谈信项目实施与部署详细教程

# 一、谈信结构简述

## 1、聊天-im模块

### （1）仓库地址

https://github.com/txchat/im.git

### （2）项目运行说明

#### 打包环境

go1.19

docker、docker-compose

make

#### 运行环境

docker、docker-compose

#### 打包与运行

1、编译和go依赖构建：go mod tidy

2、初始化docker环境：make init-compose

3、打包和启动程序：make docker-compose-up

执行程序位置：./run\_compose

说明：第2步会构建自定义容器镜像，生产环境中可将镜像导出使用

## 2、聊天-dtalk模块

### （1）仓库地址

https://github.com/txchat/dtalk.git

### （2）项目运行说明

#### 打包环境

go1.19

docker、docker-compose

make

#### 运行环境

docker、docker-compose

#### 打包与运行

1、编译和go依赖构建：go mod tidy

2、初始化docker环境：make init-compose

3、打包和启动程序：make docker-compose-up

执行程序位置：./run\_compose

说明：第2步会构建自定义容器镜像，生产环境中可将镜像导出使用

#### 配置说明

1、需要自建Bucket与鉴权用户后，修改oss模块配置

2、需要修改节点相关配置

可根据注释说明进行配置修改。

## 3、聊天-im-util模块

### （1）仓库地址

https://github.com/txchat/im-util.git

### （2）项目运行说明

#### 打包环境

go1.19

make

#### 运行环境

二进制可执行程序

#### 打包与运行

##### 第一步：user创建程序

操作目录：cd protocol

打包：make build

打包文件位置：cd target

#执行命令创建测试地址 此步骤会创建一个users.txt文件

./imall\_linux\_amd64 wallet gen -n 2 -w users.txt --ws ,

##### 第二步：验证程序

操作目录：cd pressure

打包：make build

打包文件位置：cd target

将第一步生成文件拷贝到当前目录：cp ../../protocol/target/users.txt .

##### 第三步：执行验证命令

./pressure\_linux\_amd64 pre -s "localhost:3102" -u2 -t 5s

./pressure\_linux\_amd64 ana

成功输出结果：message tranport success count: 8 -- failed count: 0

## 4、区块链管理后台

php后端（包含go程序），区块链节点运行程序

### （1）资源地址

生成环境代码移植，包括php代码和go编译可执行文件

### （2）项目运行说明

#### 运行环境

Php7.3

nginx

Redis 4.0.12

Mysql 5.7

#### 运行

nginx配置php运行配置文件，重启nginx使配置生效即可

Nginx –s reload

## 5、nginx前端与接口代理

### （1）静态资源

暂无

### （2）nginx配置

按照模板修改相关接口IP地址即可

## 6、红包与转账模块

### （1）资源地址

开源包：

### （2）项目运行说明

#### 运行环境

docker、docker-compose、make、jq

#### 运行

初始化构建：make build

运行：make up

测试：make test

# 二、项目打包与资源准备

## 1、打包环境搭建

概述：ubuntu22.04、go1.19 、 docker、docker-compose、make

### （1）go环境

#### 下载资源包

wget <https://golang.google.cn/dl/go1.19.linux-amd64.tar.gz>

或者从：https://golang.google.cn/dl/下载后上传到打包服务器

#### 解压到/usr/local/

tar -zxvf go1.19.linux-amd64.tar.gz -C /usr/local/

#### 配置环境变量

export PATH=$PATH:/usr/local/go/bin

说明：该指令对当前窗口临时生效，窗口关闭后失效，如需长期使用go指令，可配置至系统环境变量/etc/profile

#### 查询go版本

go version

### （2）docker环境

#### 安装docker

apt install docker.io

#### 安装docker-compose

apt install docker-compose

#### 查询版本

docker version

docker-compose version

### （3）make指令

#### 安装make

apt install make

## 2、程序编译与打包

### （1）im模块

#### 克隆代码到本地

git clone https://github.com/txchat/im.git

#### 切换分支

cd ./im/

git checkout v0.0.3

git branch

#### go依赖构建

go mod tidy

#### 打包与初始化

make init-compose

生成docker-compose容器配置文件./run\_compose

#### 资源导出

mv run\_compose run\_compose-im #修改文件名，避免重名问题

zip -q -r run\_compose-im.zip run\_compose-im #压缩文件夹

docker save -o txchat-logic.tar txchat-logic:v0.0.3 #导出自建镜像

docker save -o txchat-comet.tar txchat-comet:v0.0.3 #导出自建镜像

run\_compose-im.zip/txchat-logic.tar/txchat-comet.tar三个文件导出备用

### （2）dtalk模块

#### 克隆代码到本地

git clone https://github.com/txchat/dtalk.git

#### 切换分支

cd ./dtalk/

git checkout v0.1.1

git branch

#### go依赖构建

go mod tidy

#### 打包与初始化

make init-compose

生成docker-compose容器配置文件./run\_compose

#### 资源导出

mv run\_compose run\_compose-dtalk #修改文件名，避免重名问题

zip -q -r run\_compose-dtalk.zip run\_compose-dtalk #压缩文件夹

docker save -o txchat-answer.tar txchat-answer:v0.1.1

docker save -o txchat-backend.tar txchat-backend:v0.1.1

docker save -o txchat-backup.tar txchat-backup:v0.1.1

docker save -o txchat-call.tar txchat-call:v0.1.1

docker save -o txchat-device.tar txchat-device:v0.1.1

docker save -o txchat-discovery.tar txchat-discovery:v0.1.1

docker save -o txchat-gateway.tar txchat-gateway:v0.1.1

docker save -o txchat-generator.tar txchat-generator:v0.1.1

docker save -o txchat-group.tar txchat-group:v0.1.1

docker save -o txchat-offline-push.tar txchat-offline-push:v0.1.1

docker save -o txchat-oss.tar txchat-oss:v0.1.1

docker save -o txchat-pusher.tar txchat-pusher:v0.1.1

docker save -o txchat-store.tar txchat-store:v0.1.1

run\_compose-dtalk.zip/txchat-answer.tar/txchat-backend.tar/txchat-backup.tar/txchat-call.tar/txchat-device.tar/txchat-discovery.tar/txchat-gateway.tar/txchat-generator.tar/txchat-group.tar/txchat-offline-push.tar/txchat-oss.tar/txchat-pusher.tar/txchat-store.tar共14个文件导出备用

