ocfs2 自动化测试套件

ocfs2自动化测试套件，目前部署在CI服务器上。并由一些脚本触发执行。

http://10.67.160.200:8080/job/00-Milestone/job/SLE12-SP4/job/OCFS/job/0-deploy/configure

该套件都以一个文件夹的形式呈现。

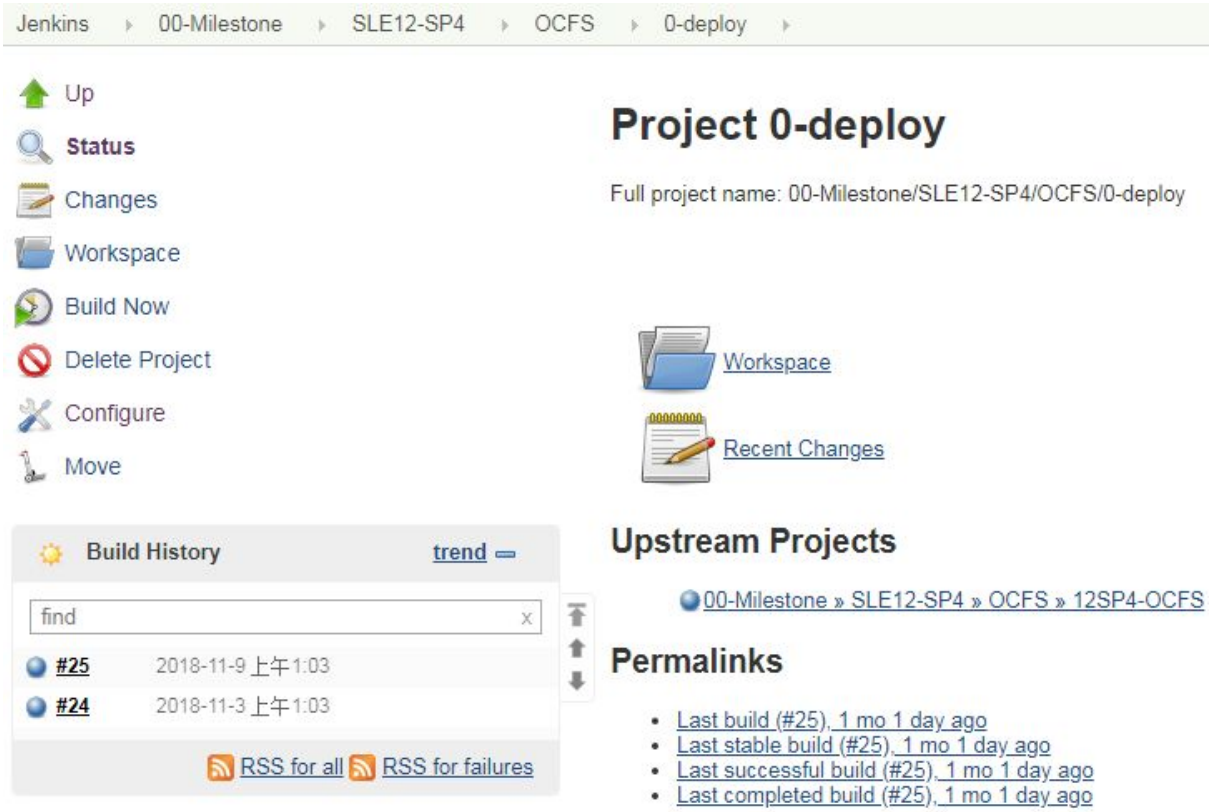
套件构成

1) 12SP4-OCFS2
2) MonitorBuildChange-ocfs2-sle12sp4
3) 0-deploy
4) 1-setup
5) 2-run

1，2 两项是nick帮忙加入用来定时触发执行的。不用深究
3. 0-deploy 用来安装部署虚拟机
4. 1-setup 用来安装HA以及ocfs2 相关的包
5. 2-run 用来执行测试套件

