Table of Contents

Creating an ssh key

#TODO

Creating new Instance

Two(+) possible ways

1. **Recommended** Without using a volume (back-up will be done using snapshots):
   - Simply click on start a new instance on the cern openstack web interface (https://openstack.cern.ch/dashboard/project/instances/)
2. Using a volume (seems to be problematic to reactivate in case of issues --- I would have to study the subject deeper, if you have time to do so, maybe you can found out how to do using this way (please contact me if you do so :D) ---, however it should allow you to recovery of VM state on death of hypervisor (http://clouddocs.web.cern.ch/clouddocs/tutorial/boot_from_volume.html)):
   - Create a new volume from the wanted SLC system base.
   - On the volume tab, click on the volume and choose start a new instance.
   - Choose highest flavor available.

Very Important

- Don't forget to add the ssh key-pair created in the last step to the instance in **both ways**, otherwise it won't be possible to log without entering password on this machine (it seems it is possible to change host key-pair information using JSON (https://answers.launchpad.net/nova/+question/223104), but I haven't tried).

Wait until it's ready for log-in, and then continue installation.

Expanding VM disk size to flavour, or volume, size

The procedure I've followed was:

```
zsh
sudo growpart /dev/vda 2
```

```
CHANGED: partition=2 start=821248 old: size=20150272 end=20971520 new: size=104035952,end=104857200
```

Reboot instance | **Reboot Instance** | **Reboot Instance** | **Reboot Instance** | **Reboot Instance** | **Reboot Instance**

using cern openstack web interface (https://openstack.cern.ch/dashboard/project/instances/), then:

```
zsh
sudo pvresize /dev/vda2
```

```
  Physical volume "/dev/vda2" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
```

```
zsh
sudo lvextend -l +100%FREE /dev/mapper/VolGroup00-LogVol00
```

```
  Size of logical volume VolGroup00/LogVol00 changed from 8.09 GiB (259 extents) to 48.09 GiB (1539 extents).
  Logical volume LogVol00 successfully resized
```

```
zsh
sudo resize2fs /dev/mapper/VolGroup00-LogVol00
```

```
resize2fs 1.41.12 (17-May-2010)
Filesystem at /dev/mapper/VolGroup00-LogVol00 is mounted on /; on-line resizing required
old desc_blocks = 1, new_desc_blocks = 4
Performing an on-line resize of /dev/mapper/VolGroup00-LogVol00 to 12607488 (4k) blocks.
```

In [5]:
```
%%bash
df -H
```

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                       83G   34G   45G  43% /
tmpfs                 4.2G     0  4.2G   0% /dev/shm
/dev/vda1             398M  106M  272M  28% /boot
AFS                   9.3G     0  9.3G   0% /afs
cvmfs2                 25G   21G  3.9G  85% /cvmfs/atlas.cern.ch
```

Installing LXPLUS like commands

```
zsh
sudo yum groupinstall "Development Tools" -y
```

```
Installed:
  autoconf.noarch 0:2.63-5.1.el6      automake.noarch 0:1.11.1-4.el6
  bison.x86_64 0:2.4.1-5.el6          byacc.x86_64 0:1.9.20070509-7.el6       cscope.x86_64 0:15.6-6.el6
  ctags.x86_64 0:5.8-2.el6            diffstat.x86_64 0:1.51-2.el6
  doxygen.x86_64 1:1.6.1-6.el6 flex.x86_64 0:2.5.35-9.el6                     gcc.x86_64 0:4.4.7-16.el6
  gcc-c++.x86_64 0:4.4.7-16.el6       gcc-gfortran.x86_64 0:4.4.7-16.el6
  git.x86_64 0:1.7.1-3.el6_4.1        indent.x86_64 0:2.2.10-7.el6            intltool.noarch 0:0.41.0-1.1.el6
  libtool.x86_64 0:2.2.6-15.5.el6     patchutils.x86_64 0:0.3.1-3.1.el6
  rcs.x86_64 0:5.7-37.el6             redhat-rpm-config.noarch 0:9.0.3-44.el6 rpm-build.x86_64 0:4.8.0-47.el6
  subversion.x86_64 0:1.6.11-15.el6_7 swig.x86_64 0:1.3.40-6.el6
  systemtap.x86_64 0:2.7-2.el6

Dependency Installed:
  apr.x86_64 0:1.3.9-5.el6_2                apr-util.x86_64 0:1.3.9-3.el6_0.1
  cloog-ppl.x86_64 0:0.15.7-1.2.el6         cpp.x86_64 0:4.4.7-16.el6
  gdb.x86_64 0:7.2-83.el6                   gettext-devel.x86_64 0:0.17-18.el6
  gettext-libs.x86_64 0:0.17-18.el6         kernel-devel.x86_64 0:2.6.32-573.7.1.el6
  libart_lgpl.x86_64 0:2.3.20-5.1.el6       libgcj.x86_64 0:4.4.7-16.el6
  libgfortran.x86_64 0:4.4.7-16.el6         libstdc++-devel.x86_64 0:4.4.7-16.el6
  mpfr.x86_64 0:2.4.1-6.el6                 neon.x86_64 0:0.29.3-3.el6_4
  pakchois.x86_64 0:0.4-3.2.el6             perl-Error.noarch 1:0.17015-4.el6
  perl-Git.noarch 0:1.7.1-3.el6_4.1         ppl.x86_64 0:0.10.2-11.el6
  systemtap-client.x86_64 0:2.7-2.el6       systemtap-devel.x86_64 0:2.7-2.el6
```

Adding CERN users

Execute following commands:

```
zsh
sudo yum install -y cern-config-users
# Add cern users:
/usr/sbin/addusercern <account>
sudo /usr/sbin/cern-config-users --setup-all
```

Adding CERN printers

First, search the printer names by doing `/usr/sbin/lpadmincern --building XXXX --list`, and then add those you want with the command:

```
zsh
/usr/sbin/lpadmincern printername --add
```

Installing ATLAS CVMFS

This is needed to have access to most of the ATLAS software, as some Athena releases, RootCore, python versions used by ATLAS and so on... The commands displayed are taken from here (https://twiki.cern.ch/twiki/bin/view/AtlasComputing/Cvmfs21):

```
zsh
# Suggestion: import the GPG key via HTTPS to ensure the integrity of the RPM file before installing
sudo rpm --import https://cvmrepo.web.cern.ch/cvmrepo/yum/RPM-GPG-KEY-CernVM

