目录

前端私有化部署文档

env.production

```
# 页面标题
VUE_APP_TITLE = 嵊州农饮水监测系统
# 生产环境配置
ENV = 'production'
# 天橙智联管理系统/生产环境
VUE_APP_BASE_API = '/prod-api'
# 路由懒加载
VUE_CLI_BABEL_TRANSPILE_MODULES = true
# 后端接口地址
VUE_APP_SERVER_API_URL = 'http://220.191.227.47:10808/prod-api'
# EMQX接口账号（后端地址和EMQX接口地址部署时，通过nginx配置代理）
VUE_APP_EMQX_API_URL = 'http://220.191.227.47:10808/emqx'
VUE_APP_EMQX_API_USER_NAME = 'admin'
VUE_APP_EMQX_API_PASSWORD = 'public'
# EMQX消息服务器连接地址
VUE_APP_EMQX_SERVER_URL = 'wss://220.191.227.47:10808/mqtt'
# 百度地图AK
# VUE_APP_BAI_DU_AK = 'nAtaBg9FYzav6c8P9rF9qzsWZXXXXXX'
VUE_APP_BAI_DU_AK = 'iwDTzotTBCFCEciUMq4yWA9Fu94r9yY2'
```

TDEngine私有化部署

版本号：3.0.3.0

docker部署：https://hub.docker.com/r/tdengine/tdengine

1. 在有网络的地方使用以下命令先拉docker镜像

`docker pull tdengine/tdengine:3.0.3.0`

2. 使用以下命令保存到本地

`docker save -o tdengine_3030.tar tdengine/tdengine:3.0.3.0`

3. tdengine_3030.tar文件copy到要装载的服务器，使用以下命令加载镜像

`docker load < tdengine_3030.tar`

4. 使用以下启动脚本启动tdengine

docker脚本启动文件内容，文件名为run.sh

```
docker rm -vf tdengine
docker run -d \
 --name tdengine \
 -e TAOS_FQDN=tdengine \
 -e TZ=Asia/Shanghai \
 -v /app/tdengine/taos/log:/var/log/taos \
 -v /app/tdengine/taos/data:/var/lib/taos \
 -v /etc/localtime:/etc/localtime \
 -p 6030:6030 \
 -p 6041:6041 \
 -p 6043-6049:6043-6049 \
 -p 6043-6049:6043-6049/udp \
 --restart=always \
 tdengine/tdengine:3.0.3.0
```

5. 启动后使用以下命令copy容器内的libtaos.so到本地

```
# 进入容器内
docker exec -it tdengine bash
# copy文件到tmp目录
cp /usr/lib/libtaos.so /tmp
# exit退出容器
exit
# copy容器内文件到本地
docker cp tdengine:/tmp/libtaos.so /app/tdengine/taos/lib
```

6. 安装客户端

按照下述文档安装客户端

https://docs.taosdata.com/develop/connect/

注意：taos.cfg文件中的firstep配置为：tdengine:6030

如果是不同服务器，则配置为：服务所在ip:6030

7. 运行taos命令，创建用户名密码及数据库

https://docs.taosdata.com/taos-sql/grant/#

https://docs.taosdata.com/taos-sql/database/

taos默认用户名:root, 密码：taosdata

EMQX私有化部署

版本号：4.0.13

docker 部署：https://hub.docker.com/r/emqx/emqx

1. 在有网络的地方使用以下命令先拉 docker 镜像

`docker pull emqx/emqx:v4.0.13`

2. 使用以下命令保存到本地

`docker save -o emqx_4013.tar emqx/emqx:v4.0.13`

3. emqx_4013.tar 文件 copy 到要装载的服务器，使用以下命令加载镜像

`docker load < emqx_4013.tar`

4. 使用以下启动脚本启动 emqx

docker 脚本启动文件内容，文件名为 run.sh，此处注意第一次启动，先不用挂在目录，因为需要先把目录里的文件都copy出来，以便重新运行挂载

```
docker rm -vf emqx
docker run -d \
--name emqx \
-v /etc/localtime:/etc/localtime \
-v /app/emqx/opt/data:/opt/emqx/data \
-v /app/emqx/opt/etc:/opt/emqx/etc \
-v /app/emqx/opt/log:/opt/emqx/log \
-p 1883:1883 \
-p 8081:8081 \
-p 8083:8083 \
-p 8084:8084 \
-p 8883:8883 \
-p 18083:18083 \
--restart=always \
emqx/emqx:v4.0.13
```

