后台环境搭建-总

版权说明

对比特项目感兴趣，可以联系这个微信。

代码 & 板书链接

https://gitee.com/bitedu-tech/cpp-chatsystem

基础工具安装：

编辑器安装：

```
C++
dev@dev-host:~/workspace$ sudo apt-get install vim
```

编译器安装：

```
C++
dev@dev-host:~/workspace$ sudo apt-get install gcc g++
```

调试器安装

```
C++
dev@dev-host:~/workspace$ sudo apt-get install gdb
```

项目构建工具安装：

```
C++
sudo apt-get install make cmake
```

传输工具安装：

```
C++
dev@dev-host:~/workspace$ sudo apt-get install lrzsz
```

版本管理工具安装：

```
C++
dev@dev-host:~/workspace$ sudo apt-get install git
```

gflags 框架安装

```C++
dev@dev-host:~/workspace$ sudo apt-get install libgflags-dev
```

gtest 框架安装

```C++
dev@dev-host:~/workspace$ sudo apt-get install libgtest-dev
```

spdlog 框架安装

```C++
dev@dev-host:~/workspace$ sudo apt-get install libspdlog-dev
```

brpc 框架安装

先安装依赖

```C++
dev@dev-host:~/workspace$ sudo apt-get install -y git g++ make
libssl-dev libprotobuf-dev libprotoc-dev protobuf-compiler
libleveldb-dev
```

安装 brpc

```C++
dev@dev-host:~/workspace$ git clone
https://github.com/apache/brpc.git
dev@dev-host:~/workspace$ cd brpc/
dev@dev-host:~/workspace/brpc$ mkdir build && cd build
dev@dev-host:~/workspace/brpc/build$ cmake -
DCMAKE_INSTALL_PREFIX=/usr .. && cmake --build . -j6
dev@dev-host:~/workspace/brpc/build$ make && sudo make install
```

etcd 框架安装

安装 etcd

```C++
dev@dev-host:~/workspace$ sudo apt-get install etcd
dev@dev-host:~/workspace$ sudo systemctl start etcd
dev@dev-host:~/workspace$ sudo systemctl enable etcd
```

安装 etcd 客户端 api：etcd-cpp-apiv3

```C++
dev@dev-host:~/workspace$ sudo apt-get install libboost-all-dev
dev@dev-host:~/workspace$ protobuf-compiler-grpc
dev@dev-host:~/workspace$ sudo apt-get install libgrpc-dev
libgrpc++-dev
dev@dev-host:~/workspace$ sudo apt-get install libcpprest-dev
dev@dev-host:~/workspace$ git clone https://github.com/etcd-cpp-
apiv3/etcd-cpp-apiv3.git
dev@dev-host:~/workspace$ cd etcd-cpp-apiv3
dev@dev-host:~/workspace$ mkdir build && cd build
dev@dev-host:~/workspace$ cmake .. -DCMAKE_INSTALL_PREFIX=/usr
dev@dev-host:~/workspace$ make -j$(nproc) && sudo make install
```

elasticsearch 框架安装：

安装 es

```C++
dev@dev-host:~/workspace$ curl -s
https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --
no-default-keyring --keyring gnupg-
ring:/etc/apt/trusted.gpg.d/icsearch.gpg --import
dev@dev-host:~/workspace$ echo "deb
https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo
tee /etc/apt/sources.list.d/elasticsearch.list
dev@dev-host:~/workspace$ sudo apt update
dev@dev-host:~/workspace$ sudo apt-get install
elasticsearch=7.17.21
```

安装中文分词插件

```C++
dev@dev-host:~/workspace$ sudo
/usr/share/elasticsearch/bin/elasticsearch-plugin install
https://get.infini.cloud/elasticsearch/analysis-ik/7.17.21
```

修改 es 配置监听地址，需要启用外部访问

```
C++
dev@dev-host:~/workspace$ sudo vim
/etc/elasticsearch/elasticsearch.yml
56 network.host: 0.0.0.0
57 #
58 # By default Elasticsearch listens for HTTP traffic on the
first free port     it
59 # finds starting at 9200. Set a specific HTTP port here:
60 #
61 http.port: 9200
```

启动 es 并设置开机启动

```
C++
dev@dev-host:~/workspace$ sudo systemctl restart elasticsearch
dev@dev-host:~/workspace$ sudo systemctl enable elasticsearch
dev@dev-host:~/workspace$ sudo systemctl status elasticsearch
```

安装 kibana（es 页面访问工具）,并进行配置

```
C++
dev@dev-host:~/workspace$ sudo apt install kibana
dev@dev-host:~/workspace$ sudo vim /etc/kibana/kibana.yml
2  server.port: 5601
7  server.host: "0.0.0.0"
32 elasticsearch.hosts: ["http://localhost:9200"]
dev@dev-host:~/workspace$ sudo systemctl start kibana
dev@dev-host:~/workspace$ sudo systemctl enable kibana
dev@dev-host:~/workspace$ sudo systemctl status kibana
```

