InfraCheck Documentation

Release 2

Wolnosciowiec Team

Apr 08, 2023

Contents:

HTTP healthcheck endpoint + shell healthcheck runner. Simple, easy to setup, easy to understand. Works perfectly with Docker. A perfectly fitting universal brick in your monitoring.

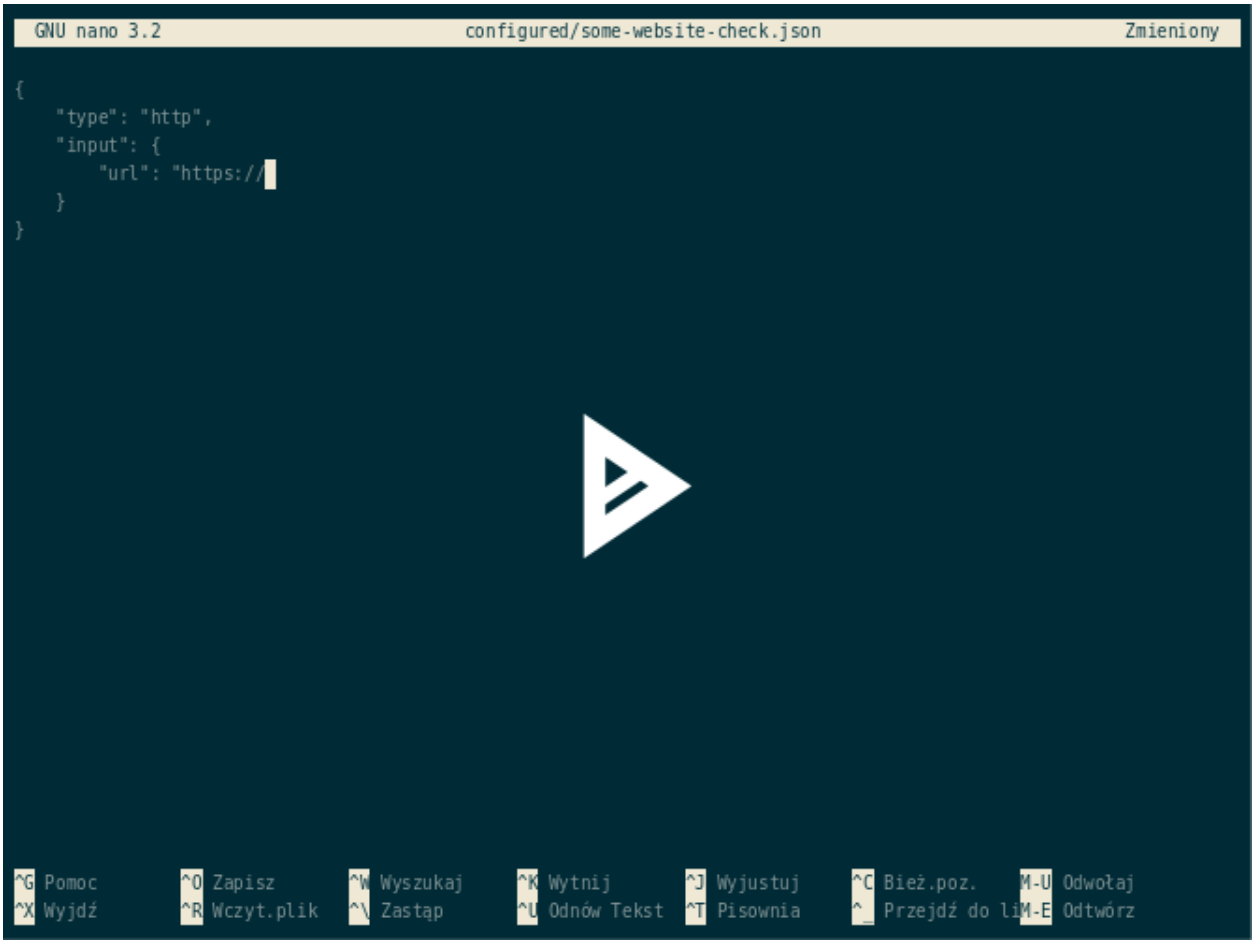

```
{
    "checks": {
        "disk-space": {
            "ident": "disk-space=True",
            "output": "There is 350.8GB disk space at '/', nothing to worry about,␣
↪defined minimum is 15GB\n",
            "status": true
        },
        "docker-health": {
            "ident": "docker-health=True",
            "output": "Docker daemon reports that there is no 'unhealthy' service␣
↪running in '' space\n",
            "status": true
        },
        "minio": {
            "ident": "minio=True",
            "output": "",
            "status": true
        },
        "replication-running": {
            "ident": "replication-running=True",
            "output": "Replica seems to be in good state\n",
            "status": true
        },
        "storage-synchronization": {
            "ident": "storage-synchronization=True",
            "output": "Storage synchronization looks fine\n",
            "status": true
        }
    },
    "global_status": true
}
```

