GIS.lab Documentation

Release 0.8

GIS.lab team

Sep 19, 2019
| 1 | Let’s get started | 3 |
| 2 | Learning about the pieces | 5 |
| 3 | GIS.lab installation | 11 |
| 4 | GIS.lab administration | 33 |
| 5 | Client Desktop layout | 35 |
| 6 | GIS project publication | 41 |
| 7 | GIS.lab in practice | 47 |
| 8 | Useful terms | 59 |
| 9 | Useful commands | 63 |
| 10 | Conventions used in this documentation | 65 |
| 11 | Communication channels | 71 |
| 12 | Credits | 73 |
Let's get started

You may be coming to GIS.lab from various operating systems, but we expect that you are probably familiar with browsing the web, writing documents, sending emails and doing other job related tasks with your computer. Whether you are somewhat experienced or completely new to computers, this documentation should be useful for you. Of course, the more experience you have, the more quickly you will learn GIS.lab basics.

In general, this documentation does not assume any computer experience on your part. We try to explain in detail GIS.lab Desktop, and its integration with Gisquick platform. We even tell you how to install and how to use every of these pieces in the most efficient way.

Firstly, there is a section related to introduction called Learning about the pieces. In case you feel sufficiently familiar with all the introductory parts, just skip them and go straight away to GIS.lab Installation. Section Client Desktop layout relates to GIS.lab client environment with GIS and other applications. To get to know about QGIS projects publishing on web, see Gisquick part. Chapters dedicated to provide practical skills with GIS.lab technology are GIS.lab in practice and GIS.lab workshop. Another useful parts can be Useful terms or Useful commands. To distinguish between different kinds of information we use some typographical and stylistic features, see chapter Conventions used in this documentation.

Moreover, when you meet some troubles or difficulties, you can find solutions to simple and difficult problems in Tips and tricks part.
Fig. 1.2: GIS.lab Desktop screen.
CHAPTER 2

Learning about the pieces

2.1 What is GIS.lab?

GIS.lab is a tool for simple, unbreakable, always ready and clean deployment of multiple machines running software mainly for Geographic Information Systems (GIS) as well as for office.

It is free technology which can instantly turn any computer network into fully equipped geospatial cluster. It is capable to deploy a complete, centrally managed and horizontally scalable geospatial infrastructure in local area network, data center or cloud in a few moments. It is provided as comprehensive set of free geospatial software seamlessly integrated into one, easy-to-use system with out-of-box running services capable to run with minimal maintenance requirements for desktop interface. Together with integrated Gisquick publishing platform also web and mobile client interfaces can be available.

GIS.lab lowers deployment and ownership cost of complex geospatial solution to absolute minimum, while still keeping whole technology in house and under full control.

GIS.lab Desktop is traditional, customized, low resources environment with office and geospatial software. This technology promotes combination of desktop performance with web accessibility.

There are two different ways of launching GIS.lab Desktop. Depending on future usage, hardware and software possibilities and other similar factors, GIS.lab Desktop can be launched in virtual or physical mode.

Virtual mode can be used for any operating system. Original operating system and GIS.lab will be available. On the other hand, physical mode represents best performance but original operating system is temporary not available.

Fig. 2.1: Schema representing what GIS.lab is all about.
2.2 What can be GIS.lab ideal for?

This solution offers very large number of possible deployment scenarios.

GIS.lab can be used in places and conditions where deployment of any other technology would not be affordable or technically possible. GIS.lab is capable to turn bunch of heterogeneous or broken computers into crisis management command center, flawlessly working in very hard conditions of natural disaster with power, internet outages and no dependencies.

It can be used at schools as central management with maintenance-free clients. Students can gain various knowledge from Linux systems principles to GIS development.

GIS.lab can be useful for science institutions which can effectively use horizontally scalable computing power, advanced tools and extensibility.

Next, small projects or poor countries can appreciate affordable, complete solution, low system requirements and already mentioned maintenance-free clients. GIS.lab can save huge amount of obtaining and maintenance costs.

Or in any case, it can serve as ideal system just for Open Source technologies popularization.

2.3 Fundamental aims

One of the main objectives is rapid deployment of complete geospatial solution for collaborative data capturing, processing, analysis and publication with fully automatic provisioning and out of box working deployment using GIS.lab unit.

It is self containing system with very quick results, possible to get geospatial computation cluster from plain hardware. GIS.lab brings high added values by integration of precisely chosen set of geospatial FOSS, i.e. one best tool for one thing to one system with consistent behaviour, many collaboration tools, user and software support.

Full client computer performance utilisation represents opposite to thin client. It provides real desktop experience without any thin-client glitches. Rather than a traditional desktop, GIS.lab provides thinking about client environment more as about some kind of specialized client interface providing tools from desktop world. It allows computer resources sharing, same deployment in LAN and data center or cloud.

2.4 How it is done?

GIS.lab consists from one machine playing a role of master node (server, see GIS.lab unit section) running Linux operating system and any number of client computers with nearly no requirements. No operating system, even no hard disk is required but no problem if they exist. All these computers must be connected with gigabit network cables and switch.
Automatic provisioning is enabled by **Ansible** characterised by human-readable automation language, self-documenting syntax, agent-less execution, independent modules, templates, support for cloud providers AWS, GCE, Digital Ocean, Azure, and the like.

Virtual machine deployment is enabled by **Vagrant** and **VirtualBox**. Client boot service is **Linux Terminal Server** fat client or own solution. GIS.lab provides out of the box OGC Open Web Service (OWS) services load balancing, **QGIS Desktop** and Server as GIS.lab Desktop, own GIS software packaging and **GRASS GIS** as processing backend under QGIS Processing plugin and WPS.

Magic command is called `vagrant up` and orders to automatically provision GIS.lab server inside of virtual machine on the host computer, see *GIS.lab installation* section for details.

![Fig. 2.3: Key softwares for GIS.lab provisioning.](image)

### 2.5 What are the key benefits?

The main advantages are **fully automatic installation** or immediate deployment with GIS.lab unit appliance, **central management** of all machines, user accounts and backups high performance, maintenance free **desktop** interfaces. Additionally thanks to **Gisquick** integration also **web** and **mobile** client interfaces are available.

Web administration interface computing resources sharing across all machines, well known free software with high added value of seamless integration and bug fixes are some of other main qualities.

Useful benefits are also central software distribution, easy customization, automatic clustering and computing power sharing, no dependencies, etc.

### 2.6 What about software requirements?

At a minimum, your PC needs some necessaries. For this kind of facts, see hardware and software requirements for **virtual** or **physical** mode.

### 2.7 What is GIS.lab unit?

GIS.lab unit appliance is a hardware solution containing installation of GIS.lab system which is ready for immediate plug-and-play deployment and user friendly management.

With a view to further specifying the hardware, GIS.lab unit can be presented as **Intel Haswell**, **16 GB RAM**, **SSD**, tested with **20 clients**, portable, pocket size (11 x 11 x 4 cm) device with automatic host network adaptation.

![Fig. 2.4: GIS.lab unit.](image)

### 2.8 GIS.lab cluster

GIS.lab cluster is managed by decentralized cluster management tool called **Serf** based on **GOSSIP** protocol. Serf is responsible for automatic joining and removing machines to and from GIS.lab cluster and OWS load balancer management and can serve as failure detection system. This automatic cluster orchestration system is also used as interface for running cluster **events** and **queries**.
2.8.1 Roles, events and queries

Machines belonging to GIS.lab cluster are divided into two roles:

1. server
2. client

All machines are capable of running different set of cluster events and queries depending on their role membership. Events and queries can be send from any machine which is a member of GIS.lab cluster using `gislab-cluster` command or programmatically using RPC mechanism. All machines in cluster will receive all events and queries and will decide to respond or not depending on existence of handler responsible for particular event or query.

The main difference between event and query is that while query is designed to send some query and receive response, the purpose of event is just to announce that something has happened or should happen without receiving any response. Response from query can be returned in two formats, text or JSON.

See also:

Public events and queries

2.9 What is Gisquick?

Gisquick is a separate project which is not directly related with GIS.lab. It is a web application built on top of modern technologies with fully responsive user interface optimized also for mobile devices. It stands on the shoulders of QGIS desktop and server components. The main purpose of Gisquick is publishing QGIS projects on web.
Integration of Gisquick into GIS.lab infrastructure brings a new modern web and mobile interfaces.

2.10 Is GIS.lab similar to OSGeo Live?

GIS.lab is automatic deployment of GIS infrastructure - more or less opposite to OSGeo Live.

OSGeo-Live is a self-contained bootable DVD, USB thumb drive or Virtual Machine based on Lubuntu that allows user to try a wide variety of free and open source geospatial software without installing anything, it is like a Linux distribution.

GIS.lab is a operative system that uses a server where it is installed and desktop clients can attach to it using a LAN. It is easily configurable and it contains by default some of the most used free and open source geospatial software.
GIS.lab version 0.8 runs on top of customized Ubuntu 18.04 Bionic release.

3.1 How to start

3.1.1 Installation of requirements

Git installation

By far the easiest way of getting Git installed and ready to use is by using default repositories. This is the fastest method, but the version may be older than the newest version. For GIS.lab version from official repositories should be normally sufficient.

The instructions below are valid for Debian/Ubuntu operating systems. At first, `apt` package management tools can be used to update local package index. Afterwards, Git can be downloaded and installed.

```
$ sudo apt install git
```
GIS.lab Documentation, Release 0.8

GIS.lab source code download

Following command will grab the most recent GIS.lab source code to user system.

```
$ git clone https://github.com/gislab-npo/gislab.git
```

Note: Git is not necessary but it is recommended. One can get GIS.lab source code also by downloading latest GIS.lab package from https://github.com/gislab-npo/gislab.git and unpacking it in working directory.

For production environment it’s recommended to grab recent stable version (source code) directly from GitHub.

Ansible installation

Ansible is an automation engine used by GIS.lab for fully automatized provisioning. Its installation can be performed by typing ordinary commands.

```
$ sudo apt install ansible
```

Attention: Since version 0.8 GIS.lab requires Ansible version >= 2.4.

Tip: Most recent version of Ansible software can be also installed also by PIP.

```
$ sudo pip install ansible
```

Or alternatively custom PPA can be used.

```
$ sudo apt install software-properties-common
$ sudo apt-add-repository ppa:ansible/ansible
$ sudo apt update
$ sudo apt install ansible
```

VirtualBox installation

Install Dynamic Kernel Module Support Framework and VirtualBox packages. These packages are needed only for installation in Virtual Mode.

```
$ sudo apt install dkms virtualbox virtualbox-qt
```

Vagrant installation

Installing vagrant package from default repositories should be normally sufficient. The latest version can be downloaded from vagrantup.com. Vagrant is required only for installation in Virtual Mode.
$ sudo apt install vagrant

Also Vagrant disksize plugin is required and must be installed.

