OPTIMISED RISK ANALYSIS

www.monarc.lu

Technical Guide

CASES Luxembourg

Version 2019-11-15

Table of Contents

Chapter 1. Introduction

1.1. Purpose

This document is intended to administrators of a MONARC instance. If you find errors or omissions in this document, please don't hesitate to submit an issue or open a pull request with a fix.

1.2. Other documents

NOTE

- **Quick Start**: Provide a quick start with MONARC.
- **User Guide**: Complete documentation of the tool.
- **Method Guide**: Complete documentation of the method.

Chapter 2. Requirements

2.1. System and software

The deployment of MONARC is mainly tested on Ubuntu 18.04 LTS. In order to install MONARC you will need the following requirements:

- Git;
- PHP (version 7.1, 7.2 or 7.3):
  - PHP libraries are managed with composer;
- JavaScript:
  - JavaScript libraries are managed with npm;
- Apache 2;
- MariaDB.

Postfix, or an equivalent software, is required for the account creation and password recovery features.

You will find more details about the installation here.

2.2. Specifications by server

Minimum requirements

	Back office	MONARC	Reverse proxy	Configuration server
Number of vCPU	2	2	Not that powerful. But all connections (for the BO and FOs) will go through it.	Not that powerful. Mainly used for the creation of clients on the different FOs.
RAM (GB)	2	4		
HDD (GB)	20	20		

Of course these values are minimum requirements. MONARC can be installed on a Raspberry Pi. It is always better to forecast more memory and disk space for MONARC (front office) since it will host the data of the analysis.

2.3. Network

The deployment on the different servers requires an Internet connection since the updates are retrieved from our GitHub repositories. If this is hardly possible in your environment due to an internal policy, you need to configure a proxy in order to access to Internet. This is an important requirement since all MONARC updates, including security updates, are provided from the Git repositories.

The ansible playbook also needs an Internet connection in order to dynamically deploy new clients.

The servers (or the proxy) should be able to contact:

- github.com
- api.github.com
- pypi.python.org
- registry.npmjs.org
- deb.nodesource.com
- getcomposer.org
- .packagist.org
- packagist.phpcomposer.com
- letsencrypt.org
- APT repositories

To make the different servers use the proxy, specify the address of your proxy in the file `/home/ansible/.bash_profile`. For example:

```
export http_proxy=http://proxy.your.domain.com:4128
export https_proxy=http://proxy.your.domain.com:4128
```

2.4. TLS certificate

Is is strongly advised to use a TLS certificate.

In the case you are using MONARC with a back office you can set the certificate on the reverse proxy. The ansible playbook is compatible Let's Encrypt auto-renew certificates. You will find more information in the ansible playbook.

If you are not using a back office you can simply configure the certificate on the front office server.

Chapter 3. Architecture

3.1. Modules of the project

The source code of MONARC is divided in several modules with dedicated Git repositories.

Summary of the different modules of the project

	MONARC	Back Office
Modules	• MonarcCore (zm-core in `vendor/monarc/core`) • MonarcFO (zm-client in `vendor/monarc/frontoffice`)	• MonarcCore (zm-core in `vendor/monarc/core`) • MonarcBO (zm-backoffice in `vendor/monarc/backoffice`)
Interfaces (in node_modules/)	• ng_anr • ng_client	• ng_anr • ng_backoffice

