Pyweb 代码分发

让运维工作变得更加简单

1、获取代码 pyweb：

git clone https://github.com/jonnywang/pyweb.git

2、安装 py 模块：

yum install python-pip

pip install redis

pip install tornado

pip install pymongo

pip install kazoo

3、修改相关配置：

cd src/

vi main.py

```
settings = {
    'debug'           : True,
    'cookie_secret'   : 'ZyVB9oXwQt8S0ROkRvJ5/bZJc2sWuQLTos6GkHn/todo=',
    'static_path'     : os.path.dirname(__file__) + '/static',
    'template_path'   : os.path.dirname(__file__) + '/tpl',
    'ui_modules'      : {},
    'redis_server'    : {'host' : '192.168.170.8', 'port' : 6379, 'db' : 1},
    'mongo_server'    : {'host' : '192.168.170.8', 'port' : 27017},
    'zookeeper_server' : {'host' : '192.168.170.8', 'port' : 2181, 'root_node' : '/test'},
}
```

网页运行需要设置 redis/mongo/zookeeper 地址

vi LoginHandler.py

```
        if username == 'admin' and password == 'todo':
            self.session.init({'name':username,'time':Tools.g_time()})
            self.redirect('/')
```

可修改页面登陆的帐号和密码

vi static/js/app.common.js

```
App.runSocket = function() {
    var _this = this;
    this.ws = new WebSocket("ws://[redacted]:8888/socket")
    this.ws.onopen = function() {
        _this.connected = true;
        _this.debug('connected server')
        _this.loadServers();
    }
```

修改 socket 地址（此地址一定是外网访问地址）

vi SocketHandler.py　　# return True

```
  def check_origin(self, origin):
      #return urlparse.urlparse(origin).netloc.lower() == '127.0.0.1:8888'
      return True
```

4、运行 pyweb：

/usr/local/pyweb/src/main.py

[INFO 2015-04-24 12:46:02 connection.py 569] Connecting to 192.168.1.71:2181

[INFO 2015-04-24 12:46:02 client.py 435] Zookeeper connection established, state: CONNECTED

[INFO 2015-04-24 12:46:02 Publish.py 56] refresh server list {"update_time": 1429848213.81111, "server_id": "1", "server_name": "s1"}

调试成功后，可用 supervisor 进行管理

$ cat /etc/supervisord.conf.d/pyweb.ini

[program:pyweb]

stdout_logfile=/var/log/supervisor/pyweb_stdout.log

stderr_logfile=/var/log/supervisor/pyweb_stderr.log

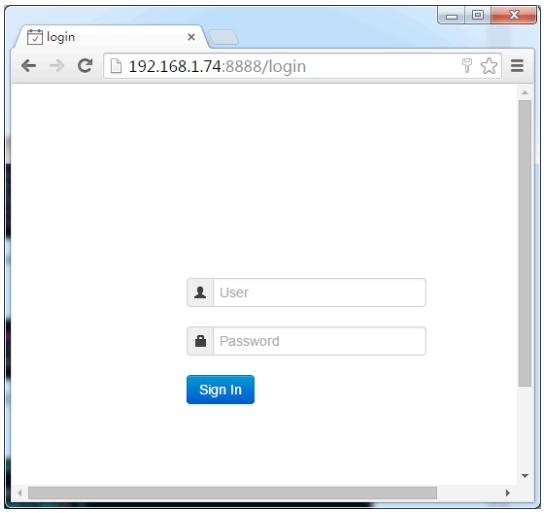
command=/data/pyweb/src/main.py

user=root

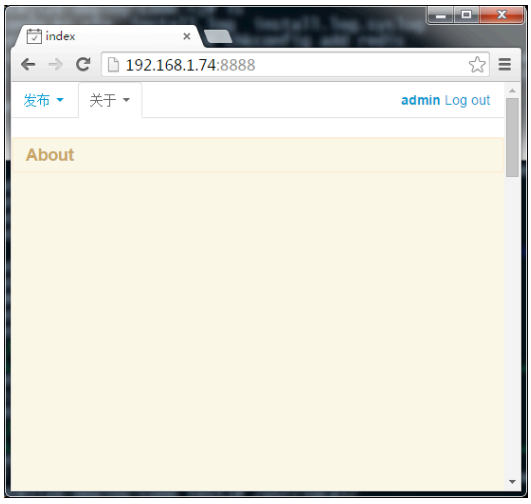
autostart=true

autorestart=true

startsecs=3



输入帐号密码 admin/todo



拉取代码（server 端和 pyweb 部署在一台服务器上），暂且称其为拉取代码的过程，启动如下：

/usr/local/pyweb/test/mock_zip.py 192.168.170.8

（mock_zip 触发运行 zip.sh 脚本，由 zip 拉取、打包等等各种操作，灵活性较大）

你也可以使用 supervisor 进行管理，配置文件如下：

$ cat /etc/supervisord.conf.d/mock_zip.ini

[program:mock_zip]