### （3）im-util模块

#### 克隆代码到本地

git clone https://github.com/txchat/im-util.git

#### 切换分支

cd ./im-util/

git checkout v0.1.0

git branch

#### 打包与初始化

cd ./protocol/ #切换目录

make build #编译与打包

cd ./target/ #打包可执行文件目录

生成可执行文件imall\_linux\_amd64导出备用

cd ../../pressure/ #切换目录

make build #编译与打包

cd ./target/ #打包可执行文件目录

生成可执行文件pressure\_linux\_amd64导出备用

#### 资源导出

将生成的可执行文件imall\_linux\_amd64/pressure\_linux\_amd64共2个文件导出备用

### （4）php模块

使用本地资源包，包括php程序、go程序、节点运行程序

### （5）nginx前端模块

使用本地资源包，包括php-nginx配置文件、聊天服务nginx配置文件

### （6）红包与转账

使用开源包

# 三、部署与实施

## 1、部署结构

可以按3台服务器去部署实施：

服务器1：4核8G 200G数据盘，部署区块链节点+聊天模块；

服务器2：2核4G 100G数据盘，部署php+nginx模块；

服务器3：2核4G 100G数据盘，部署红包与转账模块；

服务器:4：2核4G或2核2G 40G系统盘，部署nginx接口代理，域名指向服务器；

网络：四台服务器安全组独立，内网互通。

数据盘默认挂载：/data/

## 2、区块链节点部署

上传文件paraChain.zip至/data/

解压文件：unzip paraChain.zip

进入目录：cd /data/paraChain/

修改平行链配置：vim ./chatpro.para.toml

修改内容：

# 平行链title，用来唯一标识一条平行链

Title="user.p.xxx." #xxx由客户确定“平行链名称”

···

CoinSymbol="para" #para为代币名称，客户指定

···

[consensus]

···

genesisBlockTime=1739721600 #时间戳，取2025-02-17 00:00，在取主链高度时间之前即可

genesis="1BmFaA6unrCFYEW..." #这里为钱包超级管理员地址，用户自行创建填入

···

[consensus.sub.para]

# 主链rpc地址，根据主链节点的具体情况修改，无自建主链节点无需修改

ParaRemoteGrpcClient="cloud.bityuan.com"

#主链指定高度的区块开始同步

startHeight = 37524500 #取自2025-02-17 02:00，无特别要求可不修改

#创世地址额度，无特别要求可不修改

genesisAmount=100000000

···

[exec.sub.relay]

genesis="1BmFaA6unrCFYEW..." #同超管地址

[exec.sub.manage]

superManager=["1BmFaA6unrCFYEW..."] #同超级管理员地址

[exec.sub.token]

tokenApprs=["1BmFaA6unrCFYEW..."] #同超级管理员地址

启动节点

nohup ./chain33 -f chatpro.para.toml >/dev/null 2>&1 &

查询节点高度

./chain33-cli --rpc\_laddr="http://127.0.0.1:8901" block last\_header

## 3、聊天模块部署

### （1）环境与资源准备

安装docker：apt install docker.io

安装docker-compose：apt install docker-compose

修改docker容器存储位置：

建议在安装docker和docker-compose安装完成时就修改存储位置

首先停止docker服务：

sudo systemctl stop docker

创建或编辑配置文件（默认没有这个配置文件）：

sudo nano /etc/docker/daemon.json

添加内容：

{

"data-root": "/data/docker/data",

"storage-driver": "overlay2",

"storage-opts": [

"overlay2.override\_kernel\_check=true"

]

