install the MySQL repo.
3      yum:
4        name: http://repo.mysql.com/mysql-community-release-el7-5.
           noarch.rpm
5        state: present
6        when: ansible_os_family == "RedHat"
7
8    - name: Override variables for MySQL (RedHat).
9      set_fact:
10        mysql_daemon: mysqld
11        mysql_packages: ['mysql-server']
12        mysql_log_error: /var/log/mysqld.err
13        mysql_syslog_tag: mysqld
14        mysql_pid_file: /var/run/mysqld/mysqld.pid
15        mysql_socket: /var/lib/mysql/mysql.sock
16        when: ansible_os_family == "RedHat"
```

MariaDB usage

This role works with either MySQL or a compatible version of MariaDB. On RHEL/CentOS 7+, the mariadb database engine was substituted as the default MySQL replacement package. No modifications are necessary though all of the variables still reference 'mysql' instead of mariadb.

Ubuntu 14.04 and 16.04 MariaDB configuration On Ubuntu, the package names are named differently, so the `mysql_package` variable needs to be altered. Set the following variables (at a minimum):

```
1  mysql_packages:
2    - mariadb-client
3    - mariadb-server
4    - python-mysqldb
```

Dependencies

If you have `ansible` installed (e.g. `pip3 install ansible`), none.

If you have only installed `ansible-core`, be sure to require `community.mysql` in your `collections/requirements.yml` or install it manually with `ansible-galaxy collection install community.mysql`.

Example Playbook

```
1 - hosts: db-servers
2   become: yes
3   vars_files:
4     - vars/main.yml
5   roles:
6     - { role: geerlingguy.mysql }
```

Inside `vars/main.yml`:

```
1 mysql_root_password: super-secure-password
2 mysql_databases:
3   - name: example_db
4     encoding: latin1
5     collation: latin1_general_ci
6 mysql_users:
7   - name: example_user
8     host: "%"
9     password: similarly-secure-password
10    priv: "example_db.*:ALL"
```

License

MIT / BSD

Author Information

This role was created in 2014 by Jeff Geerling, author of Ansible for DevOps.