stdout_logfile=/var/log/supervisor/mock_zip_stdout.log

stderr_logfile=/var/log/supervisor/mock_zip_stderr.log

command=/data/pyweb/test/mock_zip.py 192.168.170.8

user=root

autostart=true

autorestart=true

startsecs=3

对于 zip.sh 灵活性较大，可以根据自己的习惯进行工作，比如从 git 或者 svn 上进行拉取代码、打包等操作：

```
#!/bin/sh
#
# exit 0 for success other for failed
#
#

if [ $# -ne 2 ] ; then
    echo "Error params"
    exit 1
fi

CONFIG_VERSION=$1
GAME_VERSION=$2

echo "got config_version=${CONFIG_VERSION} game_version=${GAME_VERSION}"

PULL_DIR=/data/salt/gamecode

cd $PULL_DIR
/usr/bin/git clone [redacted]:/var/git/poker_php_server.git
/usr/bin/git clone [redacted]:/var/git/poker_sql.git
/usr/bin/git clone [redacted]:/var/git/poker_prd_config.git
find ./ -name .git -a -type d -prune -o -type f ! -name "md5.txt" -exec md5sum {} \; > md5.txt

exit 0
```

通过 salt 将 single_syc.py 、syc.sh、pub.sh 、single_pub.py 同步到相应节点（或称 client）：

修改 single_syc.py

```
class mock_syc(object):

    _server_id   = '1'
    _root_node   = ''
    _zookeeper   = None
    _shell_path  = ''
```

将_server_id 设置成 1 这个字符串，注意：一定是数字

运行：

/usr/local/pyweb/test/single_syc.py 192.168.170.8

也可以使用 supervisor 进行管理：

$ cat /etc/supervisord.conf.d/single_syc.ini

[program:single_syc]
stdout_logfile=/var/log/supervisor/single_syc_stdout.log
stderr_logfile=/var/log/supervisor/single_syc_stderr.log
command=/usr/local/pyweb/test/single_syc.py 192.168.170.8
user=root
autostart=true
autorestart=true
startsecs=3

脚本 syc.sh 如下：

```
#!/bin/sh
#
# exit 0 for success other for failed
#
#

if [ $# -ne 3 ] ; then
    echo "Error params"
    exit 1
fi

CONFIG_VERSION=$1
GAME_VERSION=$2
SERVER_ID=$3

echo "got config_version=${CONFIG_VERSION} game_version=${GAME_VERSION} target_server=${SERVER_ID}"

new_dir=$(date +%Y%m%d_%H%M%S)
mkdir -p /data/deploy/$new_dir
rsync -rlpgoDuP --exclude=.git jize@192.168.170.8::code /data/deploy/$new_dir --password-file=/etc/rsyncd.pass

exit 0
```

在这里，我在 server 端做了一个 rsync 服务，这样各个节点可同时进行同步工作。

发布：
pub.sh　single_pub.py 这两个文件，我们称为发布脚本，修改 single_pub.py

```
class mock_pub(object):

    _server_id   = '1' #str
    _root_node   = ''
    _zookeeper   = None
    _shell_path  = ''
```

