BUILDING A COMPLETE FREE AND OPEN SOURCE GIS INFRASTRUCTURE FOR HYDROLOGICAL COMPUTING AND DATA PUBLICATION USING GIS.LAB AND GISQUICK PLATFORMS

M. Landa[a], P. Kavka[b], L. Strouhal[b], J. Cepicky[c]

[a] Dept. of Geomatics, Faculty of Civil Engineering, Czech Technical University in Prague, Czech Republic - martin.landa@fsv.cvut.cz
[b] Dept. of Irrigation, Drainage and Landscape Engineering, Czech Technical University in Prague, Czech Republic - (petr.kavka, ludek.strouhal)@fsv.cvut.cz
[c] OpenGeoLabs s.r.o., Prague, Czech Republic - jachym.cepicky@opengeolabs.cz

Commission IV, WG IV/4

KEY WORDS: GIS, Open Source, Free Software, Deployment, Hydrology, GIS.lab, Gisquick

ABSTRACT:

Building a complete free and open source GIS computing and data publication platform can be a relatively easy task. This paper describes an automated deployment of such platform using two open source software projects – GIS.lab and Gisquick. GIS.lab (`http://web.gislab.io`) is a project for rapid deployment of a complete, centrally managed and horizontally scalable GIS infrastructure in the local area network, data center or cloud. It provides a comprehensive set of free geospatial software seamlessly integrated into one, easy-to-use system. A platform for GIS computing (in our case demonstrated on hydrological data processing) requires core components as a geoprocessing server, map server, and a computation engine as eg. GRASS GIS, SAGA, or other similar GIS software. All these components can be rapidly, and automatically deployed by GIS.lab platform. In our demonstrated solution PyWPS is used for serving WPS processes built on the top of GRASS GIS computation platform. GIS.lab can be easily extended by other components running in Docker containers. This approach is shown on Gisquick seamless integration. Gisquick (`http://gisquick.org`) is an open source platform for publishing geospatial data in the sense of rapid sharing of QGIS projects on the web. The platform consists of QGIS plugin, Django-based server application, QGIS server, and web/mobile clients. In this paper is shown how to easily deploy complete open source GIS infrastructure allowing all required operations as data preparation on desktop, data sharing, and geospatial computation as the service. It also includes data publication in the sense of OGC Web Services and importantly also as interactive web mapping applications.

1. INTRODUCTION

1.1 Open Source GIS Packages

In GIS (Geographic Information System) domain Free and Open Source Software (FOSS) plays historically a strong role. One of the first FOSS GIS project – GRASS GIS – started its development in early 80's (Neteler et al., 2012). Later in 90's and mainly after 2000 many other FOSS GIS projects were developed. A comprehensive overview is provided by (no longer maintained) FreeGIS.org website (FOSSGIS e.V., 2012). The list of GIS packages presented on this website appears to be impressive. But it is not an easy task to recognize which software projects are alive, solid, mature and reasonably maintained. Later in 2006 Open Source Geospatial Consorcium (OSGeo) was established. Its goal has been to support the collaborative development of open source geospatial software, and promote its widespread use (OSGeo, 2017). One of the important instruments of the foundation is an OSGeo Incubator[1]. FOSS GIS projects which pass the incubation procedure are graduated as official *OSGeo Projects*. This status is a clear sign to the users/consumers that such project is reasonably mature and has strong and diverse user and developer community. Shortly, it is a sign of the quality. Solid components are crucial for building GIS infrastructure. OSGeo Projects should fulfill such requirements in many ways.

[1] `http://www.osgeo.org/incubator`

1.2 Putting Blocks Together

Building a complete open source GIS infrastructure allowing operations from data preparation, analysis and computation to publishing results to end-user is a very complex task. There are two major conditions for creating well organized, and fully operational open source infrastructure: (a) solid bricks and (b) flexible glue to integrate them. Solid bricks represent a mature, well-driven open source GIS software projects. Crucial factor is the integration layer which puts all components together in a flexible but solid manner. Underestimating the importance of such integration layer (glue) leads to various problems in the infrastructure maintanance, with upgrading or replacing bricks (software packages), and hardware equipment.

In any case putting these blocks together in order to create an operative, well designed GIS infrastructure is a very complex task which requires experience, good decisions of choosing software components and ability to connect them into a working system. This paper presents GIS.lab as an open source software solution which helps to build such complete fully open source GIS infrastructure in an easy, but still fully customized manner.

