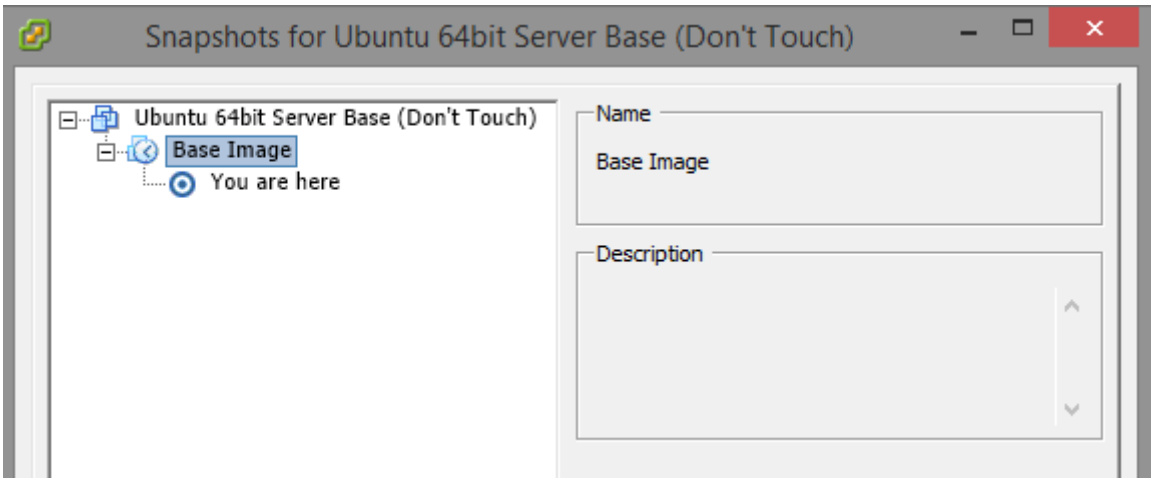
Creating a linked clone in ESXi (the easy way)

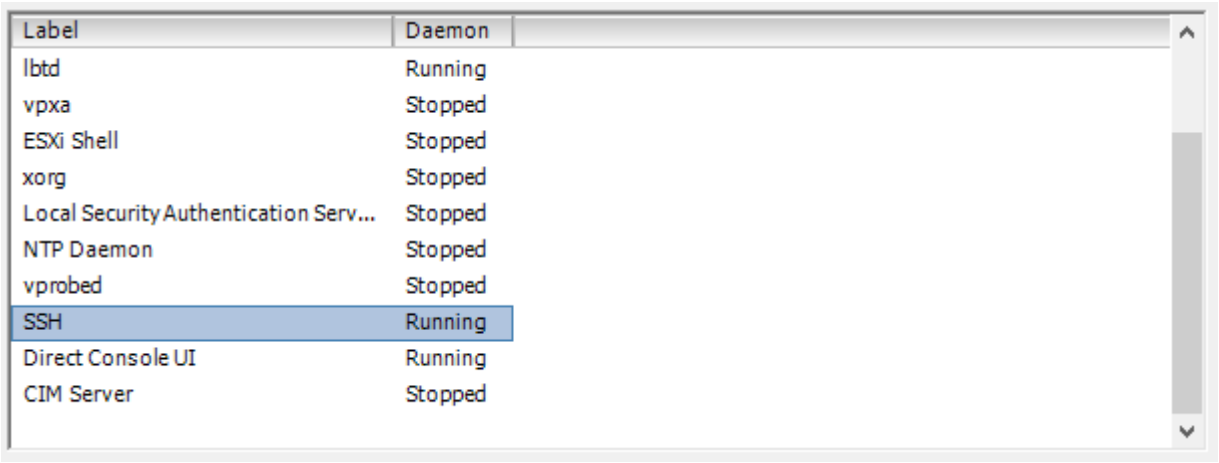
http://fewstreet.com/2014/05/30/creating-a-linked-clone-esxi.html
30 May 2014

Linked clones are a neat feature in ESXi for installations with many virtual machines that need to be deployed off the same template, because once you've created a base image you can deploy many copies of it without consuming a few GB for each install. Linked clones only consume space for the diff between them and the base image. Currently, the process of creating a linked clone is somewhat tedious if you only have vSphere, and requires downloading vmx and vdmk files to your computer and manually editing and duplicating them. I wrote a bash script to automate the process on your server which saves a lot of time and messing about.