#for SL6 (SL5 users, please migrate to SL6):
sudo rpm -Uvh http://cvmrepo.web.cern.ch/cvmrepo/yum/cvmfs/EL/6/`uname -i`/cvmfs-release-2-4.el6.noarch.rpm
```

```
Retrieving http://cvmrepo.web.cern.ch/cvmrepo/yum/cvmfs/EL/6/x86_64/cvmfs-release-2-4.el6.noarch.rpm
Preparing...                ########################################### [100%]
   1:cvmfs-release          ########################################### [100%]
```

Now we install (for updating just change `yum install` to `yum update`) cvmfs:

In [ ]:
```
# for Tier3s and others:
sudo yum install cvmfs cvmfs-config-default cvmfs-auto-setup -y
```

Then we head into configuration. Add the following file with (I've copied this file configuration from lxplus, which seems to work better then the configuration available on the link I've followed...):

```
zsh
cat << EOF > default.local
CVMFS_REPOSITORIES=atlas.cern.ch,atlas-condb.cern.ch,atlas-nightlies.cern.ch,sft.cern.ch
CVMFS_QUOTA_LIMIT='23664'
CVMFS_HTTP_PROXY='http://ca-proxy.cern.ch:3128;http://ca-proxy1.cern.ch:3128|http://ca-proxy2.cern.ch:3128|http://ca-p
roxy3.cern.ch:3128|http://ca-proxy4.cern.ch:3128|http://ca-proxy5.cern.ch:3128'
CVMFS_CACHE_BASE='/var/lib/cvmfs'
CVMFS_FORCE_SIGNING='yes'
EOF
sudo mkdir -p /etc/cvmfs/
sudo mv default.local /etc/cvmfs
```

Start services:

```
zsh
sudo service autofs start
```

```
Starting automount: automount: program is already running.
                                                           [  OK  ]
```

Make sure to check each of the following commands outputs. I got some warnings on chksetup (which I ignored completely):

```
Warning: failed to access http://cernvmfs.gridpp.rl.ac.uk/cvmfs/atlas.cern.ch/.cvmfspublished through proxy DIRECT
Warning: failed to use Geo-API with cernvmfs.gridpp.rl.ac.uk
Warning: failed to use Geo-API with cvmfs.racf.bnl.gov
Warning: failed to access http://cernvmfs.gridpp.rl.ac.uk/cvmfs/atlas-condb.cern.ch/.cvmfspublished through proxy DIRE
CT
Warning: failed to use Geo-API with cernvmfs.gridpp.rl.ac.uk
Warning: failed to use Geo-API with cvmfs.racf.bnl.gov
Warning: failed to access http://cernvmfs.gridpp.rl.ac.uk/cvmfs/sft.cern.ch/.cvmfspublished through proxy DIRECT
Warning: failed to use Geo-API with cernvmfs.gridpp.rl.ac.uk
Warning: failed to use Geo-API with cvmfs.racf.bnl.gov
```

```
zsh
# Each of the following commands should have their outputs checked:
sudo cvmfs_config chksetup # Printed some warnings for me, but seems to be ok.
sudo cvmfs_config stat
sudo cvmfs_config showconfig
sudo cvmfs_config probe
```

Finally, run the following command and check if no more ATLAS software is missing (this is done considering that the node would be a Tier3):

```
zsh
/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/utilities/installCheck.sh # Can be run without root permissions
```

```
yum install compat-db43.i686 compat-db43.x86_64 compat-expat1.i686 compat-expat1.x86_64 compat-libf2c-34.i686 compat-l
ibf2c-34.x86_64 compat-libtermcap.i686 compat-libtermcap.x86_64 compat-openldap.x86_64 compat-readline5.i686 compat-re
adline5.x86_64 freetype-devel.i686 freetype-devel.x86_64 freetype.i686 glibc-devel.i686 glibc.i686 libaio.i686 libpng-
devel.i686 libpng-devel.x86_64 libstdc++.i686 libuuid-devel.i686 libuuid-devel.x86_64 libXext-devel.i686 libXext-devel
.x86_64 libXext.i686 libXft.i686 libxml2-devel.i686 libxml2-devel.x86_64 libxml2.i686 libXpm.i686 libXpm.x86_64 mesa-l
ibGL-devel.i686 mesa-libGL-devel.x86_64 mesa-libGL.i686 mesa-libGLU-devel.i686 mesa-libGLU-devel.x86_64 mesa-libGLU.i6
86 ncurses-devel.i686 ncurses-devel.x86_64 openldap.i686 openssl098e.i686 openssl098e.x86_64 pam.i686 zlib-devel.i686
zlib-devel.x86_64 zlib.i686
. Checking for missing yum groups ...
. Checking that SELinuz is disabled ...
.. SELinux is not disabled.
..  You can disable it /etc/selinux/config and reboot
```

So, in this case:

```
zsh
sudo yum install compat-db43.i686 compat-db43.x86_64 compat-expat1.i686 compat-expat1.x86_64 compat-libf2c-34.i686 com
pat-libf2c-34.x86_64 compat-libtermcap.i686 compat-libtermcap.x86_64 compat-openldap.x86_64 compat-readline5.i686 comp
at-readline5.x86_64 freetype-devel.i686 freetype-devel.x86_64 freetype.i686 glibc-devel.i686 glibc.i686 libaio.i686 li
bpng-devel.i686 libpng-devel.x86_64 libstdc++.i686 libuuid-devel.i686 libuuid-devel.x86_64 libXext-devel.i686 libXext-
devel.x86_64 libXext.i686 libXft.i686 libxml2-devel.i686 libxml2-devel.x86_64 libxml2.i686 libXpm.i686 libXpm.x86_64 m
esa-libGL-devel.i686 mesa-libGL-devel.x86_64 mesa-libGL.i686 mesa-libGLU-devel.i686 mesa-libGLU-devel.x86_64 mesa-libG
LU.i686 ncurses-devel.i686 ncurses-devel.x86_64 openldap.i686 openssl098e.i686 openssl098e.x86_64 pam.i686 zlib-devel.
i686 zlib-devel.x86_64 zlib.i686
```

And also run `sudo vim /etc/selinux/config`, and edit it to be as follows:

```
zsh
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled # Changed to disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Re-running the command after installing the other dependencies:

In [4]:
```
%%bash
/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/utilities/installCheck.sh
```

```
--------------------------------------------------------------------
RedHat derived OS:
Scientific Linux CERN SLC release 6.7 (Carbon)
--------------------------------------------------------------------
Checking RedHat 6 derived OS ...
. Checking for missing rpms ...
. Checking for missing yum groups ...
. Checking that SELinuz is disabled ...
Completed check.

********************************************************************************
What to expect if there are no problems:
  No missing rpms. (You can ignore strace64 if it shows up.)
  SELinux disabling is only a recommendation if you have problems.
********************************************************************************
```

Changing afs configuration

You might want to remove most of the afs default servers which are on the file `/usr/vice/etc/CellServDB.dist` and then reboot the machine. Using only /afs/cern.ch might be enought in most cases.