2. METHODS

2.1 GIS.lab as a Core Component

GIS.lab (`http://web.gislab.io`) has been originally designed with a goal to enable simple, unbreakable deployment of a com-

Figure 1. GIS.lab logo (source: GIS.lab Documentation)

plete, centrally managed, horizontally scalable GIS infrastructure in the local network area (LAN), data center or cloud in a few steps. GIS.lab is able to turn diverse open source GIS software packages into a seamlessly integrated easy-to-use system. As a result GIS.lab significantly decreases deployment of such complex GIS infrastructure to absolute minimum, but still keeping the whole technology under a full control of the system operator. The whole technology is open source licensed under GNU GPL license.

GIS.lab cluster consists of a master node (server) and client nodes (desktop clients). All the components are running on Linux Ubuntu distribution provided by Canonical.

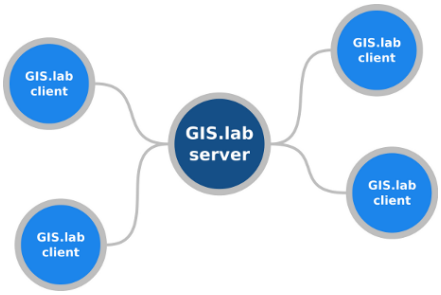

Figure 2. GIS.lab server-client based architecture (source: GIS.lab documentation)

There is a large number of possible deployment scenarios in which GIS.lab can be successfully used. GIS.lab can play a key role for building geospatial computation cluster with effective horizontally scalable computer power providing geospatial services to be consumed by different clients within a local network area or even in the data center or cloud. GIS.lab is able to turn heterogeneous computers into fully operative, centrally managed GIS easy-to-use system and maintenance-free clients in a few moments. It is an ideal platform for GIS *education* and *popularization* of open source technologies. GIS.lab technology can be incorporated into an existing computer network, or create its own computer network. The latter option can be useful for crisis management in very hard conditions of natural disaster with Internet outages. GIS.lab can be customized in many ways in order to fulfill different requirements.

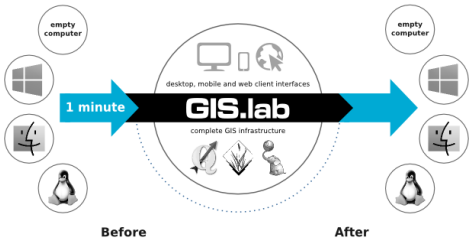

Figure 3. GIS.lab infrastructure as a platform for teaching, computation, or crisis management (source: GIS.lab Documentation)

This paper shows how to easily build a complete open source GIS infrastructure on the top of GIS.lab enabling highly specialized hydrological data processing. Such system is performed by automated customization of GIS.lab ecosystem including master (server) and client nodes.

Figure 4. GIS.lab components running in the real environment (source: GIS.lab Documentation)

2.2 Gisquick as a Publication Platform

Gisquick (`http://gisquick.org`) is a separate project not directly related to GIS.lab. It is a web application based on modern technologies as Django, Angular and OpenLayers 3 with fully responsive design optimized also for mobile devices. The main purpose of Gisquick is to provide the capability for easy publishing the QGIS projects on the web. From this perspective Gisquick is strongly connected to QGIS Desktop environment and stands on QGIS Server component.

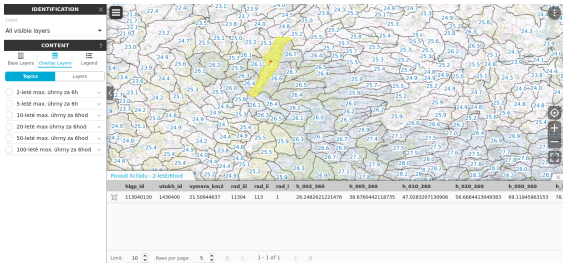

Figure 5. Gisquick web application interface (source: author)

Combining GIS.lab and Gisquick technologies leads to a complete, seamlessly integrated platform capable to prepare the input data, perform geospatial analysis and publish results easily on the web in the sense of interactive web mapping application.

