```
#!/bin/bash
#
#  This script will create a script which can be used to setup virtualbox VMs
#    for a RAC/GI lab
#
#  Input Parameters are:
#             - the name of the lab
#             - the number of participants expected for the lab
#             - the number of VMs a participant will get, the default is 2
#
#  Example call:
#    ./create_lab.sh mylab 2
#
#    This will create a lot files in the /var/www/html directory. Some of the files, e.g.
#      the hostfile, ssh keys are fetched when kickstarting the VMs
#
#    TODO:
#       - improve exception handling
#       - improve usage sub
#       - add windows support
#       - remove hard codings (ip subnet, oracle version, etc )
#

# the name of the lab
LABNAME=${1}
# the number of participants expected for the lab
NUMBER_OF_PARTICIPANTS=${2}
# the number of VMs per participant, the default is 2
NUMBER_OF_VMS_PER_PARTICIPANT=${3}

# the prefix which will be used for the names of nodes
SYSTEM_PREFIX=`echo ${LABNAME} | awk '{print tolower($LABNAME)}'`
# the cobbler profile which will be used for the nodes
PROFILE=OracleDatabaseServer
# the netmask for the network configuration
NETMASK="255.255.255.0"
# start of the public ip-addresses ( 192.168.56.110 )
IPPUBSTART=110
# start of the public virtual ip-addresses ( 192.168.56.170 )
IPVIPPUBSTART=170
# start of the scan address ( 192.168.56.231 )
IPSCANSTART=231
# start of the private ip-addresses ( 10.1.1.1 )
IPPRIVSTART=1
# the base directory for storing all the files generated
BASEDIR="/var/www/html/${LABNAME}"
# the cobbler server
COBBLERSERVER="192.168.56.101"

# print the usage of this script
```

```
_usage() {
  echo " "
  echo "*******************************************************************************************"
  echo " Usage of this script                                                                      *"
  echo " $0 [LABNAME] [NUMBER_OF_PARTICIPANTS] [NUMBER_OF_VMS_PER_PARTICIPANT]    *"
  echo "*******************************************************************************************"
  echo " "
  exit 1
}

# some basic checks on the parameters
_do_checks() {
  # we need a lab name
  if [ "${LABNAME}" == "" ]; then
    _usage
  fi
  # we need the number of participants
  if [ "${NUMBER_OF_PARTICIPANTS}" == "" ]; then
    _usage
  fi
  # default the number of VMs per participant to 2, if nothing is provided
  #  can be any number as long as the ip range permits it
  if [ "${NUMBER_OF_VMS_PER_PARTICIPANT}" == "" ]; then
    NUMBER_OF_VMS_PER_PARTICIPANT=2
  fi
}

# setup the lab
_setup_lab() {
  mkdir -p ${BASEDIR}
  # for each participant...
  for (( i=1; i <= ${NUMBER_OF_PARTICIPANTS}; i++ ))
  do
    # the filename for the vm setup script
    VBSETUPSCRIPT="${BASEDIR}/setup_p_${i}.sh"
    # cleanup any previous generated setup scripts
    rm -f ${VBSETUPSCRIPT}
    # array holding the vm names
    VMNAMES=
    echo "Creating ISOs and VM Setup Script for Participant $i"
