|  |  |  |
| --- | --- | --- |
| Document Name | **Setting Up GSKY Environment** | |
| Version: 1.0 | 18 Oct, 2018 | |
| Revision: 1.0 | 18 Oct, 2018 | |
| Author(s) | Dr. Arapaut V. Sivaprasad | |
| Approved By |  | Date: |

Contents

[INTRODUCTION 2](#_Toc527728003)

[TL;DR 2](#_Toc527728004)

[SHORT INSTRUCTIONS 2](#_Toc527728005)

[Setting up a VM on Tenjin 2](#_Toc527728006)

[Building the GSKY environment 5](#_Toc527728007)

[DETAILS OF INSTALLATION 5](#_Toc527728008)

[Reverse pull GitHub repository 9](#_Toc527728009)

Setting Up GSKY Environment

**DISCLAIMER**

This document is for beginners to setup GSKY environments on a virtual machine. Experienced users may use it as a reference. While all care has been taken to ensure the absence of errors, such errors can creep up when and if component programs change in content and/or locations. Hence, this document needs continual updating. Text from external sites are quoted in this document, with a link to the original source as credit for such usage, and no claim of ownership is made of their contents.

# INTRODUCTION

This document is a detailed instruction set for creating an instance of a Virtual Machine (VM) on tenjin.nci.org.au and to build the GSKY environment on it. If you have already set it up once, then may require only the [short instructions](#short_instructions). To refresh memory, read the TL;DR section or the detailed text.

# TL;DR

* Create a VM instance on tenjin.nci.org.au.
  + Choose the option that gives at least **4GB** RAM and **80GB** disk.
  + Choose **CentOS 7** as the operating system.
* Remember that the VM is accessible only from within NCI network.
  + Use the ethernet for connection when in the office.
  + WiFi through ANU-Secure will not be enough, unless your PC’s IP is added to the [firewall](#firewall).
  + Use VPN to connect from remote locations or from within the office.
* The GSKY environment must be setup on each new VM.
  + [Short](#short_instructions) instructions
  + [Detailed](#build_gsky) instructions
* If no error during installation, the first step will take about 45 minutes, and the second about 15 min.
  + Ignore the warning messages during the installation.
  + Pay attention to error (crash) and exit messages.
  + If no error, both steps will display “Finished…”

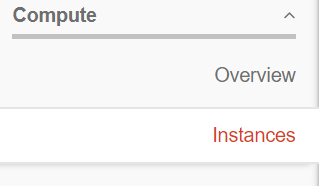
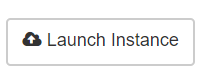
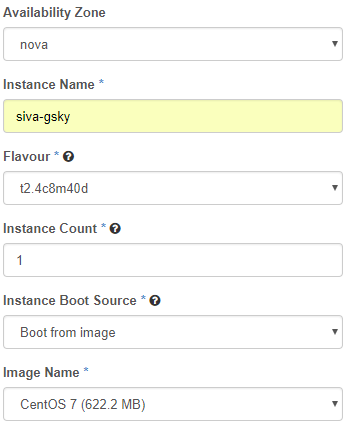
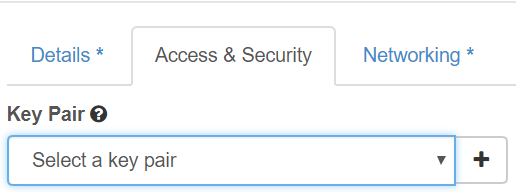
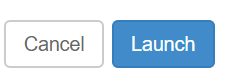
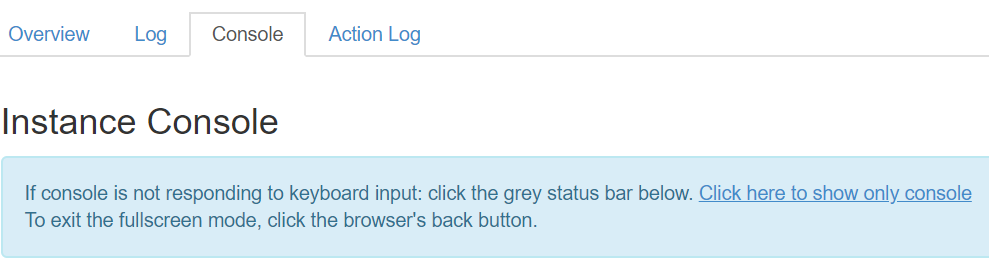
# SHORT INSTRUCTIONS

* **[Build](#build_a_vm) a VM**
  + <https://tenjin.nci.org.au/dashboard/auth/login/>
  + [Instances](https://tenjin.nci.org.au/dashboard/project/instances/) >> Launch Instances >> Specify the details >> Launch
  + Wait at least 15 minutes for setup.
* [**Build**](#build_gsky) **GSKY**
  + Login to the VM via SSH
  + *cd ~*
  + *git clone* [*https://github.com/nci/gsky.git*](https://github.com/nci/gsky.git)
  + *sudo -i*
  + *cd ~/gsky/install*
  + *./build\_all.sh*

# Setting up a VM on Tenjin

The VM instances on ‘https://tenjin.nci.org.au/dashboard/project/instances/’ are where the GSKY installation and servers are built and managed.