3. CASE STUDY

3.1 GIS Infrastructure for Hydrological Computation and Data Processing

At first, let's put together basic requirements for presented platform. Desired system should allow the user to collect, prepare, and preprocess data from heterogeneous data sources for hydrological computation using GIS software packages. The user will be able to perform hydrological computation locally on desktop clients, and also consume dedicated geoprocessing service provided by a master node running in the infrastructure. The results of hydrological data processing can be easily published via web services from user desktop environment and ideally also as the interactive web mapping applications provided by the server component (master node) in the infrastructure.

Most of the requirements are already fulfilled by GIS.lab platform. It provides fully established computer network consisting of server (master node) and desktop clients (client nodes). Main objective of GIS.lab is a rapid deployment of complete geospatial computation platform enabling collaborative data managing, storing, processing, and analysing thanks to the fully automatic provisioning. There are Lighttpd web server running on the master node providing web services, PostGIS geospatial database server for collecting, storing and manipulating geospatial data, GRASS GIS engine for performing geospatial analysis, and QGIS Server providing OGC Web Services (OWS). On the client side resides the requested desktop GIS application for collecting, managing and preprocessing input geospatial data. For this task GIS.lab desktop client offers the well known QGIS Desktop application. Hydrological data processing in the desktop environment can be performed by the integrated GRASS GIS and QGIS GRASS plugin.

Missing features can be easily integrated into GIS.lab eco-system by customizing the deployment of the both master/server and desktop client components. Since the geoprocessing service should be provided by the customized infrastructure, the master node must be extended by an application able to provide OGC Web Processing Service (WPS). In this paper seamless deployment integration of PyWPS version 4 was demonstrated.

Another missing component is a publishing platform providing the ability to easily publish the results of computation in the form of an interactive web mapping application. For this purpose Gisquick open source project was used. Gisquick integration was executed by using Docker containers.

The last missing component – a tool for hydrological computation – is shown on GRASS AddOn tool *r.subdayprecip.design* integration in the sense of desktop tool as well as a geoprocessing service implemented using OGC WPS. Presented *r.subdayprecip.design* GRASS module provides the subday design precipitation totals based on hydrological Hradek's method of reduction of daily maximums to a chosen duration (Landa et al., 2015).

3.2 Fundamentals

As mentioned earlier, one of the main objectives of GIS.lab technology is a rapid and fully automated deployment. For this task GIS.lab uses *Ansible* framework. Ansible offers human-readable automatizing language, agent-less execution, various modules and support for different providers like AWS (Amazon Web Services), Azure and others (Hochstein, 2014). It is important to mention that current GIS.lab (version 0.7) supports two providers: GIS.lab Unit and AWS. In this paper we will focus on GIS.lab Unit provider in order to have the whole infrastructure under the full control.

Described procedure consists of three major steps:

1. Deployment of a master node using GIS.lab technology
2. Master node customization
3. Gisquick integration

3.3 Deployment of a Master Node

Master node (playing a role of a server) of geospatial cluster can be automatically deployed thanks to GIS.lab technology. There are some hardware and software requirements which have to satisfied. First of all, hardware on which the master node will be running need to be available, it can be everything from Intel NUC unit (ideal for testing) to real server rack hardware inhouse or provided by AWS, Azure or similar services. The operator runs host (controlling) machine which will be used for master node provision. On controlling machine Ansible framework must be installed. Node(s) described in so-called *inventory* file are accessed over SSH.

Figure 6. GIS.lab master node running on different providers (source: GIS.lab Documentation)

At first on controlling machine needs to downloaded GIS.lab source code (available freely from GitHub repository[2]) and standard Ubuntu Server installation ISO image. In the next step a new customized ISO will be created by GIS.lab utility `gislab-unit-iso.sh` and used for installation on master node hardware unit. After successful installation a plain Ubuntu operating system will be available on that machine.

Afterwards two Ansible playbooks (both located in GIS.lab source code tree) will be launched from the controlling machine. The first playbook `gislab-unit.yml` is dedicated for Intel NUC only and stands for GIS.lab Unit initialization. The second – core – playbook `gislab.yml` performs provisioning the whole GIS.lab technology on master node equipment. It installs all required software packages, sets up all the services, creates and configures desktop client images. This step can take from few dozen of minutes to few hours depending on Internet connection speed and hardware limits. Simplified commands are shown below.

```
$ ansible-playbook ... providers/../gislab-unit.yml
$ ansible-playbook ... system/gislab.yml
```

Figure 7. All required hardware components for GIS.lab deployment (source: GIS.lab Documentation)

Detail information about GIS.lab deployment procedure including its configuration is part of an official GIS.lab documentation (GIS.lab Team, 2017).