通过 kibana 访问 es 实现索引创建测试：http://192.168.65.138:5601/

```
C++
POST /user/_doc
{
    "settings" : {
        "analysis" : {
            "analyzer" : {
                "ik" : {
                    "tokenizer" : "ik_max_word"
                }
            }
```

```
        }
    },
    "mappings" : {
        "dynamic" : true,
        "properties" : {
            "nickname" : {
                "type" : "text",
                "analyzer" : "ik_max_word"
            },
            "user_id" : {
                "type" : "text",
                "analyzer" : "ik_max_word"
            },
            "phone" : {
                "type" : "text",
                "analyzer" : "ik_max_word"
            },
            "description" : {
                "type" : "text",
                "analyzer" : "ik_max_word"
            },
            "avatar_id" : {
                "type" : "text",
                "analyzer" : "ik_max_word"
            }
        }
    }
}
```

es 客户端 api 安装

```
C++
dev@dev-host:~/workspace$ sudo apt-get install libmicrohttpd-dev
dev@dev-host:~/workspace$ git clone
https://github.com/seznam/elasticlient.git
dev@dev-host:~/workspace$ cd elasticlient
dev@dev-host:~/workspace$ git submodule update --init --recursive
dev@dev-host:~/workspace$ mkdir build && cd build
dev@dev-host:~/workspace$ cmake -DCMAKE_INSTALL_PREFIX=/usr ..
dev@dev-host:~/workspace$ make && sudo make install
```

cpp-httplib 框架安装：

```C++
dev@dev-host:~/workspace$ git clone
https://github.com/yhirose/cpp-httplib.git
```

websocketpp 框架安装：

这个库其实在前边安装工具的时候已经当作依赖项被安装了，不过这不妨碍再安装一次

查看是否存在：`ls /usr/include/websocketpp/` 即可

```C++
dev@dev-host:~/workspace$ ls /usr/include/websocketpp/
base64       connection.hpp       frame.hpp       random
client.hpp   connection_base.hpp  http            roles
close.hpp    endpoint.hpp         impl            server.hpp
common       endpoint_base.hpp    logger          sha1
concurrency  error.hpp            message_buffer  transport
config       extensions           processors      uri.hpp
```

安装：

```C++
dev@dev-host:~/workspace$ sudo apt-get install libwebsocketpp-dev
```

redis 安装

```C++
dev@dev-host:~/workspace$ sudo apt install redis -y
```

配置：

修改 `/etc/redis/redis.conf`，支持远程连接

- 修改 `bind 127.0.0.1` 为 `bind 0.0.0.0`
- 修改 `protected-mode yes` 为 `protected-mode no`

```C++
dev@dev-host:~/workspace$ sudo vim /etc/redis/redis.conf
# bind 127.0.0.1    # 注释掉这行
bind 0.0.0.0        # 添加这行
```

```
protected-mode no  # 把 yes 改成 no
```

启动

```
C++
dev@dev-host:~/workspace$ sudo systemctl start redis-server
dev@dev-host:~/workspace$ sudo systemctl enable redis-server
```

安装客户端 SDK

```
C++
dev@dev-host:~/workspace$ sudo apt install libhiredis-dev
dev@dev-host:~/workspace$ git clone
https://github.com/sewenew/redis-plus-plus.git
dev@dev-host:~/workspace$ cd redis-plus-plus
dev@dev-host:~/workspace$ mkdir build && cd build
dev@dev-host:~/workspace$ cmake -DCMAKE_INSTALL_PREFIX=/usr ..
dev@dev-host:~/workspace$ make && sudo make install
```

ODB 安装

安装 build2：过程可能需要 1~3 小时

```
C++
dev@dev-host:~/workspace$ curl -sSfO
https://download.build2.org/0.17.0/build2-install-0.17.0.sh
dev@dev-host:~/workspace$ sh build2-install-0.17.0.sh
```

安装 odb-compiler

```
C++
dev@dev-host:~/workspace$ #注意这里的 gcc-11 需要根据你自己版本而定
dev@dev-host:~/workspace$ sudo apt-get install gcc-11-plugin-dev
dev@dev-host:~/workspace$ mkdir odb-build && cd odb-build
dev@dev-host:~/workspace/odb-build$ bpkg create -d odb-gcc-N cc
\
  config.cxx=g++                \
  config.cc.coptions=-O3        \
  config.bin.rpath=/usr/lib \
  config.install.root=/usr/  \
  config.install.sudo=sudo
dev@dev-host:~/workspace/odb-build$ cd odb-gcc-N
dev@dev-host:~/workspace/odb-build/odb-gcc-N$ bpkg build
```

```
odb@https://pkg.cppget.org/1/beta
dev@dev-host:~/workspace/odb-build/odb-gcc-N$ bpkg test odb
test odb-2.5.0-b.25+1/tests/testscript{testscript}
tested odb/2.5.0-b.25+1
dev@dev-host:~/workspace/odb-build/odb-gcc-N$ bpkg install odb
dev@dev-host:~/workspace/odb-build/odb-gcc-N$ odb --version
bash: /usr/bin/odb: No such file or directory
#如果报错了，找不到 odb，那就在执行下边的命令
dev@dev-host:~/workspace/odb-build/odb-gcc-N$ sudo echo 'export
PATH=${PATH}:/usr/local/bin' >> ~/.bashrc
dev@dev-host:~/workspace/odb-build/odb-gcc-N$ export
PATH=${PATH}:/usr/local/bin
dev@dev-host:~/workspace/odb-build/odb-gcc-N$ odb --version
```