5. 启动后使用以下命令 copy 容器内的data，etc与log目录及里面的文件到本地

```
# copy容器内文件到本地
docker cp emqx:/opt/emqx/data /app/emqx/opt
docker cp emqx:/opt/emqx/etc /app/emqx/opt
docker cp emqx:/opt/emqx/log /app/emqx/opt
# 修改目录权限
chown -R 1000:1000 /app/emqx/opt
chmod -R 755 /app/emqx/opt
```

6. 修改配置文件后运行run.sh重启emqx

/app/emqx/opt/data/loaded_plugins

```
{emqx_management,true}.
{emqx_recon,true}.
{emqx_retainer,true}.
{emqx_dashboard,true}.
{emqx_rule_engine,true}.
{emqx_bridge_mqtt,false}.
{emqx_auth_http,true}.
{emqx_web_hook,true}.
{emqx_delayed_publish,true}.
```

/app/emqx/opt/etc/plugins/emqx_auth_username.conf

```
auth.user.1.username = tianjian
auth.user.1.password = tjfw888
```

/app/emqx/opt/etc/plugins/emqx_auth_http.conf

```
# java:8080可替代，换成主服务nginx转发地址，比如http://宿主机ip:port/prod-api/等
auth.http.auth_req = http://java:8080/iot/tool/mqtt/auth
auth.http.auth_req.method = post
auth.http.auth_req.params = clientid=%c,username=%u,password=%P
auth.http.request.retry_times = 3
auth.http.request.retry_interval = 1s
auth.http.request.retry_backoff = 2.0
```

/app/emqx/opt/etc/plugins/emqx_web_hook.conf

```
# java:8080可替代，换成主服务nginx转发地址，比如http://宿主机ip:port/prod-api/等
web.hook.api.url = http://java:8080/iot/tool/mqtt/webhook
web.hook.rule.client.connected.1     = {"action": "on_client_connected"}
web.hook.rule.client.disconnected.1  = {"action": "on_client_disconnected"}
```

7. 验证 http://ip:18083 用户名：admin 密码：public

应用私有化部署文档

依赖安装：TDEngine，EMQX，请参考其部署文档

主程序application.yml文件配置：

```
logging:
  level:
    com.ruoyi: debug # 此处生产环境可配置成error
    org.springframework: warn
```

```
redis:
    # 宿主机地址
    host: 172.16.5.102
    # 端口
    port: 6379
    # 数据库索引
    database: 0
    # 密码
    password: tianjian
```

```
mqtt:
    username: tianjian                  # 账号
    password: tjfw888                  # 密码
    host-url: tcp://172.16.5.101:1883          # mqtt连接tcp地址，宿主机地址
    client-id: ${random.int}                # 客户端Id，不能相同，采用随机数 ${random.value}
    default-topic: test                  # 默认主题
    timeout: 30                         # 超时时间
    keepalive: 30                       # 保持连接
    clearSession: true                    # 清除会话(设置为false,断开连接，重连后使用原来的会话 保留订阅的主题，能接收离线期间的消息)
```

```
# Swagger配置
swagger:
  # 是否开启swagger
  enabled: false # 生产环境配置为false
  # 请求前缀
  pathMapping: /dev-api
```

```
druid:
      # 主库数据源，地址与用户名密码根据实际设置配置
      master:
        url: jdbc:mysql://192.168.251.5:3307/wumeismart?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useS
        username: root
        password: tjfw888
```

```
tdengine-server:
        # 默认不启用TDengine，true=启用，false=不启用 原生连接，地址根据实际设置配置
        enabled: true
        driverClassName: com.taosdata.jdbc.TSDBDriver
        url: jdbc:TAOS://172.16.5.101:6030/tczl_smart_log?timezone=Asia/Beijing&charset=utf-8
        username: tczl
        password: 123456
        dbName: tczl_smart_log
```

