鲲鹏软件栈 数据库

# 移植指南

文档版本 02

发布日期 2020-08-06

华为技术有限公司

## 商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

# 目 录



# 1 移植 MySQL 5.7

## 1.1 环境要求

本文以MySQL 5.7版本，基于新安装的CentOS Linux release 7.6.1810系统环境进行描述。

准备的云服务器要求如表1-1所示。

**表 1-1** 云服务器信息

| 名称 | 说明 |
| --- | --- |
| 规格 | 16vCPUs \| 64GB \| kc1.4xlarge.4 |
| 镜像 | CentOS 7.6 64bit with ARM（ CentOS Linux release 7.6.1810 ） |
| 公网IP | 申请弹性公网IP地址，带宽为5 Mbit/s |
| 私有IP | 192.168.0.166 |
| 系统盘 | 40GB |

## 1.2 配置编译环境

### 步骤一：检查 cmake 是否安装

检查是否安装。

**rpm -qa |grep cmake**

如果未安装，需要安装。可参见Cmake 3.9.2 安装指南（ CentOS 7.6 ）进行安装。

### 步骤二：安装依赖包

执行以下命令安装依赖包：

**yum install bison* ncurses***

**yum install -y bzip2 wget**

## 步骤三：升级 gcc 版本至 5.3 或者以上

**步骤1** 检查gcc的版本。

**gcc --version**

**步骤2** （可选）安装gcc7.3。

**说明**

本文档以7.3版本为例。

当版本不满足要求时，需要安装gcc。

1. 下载gcc7.3。

   下载地址：https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-7.3.0/gcc-7.3.0.tar.gz

2. 将“gcc-7.3.0.tar.gz”放置于“/home”目录下，并解压安装。

   **cd /home**

   **tar -xvf gcc-7.3.0.tar.gz**

**步骤3** 下载isl、gmp、mpc、mpfr。

1. 在“gcc-7.3.0”目录下，检查gcc的依赖包是否已下载和安装。

   **./contrib/download_prerequisites**

2. （可选）根据需要，下载“gmp-6.1.0.tar.bz2”、“isl-0.16.1.tar.bz2”、“mpc-1.0.3.tar.gz”或“mpfr-3.1.4.tar.bz2”。