To start with, you need a master VM created with the full install you want on each clone. In my case I installed Ubuntu LTS on a VM called "Ubuntu Base Server (Don't Touch)", installed various utility programs I would need on every clone, and copied my public SSH key to it for secure access. Any configuration that you do now will save you considerable time down the road because it won't need to be done individually on each clone. Once you are satisfied, power down the VM and take a single snapshot of it called "Base Image" or something similar. After this point you should not modify the master VM at all.



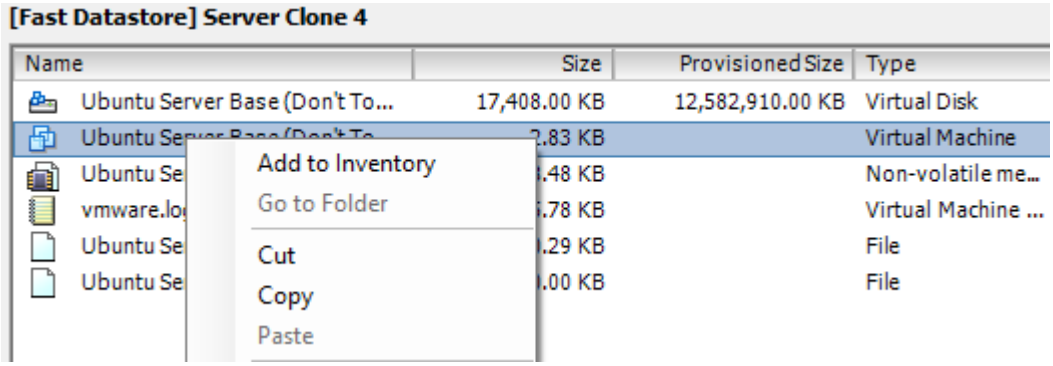
At this point, turn on the ESXi SSH service and connect to it.



Browse to your datastore containing the master VM, copy/paste my script from its GitHub repo (ESXi wget doesn't support direct wget from https), and run it. The first argument should be the folder name of your base image and the second argument should be the name of the folder you want the clone output to.

```
ls /vmfs/volumes/Datastore/
vi clone.sh # enter insert mode (i), paste code in, press escape and save file with :wq
./clone.sh Ubuntu\ Server\ Base\ \(Don\'t\ Touch\)/ Server\ Clone\ 1
```

The script will copy the virtual machine and the base image snapshot to the new directory and modify the files as necessary. All that is left is for you to add them to your inventory from the datastore browser in ESXi, naming them something like Clone 1, Clone 2, etc.



Any time you need more linked clones in the future you can just run the script again with more unique names and add the resulting copies. Hopefully this script will be helpful to you if you need to quickly make a large number of linked clones in ESXi!

Update 10/30/14:

oliverbock created a nice pull request to add some new features to clone.sh. The changes are pretty self explanatory, but the big new feature is that clones are automatically registered when they are created. He also added a deleteclone.sh script that you can use to safely remove clones (using ESXi to delete a clone will also delete the base disk.)

