RIR Dashboard

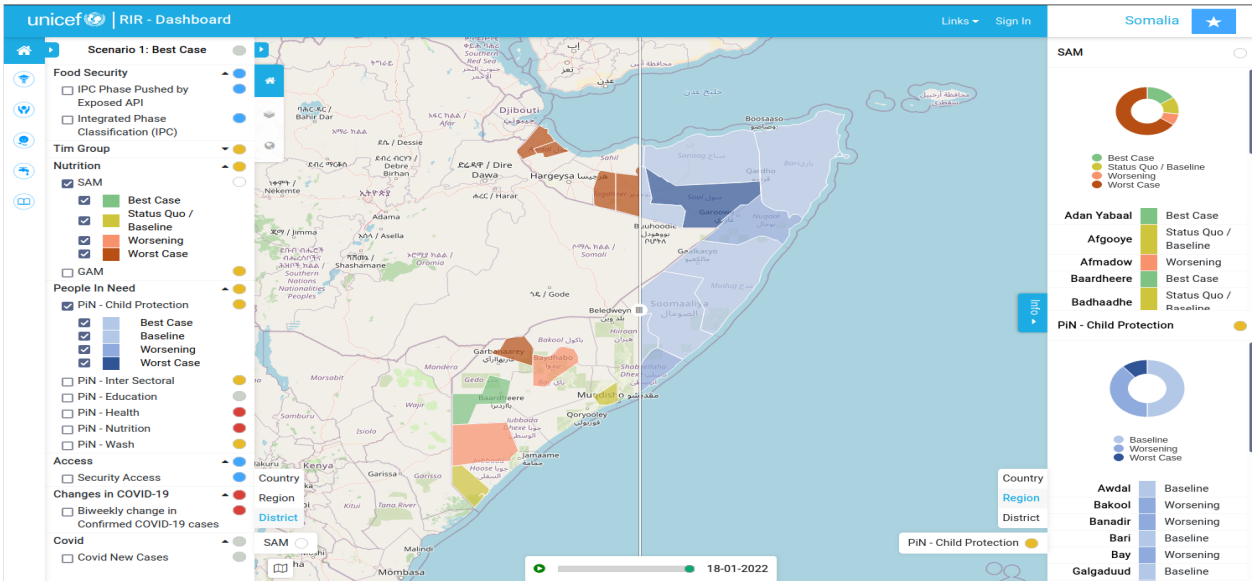
Release 1.0

Tim Sutton, Irwan Fathurrahman

Jan 25, 2022

CONTENTS:

Welcome to the Risk Informed Response (RIR) Platform!



The RIR Platform is a tool for humanitarian workers to plan for and mitigate against risk. The platform is Open Source and can be shared and reused for your own purposes under the terms of our license.