    # for the amount of vms per participant, do ....
    for (( e=1; e <= ${NUMBER_OF_VMS_PER_PARTICIPANT}; e++ ))
    do
     # the name the node will get in cobbler as well as the OS itself
     SYSTEM_NAME=${SYSTEM_PREFIX}p${i}vm${e}
     # the name of the host file which will be generated per vm
     HOSTFILE=${SYSTEM_PREFIX}p${i}vm${e}_HOSTS
     echo "  creating VM ${e}, name ${SYSTEM_NAME}"
     # add the system to cobbler
     cobbler system add --name=${SYSTEM_NAME} --profile=${PROFILE}
```

```
    # add the public interface and hostname
    cobbler system edit --name=${SYSTEM_NAME} --interface=eth0 \
                        --ip-address=192.168.56.${IPPUBSTART}  \
                        --netmask=${NETMASK} \
                        --hostname=${SYSTEM_NAME} \
                        --static=1 --dns-name=${SYSTEM_NAME}.lab.ch
    # add the private interface
    cobbler system edit --name=${SYSTEM_NAME} --interface=eth1 \
                        --ip-address=10.1.1.${IPPRIVSTART} \
                        --netmask=${NETMASK} \
                        --static=1
    # generate the boot iso
    cobbler buildiso --systems="${SYSTEM_NAME}" \
                     --iso="${BASEDIR}/${SYSTEM_NAME}.iso" 2>1 > /dev/null
    # create the vm setup script
    echo "VM_NAME=${SYSTEM_NAME}" >> ${VBSETUPSCRIPT}
    echo "VM_SHARED_DISKS=\${HOME}/Downloads/" >> ${VBSETUPSCRIPT}
    echo "VM_HOME=\${HOME}/Downloads/\${VM_NAME}" >> ${VBSETUPSCRIPT}
    echo "HD=\${VM_HOME}/hd_root.vdi" >> ${VBSETUPSCRIPT}
    echo "BOOTISONAME=${SYSTEM_NAME}.iso" >> ${VBSETUPSCRIPT}
    echo "BOOTISOURL=\"http://${COBBLERSERVER}/${LABNAME}/\${BOOTISONAME}\"" >> ${VBSETUPSCRIPT}
    echo "vboxmanage unregistervm \${VM_NAME} --delete >> /dev/null 2>&1" >> ${VBSETUPSCRIPT}
    echo "vboxmanage closemedium disk \${HD} --delete >> /dev/null 2>&1" >> ${VBSETUPSCRIPT}
    echo "rm -f \"\${VM_HOME}/*\"" >> ${VBSETUPSCRIPT}
    echo "vboxmanage createvm --name \${VM_NAME}  --register --ostype Oracle_64" >> ${VBSETUPSCRIPT}
    # create the boot order for the vm
    echo "vboxmanage modifyvm \${VM_NAME} --boot1 disk" >> ${VBSETUPSCRIPT}
    echo "vboxmanage modifyvm \${VM_NAME} --boot2 dvd" >> ${VBSETUPSCRIPT}
    # create the root disk (which will hold everything in this lab setup. in production systems there will
    #   be seperate disks, or at least seperate lvs )
    echo "vboxmanage createhd --filename \"\${HD}\" --size 102400 --format VDI --variant Standard" >> ${VBSETUPSCRIPT}
    # add a storage controller
    echo "vboxmanage storagectl \${VM_NAME} --name ctl1 --add sata" >> ${VBSETUPSCRIPT}
    # attach the root disk to the storage controller
    echo "vboxmanage storageattach \${VM_NAME} --storagectl ctl1 --type hdd --medium \"\${HD}\" --port 1" >> ${VBSETUPSCRIPT}