CHAPTER 1

Quick start

To monitor applications and the infrastructure parts you need to configure **checks**. A configured check is a json file that defines a method name (script to be used) and the input parameters. Each check is executed when your external monitoring software invokes the HTTP endpoint, or when you execute the shell command.

Infracheck can work as a HTTP endpoint responding with JSON, or as a console command.



1.1 1. Requirements

You need to install all requirements manually if you decide not to use a docker container.

Requirements:

- Python 3.7+
- OpenSSH Client
- sshpass (for SSH checks)
- whois (for domain checks)
- mysql-client (for MySQL checks)
- postgresql-client (for PostgreSQL checks)
- docker client (for Docker checks)
- curl

Python package requirements:

```
ovh >= 0.5.0, < 1.1
psutil >= 5.7.2, < 6
```

(continues on next page)

(continued from previous page)

```
psycopg2-binary >= 2.8.4, < 3
python-dateutil >= 2.8.1, < 3
pytz >= 2019.3
six >= 1.15.0, < 2
tornado >= 5.1.1, < 7
whois >= 0.9.13, < 1
influxdb >= 5.3.1, < 6
msgpack >= 1.0, < 2
rkd>=2.3.3, <3
rkd-python>=2.3.3, <3
docker >= 5
croniter >= 1.0.13, < 1.4
```

1.2 2. Structure

You need to create a **project structure** from following template:

```
- checks/
    - http
    - smtp
    - port
- configured/
    - redis
    - duckduckgo_http
    - smtp_is_alive
```

In **checks** there should be scripts that will take parameters as environment variables, process and give results. For simpler cases you may not need to define any scripts, just configure pre-defined ones.

configured should contain your actual use cases, for example "duckduckgo_http" from above example could use "http" check with url "https://duckduckgo.com" as a parameter.

1.3 3. Configuring a first check

Let's assume that we need to check if a page contains given keyword, and does not contain another defined one. Following check will use **curl** to fetch page content.

Test cases:

- If page will not load, then THE CHECK RETURNS FAILURE
- If page contains "Server error", then THE CHECK RETURNS FAILURE
- If page will not contain keyword "iwa", then THE CHECK RETURNS FAILURE
- If page loads properly and contains "iwa" keyword, then THE CHECK RETURNS SUCCESS

```
{
    "type": "http",
    "input": {
        "url": "http://iwa-ait.org",
        "expect_keyword": "iwa",
        "not_expect_keyword": "Server error"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Hint: You can pass environment variables in parameters - see: *Templating* section.

1.4 4. Running checks

With Docker

You can use a ready-to-use docker image **quay.io/riotkit/infracheck** or **quay.io/riotkit/infracheck** for ARM. Please check the list of available versions.

The image will by default expose a HTTP endpoint.

