ghettoVCB を使う方法

https://github.com/lamw/ghettoVCB
からスクリプトをダウンロードして設置する。

独自にスクリプトを作成する方法

ネットで拾ったスクリプト

```
 #!/bin/sh


 #Edit these values to match you environment
 ####################################
 #The datastore to backup to
 backupDataStore=<backupDataStore>

 #The directory on the above datastore to backup to(the default is mm-dd-yyyy)
 backupDirectory=$(date +%m-%d-%Y)

 #The list of virtual machine names(separated by a space) to backup
 vmsToBackup="VM1 VM2 VM3"

 #The amount of time to wait for the snapshot to complete, some systems are slower than others and
snapshot operations may take longer to complete
 waitTime=40s
 ####################################

 startTime=$(date)
 echo Backup start time: $startTime

 echo Creating backup directory /vmfs/volumes/$backupDataStore/$backupDirectory
 mkdir -p /vmfs/volumes/$backupDataStore/$backupDirectory

 echo Backing up ESXi host configuration...
 vim-cmd hostsvc/firmware/backup_config
 cp /scratch/downloads/*.tgz /vmfs/volumes/$backupDataStore/$backupDirectory/

 for vm in $vmsToBackup;do
   vmName=$vm
    vmIdAndConfigPath=$( vim-cmd vmsvc/getallvms | awk '{ if ($2 == vmname) print $1 ";" $3 $4}'
vmname=$vm)
   vmId=${vmIdAndConfigPath%;*}

   if [ "$vmId" != "" ]; then

     echo Backing up virtual machine: $vmName


     echo Backing up the virtual machines configuration...
     vmConfigurationFilePath=$(echo ${vmIdAndConfigPath#*;} | sed -e 's/\[\(.*\)\]\(.*\)/\1;\2/')
     vmConfigurationSourceDataStore=${vmConfigurationFilePath%;*}
     vmConfigurationFile=${vmConfigurationFilePath#*;}
     echo Making directory /vmfs/volumes/$backupDataStore/$backupDirectory/${vmConfigurationFile%/*}
     mkdir -p /vmfs/volumes/$backupDataStore/$backupDirectory/${vmConfigurationFile%/*}
            echo    Copying    /vmfs/volumes/$vmConfigurationSourceDataStore/$vmConfigurationFile    to
/vmfs/volumes/$backupDataStore/$backupDirectory/$vmConfigurationFile
                         cp         /vmfs/volumes/$vmConfigurationSourceDataStore/$vmConfigurationFile
/vmfs/volumes/$backupDataStore/$backupDirectory/$vmConfigurationFile

     echo Taking the snapshot...
     vim-cmd vmsvc/snapshot.create $vmId "Backup"

     echo Waiting $waitTime for the snapshot to complete...
     sleep $waitTime

     echo Getting diskFile list...
      vmDiskFilePaths=$(vim-cmd vmsvc/get.filelayout $vmId | grep -i snapshotFile -A2000 | sed -n -e
's/\"\[\(.*\)\]\s\(.*\.vmdk\)\"\,/\1;\2/pg')
     echo Found $(echo $vmDiskFilePaths | wc -l) disk file\(s\)...
     for vmDiskFilePath in $vmDiskFilePaths; do
       vmDiskFileSourceDataStore=${vmDiskFilePath%;*}
       vmDiskFile=${vmDiskFilePath#*;}
       if [ -e /vmfs/volumes/$vmDiskFileSourceDataStore/$vmDiskFile ]; then
         if [ ! -d /vmfs/volumes/$backupDataStore/$backupDirectory/${vmDiskFile%/*} ]; then
           mkdir -p /vmfs/volumes/$backupDataStore/$backupDirectory/${vmDiskFile%/*}
         fi

                         echo    Cloning    /vmfs/volumes/$vmDiskFileSourceDataStore/$vmDiskFile    to
/vmfs/volumes/$backupDataStore/$backupDirectory/$vmDiskFile
                vmkfstools  -d  2gbsparse  -i  /vmfs/volumes/$vmDiskFileSourceDataStore/$vmDiskFile
/vmfs/volumes/$backupDataStore/$backupDirectory/$vmDiskFile
       fi
     done

     echo Removing the snapshot...
     vim-cmd vmsvc/snapshot.removeall $vmId

   else
     echo ERROR: Could not get an id for $vmName
   fi
 done

 endTime=$(date)
 echo Backup end time: $endTime
 #echo Elapsed time: $(($startTime - $endTime))
```