安装 ODB 运行时库

```
C++
dev@dev-host:~/workspace/odb-build/odb-gcc-N$ cd ..
dev@dev-host:~/workspace/odb-build$ bpkg create -d libodb-gcc-N cc
\
  config.cxx=g++                \
  config.cc.coptions=-O3        \
  config.install.root=/usr/  \
  config.install.sudo=sudo
dev@dev-host:~/workspace/odb-build$ cd libodb-gcc-N
dev@dev-host:~/workspace/odb-build/libodb-gcc-N$ bpkg add
https://pkg.cppget.org/1/beta
dev@dev-host:~/workspace/odb-build/libodb-gcc-N$ bpkg fetch
dev@dev-host:~/workspace/odb-build/libodb-gcc-N$ bpkg build libodb
dev@dev-host:~/workspace/odb-build/libodb-gcc-N$ bpkg build
libodb-mysql
```

安装 mysql 和客户端开发包

```
C++
dev@dev-host:~/workspace$ sudo apt install mysql-server
dev@dev-host:~/workspace$ sudo apt install -y libmysqlclient-dev
```

配置 mysql

```
C++
sudo vim /etc/my.cnf  或者  /etc/mysql/my.cnf 有哪个修改哪个就行
#添加以下内容
[client]
```

```
default-character-set=utf8
[mysql]
default-character-set=utf8
[mysqld]
character-set-server=utf8
bind-address = 0.0.0.0
```

修改 root 用户密码

```
C++
dev@bite:~$ sudo cat /etc/mysql/debian.cnf
# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host     = localhost
user     = debian-sys-maint
password = UWcn9vY0NkrbJMRC
socket   = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
host     = localhost
user     = debian-sys-maint
password = UWcn9vY0NkrbJMRC
socket   = /var/run/mysqld/mysqld.sock
dev@bite:~$ sudo mysql -u debian-sys-maint -p
Enter password: #这里输入上边第 6 行看到的密码
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH
mysql_native_password BY 'xxxxxx';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql> quit
```

重启 mysql，并设置开机启动

```
C++
dev@dev-host:~/workspace$ sudo systemctl restart mysql
dev@dev-host:~/workspace$ sudo systemctl enable mysql
```

安装 ODB 的 boost profile 库

```
C++
dev@dev-host:~/workspace/odb-build/libodb-gcc-N$ bpkg build
```

```
libodb-boost
```

总体打包安装

```
C++
dev@dev-host:~/workspace/odb-build/libodb-gcc-N$ bpkg install --
all --recursive
```

RabbbitMQ 安装

```
C++
dev@dev-host:~/workspace$ sudo apt install rabbitmq-server
```

命令配置

```
C++
# 启动服务
sudo systemctl start rabbitmq-server
 # 查看服务状态
sudo systemctl status rabbitmq-server

# 安装完成的时候默认有个用户 guest ，但是权限不够，要创建一个
administrator 用户，才可以做为远程登录和发表订阅消息：
 #添加用户
sudo rabbitmqctl add_user root 123456
 #设置用户 tag
sudo rabbitmqctl set_user_tags root administrator
 #设置用户权限
sudo rabbitmqctl set_permissions -p / root "." "." ".*"

# RabbitMQ 自带了 web 管理界面,执行下面命令开启
sudo rabbitmq-plugins enable rabbitmq_management
```

页面访问：访问 webUI 界面， 默认端口为 15672

安装客户端 SDK：

- C 语言库：https://github.com/alanxz/rabbitmq-c
- C++库：https://github.com/akalend/amqpcpp (弃用)
- C++库: https://github.com/CopernicaMarketingSoftware/AMQP-CPP/tree/master

我们这里使用 AMQP-CPP 库来编写客户端程序。

```C++
sudo apt-get install librabbitmq-dev
git clone https://github.com/CopernicaMarketingSoftware/AMQP-CPP.git
cd AMQP-CPP/
mkdir build && cd build
cmake -DCMAKE_INSTALL_PREFIX=/usr ..
make  && sudo make install
sudo apt install libev-dev     #libev 网络库组件
```