如何配置

以0-deploy为例，进入上述链接，点击0-deploy，进入如下界面

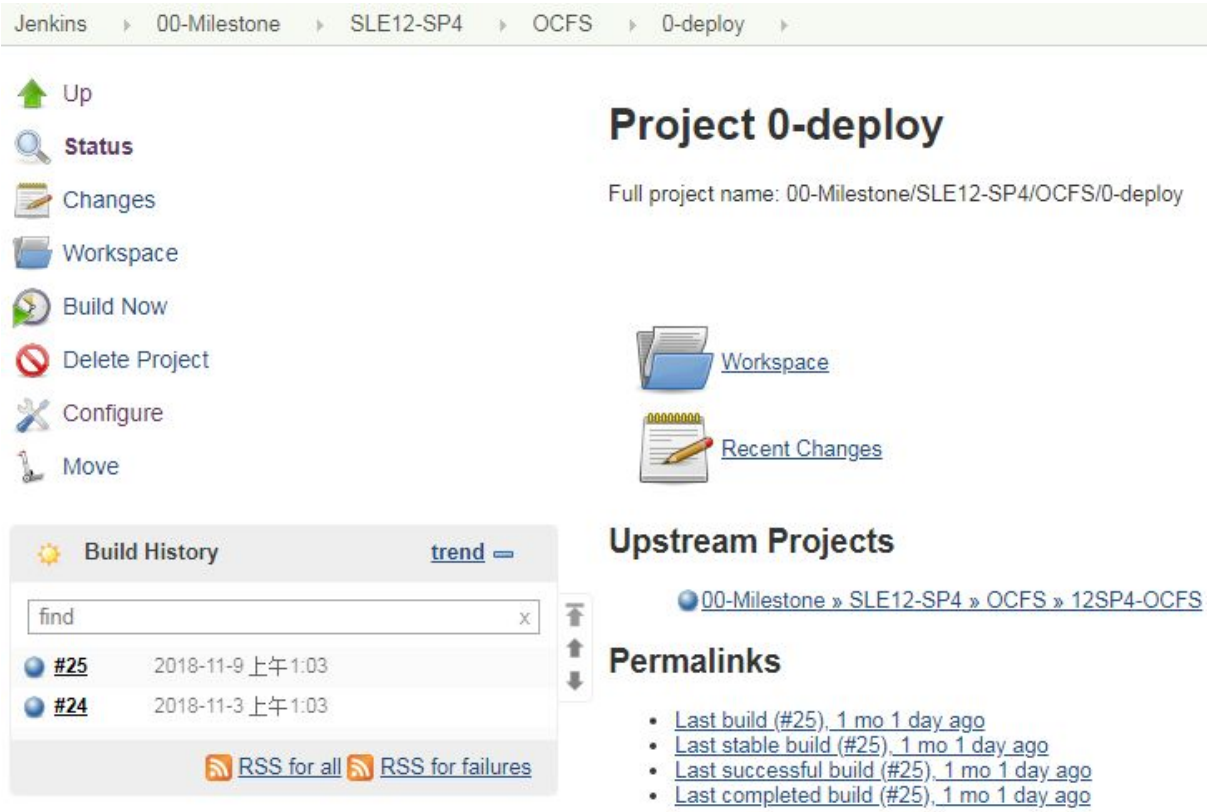


点击Configure 进入配置界面

如何运行

运行一个任务

以0-deploy为例



点击 Build Now（或者Build with Parameters），Jenkins会按照Configure中的配置情况执行任务

运行整个测试套件

为了方便，我们把整个套件拆成了三个任务，分别是0-deploy，1-setup 和 2-run
请针对每个任务执行上述运行操作，完成整个测试流程

套件使用说明

0-deploy

目的

1. 构建虚拟机
2. 配置共享磁盘或者iscsi等

配置

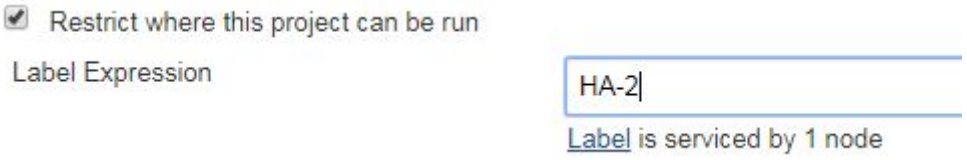
1. 进入0-deploy的配置界面
2. 修改环境变量，与shell语法类似

Properties Content

```
LIB_DIR=/tmp/jenkins-work/ha-share/deploy/scripts
WORK_DIR=/mnt/vm/larry/jenkins-work
CONFIG_CLUSTER_DIR=/tmp/cluster-configuration
CLUSTER_FILE=${WORK_DIR}/cluster_conf
YAML_FILE=${WORK_DIR}/vm_list.yaml
BUILD_LOG_DIR=${WORKSPACE}/${BUILD_NUMBER}

RAW_FILE=/mnt/vm/larry/ocfs2-test-disk.do.not.remove
AS_IN_GUEST=vdb
ISCSI_SERVER=10.67.162.97
ISCSI_TARGET=iqn.2016-04.world.srv:storage.target00
```

3. 选择哪个host用来执行该项目



4. 配置要执行的命令

Execute shell

Command

```
# Create the dirs
mkdir -p ${WORK_DIR}

# Create YAML_FILE
cat > ${YAML_FILE} << !EOF!

resources:
  sle_source: http://mirror.suse.asia/dist/install/SLP/SLE-12-SP4-Server-LATEST/x86_64/DVD1/
  ha_source:  http://mirror.suse.asia/dist/install/SLP/SLE-12-SP4-HA-LATEST/x86_64/DVD1/

nodes:
  - name: lchen-sle12-sp4-node1
  - name: lchen-sle12-sp4-node2
  #- name: lchen-sle15-rc2-node2
  #- name: lchen-sle15-rc2-node3



repos:
  - repo: https://download.opensuse.org/repositories/network:/ha-clustering:/Factory/openSUSE_Tumbleweed/
  - repo: http://mirror.suse.asia/dist/install/SLP/SLE-15-Packages-RC2/x86_64/dvd1/Module-Legacy/


#devices:
#  disk_dir: /mnt/vm/larry
#  nic: virbr1
#  second_nic: br0
#  disk_size: 61440

devices:
  disk_dir: /mnt/vm/larry
  nic: br0
  disk_size: 61440
  sharing_backing_file: no


!EOF!
#iscsi:
#  target_ip:
#  shared_target_luns:
#  - shared_target_lun: iqn.2018-01.suse.com:larry-jenkins-test-sbd
#    shared_target_ip:
#sharing_backing_file: yes
```

如图，WORK_DIR在第三步中已经定义
然后向YML_FILE中写入相关的配置信息，其中包括
resources:
源名称及对应的源，用来装系统

nodes:
虚拟机的名字

repos:
一些其他的repos 用户自定义的， 如HA的源

devices:

disk_dir	虚拟机镜像所在路径
nic	虚拟机用哪个网桥
disk_size	虚拟机镜像的大小
sharing_backing_file	保持不动吧，这个是nick给的用来加速的，咱们不在乎那点速度，老老实实按龟速装吧

cleanVM.py	用来清除已经重名的虚拟机
installVM.py	安装虚拟机
getClusterConf.py	安装完虚拟机后，会把安装好的虚拟机的相关信息写入 CLUSTER_FILE里 其中包含了每个虚拟机的ip 注：虚拟机内的hostname和虚拟机名是一样的

attach-disk.sh	用来创建共享磁盘 如图所示，有两条语句，下面一条被注释了 第一条表示创建一个本地的raw文件，用来作为虚拟机之间的共享块设备 第二条表明通过iscsi协议连接远程的LUN，使得虚拟机之间有共享块设备

1-setup

目的

完成相关的安装包任务

配置环境变量

Properties Content

```
LIB_DIR=/tmp/jenkins-work/ha-share/deploy/scripts/
WORK_DIR=/mnt/vm/larry/jenkins-work
CLUSTER_FILE=${WORK_DIR}/cluster_conf
OCFS2_DIR=${LIB_DIR}/ocfs2/
SCRIPTS_RUN_IN_GUEST=${OCFS2_DIR}/guest/
OCFS2_TEST_DIR=/var/lib/ocfs2test/

BUILD_LOG_DIR=${WORKSPACE}/${BUILD_NUMBER}
```