$ vagrant plugin install vagrant-disksize

Tip: If plugin installation fails, try to install more recent version of Vagrant.

### 3.1.2 Configuration

It is recommended to set at least some basic configuration before GIS.lab installation is performed.

GIS.lab is designed to install and run out of box with default configuration. However, it is required to change at least default network configuration variable GISLAB_NETWORK, if GIS.lab’s default network range 192.168.50.0/24 already exists in LAN to prevent IP conflicts.

Default GIS.lab configuration file named all exists in system/group_vars directory located in GIS.lab source code, see Fig. 3.2. When user decides to adjust it, this file should not be modified directly. Instead a custom configuration file in system/host_vars directory should be created.

Tip: ![Find the system/group_vars/all file in GIS.lab source code tree and see its content to become acquainted with all possibilities of configuration settings. It is full of commented out information.](image)

For installation in Virtual mode it is recommended to create file named gislab_vagrant in system/host_vars directory for host specific GIS.lab configuration and put various changes there.

When Physical mode is used, file in system/host_vars directory should be named according to name of GIS.lab unit. This name is a part of Ansible inventory file content, script that Ansible uses to determine what to provide. All file names must always match unique host name specified in inventory file.

Fig. 3.2: File layout related to configuration.

File gislab_vagrant will be loaded automatically by Vagrant without need to manually create the Ansible inventory file.

Tip: ![See practical example of configuration file.](image)

See also:

Network configuration

### 3.2 Virtual Mode

This part describes installation of GIS.lab in virtual machine using Vagrant and VirtualBox. Only Linux and MAC OS X operating systems on host machine are supported. Installation process will NOT modify anything on your host machine. Every operation is done inside of virtual machine.
GIS.lab server contains its own DHCP server which is by default disabled. DHCP server is used in almost every local network for automatic network configuration. If there is no DHCP server in your network configured to support GIS.lab (it’s typical situation in Virtual mode), own DHCP server on GIS.lab server must be enabled otherwise GIS.lab clients will not able to boot. GIS.lab DHCP server can be controlled by `gislab-network` administrator command, see installation process for details. By default, access to GIS.lab’s DHCP server is restricted only for GIS.lab client machines (by list of their MAC addresses) and it is managed by `gislab-machines` administrator command. It is possible to switch access policy to accept all clients, but it is required to double check that no DHCP servers conflict will occur. Otherwise, serious network breakage may be done.

**Attention:** Please, never change policy to allow all clients when connecting to your corporate LAN! It’s also recommended in this case to use corporate DHCP server (and keep GIS.lab DHCP server disabled, see Network management section for details).

If DHCP server on GIS.lab is enabled than own network is created. By default it is in range 192.168.50.0/24. If this range already exists in LAN where GIS.lab is going to be deployed, it is required to change it using `GISLAB_NETWORK` configuration variable.

See Configuration section for more information.

**Important:** Without changing network configuration variable IP conflicts may occur.

To run GIS.lab in Virtual mode, there are some hardware and software requirements. Another important point is GIS.lab source code, see GIS.lab source code download. See information below with graphical representation in Fig. 3.3.

**Hardware**
- at least 4 GB RAM on host machine

**Software**
- host machine running Linux or MAC OSX
- Git, see Git installation
- Ansible 2.4 or higher, see Ansible installation
- VirtualBox 4.3 or higher, see VirtualBox installation
- Vagrant 1.9 or higher, see Vagrant installation

Fig. 3.3: Requirements for installation in virtual mode.

**Tip:** Check the version of software that are installed by typing

```bash
$ ansible --version
$ vboxmanage --version
$ vagrant --version
```
3.2.1 Master

GIS.lab installation takes from 30 minutes to few hours depending on your machine performance and Internet connection speed.

Run following command in source code directory to power on the Virtual Machine providing GIS.lab master (server).

```
$ vagrant up
```

Everytime `up` command is performed, Vagrantfile will be used for configuration of virtual machine. If the `up` command is run first time, it also run the `provision` command internally used to provision, i.e. install and configure a virtual machine.

The output should be as follows.

```
Bringing machine 'gislab_vagrant' up with 'virtualbox' provider...
  ==> gislab_vagrant: Importing base box 'xenial-canonical'...
  ==> gislab_vagrant: Matching MAC address for NAT networking...
  ==> gislab_vagrant: Setting the name of the VM: gislab-vagrant-xenial
  ==> gislab_vagrant: Clearing any previously set network interfaces...
  ==> gislab_vagrant: Available bridged network interfaces:
      1) wlan0
      2) eth0
      3) docker0
  ==> gislab_vagrant: When choosing an interface, it is usually the one that is
  ==> gislab_vagrant: being used to connect to the internet.
  ==> gislab_vagrant: Which interface should the network bridge to?

If host machine contains multiple network adapters, user is asked to choose one corresponding adapter. For example, in case of `eth0` connection, selection 2 should be chosen. Then the installation goes ahead.

Tip: Typically choose network interface which is NOT currently used for Internet connection.

```
  ==> gislab_vagrant: Preparing network interfaces based on configuration...
      gislab_vagrant: Adapter 1: nat
      gislab_vagrant: Adapter 2: bridged
  ==> gislab_vagrant: Forwarding ports...
      gislab_vagrant: 22 (guest) => 2222 (host) (adapter 1)
  ==> gislab_vagrant: Running 'pre-boot' VM customizations...
  ==> gislab_vagrant: Booting VM...
  ==> gislab_vagrant: Waiting for machine to boot. This may take a few minutes...
      gislab_vagrant: SSH address: 127.0.0.1:2222
      gislab_vagrant: SSH username: ubuntu
      gislab_vagrant: SSH auth method: password
  ==> gislab_vagrant: Machine booted and ready!
  ==> gislab_vagrant: Checking for guest additions in VM...
  ...  
  ==> gislab_vagrant: Guest Additions Version: 5.0.18_Ubuntu r106667
  ==> gislab_vagrant: VirtualBox Version: 5.1
  ==> gislab_vagrant: Configuring and enabling network interfaces...
  ==> gislab_vagrant: Running provisioner: install (ansible)...
  ==> gislab_vagrant: Running ansible-playbook...
  [WARNING]: Not prompting as we are not in interactive mode
```

(continues on next page)
User accounts

GIS.lab user accounts are created by administrator on demand. Due to LDAP integration the same login credentials are valid for logging in to GIS.lab client session and to all GIS.lab services as well.

By default, GIS.lab installation creates only a superuser account gislab. Ordinary user account can be created by logging in to GIS.lab server, i.e. running Vagrant machine in source code directory via SSH.

Any IP address, username or password are not needed to login via SSH. Just `vagrant ssh` command is enough. The only obligation is to be present in the folder where `Vagrantfile` is. That file contains all necessary information.

```bash
$ vagrant ssh
```

A new user account can be created by `gislab-adduser` administration command. See *Creating new user* section for details.

---

**Tip:** Demo user accounts can be easily created from host machine by running

```bash
./utils/demo-create-users.sh 1
```

where the argument is number of accounts to be created. The user account starts with `lab` basename. The first user account created by this script is `lab1`. All the created accounts have the same password: `lab`.

### 3.2.2 Client

Running GIS.lab client in virtual mode is very useful when one wants to keep working in his favourite operating system, e.g. Windows 7 OS but also wants to use GIS.lab environment. GIS.lab virtual client is running in VirtualBox virtual machine, which is capable to run on Windows, Linux or Mac OS X operating systems. The process consists of four main steps:

1. Virtual machine creation
2. Booting
3. Enabling GIS.lab client on GIS.lab server
4. Running virtual GIS.lab client
Virtual machine creation

Fig. 3.5: Initial settings of Virtual Client Machine.

Machines are created in VirtualBox environment and their creation depends on type of booting, see Fig. 3.6 and Fig. 3.7.

Booting

There are two options how to boot GIS.lab Desktop clients either using PXE or HTTP boot.

PXE boot

PXE boot is a default boot mode for GIS.lab clients. It is a simplest method to get client up and running, but it may not work if multiple DHCP boot servers or GIS.lab servers exists in network.

It is necessary to configure in System tab boot order to boot only from network and enable IO APIC. In Network tab configure network adapter in bridged mode, make sure that PCnet-FAST III (Am79C973) is selected as the adapter type and allow promiscuous mode for all, see Fig. 3.6.

Fig. 3.6: Settings in VirtualBox using PXE boot.

Following steps are same as for HTTP mentod, see next section for details.

HTTP boot

HTTP boot is an alternative boot method for launching GIS.lab Desktop clients, which offers some advanced features and allows to boot if multiple DHCP boot servers or GIS.lab servers exists in LAN. HTTP boot is performed by loading system from special GIS.lab bootloader ISO image file, which exists in http-boot/gislab-bootloader.iso. Here is a list of notable advantages of HTTP boot over PXE:

- it is the only way to boot if multiple DHCP boot servers or GIS.lab servers exists in network
- it allows to manually choose target GIS.lab server which is very handy if multiple GIS.lab servers are running in one network
- it is easier to boot from HTTP (which is actually done by booting from USB stick) than to setup PXE boot on some new machines
- boot process is faster
- it allows to use para-virtualized network adapter for Virtual clients (VirtualBox), which is many times faster than network adapter used for PXE

3.2. Virtual Mode
Using HTTP boot it is necessary to add virtual `gislab-bootloader.iso` file as virtual CD/DVD (Storage tab), configure boot order to boot only from virtual CD/DVD, enable IO APIC (System tab), and in Network tab to configure network adapter in bridged mode, make sure Paravirtualized Network (virtio-net) is selected as the adapter type and allow promiscuous mode for all, see Fig. 3.7.

**Important**: For next steps assigned MAC address is needed. See Network section in VirtualBox environment and make a note of this address.

Selection of the network adapter on the host system that traffic to and from which network card will go through should be different from current internet connection, e.g. in case of wifi connection, the `eno0` should be set as Name of Bridged Adapter, see Fig. 3.7.

After virtual client is created, log in to GIS.lab server by `vagrant ssh` and with `gislab-machines administra-`tion command allow client machine to connect, see Enabling GIS.lab client section for details.

**Important**: Since GIS.lab version 0.6 DHCP service is disabled by default. In order to boot virtual client DHCP service must be running. See Network management section for details.

**Running virtual GIS.lab client**

Start GIS.lab client virtual machine by pressing Start button in VirtualBox Manager, log in and enjoy.

**Note**: Make sure that GIS.lab master (server) is running.