    # get the boot iso to boot from
    echo "cd \"\${VM_HOME}\"" >> ${VBSETUPSCRIPT}
    echo "rm -f \${BOOTISONAME}" >> ${VBSETUPSCRIPT}
    echo "wget \${BOOTISOURL}" >> ${VBSETUPSCRIPT}
    # add a storage controller for attaching the boot.iso
    echo "vboxmanage storagectl \${VM_NAME} --name ctl2 --add ide" >> ${VBSETUPSCRIPT}
    # ... and attach the iso we generated above
    echo "vboxmanage storageattach \${VM_NAME} --storagectl ctl2 --type dvddrive --port 1 --device 1 --medium \"\${VM_HOME}/\${BOOTISONAME}\"" >> $
{VBSETUPSCRIPT}
    # set memory, cpu and network parameters
    echo "vboxmanage modifyvm \${VM_NAME} --memory 2048 --cpus 1 --nic1 hostonly --cableconnected1 on --hostonlyadapter1 vboxnet0 \
                                        --nic2 intnet --cableconnected2 on"  >> ${VBSETUPSCRIPT}
    # we do not need audio and usb
    echo "vboxmanage modifyvm \${VM_NAME} --audio none"  >> ${VBSETUPSCRIPT}
    echo "vboxmanage modifyvm \${VM_NAME} --usb off"  >> ${VBSETUPSCRIPT}
```

```
 # create the host file which is required by the oracle installer
 echo "127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4" > ${BASEDIR}/${HOSTFILE}
 # increase ip variables for the next vm of this participant
 let IPPUBSTART=IPPUBSTART+1
 let IPPRIVSTART=IPPRIVSTART+1
 let IPVIPPUBSTART=IPVIPPUBSTART+1
done
# save the last assigned ips
# this is required for creating the /etc/hosts file below
let LASTASSIGNEDIP=IPPUBSTART-1
let LASTASSINGNEDINTERCONNECTIP=IPPRIVSTART-1
let LASTASSINGNEDVIPIP=IPVIPPUBSTART-1
# creating shared storage devices
if [ ${NUMBER_OF_VMS_PER_PARTICIPANT} -gt 1 ]; then
  # create the virtual disks that will be used by asm
  # these need to be of "--variant Fix"
  echo "CRSDISK=\"\${VM_SHARED_DISKS}/crs.vdi\"" >> ${VBSETUPSCRIPT}
  echo "ASMDATA=\"\${VM_SHARED_DISKS}/asmdata.vdi\"" >> ${VBSETUPSCRIPT}
  echo "ASMARCH=\"\${VM_SHARED_DISKS}/asmarch.vdi\"" >> ${VBSETUPSCRIPT}
  echo "vboxmanage closemedium disk \${CRSDISK} --delete >> /dev/null 2>&1" >> ${VBSETUPSCRIPT}
  echo "vboxmanage closemedium disk \${ASMDATA} --delete >> /dev/null 2>&1" >> ${VBSETUPSCRIPT}
  echo "vboxmanage closemedium disk \${ASMARCH} --delete >> /dev/null 2>&1" >> ${VBSETUPSCRIPT}
  echo "vboxmanage createhd --filename \${CRSDISK} --size 5000 --variant Fix" >> ${VBSETUPSCRIPT}
  echo "vboxmanage createhd --filename \${ASMDATA} --size 5000 --variant Fix" >> ${VBSETUPSCRIPT}
  echo "vboxmanage createhd --filename \${ASMARCH} --size 5000 --variant Fix" >> ${VBSETUPSCRIPT}
  # get all the vm names based on the filenames for the current participant
  VMNAMES=`ls ${BASEDIR} | grep iso | grep "p${i}" | awk -F "." '{print $1}'`
  # we need an array of vm names for generating the hostfile
  VMNAMESARR=(`ls ${BASEDIR} | grep iso | grep "p${i}" | awk -F "." '{print $1}'`)