カスタマイズ

vm 側からサーバのバックアップスクリプトを実行し、vm でマウントした samba にコピーする

■サーバー側

```
  #!/bin/sh


  #Edit these values to match you environment
  #####################################
  #The datastore to backup to
  backupDataStore=datastore1

  #The directory on the above datastore to backup to(the default is mm-dd-yyyy)
  #backupDirectory=back_temp/$(date +%m-%d-%Y)
  backupDirectory=back_temp

  distBaseDir=/vmfs/volumes/$backupDataStore/$backupDirectory

  #The name of a virtual machine. allow to contain space in name.
  # vmsToBackup="$*"
  vmName="$*"

  LINE_SEP='
  '
  #####################################

  check_running_task (){
    for i in `vim-cmd vimsvc/task_list | grep vim.Task:haTask-$1 | grep $2 | sed -e 's/.*vim.Task://'
-e "s/[', ]//g"`; do
      if [ "`vim-cmd vimsvc/task_info $i | grep running`" != "" ] ; then
        return 0
      fi
    done
    return 1
  }

  # getVmId (){
  # vim-cmd vmsvc/getallvms | sed 's/[[:blank:]]\{3,\}/   /g' | awk -F'   ' '{ if ($2 == vmname)
print $1}' vmname="$*"
  # }
  #
  # getVmx (){
  # vim-cmd vmsvc/getallvms | sed 's/[[:blank:]]\{3,\}/   /g' | awk -F'   ' '{ if ($1 == vmid) print
$3}' vmid=$1
  # }

  #
  getVMDKs(){
    OLD_IFS=$IFS
    IFS=$LINE_SEP

    for i in $(grep '.vmdk"' "$*" | awk -F "\"" '{print $2}'); do
      firstchar=`echo "$i" | cut -c 1-1`
      if [ "$firstchar" = "/" ]; then
        echo $i
      else
        echo ${*%/*}/$i
      fi
      #echo $i
    done

    IFS=$OLD_IFS
  }


  startTime=$(date)
  echo Backup start time: $startTime

  echo Creating backup directory "$distBaseDir"
  mkdir -p "$distBaseDir"

  echo Backing up ESXi host configuration...
  vim-cmd hostsvc/firmware/backup_config
  cp /scratch/downloads/*.tgz "$distBaseDir"

  # vmid=`getVmId $vmName`
  # echo $vmid
  # vmx=`getVmx $vmid`
  # echo $vmx

  vmIdAndConfigPath=$( vim-cmd vmsvc/getallvms | sed 's/[[:blank:]]\{3,\}/   /g' | awk -F'   ' '{ if
($2 == vmname) print $1 ";" $3}' vmname="$vmName")
  vmId=${vmIdAndConfigPath%;*}

  if [ "$vmId" != "" ]; then
    echo Backing up virtual machine: $vmName

    echo Backing up the virtual machines configuration...
    vmConfigurationFileInfo=$(echo ${vmIdAndConfigPath#*;} | sed -e 's/\[\(.*\)\] \(.*\)/\1;\2/')
    vmConfigurationSourceDataStore=${vmConfigurationFileInfo%;*}
    vmConfigurationFile=${vmConfigurationFileInfo#*;}
    vmBaseDir=${vmConfigurationFile%/*}
    vmConfigurationFile=${vmConfigurationFile##*/}
    srcBaseDir="/vmfs/volumes/$vmConfigurationSourceDataStore/${vmBaseDir}"

    echo Backup source directory : "$srcBaseDir"
    echo vmx : "$vmConfigurationFile"

    echo Making directory "$distBaseDir/$vmBaseDir"
    mkdir -p "$distBaseDir/$vmBaseDir"
                 echo          Copying          "$srcBaseDir/$vmConfigurationFile"              to
"$distBaseDir/$vmBaseDir/$vmConfigurationFile"
    cp "$srcBaseDir/$vmConfigurationFile" "$distBaseDir/$vmBaseDir/$vmConfigurationFile"

    vmDiskFilePaths=`getVMDKs "$srcBaseDir/$vmConfigurationFile"`
    # echo $vmdks


    echo Taking the snapshot...
    vim-cmd vmsvc/snapshot.create $vmId "Backup"

    # echo Waiting $waitTime for the snapshot to complete...
    #sleep $waitTime
    echo Waiting for the snapshot to complete...
    while check_running_task $vmId createSnapshot; do sleep 1; done

    # echo Getting diskFile list...
    # vmDiskFilePaths=$(vim-cmd vmsvc/get.filelayout $vmId | grep -i snapshotFile -A2000 | sed -n -e
's/\"\[\(.*\)\]\s\(.*\.vmdk\)\"\,\?/\1;\2/pg')
    echo $vmDiskFilePaths
    echo Found $(echo "$vmDiskFilePaths" | wc -l) disk file\(s\)...
    OLD_IFS=$IFS
    IFS=$LINE_SEP
    for vmDiskFilePath in $vmDiskFilePaths; do
      # vmDiskFileSourceDataStore=${vmDiskFilePath%;*}
      vmDiskFile="${vmDiskFilePath##*/}"
      echo $vmDiskFilePath

      if [ -e "$vmDiskFilePath" ]; then
        echo Cloning "$vmDiskFilePath" to "$distBaseDir/$vmBaseDir/$vmDiskFile"
        vmkfstools -d monosparse -i "$vmDiskFilePath" "$distBaseDir/$vmBaseDir/$vmDiskFile"
        # vmkfstools -d thin -i "$vmDiskFilePath" "$distBaseDir/$vmBaseDir/$vmDiskFile"
      fi
    done
    IFS=$OLD_IFS

    echo Removing the snapshot...
    # vim-cmd vmsvc/snapshot.removeall $vmId
    SNAPSHOT_ID=$(vim-cmd vmsvc/snapshot.get $vmId | grep -E '(Snapshot Name|Snapshot Id)' | grep -A1
"Backup"    |    grep    "Snapshot    Id"    |    awk    -F    ":"    '{print    $2 }'    |    sed    -e
's/^[[:blank:]]*//;s/[[:blank:]]*$//')
    for id in $SNAPSHOT_ID; do
      vim-cmd vmsvc/snapshot.remove $vmId ${id}
    done

  else
    echo ERROR: Could not get an id for $vmName
  fi
  # done


  #cd /vmfs/volumes/$backupDataStore/$backupDirectory/
  #dirnames=`ls -F /vmfs/volumes/$backupDataStore/$backupDirectory/ | grep /`
  #
  #for d  in ${dirnames};do
  #  dirname=`echo $d | cut -d / -f 1`
  #  echo compress ${dirname}
  #
  #  tar -czvf ${dirname}.tar.gz ${dirname}
  #  rm -r ${dirname}
  #done


  endTime=$(date)
  echo Backup end time: $endTime
  #echo Elapsed time: $(($startTime - $endTime))
```