执行命令

Execute shell

Command

```
# for ocfs2
cd ${OCFS2_DIR}

# install vanilla kernel
#./installKernel.sh --cluster-config=${CLUSTER_CONF}

sleep 3

# copy scripts to /var/lib/ocfs2test/ in guests
./scpFils2Guest.sh --cluster-config=${CLUSTER_FILE} ${SCRIPTS_RUN_IN_GUEST} ${OCFS2_TEST_DIR}

# ssh passwordless among cluster nodes for ocfs2test user
./sshTestUsr.sh --cluster-config=${CLUSTER_FILE}

# setup env for ocfs2 test
./runProg.sh --cluster-config=${CLUSTER_FILE} /var/lib/ocfs2test/prepTestEnv.sh
```

scpFiles2Guest.sh用来把host上的测试脚本拷贝到虚拟机里面，前面说过虚拟机的相关信息全部放在CLUSTER_FILE里面

后面两个是用来执行相关的安装任务的，包括安装ocfs2-test ocfs2-tools 和HA 相关的软件包

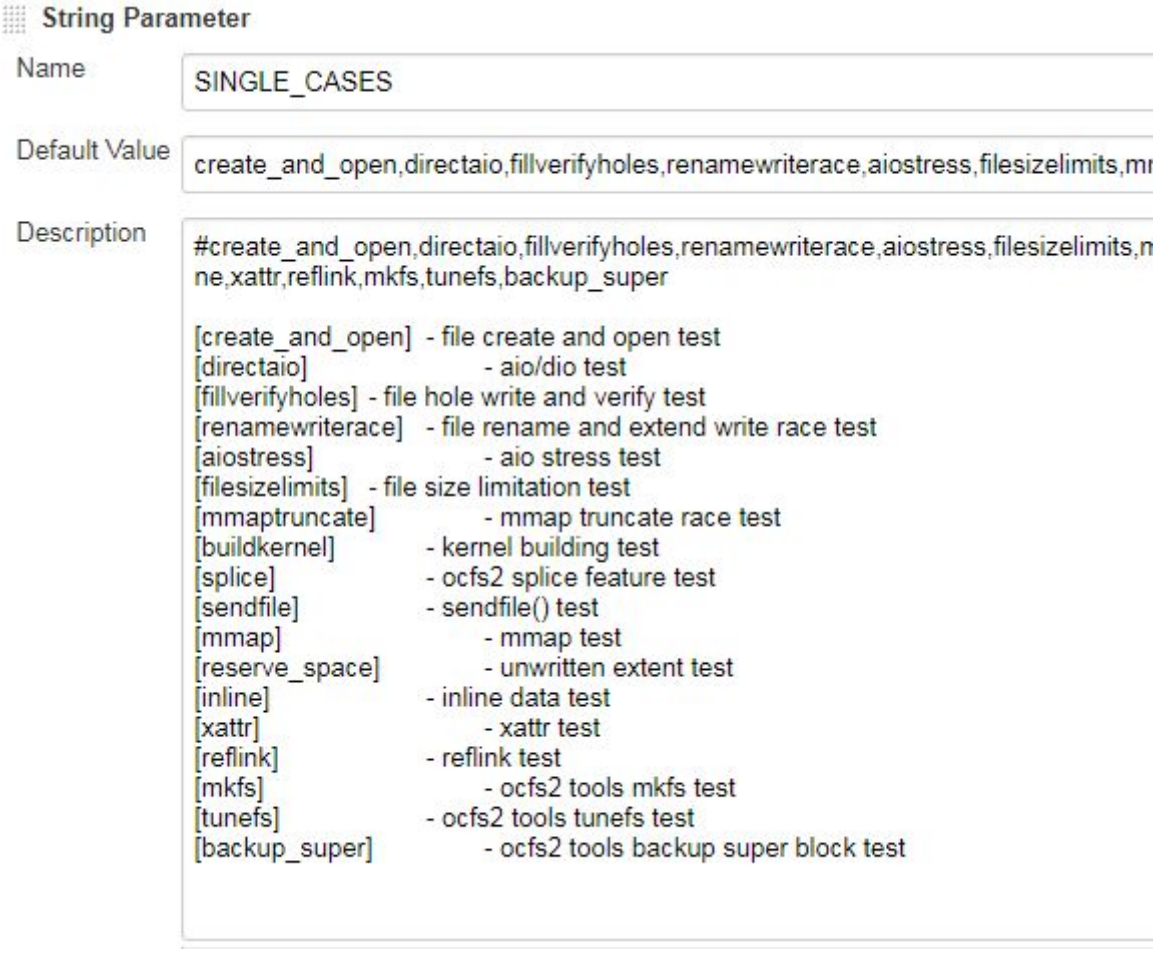
2-run

目的

这个任务是用来执行具体的测试套件的

配置测试用例参数

其中的配置非常详细，以SINGLE_CASES为例



Default Value表明要跑的测试集合，以“,” 分隔

配置环境变量

Properties File Path

```
${WORK_DIR}/cluster_conf
```

Properties Content

```
LIB_DIR=/tmp/jenkins-work/ha-share/deploy/scripts/
CLUSTER_CONF=${WORK_DIR}/cluster_conf

OCFS2_DIR=${LIB_DIR}/ocfs2/
OCFS2_TEST_DIR=/var/lib/ocfs2test/
TEST_CONFIG=${OCFS2_TEST_DIR}/test_config
TEST_CONFIG_IN_HOST=${WORK_DIR}/test_config

OCFS2_INSTALL_DIR=/usr/local/ocfs2-test/

WORK_DIR=/tmp/jenkins-work/${JOB_NAME}/${BUILD_NUMBER}
BUILD_LOG_DIR=${WORKSPACE}/${BUILD_NUMBER}
```

执行测试任务

Execute shell

Command

```