   在**步骤3.1**中检查到存在上述依赖包未下载安装时，请根据需要执行相应的命令下载。

   **wget https://gcc.gnu.org/pub/gcc/infrastructure/gmp-6.1.0.tar.bz2**

   **wget https://gcc.gnu.org/pub/gcc/infrastructure/isl-0.16.1.tar.bz2**

   **wget https://gcc.gnu.org/pub/gcc/infrastructure/mpc-1.0.3.tar.gz**

   **wget https://gcc.gnu.org/pub/gcc/infrastructure/mpfr-3.1.4.tar.bz2**

3. 将安装包放置于“/home/gcc-7.3.0”目录下。

**步骤4** 编译安装gcc。

1. 编译gcc。

   **说明**

   “-j”参数可利用多核CPU加快编译速度，在本示例中，使用的是16核CPU，所以此处为“-j16”。

   可通过下述命令查询CPU核数：

   **cat /proc/cpuinfo| grep "processor"| wc -l**

   **cd /home/gcc-7.3.0**

   **mkdir gcc-build-7.3.0**

   **cd gcc-build-7.3.0**

   **../configure --enable-checking=release --enable-language=c,c++ --disable-multilib --prefix=/usr**

**make -j*16***

2. 执行安装。

   **make install**

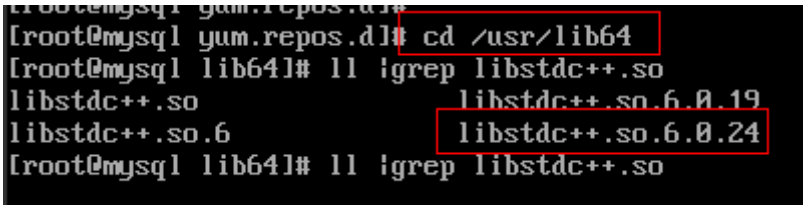
3. 确定“libstdc++.so”是在“/usr/lib64”目录下。

   **cd /usr/lib64**

   **ll |grep libstdc++.so**

   查询结果如图1-1所示。

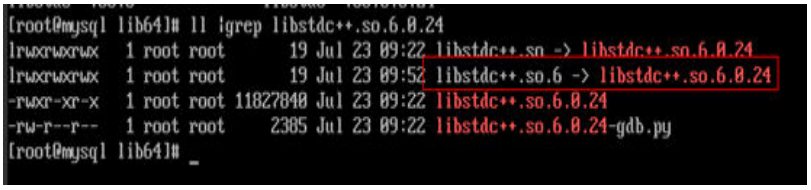
   **图 1-1** 查询“libstdc++.so”所在位置

4. 确定软连接存在，如图1-2所示。

   **ll |grep libstdc++.so.6.0.24**

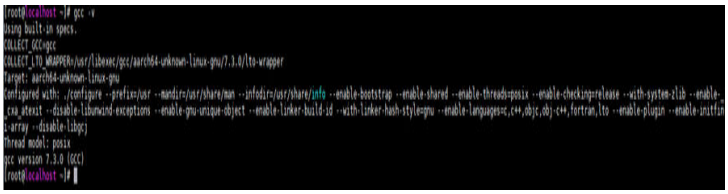
   **图 1-2** 确定软连接存在

5. 查看版本，如图1-3所示。

   **gcc -v**

   **图 1-3** 查看 gcc 版本

**----结束**

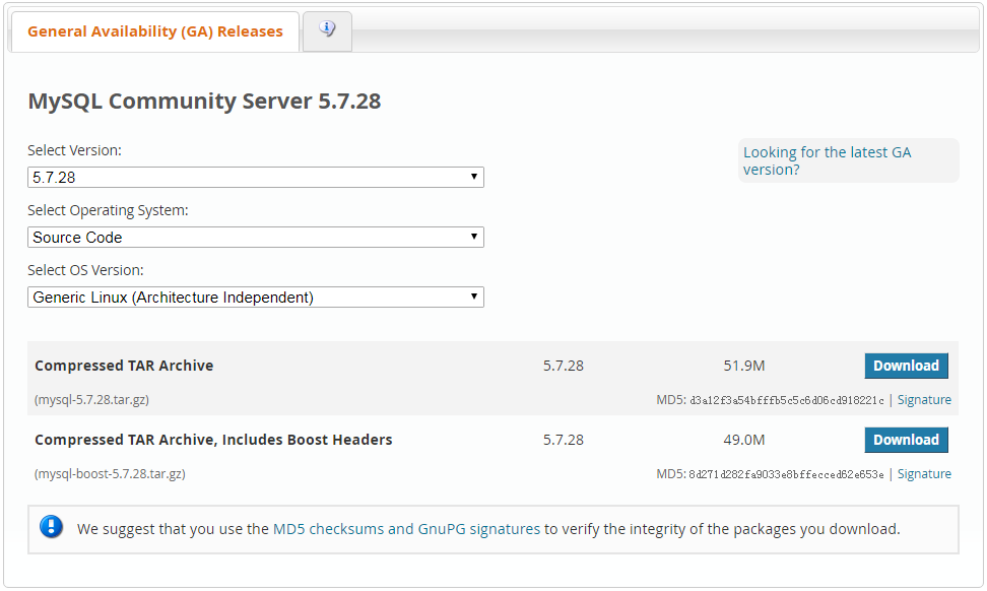
# 1.3 下载源码

在“https://dev.mysql.com/downloads/mysql/5.7.html#downloads”下载MySQL源码包，如图1-4所示，并上传至服务器。

**须知**

下载MySQL源码包时，如果版本为“5.x.x”，则必须为5.7.27以上，如果版本为“8.x.x”，则必须为8.0.17以上。否则需要安装补丁，补丁的安装请参见本文档“编译安装”中的描述。

**图 1-4** 下载 MySQL 源码




# 1.4 编译安装

**步骤1** 进入下载的源码包所在的目录，解压缩源码包。

解压后生成“mysql-5.7.x”文件夹，具体版本号以实际为准。

**tar -zvxf mysql-boost-5.7.*x*.tar.gz**

**步骤2** （可选）安装补丁。

MySQL源码包版本为“5.x.x”，则必须为5.7.27以上，版本为“8.x.x”，则必须为8.0.17以上。否则需要安装补丁。

1. 下载补丁文件，并放置到“mysql-5.7.x”路径下。
   补丁下载路径：https://bugs.mysql.com/file.php?id=28180&bug_id=94699
2. 进入目录并打补丁。
   **cd mysql-5.7.*XX***
   **patch -p1 < 0001-Bug-94699-Mysql-deadlock-and-bugcheck-on-aarch64.patch**

**步骤3** 在解压后的源码包路径“mysql-5.7.x”下，创建“cmake.sh”。

**vim cmake.sh**

文件内容如下，其中，“DWITH_BOOST”的取值请根据实际的boost路径修改。

```
cmake -DCMAKE_INSTALL_PREFIX=/usr/local/mysql \
-DMYSQL_DATADIR=/data/data \
```

```
-DSYSCONFDIR=/etc \
-DWITH_INNOBASE_STORAGE_ENGINE=1 \
-DWITH_PARTITION_STORAGE_ENGINE=1 \
-DWITH_FEDERATED_STORAGE_ENGINE=1 \
-DWITH_BLACKHOLE_STORAGE_ENGINE=1 \
-DWITH_MYISAM_STORAGE_ENGINE=1 \
-DENABLED_LOCAL_INFILE=1 \
-DENABLE_DTRACE=0 \
-DDEFAULT_CHARSET=utf8mb4 \
-DDEFAULT_COLLATION=utf8mb4_general_ci \
-DWITH_EMBEDDED_SERVER=1 \
-DDOWNLOAD_BOOST=1 \
-DWITH_BOOST=/mysql/mysql-5.7.28/boost/boost_1_59_0
```

**步骤4** 给“cmake.sh”赋以权限并运行，等待运行完成。

**chmod +x cmake.sh**

**./cmake.sh**

**说明**

- 在执行**./cmake.sh**时会自动下载“boost_1_59_0.tar.gz”，若出现超时报错的情况，可手工下载，并将文件放置到“cmake.sh”配置文件中“DWITH_BOOST”指定对应的路径下。

  **wget http://sourceforge.net/projects/boost/files/boost/1.59.0/boost_1_59_0.tar.gz**
- 若在预编译时出现依赖包不全的情况，可自行查阅资料安装依赖包，并重新预编译。重新预编译前，需要执行**rm -f CMakeCache.txt**删“CMakeCache.txt文件。”

**步骤5** 在MySQL源码路径下运行**make -j*16***，等待编译完成。

**说明**

“-j”参数可利用多核CPU加快编译速度，在本示例中，使用的是16核CPU，所以此处为“-j16”。

可通过下述命令查询CPU核数：

**cat /proc/cpuinfo| grep "processor"| wc -l**

**make -j*16***

**步骤6** 运行**make install**，等待安装过程结束。

**----结束**

# 1.5 配置 MySQL

1. 创建“mysql”用户及用户组。

   **groupadd mysql**

   **useradd -g mysql mysql**
2. 修改“/usr/local/mysql”权限。

   **chown -R mysql:mysql /usr/local/mysql**
3. 进入安装路径，创建“data”、“log”、“run”文件夹，执行初始化配置脚本，生成初始的数据库和表。

   需要指出的是，执行下述命令后，会产生初始随机密码，需要记录。

   **cd /usr/local/mysql**

   **mkdir -p /data/log /data/data /data/run**

**bin/mysqld --initialize --basedir=/usr/local/mysql --datadir=/data/data --user=mysql**

4. 创建“mysql.log”和“mysql.pid”文件，赋予“mysql”用户及用户组权限。

   其中，创建的“mysql.log”和“mysql.pid”文件是空文件，创建后保存退出即可。

   **vi /data/log/mysql.log**

   **vi /data/run/mysql.pid**

   **chown -R mysql:mysql /data**

5. 修改“my.cnf”中的文件路径，如图1-5所示。

   **vim /etc/my.cnf**

**图 1-5** 修改“my.cnf”中的文件路径

```
[mysqld]
datadir=/data/data
socket=/data/data/mysql.sock
#datadir=/opt/mysql/data
#socket=/opt/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mariadb according to the
# instructions in http://fedoraproject.org/wiki/Systemd

[mysqld_safe]
log-error=/data/log/mysql.log
pid-file=/data/run/mysql.pid
#log-error=/var/log/mariadb/mariadb.log
#pid-file=/var/run/mariadb/mariadb.pid

#
# include all files from the config directory
#
!includedir /etc/my.cnf.d
```