}

这里/data/docker/data为指定的新的容器和镜像存储目录，storage-driver和storage-opts可以根据你的需求进行配置。

移动或拷贝现有数据到新目录：

sudo mv /var/lib/docker /data/docker/data

Sudo cp –R /var/lib/docker /data/docker/data

重新启动docker服务：

sudo systemctl start docker

上传资源文件：

将im、dtalk、im-util导出所有资源文件上传至服务器/data目录

导入2+13共计15个自建镜像：docker load < tar镜像文件名

例：docker load < txchat-comet.tar

解压两个zip压缩文件

unzip run\_compose-im.zip

unzip run\_compose-dtalk.zip

### （2）启动im模块

cd ./run\_compose-im/

docker-compose -f components.compose.yaml -f service.compose.yaml up -d

### （3）启动dtalk模块

cd ./run\_compose-dtalk/

docker-compose -f components.compose.yaml -f service.compose.yaml up -d

如有容器启动失败，可手动重启该容器：docker restart 容器名

例：docker restart txchat-nginx

### （4）im-util验证

创建文件夹：mkdir /data/im-util

移动可执行文件到该文件夹：

mv ./imall\_linux\_amd64 /data/im-util

mv ./pressure\_linux\_amd64 /data/im-util

进入目录：cd /data/im-util

按顺序执行指令：

./imall\_linux\_amd64 wallet gen -n 2 -w users.txt --ws ,

./pressure\_linux\_amd64 pre -s "localhost:3102" -u2 -t 5s

./pressure\_linux\_amd64 ana

输出类似结果 message tranport success count: 8 -- failed count: 0 表示成功

### （5）配置修改

#### 存储桶配置

##### 创建存储桶

根据minio容器开放服务器9001端口，

浏览器访问：http://公网IP:9001/login

登录用户名和密码可见：vim /data/run\_compose-dtalk/.env 文件中

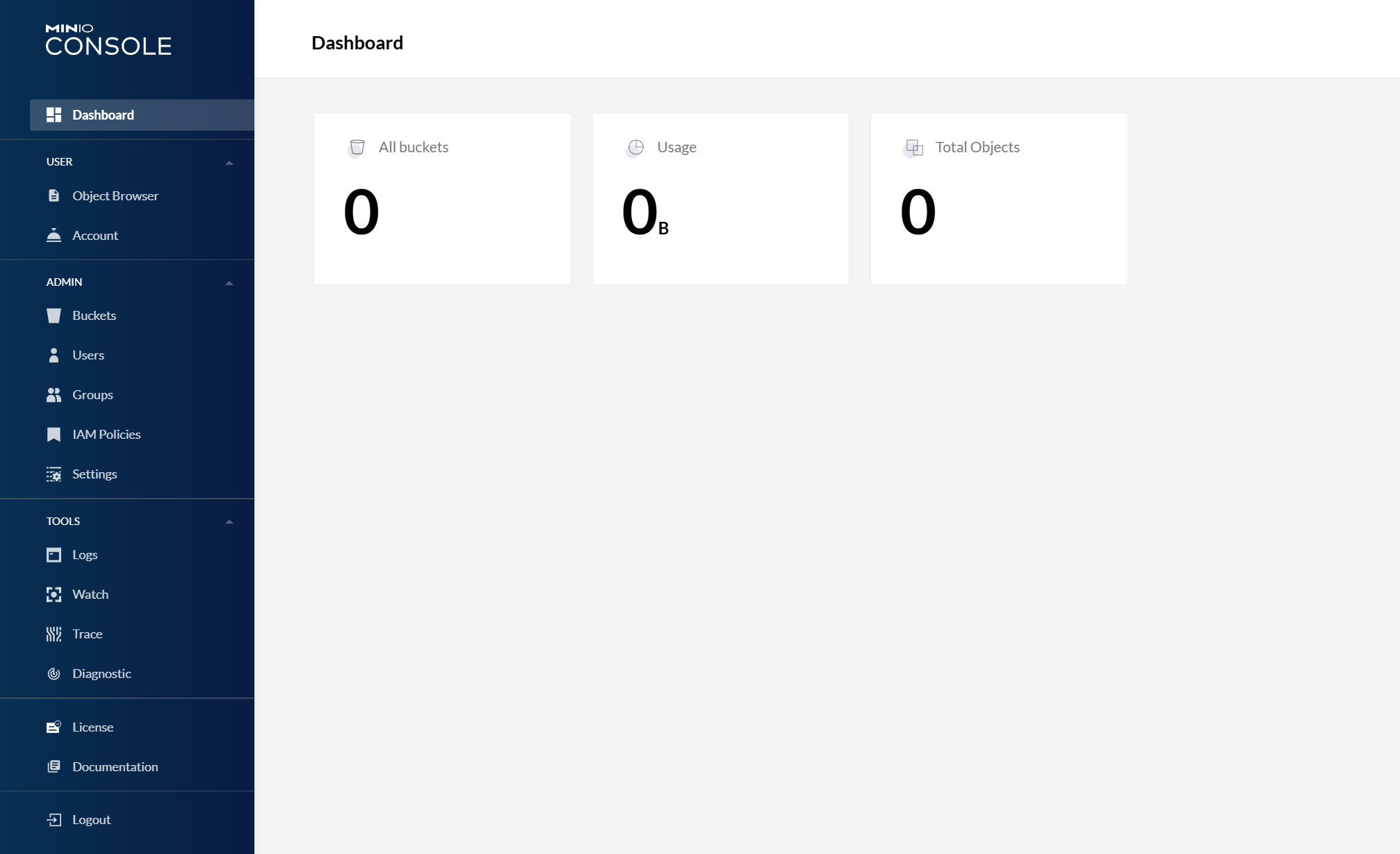
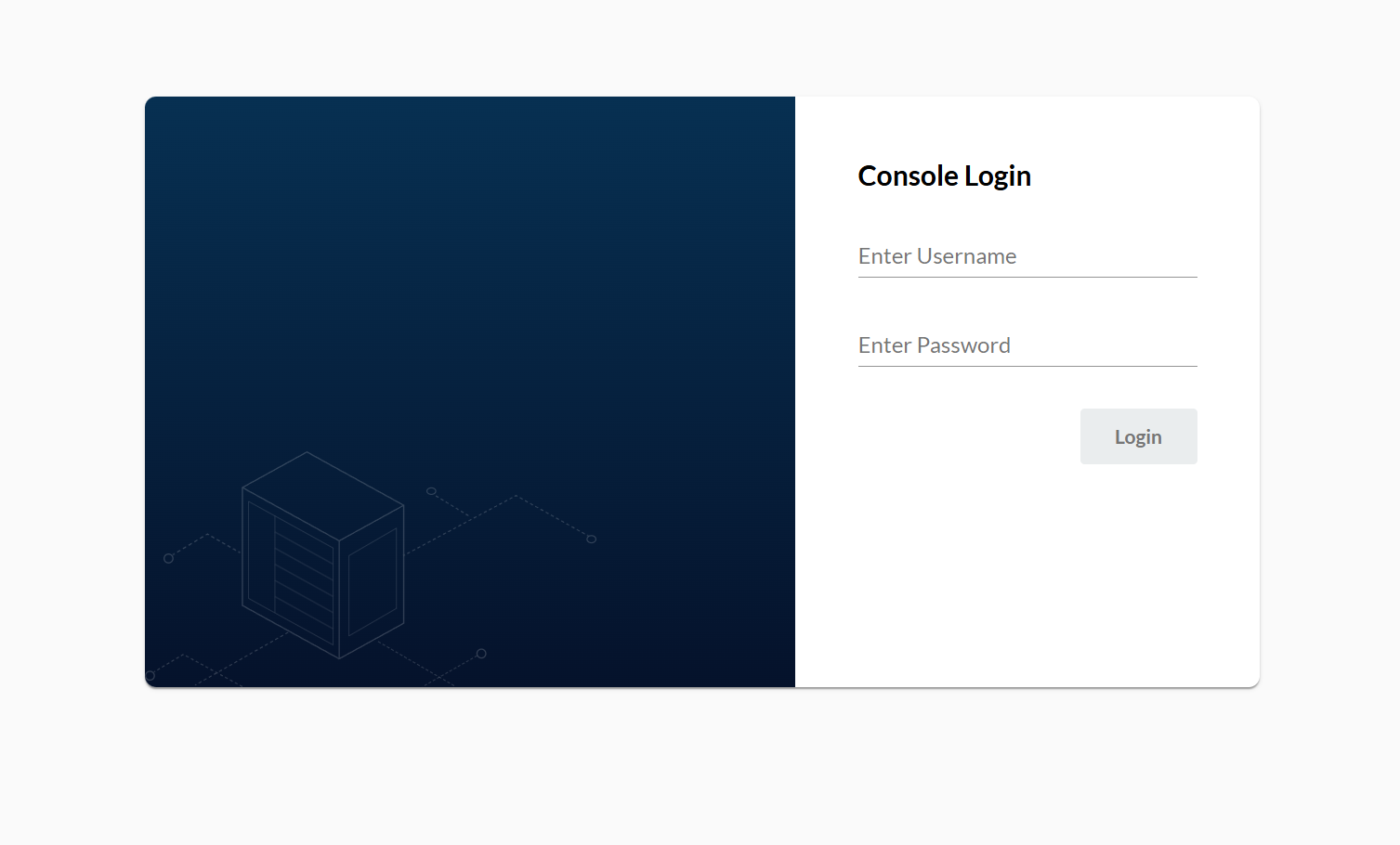
# minio超级用户账户