**HOW TO**

* <https://tenjin.nci.org.au/dashboard/project/>
* Click on ‘Instances’ and then on ‘Launch Instance’
* 
* 
* Fill in details as per the example below.
* 
* [Create](#create_ssh_key)/choose a key pair if you wish to use it for password-less login. Otherwise, leave it as blank.[\*](#create_ssh_key)
  + Click the + icon to create a new key pair.
* 
* Create the instance by clicking ‘Launch’
* 
* Wait 15 min for the installation to complete.
* Click on the instance name.
* 
* Click on ‘Console’ tab and then on the hypertext link, “Click here…”
* 
* Login with NCI username/password when prompted. This is now a regular VM console.
* If you are on the NCI network, either through ethernet or VPN, the following step is not required.
* Change to ‘sudo -I’ and add the IP address of your local PC to the iptables
  + This step is necessary only when not on the NCI network through ethernet.
    - cd /etc/puppetlabs/code/environments/production/
    - cat > hieradata/node/siva-gsky.yaml (same name as instance.yaml)
      * nci::firewall::ruleset::ssh::sources\_array:
        + – “130.56.84.195”
    - Or
      * vim hieradata/node/siva-gsky.yaml
      * Change the IP address
    - puppet-apply
    - iptables –list

**Create an SSH key pair:**

NOTE: You will only need this if intending to login and/or transfer files between the VM and another Unix server. There is not much use of it under Microsoft Windows, as ‘putty’ and ‘winscp’ cannot use the openssh keys. Converting them to putty key (\*.ppk) does not work either, as the VM does not recognise it.

To create:

* SSH to your “home machine”

*ssh-keygen -t rsa -f username.key*

* Copy the content of ‘username.key.pub’ and paste it into the box.
* Keep the ‘username.key’ safely and securely.

# Building the GSKY environment

In order to run an OWS server, it is required to install GSKY and its dependencies on the VM. This step is required on each new VM being created. UNIX commands are *highlighted in blue.*

**STEPS**

* Install the dependencies and GSKY
  + Clone the git repo from below, or get the two shell scripts highlighted in red.

Clone it under your username so that files can be edited by yourself.

* *cd ~*
* *git clone https://github.com/nci/gsky.git*
  + - * install/build\_all.sh
      * install/pgsql2shp
      * install/shp2pgsql
      * 26 other files and dirs.

*sudo -i*

*cd gsky/install*

*./build\_all.sh [*[*details*](#build_deps)*]*

This will take several minutes to complete. It is normal to see many warning messages. Ignore the warnings, but make sure that there is no fatal error. If installation goes to completion, it will display “\*\*\*\* Finished installing the GSKY server. \*\*\*\*”

# DETAILS

## sudo -i

This command starts a super user session using the current shell (e.g. /bin/bash) so that all system commands can be run without having to do a ‘sudo command’ every time. This method, however, has the disadvantage that you are now not in your username’s login shell and therefore the environments, aliases, etc. for your shell are not available. Another risk is that now you are a super user and can accidentally delete/modify things you could not have done before. The safe way is to use ‘sudo command’. For this, avoid the ‘sudo -i’ and prepend ‘sudo ’ to all the commands shown.

## yum groupremove "Development Tools"

Remove the previously installed programs, if any, so that the latest versions will be installed when the next command is run.

## yum groupinstall "Development Tools" [[Ref](https://serverfault.com/questions/274279/whats-the-difference-between-yum-groupinstall-vs-regular-yum-install)]

The groupinstall command installs a bundle of packages that are designated as a group so that you don't need to install a bunch of individual packages yourself to have all of the features. So yum groupinstall "Development Tools" would install a bunch of packages necessary for development, such as gcc, make, git, etc.

yum grouplist will list the possible groups. yum groupinfo groupname will show what packages are members of groupname (you will need "quotes" if there are spaces in the group name).

The regular install just installs individual packages (and their dependencies) by name.

The following programs are not part of the “groupinstall Development Tools” and, therefore, must be installed separately.