```
# create directory structure that will be present in "/data" inside container (see␣
↪one of previous steps about the structure)
mkdir checks configured

sudo docker run --name infracheck -p 8000:8000 -v $(pwd):/data -d --rm quay.io/
↪riotkit/infracheck:v2.0-x86_64 \
    --directory=/data --server-path-prefix=/your-secret-code-there

# now test it
curl http://localhost:8000/your-secret-code-there/
```

List of supported environment variables:

- REFRESH_TIME=120
- CHECK_TIMEOUT=120
- WAIT_TIME=0

Without Docker

```
git clone https://github.com/riotkit-org/infracheck
cd infracheck
rkd :install

# run checks in the shell
infracheck --directory=/your-project-directory-path-there --no-server

# run the application with webserver and background worker
infracheck --directory=/your-project-directory-path-there --server-port=8000 --
↪refresh-time=120 --log-level=info
```

Using PIP

```
sudo pip install infracheck

# run checks in the shell
infracheck --directory=/your-project-directory-path-there --no-server

# run the application with webserver and background worker
infracheck --directory=/your-project-directory-path-there --server-port=8000 --
↪refresh-time=120 --log-level=info
```

1.5 Advanced

Setting timeout per check: Set `INFRACHECK_TIMEOUT` environment variable in json file to adjust timeout for given check.

CHAPTER 2

Hooks

After each execution of your checks there is a possibility to execute some commands.

Example:

```
{
    "type": "disk-space",
    "input": {
        "dir": "/",
        "min_req_space": "6"
    },
    "hooks": {
        "on_each_up": [
            "rm -f /tmp/maintenance.html"
        ],
        "on_each_down": [
            "echo \"Site under maintenance\" > /tmp/maintenance.html"
        ]
    }
}
```

Example above will delete a */tmp/maintenance.html* file when disk space will be at acceptable level. If there will be no enough disk space, then "Site under maintenance" will be written to the /tmp/maintenance.html With this practical example you can add a rule to your NGINX/Apache gateway to show a maintenance page, when a file is present.

CHAPTER 3

Predefined check types reference

Infracheck comes by default with some standard checks, there is a list of them:

3.1 http

Performs a HTTP call using curl.

Example:

```
{
    "type": "http",
    "input": {
        "url": "http://iwa-ait.org",
        "expect_keyword": "iwa",
        "not_expect_keyword": "Server error"
    }
}
```

Parameters:

- url
- expect_keyword
- not_expect_keyword

3.2 rkd://

Infracheck can execute RiotKit-Do tasks. RKD is a task executor, similar to Makefile or Gradle. It's essential feature is a possibility to load tasks from PyPI (Python packages).

Using RKD you can write a Python class, version and release it to PyPI with a list of dependencies, and install in any place with PIP. A packaged task can require extra dependencies you do not want always to install eg. MySQL, PostgreSQL, Redis or other clients you want to selectively install on your Infracheck instances.

More information on how to write RKD tasks: in RiotKit-Do's documentation

```
{
    "type": "rkd://rkd.standardlib.shell:sh",
    "input": {
        "-c": "ps aux |grep X11"
    }
}
```

```
{
    "type": "rkd://my_rkd_check:mysql:temporary-table-size-check",
    "input": {
        "--max": "100000",
        "--host: "localhost",
        "--port": 3306,
        "--user": "infracheck",
        "--password": "${TEMP_TABLE_SIZE_CHECK_PASSWORD}"
    }
}
```

3.3 dir-present

Checks whenever a directory exists.

Parameters:

- dir

3.4 file-present

Checks if file is present.

Parameters:

- file_path

3.5 docker-health

Checks if containers are healthy.

Parameters:

- docker_env_name (it's a prefix, to check only containers that names begins with this - idea of docker-compose)

3.6 port-open

Checks if the port is open.

Parameters:

- po_host
- po_port (in seconds)
- po_timeout (in seconds)

3.7 replication-running

Checks if the MySQL replication is in good state. Works with Docker only.

Parameters:

- container
- mysql_root_password

3.8 free-ram

Monitors RAM memory usage to notify that a maximum percent of memory was used.

Parameters:

- max_ram_percentage (in percents eg. 80)

3.9 domain-expiration

Check if the domain is close to expiration date or if it is already expired.

Notice: Multiple usage of this check can cause a "request limit exceeded" error to happen

Warning: *Due to limits per IP on whois usage we recommend to strongly cache the health check ex. 1-2 days cache, and in case of checking multiple domains to use feature called "wait time" to sleep between checks, to not send too many requests a once*

Parameters:

- domain (domain name)
- alert_days_before (number of days before expiration date to start alerting)

3.10 disk-space

Monitors disk space.

Parameters:

- min_req_space (in gigabytes)
- dir (path)

Example JSON:

```
{
    "type": "disk-space",
    "input": {
        "dir": "/",
        "min_req_space": "6"
    }
}
```

3.11 ovh-expiration

Checks if a VPS is not expired. Grab credentials at https://api.ovh.com/createToken/index.cgi

Required privileges on OVH API: "GET /vps*"

Parameters:

- endpoint (ex. ovh-eu)
- app_key
- app_secret
- app_consumer_key
- service_name (ex. somevps.ovh.net)
- days_to_alert (ex. 30 for 30 days)

Example JSON:

```
{
    "type": "ovh-expiration",
    "input": {
        "endpoint": "ovh-eu",
        "app_key": "xyyyyyyyyyyyyzz",
        "app_secret": "xyxyxyxyyxyxyxyxyxyxxyyxyxyxyxy",
        "app_consumer_key": "xyxyyxyxyxyxyxyxyyyxyxyxyxyxy",
        "service_name": "vps12345678.ovh.net",
        "days_to_alert": 5
    }
}
```

3.12 ssh-fingerprint

Verifies if remote host fingerprint matches. Helps detecting man-in-the-middle and server takeover attacks.

Parameters:

- expected_fingerprint (example: zsp.net.pl ssh-rsa SOMESOMESOMESOMESOMEKEYHERE)
- method (default: rsa)
- host (example: zsp.net.pl)
- port (example: 22)

3.13 ssh-files-checksum

Calls remote process using SSH and expects: the listed files and checksums will be matching

Parameters:

- user (default: root)
- host
- port (default: 22)
- private_key
- password
- ssh_bin (default: ssh)
- sshpass_bin (default: sshpass)
- ssh_opts (example: -o StrictHostKeyChecking=no)
- known_hosts_file (default: ~/.ssh/known_hosts)
- command (default: uname -a)
- timeout: (default: 15, unit: seconds)
- method (default: sha256sum)
- expects (json dict, example: {"/usr/bin/bahub": "d6e85b50756a08e24c1d46f07b68e288c9e7e565fd662a15baca214f576c34be"})

3.14 ssh-command

Calls remote process using SSH and expects: exit code, keywords in the output

Parameters:

- user (default: root)
- host
- port (default: 22)
- private_key
- password
- ssh_bin (default: ssh)
- sshpass_bin (default: sshpass)
- ssh_opts (example: -o StrictHostKeyChecking=no)
- known_hosts_file (default: ~/.ssh/known_hosts)
- command (default: uname -a)
- timeout: (default: 15, unit: seconds)
- expected_keywords (Keywords expected to be in stdout/stderr. Separated by ";")
- unexpected_keywords (Keywords not expected to be present in stdout/stderr. Separated by ";")
- expected_exit_code (default: 0)

3.15 reminder

Reminds about the recurring date. Example: To extend validity of your hosting account

Parameters:

- ref_date (example: 2019-05-01 for a 1th of May 2019)
- each (values: week; month; year, default: year)
- alert_days_before (default: 5, the health check will be red when there will be 5 days before)

3.16 load-average-auto

Checks if the load average is not more than 100%

Parameters:

- maximum_above (unit: processor cores, default: 0.5 - half of a core)
- timing (default: 15. The load average time: 1, 5, 15)

3.17 load-average

Checks if the load average is not below specified number

Parameters:

- max_load (unit: processor cores, example: 5.0, default: 1)
- timing (default: 15. The load average time: 1, 5, 15)

3.18 swap-usage-max-percent

Defines maximum percentage of allowed swap usage

Parameters:

- max_allowed_percentage (default: 0.0)

3.19 influxdb-query

Queries an InfluxDB database and compares results.

Parameters:

- host
- port (default: 8086)
- user
- password
- database

- query
- expected: A json serialized result (not pretty formatted)

Example of JSON serialized result for query 'select value from cpu_load_short;':