Installing VOMS (for Grid)

What follows was taken from here (https://wiki.egi.eu/wiki/EGI_IGTF_Release) (check using `yum` when installing for the first time):

```
zsh
rpm -ivh http://linuxsoft.cern.ch/wlcg/sl6/x86_64/wlcg-voms-atlas-1.0.0-1.noarch.rpm
```

Install voms command (the version may change as time passes, check it using `yum provides "*/voms-proxy-init"`):

```
zsh
sudo yum install voms-clients-cpp -y
```

We now install the grid host server references:

```
zsh
sudo mkdir -p /etc/yum.repos.d;
sudo wget http://repository.egi.eu/sw/production/cas/1/current/repo-files/EGI-trustanchors.repo -P /etc/yum.repos.d/
```

```
zsh
sudo yum install ca-policy-egi-core -y
```

```
Installed:
  ca-policy-egi-core.noarch 0:1.67-1

Dependency Installed:
  ca_AAACertificateServices.noarch 0:1.67-1              ca_AEGIS.noarch 0:1.67-1                                ca
_ANSPGrid.noarch 0:1.67-1
  ca_ASGCCA-2007.noarch 0:1.67-1                         ca_AddTrust-External-CA-Root.noarch 0:1.67-1            ca
_ArmeSFo.noarch 0:1.67-1
  ca_AustrianGrid.noarch 0:1.67-1                        ca_BEGrid2008.noarch 0:1.67-1                           ca
_BG-ACAD-CA.noarch 0:1.67-1
  ca_BYGCA.noarch 0:1.67-1                               ca_BalticGrid.noarch 0:1.67-1                           ca
_BrGrid.noarch 0:1.67-1
  ca_CALG.noarch 0:1.67-1                                ca_CERN-GridCA.noarch 0:1.67-1                          ca
_CERN-Root-2.noarch 0:1.67-1
  ca_CESNET-CA-3.noarch 0:1.67-1                         ca_CESNET-CA-Root.noarch 0:1.67-1                       ca
_CNIC.noarch 0:1.67-1
  ca_CNRS2.noarch 0:1.67-1                               ca_CNRS2-Grid-FR.noarch 0:1.67-1                        ca
_CNRS2-Projets.noarch 0:1.67-1
  ca_COMODO-RSA-CA.noarch 0:1.67-1                       ca_CyGrid.noarch 0:1.67-1                               ca
_DFN-GridGermany-Root.noarch 0:1.67-1
  ca_DFN-SLCS.noarch 0:1.67-1                            ca_DZeScience.noarch 0:1.67-1                           ca
_DigiCertAssuredIDRootCA-Root.noarch 0:1.67-1
  ca_DigiCertGridCA-1-Classic.noarch 0:1.67-1            ca_DigiCertGridCA-1G2-Classic.noarch 0:1.67-1           ca
_DigiCertGridCA-1G2-Classic-2015.noarch 0:1.67-1
  ca_DigiCertGridRootCA-Root.noarch 0:1.67-1             ca_DigiCertGridTrustCA-Classic.noarch 0:1.67-1          ca
_DigiCertGridTrustCAG2-Classic.noarch 0:1.67-1
  ca_EG-GRID.noarch 0:1.67-1                             ca_FNAL-SLCS.noarch 0:1.67-1                            ca
_GermanGrid.noarch 0:1.67-1
  ca_GridCanada.noarch 0:1.67-1                          ca_HKU.noarch 0:1.67-1                                  ca
_HPCI.noarch 0:1.67-1
  ca_HellasGrid-CA-2006.noarch 0:1.67-1                  ca_HellasGrid-Root.noarch 0:1.67-1                      ca
_IGCA.noarch 0:1.67-1
  ca_IHEP-2013.noarch 0:1.67-1                           ca_INFN-CA-2006.noarch 0:1.67-1                         ca
_IRAN-GRID.noarch 0:1.67-1
  ca_InCommon-IGTF-Server-CA.noarch 0:1.67-1             ca_KEK.noarch 0:1.67-1                                  ca
_KISTI-2007.noarch 0:1.67-1
  ca_LACGridCA.noarch 0:1.67-1                           ca_LIPCA.noarch 0:1.67-1                                ca
_MARGI.noarch 0:1.67-1
  ca_MD-Grid.noarch 0:1.67-1                             ca_MREN-CA.noarch 0:1.67-1                              ca
_MYIFAM.noarch 0:1.67-1
  ca_MaGrid.noarch 0:1.67-1                              ca_NCSA-slcs-2013.noarch 0:1.67-1                       ca
_NCSA-tfca-2013.noarch 0:1.67-1
  ca_NECTEC.noarch 0:1.67-1                              ca_NERSC-SLCS.noarch 0:1.67-1                           ca
_NIIF-Root-CA-2.noarch 0:1.67-1
  ca_NIKHEF.noarch 0:1.67-1                              ca_NorduGrid.noarch 0:1.67-1                            ca
_PK-Grid-2007.noarch 0:1.67-1
  ca_PSC-Myproxy-CA.noarch 0:1.67-1                      ca_PolishGrid.noarch 0:1.67-1                           ca
_QuoVadis-Grid-ICA.noarch 0:1.67-1
  ca_QuoVadis-Root-CA1.noarch 0:1.67-1                   ca_RDIG.noarch 0:1.67-1                                 ca
_REUNA-ca.noarch 0:1.67-1
  ca_RomanianGRID.noarch 0:1.67-1                        ca_SDG.noarch 0:1.67-1                                  ca
_SRCE.noarch 0:1.67-1
  ca_SiGNET-CA.noarch 0:1.67-1                           ca_SlovakGrid.noarch 0:1.67-1                           ca
_TERENA-eScience-SSL-CA.noarch 0:1.67-1
  ca_TERENA-eScience-SSL-CA-2.noarch 0:1.67-1            ca_TERENA-eScience-SSL-CA-3.noarch 0:1.67-1             ca
_TERENAeSciencePersonalCA.noarch 0:1.67-1
  ca_TERENAeSciencePersonalCA2.noarch 0:1.67-1           ca_TERENAeSciencePersonalCA3.noarch 0:1.67-1            ca
_TRGrid.noarch 0:1.67-1
  ca_TSU-GE.noarch 0:1.67-1                              ca_UGRID.noarch 0:1.67-1                                ca
_UKeScienceCA-2A.noarch 0:1.67-1
  ca_UKeScienceCA-2B.noarch 0:1.67-1                     ca_UKeScienceRoot-2007.noarch 0:1.67-1                  ca
_UNAMgrid-ca.noarch 0:1.67-1
  ca_UNLPGrid.noarch 0:1.67-1                            ca_UTN-USERFirst-Hardware.noarch 0:1.67-1               ca
_UTN-USERTrust-RSA-CA.noarch 0:1.67-1
  ca_UTNAAAClient.noarch 0:1.67-1                        ca_UniandesCA.noarch 0:1.67-1                           ca
_cilogon-silver.noarch 0:1.67-1
  ca_pkIRISGrid.noarch 0:1.67-1                          ca_seegrid-ca-2013.noarch 0:1.67-1

Complete!
```