```
$ vagrant status
Current machine states:
gislab_vagrant running (virtualbox)
```

Using HTTP boot there are two possible choices to choose from:

A) *Automatic GIS.lab detection*

B) *Manual GIS.lab selection.*
Automatic detection

This mode will run DHCP request to set initial network DNS server configuration. It will use the first response from any DHCP server in network. Then, it will try to boot from `http://boot.gis.lab`. It means, that if DHCP server response was from GIS.lab server, client machine will successfully launch. If that response was from some third-party DHCP server running in LAN, it will fail unless DNS server provided by that DHCP response will be aware of `boot.gis.lab`. It also means, that if multiple GIS.lab server instances are running in one LAN, it is not possible to predict which one will be used.

![Automatic detection using HTTP boot.](image1)

Fig. 3.9: Automatic detection using HTTP boot.

Manual selection

Manual GIS.lab server selection can be used to choose GIS.lab server by entering its IP address. It means, that it is not vulnerable from third-party DHCP responses and it is possible to choose particular GIS.lab server, if multiple ones are running in LAN. GIS.lab server is using multiple IP addresses, i.e. IP address from GIS.lab network range `GISLAB_NETWORK.5` or IP address assigned by LAN. Both of them can be used for choosing GIS.lab server to boot.

![Manual network selection using HTTP boot.](image2)

Fig. 3.10: Manual network selection using HTTP boot.

Tip: IP address can be found out after typing `ip a | grep eth0` on GIS.lab server after log in by `vagrant ssh` command.

In Fig. 3.11 and Fig. 3.12 one can see GIS.lab client logging screen and Desktop of running virtual GIS.lab client.

Tip: To set custom client display resolution run following command on host machine.
Fig. 3.11: GIS.lab client logging screen.

Fig. 3.12: GIS.lab client running environment.

```bash
$ VBoxManage controlvm "<GIS.lab client name>" setvideomodehint <xresolution> <yresolution> 32
# For example
$ VBoxManage controlvm "GIS.lab client PXE" setvideomodehint 1000 660 32
```

Note: Getting a list of all running VirtualBox virtual machines by name and UUID is possible with following command on host machine.

```bash
$ VBoxManage list runningvms
```

For logging out from GIS.lab server use `logout` and then use `vagrant halt` to shut down the running machine Vagrant is managing. It does not remove the Virtual Machine from the hard disk. Machine (GIS.lab master/server) can be started again by using `vagrant up` command.

Tip: Use `-f` or `-force` flag to forcefully power off the Virtual Machine.

Note: GIS.lab master virtual machine can be deleted by:

```bash
$ vagrant -f destroy
```
### 3.2.3 How to upgrade GIS.lab?

GIS.lab upgrade procedure consists from three steps:

1. server software upgrade
2. client images upgrade
3. GIS.lab system itself upgrade

Although, it is possible to run each step separately by hand, GIS.lab provisioner is designed as idempotent task which is capable of both, GIS.lab installation and also upgrade. This means, that GIS.lab upgrade is performed by the same provisioner command as used for GIS.lab installation. Using GIS.lab provisioner for upgrade is recommended to keep all parts of GIS.lab in consistent state.

In GIS.lab source code directory run:

```bash
$ git pull
```

And upgrade GIS.lab master (server) virtual machine with Vagrant:

```bash
$ vagrant provision
```

**Note:** Note that virtual machine must be running when performing provisioning.

### 3.3 Physical Mode

To run GIS.lab in physical mode, there are some hardware and software requirements. As well as using virtual mode, *GIS.lab source code* is needed. Information about requirements can be found below together with graphical figuration, see Fig. 3.13.

**Hardware**

- GIS.lab Unit machine
  - Intel NUC
  - at least 8GB RAM
  - SSD, 60-480GB
- at least 4 GB RAM on host machine
- networking accessories
  - 1 Gb Ethernet switch
  - ethernet cables

**Software**

- host machine running Linux or MAC OSX
- Git, see *Git installation*
- Ansible 2.4 or higher, see *Ansible installation*

Fig. 3.13: Requirements for installation in physical mode.
Important: In following procedure it is assumed that GIS.lab unit machine is going to be installed in network with automatic IP address assigning from DHCP server.

### 3.3.1 Master

The process of installation consists of three main steps:

1. Adjusted operating system installation
2. GIS.lab initialization
3. GIS.lab unit installation

**Adjusted operating system installation**

Following steps will guide user to install basic Ubuntu operating system on GIS.lab unit machine. Network is configured to automatically obtain IP address from DHCP server.

In the first step download latest 64-bit PC (AMD64) Server Install type of ISO image (ie. ubuntu-18.04. X-server-amd64.iso file).

Furthermore, it is important to create SSH keypair. Generated public part of keypair will be used as a way to identify trusted computers without involving passwords. It can be generated on host machine with ssh-keygen command. By default, the keypair will be saved in $HOME/.ssh directory. It is recommended to rename new key suitably, for example id_rsa_gislab_unit.

Then use script providers/gislab-unit/gislab-unit-iso.sh from GIS.lab source code directory to create custom GIS.lab unit installation ISO image file from original Ubuntu server ISO image file downloaded in above step. Adjusted image will be used for automatic installation of basic Ubuntu operating system on GIS.lab unit machine.

**Tip:** Run ./providers/gislab-unit/gislab-unit-iso.sh -h command to see details of required options. Options are written below.

```bash
USAGE: gislab-unit-iso.sh [OPTIONS]
Create GIS.lab base system installation ISO image from Ubuntu Server ISO.
Script must be executed with superuser privileges.

OPTIONS
-s country code used for choosing closest repository mirror (e.g. SK)
-t timezone (e.g. Europe/Bratislava)
-d disk size in GB (valid options: 60, 120, 240, 480; default: 60)
-a swap size in GB (default: 4)
-k SSH public key file, which will be used for GIS.lab installation or update
-w working directory with enough disk space (2.5 x larger than ISO image size)
-i Ubuntu Server installation ISO image file
-h display this help
```
For example, assuming that downloaded original Ubuntu server installation ISO image is located in Downloads directory, user wants to use Italian official archive mirror, Rome timezone, SSH public key file particularly created for GIS.lab installation is located in .ssh directory and new adjusted image should be saved in tmp directory, then the script can be run as follows.

```
sudo ./providers/gislab-unit/gislab-unit-iso.sh -s IT -t Europe/Rome \
-k ~/.ssh/id_rsa_gislab_unit.pub -w /tmp \
-i ~/Downloads/ubuntu-18.04.1-server-amd64.iso
```

**Important:** The gislab-unit-iso.sh script must be run with superuser (sudo) privileges. The script also assumes that 60GB SSD disk is available on GIS.lab unit machine. Different disk size can be prompted by –d option. Currently 60, 120, 240, and 480 GB SSD disks are supported.

![Fig. 3.14: Creation of custom GIS.lab unit installation ISO image.](image)

Continue with preparation of bootable installation USB stick from custom GIS.lab Unit ISO image file created in previous step. On Ubuntu Startup Disk Creator or UNetbootin applications can be used. Recommended procedure is based of dd command. See example below.

```
sudo mkdosfs -n 'GIS.lab Base System' -I /dev/sdf -F 32
isohybrid /path/to/your/gislab.iso
sudo dd if=/path/to/your/gislab.iso of=/dev/sdf bs=4M conv=fdatasync
```

**Important:** In example above is assumed that USB stick has been connected to host machine as /dev/sdf device. Your configuration can be different, please check connected devices by dmesg command! Note that formatting can be also done by gparted GUI application. For isohybrid command syslinux or syslinux-utils package must be installed.

When above process is done, together with ready USB stick attach also power supply, HDMI display, keyboard and Ethernet cable into GIS.lab unit machine, see Fig. 3.15. Power it on, press F10 key to run boot manager and select Boot from USB option. Then fully automatic installation should start. When finished, machine will be turned off. USB stick should then be removed.

![Fig. 3.15: Necessary hardware components in adjusted operating system installation process.](image)

**Note:** In installation process there is only one notification related to cached packages that allows to use Apt Cache server. Otherwise just Continue option should be selected.

As a next step, power on GIS.lab unit. In the case that monitor and keyboard is connected to the unit it is possible to log in to machine using username ubuntu and password ubuntu. It is also possible to log in to unit from host machine using SSH. That is why SSH key was generated.

**Important:** GIS.lab unit has to be registered in the network. In other words IP address has to be assigned to unit. Run ip a command on the unit to detect this address.
In case unit is not registered automatically, run DHCP client that apply for IP address. Then verify working internet connection, e.g. with ping command.

```bash
sudo dhclient eth0 -v
ping 8.8.8.8
```

**Tip:** To restart network use `sudo /etc/init.d/networking restart` command.

To log in on GIS.lab unit machine via SSH enter on host machine `ssh ubuntu@<ip addr>`

**Note:** Instead of IP address also assigned name of registered unit should work, for example `gislab.intra.ismaa.it`. This name can be found in output of `nslookup <ip address>` command.

```bash
$ ssh gislab@server.intra.ismaa.it -i ~/.ssh/id_rsa_gislab_unit.pub
```

## GIS.lab unit initialization

With regards to the recommended initialization, there are two important Ansible files, `<name-of-gislab-unit>.inventory` file and configuration file in `host_vars` directory which has already been stated in configuration section of this documentation.

Let's create Ansible inventory file. The name depends on unit's name which is the same as customization file in `host_vars` directory. The inventory file contains information about:

- name of GIS.lab unit
- IP address or hostname of unit
- the name of provisioning user able to log in to GIS.lab unit (always keep `ubuntu`)

```bash
<name-of-gislab-unit> ansible_ssh_host=<host-url> ansible_ssh_user=<provisioning-user-˓→account-name>
```

Content of Ansible inventory file called `gislab-unit-fem.inventory` could be as follows.

```bash
gislab-unit-fem ansible_ssh_host=10.234.1.44 ansible_ssh_user=ubuntu
```

In the next phase provisioning will be performed by `ansible-playbook` commands. For more detailed information about playbooks, see Ansible playbooks manual page. Example with above mentioned names and files is below.

```
ansible-playbook --inventory=gislab-unit-fem.inventory --private-key=~/.ssh/id_rsa_˓→gislab_unit providers/gislab-unit/gislab-unit.yml
```

**Fig. 3.16:** Placement of important file for initialization in GIS.lab source code layout.

The initialization of GIS.lab unit will be performed by following command run from host machine:
**Important:** It’s important to use private SSH key from the same keypair as used when creating customized ISO image for unit installation. In example above is assumed that the command is run from GIS.lab source code directory where is also placed previously created inventory file.

GIS.lab unit will reboot when finished.

**Tip:** See `gislab-unit.yml` context to be well aware of what this script is exactly performing.

**Note:** The initialization process depends on platform. Currently GIS.lab supports also AWS, see `providers` directory.

**GIS.lab unit installation**

Once GIS.lab is configured, installation can be performed. Run following command to execute another `ansible-playbook`. In this step all the work is made by `gislab.yml` file located in `system` directory.