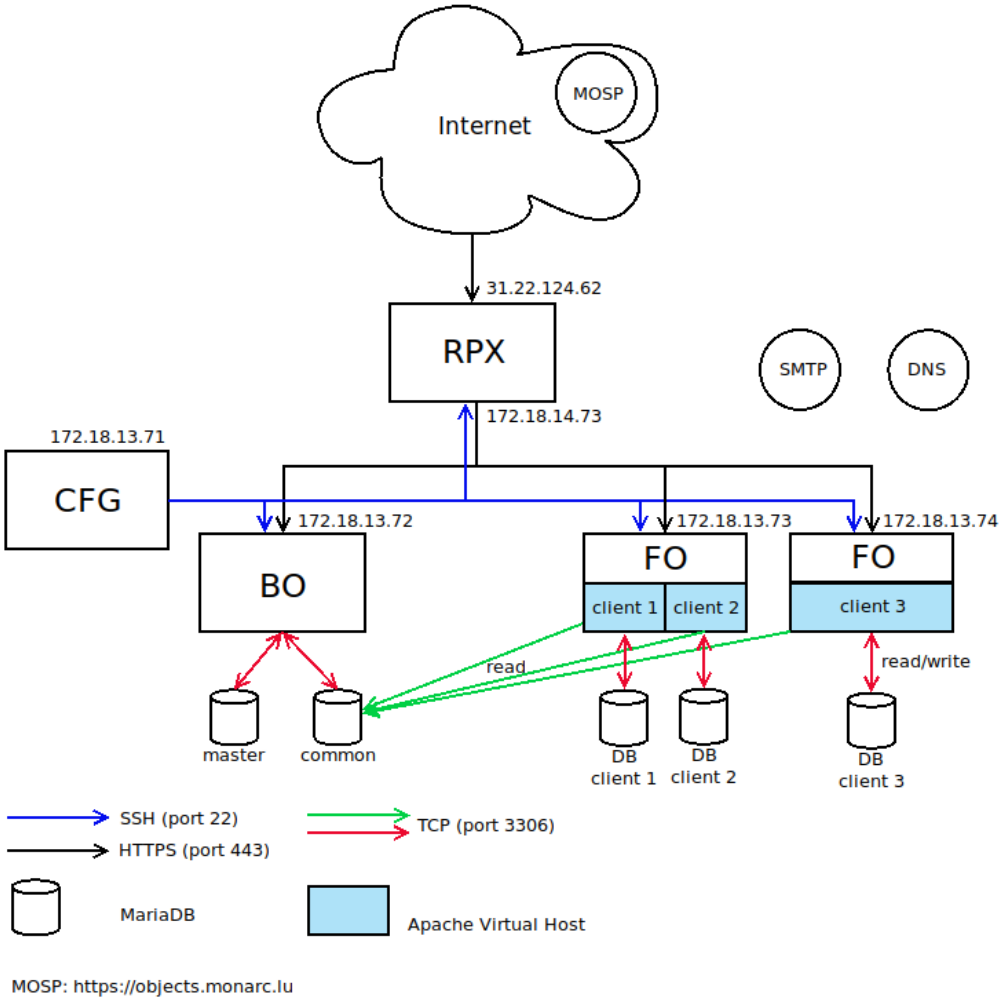
3.2. Global architecture

Basically MONARC is composed of two main parts:

- a back office: for the management of the security models, FO servers and clients of MONARC installations;
- a front office: for the management of the risks analysis. A front office instance can be used without the back office.

NOTE If you want run your own risk analysis you only need the front office. More information about the installation in this section.

The whole architecture includes an additional reverse proxy (**RPX**) and a configuration server (**CFG**). It is possible to connect many front offices (**FO**) to one back office (**BO**). The deployement is managed with ansible. More information in the repository of the ansible playbook.



MOSP is a platform to create, edit and share JSON objects. The goal is to gather security related JSON objects, in the first place aimed to be used with MONARC. You can use any available schemas in order to create new JSON objects. You can have a look at official instance operated by CASES. MONARC instances can directly query MOSP in order to update their knowledge base. You can have a look at the documentation of MOSP.

NOTE

Example of IPs and FQDNs:

- 172.18.14.73 - monarc-rpx.private.your.domain.com - my.monarc.lu
- 172.18.13.71 - monarc-conf.private.your.domain.com
- 172.18.13.72 - monarc-master.private.your.domain.com
- 172.18.13.73 - monarc-fo01.private.your.domain.com
- 172.18.13.74 - monarc-fo02.private.your.domain.com

In this example, the **FO** server *monarc-fo01* hosts two distinct databases for two clients. An Apache Virtual Host is also configured for each clients. A user who wants to do a security analysis with MONARC will access to his MONARC account via the address https://my.monarc.lu/formation/

Note that the *common* database of the **BO** is listening on `0.0.0.0:3306`.

It is strongly recommended to use a TLS certificate on the reverse proxy in order to provide a HTTPS connection between Internet and the reverse proxy. In our example, a TLS certificate for my.monarc.lu is used.

3.3. Network requirements

The configuration server (**CFG**) manages the configurations of the back office, reverse proxy and different front office(s) via SSH.

The reverse proxy (**RPX**) should of course be available from Internet. The **DNS** is important for the reverse proxy in order to resolve the FQDNs of the servers inside the network.

Through the reverse proxy you should be able to contact the **BO** and the **FO**s from Internet. No need to able able to contact the configuration server from the outside.

The database *common* of the BO should also be available to the FO servers.