# dump variables to TEST_CONFIG file
mkdir -p ${WORK_DIR}
echo ${TEST_CONFIG_IN_HOST}
# get the host ip address
HOST_IP=$(ip address | grep -A 2 "state UP" | grep -m 1 "10.67.[160|161]" | awk -F'[/ ]+' '{print $3}')

cat > ${TEST_CONFIG_IN_HOST} << !EOF!
BLOCK_SIZE=${BLOCK_SIZE}
CLUSTER_SIZE=${CLUSTER_SIZE}
TESTMODE=${TESTMODE}
SINGLE_CASES=${SINGLE_CASES}
MULTIPLE_CASES=${MULTIPLE_CASES}
CLUSTER_STACK=${CLUSTER_STACK}
CLUSTER_NAME=${CLUSTER_NAME}
SHARED_DISK=${SHARED_DISK}
MOUNT_POINT=${MOUNT_POINT}
KERNEL_SOURCE=${KERNEL_SOURCE}
HOST_IP=${HOST_IP}
!EOF!

# scp TEST_CONFIG to guest
cd ${OCFS2_DIR}
./scpFils2Guest.sh --cluster-config=${CLUSTER_CONF} ${TEST_CONFIG_IN_HOST} ${OCFS2_TEST_DIR}

# scp kernel source to guest
./scpFils2Guest.sh --cluster-config=${CLUSTER_CONF} ${KERNEL_SOURCE} ${OCFS2_INSTALL_DIR}/tmp/
```

Execute shell

Command

```
# run ocfs2-test
cd ${OCFS2_DIR}
./runProg.sh --cluster-config=${CLUSTER_CONF} --test-config=${TEST_CONFIG} --this-node=${IP_NODE1} ${OCFS2_TEST_DIR}/runTest.sh

# take care of test logs
./downloadTestLogs.sh ${IP_NODE1} ${BUILD_LOG_DIR}
./genTestReport.py ${BUILD_LOG_DIR}
```

See the list of available environment variables

Execute shell

Command

```
# sort out the test result
cd ${LIB_DIR}

./runCases.py -f ${CLUSTER_CONF} -d ${BUILD_LOG_DIR}/test_reports -r "parseOCFS2Report.py"
```