esProc

Multitenancy Scheme under Docker

By Raqsoft

Introduction

This PPT includes server environment configuration for running esProc under Docker, esProc client-side usage and DFX script explanation.

After loading the esProc mirror, a Docker administrator will configure Docker administration service and user resource (disk, RAM and CPU) management.

Users can call a remote dfx script to execute or start cluster computing after they log onto through esProc clientside.

Contents

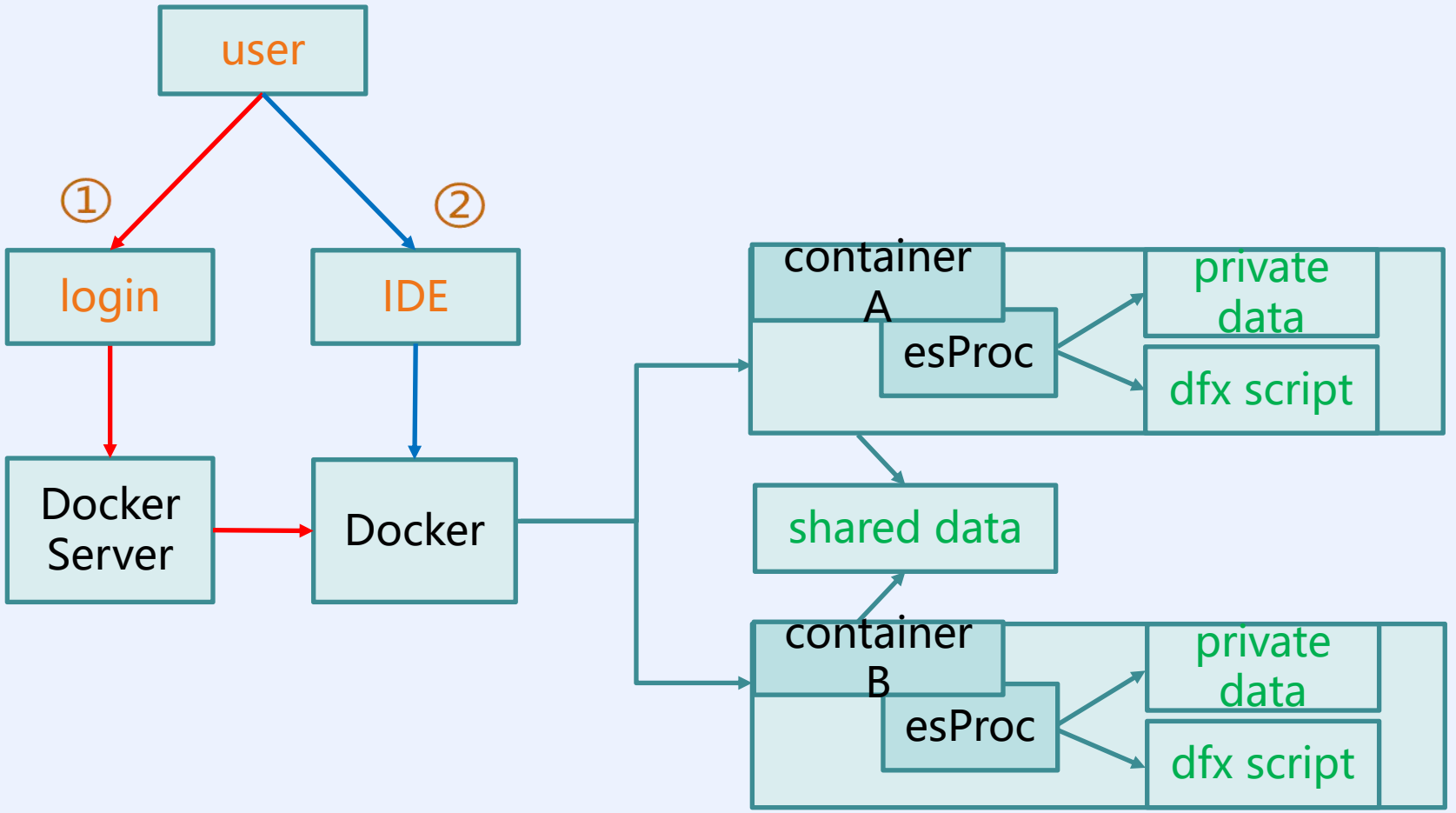
I Overview

Docker uses virtualization that covers from file system and network to process segregation to create segregated, independent and easy to maintain containers.

esProc can be deployed on servers to form a cluster to implement complicated data analysis and to help increase data processing performance.

The data duo of esProc and Docker can cover more extensive computing scenarios without upgrading resources, and achieves efficiency and reasonable resource allocation and management thanks to their independency.

II Architecture



II Architecture

User flow:

A login: Once users log in Docker on client side, dockerServer starts Docker containers and esProc service according to the existing configurations.

B work: Users then access esProc on the server side to execute dfx scripts from client side esProc IDE.

C logout: Relevant Docker containers will be closed at logout.

Relations between **server**, **Docker container** and **esProc**:

A One-to-one relationship between Docker container and esProc.

B Multiple Docker containers can run on one server.

C One user can use one or more Docker containers.

D The system supports multiple users working concurrently and independently.

Shared data: Directories or files available to every Docker user; they are read-only and maintained by admins.

Private data: Each user has their own private data zone holding directories and files which only they can access after signing in.