After successful deployment, a fully operative master node is running in our infrastructure providing all the services which GIS.lab

[2] `https://github.com/gislab-npo/gislab`

offers out of the box. For our use-case is crucial database server (PostgreSQL), map server (QGIS Server) and GIS computation engine (GRASS GIS). All these components are available and configured by GIS.lab. The clients can boot from master node using standard protocols like PXE or HTTP. The master and client nodes creating geospatial cluster includes technologies like load balancing and centrally managed system to control them. It also covers user accounts management which is launched centrally by LDAP server running on a master node.

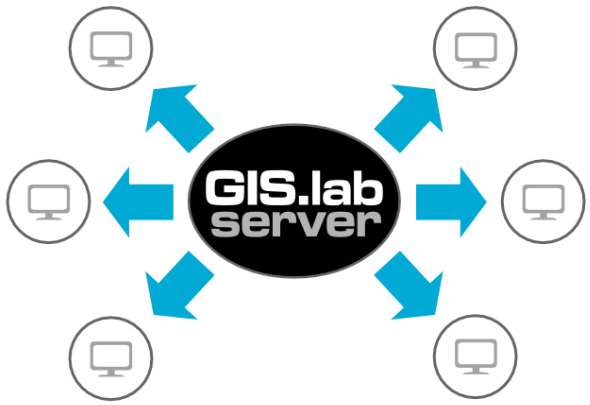

Figure 8. Building geospatial cluster using master and client nodes (source: GIS.lab Documentation)

3.4 Master Node Customization

Master node has been successfully deployed using GIS.lab technology. In the following steps customization will be performed. Customization rules are defined by *Ansible Playbooks* similarly how GIS.lab provision works. Ansible Playbooks express configurations, deployment, and orchestration in Ansible (Shah, 2015). Playbooks are based on YAML human-readable data serialization language and Jinja templates. As described in section 3.1 customization procedure will performed. Let's summarize our requirements:

1. Install and configure PyWPS4 on GIS.lab master node
2. Install GRASS AddOn module *r.subdayprecip.design* on server-side (master node) and also in client images in order to use this tool locally on desktop clients and also as remote geoprocessing service provided by PyWPS4 running on master node (server)
3. Set up PyWPS4 process based on *r.subdayprecip.design* functionality.

Ansible Playbooks used for customization will be saved in *provision* directory and separated into set of *roles*. Each role is represented by *tasks* which perform calls to Ansible. Tasks for each role are defined in *main.yml* YAML file. Example below demonstrates how can be implemented a role for installing and configuring PyPWS4.

```
1  - name: Install PyWPS4
2    pip:
3      name: pywps
4      version: 4.0.0
5
6  - name: Set up PyWPS4 configuration
7    template:
8      src: pywps.cfg.j2
9      dest: /opt/pywps4/pywps.cfg
10     mode: 0644
```

PyWPS4 package will be installed by standard pythonic *pip* command which is implemented by Ansible *pip module*, see lines 1-4. In the next step, PyWPS4 configuration file is placed in the destination folder on the server, see lines 6-10. Similarly is processed PyWPS4 WSGI application file. From provisioning point of view, the both files are placed in *templates* directory. Template module based on Jinja templating language allows variable propagation. Simplified and shorten example of sample PyWPS4 configuration file *pywps.cfg.j2* is shown below.

```
1  [server]
2  url={{ base_url }}/services/wps
```

Variable values can be defined in *main.yml* file located in *vars* directory.

```
1  base_url: http://gislab.mydomain.org
```

In a similar way other two roles for installing GRASS tool *r.subdayprecip.design* and related WPS process will be defined. The roles are put together in main *deployment.yml* YAML file.

```
1  roles:
2      - { role: pywps4 }
3      - { role: subdayprecip-tool }
4      - { role: subdayprecip-wps }
```

Automated customization of master node will be performed by *Ansible Playbook* command similarly to GIS.lab deployment procedure described in section 3.3.

```
$ ansible-playbook ... provision/deployment.yml
```

As a result customizated master node is providing a new service based on OGC WPS thanks to integrated PyWPS4 software package. Newly installed GRASS tool for hydrological computation of subday design precipitation totals based on hydrological Hradek's method of reduction of daily maximums to chosen duration is available on master node and desktop clients. Also ready-to-use WPS process providing described functionality is offered by a master node via OGC WPS service and accessible using HTTP(S) protocol.