```
$ ansible-playbook --inventory=gislab-unit-fem.inventory --private-key=~/.ssh/id_rsa_˓
→gislab_unit system/gislab.yml
```

Now, GIS.lab unit machine is installed with GIS.lab system. Do not forget to create user accounts by `gislab-adduser` command and allow client machines to connect by running `gislab-machines` command.

**3.3.2 Client**

GIS.lab machines are initialized from GIS.lab network using PXE or HTTP. This means always clean system, maintenance free with no HDD required using full hardware potential what make it opposite to thin client.

![GIS.lab machines launching](image)

Fig. 3.18: GIS.lab machines launching.

Physical client mode is preferred way of launching GIS.lab client, because it provides best performance. It will run GIS.lab client session on client machine instead of original operating system installed (if any) on hard drive. Original operating system and local data will stay **untouched** and will be ready to run again after GIS.lab client is shut down.

**3.3. Physical Mode**
To run physical client, it is required to connect machine running GIS.lab server and client machines via Gigabit switch and cables, CAT 5e or higher.

There is no reason to be afraid of loosing domestic operating system. GIS.lab client is capable to run even if you have Windows, Linux or MAC OSX installed on client machine.

Complete process of running GIS.lab client using physical mode, i.e. GIS.lab unit consists of three main steps.

1. **Booting**
2. **Enabling GIS.lab client on GIS.lab server**
3. **Running physical GIS.lab client**

![Diagram](image)

Fig. 3.19: Any computer can be GIS.lab client.

**Booting**

As well as in virtual mode it is possible to boot using using PXE or HTTP boot.

**Important:** Client machine must be enabled on master, see Enabling GIS.lab client section for details.

**PXE boot**

PXE is a method of having a client boot using only its network card. Using this method of booting it is possible to circumvent the normal boot procedure, what means booting from CD/DVD/CD-RW Drive to Network Interface Card, usually known as NIC.

PXE boot is a default boot mode for GIS.lab clients. Booting from PXE requires to instruct client machine to boot from other device as it is usually doing so. On newer computers it is also required to disable Secure boot and/or enable Legacy mode.

**Important:** It is necessary to enabling NIC in BIOS.

The way how to enabling NIC is going into BIOS and look for it. It depends on machine. BIOS boot order can be changed for one time using F9 or F12 key, for permanent setup from BIOS configuration using DEL, F2 or F12, but it can differ from one to another machine brand.

It is recommended to look for Preferal devices, System Configuration, Integrated Devices or something similar and find NIC card there. When it is found, enabled and then back out, save and reboot should be selected.
In general, there are multiple possibilities how to instruct client machine to boot from PXE. See potential instructions below.

A. Depending on vendor, pressing some F key at machine start will temporary instruct machine to boot from PXE.
B. Depending on vendor, pressing some F key at machine starts to launch boot manager and enables to choose PXE or PCI LAN in boot menu to boot from PXE.
C. PXE or LAN option set as first boot device in BIOS configuration enable to boot from PXE after machine restart.

See also:
See procedure of enabling PXE boot for Lenovo or Dell machine in GIS.lab in practice section.
For more information about how it works see for example PXE Boot Server Installation Steps in Ubuntu Server VM.

HTTP boot

In addition to default PXE boot method, GIS.lab clients can boot over HTTP, which can provide some advantages.

To enable HTTP boot, it is needed to create bootable USB stick from special ISO image which exists in http-boot directory. Recipe is as follows.

Insert free USB stick into Linux workstation machine. If it is automatically mounted, unmount it. Run `dmesg` command to detect device assigned to USB stick by operating system.

**Note:** It should be something like `/dev/sd[x]`.

Burn GIS.lab Desktop bootloader into USB stick with command below. Be careful to choose correct output device without a partition number.

```
$ sudo dd if=http-boot/gislab-bootloader.iso of=/dev/sd[x]
```

Insert prepared USB stick into client machine and instruct it to boot from it.

**Running physical GIS.lab client**

After successful booting, there will be welcome screen with login dialog, see figure Fig. 3.20. Creation of user accounts and running GIS.lab clients are the same as in virtual mode. Find more details in User accounts and Running virtual GIS.lab client sections.

![Fig. 3.20: GIS.lab client logging in.](image)

Enjoy!
3.3.3 How to upgrade GIS.lab Desktop?

GIS.lab upgrade procedure consists from three steps described in virtual mode section. Only difference is command used for upgrade, Ansible is used instead of Vagrant.

GIS.lab source code update:

```bash
$ git pull
```

Upgrade with Ansible:

```bash
$ ansible-playbook --inventory=gislab-unit.inventory --private-key=<private-SSH-key-file> system/gislab.yml
```

3.4 Customization

When talking about customization, we should distinguish between:

1. **Server customization**
2. **User accounts customization**
3. **Client customization**

3.4.1 Server customization

GIS.lab Server (master node) can be customized by running standard Linux/Ubuntu commands. See GISMentors training group customization for example of customization performed by specific Ansible Playbooks.

But it is recommended to use some isolated environment like LXC or Docker containers when deploying custom service. See Gisquick integration section for example of such customization.

See also:

Understanding the key differences between LXC and Docker

3.4.2 User accounts customization

Process of creation and removal of GIS.lab user accounts can be customized by special scripts.
Important: Scripts must have executable permissions assigned and can’t contain file extension, see \texttt{man run-parts}.

Directory /opt/gislab/custom/accounts in Fig. 3.22 (located on GIS.lab master node) contains following directories with customization scripts.

- \texttt{before-add} - executed before account is created
- \texttt{after-add} - executed after account is deleted
- \texttt{before-delete} - executed before account is deleted
- \texttt{after-delete} - executed after account is deleted
- \texttt{files} - content of this directory is copied to user’s home directory before \texttt{after-add} hooks are executed

Fig. 3.22: File layout related to customization.

In above customization scripts it is possible to use various variables. When creating or deleting GIS.lab user account using \texttt{gislab-adduser} and \texttt{gislab-deluser} administration commands, following variables can be used.

- \texttt{GISLAB_USER} - user name
- \texttt{GISLAB_USER_GIVEN_NAME} - first name
- \texttt{GISLAB_USER_SURNAME} - last name
- \texttt{GISLAB_USER_EMAIL} - email
- \texttt{GISLAB_USER_DESCRIPTION} - description
- \texttt{GISLAB_USER_SUPERUSER} - superuser status
- \texttt{GISLAB_USER_GROUPS} - groups membership

For content stored in \texttt{files} directory, it is possible to use template variables in following format.

- \texttt{gislab-adduser} -
  - {+ \texttt{GISLAB_USER} +} - user name
  - {+ \texttt{GISLAB_USER_GIVEN_NAME} +} - first name
  - {+ \texttt{GISLAB_USER_SURNAME} +} - last name
  - {+ \texttt{GISLAB_USER_EMAIL} +} - email
  - {+ \texttt{GISLAB_USER_DESCRIPTION} +} - description
  - {+ \texttt{GISLAB_USER_SUPERUSER} +} - superuser status
  - {+ \texttt{GISLAB_USER_GROUPS} +} - groups membership

Example customization script \texttt{db} in \texttt{after-add} directory for automatic database schema creation in \texttt{schemaname} after new GIS.lab user is added is shown below.

```bash
#!/bin/sh
create_schema() {   
    if ! -z `sudo -u postgres psql -lqt | cut -d | -f 1 | grep -w $1` ; then   
        echo "CREATE SCHEMA $GISLAB_USER;"   
        GRANT USAGE on SCHEMA $GISLAB_USER to $GISLAB_USER;   
    fi   
}
```

(continues on next page)
3.4.3 Client customization

GIS.lab desktop client can be customized by running standard Linux/Ubuntu commands as well as *GIS.lab server*. Difference is that they must be executed in isolated environment called *chroot*. Administrator commands `gislab-client-shell` and `gislab-client-image` are used to perform this action.

Fundamental is an *image file*. It is a binary file with the `.img` filename extension and represents a snapshot GIS.lab client operating system. More specifically, it is compressed client’s *root*. GIS.lab clients boots from this image. The first of above mentioned commands enables to enter chroot environment. The second command creates a new updated image file.

**Danger:** ⚠️ Client’s *root* and resulting *image* are always restored to original state after every GIS.lab upgrade, so customization must be applied again.

**Note:** 📝 This behaviour is planed to be changed in future.

Important danger note written above is precisely why *backup* should always be used. In general, it is very good idea to backup client’s *root* and also *image* in case if something will go wrong in process of customization or rollback is required. Approximate total backup size is about 3 GB.

Backup of client’s *root* directory can be created by following statement using tape archive command. Command for client’s *image* backup is introduced below.

```
$ sudo tar cjf /mnt/backup/<root>.tar.bz2 /opt/gislab/system/clients/desktop/root
$ sudo cp -a /opt/gislab/system/clients/desktop/image /mnt/backup/<image>
```

See Fig. 3.23 for clearer understanding.

**Fig. 3.23:** Recommended backup of client’s files.

**Note:** 📝 Backup of client’s *image* file is not necessary because it can always be created by `gislab-client-image` command from particular GIS.lab client’s *root*. Why also this backup is useful will be introduced later.

When the recommended backups are created, it is time to start with customization.

**See also:**
See practical example of custom installation of latest GDAL version on GIS.lab client from source code.

Backups of client's root and image is possible to recover if needed. First, current client’s root and image should be removed and afterwards, selected backup can be recovered, see commands below.

```
$ sudo rm -r /opt/gislab/system/clients/desktop/root
$ sudo rm -r /opt/gislab/system/clients/desktop/image
$ sudo tar xjf /mnt/backup/<root>.tar.bz2 -C / 
$ sudo cp -a /mnt/backup/<image>/ /opt/gislab/system/clients/desktop/image
```

Using symbolic links

Violet diagram with blue line color called image in Fig. 3.23 containing gislab.img binary file and gislab-desktop.buildstamp in /opt/gislab/system/clients/desktop directory can be a directory or a symbolic link. By default it is a directory.

But it is very smart and handy to have more versions of image and just switching between them by symlink with nickname image and refer to particular directory with gislab.img and gislab-desktop.buildstamp files.

Note: File gislab.img is a compressed client’s root directory.

Let’s say there are two different images for two different courses - beginner and advanced. They cause different customization of users.

When one wants to select desired image, following steps from /opt/gislab/system/clients/desktop directory should be used.