Also install java! This is needed by rucio/dq2:

```
zsh
sudo yum install java-1.6.0-openjdk.x86_64 -y # It may be that rucio/dq2 needs other version in the future, so make su
re to download the needed version
```

Installing development package (newer gcc, valgrind etc.)

This is quite important if you want to have one of the last versions of gcc and so on. The installation should be straight forward (as seen on CERN reference (http://linux.web.cern.ch/linux/scientific6/docs/softwarecollections.shtml)):

```
zsh
sudo yum install sl-release-scl -y
yum install devassist09 devtoolset-3 git19 httpd24 mariadb55 mongodb24 \
               mysql55 nginx16 nodejs010 rh-passenger40 perl516 php54 php55 \
               postgresql92 python27 python33 rh-python34 rh-mariadb100 rh-mongodb26 \
               rh-mysql56 rh-nginx18 rh-perl520 rh-php56 rh-postgresql94 rh-varnish4 \
               ror40 rh-ror41 ruby193 ruby200 rh-ruby22 thermostat1 v8314 sclo-vagrant1
```

However, I had an error. To solve this I edited `/etc/yum.repos.d/slc6-scl.repo` and enabled the first repository (`[slc6-scl]`) by changing the enabled variable from 0 to 1.

If you don't have this file, you can download the sample from here (http://linuxsoft.cern.ch/cern/scl/slc6-scl.repo), which is an obselete link installation found here (http://linux.web.cern.ch/linux/scientific6/docs/softwarecollections-obsolete.shtml).

To use the developer kit do:

```
zsh
scl enable devtoolset-3 zsh # or bash
```

Some other very important packages

Latex (TexLive 2015)

The version available on the SLC6 yum repositories is just a basic one. I wanted to have the full TexLive version, for doing so, these were the steps I've made (follow download and quickinstall instructions from TexLive (https://www.tug.org/texlive/)):

```
zsh
wget http://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz
tar xfvz install-tl-unx.tar.gz
rm install-tl-unx.tar.gz
cd install-tl-*
sudo ./install-tl
```

You will see the following screen:

```
Loading http://ctan.math.utah.edu/ctan/tex-archive/systems/texlive/tlnet/tlpkg/texlive.tlpdb
Installing TeX Live 2015 from: http://ctan.math.utah.edu/ctan/tex-archive/systems/texlive/tlnet
Platform: x86_64-linux => 'GNU/Linux on x86_64'
Distribution: net  (downloading)
Using URL: http://ctan.math.utah.edu/ctan/tex-archive/systems/texlive/tlnet
Directory for temporary files: /tmp

======================> TeX Live installation procedure <======================

======>   Letters/digits in <angle brackets> indicate   <=======
======>   menu items for commands or options            <=======

 Detected platform: GNU/Linux on x86_64

 <B> binary platforms: 1 out of 19

 <S> set installation scheme (scheme-full)

 <C> customizing installation collections
     47 collections out of 48, disk space required: 4011 MB

 <D> directories:
   TEXDIR (the main TeX directory):
     /usr/local/texlive/2015
   TEXMFLOCAL (directory for site-wide local files):
     /usr/local/texlive/texmf-local
   TEXMFSYSVAR (directory for variable and automatically generated data):
     /usr/local/texlive/2015/texmf-var
   TEXMFSYSCONFIG (directory for local config):
     /usr/local/texlive/2015/texmf-config
   TEXMFVAR (personal directory for variable and automatically generated data):
     ~/.texlive2015/texmf-var
   TEXMFCONFIG (personal directory for local config):
     ~/.texlive2015/texmf-config
   TEXMFHOME (directory for user-specific files):
     ~/texmf

 <O> options:
   [ ] use letter size instead of A4 by default
   [X] allow execution of restricted list of programs via \write18
   [X] create all format files
   [X] install macro/font doc tree
   [X] install macro/font source tree
   [ ] create symlinks to standard directories

 <V> set up for portable installation

Actions:
 <I> start installation to hard disk
 <H> help
 <Q> quit
```

Insert I and wait the installation. If you want, you can specify a custom installation by changing the installation scheme (S). The default installation needs 4 GB space.

Now we add the `TexLive` paths for all users:

```
zsh
cat << EOF > TexLive_env.sh
export PATH="/usr/local/texlive/2015/bin/x86_64-linux:\$PATH"
export INFOPATH="/usr/local/texlive/2015/texmf-dist/doc/info:\$INFOPATH"
export MANPATH="/usr/local/texlive/2015/texmf-dist/doc/man:\$MANPATH"
EOF
test \! -d /etc/profile.d/ && sudo mkdir /etc/profile.d/
sudo mv TexLive_env.sh /etc/profile.d/ # This will be sourced next time you log-in.
source /etc/profile.d/TexLive_env.sh
```

If you want to test the installation, do:

```
zsh
latex small2e
```

ATLAS, blas libraries

```
zsh
sudo yum install -y atlas.x86_64 atlas-devel.x86_64 blas.x86_64 blas-devel.x86_64
```

X11 Access

sudo yum install -y xorg-x11-xauth

Armadillo

```
zsh
sudo yum install -y armadillo
```

screen

```
zsh
sudo yum install screen.x86_64 -y
```

pip install

```
zsh
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
# Update via: pip install -U pip
```

pyenv

This is needed to change the ipython installation accordenly to the environment you are using on RootCore or Athena. The installation will be done for your cern account, and not for the node.