```
[
    [
        {"time": "2009-11-10T23:00:10Z", "value": 10.64},
        {"time": "2009-11-10T23:00:20Z", "value": 20.64},
        {"time": "2009-11-10T23:00:30Z", "value": 30.64},
        {"time": "2009-11-10T23:00:40Z", "value": 40.64}
    ]
]
```

3.20 postgres

Uses *pg_isready* tool to verify if PostgreSQL is up and ready to connect.

Parameters:

- pg_host (hostname or socket path, defaults to "localhost" which will use local unix socket, use IP address eg. 127.0.0.1 to connect via tcp)
- pg_port (port, defaults to 5432)
- pg_db_name (database name to connect to, defaults to "postgres")
- pg_user (username, defaults to "postgres")
- pg_conn_timeout (defaults to 15 which means 15 seconds)

3.21 postgres-primary-streaming-status

Verifies if local PostgreSQL instance is currently serving WALs to a specified replica. The SQL command that is validated: *select * from pg_stat_replication;*

Parameters:

- pg_host (hostname or socket path, defaults to "localhost" which will use local unix socket, use IP address eg. 127.0.0.1 to connect via tcp)
- pg_port (port, defaults to 5432)
- pg_db_name (database name to connect to, defaults to "postgres")
- pg_user (username, defaults to "postgres")
- pg_password
- pg_conn_timeout (defaults to 15 which means 15 seconds)
- expected_status (defaults to "streaming")
- expected_replication_user: Expected user that will be used for replication connection (defaults to "replication")

3.22 postgres-replica-status

Checks if local PostgreSQL server acts as a replication server, by validating the list of active wal receivers. The SQL command that is validated: *select * from pg_stat_wal_receiver;*

Parameters:

- pg_host (hostname or socket path, defaults to "localhost" which will use local unix socket, use IP address eg. 127.0.0.1 to connect via tcp)
- pg_port (port, defaults to 5432)
- pg_db_name (database name to connect to, defaults to "postgres")
- pg_user (username, defaults to "postgres")
- pg_password
- pg_conn_timeout (defaults to 15 which means 15 seconds)
- expected_status (defaults to "streaming")
- expected_replication_user: Expected user that will be used for replication connection (defaults to "replication")

3.23 docker-container-log

Searches docker container logs for matching given regular expression.

Parameters:

- container: Docker container name
- regexp: Regular expression
- max_lines: Number of last lines to check (defaults to 5)
- since_seconds: Get only logs since this time (eg. last 5 minutes = 5 * 60 = 300) (defaults to 300)
- present: Boolean, if the string should be present in the output or not

3.24 smtp_credentials_check.py

Verifies connection, TLS certificate and credentials to a SMTP server by doing a ping + authorization try.

Parameters:

- smtp_host (example: bakunin.example.org)
- smtp_port (example: 25)
- smtp_user (example: noreply@example.org)
- smtp_password (example: bakunin-1936)
- smtp_encryption (example: starttls. Values: "", "ssl", "starttls")
- smtp_timeout (default: 30, unit: seconds)

3.25 tls

TLS/SSL certificate expiration validation

Parameters:

- domain: TLS certificate domain for which the certificate was created
- host: IP address or DNS hostname from which the certificate should be downloaded (defaults to domain value)
- port: Port (defaults to 443)
- alert_days_before: Number of days before expiration date to start alerting (defaults to 3)

3.26 tls-docker-network

Automated TLS certificate verification for docker-based flows like docker-gen. Scans list of docker containers basing on a label or environment variable that contains a domain name.

Parameters:

- parameter_type: Label or environment variable
- parameter_name: Name of the label or environment variable
- alert_days_before: Number of days before expiration date to start alerting (defaults to 3)
- docker_host: (Optional) The URL to the Docker host.
- docker_tls_verify: (Optional) Verify the host against a CA certificate.
- docker_cert_path: (Optional) A path to a directory containing TLS certificates to use when connecting to the Docker host
- debug: (Optional) Debugging mode

CHAPTER 4

Check configuration reference