```
# get list of all saved versions of image
$ (cd /mnt/backup/; ls -la) 
root-advanced 
root-beginner 
root-advanced.tar.bz2 
root-beginner.tar.bz2 
...  
# remove current image and root
$ sudo rm -r image 
$ sudo rm -r root  
# switch to image for advanced course
$ sudo ln -s /mnt/backup/root-advanced image  
# extract corresponding root from backup
$ sudo tar xjf /mnt/backup/root-advanced.tar.bz2 -C /
```

Then continue with creation of new user prepared for advanced course. See principle in Fig. 3.24.

Fig. 3.24: Principle of using symlinks for effective customization.

Note: When new image is set up on master, the clients are notified about new image and rebooted automatically after logout.
It is recommended to use Ansible to execute customization scripts directly from local controlling machine. See *Executing customization scripts from Ansible* example.

**Important:** GIS.lab master has to run during customization.

### 3.4.4 Boot loader

To customize GIS.lab Desktop client boot loader (see *HTTP boot*), create copy of boot loader source file `http-boot/gislab-bootloader.ipxe` and modify it as required. For more information about iPXE syntax see documentation. Then follow build process below.

Firstly, download iPXE source code.

```bash
$ git clone git://git.ipxe.org/ipxe.git && cd ipxe
```

Optionally checkout to version used by GIS.lab by typing

```bash
$ git checkout d644ad41f5a17315ab72f6ebeeeecf895f7d41679
```

Finally build customized ISO image `bin/pxe.iso`

```bash
$ cd src
$ make EMBED=CUSTOM-BOOT-LOADER-SOURCE-FILE.ipxe
```
Chapter 4

GIS.lab administration

4.1 User accounts management

4.1.1 Creating new user

New user accounts can be created by using `gislab-adduser` command, the command below creates ordinary user `lab1` with `lab` as password.

```
$ sudo gislab-adduser -g User -l GIS.lab -m lab1@gis.lab -p lab lab1
```

**Note:** Superuser accounts can be created by `--superuser` flag, such user will be able to perform `sudo` operations on client machines.

**Tip:** See User accounts customization section to perform customization when creating or deleting user accounts.

4.1.2 List existing users

With `gislab-listusers` list of all GIS.lab users is displayed, see example below.

```
$ sudo gislab-listusers | grep uid:
uid: uid=gislab
uid: uid=lab1
```
4.2 Machines management

4.2.1 Enabling GIS.lab client

By default, no client machines are allowed to boot from GISlab server. To allow client machine, there are similar steps to steps described for virtual mode. Simply run gislab-machines command on GIS.lab server and enable the client.

```bash
sudo gislab-machines -a <MAC-address>
```

**Tip:** Good way to collect MAC addresses of client machines is to plainly let them try to boot and than run following command to get list of denied MAC addresses on GIS.lab server.

```bash
$ sudo grep -e 'DHCPDISCOVER.*no free leases' /var/log/syslog
```

4.3 Network management

GIS.lab network can operate in two modes. GIS.lab is possible to integrate into existing (corporate) local area network (LAN) or to run its own computer network controlled by DHCP server on GIS.lab master machine. By default (since GIS.lab version 0.6) DHCP service is **disabled** on master. GIS.lab network is managed by gislab-network administration command.

**Note:** This functionality has been added in GIS.lab version 0.6.

Current status is reported by

```bash
$ sudo gislab-network status
```

```bash
[GIS.lab]: Connection forwarding service **is disabled and inactive.**
[GIS.lab]: DNS service **is disabled and inactive.**
[GIS.lab]: DHCP service **is disabled and inactive.**
```

DHCP and DNS service can be started on master node by

```bash
$ sudo gislab-network start
```

This settings is not persistent, to enable DHCP/DNS service automatically after booting master run:

```bash
$ sudo gislab-network enable
```
CHAPTER 5

Client Desktop layout

Ordinary GIS.lab client environment is shown below. Desktop layout is created by main panel, background with GIS.lab logo and in bottom part one can see also some basic system information.

![Basic GIS.lab client Desktop layout](image)

Fig. 5.1: Basic GIS.lab client Desktop layout.

Main panel contains:

1. **applications launcher**  list of all available applications, settings and log out menu
2. **quick launcher**  home directory browser, launchers of the most frequently used applications
3. **virtual desktops**  virtual desktops switcher and overview
4. **running applications**  running applications list and switcher
5. **keyboard layout**  available keyboard layouts switcher
6. **battery**  battery status
7. **chat**  messaging status
8. **sound**  sound control and network status
9. **time**  calendar and time information
5.1 GIS applications

GIS.lab comes with a bunch of preinstalled GIS applications and services. **QGIS** and **GRASS GIS** are two core components. **QGIS** is used for GIS project creation, data preparation, analysis and publication. **GRASS GIS** can be used for complex data analysis and manipulation. Due to preinstalled QGIS GRASS plugin GRASS tools can be accessed also using QGIS user interface.

![QGIS](image1.png)

Fig. 5.2: QGIS with preinstalled Natural Earth project.

![GRASS GIS](image2.png)

Fig. 5.3: Powerful GRASS GIS in GIS.lab client environment.

**Note:** Most of applications are taken from standard Ubuntu repositories for Bionic distribution. Up-to-date GIS software packages are provided by GIS.lab PPA.

Beside QGIS and GRASS on desktop client many other GIS tools are preinstalled. See list below including version numbers.

**Desktop apps**
- QGIS 2.18
- GRASS GIS 7.4
- SAGA 2.3

**Image processing**
- OTB 6.6
- Monteverdi GUI

**Geodatabase**
- PostGIS 2.5 import/export (shp2pgsql and psql2shp) command line tools including OSM-related osm2pgsql and osm2pgrouting utilities
• SpatiaLite 4.3 command line and GUI tools

Libraries
• GDAL 2.3 raster and vector tools
• Proj.4 5.2 command line utilities

GeoPython
• Fiona
• GDAL Python bindings
• PyModis
• OTB Python bindings
• OWSlib
• QGIS Python bindings
• Rasterio
• SAGA Python bindings
• Shapely
• Sentinelsat

OSM
• JOSM editor

5.2 Common applications

GIS.lab brings a lot of useful applications for basic work.

Accessories
• Calculator - calculator
• KeePassX - secure personal data management
• Leafpad - plain text editor
• Screenshot - screen shots creator

Graphics
• GIMP Image Editor - raster images editor
• Inkscape - vector drawings editor

Internet
• Firefox Web Browser - Internet browser
• Pidgin Internet Messenger - GIS.lab chat client

Note: IRC server - To allow effective communication between all GIS.lab users without a need of internet connection, GIS.lab provides built-in IRC server and prepared #gislab chat room.

Multimedia
Fig. 5.4: GIS.lab chat client.

- **VLC media player** - video and media player

**Office**

- **LibreOffice Calc** - spreadsheet editor
- **LibreOffice Impress** - presentations editor
- **LibreOffice Writer** - text editor

Fig. 5.5: Office applications in GIS.lab.

**System**

- **GIS.lab client informations** - GIS.lab client informations summary important for technical support

### 5.3 Directory layout

Directory layout consists of traditional folders as *Desktop, Documents, Downloads, and Templates*. GIS.lab adds *Project* folder dedicated for storing user GIS projects. There are also four shared directories:

- **Barrel** - read and write access for all - ideal for quick sharing data, files between clients
- **Booster** - mounted part of client machine RAM - ideal for quick reading and writing data
- **Publish** - shared folder dedicated for GIS projects publication, see QGIS Server and Gisquick publication section for details
- **Repository** - read-only access for normal users - ideal for sharing permanent data
5.4 Logout

User must log out to correctly finish work with GIS.lab client by using Log out menu available from applications launcher.

Note: Client machine shut down is available only from login screen.
GIS project publication

6.1 GIS project publication on Web

GIS projects are created and managed by QGIS application, which is a main tool for geospatial tasks in GIS.lab environment. QGIS is accessible under GIS.lab → GIS.lab Desktop item in GIS.lab applications menu.

Due to integrated Gisquick (http://gisquick.org) it is possible to publish GIS projects created in Desktop to Web environment just in few steps.

In following steps simplest possible GIS project will be created and published on Web.

1. Log in to GIS.lab desktop client session

Use login and password for user account that has been created by gislab-adduser command on GIS.lab master node (see Creating new user section for details).

2. Copy sample GIS project

Copy sample GIS project (Repository/gislab-project/natural-earth) to your Projects folder.

3. Open GIS project

Select central-europe.qgs project file and open by QGIS application.

Feel free to modify sample project based on your needs.

Now the first GIS project is ready.

4. Publish project on web

Open Gisquick plugin, and follow publishing wizard.

Now publish project by pressing Publish button. Publishing target is automatically set on GIS.lab to Publish/user directory. There is no need to change any settings or create zip file.

Launch Gisquick as GIS.lab → GIS.lab Web applications menu from main GIS.lab panel.

Log in with user’s credentials.

Then inspect published project which should be listed.
Fig. 6.1: Copy sample Natural Earth sample GIS.lab project.

Fig. 6.2: Paste sample project to Projects directory.

Fig. 6.3: Open sample GIS.lab project.
Fig. 6.4: Modify GIS.lab project for your needs.

Fig. 6.5: Launch Gisquick plugin to start publication process. Define basemaps, layers, and project metadata.

Fig. 6.6: Define topics.
Fig. 6.7: Check configuration summary.

Fig. 6.8: Publish project by simply pressing Publish button.

Fig. 6.9: Launch Gisquick from main panel.

Fig. 6.10: Log in by user’s credentials.

Fig. 6.11: Continue to enter Gisquick web environment.
Fig. 6.12: Click on project’s link in URL column to launch project in Gisquick.

Click on project’s link in URL column to launch project in web environment.

Fig. 6.13: Enjoy first published GIS.lab project in web environment.

For more investigation about Gisquick download more complex Prague sample project and study Gisquick documentation.

6.2 QGIS Server OWS publication

Todo: ...
CHAPTER 7

GIS.lab in practice

Now let’s see some practical examples.

7.1 GIS.lab configuration

7.1.1 Example of configuration file

This section shows example of configuration used for GIS.lab master provision, see related sections virtual and physical installation.

The name of file determines machine name for master. In the case of virtual mode, the name of file should be gislab_vagrant or other when more master virtual machines are provisioned. In physical mode, the name of file will be probably more generic <name-of-gislab-unit>, eg. gislab-my-organization. The file must be placed in system/host_vars directory located in GIS.lab code tree. See Configuration section for details.

Let’s see practical example of configuration with some changes related to GIS.lab network and client keyboards.

| GISLAB_ADMIN_FIRST_NAME: Ludmila |
| GISLAB_ADMIN_SURNAME: Furtkevicova |
| GISLAB_ADMIN_EMAIL: ludmilafurtkevicov@gmail.com |

| GISLAB_NETWORK: 192.168.50 |
| GISLAB_TIMEZONE: Europe/Rome |
| GISLAB_DNS_SERVERS: |
| - 10.234.10.10 |
| - 8.8.8.8 |

| GISLAB_CLIENT_ARCHITECTURE: amd64 |
| GISLAB_CLIENT_LANGUAGES: |
| - en |
| - sk |
| - it |
GISLAB_CLIENT_KEYBOARDS:
- layout: en
  variant: qwerty
- layout: sk
  variant: qwerty
- layout: it
  variant: qwerty
GISLAB_CLIENT_OWS_WORKER_MIN_MEMORY: 4000

Variables GISLAB_NETWORK and GISLAB_CLIENT_KEYBOARDS in gislab_vagrant file will be different. Results after the successful installation for both cases is demonstrated in Fig. 7.1.

<table>
<thead>
<tr>
<th>file gislab_vagrant 'A'</th>
<th>file gislab_vagrant 'B'</th>
</tr>
</thead>
<tbody>
<tr>
<td>GISLAB_NETWORK: 192.168.50</td>
<td>GISLAB_NETWORK: 192.168.30</td>
</tr>
<tr>
<td>GISLAB_CLIENT_KEYBOARDS:</td>
<td>GISLAB_CLIENT_KEYBOARDS:</td>
</tr>
<tr>
<td>- layout: sk</td>
<td>- layout: it</td>
</tr>
<tr>
<td>variant: qwerty</td>
<td>variant: qwerty</td>
</tr>
</tbody>
</table>

Fig. 7.1: Two different results using different Vagrant configuration file.

Fourth number of server’s IP address will always be 5. In our case client’s IP address terminates with 50. For left case of Fig. 7.1 these addresses would look like 192.168.50.5 and 192.168.50.50, for right case 192.168.30.5 and 192.168.30.50

Note: This information is useful in manual GIS.lab server selection using HTTP boot when server’s IP address is required.

### 7.2 Software integration

Software customization is performed on a server (master). At first log in locally using keyboard or via SSH protocol from controlling machine. In virtual mode use `vagrant ssh` command, see login to GIS.lab.

Before any modification it is recommended to **backup** current client’s root and image.