III Server environment configuration

The Docker service esProc provides runs only on Linux platform in C/S structure. The following instructions & explanations are based on ubuntu15 + jdk1.8 + docker 1.9.1 with root account located in */home/docker/java*.

1 Install Docker:

apt-get install docker-engine

View Docker info:

docker info

Start Docker service:

/etc/init.d/docker start

Running test:

```
root@master:~# docker run busybox /bin/echo "Hello World"
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox

234382100b69: Pull complete
2ed2a0eb52cd: Pull complete
Digest: sha256:26820e4a4b7b507d71fc0d1983ca00a1a2774e03e590e68875669f0e2a768497
Status: Downloaded newer image for busybox:latest
Hello World
root@master:~#
```

If **Java** isn't installed, use
apt-get install oracle-java8-installer to install it

2 Install esProc mirror

Download esProc mirror files & Docker management files
wget http://download.raqsoft.com.cn/esproc/docker/esproc.tar.gz
wget http://download.raqsoft.com.cn/esproc/dockerServer/dockerServer.zip

unzip mirror file: esproc.tar.gz
tar -zxvf esproc.tar.gz

Load esProc mirror:
docker load < esproc.tar

View Docker image:
docker images

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
esproc	latest	ea30c50c8893	7 hours ago	783.4 MB

Test mirror (Source of raq_start.sh is explained in Docker server management later):
cd /home/docker/java
./raq_start.sh aaa 500m 512 /opt/app/aaa 8282 /home/docker/share

2 Install esProc mirror

View server processes:
ps –ef

```
[kworker/3:1]
docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8281 -container-ip 172.17.0.2 -container-port 8281
/bin/bash
/bin/bash /raqsoft/esProc/bin/ServerConsole.sh -plocalhost:8281
java -Xms128m -Xmx1024m -cp /raqsoft/esProc/classes:/raqsoft/esProc/lib/*:/raqsoft/common/jdbc/* -Duser.language=en
docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8282 -container-ip 172.17.0.3 -container-port 8282
[kworker/1:2]
/bin/bash
/bin/bash /raqsoft/esProc/bin/ServerConsole.sh -plocalhost:8282
java -Xms128m -Xmx1024m -cp /raqsoft/esProc/classes:/raqsoft/esProc/lib/*:/raqsoft/common/jdbc/* -Duser.language=en
```

Processes running as duo: Make sure both *docker-proxy* and *ServerConsole.sh* operate in Docker containers for each esProc application. The processes use same port. *ServerConsole.sh* is absent if esProc service isn’ t successfully started. Users can check the log info by starting the service in Docker containers.

View running containers:
docker ps

2 Install esProc mirror

Check logs:

```
# ./go.sh aaa                              //User aaa enters a Docker container
# ps -ef                                   //Check status of processes
# ./ServerConsole.sh -plocalhost:8285      //Start esProc service
```

```
root@master:/home/docker/java# ./go.sh aaa
root@cdc49cbd17a2:/raqsoft/esProc/bin# ./ServerConsole.sh -plocalhost:8285
Log level:DEBUG
[2019-11-27 09:17:15]
DEBUG: Dfx path: /app/demo


[2019-11-27 09:17:15]
INFO: load library [mongo] from MongoCli

[2019-11-27 09:17:15]
INFO: Inner version: 20191014

[2019-11-27 09:17:15]
INFO: Starting unit server...

[2019-11-27 09:17:15]
INFO: Using start.home=/raqsoft/esProc

[2019-11-27 09:17:16]
DEBUG: Using TempTimeOut=12 hour(s).

[2019-11-27 09:17:16]
DEBUG: Using ProxyTimeOut=12 hour(s).

java.lang.Exception: Starting 172.17.0.2:8285 failed.Please check does the port is occupied or no license permit.
        at com.raqsoft.parallel.UnitContext.<init>(UnitContext.java:265)
        at com.raqsoft.server.unit.UnitServer.checkMainProcess(UnitServer.java:328)
        at com.raqsoft.server.unit.UnitServer.run(UnitServer.java:421)
        at java.lang.Thread.run(Thread.java:745)
Exit
root@cdc49cbd17a2:/raqsoft/esProc/bin# more ../
```