CHAPTER

ONE

INTRODUCTION

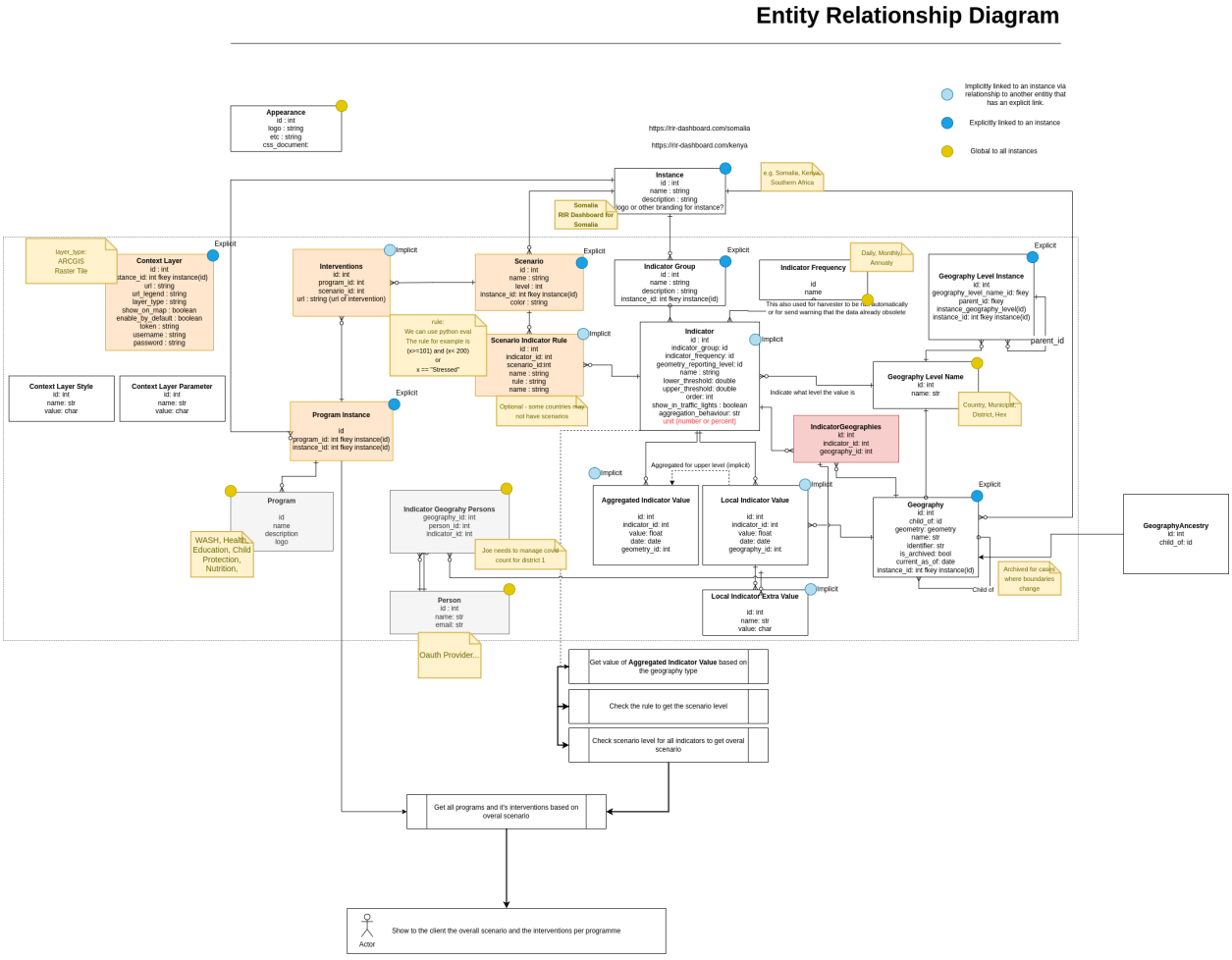
The Open Source GIS Stack by Kartoza is maintained in the kartoza / osgs repository.

1.1 What is the Open Source GIS Stack?

The platform presents the user with a geographical view of a country or region with the administrative units colour coded according to their risk factors. Some other key features include:

- classification of risk into different scenarios and linking to contingency plans based on those scenarios
- temporal view of factors and how they change over time
- side-by-side comparison of risk factors (indicators) using a map swiping tool
- ad hoc creation of indicators
- harvesters that can automatically update indicators from different data sources such as web API's
- a complete and user friendly administration environment to manage the system
- role based user management