```bash
$ sudo tar cjf /mnt/backup/root-`date -I`.tar.bz2 /opt/gislab/system/clients/desktop/
  → root
$ sudo cp -a /opt/gislab/system/clients/desktop/image /mnt/backup/image-`date -I`
```
Commands in client root can be performed by `gislab-client-shell` administration command. To enter client's root in interactive mode `-i` must be given.

```bash
$ sudo gislab-client-shell -i
```

### 7.2.1 Software (un)installation

This section shows how to easily install or uninstall new software on GIS.lab clients.

Example with uninstalling Geany software is shown below.

```bash
# display geany package status details
$ dpkg -s geany
Status: install ok installed
...
# check geany version
$ geany --version
geany 1.27 (built on 2016-04-17 with GTK 2.24.30, GLib 2.48.0)
# uninstall geany
$ sudo apt remove geany
```

Another example demonstrates Vim editor installation process below.

```bash
$ dpkg -s vim
dpkg-query: package 'vim' is not installed and no information is available
...
$ sudo apt update
$ sudo apt install vim
```

When all desired changes are done the client’s root is exited by `exit` command.

```bash
$ exit
```

Then new client’s image must be generated from modified client’s root by `gislab-client-image` administration command.

```bash
$ sudo gislab-client-image
```

---

**Note:** Main panel in client Desktop layout is generated when user log in firstly, so changes related to panel are displayed only for new user.

### 7.2.2 Custom software compilation

Let’s see example custom installation of latest development GDAL version from source code.

At first interactive shell in GIS.lab client’s root must be entered.

```bash
$ sudo gislab-client-shell -i
```

Then compilation and installation of GDAL can be executed.
After client’s root is left by `exit` command, then image should be updated by `sudo gislab-client-image`. Continue with creation of new user booting with latest GDAL version.

Important: Do not forget to set `LD_LIBRARY_PATH` variable and configure dynamic linker run-time bindings on client before running GDAL commands.

```bash
$ export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
$ sudo ldconfig
$ /usr/local/bin/ogr2ogr --version
GDAL 2.3.0dev, released 2017/09/09
```

### 7.2.3 Gisquick Docker integration

Gisquick is integrated to GIS.lab environment as `Docker service`. Gisquick application is split into 3 services running in Docker containers:

- QGIS server
- Django Application served with Gunicorn
- Nginx Server

See more information in Gisquick documentation.

Gisquick docker files are located on GIS.lab master node in `/opt/gislab/system/docker/gisquick/` folder. GIS.lab comes with modified Django image with LDAP support and HTTP as default protocol.

Gisquick service can be managed from `/opt/gislab/system/docker/gisquick/` folder on GIS.lab master node by standard `docker-compose` command, see Gisquick documentation for details.

### 7.3 Tips and tricks

#### 7.3.1 Tips and tricks

**Apt Cacher service**

Apt Cacher service can be useful when installing GIS.lab several times in a row. Ubuntu software packages are cached by Apt Cacher service which is performed by Vagrant and running as virtual machine.

Vagrant file for Apt Cacher service:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
```
GISLAB_NETWORK="192.168.50"
VAGRANTFILE_API_VERSION = "2"
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "xenial-canonical"

  config.vm.provider "virtualbox" do |v|
    v.customize ["modifyvm", :id, "--memory", "512"]
    v.customize ["modifyvm", :id, "--nictype1", "virtio"]
    v.customize ["modifyvm", :id, "--nictype2", "virtio"]

    config.vm.network "forwarded_port", guest: 3142, host: 3142, auto_correct: true
  end

  config.vm.hostname = "apt-cacher"
  config.vm.provision "shell", inline: "apt install apt-cacher-ng"
  config.vm.network "public_network", ip: "%s.%s % [GISLAB_NETWORK, "6"]
end

Run Apt Cacher server by typing `vagrant up` and add following line to GIS.lab configuration file:

```
GISLAB_APT_HTTP_PROXY: http://192.168.50.6:3142
```

**Executing customization scripts from Ansible**

Following example will execute the same script first on GIS.lab Server and then in GIS.lab client’s root. See `gislab-customize.yml` Ansible playbook below.

```yaml
---

# Example GIS.lab customization playbook.
-
  hosts: all
  become: yes

  vars:
    SERVER_SCRIPT: gislab-customize.sh
    CLIENT_SCRIPT: gislab-customize.sh
    GISLAB_INSTALL_CLIENTS_ROOT: /opt/gislab/system/clients

  tasks:
    # Customize GIS.lab Server
    - name: Run script on server
      script: "{{ SERVER_SCRIPT }}"
      tags: customize-server

    # Customize GIS.lab Desktop client
    - name: Copy script to client's root
      copy:
        src: "{{ CLIENT_SCRIPT }}"
        dest: "/opt/gislab/system/clients/desktop/root/tmp/customize.sh"
        owner: root
        group: root
        mode: 0755
```

(continues on next page)
Example customization script `gislab-customize.sh` would be as follows.

```bash
#!/bin/bash
# Example GIS.lab customization script.
# Author: Ivan Mincik, ivan.mincik@gmail.com

# detect if we are running on GIS.lab Server or inside GIS.lab Desktop
# Client root
if [ "$(ls -di /)" == "2 /" ]; then
  echo "Hello from GIS.lab Server."
else
  echo "Hello from GIS.lab Client's root."
fi

# vim: set ts=4 sts=4 sw=4 noet:
```

Customization Ansible Playbook can be applied using `ansible-playbook` command from host/controlling machine. In `virtual mode` the command will be as follows:

```
PYTHONUNBUFFERED=1 \ 
ANSIBLE_FORCE_COLOR=true \ 
ANSIBLE_HOST_KEY_CHECKING=false \ 
ANSIBLE_SSH_ARGS='-o UserKnownHostsFile=/dev/null -o ForwardAgent=yes -o ControlMaster=auto -o ControlPersist=60s' \ 
ansible-playbook -v \ 
--private-key=$({pwd})/vagrant/machines/gislab_vagrant/virtualbox/private_key \ 
--user=vagrant \ 
--connection=ssh \ 
--limit='gislab_vagrant' \ 
--inventory-file=$({pwd})/vagrant/provisioners/ansible/inventory \ 
--tags customize-server,customize-client,build-client-image \ 
gislab-customize.yml
```
In **physical mode** the Ansible playbook will be applied similarly as core GIS.lab playbooks:

```
$ ansible-playbook --inventory=gislab-unit-fem.inventory --private-key=~/.ssh/id_rsa_gislab_unit gislab-customize.yml
```

### Running commands on whole cluster with parallel-ssh

Log in to the server and get list of currently running client machines. At first SSH key must be generated and authorized.

```
ssh-keygen
cat .ssh/id_rsa.pub >> /mnt/home/gislab/.ssh authorized_keys
```

**Note:** On virtual server the command must be run as `sudo`.

```
sudo sh -c 'cat .ssh/id_rsa.pub >> /mnt/home/gislab/.ssh/authorized_keys'
```

List of alive client machines can be determined by `gislab-cluster` administration command.

```
$ MACHINES="$(sudo gislab-cluster members -tag role=client -status=alive | awk -F " " '{printf "%s ", $1}')"
```

Install `gedit` on all running client machines

```
$ parallel-ssh -O StrictHostKeyChecking=no -i -H "$MACHINES" sudo DEBIAN_FRONTEND=noninteractive apt install -y --no-install-recommends gedit
```

### Todo: output

Perform performance test of parallel write to network share

```
$ parallel-ssh -O StrictHostKeyChecking=no -i -H "$MACHINES" 'dd if=/dev/zero of=/mnt/\n  barrel/file-$(hostname).io bs=1M count=1024'
```

Stderr: 1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 37.7824 s, 28.4 MB/s
Stderr: 1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 38.1136 s, 28.2 MB/s
Stderr: 1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 38.4403 s, 27.9 MB/s
Stderr: 1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 38.6802 s, 27.8 MB/s
```

Perform performance test of parallel read from network share

---

### 7.3. Tips and tricks

---

53
Perform CPU performance test