With it you can change python version and installation options. Thus, you can run python build with debug options and so on.

```
zsh
git clone https://github.com/yyuu/pyenv.git ~/.pyenv
shell_config="$HOME/.bash_profile" # If using zsh set it to $HOME/.zshenv
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> $shell_config
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> $shell_config
echo 'eval "$(pyenv init -)"' >> $shell_config
```

NOTE: You may also want to not include it directly on bash_profile, but rather a file you could source later on. This is something you might consider, specially because RootCore won't set the python to the cvmfs version if it detects that you are using *pyenv*.

For being able to install and compile python on my afs account using the node configuration I had to also install the following packages:

```
zsh
sudo yum install bzip2-libs.x86_64 readline-devel.x86_64 sqlite-devel.x86_64 openssl-devel.x86_64 -y
```

Usage:

```
zsh
# Install python 2.7.4 for instance
pyenv install 2.7.4
# maybe you will want to install it with a shared library (I've set unicode to ucs4 b/c that's the version CERN uses):
env PYTHON_CONFIGURE_OPTS="--enable-shared --enable-unicode=ucs4" pyenv install -v 2.7.4
# Change the python to the locally installed 2.7.4 and also add the system python to path:
pyenv global 2.7.4 system
# Now you can both use the python
```

Btw: In order to install numpy and scipy, I had to use easy_install instead of pip:

```
zsh
easy_install numpy
easy_install pandas
pip install tornado
easy_install bokeh
```

Maybe you will need to change every pip install by easy_install in order to use the right unicode...

IPython

Note: ipython on current SLC6 version won't run the notebook because it seems not to be possible to install it on version 2.6.

```
zsh
sudo yum install python-ipython-doc.noarch python-ipython-console.noarch -y
```

IPython Notebook and others:

```
zsh
pip install ipython[notebook] # or ipython\[notebook\] on zsh
pip install pyzmq
pip install jinja2
pip install tornado
```

For forwarding the notebook so that it can be accessed outside, you will have to do as follows (reference (http://www.hydro.washington.edu/~jhamman/hydro-logic/blog/2013/10/04/pybook-remote/)):

```
zsh
# When inside CERN network
ssh -N -f -L localhost:<any_port_you_want_to_use>:localhost:<port_used_by_jupyter> <your_account>@<your_node>
# When outside CERN network
ssh -f -L 8080:localhost:8080 <your_account>@lxplus.cern.ch ssh -f -L 8080:localhost:<port_used_by_jupyter> -N <your_a
ccount>@<your_node>
```

If you need to end the port forwarding, find the process id by doing:

```
zsh
ps aux | grep ssh
```

And then kill it using `kill <PID>`

Jupyter

Since you probably will get installed IPython 4.x (unless you use pip install for some wanted version), you will need to install jupyther as well. This is done following the documentation available here (http://jupyter.readthedocs.org/en/latest/install.html#existing-python-new-jupyter).

I've considered the case where we don't use anaconda:

```
zsh
pip install jupyter # I'm considering that you are installing using pyenv environment,
                    # then run command with admin rights.
# We also need to install yalm python support:
pip install pyyaml
# Either:
pip install ipywidgets
pip install ipywidgets --upgrade
```

Extending Jupyter with plugins

Installing plugins will definetly improve the jupyther capabilities.

Several extensions in one pack

First we will install the IPython-notebook-extensions using the github.

```
zsh
pip install psutil
git clone https://github.com/ipython-contrib/IPython-notebook-extensions
cd IPython-notebook-extensions
python setup.py install
cd -
```

Then on your browser add after the port `/nbextensions` and turn on those you like.

Calico-spell

Instead using aspell, jupyter indicates the use of calico aspell.

On notebook, open a code cell and run (this can be also run on terminal):

In [2]:
```
!ipython install-nbextension --user https://bitbucket.org/ipre/calico/downloads/calico-spell-check-1.0.zip
```

```
downloading https://bitbucket.org/ipre/calico/downloads/calico-spell-check-1.0.zip to /tmp/tmpI4ePbv/calico-spell-check-1.0.zip
extracting /tmp/tmpI4ePbv/calico-spell-check-1.0.zip to /afs/cern.ch/user/w/wsfreund/.local/share/jupyter/nbextensions
```

Then add another cell to load, in each notebook:

In [ ]:
```
%%javascript
require(['base/js/utils'],
function(utils) {
   utils.load_extensions('calico-spell-check');
});
```

If it is wanted to have it available on every jupyter notebook, modify `~/.jupyter_tuningtools/custom/custom.js` by adding:

In [ ]:
```
# %load ~/.jupyter_tuningtools/custom/custom.js
# Add to the file:
require(['base/js/utils'],
        function(utils) {
            utils.load_extensions('calico-spell-check');
        });
```

Aspell

References 1 (https://github.com/ipython-contrib/IPython-notebook-extensions/wiki/aspell), 2 (http://calicoproject.org/ICalico#Installation_2). **NOTE**: Still working on this solution...

```
zsh
# Aspell plugin
sudo yum install aspell aspell-devel.x86_64 -y
git clone https://github.com/WojciechMula/aspell-python.git
cd aspell-python
python setup.2.py build # or setup.3 for python 3
python setup.2.py install
cd -
wget http://ftp.gnu.org/gnu/aspell/dict/en/aspell6-en-2015.04.24-0.tar.bz2
tar xfvj aspell6-en-2015.04.24-0.tar.bz2
rm aspell6-en-2015.04.24-0.tar.bz2
cd aspell6-en-2015.04.24-0/
./configure
sudo make install
cd -
aspell dump dicts
nohup python ~/.local/share/jupyter/nbextensions/usability/aspell/ipy-aspell-server.py & # Start ipy aspell server
# To change the language on aspell modify the line s = aspell.Speller('lang', 'en') content on ipy-aspell-server
```