1.2 Overview Diagram



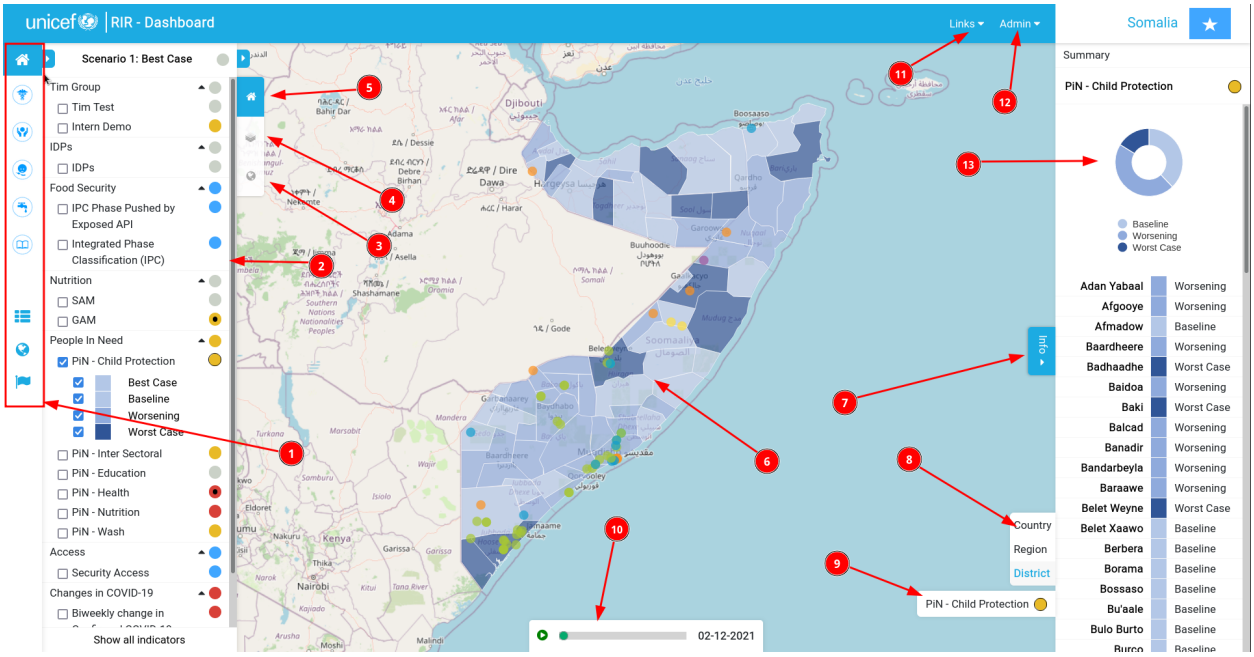
CHAPTER

TWO

RIR USER DOCUMENTATION

2.1 Platform Tour

Before we dive into details about the different functions available on the platform, let's first start with a brief tour of the main user interface so that we can refer to things using shared language.



1. **The main toolbar**: The main toolbar is used to navigate around the key areas of the RIR Platform. Some of the icons may not be visible to you if you do not have administrator access on the platform.

2. **The indicator panel**: This panel contains a list of indicators that relate to health, security, human welfare etc. The coloured circles next to each indicator shows the current severity level for that indicator. The indicators are arranged in groups which also have an severity level for the group. The entire collection of indicators is given an overall status at the top of the list.

2.2 Indicators

2.2.1 Level 3

2.3 Map Interactions

CHAPTER

THREE

RIR ADMINISTRATOR DOCUMENTATION

CHAPTER

FOUR

RIR TECHNICAL DOCUMENTATION

4.1 Installation

4.1.1 Preparing the server

Basic Security

Unattended upgrades

This will automatically install only security fixes on a continual basis on your server.

```
sudo apt install unattended-upgrades
```

ssh

Disable password authentication for SSH

```
sudo vim /etc/ssh/sshd_config
```

Set this:

```
PasswordAuthentication no
Then do
sudo systemctl restart sshd.service
```

Crowdsec

https://crowdsec.net/

```
wget -qO - https://s3-eu-west-1.amazonaws.com/crowdsec.debian.pragmatic/crowdsec.asc␣
↪|sudo apt-key add - && echo "deb https://s3-eu-west-1.amazonaws.com/crowdsec.debian.
↪pragmatic/$(lsb_release -cs) $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.
↪d/crowdsec.list > /dev/null;
sudo apt-get update
sudo apt-get install crowdsec
```

Fail2ban

```
sudo apt install fail2ban
```

See: https://www.fail2ban.org/wiki/index.php/Main_Page

Firewall

```
sudo ufw allow ssh
sudo ufw enable
sudo ufw status
```

Should show something like this:

```
Status: active

To              Action   From
--              ------   ----
22/tcp            ALLOW    Anywhere
22/tcp (v6)         ALLOW    Anywhere (v6)
```

We will open more ports as they are needed.

Status monitoring

gotop is a great console based dashboard for monitoring your server.

Ubuntu:

```
sudo apt-get install golang
cd
go get github.com/cjbassi/gotop
chmod +x go/bin/gotop
sudo cp go/bin/gotop /usr/local/bin/
```

Fedora:

```
sudo dnf install golang
cd
go get github.com/cjbassi/gotop
chmod +x go/bin/gotop
sudo cp go/bin/gotop /usr/local/bin/
```

Now just type gotop whenever you want to see your terminal system monitor.