  for x in ${VMNAMES}
  do
    # build the /etc/hosts filename for the current VM
    HOSTSFILE=${BASEDIR}/${x}_HOSTS
    # for each of the networks we need to add the public name, the public vip name,
    # the interconnect name as well as the scan name to the /etc/hosts file as
    # we do not have a dns server available for this lab setup
    #
    # the first assigned IP per participant is:
    #   the last assigned IP of the participant (which is stored above) minus
    #   the number of VMs that got created for the participant plus one
    # example:
    #   the number of VMs requested per participant is 3
    #      so, for the last VM that got created for the participant the public IP will be ${LASTASSIGNEDIP}
    #          to get the IPs for the other VMs that got created for the participant we need to substract the
    #          number of VMs, eg:
    #            ${LASTASSIGNEDIP} minus ${NUMBER_OF_VMS_PER_PARTICIPANT} + 1
    #            192.168.56.171    minus               2                + 1 = 170
    #            !! the ips assigned are 192.168.56.110, 192.168.56.111, 192.168.56.112. but the last run of the loop
    #               will increase this to 192.168.56.113, that's why we need to do a minus 1 !!
    #  This procedure is the same for all the networks ( public, vip and private )
```

```
  #  The scan is different as we need only one scan address per vm set ( that is for all VMs per participant we need one scan )
  let FIRSTVMASSIGNEDIP=${LASTASSIGNEDIP}-${NUMBER_OF_VMS_PER_PARTICIPANT}+1
  let FIRSTVMASSIGNEDPRIVIP=${LASTASSINGNEDINTERCONNECTIP}-${NUMBER_OF_VMS_PER_PARTICIPANT}+1
  let FIRSTVMASSIGNEDVIPIP=${LASTASSINGNEDVIPIP}-${NUMBER_OF_VMS_PER_PARTICIPANT}+1
  # public network, add all public ips and hostnames
  COUNTER=0
  rm -f $BASEDIR/${x}_KEYSCAN
  for ((m=${FIRSTVMASSIGNEDIP}; m <= ${LASTASSIGNEDIP}; m++ ))
  do
    echo "192.168.56.${m} ${VMNAMESARR[$COUNTER]} ${VMNAMESARR[$COUNTER]}.lab" >> ${HOSTSFILE}
    # the *_KEYSCAN files are used while kickstarting the VMs for getting the host
    # identification for the other VMs in the cluster. this results in the known_hosts
    # file for the grid and oracle users
    echo "ssh-keyscan ${VMNAMESARR[$COUNTER]} >> ~/.ssh/known_hosts" >> $BASEDIR/${x}_KEYSCAN
    let COUNTER++
  done
  # public vips, add all public vips and hostnames
  COUNTER=0
  for ((m=${FIRSTVMASSIGNEDVIPIP}; m <= ${LASTASSINGNEDVIPIP}; m++ ))
  do
    echo "192.168.56.${m} ${VMNAMESARR[$COUNTER]}-vip ${VMNAMESARR[$COUNTER]}-vip.lab" >> ${HOSTSFILE}
    let COUNTER++
  done
  # interconnects, all private interconnect ips and hostnames
  COUNTER=0
  for ((m=${FIRSTVMASSIGNEDPRIVIP}; m <= ${LASTASSINGNEDINTERCONNECTIP}; m++ ))
  do