```bash
$ parallel-ssh -O StrictHostKeyChecking=no -i -H "$MACHINES" 'dd if=/dev/zero bs=1M count=1024 | md5sum'
```

Here is an example procedure of enabling PXE boot for Lenovo ThinkPad.

**Procedure of enabling PXE boot for Lenovo machine**

```bash
$ parallel-ssh -O StrictHostKeyChecking=no -i -H "$MACHINES" 'dd if=/mnt/barrel/file-$(hostname).io of=/dev/zero bs=1M'
```

```bash
Stderr: 1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 0.207453 s, 5.2 GB/s
Stderr: 1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 0.210259 s, 5.1 GB/s
Stderr: 1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 0.227793 s, 4.7 GB/s
Stderr: 1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 0.207774 s, 5.2 GB/s
```

```bash
Stderr: Warning: Permanently added 'c52,192.168.19.52' (ECDSA) to the list of known hosts.
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 2.51008 s, 428 MB/s
Stderr: Warning: Permanently added 'c53,192.168.19.53' (ECDSA) to the list of known hosts.
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 2.50255 s, 429 MB/s
Stderr: Warning: Permanently added 'c54,192.168.19.54' (ECDSA) to the list of known hosts.
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 2.50255 s, 429 MB/s
Stderr: Warning: Permanently added 'c51,192.168.19.51' (ECDSA) to the list of known hosts.
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 2.56706 s, 418 MB/s
```
Firstly, boot up computer. Press F2, then press Enter and F1 key. This should take you to the BIOS screen. Select Security, Secure Boot, set to Disable, select Start Up, UEFI/Legacy Boot, set to Legacy Only and press F10. Once you press F10, reboot and then press F12. You should now be at the boot menu. Select PCi LAN and press Enter.

Procedure of enabling PXE boot for Dell machine

Following examples shows enabling PXE boot for Dell Precision M4400 Mobile Workstation.

Start with machine booting. Press F12, go to BIOS Setup, find Settings, System configuration, Integrated NIC and set Enabled w/PXE. Then press Exit button, reboot and boot from Onboard NIC.

Public events and queries

Here is a list of publicly available events and queries designed for ordinary usage. This list does not contain system events and queries which are used for internal GIS.lab cluster management.

Get a list of cluster members of a Serf cluster by typing gislab-cluster members.

<table>
<thead>
<tr>
<th>server.gis.lab</th>
<th>192.168.50.5:7946</th>
<th>alive</th>
<th>role=server</th>
</tr>
</thead>
<tbody>
<tr>
<td>c51</td>
<td>192.168.50.51:7946</td>
<td>alive</td>
<td>role=client</td>
</tr>
</tbody>
</table>

Or get this list in JSON format with gislab-cluster members -format json command.

```json
{
    "members": [
        {
            "name": "server.gis.lab",
            "addr": "192.168.50.5:7946",
            "port": 7946,
            "tags": {
                "role": "server"
            },
            "status": "alive",
            "protocol": {
                "max": 4,
                "min": 2,
                "version": 4
            }
        },
        {
            "name": "c51",
            "addr": "192.168.50.51:7946",
            "port": 7946,
            "tags": {
                "role": "client"
            },
            "status": "alive",
            "protocol": {
                "max": 4,
                "min": 2,
                "version": 4
            }
        }
    ]
}
```
For more commands see *Useful commands* section with `<cluster>` key word. For example command `gislab-cluster members -tag session-active=*` lists client machines which are currently running user session. After GIS.lab user’s login there will be list as follows.

<table>
<thead>
<tr>
<th>Machine</th>
<th>IP Address</th>
<th>Status</th>
<th>Role</th>
<th>Session-active</th>
</tr>
</thead>
<tbody>
<tr>
<td>server.gis.lab</td>
<td>192.168.50.5:7946</td>
<td>alive</td>
<td>role=server</td>
<td></td>
</tr>
<tr>
<td>c51</td>
<td>192.168.50.51:7946</td>
<td>alive</td>
<td>role=client, session-active=ludka</td>
<td></td>
</tr>
</tbody>
</table>

See also:

Running commands on whole cluster with parallel-ssh

**Remote desktop management**

Connect to running remote desktop session using following command.

```
HOST=<REMOTE-HOST-NAME> ssh gislab@$HOST "x11vnc -bg -safer -once -nopw -scale 0.9x0.9 -display :0 -allow $(hostname -f)" && vncviewer $HOST
```

**Todo:** check

**Network configuration**

**Todo:** ?

This section tries to collect documentation to some of the most common network configurations used for GIS.lab deployment. We assume, that in all cases, machines are connected to Ethernet network with Gigabit switch and at least CAT5e Ethernet cables.

**Virtual Mode**

This part of documentation assumes that GIS.lab server is installed on Linux laptop in VirtualBox virtual machine using Vagrant as it is documented in *Virtual Mode* installation section.

**Existing LAN with DHCP server**

GIS.lab is deployed in existing LAN 192.168.1.0/24 which already contains DHCP server and many non GIS.lab machines and network is connected to Internet.

**Configuration**

- Laptop - wired adapter: automatic IP address assignment (Network Manager)
- Laptop - wireless adapter: disabled (Network Manager)
- GISLAB_NETWORK: 192.168.50
Separate network

GIS.lab is deployed in separate network, specially created by GIS.lab vendor, where only GIS.lab machines are connected. Internet access is provided by host laptop’s WiFi connection and it is connected to GIS.lab network via Ethernet cable. Network contains only GIS.lab machines.

**Configuration**

- Laptop - wired adapter: static IP address 192.168.5.1, mask 255.255.255.0, gateway 0.0.0.0, DNS 8.8.8.8 (Network Manager)
- Laptop - wireless adapter: connected to Internet (Network Manager)
- GISLAB_NETWORK: 192.168.50

Physical Mode

This section assumes that GIS.lab Unit machine is installed as it is documented in *Physical Mode* installation part.

Existing LAN with DHCP server

GIS.lab Unit is deployed in existing LAN 192.168.1.0/24 which already contains DHCP server and many non GIS.lab machines and network is connected to Internet.

**Configuration**

- GISLAB_NETWORK: 192.168.50

Separate network

GIS.lab Unit is deployed in separate network, specially created by GIS.lab vendor, where only GIS.lab machines are connected. Internet access is provided by laptop running Linux, which is connected to Internet via WiFi and to GIS.lab network via Ethernet cable. Network contains only GIS.lab machines.

In this case, it is required to change GIS.lab Unit’s wired network adapter configuration to static IP address and allow connection forwarding on laptop.

**Configuration**

- Laptop - wired adapter: static IP address 192.168.5.1, mask 255.255.255.0, gateway 0.0.0.0, DNS 8.8.8.8 (Network Manager)
- Laptop - wireless adapter: connected to Internet (Network Manager)
- GISLAB_NETWORK: 192.168.50
- GISLAB_SERVER_INTEGRATION_FALLBACK_IP_ADDRESS: 192.168.5.5
- GISLAB_SERVER_INTEGRATION_FALLBACK_GATEWAY: 192.168.5.1

To allow using laptop as Internet gateway, run following commands on laptop.

```bash
$ sudo sysctl -w net.ipv4.ip_forward=1
$ sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```
CHAPTER 8

Useful terms

AMD64 platform, the 64-bit version of the x86 instruction set, x86 means 32-bit OS

Ansible an easy to use configuration management and orchestration tool; it is platform that can combine multi-node software deployment and ad hoc task execution; it’s configuration, deployment, and orchestration language are playbooks expressed in YAML format; Ansible is an automation engine similar to Chef or Puppet, that can be used to ensure deployment and configuration consistency across many servers, and keep servers and applications up-to-date

Bridged networking connects a virtual machine to a network by using the network adapter on the host system, the virtual machine is a full participant in the network

![Bridged Networking Configuration](image)

Fig. 8.1: Bridged Networking Configuration

Client machine machine which runs GIS.lab client system launched from GIS.lab server

Client’s root a directory on GIS.lab server in which GIS.lab client is installed with a special tool named debootstrap; it has almost the same filesystem hierarchy as a standard Ubuntu desktop installation with GIS.lab integration changes - and this is what one can see in running GIS.lab client; from this directory is then created GIS.lab client image which is mounted over network as a client’s root partition during boot process

Configuration an arrangement of functional units, often pertains to the choice of hardware, software, firmware, and documentation; it affects system function and performance

---

1 Configuring Bridged Networking
Containers can share a single Linux Kernel and optionally other binary and library resources, compared with virtual machines that require a full operating system image for each instance.

Customization means doing some modification over an existing applications according to requirements of the client.

Dev development tools, software developer or device in PC.

DHCP server Dynamic Host Configuration Protocol is way how to set host’s IP address to automatically assign an IP address to any host on the network upon the request (in order for two hosts communicate on the same network using TCP/IP model, both hosts need to have an unique IP address); GIS.lab can run its own DHCP server.

Fig. 8.2: DHCP server

Django is a free and open-source web framework, written in Python, which follows the model–view–controller architectural pattern with primary goal to ease the creation of complex, database-driven websites, reuse various components emphasizing principle of don’t repeat yourself.

DNS server Domain Name System is a protocol within the set of standards for how computers exchange data on the internet and on many private networks, known as the TCP/IP protocol suite; its basic job is to turn a user-friendly domain name like qgis.org into an IP address like 91.142.249.5 that computers use to identify each other on the network; also it is used to don’t have to remember IP addresses, domain names are much easier to remember; user will need at least one DNS server address but can enter up to three addresses in case one server is unavailable; if user does not know it’s DNS server addresses, Google has DNS servers that anyone in the world can use for free, the addresses of these servers are Primary 8.8.8.8 and Secondary 8.8.4.4.

Fig. 8.3: Domain name system workflow

Gateway is an IP address of the device that machine looks to for access to the internet; usually, it is the router’s IP address.

GIS.lab node GIS.lab client environment which runs on client machine.

GIS.lab master GIS.lab server environment which runs on host machine; it acts as conventional server providing boot service, file, geo-database or chat server.

Git is a distributed version control system often used to work on some code together, it allows groups of people to work on the same files at the same time to develop various software; it runs command line on local machine and allows to keep track of files and modifications to those files in something called repository; every developer can work independently and then merge changes together while everything is recorded by Git.

GitHub is a web side that allows to upload Git repositories online; unlike Git, GitHub provides graphical interface for private repositories and free accounts usually used to host open-source projects.