以上配置好编译，主程序文件名：tczl-admin.jar

新建目录/app/tczl-smart，copy朱程序文件，DockerFile，build.sh, start.sh文件到目录下，依次运行build.sh，start.sh启动应用

DockerFile文件

```
FROM recallcode/jdk:8u172
```

```
RUN mkdir -p /app/tczl-smart/run && mkdir -p /app/tczl-smart/run/uploadPath && mkdir -p /app/tczl-smart/run/logs

COPY tczl-admin.jar /app/tczl-smart/run/tczl-admin.jar

RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

ENV JAVA_OPTS="-Xms256m -Xmx1024m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=512m"

WORKDIR /app/tczl-smart/run

ENTRYPOINT exec java $JAVA_OPTS -jar tczl-admin.jar
```

build.sh文件

```
docker load < /app/jdk8u172.tar
docker rmi tczl-admin:3.8.0
docker build -t tczl-admin:3.8.0 .
```

start.sh文件

```
docker rm -vf tczl-admin
docker run -itd --privileged=true \
    --name=tczl-admin \
 -v /app/tczl-smart/run/uploadPath:/uploadPath \
 -v /app/tczl-smart/run/logs:/logs \
 -v /app/tdengine/taos/lib/libtaos.so:/usr/lib/libtaos.so \
 -v /etc/localtime:/etc/localtime \
 -e JAVA_OPTS='-Xms256m -Xmx1024m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=512m' \
 --restart=always \
 -p 8080:8080 \
    tczl-admin:3.8.0
```

Nginx私有化配置

nginx.conf文件

```
# For more information on configuration, see:
#   * Official English Documentation: http://nginx.org/en/docs/
#   * Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;
worker_processes auto;
error_log /usr/local/nginx/logs/error.log;
pid /usr/local/nginx/logs/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /usr/local/nginx/logs/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

    include             /usr/local/nginx/conf/mime.types;
    default_type        application/octet-stream;

    #gzip  on;
    # 开启gzip压缩
    gzip on;
    # 不压缩临界值，大于1K的才压缩，一般不用改
    gzip_min_length 1k;
    # 压缩缓冲区
    gzip_buffers 16 64K;
    # 压缩版本（默认1.1，前端如果是squid2.5请使用1.0）
    gzip_http_version 1.1;
    # 压缩级别，1-10，数字越大压缩的越好，时间也越长
    gzip_comp_level 5;
    # 进行压缩的文件类型
    gzip_types text/plain application/x-javascript text/css application/xml application/javascript;
    # 跟Squid等缓存服务有关，on的话会在Header里增加"Vary: Accept-Encoding"
    gzip_vary on;
    # IE6对Gzip不怎么友好，不给它Gzip了
    gzip_disable "MSIE [1-6]\.";

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /usr/local/nginx/conf/conf.d/*.conf;

    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /app/tczl_smart_front/front/dist;

        # Load configuration files for the default server block.
        include /usr/local/nginx/default.d/*.conf;
        error_page 404 /404.html;

      location /smartview/ {
```

```
    rewrite ^/smartview/(.*)$ /$1 break;
    proxy_pass  http://localhost:8201;
        }
       location /h5/ {
    rewrite ^/h5/(.*)$ /$1 break;
    proxy_pass  http://localhost:8202;
        }
       location /prod-api/ {
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header REMOTE-HOST $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

            proxy_pass http://localhost:8080/;
      }

     location /videos/ {
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header REMOTE-HOST $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

            proxy_pass http://183.248.212.161:7086/;
            add_header Content-Security-Policy upgrade-insecure-requests;

       }
     location /emqx {
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header REMOTE-HOST $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_pass http://localhost:8081;
        }
     location /api/v4/ {
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header REMOTE-HOST $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_pass http://localhost:8081/api/v4/;
        }
        # wss连接
       location /mqtt {
            proxy_pass http://localhost:8083/mqtt; # mqtt:wss连接代理到mqtt:ws
            proxy_read_timeout 60s;
            proxy_set_header Host $host;
            proxy_set_header X-Real_IP $remote_addr;
            proxy_set_header X-Forwarded-for $remote_addr;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'Upgrade';
        }
      location = /404.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
    }

# Settings for a TLS enabled server.
#
#    server {
#        listen       443 ssl http2;
#        listen       [::]:443 ssl http2;
#        server_name  172.16.5.101;
#        root         /usr/share/nginx/html;
#
#        ssl_certificate "/etc/pki/nginx/server.crt";
#        ssl_certificate_key "/etc/pki/nginx/private/server.key";
#        ssl_session_cache shared:SSL:1m;
#        ssl_session_timeout  10m;
```

```
#        ssl_ciphers HIGH:!aNULL:!MD5;
#        ssl_prefer_server_ciphers on;
#
#        # Load configuration files for the default server block.
#        include /etc/nginx/default.d/*.conf;
#
#        error_page 404 /404.html;
#            location = /40x.html {
#        }
#
#        error_page 500 502 503 504 /50x.html;
#            location = /50x.html {
#        }
#    }

}
```