    echo "10.1.1.${m} ${VMNAMESARR[$COUNTER]}-priv ${VMNAMESARR[$COUNTER]}-priv.lab" >> ${HOSTSFILE}
    let COUNTER++
  done
  # scan address (as we do not have dns entries we use one hosts entry)
  #  in a production setup there would be three scan ip-addresses which would resolve to the
  #  same hostname (dns round robin)
  echo "192.168.56.${IPSCANSTART} lab-scan lab-scan.lab" >> ${HOSTSFILE}
  # attach the shared devices ( "-mtype shareable" is the magic that allows us to attach the disk to more than one node )
  echo "vboxmanage storagectl ${x} --name ctl3 --add scsi" >> ${VBSETUPSCRIPT}
  echo "vboxmanage storageattach ${x} --storagectl ctl3 --type hdd --medium \"\${CRSDISK}\" --port 1 --mtype shareable" >> ${VBSETUPSCRIPT}
  echo "vboxmanage storageattach ${x} --storagectl ctl3 --type hdd --medium \"\${ASMARCH}\" --port 2 --mtype shareable" >> ${VBSETUPSCRIPT}
  echo "vboxmanage storageattach ${x} --storagectl ctl3 --type hdd --medium \"\${ASMDATA}\" --port 3 --mtype shareable" >> ${VBSETUPSCRIPT}
done
# increase the scan address for the next participant
# this is only required per participant, _not_ per vm per participant
let IPSCANSTART=IPSCANSTART+1
# create the crsconfig_params
# this is the same file for all hosts per participant
CRSCONFIGPARAMS=${BASEDIR}/${SYSTEM_PREFIX}p${i}_CRSCONFIGPARAMS
# get all the vm names based on the filenames for the current participant
VMNAMES=`ls ${BASEDIR} | grep iso | grep "p${i}" | awk -F "." '{print $1}'`
echo ${VMNAMES} > /var/tmp/tmp
sed -i 's/ /,/g' /var/tmp/tmp
```

```
VMNAMES=`cat /var/tmp/tmp`
# we need one of hosts file for putting together the strings for
# the crsconfig_params and to determine the amount of nodes
# that will participate in the cluster
AHOSTFILE=$BASEDIR/`ls $BASEDIR | grep "p${i}" | grep HOSTS | head -1`
VIPNAMES=`cat ${AHOSTFILE} | grep vip | awk -F " " '{print $2}'`
echo ${VIPNAMES} > /var/tmp/tmp
sed -i 's/ /\/255.255.255.0\/eth0,/g' /var/tmp/tmp
sed -i 's/$/\/255.255.255.0\/eth0/' /var/tmp/tmp
VIPNAMES=`cat /var/tmp/tmp`
rm -f /var/tmp/tmp
echo "SILENT=true" > ${CRSCONFIGPARAMS}
echo "ORACLE_OWNER=grid" >> ${CRSCONFIGPARAMS}
echo "ORA_DBA_GROUP=asmdba" >> ${CRSCONFIGPARAMS}
echo "ORA_ASM_GROUP=asmadmin" >> ${CRSCONFIGPARAMS}
echo "LANGUAGE_ID=AMERICAN_AMERICA.AL32UTF8" >> ${CRSCONFIGPARAMS}
echo "TZ=Europe/Zurich" >> ${CRSCONFIGPARAMS}
echo "ISROLLING=true " >> ${CRSCONFIGPARAMS}
echo "REUSEDG=false" >> ${CRSCONFIGPARAMS}
echo "ASM_AU_SIZE=1" >> ${CRSCONFIGPARAMS}
echo "USER_IGNORED_PREREQ=true" >> ${CRSCONFIGPARAMS}
echo "ORACLE_HOME=/opt/oracle/product/crs/11.2.0.4" >> ${CRSCONFIGPARAMS}
echo "ORACLE_BASE=/opt/oracle/product/base" >> ${CRSCONFIGPARAMS}
echo "OLD_CRS_HOME=" >> ${CRSCONFIGPARAMS}
echo "JREDIR=/opt/oracle/product/crs/11.2.0.4/jdk/jre/" >> ${CRSCONFIGPARAMS}