# 1.6 运行 MySQL

1. 启动MySQL服务。

   **cp support-files/mysql.server /etc/init.d/mysql**

   **chkconfig mysql on**

   **service mysql start**

2. 将以下内容添加进环境变量，并使之生效。

   a. 编辑文件并添加内容。

      **vim ~/.bash_profile**

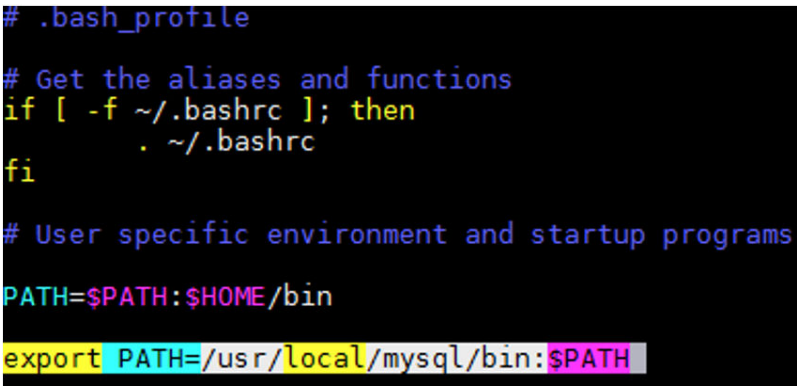
      添加的内容如下：

      ```
      export PATH=/usr/local/mysql/bin:$PATH
      ```

      添加环境变量后如图1-6所示。

**图 1-6** 添加环境变量




b. 使环境变量生效

**source ~/.bash_profile**

3. 建立套接字软链接，接入MySQL环境。

需要输入的密码为**配置MySQL**时产生的初始密码，请留意初始密码包含了特殊字符。

**ln -s /data/data/mysql.sock /tmp/mysql.sock**

**mysql -uroot -p**

**说明**

如果忘记初始密码，可参见**问题三：忘记MySQL的初始密码**设置密码并重新连接。

4. 修改密码。

下述命令中的“mypassword”需要根据实际修改成要配置的密码。

**SET PASSWORD = PASSWORD('mypassword');**

**UPDATE mysql.user SET authentication_string =PASSWORD('mypassword') WHERE User='mysql';**

**GRANT ALL PRIVILEGES ON *.* TO mysql@localhost IDENTIFIED BY 'mypassword' WITH GRANT OPTION;**

**GRANT ALL PRIVILEGES ON *.* TO mysql@"%" IDENTIFIED BY 'mypassword' WITH GRANT OPTION;**

**GRANT ALL PRIVILEGES ON *.* TO root@localhost IDENTIFIED BY 'mypassword' WITH GRANT OPTION;**

**GRANT ALL PRIVILEGES ON *.* TO root@"%" IDENTIFIED BY 'mypassword' WITH GRANT OPTION;**

5. 使用新的密码重新登录。

下述命令中的“mypassword”需要根据实际修改成要配置的密码。

**mysql -uroot –pmypassword**

# 1.7 常见问题

## 问题一：启动 MySQL 服务时提示 log 文件不存在

**问题描述：**

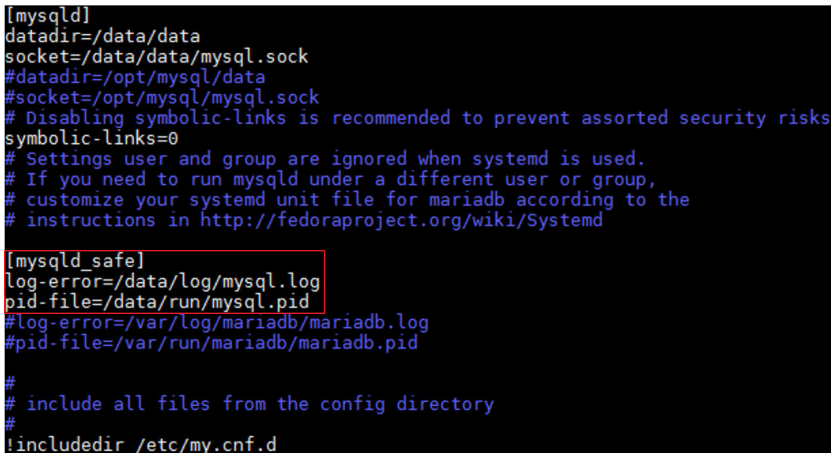
启动MySQL时，提示如**图1-7**所示。

**图 1-7** 提示 log 文件不存在


**解决方法：**

在“/etc/my.cnf”下配置正确的log路径，如图1-8所示，并赋予“mysql”用户及用户组权限。

**图 1-8** 配置 log 路径




## 问题二：启动 MySQL 服务提示“ERROR! The server quit without updating PID file”

**问题描述：**

启动MySQL时，提示如图1-9所示。

**图 1-9** 提示更新错误


**解决方法：**

1. MySQL安装及“data/log/run”路径未赋予用户及用户组正确权限，请使用以下命令赋权。

   **chown -R mysql:mysql /usr/local/mysql**

   **chown -R mysql:mysql /data**

2. 查看是否已有mysql进程在运行，kill掉后再尝试。

   **ps -ef | grep mysqld**

   **kill -9 *进程号***

## 问题三：忘记 MySQL 的初始密码

**问题描述：**

在连接MySQL时，忘记初始密码。

**解决方法：**

用以下命令启动MySQL，以不检查权限的方式启动，然后重新设置密码，重新登录。

**service mysql stop**

**service mysql start --skip-grant-tables**

**mysql -uroot -p**

回显内容如下，请按照说明操作。

```
[root@ecs mysql]# mysql -uroot -p
Enter password:    //直接按回车键
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.28 Source distribution

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> UPDATE mysql.user SET authentication_string=PASSWORD('password') where USER='root'; //在此处更改密码，password设置为实际要配置的密码
mysql> flush privileges; //刷新权限
mysql> exit //退出MySQL
[root@ecs mysql]# mysql -uroot -p
Enter password: //此处输入设置的密码
```