conf.d/smartview.conf

```
server {
 listen       8201;
 server_name  172.16.5.101; # 宿主机ip

 location / {
    add_header Access-Control-Allow-Origin *;
           add_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';
           add_header Access-Control-Allow-Headers *;

           if ($request_method = 'OPTIONS') {
               return 204;
           }
           root /app/tczl-smart-front/smartview/dist;
           index index.html;
           try_files $uri $uri/ /index.html;
 }
}
```

conf.d/h5.conf

```
server {
 listen       8202;
 server_name  172.16.5.101; # 宿主机ip

 location / {
    add_header Access-Control-Allow-Origin *;
           add_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';
           add_header Access-Control-Allow-Headers *;

           if ($request_method = 'OPTIONS') {
               return 204;
           }
           root /app/tczl-smart-front/h5/dist;
           index index.html;
           try_files $uri $uri/ /index.html;
 }
}
```

FineReport部署包部署

1. 概述

1.1 版本

服务器部署包版本	操作系统
V 11.0	64 位

1.2 部署包简介

FineReport 提供了 Linux 服务器部署包，该部署包内置有 JDK 和 Finereport 工程，用户部署完即可使用。

注：不支持在 32 位操作系统上安装。

2. 操作步骤

2.1 环境准备

需参考 部署环境准备 文档，准备相关环境。

2.2 下载部署包

1）帆软提供 Linux X86 和 Linux ARM 两种类型的部署包，用户下载部署包前需使用uname -m确认系统架构，如下图所示：

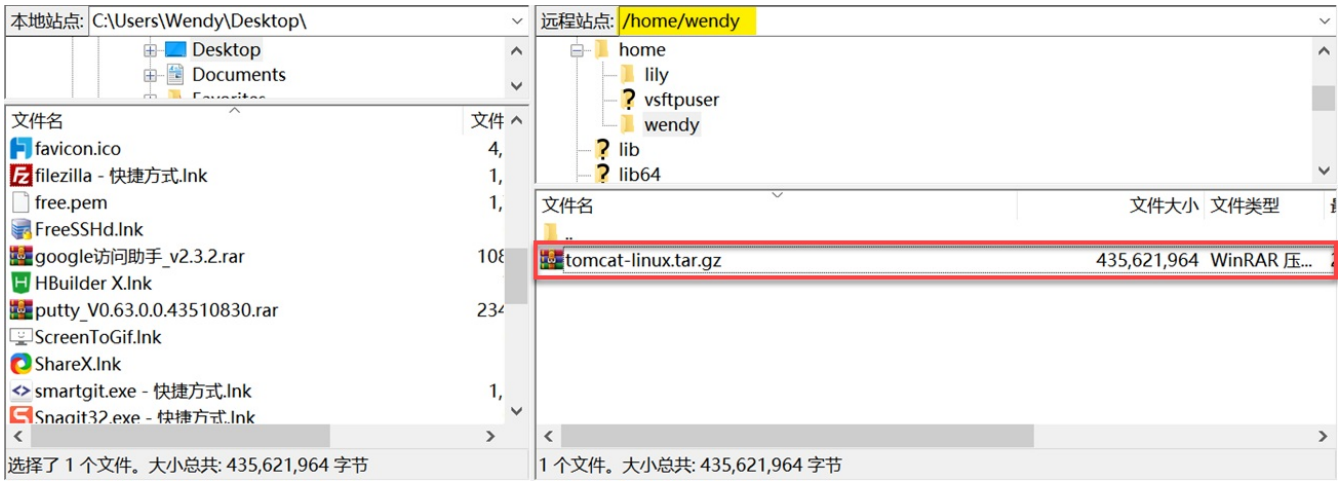
```
[root@ecs-new-0017 ~]# uname -m
x86_64
[root@ecs-new-0017 ~]#
```

2）进入 FineReport 官网，点击 帆软下载中心 ，下载 Linux 版服务器部署包，如下图所示：

3）下载完成后上传到 Linux 系统里，如下图所示：

注 1：本文示例上传到/home/wendy 路径下，即为下文的部署包所在目录，用户可自行调整。

注 2：示例使用 FTP 工具，也可使用其他文件传输工具。



2.3 解压部署包

执行语句如下所示：

```
cd /home/wendy  #进入上传部署包所在目录
tar -zxvf tomcat-linux.tar.gz  # 解压 tomcat 安装包
mv tomcat-linux tomcat  # 重命名文件夹
```

2.4 启动 Tomcat

1）执行语句如下所示：

```
cd /home/wendy/tomcat/bin  # 进入bin目录
./startup.sh  # 启动 Tomcat
```

```
[root@ecs-new-0017 wendy]# cd /home/wendy/tomcat/bin
[root@ecs-new-0017 bin]# ./startup.sh
Using CATALINA_BASE:   /home/wendy/tomcat
Using CATALINA_HOME:   /home/wendy/tomcat
Using CATALINA_TMPDIR: /home/wendy/tomcat/temp
Using JRE_HOME:        /home/wendy/tomcat/jre
Using CLASSPATH:       /home/wendy/tomcat/bin/bootstrap.jar:/home/wendy/tomcat/bin/tomcat-juli.jar
Tomcat started.
[root@ecs-new-0017 bin]#
```