echo "JLIBDIR=/opt/oracle/product/crs/11.2.0.4/jlib" >> ${CRSCONFIGPARAMS}
echo "VNDR_CLUSTER=false" >> ${CRSCONFIGPARAMS}
echo "OCR_LOCATIONS=NO_VAL" >> ${CRSCONFIGPARAMS}
echo "CLUSTER_NAME=lab-cluster" >> ${CRSCONFIGPARAMS}
echo "HOST_NAME_LIST=${VMNAMES}" >> ${CRSCONFIGPARAMS}
echo "NODE_NAME_LIST=${VMNAMES}" >> ${CRSCONFIGPARAMS}
echo "PRIVATE_NAME_LIST=" >> ${CRSCONFIGPARAMS}
echo "VOTING_DISKS=NO_VAL" >> ${CRSCONFIGPARAMS}
echo "ASM_UPGRADE=false" >> ${CRSCONFIGPARAMS}
echo "ASM_SPFILE=" >> ${CRSCONFIGPARAMS}
echo "ASM_DISK_GROUP=CRS_DG" >> ${CRSCONFIGPARAMS}
echo "ASM_DISCOVERY_STRING=/dev/sd*1" >> ${CRSCONFIGPARAMS}
echo "ASM_DISKS=/dev/sdb1" >> ${CRSCONFIGPARAMS}
echo "ASM_REDUNDANCY=EXTERNAL " >> ${CRSCONFIGPARAMS}
echo "CRS_STORAGE_OPTION=1" >> ${CRSCONFIGPARAMS}
echo "CSS_LEASEDURATION=400" >> ${CRSCONFIGPARAMS}
echo "CRS_NODEVIPS='${VIPNAMES}'" >> ${CRSCONFIGPARAMS}
echo "NODELIST=${VMNAMES}" >> ${CRSCONFIGPARAMS}
echo "NETWORKS=\"eth0\"/192.168.56.0:public,\"eth1\"/10.1.1.0:cluster_interconnect" >> ${CRSCONFIGPARAMS}
echo "SCAN_NAME=lab-scan" >> ${CRSCONFIGPARAMS}
echo "SCAN_PORT=1521" >> ${CRSCONFIGPARAMS}
echo "GPNP_PA=" >> ${CRSCONFIGPARAMS}
echo "OCFS_CONFIG=" >> ${CRSCONFIGPARAMS}
echo "GNS_CONF=false" >> ${CRSCONFIGPARAMS}
echo "GNS_ADDR_LIST=" >> ${CRSCONFIGPARAMS}
```

```
      echo "GNS_DOMAIN_LIST=" >> ${CRSCONFIGPARAMS}
      echo "GNS_ALLOW_NET_LIST=" >> ${CRSCONFIGPARAMS}
      echo "GNS_DENY_NET_LIST=" >> ${CRSCONFIGPARAMS}
      echo "GNS_DENY_ITF_LIST=" >> ${CRSCONFIGPARAMS}
      echo "NEW_HOST_NAME_LIST=" >> ${CRSCONFIGPARAMS}
      echo "NEW_NODE_NAME_LIST=" >> ${CRSCONFIGPARAMS}
      echo "NEW_PRIVATE_NAME_LIST=" >> ${CRSCONFIGPARAMS}
      echo "NEW_NODEVIPS='${VIPNAMES}'" >> ${CRSCONFIGPARAMS}
      echo "GPNPCONFIGDIR=\$ORACLE_HOME" >> ${CRSCONFIGPARAMS}
      echo "GPNPGCONFIGDIR=\$ORACLE_HOME" >> ${CRSCONFIGPARAMS}
      echo "OCRLOC=" >> ${CRSCONFIGPARAMS}
      echo "OLRLOC=" >> ${CRSCONFIGPARAMS}
      echo "OCRID=" >> ${CRSCONFIGPARAMS}
      echo "CLUSTER_GUID=" >> ${CRSCONFIGPARAMS}
      echo "CLSCFG_MISSCOUNT=" >> ${CRSCONFIGPARAMS}
      echo "CRFHOME=\"/opt/oracle/product/crs/11.2.0.4\"" >> ${CRSCONFIGPARAMS}
      # copy the crsconfig_params for each host, delete the template and create the ssh keys
      VMNAMES=`ls ${BASEDIR} | grep iso | grep "p${i}" | awk -F "." '{print $1}'`
      for v in ${VMNAMES}
      do
        cp ${CRSCONFIGPARAMS} $BASEDIR/${v}_CRSCONFIGPARAMS
        # create the public and private ssh keys for the grid and oracle user
        # this looks a little bit nasty inside this script. the reason for doing
        # it here is, that this script should be as much self contained as possible