A complete overview of customization Ansible Playbooks is available from GitHub repository[3]. For another example of GIS.lab customization see GISMentors training group GitHub repository[4].

3.5 Gisquick integration

On Gisquick publishing platform a different approach is demonstrated. Instead of specific Ansible Playbooks, integration based on *Docker* containers is performed. Gisquick project provides core Docker images needed for successful running of this publishing platform. It significantly simplifies Gisquick integration into GIS.lab infrastructure.

The core Gisquick Docker images are listed below:

- gisquick/nginx
- gisquick/django
- gisquick/qgisserver

[3] https://github.com/ctu-geoforall-lab/subdayprecip-design

[4] https://github.com/GISMentors/gislab-customization

These Docker images can be automatically composed on GIS.lab master node by running *docker-compose* command.

```
$ sudo docker-compose up
```

This command need to be run in the directory where Dockerfiles for Gisquick are located. Official Dockerfiles are available from Gisquick Git repository on GitHub[5]. Before running *docker-compose* command a configuration rules for composing Docker images must be created. The rules are stored in the file named *docker-compose.yml* located in the same directory as Dockerfiles. Shorten example of compose YAML file is shown below.

```
1  version: "2"
2  services:
3    qgisserver:
4      image: gisquick/qgis-server
5      volumes:
6        - /storage/publish:/publish/:ro
7
8    django:
9      image: gisquick/django
10     volumes:
11       - /storage/media:/var/www/gisquick/media/
12       - /storage/data:/var/www/gisquick/data/
13
14   nginx:
15     image: gisquick/nginx
16     volumes:
17       - /storage/letsencrypt/:/etc/letsencrypt/
```

The most important part of configuration from GIS.lab integration point of view is definition of volumes, see lines 5, 10, 16. Here are listed local directories which will be mounted into running Docker containers. This procedure can be automized also by Ansible Playbooks.

Gisquick is running on master node and available to the clients using HTTP(S) protocol in the GIS.lab network. Clients can publish their projects by QGIS Gisquick plugin and simply copying them to shared directories mounted over the Network File System from master node.

4. CONCLUSIONS

Geospatial cluster can be arranged thanks to open source orchestration technologies in an automated and easily maintainable manner. This paper presents Ansible framework allowing fully automated software provisioning, configuration management, and application deployment. On the top of Ansible framework a GIS.lab project is built. GIS.lab significantly reduces the whole procedure of design, deployment and configuration of GIS infrastructure to the virtual minimum. GIS.lab itself offers basic components, tools, and services for building GIS (and not only GIS) infrastructure. GIS cluster consists of the master node and the client nodes. Client nodes are booting from the master using PXE or HTTP boot protocols. Master node services, user accounts and data volumes are centrally managed and consumed by the clients over the Network File System.

The master and client nodes can be customized using the specific Ansible Playbooks including software provisioning and configuration management. Customized infrastructure can be easily deployed on a new hardware equipment. Integration of the new components can be simplified by the Docker containers in order to combine them into the working system seamlessly integrated into the customized infrastructure.

[5] `https://github.com/gislab-npo/gisquick`

REFERENCES

FOSSGIS e.V., 2012. FreeGIS.org. http://freegis.org (21 May 2017).

GIS.lab Team, 2017. Online GIS.lab Documentation. http://gislab.readthedocs.io (24 May 2017).

Hochstein, L., 2014. *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way*. O'Reilly Media.

Landa, M., Kavka, P. and Strouhal, L., 2015. A GIS Tool for Reduction Day Precipitation to Subday. In: *Geomatics WorkBooks 12*, p. 725. ISSN 1591-092X.

Neteler, M., Bowman, M. H., Landa, M. and Metz, M., 2012. GRASS GIS: A multi-purpose open source GIS. *Environmental Modelling & Software* 31, pp. 124–130.

OSGeo, 2017. About the Open Source Geospatial Foundation. http://www.osgeo.org/content/foundation/about.html (22 May 2017).

Shah, G., 2015. *Ansible Playbook Essentials*. Packt Publishing.

ACKNOWLEDGEMENTS

This work has been supported by the research project QJ1520265 - "Variability of Short-term Precipitation and Runoff in Small Czech Drainage Basins and its Influence on Water Resources Management".