2）查看实时日志，实时日志中出现重启耗时则意味着启动成功。如下图所示：

```
cd /home/wendy/tomcat/logs
tail -f catalina.out
```

```
[root@ecs-new-0017 bin]# cd /home/wendy/tomcat/logs
[root@ecs-new-0017 logs]# tail -f catalina.out

-------------------------------------------------------------------------------------------
[GMS]node=517f7fdd-94ad-1f8d-0d02-ea07bd446e60, channel=DB_CACHE(clusterID_1421981710):DecisionDB:0, address=192.168.1.24:7850
-------------------------------------------------------------------------------------------
[View]:[517f7fdd-94ad-1f8d-0d02-ea07bd446e60|0] (1) [517f7fdd-94ad-1f8d-0d02-ea07bd446e60]
-------------------------------------------------------------------------------------------
16:34:12 localhost-startStop-1 ERROR [standard]  Dynamic transform error:com.fr.plugin.bridge.PluginClusterHelper
16:34:12 localhost-startStop-1 ERROR [standard]  Dynamic transform error:com.fr.plugin.bridge.TaskCallbackSerializer
16:34:12 localhost-startStop-1 ERROR [standard]  Dynamic transform error:com.fr.plugin.bridge.PluginFileSerializer
16:34:12 localhost-startStop-1 ERROR [standard]  Dynamic transform error:com.fr.plugin.bridge.PluginFileSerializer$1
Apr 01, 2021 4:35:05 PM org.apache.tomcat.util.descriptor.web.SecurityConstraint findUncoveredHttpMethods
SEVERE: For security constraints with URL pattern [/*] only the HTTP methods [TRACE HEAD OPTIONS] are covered. All other methods are u
ncovered.
01-Apr-2021 16:35:05.627 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web applica
tion directory [/home/wendy/tomcat/webapps/webroot] has finished in [132,741] ms
01-Apr-2021 16:35:05.632 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8081"]
01-Apr-2021 16:35:05.645 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 132778 ms
```

3）在浏览器中输入访问地址**http://IP:端口号/工程名/decision**，即可访问工程。如下图所示：

注 1：「工程名」默认为「webroot」 ，若修改过，使用最新名字。

注 2：启动 Tomcat 后，如果报错“Database is not initialized”，可以检查下是不是读写权限的问题，如果是可以用命令 chmod -R 777 tomcat 开下权限。