# 2 移植 MySQL 5.6

## 介绍

### 简要介绍

MySQL是一个关系型数据库管理系统。

语言：C/C++

一句话描述：关系型数据库

### 建议的版本

建议使用版本为“mysql-5.6.44”。

## 环境要求

### 云服务器要求

本文以云服务器KC1实例测试，云服务器配置如表2-1所示。

**表 2-1** 云服务器配置

| 项目 | 说明 |
| --- | --- |
| 规格 | kc1.large.2 \| 4vCPUs \| 8GB |
| 磁盘 | 系统盘：高IO（40GB） |

### 操作系统要求

操作系统要求如表2-2所示。

**表 2-2** 操作系统要求

| 项目 | 说明 | 下载地址 |
| --- | --- | --- |
| CentOS | • 版本：7.5<br>• Kernel：4.14.0 | 在公共镜像中已提供 |

| 项目 | 说明 | 下载地址 |
| --- | --- | --- |
| EulerOS | • 版本：2.8<br>• Kernel：4.19.36 | 在公共镜像中已提供 |

## 配置编译环境

**步骤1** 安装依赖包。

**yum install gcc gcc-c++ make cmake libaio-devel openssl-devel zlib-devel ncurses-devel bison -y**

**步骤2** 下载解压boost。

**mkdir /usr/local/src/boost && cd /usr/local/src/boost**

**wget -c https://kent.dl.sourceforge.net/project/boost/boost/1.59.0/boost_1_59_0.tar.gz --no-check-certificate**

**tar –zxvf boost_1_59_0.tar.gz**

**----结束**

## 获取源码

执行如下命令，获取“mysql-5.6.44”源码包。

**mkdir /usr/local/src/mysql && cd /usr/local/src/mysql**

**wget -c https://downloads.mysql.com/archives/get/p/23/file/mysql-5.6.44.tar.gz**

## 编译和安装

**步骤1** 解压软件包。

**tar -zxvf mysql-5.6.44.tar.gz**

**步骤2** 进入MySQL安装目录。

**cd /usr/local/src/mysql/mysql-5.6.44**

**步骤3** 建立编译目录并进入编译目录。

**mkdir build && cd build**

**步骤4** 进行配置。

**mkdir /usr/local/mysql**

**cmake /usr/local/src/mysql/mysql-5.6.44 -DCMAKE_INSTALL_PREFIX=/usr/local/mysql -DWITH_BOOST=/usr/local/src/boost/boost_1_59_0**

**步骤5** 编译并安装MySQL源码。

**make && make install**

**----结束**

## 运行和验证

版本检查。

如果安装成功，会正确显示版本。

**/usr/local/mysql/bin/mysql --version**

# 3 安装 MariaDB

## 介绍

### 简要介绍

MariaDB数据库管理系统是MySQL的一个分支，主要由开源社区在维护，采用GPL授权许可MariaDB的目的是完全兼容MySQL，包括API和命令行，使之能轻松成为MySQL的代替品。

语言：C

一句话描述：MySQL开源分支。

开源协议：GPL

## 环境要求

### 云服务器要求

本文以云服务器KC1实例测试，云服务器配置如**表3-1**所示。

**表 3-1** 云服务器配置

| 项目 | 说明 |
| --- | --- |
| 规格 | kc1.large.2 \| 4vCPUs \| 8GB |
| 磁盘 | 系统盘：高IO（40GB） |

### 操作系统要求

操作系统要求如**表3-2**所示。

**表 3-2** 操作系统要求

| 项目 | 说明 | 下载地址 |
| --- | --- | --- |
| CentOS | • 版本：7.5<br>• Kernel：4.14.0 | 在公共镜像中已提供 |
| EulerOS | • 版本：2.8<br>• Kernel：4.19.36 | 在公共镜像中已提供 |

## 安装 Maraidb

安装Mariadb。

**yum install mariadb -y**

## 运行和验证

执行如下命令，验证Mariadb。

**mysql -V**

# 4 移植 PostgreSQL

## 介绍

### 简要介绍

PostgreSQL是以加州大学伯克利分校计算机系开发的POSTGRES，现在已经更名为PostgreSQL，版本 4.2为基础的对象关系型数据库管理系统（ORDBMS）。PostgreSQL支持大部分SQL标准并且提供了许多其他现代特性：复杂查询、外键、触发器、视图、事务完整性、MVCC。同样，PostgreSQL可以用许多方法扩展，比如，通过增加新的数据类型、函数、操作符、聚集函数、索引。免费使用、修改、和分发PostgreSQL，不管是私用、商用、还是学术研究使用。

语言：C/C++

一句话描述：对象关系型数据库管理系统。

开源协议：PostgreSQL

### 建议的版本

- 可通过yum安装和源码安装两种安装方式，请根据需要选择安装方式。
- 已在鲲鹏云服务器上验证过PostgreSQL-10.11版本，并以此版本为例进行说明，请根据实际需要选择版本。

## 环境要求

### 云服务器要求

本文以KC1实例测试，配置如表4-1所示

**表 4-1** 云服务器配置

| 项目 | 说明 |
| --- | --- |
| 规格 | kc1.large.2 \| 2vCPUs \| 4GB |
| 磁盘 | 系统盘：高IO（40GB） |