Additional Software

Docker

```
sudo apt install docker.io
sudo apt-get -y install python3-pip
sudo pip3 install docker-compose
```

Git, rpl, pwgen, Make and openssl

Needed for checking out our docker project and running the various make commands we provide.

```
sudo apt install git make rpl pwgen openssl apache2-utils
```

or fedora:

```
sudo dnf install openssl rpl git pwgen
```

Firewall

If you are using ufw, open port 80 and 443 as minimum. After the initial setup, you can again close port 80.

```
sudo ufw allow 80
sudo ufw allow 443
```

Move on to OSGS Installation

Ok we are ready to install OSGS! Go ahead to the *initial configuration page* now.

4.1.2 Initial Configuration

User Group

Add yourself to the user group of docker so you don't need to sudo docker commands.

```
sudo usermod -a -G docker $USER
```

On linux you can run:

```
newgrp docker
```

To become part of the docker group. On other Operating Systems you should log out and in again to assume the upgraded permissions.

Project Checkout

Note you can check out the project anywhere, but for our examples we will use /home/web/rir-dashboard.

```
cd /home
sudo mkdir web
sudo chown timlinux.timlinux web
cd web
git clone https://github.com/kartoza/rir-dashboard
cd rir-dashboard
```

Configuration

If you are going to use a self-signed certificate on a localhost (for testing):

```
make configure-ssl-self-signed
```

If you are going to use a letsencrypt signed certificate on a name host (for production):

```
make configure-letsencrypt-ssl
```

Fetching Docker Images

You can optionally prefetch all the docker images that are used in the stack.

```
docker-compose pull
```

4.1.3 Production Stack

Overview

In this section we will bring up the full production stack, but to do that we first need to get an SSL certificate issued. To facilitate this, there is a special, simplified, version of Nginx which has no reverse proxies in place and not docker dependencies. Here is an overview of the process:

1. Replace the domain name in your letsencrypt init script
2. Replace the email address in your letsencrypt init script
3. Replace the domain name in the certbot init nginx config file
4. Open up ports 80 and 443 on your firewall
5. Run the init script, ensuring it completed successfully
6. Shut down the minimal nginx
7. Replace the domain name in the production nginx config file
8. Generate passwords for geoserver, postgres, postgrest and update .env
9. Copy over the mapproxy template files
10. Run the production profile in docker compose

At the end of the process you should have a fully running production stack with these services:

IMAGE	PORTS	NAMES
nginx:alpine	0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp	osgisstack_nginx_1
quay.io/lkiesow/docker-scp	22/tcp	osgisstack_scp_1
kartoza/hugo-watcher:main	1313/tcp	osgisstack_scp_1

The following ports will be accessible on the host to the docker services. You can, on a case by case basis, allow these through your firewall using ufw (uncomplicated firewall) to make them publicly accessible:

1. 80 - http:Only really needed during initial setup of your letsencrypt certificate
2. 443 - https: All web based services run through this port so that they are encrypted
3. 5432 - postgres: Only expose this publicly if you intend to allow remote clients to access the postgres database.
4. 2222 - scp: The is an scp/sftp upload mechanism to mobilise data and resources to the web site

For those services that are not exposed to the host, they are generally made available over 443/SSL via reverse proxy in the Nginx configuration.

Some things should still be configured manually and deployed after the initial deployment:

1. Mapproxy configuration
2. setup.sql (especially needed if you are planning to use postgrest)
3. Hugo content management
4. Landing page static HTML

And some services are not intended to be used as long running services. especially the ODM related services.

Configuration

We have written 2 make targets that automate the steps 1-10 described in the overview above. Either target will ask you for your domain name, legitimate email address and then go ahead and copy the templates over, replace placeholder domain names and email address, generate passwords for postgres etc. and then run the production stack. Remember you need to have ufw, rpl, make and pwgen installed before running either of the commands below.

If you are going to use a self-signed certificate on a localhost (for testing) run:

```
make configure-ssl-self-signed
```

If you are going to use a letsencrypt signed certificate on a name host (for production) run:

```
make configure-letsencrypt-ssl
```

CHAPTER

FIVE

RIR ROADMAP

Like most Open Source Software, this project is an ongoing work in progress.

This document outlines the various ongoing activities and critical changes expected to be introduced.

5.1 Future plans

- A data model for scenarios instead of pulling in google sheets or similar
- More ingestors for risk data
- Hazard data support with auto aggregation of risk factors such as population counts etc.
- Single sign on support

5.1.1 Considerations

The current *main* branch is under heavy development and should be considered unstable.