Track errors by checking the logs

3 Resource configuration management

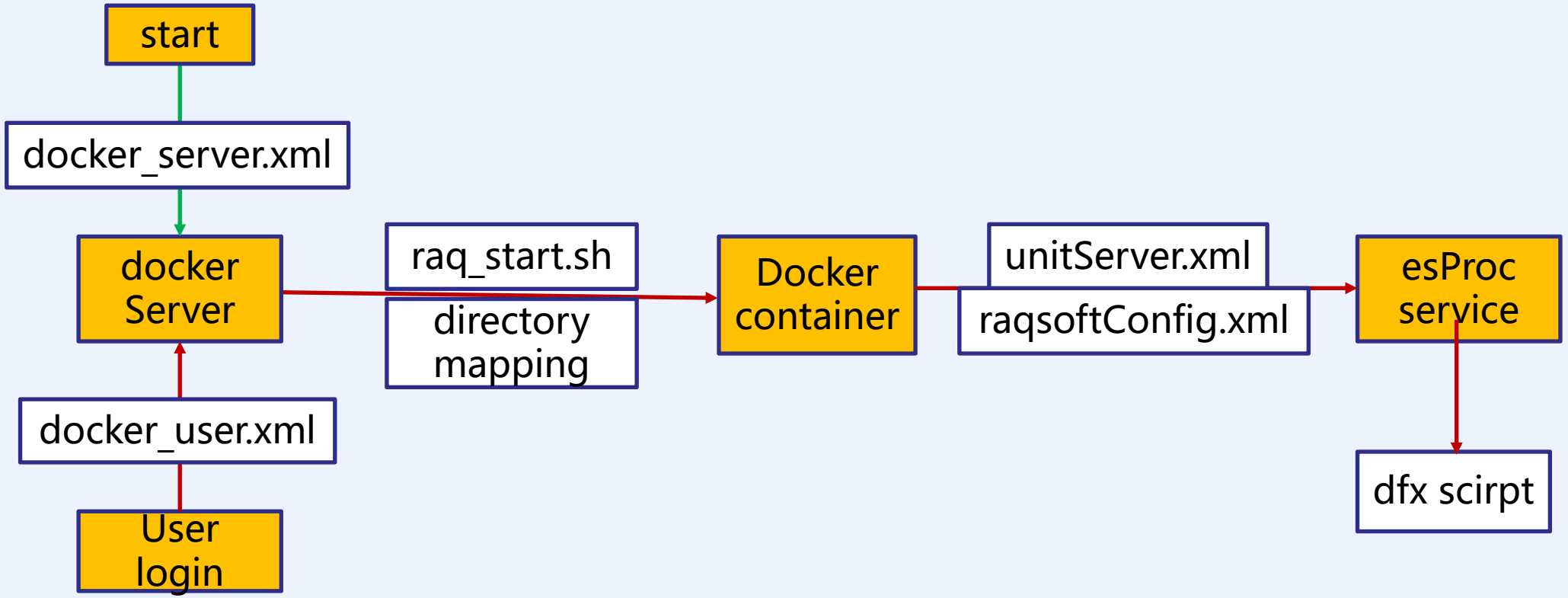
A Configuration files:

Option	Directory	Description
dockerServer	/home/docker/java	Docker management server
docker_server.xml	/home/docker/java	Docker server configuration file; <NodeType> option exists only on the main server
docker_user.xml	/home/docker/java	Docker user management file that exists only on the main server
unitServer.xml	/home/docker/share	esProc cluster resource configuration file
user	/opt/app/user	Docker user’ s private directory
raqsoftConfig.xml	/opt/app/user/config	esProc configuration file

/opt/app stores user data. The directory will be mapped to the Docker containers.

B Uses of configuration files

Use configuration files in the workflow:



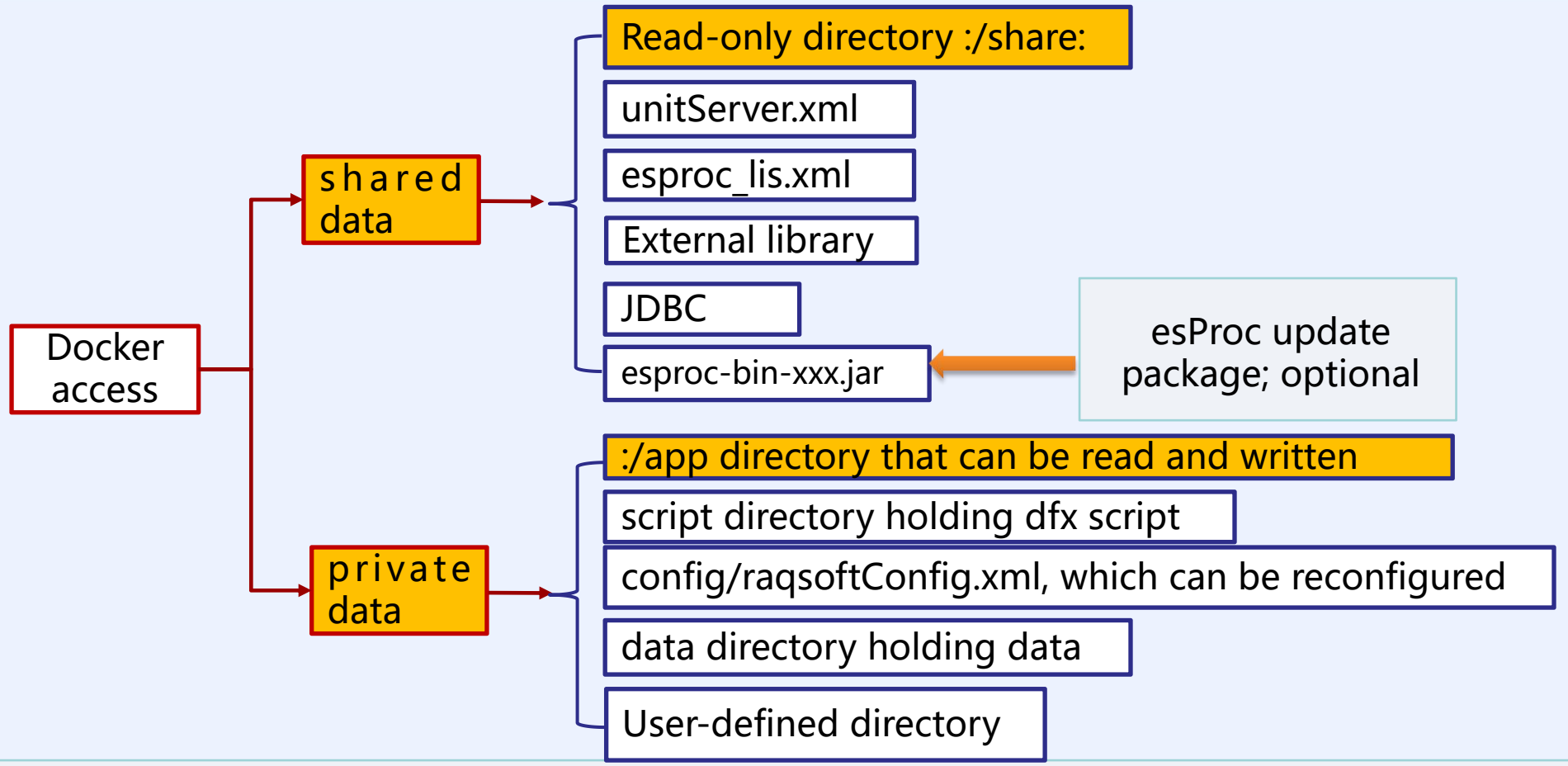
C Container directory mapping

A Docker image is read-only and can't be modified. The container's contents will be deleted as a container is closed. So we load the shared directory and user directory on the host machine into the Docker containers to enable esProc to read and write user data and thus simplify users' work.

host machine directory	Docker container directory	description
/home/docker/share	/share	Global shared directory
/home/docker/share	/share/jdbc	Directory holding JDBC jar
/home/docker/share	/share/extlib	Directory holding external library files
/opt/app/user	/app	User's main directory <user>
/opt/app/user/script	/app/script	Directory holding dfx script
/opt/app/user/data	/app/data	User data directory
/opt/app/user/config	/app/config	Directory holding configuration files

The directories are auto-created. Users can create a subdirectory under <user> and access it via "/app/subdirectory".

D Docker access control



Shared data: Managed by Docker admin and accessible by all users at read-only level.
Private data: Managed and accessible by Docker users only at read-write & modify level; can be **permanently** stored in a directory on host machine even if the Docker containers are closed.

4 Docker server management

dockerServer manages user login and container startup & closeup.

A Files:

file	description
dockerServer.jar	jars for starting Docker server
go.sh	Enter a Docker container image
raq_get_image.sh	View a running Docker container image
raq_start.sh	Start Docker image
raq_stop.sh	Close Docker image
server.sh	Docker server startup script
docker_server.xml	Docker server configuration file
docker_users.xml	Docker user management file

Put all scripts and configuration files in *dockerServer.zip* and store it in */home/docker/java*.

4 Docker server management

B Server configuration file：

Content of configuration file *docker_server.xml*:

```
<Config>
    <Server>192.168.0.76:9001</Server>
    <Share>/home/docker/share</Share>
    <NodeType>manager</NodeType>
</Config>
```

<NodeType> option: Used for setting the main server.
<Server> option: Server IP and port, which are used at login on the client side.
<Share> option: Shared directory accessible by Docker containers, esProc cluster configuration file unitServer.xml, etc. It is read-only for Docker users.

4 Docker server management

C User management:

Content of Docker user management file **docker_user.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>
 <Users>
        <User>
                <Name>aaa</Name>
                <Alias>Jason</Alias>
                <Pwd>123456</Pwd>
                <Memory>500m</Memory>
                <CPU>512</CPU>
                <Home>/opt/app/aaa</Home>
                <Unit>192.168.0.76:8281,192.168.0.76:8282,192.168.0.77:8281</Unit>
        </User>
        <User>
                <Name>bbb</Name>
                <Alias>Luna</Alias>
                <Pwd>123456</Pwd>
                <Memory>100m</Memory>
                <CPU>300</CPU>
                <Home>/opt/app/bbb</Home>
                <Unit>192.168.0.77:8282</Unit>
        </User>
        ……
```

4 Docker server management

Docker admin is responsible for allocating resources to users. The allocation information is stored in a separate file:
Content: name, password, CPU & memory quota, user directory and the node running esProc.

CPU resource quota: Set CPU resource quota for containers having instantiated images through --cpu-shares option.

Memory resource quota: through –m parameter, whose value must be an integer in the unit of k, m or g. Can be also used to set Java memory size at the start of esProc (startup will fail if the value is a decimal).

<Home> option: Set a user directory where user stores their data. Docker will map the directory to containers for their access of data on the host machine.

<Unit> option: Set a user node whose **ip:port is unique** in case of resource collision. Each ip:port points to a Docker container. One user could be given one or multiple containers.

4 Docker server management

D Start Docker server:

Modify parameters in server.sh:
start_home=/home/docker/java

Start dockerServer service:
/home/docker/java/server.sh

Check network status fo dockerServer:
netstat -na|grep 9001
tcp6 0 0 192.168.0.76:9001 :::* LISTEN

5 esProc node configuration

Cluster resource configuration file **unitServer.xml**:

```
<SERVER Version="3">
  <TempTimeOut>12</TempTimeOut>
  <Interval>1800</Interval>
  <ProxyTimeOut>12</ProxyTimeOut>
  <Hosts>
    <Host ip="localhost" maxTaskNum="4" preferredTaskNum="2">
      <Partitions>
        <Partition name="0" path="/app/data">
        </Partition>
      </Partitions>
      <Units>
        <Unit port="8281">
        </Unit>
      </Units>
    </Host>
    ....................
    <Host ip="localhost" maxTaskNum="4" preferredTaskNum="2">
      <Partitions>
        <Partition name="0" path="/app/data">
        </Partition>
      </Partitions>
      <Units>
        <Unit port="8290">
        </Unit>
      </Units>
    </Host>
  </Hosts>
  <EnabledClients check="false">
  </EnabledClients>
</SERVER>
```

The Docker admin allocates esProc node access port and map it to Docker access port to enable Docker users to access esProc. The Host ip is localhost and each host is given an available, unique port.

6 esProc configuration

Content of **esProc configuration file** *raqsoftConfig.xml*:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config Version="2">
  <Runtime>
    <DBList encryptLevel="0">
    </DBList>
    <Esproc>
      <charSet>GBK</charSet>
      <dfxPathList>
        <dfxPath>/app/script</dfxPath>
      </dfxPathList>
      <dateFormat>yyyy-MM-dd</dateFormat>
      <timeFormat>HH:mm:ss</timeFormat>
      <dateTimeFormat>yyyy-MM-dd HH:mm:ss
      </dateTimeFormat>
      <mainPath>/app/script</mainPath>
      <tempPath></tempPath>
      <bufSize>65536</bufSize>
      <localHost></localHost>
      <localPort>0</localPort>
      <parallelNum>64</parallelNum>
<zoneLockTryTime>3600</zoneLockTryTime>
<simpleTableBlockSize>1048576</simpleTable
BlockSize>
      <nullStrings>nan,null,n/a</nullStrings>
      <extLibsPath>/share/extlib</extLibsPath>
        <importLibs>
          <lib>HdfsCli</lib>
        </importLibs>
    </Esproc>
    <Logger>
      <Level>DEBUG</Level>
    </Logger>
  </Runtime>
</Config>
```

6 esProc configuration

Configuration file description:

dfx directory: Set dfxPth and mainPath in the form of /app/script to call a dfx script conveniently.

External library directory: Set external library directory extLibsPath in the form of /share/extlib. User can define an external library as needed through importLibs.

Multithreaded processing: Set parameter parallelNum’ s value >1 (default is 64) to take advantage of a multicore CPU.

Note: Users are advised not to modify the path directories since they need to be accessible by Docker containers. To make configurations in *raqsoftConfig.xml* take effect, put it under config in user directory and then restart Docker.

7 Subserver configuration

Configurations of **subservers** are similar to those of the **main server** previously described. Most of the relevant files are those included in <resource files>. Only certain modifications are needed:

Modfy **docker_server.xml** as follows:

```
<Config>
    <Server>192.168.0.77:9001</Server>
    <Share>/home/docker/share</Share>
</Config>
```

Though docker_user.xml isn’ t needed here, for a Docker user the user directory structure and deployment on subservers is the same as those on the main server. Both the main server and the subservers use one *docker_user.xml* file.

Start **dockerServer** on each server before getting started with Docker.

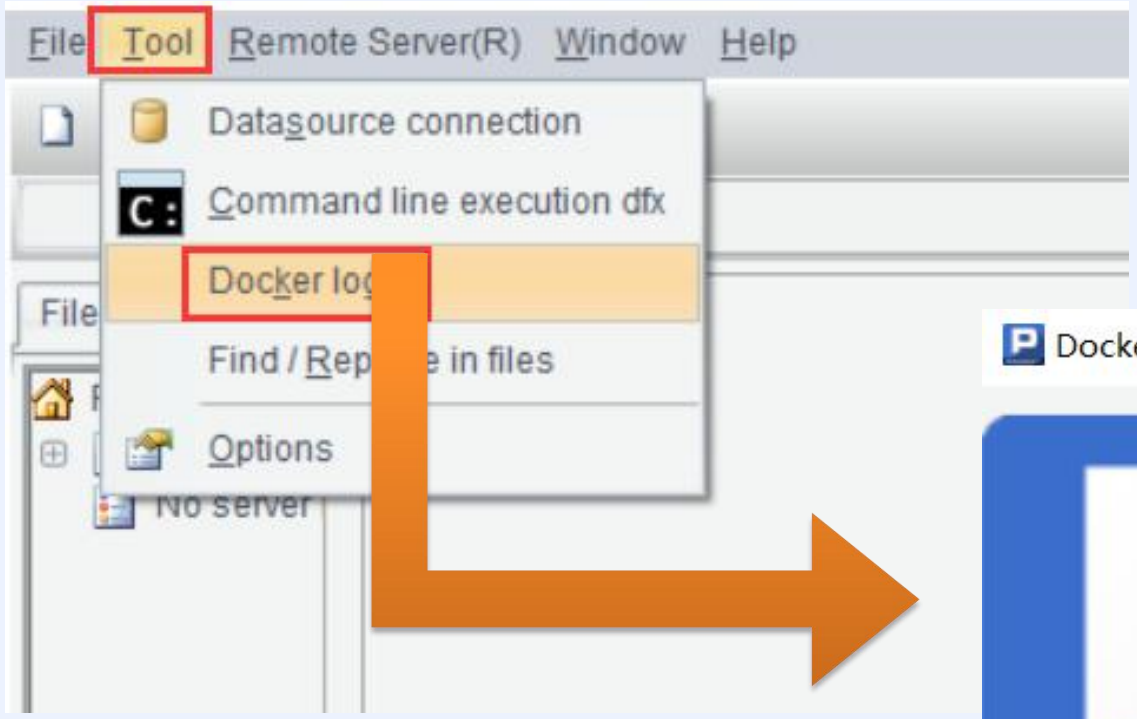
```
# /home/docker/java/server.sh
```

Refer to main server configurations to finish other configurations for subservers.