### 操作系统要求

操作系统要求如表4-2所示。

**表 4-2** 操作系统要求

| 项目 | 说明 | 下载地址 |
| --- | --- | --- |
| CentOS | 7.6 | 在公共镜像中已提供。 |
| Kernel | 4.14.0 | 包含在操作系统镜像中。 |

## 配置编译环境

安装依赖包。

- 通过yum安装方式，安装postgresql-contrib。
  **yum install postgresql-contrib.aarch64**
- 通过源码安装方式进行安装。
  **yum install -y readline readline-devel openssl openssl-devel zlib zlib-devel**

## 获取源码

通过源码安装方式，需下载PostgreSQL源码

**wget https://ftp.postgresql.org/pub/source/v10.11/postgresql-10.11.tar.gz**

## 编译和安装

yum安装方式：

**yum install postgresql-server.aarch64**

源码安装方式：

**步骤1** 解压源码。

**tar -zxvf postgresql-10.11.tar.gz**

**步骤2** 进入源码目录。

**cd postgresql-10.11**

**步骤3** 配置并编译安装。

**./configure && make && make install**

**----结束**

## 运行和验证

**yum安装方式：**

**步骤1** 初始化数据库。

**/usr/bin/postgresql-setup initdb**

**步骤2** 设置开启启动。

**systemctl start postgresql**

**systemctl enable postgresql.service**

**----结束**

**源码安装方式：**

**步骤1** 创建数据目录。

**mkdir /Data**

**mkdir /Data/postgresql**

**步骤2** 创建“pgsql”用户。

**useradd pgsql**

**步骤3** 变更目录权限。

**chown -R pgsql:pgsql /usr/local/pgsql**

**chown -R pgsql:pgsql /Data/postgresql**

**步骤4** 切换到“pgsql”用户。

**su - pgsql**

**步骤5** 初始化数据。

**/usr/local/pgsql/bin/initdb -D /Data/postgresql/**

```
The files belonging to this database system will be owned by user "pgsql".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /Data/postgresql ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default timezone ... PRC
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    /usr/local/pgsql/bin/pg_ctl -D /Data/postgresql/ -l logfile start
```

**步骤6** 创建日志目录及日志。

**cd /Data/postgresql**

**mkdir log**

**touch log/server.log**

**步骤7** 启动进程。

**/usr/local/pgsql/bin/pg_ctl -D /Data/postgresql/ -l /Data/postgresql/log/server.log start**

**步骤8** 检查进程。

**ps -ef | grep pgsql**

```
pgsql    15635     1  0 10:07 pts/0    00:00:00 /usr/local/pgsql/bin/postgres -D /Data/postgresql
pgsql    15637 15635  0 10:07 ?        00:00:00 postgres: checkpointer process
pgsql    15638 15635  0 10:07 ?        00:00:00 postgres: writer process
pgsql    15639 15635  0 10:07 ?        00:00:00 postgres: wal writer process
pgsql    15640 15635  0 10:07 ?        00:00:00 postgres: autovacuum launcher process
pgsql    15641 15635  0 10:07 ?        00:00:00 postgres: stats collector process
pgsql    15642 15635  0 10:07 ?        00:00:00 postgres: bgworker: logical replication launcher
```

**步骤9** 检查监听端口。

**netstat -tlnp**

```
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      15635/postgres
```

**----结束**

# 5 移植 SQLite

## 介绍

### 简要介绍

SQLite是一款轻量级的关系型数据库，它的运算速度非常快，占用资源很少，不仅支持标准的SQL语法，还遵循了数据库的ACID事务。

编写语言：C

一句话概述：轻量级的关系型数据库

### 建议的版本

建议使用版本为“3.7.17”。

## 环境要求

### 云服务器要求

本文以云服务器KC1实例测试，云服务器配置如表5-1所示。

**表 5-1** 云服务器配置

| 项目 | 说明 |
| --- | --- |
| 规格 | kc1.xlarge.2 \| 4vCPUs \| 8GB |
| 磁盘 | 系统盘：高IO（40GB） |

### 操作系统要求

操作系统要求如表5-2所示。

**表 5-2** 操作系统要求

| 项目 | 版本 | 下载地址 |
| --- | --- | --- |
| CentOS | 7.5 | 在公共镜像中已提供。 |

| 项目 | 版本 | 下载地址 |
| --- | --- | --- |
| Kernel | 4.14.0 | 在公共镜像中已提供。 |

## 获取源码

下载和解压SQLite软件包。

**cd /usr/local/src**

**wget https://www.sqlite.org/2019/sqlite-autoconf-3280000.tar.gz**

**tar -zxvf sqlite-autoconf-3280000.tar.gz**

## 编译和安装

编译和安装SQLite。

**cd /usr/local/src/sqlite-autoconf-3280000**

**./configure --prefix=/usr/local/sqlite/install && make -j4 && make install**

## 运行和验证

**步骤1** 查看SQLite版本

**sqlite3 -version**

回显内容如下：

```
3.7.17 2013-05-20 00:56:22 118a3b35693b134d56ebd780123b7fd6f1497668
```

**步骤2** 测试SQlite。

**cd /usr/local/src/**

**touch test.db**

**sqlite3 test.db**

进入SQLite后，再输入.databases查看数据库信息。

**.databases**

回显内容如下：

```
[root@ecs-0001 src]# sqlite3 test.db
SQLite version 3.7.17 2013-05-20 00:56:22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .databases
seq  name             file
---  ---------------  ----------------------------------------------------------
0    main             /usr/local/src/test.db
```

**----结束**

# 6 安装 LevelDB

## 介绍

### 简要介绍

LevelDB是Google用C++开发的一个快速的键值对存储数据库，提供从字符串键到字符串值的有序映射。

语言：C/C++

一句话描述：基于C++的存储数据库

### 建议的版本

建议使用版本为“leveldb-1.20”。

## 环境要求

### 云服务器要求

本文以云服务器KC1实例测试，云服务器配置如表6-1所示。

**表 6-1** 云服务器配置

| 项目 | 说明 |
| --- | --- |
| 规格 | kc1.xlarge.2 \| 4vCPUs \| 8GB |
| 磁盘 | 系统盘：高IO（40GB） |