MINIO\_ROOT\_USER=minio

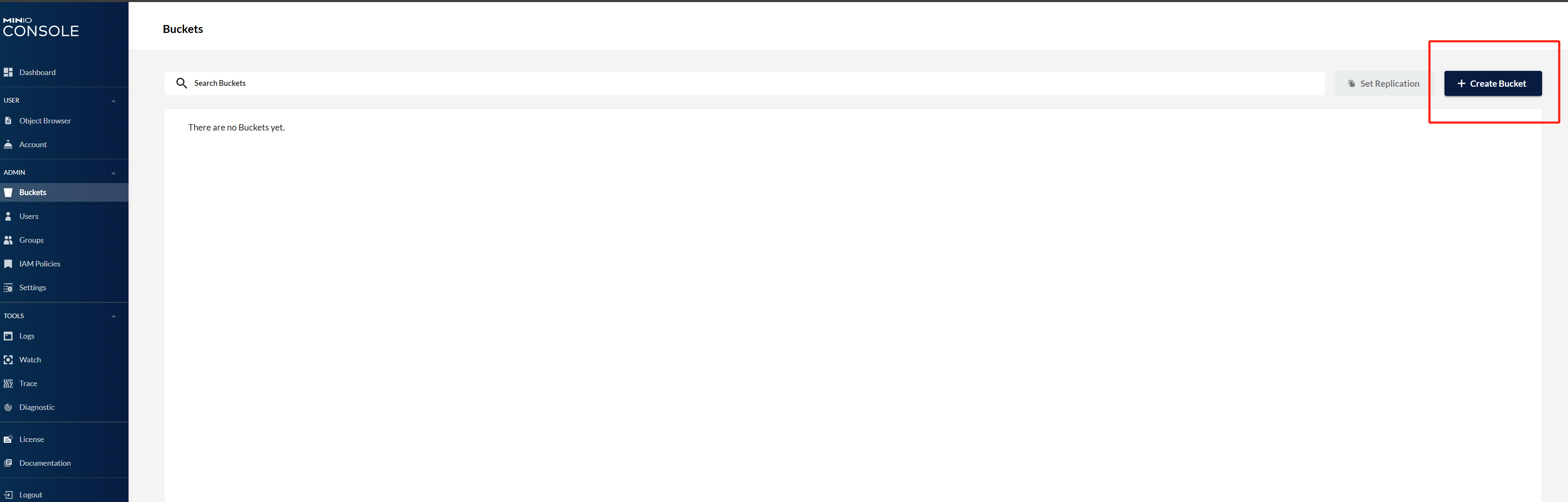
# minio超级用户密码

MINIO\_ROOT\_PASSWORD=JKBjqtOeNfx1ty

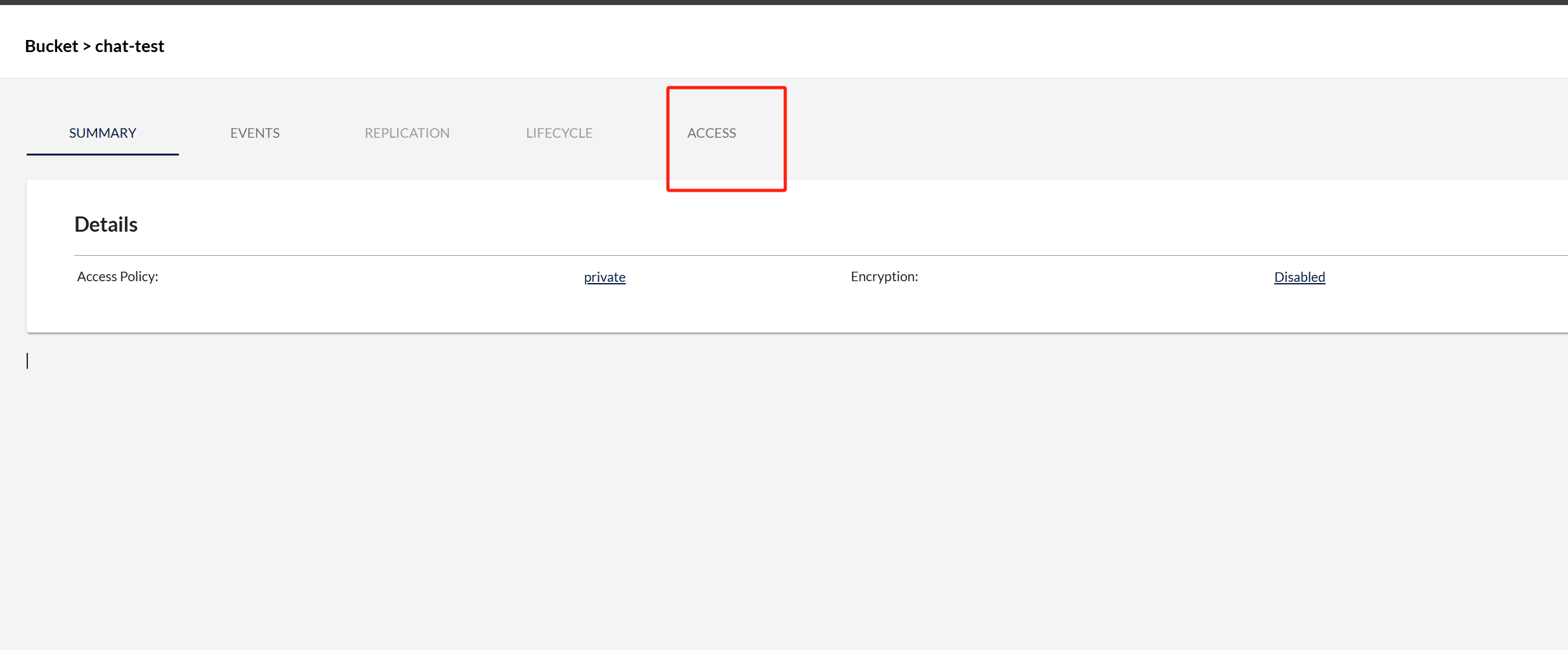