Postfix should be installed on the BO and on the FO. Indeed, SMTP is used for the account creation and the password recovery. The easiest configuration is to set up Postfix for relaying emails through your internal mail server (**SMTP**).

Chapter 4. Deployment

4.1. MONARC

If you want to install MONARC to manage your risk analysis.

4.1.1. Prepared virtual machine

A virtual machine for use with VirtualBox is available. The best way if you want an up-to-date version of MONARC.

4.1.2. Manual deployment

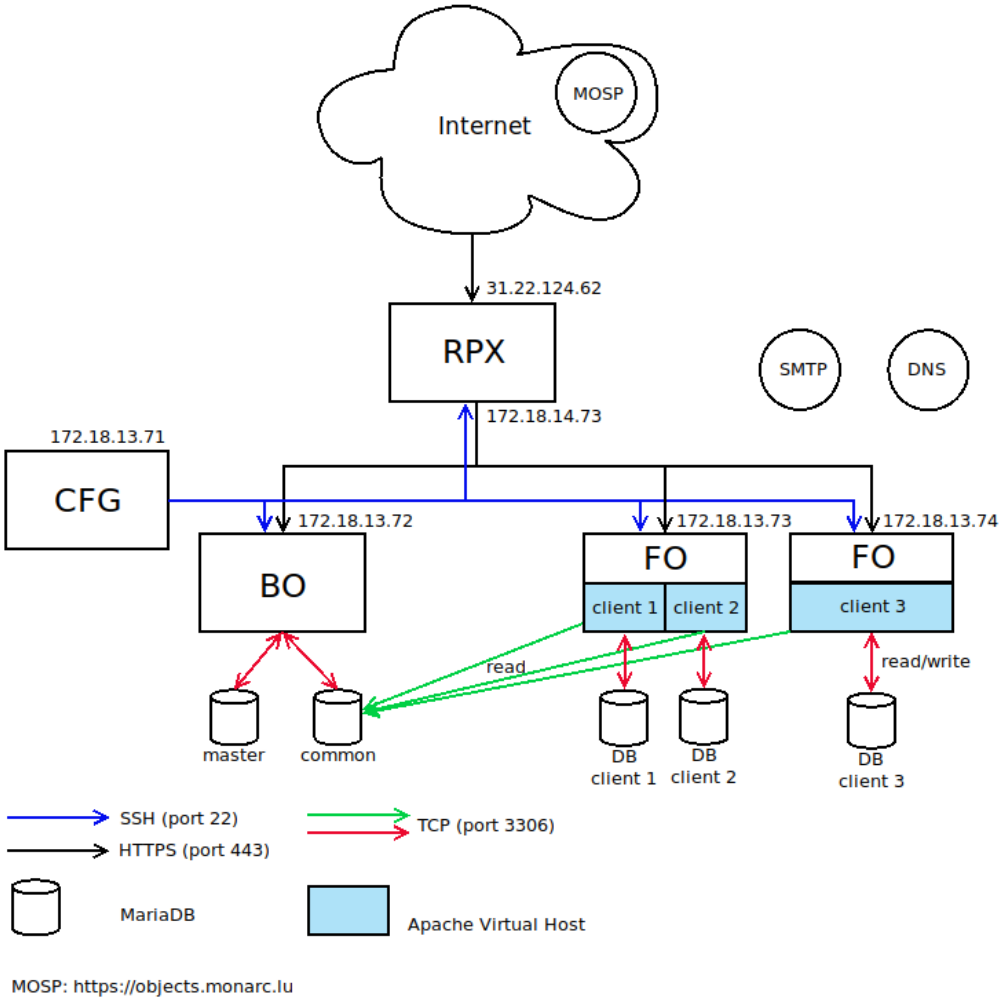
Follow the instructions here.

4.2. MONARC and the back office

If you want to install MONARC and the back office.

NOTE	This is useful if you plan to manage several clients. If you want run your own risk analysis you only need the front office. More information about the installation in this section.

The whole architecture can be deployed with ansible.



4.2.1. Requirements

- Git and Python on all servers;
- ansible must be installed on the configuration server;
- Postfix on the BO and all FO servers (for the password recovery feature of MONARC).