### 操作系统要求

操作系统要求如表6-2所示。

**表 6-2** 操作系统要求

| 项目 | 版本 | 下载地址 |
| --- | --- | --- |
| EulerOS | 2.8 | 在公共镜像中已提供。 |

| 项目 | 版本 | 下载地址 |
| --- | --- | --- |
| Kernel | 4.19.36 | 在公共镜像中已提供。 |

## 配置安装环境

安装依赖包。

**yum install gcc-c++**

## 获取软件包

执行以下命令，获取LevelDB软件包。

**wget https://github.com/google/leveldb/archive/v1.20.tar.gz**

## 安装

**步骤1** 解压软件包。

**tar -zxvf v1.20.tar.gz**

**步骤2** 进入“leveldb-1.20”的安装目录。

**cd leveldb-1.20**

**步骤3** 编译OpenLDAP源码。

**make -j4**

**步骤4** 安装LevelDB。

**cp -r include/leveldb /usr/include/**

**cp out-shared/libleveldb.so.1.20 /usr/lib/**

**ln -s /usr/lib/libleveldb.so.1.20 /usr/lib/libleveldb.so.1**

**ln -s /usr/lib/libleveldb.so.1.20 /usr/lib/libleveldb.so**

**ldconfig**

**----结束**

## 运行和验证

**步骤1** 编写一个LevelDB程序，文件名为“hello_leveldb.cc”， 代码内容如下：

```
#include <iostream>
#include <cassert>
#include <cstdlib>
#include <string>
#include <leveldb/db.h>
using namespace std;
int main(void)
{
    leveldb::DB *db = nullptr;
    leveldb::Options options;
    options.create_if_missing = true;
```

```
    leveldb::Status status = leveldb::DB::Open(options, "/tmp/testdb", &db);
    assert(status.ok());
    std::string key = "A";
    std::string value = "a";
    std::string get_value;
    leveldb::Status s = db->Put(leveldb::WriteOptions(), key, value);
    if (s.ok())
        s = db->Get(leveldb::ReadOptions(), "A", &get_value);

    if (s.ok())
        cout << get_value << endl;
    else
        cout << s.ToString() << endl;

    delete db;
    return 0;
}
```

**步骤2** 编译程序。

**g++ -std=c++11 hello_leveldb.cc -o hello_leveldb -lpthread -lleveldb**

**步骤3** 执行程序。

**./hello_leveldb**

回显内容如下，表示LevelDB能够正常使用：

```
a
```

**----结束**

# 7 安装 Cassandra

## 7.1 安装指导

### 介绍

**简要介绍**

Cassandra是一套开源分布式NoSQL数据库系统。

语言：Java

一句话描述：一套开源分布式NoSQL数据库系统

开源协议：Apache

**建议的版本**

已在鲲鹏云服务器上验证过cassandra-3.11.6版本，并以此版本为例进行说明，请根据实际需要选择版本。

### 环境要求

**云服务器要求**

本文以云服务器KC1实例测试，云服务器配置如表7-1所示。

**表 7-1** 云服务器配置

| 项目 | 说明 |
| --- | --- |
| 规格 | kc1.large.2 \| 4vCPUs \| 8GB |
| 磁盘 | 系统盘：高IO（40GB） |

**操作系统要求**

操作系统要求如表7-2所示。

**表 7-2** 操作系统要求

| 项目 | 版本 | 下载地址 |
| --- | --- | --- |
| CentOS | 7.5 | 在公共镜像中已提供 |
| Kernel | 4.14.0 | 包含在操作系统镜像中。 |