IV Uses of client side

1 Login

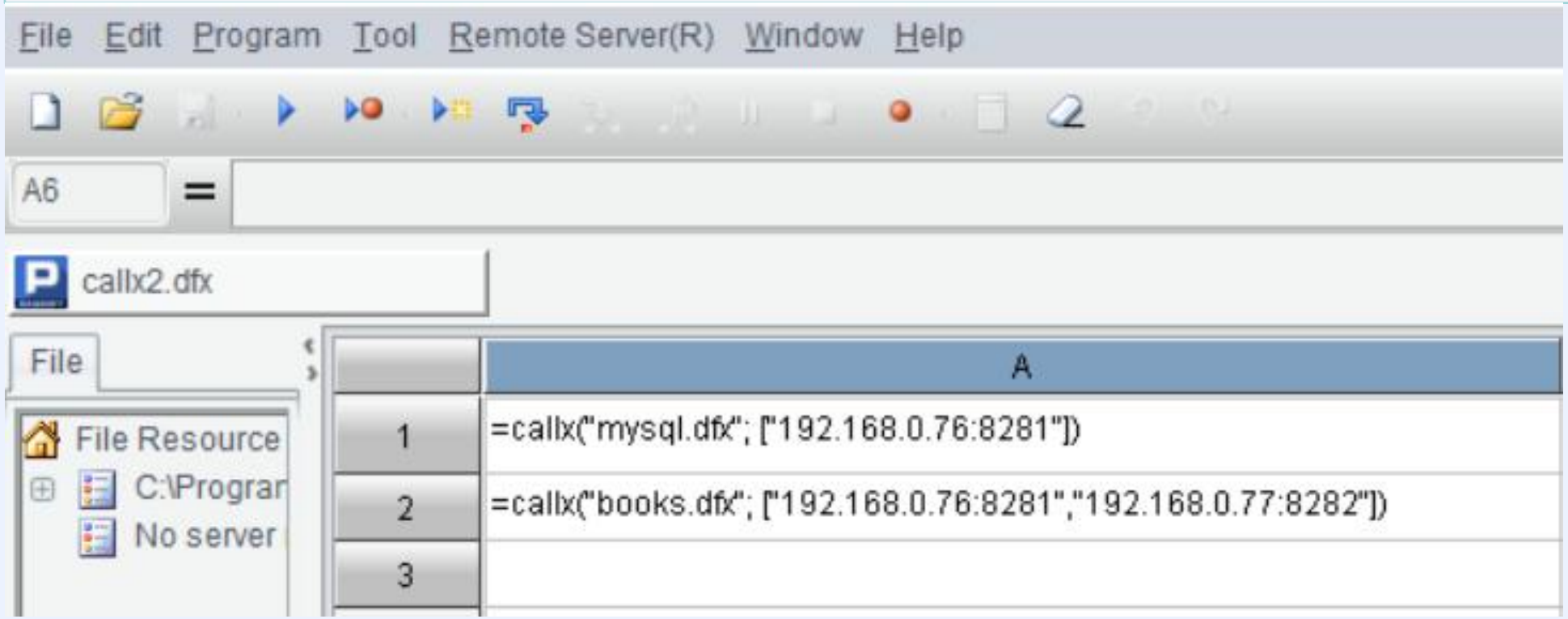
Users can log Docker through **Menu->Tool->docker login** on esProc IDE.