_server_id 修改为 1（一定是数字），默认空

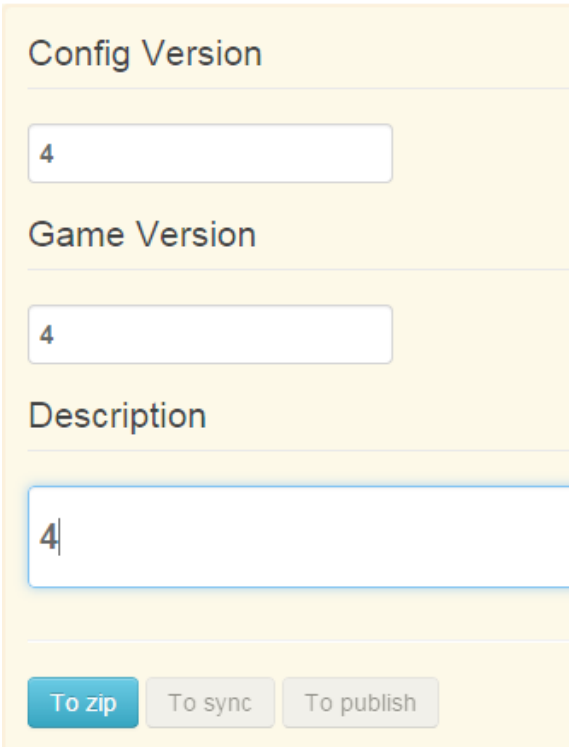
启动：
$ ./single_pub.py 192.168.170.8
使用 upervisor 管理：
$ cat /etc/supervisord.conf.d/single_pub.ini
[program:single_pub]
stdout_logfile=/var/log/supervisor/single_pub_stdout.log
stderr_logfile=/var/log/supervisor/single_pub_stderr.log
command=/usr/local/pyweb/test/single_pub.py 192.168.170.8
user=root
autostart=true
autorestart=true

startsecs=3

发布 pub.sh 如下：

```
#!/bin/sh
#
# exit 0 for success other for failed
#
#

if [ $# -ne 3 ] ; then
    echo "Error params"
    exit 1
fi

CONFIG_VERSION=$1
GAME_VERSION=$2
SERVER_ID=$3

echo "got config_version=${CONFIG_VERSION} game_version=${GAME_VERSION} target_server=${SERVER_ID}"


check_md5()
{
    md5sum -c --status md5.txt
}

time=$(date "+%Y-%m-%d %H:%M:%S")
new_dir=$(ls -ltr /data/deploy/ | tail -1 | awk -F ' ' '{print $9}')
echo $new_dir
cd /data/deploy/$new_dir
rm -rf /data/www/code
if check_md5; then
    ln -s /data/deploy/$new_dir /data/www/code
else
    echo "$time         check md5 is faild" >> /var/log/supervisor/check_md5_stderr.log
fi

exit 0
```

登陆网页操作：



点击：To zip

选择服务器并 To sync

Config Version

4

Game Version

4

Description

4

Servers

ALL
s1 s2 s3

Finished zip | To sync | To publish

· start zip (pub id=4)

· zip finished

Config Version

4

Game Version

4

Description

4

Servers

ALL
s1 s2 s3

Finished zip | Finished syc | To publish

· start zip (pub id=4)

· zip finished

· start sync

· server s2 sync finished

· server s1 sync finished

· server s3 sync finished

· sync all finished

最后一步 Finished pub

执行完成后，让我们看一下，节点的代码发布情况：

```
$ ll -h /data/deploy/
total 16K
drwxr-xr-x 5 root root 4.0K May 15 09:47 20150515_094734
drwxr-xr-x 5 root root 4.0K May 15 09:58 20150515_095808
drwxr-xr-x 5 root root 4.0K May 15 09:59 20150515_095936
drwxr-xr-x 5 root root 4.0K May 15 10:59 20150515_105935
```

```
$ ll -h /data/www/
total 4.0K
lrwxrwxrwx 1 root root 28 May 15 11:00 code -> /data/deploy/20150515_105935
-rw-r--r-- 1 root root 16 May 13 19:19 index.html
```

可以看到，我们将 git 上面拉取下来的代码 同步到/data/deploy/日期_时间/目录下,并创建软链接到/data/www/code 下，操作完成!

日志如下：

History

vid	config version	game version	desc	target servers	status	pub time
v4	4	4	4	1,2,3	pub_success	2015-05-15 10:58:58
v3	3	3	3	1,2,3	pub_success	2015-05-15 09:56:24
v2	2	2	第二次发布测试	1,2,3	pub_success	2015-05-15 09:55:13
v1	1	1	aa	1,2,3	pub_success	2015-05-15 09:31:36

附：rsync 服务
创建 rsync 服务：
cat /etc/rsyncd.conf
#global
secrets file = /etc/rsyncd.secrets
motd file = /etc/rsyncd.motd
read only = no
list = no
uid = root
gid = root
#hosts allow = 192.168.1.62
max connections = 20
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
ignore errors
slp refresh = 300

#modules
[code]
comment = jize’s code sync
path = /data/release/
auth users = jize
注释：架设 rsync 服务是方便客户端同时过来同步代码（个人认为，它是一对多，不是一对一）

感谢同学 QQ122167504 的整理！