登录页面 输入用户名密码进行登录，登录成功后如下



点击左侧栏的Buckets 进行桶创建

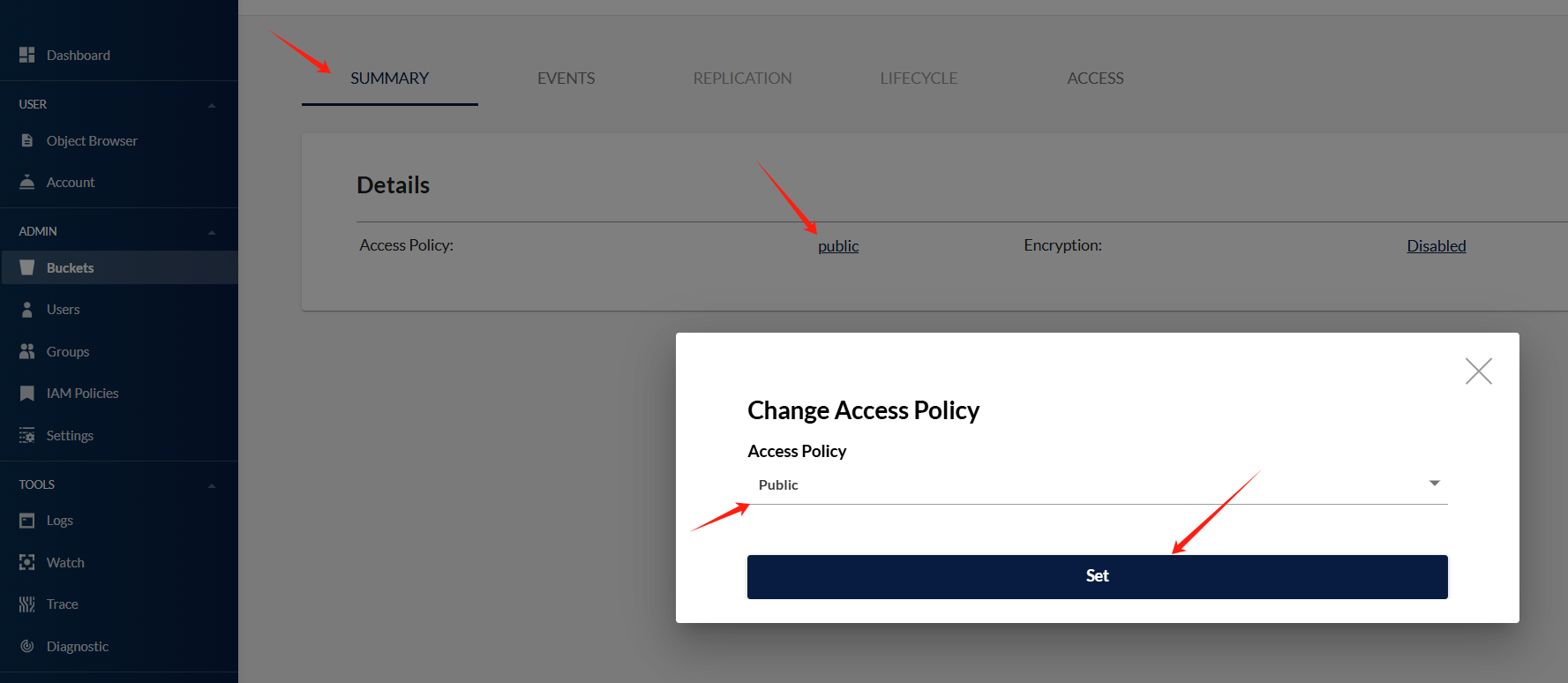


填写Bucket Name ，创建好之后选择创建的bucket ,进入ACCESS ,设置访问权限