Table Of Contents

With the table of contents plugin available, add the following markdown text in the first cell of the notebook:

```
<h1 id="tocheading">Table of Contents</h1>
<div id="toc"></div>
```

Use this other markdown cell to expand the table of contents information (adapted from 1 (http://blog.nextgenetics.net/?e=102) and 2 (https://github.com/kmahelona/ipython_notebook_goodies)). It is a good idea to have it as the last cell, since after you run it with shift-enter, it will get invisible, but you can click on it if you find the small area which it is occupying. This code is important to enable the TOC code above to get expanded.

```
<script type="text/javascript">
    show=true;
    function toggle(){
        if (show){
            $('div.input').hide();
        }else{
            $('div.input').show();
        }
        show = !show
    }
$.getScript('https://kmahelona.github.io/ipython_notebook_goodies/ipython_notebook_toc.js')
</script>
<a href="javascript:toggle()" target="_self"></a>
```

Maybe you'll also need to download the TOC json files over as described here, but rather doing as follows:

```
zsh
curl -L https://rawgithub.com/minrk/ipython_extensions/master/nbextensions/toc.js > $JUPYTER_PATH/nbextensions/usabili
ty/toc2/toc.js
curl -L https://rawgithub.com/minrk/ipython_extensions/master/nbextensions/toc.css > $JUPYTER_PATH/nbextensions/usabil
ity/toc2/toc.css
```

After you will have to do edit file `~/.jupyter/nbconfig/notebook.json` (if you don't have, you can create one by accessing the `/nbextensions` folder on the jupyter browser, or just create one with the following template):

```
{
  "load_extensions": {
    "usability/toc2/toc": true
  },
}
```

Gist upload button

One very useful thing is to have a buttom to upload the notebook to your gist with only one click. This can be done as follows:

```
zsh
JUPYTER_PATH=$(python -c "from jupyter_core.paths import jupyter_data_dir; print(jupyter_data_dir());")
mkdir -p $JUPYTER_PATH/nbextensions/usability/gist
curl -L https://rawgithub.com/minrk/ipython_extensions/master/nbextensions/gist.js > $JUPYTER_PATH/nbextensions/usabil
ity/gist/gist.js
```

Activate the plugin by editing `~/.jupyter/nbconfig/notebook.json` and adding the following line to "load_extensions":

```
"usability/gist/gist": true
```

Then go to github token generation (https://github.com/settings/tokens) and create one for you. I recommend that you add only gist permissions for this token.

Copy the token generated and keep it somewhere so that you can copy and paste it everytime you want to upload the gist to github.

NOTE: If you for some reason delete the gist file on github and cannot upload it anymore using the upload button, click on the menu edit button and choose `Edit Notebook Data`. Once there, just delete the gist file flag.

NOTE2: When you rename one gist file, it will keep the old naming file on the gist. You can simply git clone your gist, remove the old files and commit, as:

```
zsh
git clone https://gist.github.com/1234567.git
cd 1234567
rm oldfile
git commit -a -m "Removed old file"
git push
```

Exporting pdfs and others

I had issues when trying to install pandoc which is needed to create pdfs due to conflict with SLC-EPEL and epel repository management.

The solution I found needs to temporarly overwrite CERN epel repository management to the latest management version so that we can retrieve pandoc rpms (which is 600 MB large and contains many libraries).

To avoid other conflicts later on, I moved the epel repository management files to inactive status. If you have to update pandoc or uninstall it, just remove the -inactive flags from the epel files.

```
zsh
pip install six # Needed to export pdfs and so on.
## Installing pandoc
# Here I do something which should be avoided, but I couldn't find another solution but to force
# the temporary installation of the epel package management:
# This is only needed if not using SLC6 CERN: sudo rpm -ivh --force http://dl.fedoraproject.org/pub/epel/6/x86_64/epel
-release-6-8.noarch.rpm
# We then install the pandoc packages:
sudo yum install pandoc ghc-pandoc ghc-pandoc-devel ghc-pandoc-types ghc-pandoc-types-devel -y
# And disable the epel package management
# Only if not using SLC6 CERN: sudo mv /etc/yum.repos.d/epel.repo /etc/yum.repos.d/epel.repo-inactive
# Only if not using SLC6 CERN: sudo mv /etc/yum.repos.d/epel-testing.repo /etc/yum.repos.d/epel-testing.repo-inactive
# Install python support to pandoc
pip install pandoc
```

Now you will be able to export HTML files, but for the pdf we will also need to install latex... this was explained before here. I had issues when trying to export, for better debugging your issues, you can use the following command line to check the errors:

```
zsh
ipython nbconvert --to pdf <NotebookSample.ipynb>
```

I had then to install the following package:

```
zsh
pip install requests
```

I've also opened an issue (https://github.com/jupyter/nbviewer/issues/513#issuecomment-146835643) on GitHub to try to solve it. First, I've updated jupyter nbconvert to be as the HEAD, for this, I've needed to execute the following commands:

```
zsh
sudo yum install libcurl-devel -y
export PYCURL_SSL_LIBRARY="nss"
easy_install pycurl
# Still waiting for updates:
pip install -e git+https://github.com/jupyter/nbconvert#egg=nbconvert
```

Jupyter Drive

This doesn't work, I'll wait until documentation and community suport improves for IPython 4.

```
zsh
git clone git://github.com/jupyter/jupyter-drive.git
cd jupyter-drive
python setup.py build
python setup.py install
## On >=2.7
#python -m jupyterdrive
## Otherwise run like this
#python -m jupyterdrive.__main__
```

Dark code cells

#TODO Follow installation from here (https://www.pfenninger.org/posts/ipython-extension-to-toggle-dark-code-cells/)

openstack

The openstack installation deserves a little bit more attention. During the tutorial writing time, the project which would make the openstack available on SLC6 removed the support for their binaries on Redhat 6, where the SLC6 is build upon (at least that's what I understood so far). So, it isn't possible to install using the epel and RDO project repositories.

The solution I found is available here (https://clouddocs.web.cern.ch/clouddocs/clients/linux_client_installation.html).