clone.ssh

```
readonly NUMARGS=$#
readonly INFOLDER=$1
readonly OUTFOLDER=$2

usage() {
  echo "USAGE: ./clone.sh base_image_folder out_folder"
}
makeandcopy() {
  mkdir "$OUTFOLDER"
  cp "$INFOLDER"/*-"$VMFILE"* "$OUTFOLDER"/
  cp "$INFOLDER"/*.vmx "$OUTFOLDER"/
}
main() {
  if [  $NUMARGS -le 1 ]
  then
    usage
    exit 1
  fi

  VMFILE=`grep scsi0\:0\.fileName "$INFOLDER"/*.vmx | grep -o "[0-9]\{6,6\}"`

  makeandcopy

  #reference snapshot
  SNAPSHOT=`grep -o "[^\"]*.vmsn" "$INFOLDER"/*.vmx | tail -1`
  sed -i -e '/checkpoint.vmState =/s/= .*/= "..\/'$INFOLDER'\/'$SNAPSHOT'"/' $OUTFOLDER/*.vmx

  local fullbasepath=$(readlink -f "$INFOLDER")/
  cd "$OUTFOLDER"/
  sed -i '/sched.swap.derivedName/d' ./*.vmx #delete swap file line, will be auto recreated
  sed -i -e '/displayName =/ s/= .*/= "'$OUTFOLDER'"/' ./*.vmx #Change display name config value
  local escapedpath=$(echo "$fullbasepath" | sed -e 's/[\/&]/\\&/g')
  sed -i -e '/parentFileNameHint=/ s/="/="'"$escapedpath"'/' ./*-"$VMFILE".vmdk #change parent disk
path

  # Forces generation of new MAC + DHCP, I think.
  sed -i '/ethernet0.generatedAddress/d' ./*.vmx
  sed -i '/ethernet0.addressType/d' ./*.vmx
```

```
  # Forces creation of a fresh UUID for the VM.  Obviates the need for the line
  # commented out below:
  #echo 'answer.msg.uuid.altered="I copied it" ' >>./*.vmx
  sed -i '/uuid.location/d' ./*.vmx
  sed -i '/uuid.bios/d' ./*.vmx

  # Things that ghetto-esxi-linked-clones.sh did that we might want.  I can only guess at their
use/value.
  #sed -i '/scsi0:0.fileName/d' ${STORAGE_PATH}/$FINAL_VM_NAME/$FINAL_VM_NAME.vmx
  #echo "scsi0:0.fileName = \"${STORAGE_PATH}/${GOLDEN_VM_NAME}/${VMDK_PATH}\"" >>
${STORAGE_PATH}/$FINAL_VM_NAME/$FINAL_VM_NAME.vmx
  #sed -i 's/nvram = "'${GOLDEN_VM_NAME}.nvram'"/nvram = "'${FINAL_VM_NAME}.nvram'"/'
${STORAGE_PATH}/$FINAL_VM_NAME/$FINAL_VM_NAME.vmx
  #sed -i 's/extendedConfigFile = "'${GOLDEN_VM_NAME}.vmxf'"/extendedConfigFile =
"'${FINAL_VM_NAME}.vmxf'"/' ${STORAGE_PATH}/$FINAL_VM_NAME/$FINAL_VM_NAME.vmx

  # delete machine id
  sed -i '/machine.id/d' *.vmx

  # add machine id
  sed -i -e "\$amachine.id=$OUTFOLDER" *.vmx

  # Register the machine so that it appears in vSphere.
  FULL_PATH=`pwd`/*.vmx
  VMID=`vim-cmd solo/registervm $FULL_PATH`

  # Power on the machine.
  vim-cmd vmsvc/power.on $VMID
}

Main
```

Deleteclone.ssh

```
# Deletes a cloned VM created by clone.sh.
if [ $# -le 0 ]
then
  echo "USAGE: ./deleteclone.sh vm_name"
  exit 1
fi
readonly VMNAME=$1

VMID=$(vim-cmd vmsvc/getallvms | awk "/^[0-9]+ +$VMNAME /{print \$1}")
vim-cmd vmsvc/power.off $VMID
vim-cmd vmsvc/unregister $VMID
rm -rf $VMNAME
```