Dockerfile:

|  |
| --- |
| FROM alpine:3.3  RUN apk update && apk add mariadb mariadb-client && apk add bash  RUN mysql\_install\_db --user=mysql  ENV TERM xterm  COPY mariadb\_conf/\* /etc/mysql/  VOLUME /var/lib/mysql  VOLUME /var/log/mysql  RUN mkdir -p /var/backups/mysql  RUN chmod a+r /var/backups/mysql/  VOLUME /var/backups/mysql  ENTRYPOINT ["mysqld\_safe"]  EXPOSE 3306 |

В директории где лежит докерфайл нужно создать директорию mariadb\_conf и в ней создать конфигурационный файл для MariaDB – **my.cnf**, который содержит:

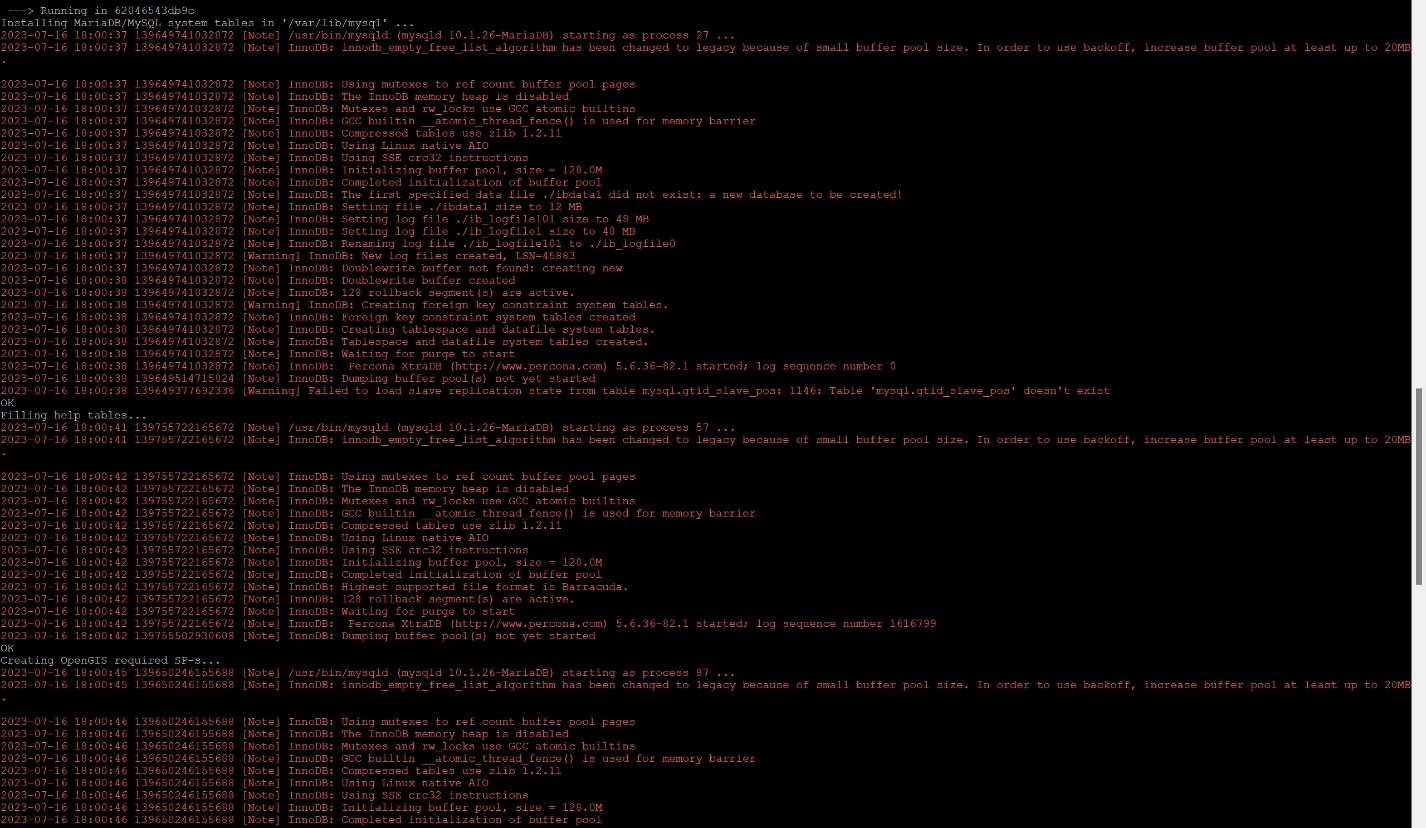
my.cnf:

|  |
| --- |
| # MariaDB database server configuration file.  #  # You can copy this file to one of:  # - "/etc/mysql/my.cnf" to set global options,  # - "~/.my.cnf" to set user-specific options.  #  # One can use all long options that the program supports.  # Run program with --help to get a list of available options and with  # --print-defaults to see which it would actually understand and use.  #  # For explanations see  # http://dev.mysql.com/doc/mysql/en/server-system-variables.html  # This will be passed to all mysql clients  # It has been reported that passwords should be enclosed with ticks/quotes  # escpecially if they contain "#" chars...  # Remember to edit /etc/mysql/debian.cnf when changing the socket location.  [client]  port = 3306  socket = /var/run/mysqld/mysqld.sock  # Here is entries for some specific programs  # The following values assume you have at least 32M ram  # This was formally known as [safe\_mysqld]. Both versions are currently parsed.  [mysqld\_safe]  socket = /var/run/mysqld/mysqld.sock  nice = 0  [mysqld]  #  # \* Basic Settings  #  user = mysql  pid-file = /var/run/mysqld/mysqld.pid  socket = /var/run/mysqld/mysqld.sock  port = 3306  basedir = /usr  datadir = /var/lib/mysql  tmpdir = /tmp  lc\_messages\_dir = /usr/share/mysql  lc\_messages = en\_US  skip-external-locking  default\_storage\_engine = InnoDB  bind-address = 0.0.0.0  #  # \* Fine Tuning  #  max\_connections = 100  connect\_timeout = 5  wait\_timeout = 600  lock\_wait\_timeout = 600  max\_allowed\_packet = 64M  thread\_cache\_size = 128  bulk\_insert\_buffer\_size = 16M  tmp\_table\_size = 64M  max\_heap\_table\_size = 64M  #  # \* MyISAM  #  # This replaces the startup script and checks MyISAM tables if needed  # the first time they are touched. On error, make copy and try a repair.  myisam\_recover = FORCE,BACKUP  key\_buffer\_size = 128M  table\_open\_cache = 400  #  # \* Query Cache Configuration  #  query\_cache\_size = 0  query\_cache\_type = OFF  #  # \* Logging and Replication  #  #general\_log\_file = /var/log/mysql/mysql.log  #general\_log = 1  #  log\_error = mysql.err  # we do want to know about network errors and such  log\_warnings = 2  #  # Enable the slow query log to see queries with especially long duration  slow\_query\_log = 0  #slow\_query\_log\_file = /var/log/mysql/mariadb-slow.log  #long\_query\_time = 2  #log\_slow\_rate\_limit = 1000  #log\_slow\_verbosity = query\_plan  #log-queries-not-using-indexes  #  # The following can be used as easy to replay backup logs or for replication.  # note: if you are setting up a replication slave, see README.Debian about  # other settings you may need to change.  #server-id = 1  log\_bin = /var/log/mysql/mariadb-bin  log\_bin\_index = /var/log/mysql/mariadb-bin.index  # not fab for performance, but safer  sync\_binlog = 0  expire\_logs\_days = 10  max\_binlog\_size = 200M  # slaves  #relay\_log = /var/log/mysql/relay-bin  #relay\_log\_index = /var/log/mysql/relay-bin.index  #relay\_log\_info\_file = /var/log/mysql/relay-bin.info  #log\_slave\_updates  #read\_only  #  # If applications support it, this stricter sql\_mode prevents some  # mistakes like inserting invalid dates etc.  #sql\_mode = NO\_ENGINE\_SUBSTITUTION,TRADITIONAL  #  # \* InnoDB  #  # InnoDB is enabled by default with a 10MB datafile in /var/lib/mysql/.  # Read the manual for more InnoDB related options. There are many!  # you can't just change log file size, requires special procedure  innodb\_log\_file\_size = 512M  innodb\_buffer\_pool\_size = 512M  innodb\_log\_buffer\_size = 64M  innodb\_file\_per\_table = 1  innodb\_strict\_mode = 1  innodb\_open\_files = 400  innodb\_io\_capacity = 400  innodb\_flush\_method = O\_DIRECT  #  # \* Security Features  #  # Read the manual, too, if you want chroot!  # chroot = /var/lib/mysql/  #  # For generating SSL certificates I recommend the OpenSSL GUI "tinyca".  #  # ssl-ca=/etc/mysql/cacert.pem  # ssl-cert=/etc/mysql/server-cert.pem  # ssl-key=/etc/mysql/server-key.pem  #  # \* plugins  #  plugin-load-add=ha\_connect.so  plugin-load-add=metadata\_lock\_info.so  plugin-load-add=query\_response\_time.so  plugin-load-add=locales.so  [mysqldump]  quote-names  create-options  extended-insert  quick  max\_allowed\_packet = 64M  flush-logs  single-transaction  [mysql]  no-auto-rehash  [isamchk]  key\_buffer\_size = 64M  sort\_buffer\_size = 64M  read\_buffer = 32M  write\_buffer = 32M  #  # Include MariaDB Galera Cluster configuration  # WARNING: Galera is not yet supported in Alpine Linux  #  # !include /etc/mysql/galera.cnf |