## yum install postgresql

Install the PostgreSQL Database.

## yum install postgis

Install PostGIS

## yum install wget

The program, ‘wget’, is used to get several packages required for GSKY.

## yum install cmake

## yum install python-devel

**./build\_all.sh**

This script installs several dependencies as given below. It is required to run this script only once.

1. **Independent JPEG Group's free JPEG software [**[**Ref**](http://www.ijg.org/files/README)**]**

***wget -q*** [***http://www.ijg.org/files/jpegsrc.v${v}.tar.gz***](http://www.ijg.org/files/jpegsrc.v$%7bv%7d.tar.gz)

This package contains C software to implement JPEG image encoding, decoding, and transcoding. JPEG is a standardized compression method for full-color and grayscale images.

The distributed programs provide conversion between JPEG JFIF format and image files in PBMPLUS PPM/PGM, GIF, BMP, and Targa file formats. The core compression and decompression library can easily be reused in other programs, such as image viewers. The package is highly portable C code; we have tested it on many machines ranging from PCs to Crays. **jpegsrc.vN.tar.gz** contains source code, documentation, and test files for release N in Unix format.

1. **OPENJPEG Library and Applications [**[**Ref**](https://github.com/uclouvain/openjpeg/blob/master/README.md)**]**

***wget -q -O openjpeg-v${v}.tar.gz https://github.com/uclouvain/openjpeg/archive/v${v}.tar.gz***

OpenJPEG is an open-source JPEG 2000 codec written in C language. It has been developed in order to promote the use of [JPEG 2000](http://www.jpeg.org/jpeg2000), a still-image compression standard from the Joint Photographic Experts Group ([JPEG](http://www.jpeg.org/)). Since April 2015, it is officially recognized by ISO/IEC and ITU-T as a [JPEG 2000 Reference Software](http://www.itu.int/rec/T-REC-T.804-201504-I!Amd2).

NOTE: The above link appears to be an older version that has been archived. Their latest [release](https://github.com/uclouvain/openjpeg/releases/download/v2.3.0/openjpeg-v2.3.0-linux-x86_64.tar.gz) has a different tar structure. The current ‘build\_deps.sh’ is unable to use that archive. It is not known whether we specifically need the presumably older release.

1. **GEOS - Geometry Engine, Open Source**[**[Ref]**](https://www.osgeo.org/projects/geos/)

***wget -q http://download.osgeo.org/geos/geos-${v}.tar.bz2***

GEOS (Geometry Engine - Open Source) is a C++ port of the [​Java Topology Suite](http://tsusiatsoftware.net/jts/main.html) (JTS). As such, it aims to contain the complete functionality of JTS in C++. This includes all the [​OpenGIS Simple Features for SQL](http://www.opengeospatial.org/standards/sfs) spatial predicate functions and spatial operators, as well as specific JTS enhanced topology functions.

**Capabilities**

* Geometries: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection
* Predicates: Intersects, Touches, Disjoint, Crosses, Within, Contains, Overlaps, Equals, Covers
* Operations: Union, Distance, Intersection, Symmetric Difference, Convex Hull, Envelope, Buffer, Simplify, Polygon Assembly, Valid, Area, Length,
* Prepared geometries (pre-spatially indexed)
* STR spatial index
* OGC Well Known Text (WKT) and Well Known Binary (WKB) encoders and decoders.
* C and C++ API (C API gives long term ABI stability)
* Thread safe (using the reentrant API)

NOTE: The v3.6.2 being downloaded in the script is an older version (2017). Latest stable release is 3.7.0 (Sep 2018)

1. **Cartographic Projection Procedures for the UNIX Environment [**[**Ref**](http://download.osgeo.org/proj/OF90-284.pdf)**]**

***wget -q http://download.osgeo.org/proj/proj-${v}.tar.gz***

Program proj (release 3) is a standard Unix filter function which converts geographic longitude and latitude coordinates into cartesian coordinates, (λ, φ) → (x, y), by means of a wide variety of cartographic projection functions. For many of the projection functions the inverse conversion, (x, y) → (λ, φ), can also be performed.

NOTE: The v5.1.0 being downloaded in the script is an older version (2017). Latest stable release is v5.2.0 (Sep 2018)

1. **Zlib Data Compression Library [**[**Ref**](https://en.wikipedia.org/wiki/Zlib)**][**[**Ref**](ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4/)**]**

***wget -q*** [***ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4/zlib-${v}.tar.gz***](ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4/zlib-$%7bv%7d.tar.gz)

zlib 1.2.8 is a general purpose data compression library. All the code is thread safe. The data format used by the zlib library is described by RFCs (Request for Comments) 1950 to 1952 in the files http://tools.ietf.org/html/rfc1950 (zlib format), rfc1951 (deflate format) and rfc1952 (gzip format).

zlib compressed data are typically written with a gzip or a zlib wrapper. The wrapper encapsulates the raw [DEFLATE](https://en.wikipedia.org/wiki/DEFLATE) data by adding a header and trailer. This provides stream identification and error detection that are not provided by the raw DEFLATE data.