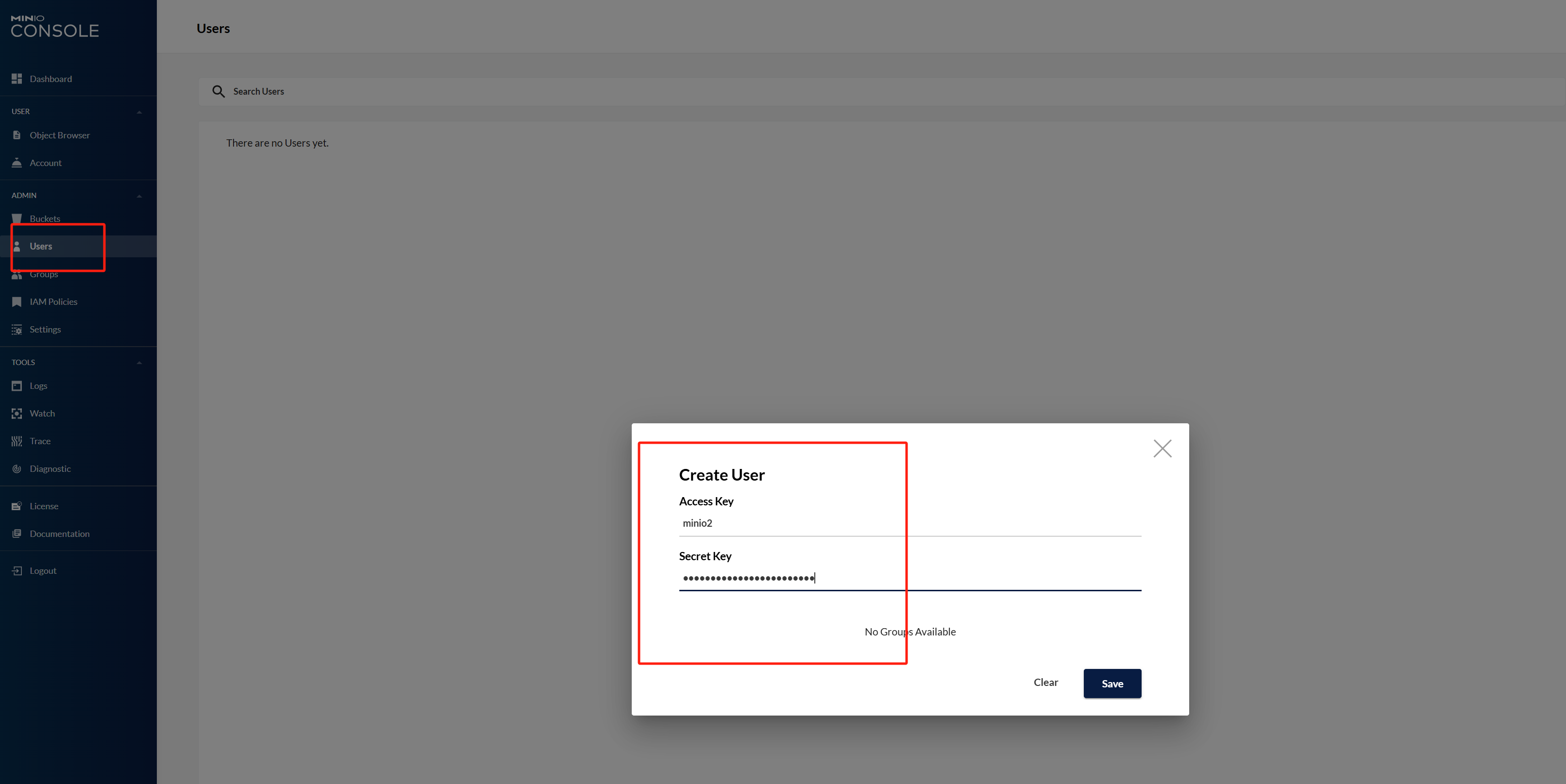


点击readwrite 然后保存

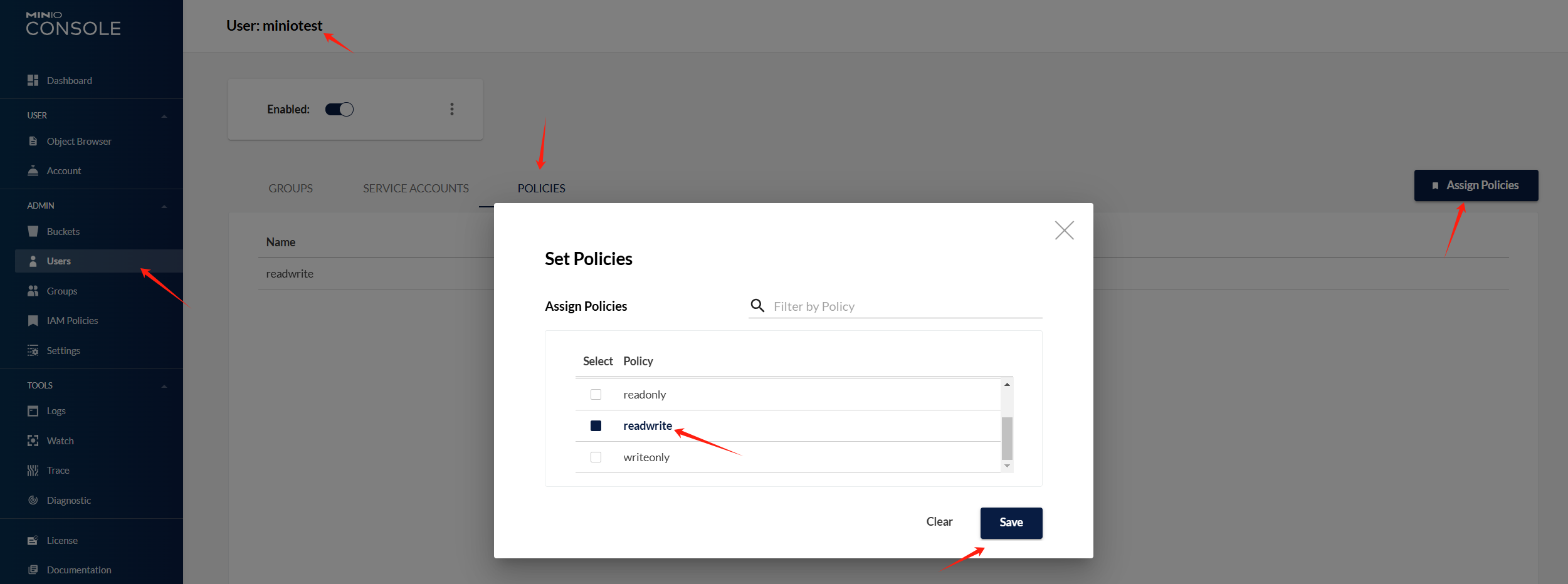
再在该页面进入SUMMARY，设置Access Policy为public，点击set保存