        echo "ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAp6NQY6EwNaLLtU2swaEJbtU7PhYqiF3fEeIKl84+5rdJGUmK3HS06oKzlqbOvmk0q3pdno2ITQL7wG5lQD9kmMKGoWznVhNRUG/HUSaZCcft239KDeBgjX5AaNP
dYxa0tgAnhvhm7w5nF2QHsr7qAcCLxckCNYgpsA7wAQ8CgZzpDWtxKJh2PK5L4HqLIEj7nY2AmAzW7dMX2cxtgPUTa7tLOJeWxdWOPcC6Qnw4IciFr5IA3DiJxVg0wo99Mkxqf8oyJrTXTmbg/E6J4q
2XMuSycdao1XKliUWMUsp1av/pyPv6JDa0+fabCcv8rbfdwIwx4xG1+XUJYj7NjzL3YQ== grid@${v}" > $BASEDIR/${v}_GRID_PUBLIC_SSH
        cp $BASEDIR/${v}_GRID_PUBLIC_SSH $BASEDIR/${v}_ORACLE_PUBLIC_SSH
        sed -i 's/grid/oracle/g' $BASEDIR/${v}_ORACLE_PUBLIC_SSH
        echo "-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAp6NQY6EwNaLLtU2swaEJbtU7PhYqiF3fEeIKl84+5rdJGUmK
3HS06oKzlqbOvmk0q3pdno2ITQL7wG5lQD9kmMKGoWznVhNRUG/HUSaZCcft239K
DeBgjX5AaNPdYxa0tgAnhvhm7w5nF2QHsr7qAcCLxckCNYgpsA7wAQ8CgZzpDWtx
KJh2PK5L4HqLIEj7nY2AmAzW7dMX2cxtgPUTa7tLOJeWxdWOPcC6Qnw4IciFr5IA
3DiJxVg0wo99Mkxqf8oyJrTXTmbg/E6J4q2XMuSycdao1XKliUWMUsp1av/pyPv6
JDa0+fabCcv8rbfdwIwx4xG1+XUJYj7NjzL3YQIBIwKCAQATKJt5Gbxd5rgx+j+h
GbfvaNOSESluq6R3EoTeJjMTDaH0QutMZRv9i0e5ciZBpZ+ekaRMonX6LDoHXRLi
xWqVHYu6rVxEWfqqG2c8eXCTZ05iOm7dA7NDXuLKJtd5CejEV8oAy+6BuIDPeSVz
g4h1OpOhk1C1qSlV8xQdYM0WIGBFT11n6WpZaM7LVGkNkbg9eja+2i0YO5FhhrT+
uCh71onQagFJ/WsColn6PCQj7belDQkYU0C0x/u+z6TMhQo8rPqDBOxaPqI3lbGQ
i8wphcLmDuIdPyMYzUahKLkkNV3IbvSgXIku7rbhoZ9ta3r+TfgHYNw8xEijoSnh
11j7AoGBANgEhM0PQLklBDql+sojlnJ/yqbvGmhSNPlgATYqT8Cd44EExXl9xP5f
R/si6JoUkQ4wN0x02ae6+1PrrW8xyam8hBHS/3T2EU00o01yC6e5wqqkFMeZ7zBs
itYMR8B0Fa4DkQ6pvaOdwdk73x2hMs6Q+FRUgYX4yiPTFGyl0/ErAoGBAMaqcDLM
HVqJVP/HXSnFt30D1Xqi/WQHlOZkcIeh2fCSQpRUMpIApyeHbCvSBWHS13CbKKTS
hmvchwYRSSwC9qcwXJBZSk5CsYhhTWq/2rUNz55wv6a7mwIDtoxEXmD6F0croA7/
T2EBjAGXpqwPGJr+aaNOAKjs9Wn3aDJxQrujAoGADFgHlrBqGTVQsuTpwmhyp3UE
RA2pvNF/XrUHYi5Nszw44s0SmTpjBzilBwlPHr9Yvvtw38wpsdArnmU9HEv85SC3
FvYdOeIs39cfRkEH+vSsCcA7s6JlcHs7IiyWYsTOCfGTQqqdH0rX4IcUEFJarLfT
```

```
rQwkqJHfqkaTc+w3/ycCgYEAu1A95sBzcp692zhQhniXD3iktXUazAcl/cxqGXtY
dRwwJXP1KpLmvt7FE1+erLDZwfFg1esRBpy50oVTnoZzh7FBVOZ5Qnlljzcrv Gu/
lMPhAxnZRW8Ofj4D41ZnnUOSSmr1/4MBszv5CNDQWRWMOl2WzSxJw9galxx4L49c
LUECgYEAtY56zS25yEHPyI3pg0DezjelMCKofC2Dt75S8dszkCkw2fsfZZYYF+Mg
BmruRzayTgD2O9HAiE45l2YeJ+hta7eWuoag7AXC56/6dvUQH1sznDb8S8OEuJ66
uTIbzXJg+rhmPciLMXMQlYmTAoQWIQ+iniI8sKHpEo3/jPySkCk=
-----END RSA PRIVATE KEY-----" > $BASEDIR/${v}_GRID_PRIVATE_SSH
        cp $BASEDIR/${v}_GRID_PRIVATE_SSH $BASEDIR/${v}_ORACLE_PRIVATE_SSH
        cat `ls ${BASEDIR}/${SYSTEM_PREFIX}p${i}*GRID_PUBLIC_SSH` > $BASEDIR/${v}_GRID_AUTHORIZED_KEYS
        cat `ls ${BASEDIR}/${SYSTEM_PREFIX}p${i}*ORACLE_PUBLIC_SSH` > $BASEDIR/${v}_ORACLE_AUTHORIZED_KEYS
      done
      rm -f ${CRSCONFIGPARAMS}
    fi
  done
  # sync cobbler
  cobbler sync 2>1 > /dev/null
}

####### main calls
_do_checks
_setup_lab
cobbler system list
```