Docker Logout will be shown once you sign in

IV Uses of client side

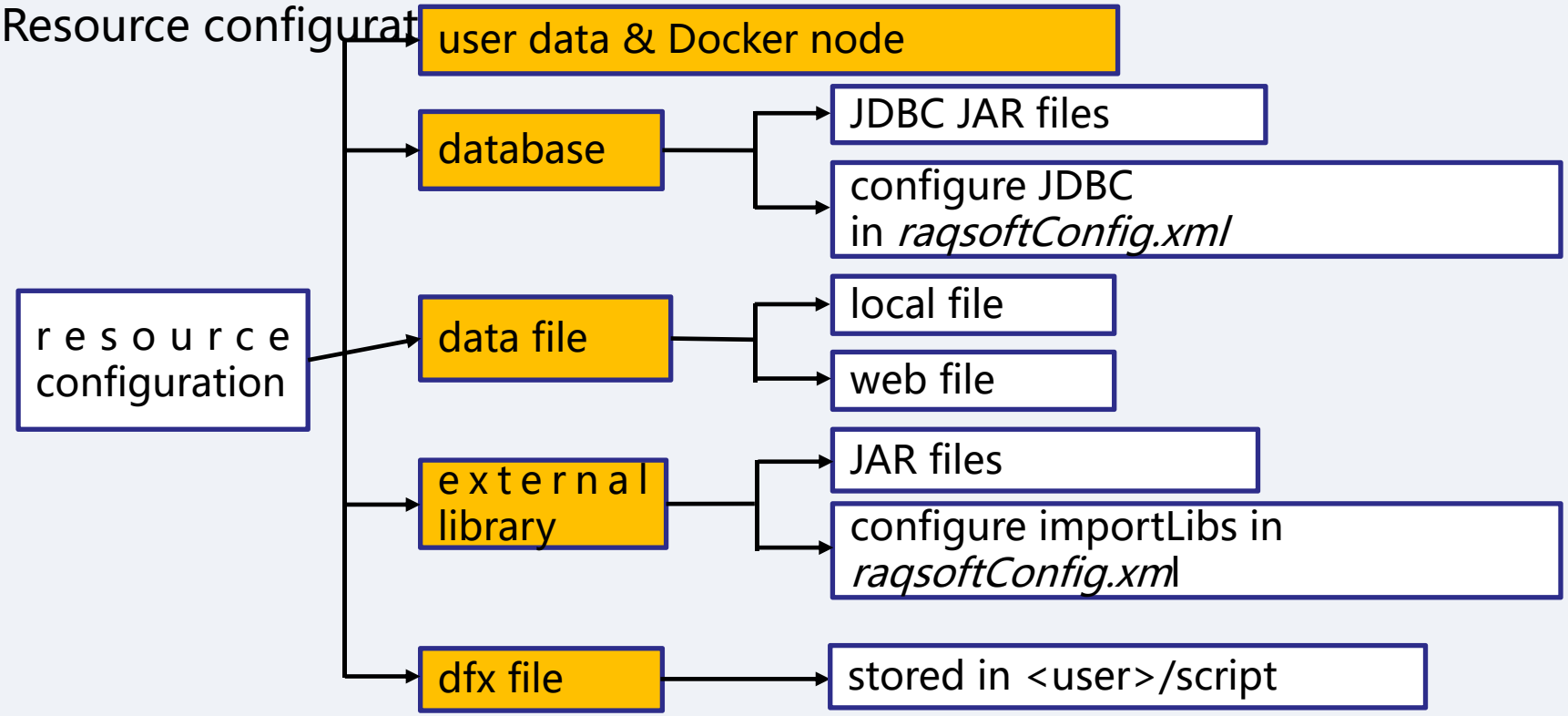
will be started on the server side after users log in. Then they can begin to work on IDE, as shown below:



Put *mysql.dfx* and *books.dfx* in user’ s script directory on the server, like */opt/app/aaa/script.*

2 esProc resource configuration

When using esProc mirror, users can put resources on the **server side** as needed to ensure that the dfx script is properly executed and, in the meantime, call a remote dfx script via esProc interface **callx** on the **server side**. (Read esProc Function Reference to learn more about esProc interface)



3 Querying files

Requirement: Get data meeting the condition STATE="New York " from *employee.txt* according to the configurations below:

	Server	Docker container	Client
IP:PORT	192.168.0.76:8281	172.17.0.2:8281	192.168.0.186
<user>	/opt/app/aaa	/app	
dfx script	script/Employee.dfx	script/Employee.dfx	loadEmp.dfx
data	data/EMPLOYEE.txt	data/EMPLOYEE.txt	

A1 Employee.dfx script on server side:

For explanations only. Handling is not needed.

	A	B
1	=file("/app/data/EMPLOYEE1.txt")	//Read the file
2	=A1.import@t()	//Load in data
3	=A2.select(STATE==pState)	//pState is the filtering condition
4	return A3	//Return a table sequence

3 Accessing files

A2 loadEmp.dfx on the client side:

	A	B
1	[192.168.0.76:8281]	//Configure node
2	=callx("Employee.dfx ","New York";A1)	//Call *Employee.dfx* script
3	=A2.conj()	

Result:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
216	Brooke	Williams	F	New York	1978-12-03	2003-11-01	2	12000
385	Taylor	Wilson	F	New York	1980-05-15	2001-04-01	7	7000
221	Sarah	Davis	F	New York	1982-09-04	2007-03-01	4	5000
196	Lauren	Thomas	F	New York	1976-08-24	2009-08-01	2	12000
166	Emily	Moore	F	New York	1977-10-22	2006-01-01	3	7000
180	Abigail	Smith	F	New York	1972-09-19	2007-05-01	5	5000
297	Julia	Johnson	F	New York	1983-06-26	2000-09-01	7	5000

V Other data sources

Let’ s take user aaa as an example to look at how to access different data sources.

Test environment:

Option	Client	Server A	Server B
os	Win10	Ubuntu15	Ubuntu15
ip	192.168.0.186	192.168.0.76	192.168.0.77
port		8281	8282
Docker management	x	main server	subserver
mysql	√	x	x
mongodb	√	x	x
<user> directory		/opt/app/aaa	/opt/app/aaa

V Other data sources

1 Databases

Requirement: data source is MySQL, but esProc mirror files don’ t include MySQL and the relevant jars.
Solution: Docker can access external libraries without installing MySQL in it.
Configurations are as below:

	Server	Docker container	Client
IP:PORT	192.168.0.76:8281	172.17.0.2:8281	192.168.0.186
<user>	/opt/app/aaa	/app	
dfx script	script/mysql.dfx	script/mysql.dfx	loadEmp.dfx
J D B C configuration	config/raqsoftConfig.xml		
Jdbc jar	/home/docker/share/jdbc/mysql-connector-java-5.1.31-bin.jar		