数据连接概述

1. 概述

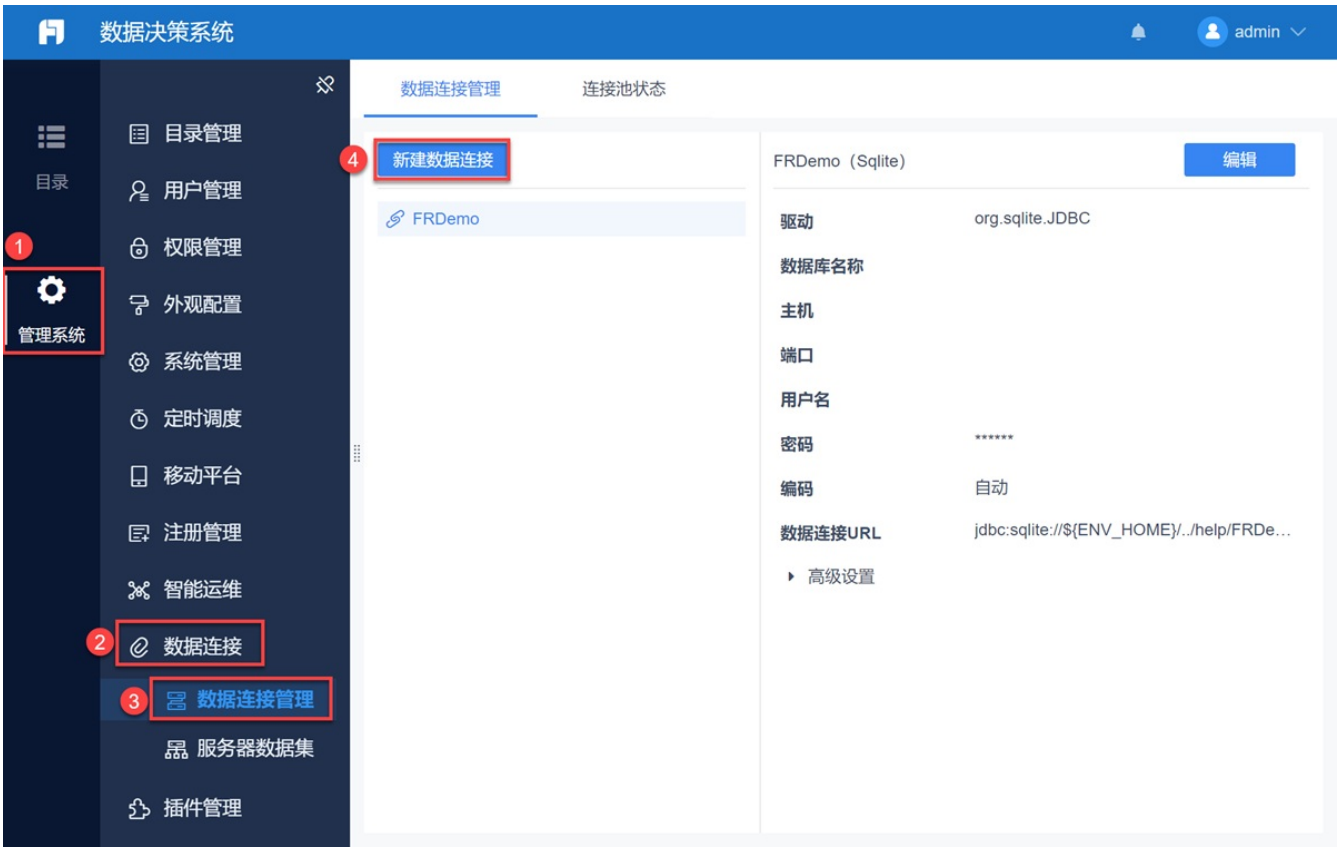
1.1 应用场景

在制作报表前，我们需要在 FineReport 中建立与数据库的连接，让 FineReport 能获取到数据库中的数据。

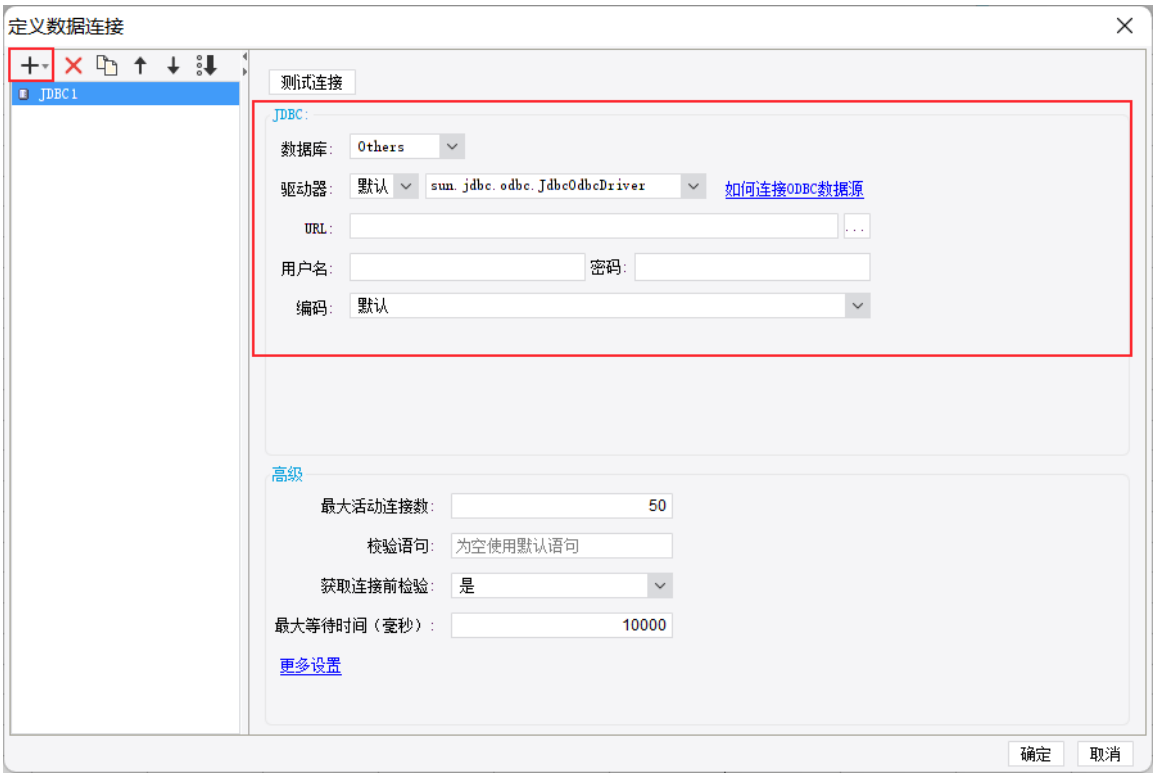
1.2 功能简介

FineReport 中有两种数据连接方式：

- 在数据决策系统中建立数据连接。选择「服务器 > 报表平台管理」，打开数据决策系统，建立数据连接。详情请参考：平台数据连接



- 在设计器中建立连接，详情请参见：[设计器]JDBC 连接数据库



2. JDBC 连接与 JNDI 连接对比

``	JDBC 连接数据库	JNDI 连接数据库

定义	JDBC（Java Data Base Connectivity，Java 数据库连接）是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一访问	JNDI (Java Naming and Directory Interface)是一个应用程序设计的 API，为开发人员提供了查找和访问各种命名和目录服务的通用、统一的接口，类似 JDBC，都是构建在抽象层上