1. **HDF4 [**[**Ref**](https://www.hdfgroup.org/solutions/hdf4/)**]**

***wget -q*** [***https://support.hdfgroup.org/ftp/HDF/HDF\_Current/src/hdf-${v}.tar.gz***](https://support.hdfgroup.org/ftp/HDF/HDF_Current/src/hdf-$%7bv%7d.tar.gz)

HDF4® is a library and multi-object file format for storing and managing data between machines. HDF4 is a very different technology than [HDF5](https://www.hdfgroup.org/hdf5/). The HDF Group recommends that, other than for working with existing HDF4 data, new applications use HDF5 since HDF5 addresses important deficiencies of HDF4.

If you’re interested in converting your data from HDF4 to HDF5, our engineers can help. Learn more about our [consulting service](https://www.hdfgroup.org/solutions/consulting/) or free and open source [h4toh5 conversion tools](https://support.hdfgroup.org/products/hdf5_tools/h4toh5/).

NASA maintains a network of [Earth Observing Satellites](https://www.hdfgroup.org/eos_vignette/) that transmit huge volumes of new imagery data daily in HDF4.  As a result, HDF4 is used extensively in the Earth Sciences. The HDF-EOS Website, [http://www.hdfeos.org](http://www.hdfeos.org/) has a wealth of information on these uses.

NOTE: We are using both HDF4 and HDF5 (see below)

1. **HDF5 [**[**Ref**](https://support.hdfgroup.org/HDF5/Tutor/HDF5Intro.pdf)**]**

***wget -q ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4/hdf5-${v}.tar.gz***

Hierarchical Data Format 5 (HDF5) is a unique open source technology suite for managing data collections of all sizes and complexity. HDF5 was specifically designed: • For high volume and/or complex data (but can be used for low volume/simple data) • For every size and type of system (portable) • For flexible, efficient storage and I/O • To enable applications to evolve in their use of HDF5 and to accommodate new models • To be used as a file format tool kit (many formats use HDF5 under the hood)

1. **NetCDF [**[**Ref**](https://www.unidata.ucar.edu/downloads/netcdf/index.jsp)**]**

***wget -q*** [*http://www.unidata.ucar.edu/downloads/netcdf/ftp/netcdf-${v}.tar.gz*](http://www.unidata.ucar.edu/downloads/netcdf/ftp/netcdf-$%7bv%7d.tar.gz)

NetCDF (network Common Data Form) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. Distributions are provided for Java and C/C++/Fortran. See the [netCDF web site](https://www.unidata.ucar.edu/software/netcdf) and the FAQ answer to [How do I get the netCDF software package?](https://www.unidata.ucar.edu/software/netcdf/docs/faq.html#howtoget) for more information.

1. **XML C parser [**[**Ref**](http://xmlsoft.org/)**]**

***wget -q*** [*ftp://xmlsoft.org/libxml2/libxml2-${v}.tar.gz*](ftp://xmlsoft.org/libxml2/libxml2-$%7bv%7d.tar.gz)

Libxml2 is the XML C parser and toolkit developed for the Gnome project (but usable outside of the Gnome platform), it is free software available under the [MIT License](http://www.opensource.org/licenses/mit-license.html). XML itself is a metalanguage to design markup languages, i.e. text language where semantic and structure are added to the content using extra "markup" information enclosed between angle brackets. HTML is the most well-known markup language. Though the library is written in C [a variety of language bindings](http://xmlsoft.org/python.html) make it available in other environments.

1. **JSON-C - A JSON implementation in C [**[**Ref**](https://github.com/json-c/json-c/blob/master/README.md)**]**

***wget -q*** [*https://s3.amazonaws.com/json-c\_releases/releases/json-c-${v}.tar.gz*](https://s3.amazonaws.com/json-c_releases/releases/json-c-$%7bv%7d.tar.gz)

JSON-C implements a reference counting object model that allows you to easily construct JSON objects in C, output them as JSON formatted strings and parse JSON formatted strings back into the C representation of JSON objects. It aims to conform to [RFC 7159](https://tools.ietf.org/html/rfc7159).