■呼び出し側

```
#!/bin/sh

if [ "$*" != "" ]; then
  host=$1
  vm=$2
  backupDirectory=$(date +%Y-%m-%d)
  backupFrom=/vmfs/volumes/datastore1/back_temp
  backupTo=/mnt/smb/vmbackup/$backupDirectory/$vm

  echo ------------------------------------
  echo host:$host
  echo vm:$vm
  echo $host:$backupFrom To $backupTo
  echo ------------------------------------

  mkdir -p $backupTo
  ssh -i ~/.ssh/id_rsa  $host rm -r $backupFrom/*
  ssh -i ~/.ssh/id_rsa  $host /bin/vmbackup_server.sh $vm
  scp -r root@$host:$backupFrom/* $backupTo
  ssh -i ~/.ssh/id_rsa  $host rm -r $backupFrom/*
fi
```

■サーバに自作シェルを置く際の注意

基本的に ESXi 再起動時にファイルが削除されてしまう。

```
# cd /
# tar cvzf hoge.tgz bin/hoge.sh
# mv hoge.tgz /bootbank/
# vi /bootbank/boot.cfg
```

起動時に解凍するファイルを追加する

```
modules=binmod.tgz  ---  environ.tgz  ---  cim.tgz  ---  oem.tgz  ---  license.tgz  ---  state.tgz  ---
```

```
    hoge.tgz
```

リストア

１．バックアップしたハードディスクイメージを統合

```
    vmkfstools -i インポート元.vmdk インポート先.vmdk
```

２．データストアブラウザで仮想マシンを「インベントリに追加」

その他

■ tar.gz を直接転送する

http://blog.xe.bz/archives/51126645.html

```
    tar cf - test\ UBuntu | gzip - | ssh hoge@hoge.hoge "cat > /path/to/backupdir/hogehoge.tar.gz"
```

みたいな感じで、tar.gz をローカルに保存することなく直接 SSH へ送れる。

■バックアップスクリプト　サンプル

http://www.witkitty.com/wordpress/2009/07/05/esxi-backup-part2/

スナップショットを取る際に、プロセスを監視して待つところが良い。