---

2 DHCP server
3 How does DNS work?
GDAL  *Geospatial Data Abstraction Library* provides access to geospatial data, it is a cross platform C++ translator used for reading, writing and transforming raster (GDAL) and vector (OGR) data to the calling application for all supported formats; a variety of useful command line utilities for data translation and processing are used

**GUI**  *Graphical User Interface*

**Hardware virtualization** refers to the creation of a virtual machine that acts like a real computer with an operating system; softwares executed on these virtual machines are separated from the underlying hardware resources

**Host machine** is main machine where required softwares and GIS.lab package are installed

**Chroot** is a tool which enables installation in client environment; installation to client’s root is done within chroot operation

**I386, x86** ordinary platform, 32-bit microprocessor

**IA64** platform, 64-bit microprocessor

**IP address**  *Internet Protocol address* is unique numerical identifier assigned to each device participating in a computer network that uses the Internet Protocol for communication; it is 32-bit numeric address written as four numbers 0.0.0.0 separated by periods, each number can be zero to 255; it is assigned as part of connecting to a network; it is used for identifying computer on the network; when connecting using DHCP, this address will periodically change (hence, the name dynamic), in case of static IP address configuration, address will never change

**ISO** an archive file of an optical disc; image file is a snapshot of the data and layout of a CD or DVD, saved in ISO-9660 format widely used for storing CD content; it contains raw dumps of a magnetic disk or of an optical disc

**LAN**  *Local Area Network* is a computer network interconnecting computers within a limited area, e.g. school or office building

**LDAP**  *Lightweight Directory Access Protocol* is an open protocol that email and other programs use to look up information from a server, allows to access information directories and obtain these information

**LXC**  *Linux Container* is a lightweight virtualization technology on operating system level for running multiple isolated Linux systems, i.e. containers on a control host using a single Linux kernel

**MAC address**  *Media Access Control address* is kind of serial number assigned to every network adapter; no two anywhere should have the same MAC address; it is assigned at the time hardware is manufactured, the computer needs to have its MAC address registered in the DNS/DHCP so that it’ll be recognized and get the right IP address, six groups of two hexadecimal digits separated by colons 0:0:0:0:0:0 or by hyphens 0-0-0-0-0-0 or three groups of four hexadecimal digits separated by dots 0.0.0

**Network mask** tells computer the size of the network to which it is being connected, it is formatted the same way as the IP address, but usually looks something like 255.255.255.0

**NIC**  *Network Interface Card* is a computer hardware component that enable connecting computer to a computer network physically. It is a circuit board or card that is installed in a computer

**PXE booting**  *Preboot Execution Environment* booting, is a method of having an end computer (client) boot using only its network card, it is industry standard client/server interface that allows networked computers that are not yet loaded with an operating system to be configured and booted remotely by an administrator, PXE provides e.g. DHCP, application program interfaces that are used by the client’s Basic Input/Output Operating System (BIOS), etc.

**Serf**  *Service orchestration and management tool* is a decentralized solution for service discovery and orchestration; it uses an efficient and lightweight gossip protocol to communicate with other nodes, it can detect node failures and notify the rest of the cluster

---

4  *Preboot Execution Environment*
Software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software.

Tmux Terminal multiplexer is a software application usually used to multiplex several virtual consoles, enabling a number of terminals to be created, accessed, and controlled from a single screen; it may be detached from a screen and continue running in the background, then later reattached.

Travis CI a FOSS, hosted, distributed continuous integration service used to build and test software projects hosted at GitHub, it is configured by adding a YAML format text file to the root directory of the repository.

Vagrant a cross-platform tool for working with Virtual Machines, it does networking out-of-the-box and solves problems in a modular way, it represents the command line utility for managing the lifecycle of virtual machines, to describe the type of machine required for a project, and how to configure and provision these machines Vagrant file with Ruby syntax is used, process means set up - downloading and installing Vagrant, configure - creating a single file with description and work - running.

Vagrant box the package format for Vagrant environment, it is used to bring up an identical working environment.

Vendor supplier, a part of the supply chain.

See also:

19 Minutes With Ansible, Booting from the network with PXE, Vagrant in 5 minutes, How Domain Name Servers Work

Sources
CHAPTER 9

Useful commands

`gislab-adduser` creates GIS.lab user account
`gislab-backupall` backups all GIS.lab user accounts
`gislab-backupclient` backups GIS.lab client root and image
`gislab-backupuser` backups GIS.lab user account
`gislab-client-image` builds new GIS.lab client image
`gislab-client-shell` runs command or launches interactive shell in GIS.lab client’s root
`gislab-cluster event <event>` sends a custom event through the Serf cluster
`gislab-cluster event reboot` reboot all members of cluster
`gislab-cluster event reboot <hostname>` reboot particular client machines
`gislab-cluster event shutdown` shutdown all members of cluster
`gislab-cluster leave` gracefully leaves the Serf cluster and shuts down
`gislab-cluster members` lists the members of a serve cluster
`gislab-cluster members -tag sesion-active=*` lists client machines which are currently running user session
`gislab-deluser` removes GIS.lab user account
`gislab-listusers` lists GIS.lab users
`gislab-machines` adds or removes GIS.lab client machine’s MAC address
`gislab-password` changes GIS.lab user’s password
`gislab-restoreuser` restores GIS.lab user account from backup
`nslookup` displays information that can be used to diagnose DNS infrastructure, e.g. domain name or IP address, it is available only if TCP/IP protocol is installed
ping <hostname-or-ip-address-of-the-target-computer> sends a test packet of data to a designated IP address to test connection using the TCP/IP protocol, it finds out whether the peer host/gateway is reachable

shutdown -h now brings the system down; instructs the hardware to stop all CPU functions immediately

cmd: vagrant destroy stops the running Vagrant machine and destroys all resources that were created during the machine creation process

dest: vagrant halt shuts down the running machine Vagrant is managing

src: vagrant provision runs any configured provisioners that allow user to automatically install software, alter configurations, and more on the machine as part of the vagrant up process against the running Vagrant managed machine

cmd: vagrant provision --provision-with test runs tests with Vagrant

Important: Variable GISLAB_TESTS_ENABLE must be set as yes in system/host_vars/gislab_vagrant file.

vagrant reload the equivalent of running vagrant halt followed by vagrant up

vagrant status tells the state of the machines Vagrant is managing

vagrant up creates and configures guest machines according to Vagrantfile

vagrant version tells the version of the installed Vagrant as well as the latest version of Vagrant that is currently available

VBoxManage list runningvms gets a list of all running VirtualBox virtual machines
Conventions used in this documentation

There are many different organizational and typographical features throughout this documentation designed to help you get the most out of the material. You will find number of styles of text that distinguish between different kinds of information. Here are some types of headings, examples of typographical conventions, styles and an explanation of their meaning.

10.1 Styles of text

*Italic* indicates mainly important or key terms, URLs or email addresses.

**Bold** shows new terms and other text indicating that we wish to draw your attention.

Other roles like superscript and subscript text are displayed in this way.

`Code text` represents code, commands, options, switches, variables, attributes, keys, functions, types, classes, namespaces, methods, modules, properties, parameters, values, directories, objects, events, event handlers, tags, macros, the contents of files, or the output from commands.

More comprehensive parts are written in blocks as follows:

```python
some numbered python code
if re.match(r'^\d{3}-\d{4}$', test_string):
    some numbered python code highlighted
some numbered python code
```
Shell commands beginning with $ (dollar) should be run in command window.

```bash
$ some shell script code
$ if ! [ $MAX_NO -ge 5 -a $MAX_NO -le 9 ] ; then
$ some shell script code
```

< various code block >

Example

```
:command:`some command`
:guilabel:`Guilabel`
:menuselection:`First step --> Second step`
:file:`file.svg`
`GIS.lab project page <http://gislab-npo.github.io/gislab/>`
:numref:`some-figure-t`
:ref:`Conventions <conventions>`
```

General commands are written as `some command`, guilabel as `Guilabel`, direction through a menu is displayed as `First step → Second step`, name of file is represented by `file.svg`. For usage of footnotes, see\(^1\), external hyperlinks are represented as `GIS.lab web page`, for reference to some picture, see `Fig. 10.1`, `Fig. 10.2`, `Fig. 10.3` and `Fig. 10.4`, for reference to some part of page, see `Conventions`.

Example of useful term description ...
Example of useful command description ...

10.2 Short paragraphs

Tip:  
This signifies a tip, suggestion, or general useful note.

Attention:  
This style indicates a warning or caution.

Note:  
This is note.

Important:  
This represents something important.

Danger:  
This style indicates a warning or caution.

\(^1\) Some footnote.
See also:

This note leads the user to another material that is on the similar level of scope.

---

Todo: This signifies some issue to be done next time.

### 10.3 Types of Headings

For style of chapter names, please see chapter name above, for example of section, see subsection above, others are shown below.

#### 10.3.1 Subsection

Subsubsection

Paragraph

Part

Paragraph heading

etc.

```bash
Example

.. _some-figure-t:

.. figure:: ../img/login_text_logo.svg
    :align: center
    :width: 150

    GIS.lab unit tiny.
```

### 10.4 Figures

Fig. 10.1: GIS.lab unit tiny.

Fig. 10.2: GIS.lab unit small.

Fig. 10.3: GIS.lab unit middle.
10.5 Tables

Table 10.1: Table with GIS.lab contributors.

<table>
<thead>
<tr>
<th>Contributors to GIS.lab documentation</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ludmila Furtkevicova</td>
<td>Slovakia</td>
</tr>
<tr>
<td>Ivan Mincik</td>
<td>Slovakia</td>
</tr>
<tr>
<td>Martin Landa</td>
<td>Czech republic</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

10.6 Sidebars, lists and quote-like blocks

Some Sidebar

vagrant up

Example of lists:

1. numbered list
   1. nested numbered list

* bulleted list
  * nested bulleted list

10.7 Columns

Example of three columns is shown below.

• A
• B
• C
• D
• E
• F
10.8 Footnotes

See also:

Coding conventions.
CHAPTER 11

Communication channels

• Mailing list
  http://lists.osgeo.org/cgi-bin/mailman/listinfo/gis.lab

• IRC channel - server: freenode.net, channel: #gis.lab
  http://webchat.freenode.net/?channels=%23gis.lab&uio=d4
• Ansible - http://www.ansible.com/
• GDAL - http://www.gdal.org/
• GEOS - http://geos.osgeo.org/
• GRASS GIS - http://grass.osgeo.org/
• LTSP - http://www.ltsp.org/
• Vagrant - http://vagrantup.com/
• VirtualBox - https://www.virtualbox.org/
• Ubuntu Linux - http://ubuntu.com/
• PostGIS - http://postgis.net/
• PostgreSQL - http://www.postgresql.org/
• QGIS - http://www.qgis.org/
• SpatiaLite - http://www.gaia-gis.it/gaia-sins/
• and other Open Source libraries and server technologies