## 配置安装环境

**步骤1** 安装openjdk。

**yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel**

**步骤2** 设置环境变量。

**export JAVA_HOME=/usr/lib/jvm/JAVA-1.8.0-openjdk*aarch64**

**export CLASSPATH=.:${JAVA_HOME}/lib/dt.jar:${JAVA_HOME}/lib/tools.jar**

**步骤3** 下载ant并解压。

**wget -c http://apache.mirrors.spacedump.net//ant/source/apache-ant-1.10.6-src.tar.gz**

**tar -zxvf apache-ant-1.10.6-src.tar.gz**

**步骤4** 下载依赖包解压后放到指定目录。

**wget -c http://apache.mirrors.spacedump.net//commons/net/binaries/commons-net-3.6-bin.tar.gz**

**tar -zxvf commons-net-3.6-bin.tar.gz**

**cp commons-net-3.6/commons-net-3.6.jar ./apache-ant-1.10.6/lib/optional/**

**wget -c https://kent.dl.sourceforge.net/project/jsch/jsch.jar/0.1.55/jsch-0.1.55.jar**

**cp jsch-0.1.55.jar ./apache-ant-1.10.6/lib/optional/**

**wget -c http://apache.mirrors.spacedump.net//xerces/xml-commons/binaries/xml-commons-resolver-1.2.tar.gz**

**tar -zxvf xml-commons-resolver-1.2.tar.gz**

**cp xml-commons-resolver-1.2/resolver.jar ./apache-ant-1.10.6/lib/optional/**

**wget -c http://archive.apache.org/dist/jakarta/regexp/jakarta-regexp-1.5.tar.gz**

**tar -zxvf jakarta-regexp-1.5.tar.gz**

**cp jakarta-regexp-1.5/jakarta-regexp-1.5.jar ./apache-ant-1.10.6/lib/optional/**

**wget -c http://archive.apache.org/dist/jakarta/oro/jakarta-oro-2.0.8.tar.gz**

**tar -zxvf jakarta-oro-2.0.8.tar.gz**

**cp jakarta-oro-2.0.8/jakarta-oro-2.0.8.jar ./apache-ant-1.10.6/lib/optional/**

**步骤5** 安装ant。

**cd apache-ant-1.10.6/**

**sh build.sh -Ddist.dir=./ dist**

**export ANT_HOME=/usr/local/ant**

**sh build.sh install**

**----结束**

### 获取软件包

获取“cassandra-3.11.6”软件包。

**cd /usr/local/src**

**wget -c https://github.com/apache/cassandra/archive/cassandra-3.11.6.tar.gz**

### 安装

**步骤1** 解压软件包。

**tar -zxvf cassandra-3.11.6.tar.gz**

**步骤2** 进入安装目录。

**cd cassandra-cassandra-3.11.6/**

**步骤3** 编译安装。

**/usr/local/ant/bin/ant clean build release**

**----结束**

### 运行和验证

查看cassandra帮助：

**./bin/cassandra -h**

## 7.2 故障排除

### 问题一：启动时报错“Could not create the Java Virtual Machine”

**现象描述**：

启动时报如下信息：

```
The stack size specified is too small, Specify at least 328k
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

**可能原因**：

JVM参数设置中堆的大小设置过小。

**处理步骤**：

可通过调整conf/jvm.options选项后启动：

```
vi conf/jvm.options
# Per-thread stack size.
-Xss512k
```

# 8 移植 Ignite

## 介绍

### 简要介绍

Apache Ignite是一个以内存为中心的分布式数据库、缓存和处理平台，可以在PB级数据中，以内存级的速度进行事务性、分析性以及流式负载的处理。

语言：C/C++

一句话描述：轻量级分布式内存HTAP数据库及计算平台

开源协议：Apache

### 建议的版本

根据实际需要选择版本，本文档以“apache-ignite-2.7.5”为例进行说明。

## 环境要求

### 云服务器要求

本文以KC1实例测试，配置如表8-1所示。

**表 8-1** 云服务器配置

| 项目 | 说明 |
| --- | --- |
| 规格 | kc1.large.2 \| 2vCPUs \| 4GB |
| 磁盘 | 系统盘：高IO（40GB） |

### 操作系统要求

操作系统要求如表8-2所示。

**表 8-2** 操作系统要求

| 项目 | 说明 | 下载地址 |
| --- | --- | --- |
| CentOS | 7.6 | 在公共镜像中已提供。 |
| Kernel | 4.14.0-115 | 在公共镜像中已提供。 |

## 配置编译环境

**步骤1** 安装wget工具。

**yum install wget -y**

**步骤2** 安装OpenJDK。

**yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel**

**步骤3** 设置环境变量。

**export JAVA_HOME=export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.aarch64**

**export CLASSPATH=.:${JAVA_HOME}/lib/dt.jar:${JAVA_HOME}/lib/tools.jar**

**步骤4** 安装Maven。

**wget -c http://apache.mirrors.spacedump.net/maven/maven-3/3.6.1/source/apache-maven-3.6.1-src.tar.gz**

**tar -zxvf apache-maven-3.6.1-src.tar.gz**

**cd apache-maven-3.6.1/**

**mvn install**

**步骤5** 进入新编译的Maven所在的目录。

**cd ..**

**apache-maven-3.6.1/apache-maven/target/apache-maven-3.6.1/bin/mvn -version**

**export MAVEN_HOME=./apache-maven-3.6.1/apache-maven/target/apache-maven-3.6.1**

**----结束**

## 获取源码

获取“apache-ignite-2.7.5”源码包。

**https://ignite.apache.org/download.cgi#sources**提供ignite的源代码压缩包，可以直接下载。

**cd /usr/local/src**

**wget https://archive.apache.org/dist/ignite/2.7.5/apache-ignite-2.7.5-src.zip**

## 编译和安装

**步骤1** 解压软件包。

**cd /usr/local/src**

**unzip apache-ignite-2.7.5-src.zip**

**步骤2** 进入Ignite的安装目录。

**cd apache-ignite-2.7.5-src/**

**步骤3** 编译安装。

**${MAVEN_HOME}/bin/mvn clean package -DskipTests -Prelease,lgp**

**步骤4** 安装Ignite。

**./configure**

**make && make install**

编译后的压缩包在以下路径：

```
apache-ignite-2.7.5-src/targer/bin/apache-ignite-2.7.5-bin.zip
```

**步骤5** 将上一步获得的Ignite压缩包解压到安装目录。

**unzip apache-ignite-2.7.5-bin.zip -d /usr/local/**

**----结束**

## 运行和验证

执行如下命令。

**/usr/local/apache-ignite-2.7.5-bin/bin/ignite.sh**

返回内容如下所示，表示安装已经完成，Ignite启动成功。