1. **GDAL [**[**Ref**](https://en.wikipedia.org/wiki/GDAL)**]**

***wget -q*** [*http://download.osgeo.org/gdal/${v}/gdal-${v}.tar.gz*](http://download.osgeo.org/gdal/$%7bv%7d/gdal-$%7bv%7d.tar.gz)

The Geospatial Data Abstraction Library (GDAL) is a computer software [library](https://en.wikipedia.org/wiki/Library_(computer_science)) for reading and writing [raster](https://en.wikipedia.org/wiki/Raster_graphics) and [vector](https://en.wikipedia.org/wiki/Vector_graphics) [geospatial data formats](https://en.wikipedia.org/wiki/GIS_file_formats), and is released under the [permissive](https://en.wikipedia.org/wiki/Permissive_free_software_licence) [X/MIT style](https://en.wikipedia.org/wiki/MIT_License) free software license by the [Open Source Geospatial Foundation](https://en.wikipedia.org/wiki/Open_Source_Geospatial_Foundation). As a library, it presents a single [abstract](https://en.wikipedia.org/wiki/Abstraction_(computer_science)) [data model](https://en.wikipedia.org/wiki/Data_model) to the calling application for all supported formats. It may also be built with a variety of useful [command line interface](https://en.wikipedia.org/wiki/Command_line_interface) [utilities](https://en.wikipedia.org/wiki/Utility_program) for data translation and processing. Projections and transformations are supported by the [PROJ.4](https://en.wikipedia.org/wiki/PROJ.4) library.

1. **Install PostGreSQL**

PostGreSQL and PostGIS (see below) must be installed AFTER the GDAL, and via source compilation. Though these can be done with ‘yum install’, they will not link correctly with GDAL.

1. **Install PostGIS**

PostGIS requires GEOS and GDAL (see below for all dependencies). Hence it must be installed after GEOS, GDAL and PostGreSQL.

-------------- Dependencies --------------

GEOS config: /usr/bin/geos-config

GEOS version: 3.6.2

GDAL config: /usr/local/bin/gdal-config

GDAL version: 2.3.1

PostgreSQL config: /usr/local/pgsql/bin/pg\_config

PostgreSQL version: PostgreSQL 11.0

PROJ4 version: 51

Libxml2 config: /usr/bin/xml2-config

Libxml2 version: 2.9.8

JSON-C support: yes

protobuf-c support: no

PCRE support: no

Perl: /usr/bin/perl

-------------------------------------------------

NOTE: Due to some possible quirk with the VM there are some additional dependencies for it. These are taken care of in the ‘build\_all.sh’, but not so elegantly. A way to automate a failsafe method must be worked out.

Specifically, compilation of executables, ‘pgsql2shp’ and ‘shp2pgsql’ are crashing due to some references in GEOS, ‘GEOSVoronoiDiagram’ and ‘GEOSClipByRect’, missing. It is unclear why ‘libtool’ does not find these. As a workaround, the compilation steps for these are to be disabled and, then, copy pre-compiled versions of these from another VM. This, however, may result in errors in future if there are version dependencies.

1. **Install GO**

*wget -q -O go.tar.gz https://dl.google.com/go/go1.10.3.linux-amd64.tar.gz*

The ‘go’ executable will be installed in /local/gsky/go/bin. Add it to your PATH to run from command line.

$ export PATH=$PATH: /local/gsky/go/bin

$ which go

/local/gsky/go/bin/go

1. **Build GSKY**

The GSKY source files are in the Github repo, ‘github.com/nci/gsky’. They are first cloned into /local/gsky/gopath/src/github.com before building the GSKY binary.

1. **Install all binaries**

This step involves just copying the right files into…

* /local/gsky/bin/api\*
* /local/gsky/share/gsky/gsky\*
* /local/gsky/share/gsky/grpc\_server\*
* /local/gsky/share/gsky/gsky-gdal-process\*
* /local/gsky/share/gsky/gsky-crawl\*
* /local/gsky/share/mas

**-- END OF SECTION --**

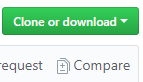
# Reverse pull GitHub repository

The forked repo (e.g. asivapra/gsk << nci/gsky) can go out of sync with its original. In such cases, a ‘reverse pull’ will sync it.

* + Open your repo.



* + Click on **Compare** (below the “Clone or Download” button)



* + Change the **base fork** to your repository

to 

* + Click on “**Compare across forks**”



* + Change **head fork** to the original repo

to 

* + Create a **pull request**



* + Click **accept**.
* The pull request is to your repo and hence you can approve it.

Now you will have all changes happened in the original repo, while retaining your changes in your forked repo.

**-- END OF THE DOCUMENT --**