For future reference, if the SLC version is upgraded, maybe the link you need to follow the instructions is available here (http://clouddocs.web.cern.ch/clouddocs/advanced_topics/installing_your_own_openstack.html) (it doesn't work right now).

```
zsh
sudo yum update -y
sudo rpm -ivh --nodeps http://cbs.centos.org/repos/cloud6-openstack-juno-candidate/x86_64/os/Packages/centos-release-o
penstack-juno-2.el6.noarch.rpm
sudo yum install -y /usr/bin/{nova,glance,cinder,keystone,openstack}
```

Important you will also need to install for having access to command `virt-sysprep` used for cloning snapshots:

```
zsh
sudo yum install libguestfs-tools.x86_64 -y
```

Enabling X11 forwarding

TODO

Some command you need is missing?

If a command is missing, it can be found on yum by doing (if not, duckduckgo (https://duckduckgo.com/) or google (https://google.com) it):

```
zsh
sudo yum provides "*/command"
```

Cloning the VM and saving snapshots for future node recovery

It's important to keep your VM configuration saved by doing snapshots from time to time. This is explained here (http://clouddocs.web.cern.ch/clouddocs/using_openstack/backups.html), in this topic we will walk through cloning our just created VM to another node as it is possible to have two large nodes for the current CERN quota.

In order to create a snapshot, you can do it via website here (https://openstack.cern.ch/dashboard/project/instances/). Latter we also explain how to do so via command line.

Configure your openstack

First, go to the cloud infrastructure site on tab access and security (https://openstack.cern.ch/dashboard/project/access_and_security/), click on download open stack rc file. Send it to your cern afs account and source it every time you want to use openstack or nova.

Create and start a snapshot

```
zsh
openstack server image create --name <snapshotname> --wait <nodename> # Create a snapshot via command line
openstack image show <snpashotname> # Show snapshot information
# openstack server rebuild --image <snapshotname> <nodename> ## Re-roll node to a previous snapshot if needed
openstack image save --file <snapshot_localfile_path>.qcow2 <snapshotname> # Copy snapshot to local host
# For this command make sure you have followed the important note on openstack installation!
virt-sysprep -a <snapshot_localfile_path>.qcow2 # Clear node information for making possible to create another node fr
om this image.
openstack image create --file <snapshot_localfile> --property os=LINUX --disk-format=qcow2 --container-format=bare <cl
ean_image_name>
# Upload to glance the cleaned snapshot
# Now, create the new host:
openstack server create --key-name <your_key_name> --flavor m1.large --image <clean_image_name> <clone_nodename>
```

After creating the new machine node, it is still needed to correct the Kerberos `keytab` file. You can do this by doing as root:

```
zsh
mv krb5.keytab krb5.keytab-bkg
cern-get-keytab
```

You may also need to update time on the snapshotted node:

```
zsh
service ntpd stop
ntpdate lxplus.cern.ch
service ntpd start
```

You might need to update the keytab from the host:

```
cern-get-keytab --hostname <node_name>
# Use `cern-config-keytab -v -f` to fix the configuration pointing to the correct host
```

and check on another host (lxplus):

```
kinit -R; kvno host/<node_ip>@CERN.CH
```

Finally, it is also needed to change the kerberos autority from the host (I found help here (https://twiki.cern.ch/twiki/bin/view/LinuxSupport/SSHatCERNFAQ#Kerberos_authentication_not_work), here (https://twiki.cern.ch/twiki/bin/view/LinuxSupport/SSHatCERNFAQ#Kerberos_authentication_not_work), here (http://clouddocs.web.cern.ch/clouddocs/using_openstack/backups.html), and contacted their support which be done via helpdesk). In order to do this, first we need to correct CERN information of the node in the LanDB (you can check the information from the node in the LanDB by accessing https://network.cern.ch/sc/fcgi/sc.fcgi?Action=SearchForDisplay&DeviceName=node_name (https://network.cern.ch/sc/fcgi/sc.fcgi?Action=SearchForDisplay&DeviceName=node_name)): **This step can be ignored as it was updated in the documentation above, however it might be useful reference if you forget to use the -os during the image creation**

```
openstack server set --property landb-os="LINUX" <node_name>
klist -kt
```

Wait 1 day (it will take some hours, but quite a few) and then do cern-get-keytab as stated above.

Issues on the node

Hope you can find some helpful information below, where I catalog some issues that I have faced.

Out of Space related issues

These errors may happen when the node get out of space.

Empty afs folders which shouldn't be empty

Do:

```
fs setca 1
fs setca 0
```

This will reset your afs cache (taken from here (https://lists.openafs.org/pipermail/openafs-info/2004-November/015351.html)).

Connection timeout to specified afs folders

You can flush a volume or a mount space by doing:

```
fs flushmount /afs/cern.ch/work
fs flushvolume /afs/cern.ch/work
```

Taken from here (https://lists.openafs.org/pipermail/openafs-info/2012-August/038457.html)

Accessing host as root

Instead of logging with your user, if you are the owner of the ssh key-pair, you can then just change the ssh tunnel user to root to log into the root account. For instance, you could have the following tunnel aliases:

```
zsh
alias prefered='ssh -A -t -l <your_account> lxplus.cern.ch ssh -A -t -l <your_account> <your_host_name>'
alias preferedroot='ssh -A -t -l <your_account> lxplus.cern.ch ssh -A -t -l root <your_host_name>'
```

Optional (and recommended) packages

Vim 7.4

To install recent VIM version do:

```
sudo yum install vim-enhanced.x86_64
```

However, if you need `lua` and `X11` access on VIM, you will have to download an install from source:

Install lua development:

```
zsh
yum -y install lua-devel
```

Install lua JIT:

```
zsh
git clone http://luajit.org/git/luajit-2.0.git
cd luajit-2.0
make
make install
```

For X11, you'll need to install the development package to have access to header files:

```
zsh
yum groupinstall "X Window System"
yum groupinstall "CERN Addons X11"
yum install /usr/include/X11/Intrinsic.h
```

And the last dependence, if you need ruby, install it:

```
zsh
yum install ruby ruby-devel
```

At last but not least, install vim (you'll have to use a valid python2.7 path):

```
zsh
git clone https://github.com/vim/vim.git
cd vim
./configure --with-features=huge  \
            --enable-multibyte \
            --enable-rubyinterp \
            --enable-pythoninterp \
            --with-python-config-dir=/afs/cern.ch/user/w/wsfreund/.pyenv/versions/2.7.4/lib/python2.7/config \
            --enable-luainterp \
            --with-x --enable-cscope --prefix=/usr --with-luajit
make VIMRUNTIMEDIR=/usr/share/vim/vim74
sudo make install
```

htop

```
sudo yum install htop.x86_64
```

evince

Used for viewing pdfs.

```
sudo yum install evince
```

Unsolved issues

SSL handshake issues

It seems that I'm having issues with the nss SSL library. I'll have to check how to fix this, as it is stopping me from using curl and other things dependent on SSL, such as pbook.

