BFS-Cluster-API

整体架构

```
bfs-cluster-api          #提供bfs-api服务
  --bfs-pnode    #每个集群部署一个{需要提供对外服务端口：上传下载服务}
  --bfs-rnode    #每台机器部署一个，对接底层bfs-core
  --bfs-tnode    #每个集群部署一个{提供本集群检索服务}
  --bfs-tnDB     #每个集群部署一台，tnDB是为实现与其他集群的互联，部署后可下载其他集群数据



# 其中rnode-api封装底层bfs-core，因此需要每台机器一个rnode-api
# tnode-api负责监管和维护rnode-api，因此tnode配置文件中需要写入所负责的rnode-api的相关信息
# tnDB用于与其他集群的互联，部署后可以下载其他集群数据
# pnpde-api作为服务的提供者，供开发使用


# 流程（文件上传为例）：
用户通过pnode-api进行文件上传请求，pnode从tnode获取rnode相关性信息，然后pnode将文件上传到rnode，
rnode-api对接底层bfs-core，进行文件的编码以及存储。

（文件下载）
用户通过pnode-api进行文件上传请求，pnode从tnDB进行文件查找，如果是本集群直接下载，如果是其他集群，调用
其他集群的pn进行下载。
```

Installation

确保本系统已经安装bfs-core系统

```
pnode需要提供对外访问端口，提供对外服务
rnode,tnode只需要内部端口即可
##安装目录
--bfs-rnode
--bfs-tnode
--bfs-pnode
--bfs-tnDB
```

1 rnode installation

```
bfs-rnode/
  ├── rnode      #可执行程序
  └── conf.json #配置文件

{
    "port":"8202",                    #rnode访问端口
```

```
    "absAFSDir":"/aos/ks/afs_bin/",
    "afsProgram":"afs-x64",
    "apiDataFolder":"afs_data",
    "maxExecTime":60,
    "rnid":"9237",                    #rnid【需要针对底层更改】
    "Log":true
}

Usage:

使用screen进行后台挂起{screen便于后期查看}：需要下载screen/或者直接 "nohup ./rnode"
screen -S rnode    #创建screen
[chmod +x rnode 赋予可执行]
./rnode            #执行rnode
ctrl +AD           #后台挂起
```

2 tnode installation

```
bfs-tode/
  ├── tnode      #可执行程序
  └── conf.json #配置文件

{
  "port":"8203",                      #访问端口，建议保持
  "rnids":["9300","9301","9302"],     #当前集群rnodes的标识【与上rnode installation保持一致】

  "rnodeRoots":
["http://172.17.39.226:8202/","http://172.17.100.118:8202/","http://172.17.100.117:8202/"
],  #对应rnid的访问地址，即1步骤中rnode安装的所在集群ip+设置的port
  "apiDataFolder":"afs_data",
  "log":true
}

Usage:
使用screen进行后台挂起
screen -S tnode    #创建screen
./tnode            #执行rnode
ctrl +AD           #后台挂起
```

3 pnode installation

```
bfs-pnode/
  ├── pnode      #可执行程序
  └── conf.json #配置文件

{
    "port":"8201",                                #访问端口
    "absAFSDir":"/aos/ks/afs_bin/",               #bfs-core可执行程序所在的位置，一般默认即可
    "afsProgram":"afs-x64",
```

```
    "apiDataFolder":"afs_data",                #log等相关信息存储目录，默认即可
    "maxExecTime":60,
    "trackRoots":["http://127.0.0.1:8203/"],  #tnode的访问端口
    "tnDB":"http://*.*.*.*:8049/",        #tnDB的访问端口 {见后tnDB installation}
    "log":true,
}

Usage:

使用screen进行后台挂起
screen -S pnode    #创建screen
./pnode            #执行rnode
ctrl +AD           #后台挂起
```

4 tnDB installation

```
包括两个安装文件
  ---bfs.tn.api
  ---bfs.tn.cron
```

4.1 bfs.tn.cron installation

install mysql 5.7

A Quick Guide to Using the MySQL APT Repository
A Quick Guide to Using the MySQL Yum Repository

create a user

e.g.

```
create user 'bfdb'@'%' identified by 'test123';
grant all privileges on *.* to 'bfdb'@'%';
flush privileges;
```

install mongodb 4.x

Install MongoDB Community Edition on Ubuntu
Install MongoDB Community Edition on Red Hat or CentOS

create a user

e.g.

```
use admin;
db.createUser({
    user:"bfdb",
    pwd:"test123",
    roles:["root"]
})
```

install node.js v10

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.2/install.sh | bash
nvm install 10 # install node.js v10
```

prerequisites

mysql user(grant all privileges)
mongodb user with root role(or no auth)

```
npm i pm2 -g
pm2 install pm2-logrotate
pm2 set # config log-rotate, e.g. max_size=1M, retain=10
pm2 update # restart pm2
```

run this program

```
cd bfs.tn/dist
cp config.js.example config.js # !!!then edit this carefully!!!
cd ..
cp ecosystem.config.js.example ecosystem.config.js
npm i --production # install only production dependencies
pm2 start ecosystem.config.js
pm2 list # after few minutes check if it start correctly
pm2 logs # auto print logs of pm2 processes
```

```
    ##confg.js需要修改的

    exports.mongodbUri = 'mongodb://bfdb:c10ud-12345@127.0.0.1:27017/admin';
    exports.knexConfig = {
        client: 'mysql',
        // mysql, 需要管理员账户
        connection: {
            host: '127.0.0.1',
            port: 3306,
            user: 'bfdb',
            password: 'c10ud-12345',
            database: 'mysql',
        },
        pool: {
            afterCreate(conn, done) {
                console.log('afterCreate');
```

```
                conn.query('SELECT 1', e => {
                    done(e, conn);
                });
            },
        },
    };
    // 本API端口，按需修改
    exports.port = 8049;
    // 本集群rnode端口，询问部署rnode的人
    // 本集群rnode端口：即上面2.1 rnode的服务端口[保证每个rnode都选择这个端口]
    exports.rnPort = 8202;
    // 本集群pnode外网地址，array格式
    // 本集群pnode地址：即上面2.3 pnode的地址（IP+端口），部署PN的机器，公网地址
    exports.pnodes = ['http://119.8.40.81:8201'];
    // 本机外网地址，其他集群需要访问本API
    exports.selfUrl = 'http://159.138.1.232:8103';
    // 启动之后先通知这个tn，详情询问初始集群负责人[如果为空，即不与其他集群互连]
    exports.firstTn = {
        cluster: '',
        url: '',
    };
```

stop this program

```
pm2 list # show your processes
pm2 stop all # stop processes without delete records
pm2 delete all # stop processes and delete records
```