4.2.2. Usage

Install ansible on the configuration server and get the playbook for MONARC:

```
sudo apt-get install python-pip
sudo -H pip install ansible dnspython
git clone https://github.com/monarc-project/ansible-ubuntu.git
cd ansible-ubuntu/
```

4.2.3. Configuration

- create a user named **ansible** on each server;
- generate a SSH key for the user **ansible** on the configuration server:

```
ssh-keygen -t rsa -C "your_email@example.com"
```

- copy the public key on the other servers:

```
ssh-copy-id ansible@BO
ssh-copy-id ansible@RPX
ssh-copy-id ansible@FO
```

- add the user **ansible** in the **sudo** group:

```
sudo usermod -aG sudo ansible
```

- add the user **www-data** in the **ansible** group:

```
sudo usermod -aG ansible www-data
```

- give the permission to ansible to use sudo without password:

```
echo 'ansible ALL=(ALL:ALL) NOPASSWD:ALL' >> /etc/sudoers
```

- create a file *inventory/hosts*:

```
[dev]
FO

[dev:vars]
master= "BO"
publicHost= "monarc.example.com"

[master]
BO monarc_sql_password="password"

[rpx]
RPX.localhost

[monarc:children]
rpx
master
dev

[monarc:vars]
env_prefix=""
clientDomain="monarc.example.com"
emailFrom="info@example.com"
github_auth_token="<your-github-auth-token>"
protocol="https"
certificate="sslcert.crt"
certificatekey="sslcert.key"
certificatechain="sslcert.crt"
bourlalias="monarcbo"
localDNS="example.net"
```

The variable **monarc_sql_password** is the password for the SQL database on the BO.

- finally, launch ansible:

```
cd playbook/
ansible-playbook -i ../inventory/ monarc.yaml --user ansible
```

ansible will install and configure the back office, the front office and the reverse proxy. Consequently the configuration server should be able to contact these servers through SSH.

You will find more information in the ansible playbook.

4.2.4. Launch ansible periodically with cron

```
$ crontab -l
# m h  dom mon dow   command
*/30 * * * * /home/ansible/ansible-ubuntu/playbook/update.sh /home/ansible/ansible-
ubuntu BO
```

This cron task should be launched on the configuration server by the **ansible** user.

If a new client is created via the web interface of the back office, this task will instantiate a new MONARC instance (new database, new Apache VirtualHost and eventually a new server) for the newly created client.

4.2.5. Update the ansible playbook

```
$ cd /home/ansible/ansible-ubuntu/
$ git pull origin master
```

Chapter 5. Updates

Before updating MONARC it is advised to configure database backup. For that you need to create a file `data/backup/credentialsmysql.cnf`:

```
[client]
host     = localhost
user     = sql-monarc-user
password = your-password
socket   = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
host     = localhost
user     = sql-monarc-user
password = your-password
socket   = /var/run/mysqld/mysqld.sock
basedir  = /usr
```

If this file is not present, a warning message will be displayed during the update.

5.1. System update

Keep the software of your distribution up-to-date (Apache, PHP, MariaDB, Postfix, etc.). At least the security updates from the GNU/Linux distribution.

5.2. MONARC update

5.2.1. Update a single MONARC instance

If you have already installed MONARC and want to update to a later version, you can use the provided script:

```
$ ./scripts/update-all.sh -c
$ rm -Rf ./data/cache
$ sudo systemctl restart apache2.service
```

Updates from the last stable release of MONARC will be retrieved.

For more details the script will:

- backup your databases in case the update fails (if enabled, as explained previously);
- check the presence of composer and if needed install it;
- retrieve the new code from the last stable release (master branch):
  - pull updates for module/[MonarcCore, MonarcBO/MonarcFO] via composer;
  - pull updates for node_modules/[ng_anr, ng_backoffice/ng_client].

- run the appropriate database upgrade scripts;
- update the translations.

Please be aware that by default the update script will run database migrations This behavior can be changed by passing the `-b` option to the update script.

5.2.2. Update MONARC when connected to a back office

When MONARC is connected to a back office and has been deployed with the ansible playbook, three steps are required for a full proper update.

1. Update of the back office
2. Update of MONARC
3. Update of the configuration server (ansible playbook)

Always use the **ansible** user.

Back office

```
$ su ansible
$ cd /var/lib/monarc/bo/MonarcAppBO/
$ ./scripts/update-all.sh
$ sudo systemctl restart apache2.service
```

Front office

```
$ su ansible
$ cd /var/lib/monarc/fo
$ ./scripts/update-all.sh -b
$ sudo systemctl restart apache2.service
```

Configuration server

```
$ su ansible
$ cd /home/ansible/ansible-ubuntu/
$ git pull origin master
```

If you have any problem you can check the section dedicated to the common issues.