Это конфигурация для MariaDB, по идее он вроде как должен сам ее создавать, но у меня по чему то не хотел, важная строка «bind-address = 0.0.0.0» - открывает для прослушивания все ip-адреса иначе Машка слушает только локальную сеть и подключиться к ней через php не получится.

Собираем образ «sudo docker build -t test .»

Вылазила такая фигня не пугайтесь, если после красных строк ОК значит все путем:



Создаем и запускаем контейнер с пробросом порта

**«sudo docker run --name test -p 3306:3306 test»**

если все нормально контейнер стартанет и терминал будет не доступен.

Открываем второй терминал, логинимся и заходим в созданный контейнер

**«sudo docker exec -it test bash»**

провалившись в контейнер заходим в mysql, что-бы добавить пользователя с правами на удаленное управление Машкой

1. **mysql**
2. Зайдя в mySQL вводим команду для добавления пользователя (user и password заменить на имя пользователя и пароль)  **GRANT ALL PRIVILEGES ON userdata.\* TO 'user'@'%' IDENTIFIED BY 'password';**

Выходим из контейнера и создаем контейнер phpMyAdmin с пробросом порта (по идее сюда же нужно включить проброс директории, но у меня при проброшенной директории с БД перестал стартовать контейнер)

**«sudo docker run --name php -d --link test:db -p 8081:80 phpmyadmin/phpmyadmin»**

В браузере заходим в phpMyAdmin и логинимся под созданным пользователем (из пункта 2)

Из косяков:

- не работает общая директория для сохранения БД

- требуются манипуляции с контейнером

Но у меня уже нет сил с это домашкой :)

Гуглил в инете:

<https://www.oslogic.ru/knowledge/1085/nastrojka-udalennogo-dostupa-mysql-i-mariadb-v-linux-ubuntu/>

<https://github.com/docker-library/mysql/issues/275>

<https://wiki.dieg.info/my.cnf>

<https://github.com/federico-razzoli/docker-mariadb-alpine/blob/master/Dockerfile>

P.S. Можно в докерфайле использовать более свежую версию alpine