V Other data sources

1 Databases

A1 Configure MySQL in *raqsoftConfig.xml*:

```
<DBList encryptLevel="0">
    <DB name="mysql">
      <property name="url" value="jdbc:mysql://192.168.0.186:3306/docker"/>
      <property name="driver" value="com.mysql.jdbc.Driver"/>
      <property name="type" value="10"/>
      <property name="user" value="un"/>
      <property name="password" value="un1234"/>
      <property name="batchSize" value="0"/>
      <property name="autoConnect" value="false"/>
      <property name="useSchema" value="false"/>
      <property name="addTilde" value="false"/>
      <property name="needTransContent" value="false"/>
      <property name="needTransSentence" value="false"/>
      <property name="caseSentence" value="false"/>
   </DB>
</DBList>
```

1 Databases

A2 Admin will check if **mysql jdbc jar** file has been put in */home/docker/share/jdbc* directory; and will put *mysql-connector-java-xxx-bin.jar* under JDBC directory if not. The jar file will be copied under esProc JDBC’ s dependency package at the start of Docker.

A3 mysql.dfx script

	A	B
1	=connect("mysql")	
2	=A1.query("select pid,username,comment from t_user")	
3	=A1.close()	
4	return A5	

Execute steps A1-A3 on every server to make sure each node’ s proper operation.

A4 loadMsql.dfx script on client side:

	A	B
1	=callx("mysql.dfx"; ["192.168.0.76:8281"])	

2 External libraries

Requirement: Below are configurations for using esProc external libraries, such as MongoDB:

	Server	Docker container	Client
IP:PORT	192.168.0.76:8281	172.17.0.2:8281	192.168.0.186
<user>	/opt/app/aaa	/app	
dfx script	script/books.dfx	script/books.dfx	loadBooks.dfx
external library configuration	config/raqsoftConfig.xml		
mongo jar	/home/docker/share/extlib/MongoCli/*.jar	/share/MongoCli	

2 External libraries

D1 external library jar:
Admin will check if the desired external library jar(s) exists; and put it(them) in /home/docker/share/extlib directory if not. Below are jars needed by MongoCli external library:

```
root@master:/home/docker/share/extlib# ls MongoCli
mongocli.jar  mongodb-driver-3.10.1.jar  mongo-java-driver-3.9.1.jar
```

D2 configuration
Add MongoCli *in raqsoftConfig.xml*:

```
<extLibsPath>/share/extlib</extLibsPath>
        <importLibs>
          <lib>MongoCli</lib>
        </importLibs>
```

D3 Re-login Docker to load Mongo external library at the restart of esProc.

D4 books.dfx script:

	A
1	=mongo_open("mongodb://192.168.0.186:27017/user")
2	=mongo_shell(A1,"books.find()").fetch()
3	=A2.groups(addr,book;count(book): Count)
4	=A3.groups(addr;sum(Count):Total)
5	=A3.join(addr,A4:addr,Total)

B External libraries

D5 loadBooks.dfx script on client side:

	A	B
1	=callx("books.dfx"; ["192.168.0.76:8281"])	

Result:

addr	book	Count	Total
address1	book1	3	4
address1	book5	1	4
address15	book1	1	1
address2	book1	2	3
address2	book5	1	3
address3	book9	1	1
address4	book3	1	1

VI Cluster

Requirement: Get data meeting the condition STATE="New York " from employee.txt in Docker containers.

	Server A	Server B	Client
IP:PORT	192.168.0.76:8281	192.168.0.77:8282	192.168.0.186
<user>	/opt/app/aaa	/opt/app/aaa	
dfx script	script/Emp.dfx	script/Emp.dfx	loadEmp2.dfx
data	data/EMPLOYEE.txt	data/EMPLOYEE.txt	
Docker management	main server	subserver	
<Unit>	/home/docker/java/docker_users.xml	x	

Deployment on Server A:
A Check <Unit> configuration in *docker_users.xml*;
B Put EMPLOYEE.txt in user directory */opt/app/aaa/data*;
C Put Emp.dfx in suer directory */opt/app/aaa/script*.

Deployment on Server B: Same as step B and step C for Server A.

VI Cluster

A Emp.dfx script on server side:

	A	B
1	=file("/app/data/EMPLOYEE1.txt")	//Read the file
2	=A1.import@t(;pPart:pAll)	//Load data by segments
3	=A2.select(STATE==pState)	//pState is filtering condition
4	return A3	//Return a table sequence

B loadEmp2.dfx on client side:

	A	B
1	[192.168.0.76:8281,192.168.0.77:8282]	//Configure two nodes
2	=callx("Emp.dfx","New York",[1,2],[2,2];A1)	//Call *Emp.dfx* script
3	=A2.conj()	//Concatenate A2

A2: Divide data into two segments and query the first half for eligible data on node 192.168.0.76:8281 and the second half for eligible data on node 192.168.0.77:8282.
A3: Concatenate A2.

VII Summary

A Docker admin needs to configure Docker management, esProc cluster, external library JARs and user data after loading esProc mirror files.

users configure esProc as needed, provide data and dfx script and call the dfx file remotely to execute on the client side.

An esProc cluster needs to be configured on multiple servers to make sure all nodes work together to perform computations.