创建用户minio2 ，名字和密码可自定义，名字和密码本地保存备用



用户创建后，点击创建的用户名，选择“POLICIES”功能选项，点击最右边“Assign Policies”标签，设置用户权限为可读可写“readwrite”，点击“Save”保存即可



##### 修改oss模块配置

进入oss容器：docker exec -it txchat-oss-node1 /bin/sh

编辑容器内配置文件：vi /etc/txchat-oss/config/oss.toml

修改内容：

AccessKeyId="minio2" #自创建的用户名

AccessKeySecret="JKUsUHgD+hV" #自创建用户名对应的密码

Policy = '''{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:\*"

],

"Resource": [

"arn:aws:s3:::\*"

]

}

]

}'''

#Bucket 为你在浏览器通过网页创建的bucket的名称

Bucket = "chat-test"

PublicUrl ="http://服务器公网IP:9000" #需要开放9000端口

修改后保存文件，退出容器

重启oss容器：docker restart txchat-oss-node1

#### 聊天节点配置

##### 修改discovery模块

进入discovery容器：docker exec -it txchat-discovery-node1 /bin/sh

编辑容器内配置文件：vi /etc/txchat-discovery/config/oss.toml

修改内容：

[[CNodes]]

name="default"

#即聊天节点，域名的方式部署 类似 https://chat.xxxxx.com 中

address="https://域名xxxx.com"

[[DNodes]]

name="default"

#address 为平行链 即通讯录功能的节点

address="http://平行链的IP:8901"

保存修改，退出容器

重启容器：docker restart txchat-discovery-node1

##### 修改backend模块

进入backend容器：docker exec -it txchat-backend-node1 /bin/sh

编辑容器内配置文件：vi /etc/txchat-backend/config/oss.toml

修改内容：

[chain33Client]

BlockChainAddr="ip:8902" # changeme (钱包查询服务的 rpc 端口)

Title="user.p.chatxxx." # 平行链的title

保存修改，退出容器

重启容器：docker restart txchat-backend-node1

##### 修改gateway模块

进入gateway容器：docker exec -it txchat-gateway-node1 /bin/sh

编辑容器内配置文件：vi /etc/txchat-gateway/config/oss.toml

修改内容：

[[modules]]

Name = "wallet"

IsEnabled = true

EndPoints = ["https://IP:8083"] #changeme(1.红包服务http服务地址,2.钱包服务http 服务地址)

保存修改，退出容器

重启容器：docker restart txchat-gateway-node1

## 4、php模块部署

### 环境准备

#### redis安装

包下载地址：https://download.redis.io/releases/?\_gl=1\*hucdil\*\_gcl\_au\*ODg3ODY4NTI3LjE3Mzk0MjkzNzk.

下载指定版本的包：wget <https://download.redis.io/releases/redis-4.0.12.tar.gz>

解压到指定目录：tar -zvxf redis-4.0.12.tar.gz -C /usr/local/src/

切换到指定目录：cd /usr/local/src/redis-4.0.12/

安装必要插件与指令：apt install make

apt install gcc

编译与安装redis：make -j 16 & make install

修改配置：vim /usr/local/src/redis-4.0.12/redis.conf

#更换绑定

#将bind 127.0.0.1 更换为本机IP，例如：18.192.52.199

bind 18.192.52.199

#关闭保护模式

protected-mode no

# 设置密码

requirepass zidingyi

# 指定日志路径

logfile "/var/log/redis.log"

设置开机启动

创建服务文件:vim /usr/lib/systemd/system/redis.service

#文件内容

[Unit]

Description=Redis Server

After=network.target

[Service]

ExecStart=/usr/local/src/redis-4.0.12/src/redis-server /usr/local/src/redis-4.0.12/redis.conf --daemonize no

ExecStop=/usr/local/src/redis-4.0.12/src/redis-cli -p 6379 -a "kakaxi" shutdown

Restart=always

[Install]

WantedBy=multi-user.target

#设置Redis服务开机启动

systemctl enable redis

#启动Redis服务

systemctl start redis

#### mysql安装

下载安装包与依赖：

wget <https://downloads.mysql.com/archives/get/p/23/file/mysql-server_5.7.34-1ubuntu18.04_amd64.deb-bundle.tar>

wget <http://archive.ubuntu.com/ubuntu/pool/universe/m/mecab/libmecab2_0.996-1.2ubuntu1_amd64.deb>

wget <http://archive.ubuntu.com/ubuntu/pool/main/liba/libaio/libaio1_0.3.110-2_amd64.deb>

wget <http://archive.ubuntu.com/ubuntu/pool/universe/n/ncurses/libtinfo5_6.2-0ubuntu2_amd64.deb>

将下载或本地上传文件移动到/opt/mysql/目录下

解压安装包：tar –zxvf mysql-server\_5.7.34-1ubuntu18.04\_amd64.deb-bundle.tar

按如下顺序依次执行指令安装mysql

sudo dpkg -i libmecab2\_0.996-1.2ubuntu1\_amd64.deb

sudo dpkg -i libaio1\_0.3.110-2\_amd64.deb

sudo dpkg -i libtinfo5\_6.2-0ubuntu2\_amd64.deb

sudo dpkg -i mysql-common\_5.7.34-1ubuntu18.04\_amd64.deb

sudo dpkg-preconfigure mysql-community-server\_5.7.34-1ubuntu18.04\_amd64.deb //此步需要输入数据的root密码

sudo dpkg -i libmysqlclient20\_5.7.34-1ubuntu18.04\_amd64.deb

sudo dpkg -i libmysqlclient-dev\_5.7.34-1ubuntu18.04\_amd64.deb

sudo dpkg -i libmysqld-dev\_5.7.34-1ubuntu18.04\_amd64.deb

sudo dpkg -i mysql-community-client\_5.7.34-1ubuntu18.04\_amd64.deb

sudo dpkg -i mysql-client\_5.7.34-1ubuntu18.04\_amd64.deb

sudo dpkg -i mysql-common\_5.7.34-1ubuntu18.04\_amd64.deb

sudo dpkg -i mysql-community-server\_5.7.34-1ubuntu18.04\_amd64.deb

sudo dpkg -i mysql-server\_5.7.34-1ubuntu18.04\_amd64.deb

登录验证：mysql -u root -p //输入密码

查看mysql服务状态：sudo systemctl status mysql.service

默认只允许本地访问，数据存储位置需修改，或者使用docker容器版本

数据、日志存储位置修改：

首先应先停止mysql服务运行：systemctl stop mysql.service

配置文件位置：/etc/mysql/mysql.conf.d/mysqld.cnf

修改内容：

[mysqld]