```
[root@ecs]# /usr/local/apache-ignite-2.7.5-bin/bin/ignite.sh
started OK
```

# 9 移植 MongoDB

## 9.1 移植指导

### 介绍

**简要介绍**

MongoDB是一个基于分布式文件存储的数据库，由C++语言编写，旨在为Web应用提供可扩展的高性能数据存储解决方案。

一句话描述：NoSQL数据库

语言：C++

开源协议：MIT

**建议的版本**

建议最低版本为“MongoDB-3.6.13”。

### 环境要求

**云服务器要求**

本文以云服务器KC1实例测试，云服务器配置如表9-1所示。

**表 9-1** 云服务器配置

| 项目 | 说明 |
| --- | --- |
| 规格 | kc1.xlarge.2 \| 4vCPUs \| 8GB |
| 磁盘 | 系统盘：高IO（40GB） |

**操作系统要求**

操作系统要求如表9-2所示。

**表 9-2** 操作系统要求

| 项目 | 版本 | 下载地址 |
| --- | --- | --- |
| CentOS | • 版本：7.6<br>• Kernel：4.14.0 | 在公共镜像中已提供。 |
| EulerOS | • 版本：2.8<br>• Kernel：4.19.36 | 在公共镜像中已提供。 |

## 配置编译环境

执行以下命令，安装依赖库。

**sudo yum -y install unzip**

**sudo yum -y install libcurl-devel**

**sudo yum -y install openssl**

**sudo yum -y install openssl-devel**

**sudo yum -y install libxml2-devel**

**sudo yum -y install libxml2**

**sudo yum -y install glibc-static**

**sudo yum -y install libstdc++-static**

**sudo yum -y install lzip**

**sudo yum -y install libffi-devel**

**sudo yum -y install wget**

## 获取源码

- 本文档所测试版本为：MongoDB-3.6.13
- 软件获取路径为：**https://github.com/mongodb/mongo/releases**
- MongoDB官网：**https://www.mongodb.com/**

## 编译和安装

本文以MongoDB-3.6.13为例，下载MongoDB-3.6.13源码，并编译安装。

**步骤1** CentOS系统需执行以下命令，安装gcc7编译环境（OpenEuler可直接进入下一步）：

**sudo yum -y install centos-release-scl**

**mv /etc/yum.repos.d/CentOS-SCLo-scl.repo /etc/yum.repos.d/CentOS-SCLo-scl.repo.ignore**

**sudo yum makecache**

**sudo yum -y install devtoolset-7-gcc***

**步骤2** 执行以下命令，获取较新的Python2.7解释环境，推荐版本为Python2.7.17。

1. 通过源码编译并安装python解释器。

   **cd /usr/local/src**

   **wget https://mirrors.huaweicloud.com/python/2.7.17/Python-2.7.17.tgz**

   **tar zxvf Python-2.7.17.tgz**

   **cd Python-2.7.17**

   **scl enable devtoolset-7 "./configure --prefix=/usr/local --enable-optimizations"**

   **scl enable devtoolset-7 "make -j4"**

   **scl enable devtoolset-7 "make altinstall"**

2. 安装配套的setuptools工具。

   **cd /usr/local/src**

   **wget https://github.com/pypa/setuptools/archive/v41.0.1.zip**

   **unzip v41.0.1.zip**

   **cd setuptools-41.0.1**

   **/usr/local/bin/python2.7 bootstrap.py**

   **/usr/local/bin/python2.7 setup.py install**

3. 安装配套的pip工具。

   **cd /usr/local/src**

   **wget https://github.com/pypa/pip/archive/19.2.2.tar.gz**

   **tar zxvf 19.2.2.tar.gz**

   **cd pip-19.2.2**

   **/usr/local/bin/python2.7 setup.py install**

**步骤3** 执行以下命令，获取MongoDB源码。

**cd /usr/local/src**

**wget https://github.com/mongodb/mongo/archive/r3.6.13.tar.gz**

**步骤4** 执行以下命令，解压包。

**tar -zxvf r3.6.13.tar.gz**

**步骤5** 执行以下命令，进入“mongo-r3.6.13”目录。

**cd mongo-r3.6.13**

**步骤6** 执行以下命令，构建编译环境。

**scl enable devtoolset-7 "/usr/local/bin/pip2 install -r buildscripts/requirements.txt"**

**步骤7** 执行以下命令，切换到gcc7及python2.7.17的编译环境，并编译MongoDB。

**scl enable devtoolset-7 "/usr/local/bin/python2.7 buildscripts/scons.py --prefix=/opt/mongo install MONGO_VERSION=3.6.13 CCFLAGS="-**

**march=armv8-a+crc" --disable-warnings-as-errors --variables-files=etc/scons/propagate_shell_environment.vars -j 4**"

引号中增加的-j参数用于实现多核编译加速，但gcc在多核编译时会消耗大量内存，请根据自己的可用内存大小设定合理的并发度。

**步骤8** 执行以下命令，创建MongoDB的数据库目录。

**mkdir -p /data/db**

可以在此步完成之后，为该目录挂载其他文件系统

**----结束**

## 运行和验证

**步骤1** 安装完成后，启动MongoDB。

1. 执行以下命令，进行MongoDB安装目录。

   **cd /opt/mongo/bin**

2. 执行以下命令，运行MongoDB。

   **./mongod --fork --logpath=/data/db/log.log**

**步骤2** 用客户端测试MongoDB数据库服务。

1. 执行以下命令，连接MongoDB服务端。

   **cd /opt/mongo/bin**

   **./mongo**

   系统回显如下，则连接成功，进入MongoDB后台管理Shell环境。

```
MongoDB shell version v3.6.13
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d7f8e07a-e4b6-41d0-a65a-cd1c92095648") }
MongoDB server version: 3.6.13
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
http://docs.mongodb.org/
Questions? Try the support group
http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-07-05T08:57:25.541+0800 I STORAGE  [initandlisten]
2019-07-05T08:57:25.541+0800 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2019-07-05T08:57:25.541+0800 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten]
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten] ** WARNING: You are running this process as the root user, which is not recommended.
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten]
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten] **          bind to all interfaces. If this
```

```
behavior is desired, start the
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to
disable this warning.
2019-07-05T08:57:25.689+0800 I CONTROL  [initandlisten]
2019-07-05T08:57:25.706+0800 I CONTROL  [initandlisten]
```

2. 执行以下命令，插入数据到MongoDB中。

   **db.runoob.insert({x:10})**

   系统回显如下，表示成功插入数据：

   ```
   WriteResult({ "nInserted" : 1 })
   ```

3. 执行以下命苦，查询插入的数据。

   **db.runoob.find()**

   系统回显如下，表示成功查询到数据:

   ```
   { "_id" : ObjectId("5d1b10c2935172824ee11e2e"), "x" : 10 }
   ```

**----结束**

# 9.2 故障排除

## 编译报“No such file or directory: '/usr/lib/python2.7/site-packages/setuptools-19.6.2-py2.7.egg'”类型错误

**问题描述**：

构建编译环境时，提示出错，如下所示：

```
OSError: [Errno 2] No such file or directory: '/usr/lib/python2.7/site-packages/setuptools-19.6.2-py2.7.egg'
```

**解决方法**：

重新执行下述命令构建编译环境：

**pip2 install -r buildscripts/requirements.txt**

# A 修订记录

| 发布日期 | 修订记录 |
| --- | --- |
| 2020-08-06 | 第二次正式发布。<br>增加Ignite的指导。 |
| 2020-03-18 | 第一次正式发布。 |