连接方式区别	JDBC 就是直接连接物理数据库，连接数据库比较快，但在程序中使用的话就比较烦琐，每次连接都要有一定的编码，和数据库的连接需要手动关闭	使用 JNDI 连接某个数据源，此数据源所连接的数据库都在应用服务器端定义
各自优势	在报表服务器部署后，如果数据库的相关参数变更，重新修改配置文件中的 JDBC 参数即可，只要保证数据源的名称不变，数据连接就无需修改；JDBC 避免了报表与数据库之间的紧耦合，和项目共用服务器的连接池，且连接速度比较快，使应用更加易于配置、易于部署	JNDI 只需要把数据库后台、驱动、URL、用户名、连接池等问题交给 J2EE 容器来配置和管理，然后对这些配置和管理进行引用即可

3. FineReport 适配数据库

内容	简介	文档教程
JDBC 连接数据库（官方）	以连接 Oracle 数据库为例，介绍 JDBC 方式连接数据库的操作步骤	JDBC 连接数据库
JNDI 连接数据库（官方）	通过 JNDI 方式定义数据连接	JNDI 连接数据库
官方插件	XMLA 数据连接	通过 XMLA 数据连接的方式来与多维数据库进行连接
SAP 数据连接	FineReport 将 SAP 数据连接功能做成一个插件，新增 SAP 数据连接类型，添加 SAP 数据集类型	SAP 数据连接``
SAP BW 数据连接	新 SAP BW 多维数据集用于连接 BW Cube 和 BW Query，从以前的多维数据集中分离出来单独做成了插件。	SAP BW 数据连接``
多维数据库插件	连接多维数据库	多维数据库插件``
JSON 数据连接	通过插件将 JSON 格式的数据转变为报表中可以使用的数据表。	JSON 数据连接
MongDB 数据连接	安装插件可以连接 MongoDB 数据库	MongoDB 数据连接
第三方插件``	Redis 数据连接``	Redis 缓存数据库也可通过插件进行连接
Elasticsearch 数据集插件	将 ElasticSearch 的查询结果通过 FineReport 展示	Elasticsearch 数据集插件-悦享版
InfluxDB 数据集插件	连接 InfluxDB 数据库	InfluxDB 数据集插件``
新 SSAS 插件``	连接 SSAS 数据库	新 SSAS 插件

已经部署完成的

嵊州n2：嵊州数据决策系统 账号：admin 密码: admin (需要先连接vpn)

阿里云n3：阿里云 n3 数据决策系统 账号：admin 密码：admin

阿里云n2：n2 数据决策系统 账号：admin 密码：admin