datadir = /var/lib/mysql #修改数据库存储目录/data/mysqldata/mysql

log-error = /var/log/mysql/error.log #日志目录可不修改

移动数据到新存储目录：

sudo cp -r /var/lib/mysql /data/mysqldata/

设置新目录所有者（如需改变）：

chown -R mysql:mysql /data/mysqldata/mysql

启动mysql服务：systemctl start mysql.service

#### Php7.3安装

添加仓库和秘钥

sudo apt-get install software-properties-common

sudo add-apt-repository ppa:ondrej/php

sudo apt update

安装php7.3

sudo apt install php7.3 php7.3-fpm php7.3-cli php7.3-common php7.3-json php7.3-opcache php7.3-xml php7.3-mbstring php7.3-zip php7.3-gd php7.3-mysql php7.3-curl php7.3-intl php7.3-readline

配置修改：vim /etc/php/7.3/fpm/pool.d/www.conf

修改内容：listen = 127.0.0.1:9000 #修改后

重启php7.3：sudo systemctl restart php7.3-fpm

查询版本：php-fpm7.3 -v

#### nginx安装

系统默认安装：apt install nginx

查询状态：systemctl status nginx

重启服务：systemctl restart nginx

重启使配置生效：nginx –s reload

### 启动php模块

上传压缩包wallet.zip至服务器数据盘/data/目录下

解压：unzip wallet.zip

设置文件权限：chmod -R 777 /data/wallet

上传nginx配置文件wallet.conf到/etc/nginx/conf.d/目录

初始化数据库：

登录mysql创建三个数据库，然后导入表,步骤如下

登录数据库：mysql –u root –p #回车，然后输入mysql密码回车

创建数据库p\_coin、p\_manage、p\_sources：指令如下

CREATE DATABASE IF NOT EXISTS p\_coin DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4\_general\_ci;

CREATE DATABASE IF NOT EXISTS p\_manage DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4\_general\_ci;

CREATE DATABASE IF NOT EXISTS p\_sources DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4\_general\_ci;

导入数据库：

查看已创建的数据库： show databases;

进入数据库p\_coin： use p\_coin;

导入数据库： source /data/wallet/p\_coin.sql;

查看数据库表： show tables;

进入数据库p\_coin： use p\_manage;

导入数据库： source /data/wallet/p\_manage.sql;

查看数据库表： show tables;

进入数据库p\_coin： use p\_sources;

导入数据库： source /data/wallet/p\_sources.sql;

查看数据库表： show tables;

最后退出数据库即可； exit;

修改配置：

Vim /data/wallet/token\_config/common/config/main-local.php

修改内容：

'components' => [

···

'db' => [

···

'password' => 'HelloChat2Te', #数据库密码，安装mysql时设置的密码

···

'db\_pwallet' => [

···

'password' => 'HelloChat2Te', #数据库密码

···

'p\_sources' => [

···

'password' => 'HelloChat2Te', #数据库密码

···

'cache' => [

···

'password' => 'T2redispass', #redis密码，安装redis时设置的密码

···

'redis' => [

···

'password' => 'T2redispass', #redis密码

···

'redis-queue' => [

···

'password' => 'T2redispass', #redis密码

···

'redis\_app' => [

···

'password' => 'T2redispass', #redis密码

···

'redis\_ticker' => [

···

'password' => 'T2redispass', #redis密码

···

'redis\_currency' => [

···

'password' => 'T2redispass', #redis密码

···

'redis\_es' => [

···

'password' => 'T2redispass', #redis密码

···

切换目录：cd /data/wallet/bwallet\_go/

修改配置：vim ./conf/app.ini

修改内容：

#mysql数据库

[database]

Type = mysql

User = root

Password = HelloChat2Te #数据库密码，安装数据库时设置的密码

Host = 127.0.0.1:3306

···

#redis设置

[redis]

Host = 127.0.0.1:6379

Password = T2redispass #redis密码，安装redis时设置的密码

运行程序：nohup ./bwallet &

nginx重启：nginx –s reload

## 5、红包与转账模块部署

### 环境准备

安装docker：apt install docker.io

安装docker-compose：apt install docker-compose

安装make：apt install make

安装jq：apt install jq

上传wallet-server.tar.gz文件至数据盘/data/中

### 启动服务

切换目录：cd /data/

解压文件：tar –zxvf wallet-server.tar.gz

切换目录：cd ./wallet-server/

初始化构建：make build

运行：make up

测试：make test

\*具体可依照说明文档READEME.md进行配置修改与操作，服务默认配置仅支持bty

### 使用付费节点教程

## getblock.io 注册和配置 rpc 节点操作

### 注册账号

https://docs.getblock.io/getting-started/how-to-set-up-an-account

### 购买套餐 ,根据实际情况 选择套餐,入门选择29每月套餐并付款

https://docs.getblock.io/getting-started/plans-and-limits/choosing-your-plan

### 配置币种RPC 节点地址

https://docs.getblock.io/getting-started/authentication-with-access-tokens

https://docs.getblock.io/guides/endpoint-setup/creating-node-endpoints

## 6、nginx前端部署

\*注：本次部署除下载页外（暂无），不包含任何前端页面

环境准备

apt install nginx

配置聊天和php的nginx

上传配置文件，修改里面转发指向IP即可

该部分主要设置域名代理，暂定设置四个域名代理、

其一：php管理后台域名代理

其二：go账单服务域名代理

其三：聊天节点域名代理

其四：平行链节点域名代理