Managing information

Here, I try to catalog what commands are usuful for retrieving and changing information on the images and nodes.

First, you can use glance to check the images information:

In [1]:
```
%%bash
glance --help
```

```
usage: glance [--version] [-d] [-v] [--get-schema] [--timeout TIMEOUT] [--no-ssl-compression] [-f] [--os-image-url
OS_IMAGE_URL] [--os-image-api-version OS_IMAGE_API_VERSION] [-k] [--os-cert OS_CERT]
              [--cert-file OS_CERT] [--os-key OS_KEY] [--key-file OS_KEY] [--os-cacert <ca-certificate-file>] [--ca
-file OS_CACERT] [--os-username OS_USERNAME] [--os-user-id OS_USER_ID]
              [--os-user-domain-id OS_USER_DOMAIN_ID] [--os-user-domain-name OS_USER_DOMAIN_NAME] [--os-project-id
OS_PROJECT_ID] [--os-project-name OS_PROJECT_NAME]
              [--os-project-domain-id OS_PROJECT_DOMAIN_ID] [--os-project-domain-name OS_PROJECT_DOMAIN_NAME] [--os
-password OS_PASSWORD] [--os-tenant-id OS_TENANT_ID] [--os-tenant-name OS_TENANT_NAME]
              [--os-auth-url OS_AUTH_URL] [--os-region-name OS_REGION_NAME] [--os-auth-token OS_AUTH_TOKEN] [--os-s
ervice-type OS_SERVICE_TYPE] [--os-endpoint-type OS_ENDPOINT_TYPE]
              <subcommand> ...

Command-line interface to the OpenStack Images API.

Positional arguments:
  <subcommand>
    image-create        Create a new image.
    image-delete        Delete specified image(s).
    image-download      Download a specific image.
    image-list          List images you can access.
    image-show          Describe a specific image.
    image-update        Update a specific image.
    member-create       Share a specific image with a tenant.
    member-delete       Remove a shared image from a tenant.
    member-list         Describe sharing permissions by image or tenant.
    bash-completion     Prints all of the commands and options to stdout so that the
    help                Display help about this program or one of its subcommands.

Optional arguments:
  --version             show program's version number and exit
  -d, --debug           Defaults to env[GLANCECLIENT_DEBUG].
  -v, --verbose         Print more verbose output
  --get-schema          Ignores cached copy and forces retrieval of schema that generates portions of the help text
. Ignored with API version 1.
  --timeout TIMEOUT     Number of seconds to wait for a response
  --no-ssl-compression  Disable SSL compression when using https.
  -f, --force           Prevent select actions from requesting user confirmation.
  --os-image-url OS_IMAGE_URL
                        Defaults to env[OS_IMAGE_URL]. If the provided image url contains a a version number and `-
-os-image-api-version` is omitted the version of the URL will be picked as the image
                        api version to use.
  --os-image-api-version OS_IMAGE_API_VERSION
                        Defaults to env[OS_IMAGE_API_VERSION] or 1.
  -k, --insecure        Explicitly allow glanceclient to perform "insecure SSL" (https) requests. The server's cert
ificate will not be verified against any certificate authorities. This option should be
                        used with caution.
  --os-cert OS_CERT     Path of certificate file to use in SSL connection. This file can optionally be prepended wi
th the private key.
  --cert-file OS_CERT   DEPRECATED! Use --os-cert.
  --os-key OS_KEY       Path of client key to use in SSL connection. This option is not necessary if your key is pr
epended to your cert file.
  --key-file OS_KEY     DEPRECATED! Use --os-key.
  --os-cacert <ca-certificate-file>
                        Path of CA TLS certificate(s) used to verify the remote server's certificate. Without this
option glance looks for the default system CA certificates.
  --ca-file OS_CACERT   DEPRECATED! Use --os-cacert.
  --os-username OS_USERNAME
                        Defaults to env[OS_USERNAME].
  --os-user-id OS_USER_ID
                        Defaults to env[OS_USER_ID].
  --os-user-domain-id OS_USER_DOMAIN_ID
                        Defaults to env[OS_USER_DOMAIN_ID].
  --os-user-domain-name OS_USER_DOMAIN_NAME
                        Defaults to env[OS_USER_DOMAIN_NAME].
```

```
  --os-project-id OS_PROJECT_ID
                        Another way to specify tenant ID. This option is mutually exclusive with --os-tenant-id. De
faults to env[OS_PROJECT_ID].
  --os-project-name OS_PROJECT_NAME
                        Another way to specify tenant name. This option is mutually exclusive with --os-tenant-name
. Defaults to env[OS_PROJECT_NAME].
  --os-project-domain-id OS_PROJECT_DOMAIN_ID
                        Defaults to env[OS_PROJECT_DOMAIN_ID].
  --os-project-domain-name OS_PROJECT_DOMAIN_NAME
                        Defaults to env[OS_PROJECT_DOMAIN_NAME].
  --os-password OS_PASSWORD
                        Defaults to env[OS_PASSWORD].
  --os-tenant-id OS_TENANT_ID
                        Defaults to env[OS_TENANT_ID].
  --os-tenant-name OS_TENANT_NAME
                        Defaults to env[OS_TENANT_NAME].
  --os-auth-url OS_AUTH_URL
                        Defaults to env[OS_AUTH_URL].
  --os-region-name OS_REGION_NAME
                        Defaults to env[OS_REGION_NAME].
  --os-auth-token OS_AUTH_TOKEN
                        Defaults to env[OS_AUTH_TOKEN].
  --os-service-type OS_SERVICE_TYPE
                        Defaults to env[OS_SERVICE_TYPE].
  --os-endpoint-type OS_ENDPOINT_TYPE
                        Defaults to env[OS_ENDPOINT_TYPE].

See "glance help COMMAND" for help on a specific command.
```

Adding ROOT keys

Configure ROOT ssh keys by adding them to the file `/root/.ssh/authorized_keys`.