```
{
    "type": "http",
    "description": "IWA-AIT check",
    "results_cache_time": 300,
    "input": {
        "url": "http://iwa-ait.org",
        "expect_keyword": "iwa",
        "not_expect_keyword": "Server error"
    },
    "hooks": {
        "on_each_up": [
            "rm -f /var/www/maintenance.html"
        ],
        "on_each_down": [
            "echo \"Site under maintenance\" > /var/www/maintenance.html"
        ]
    },
    "quiet_periods": [
        {"starts": "30 00 * * *", "duration": 60}
    ]
}
```

4.1 type

Name of the binary/script file placed in the "checks" directory. At first will look at path specified by "–directory" CLI parameter, then will fallback to Infracheck internal check library.

Example values:

- disk-space
- load-average

- http
- smtp_credentials_check.py

4.2 description

Optional text field, there can be left a note for other administrators to exchange knowledge in a quick way in case of a failure.

4.3 results_cache_time

How long the check result should be kept in cache (in seconds)

4.4 input

Parameters passed to the binary/script file (chosen in "type" field). Case insensitive, everything is converted to UPPERCASE and passed as environment variables.

Notice: *Environment variables and internal variables can be injected using templating feature - check Templating*

4.5 hooks

(Optional) Execute shell commands on given events.

- on_each_up: Everytime the check is OK
- on_each_down: Everytime the check is FAILING

4.6 quiet_periods

(Optional) Defines time, when the check results should be ignored. For example setting "30 00 * * *" and 60m duration will result in ignoring check failure at 00:30 everyday for 60 minutes - till 01:30

CHAPTER 5

Templating

In order to increase the security there is a *simple templating* mechanism that allows to inject variables into parameters you define that are passed to the checks.

Example:

```
{
    "type": "ssh-command",
    "input": {
        "user": "thesecurityman",
        "host": "iwa-ait.org",
        "port": 6200,
        "password": "${ENV.IWA_SECURITY_MAN_PASSWD}",
        "command": "/usr/bin/some-security-check --is-secure",
        "expected_exit_code": 0,
        "timeout": 30
    }
}
```

5.1 Reference table

Pattern	Example	Description
${ENV.*}	${ENV.USER}	Injects an environment variable from the host
${checkName}	http	Name of the currently executed check
${date}	2019-10-31T07:53:45.380307	Current date and time

5.2 Example strategy of deploying passwords with Docker Compose and Ansible

1. Encrypt your passwords with ansible-vault

2. Decrypt them during deployment into *.env* on target machine for docker-compose
3. In docker-compose service definition pass variable explicitly from the *.env* file

```
environment:
    # variables in checks
    - IWA_SECURITY_MAN_PASSWD=${IWA_SECURITY_MAN_PASSWD}
```

CHAPTER 6

Writing custom checks

Infracheck provides very basic scripts for health checking, you may probably want to write your own. It's really simple.

1. "check" scripts are in **"checks" directory** of your project structure, here you can add a **new check script**
2. Your script needs to take **uppercase environment variables as input**
3. It is considered a good practice to validate environment variables presence in scripts
4. **Your script needs to return a valid exit code when:**
    - Any of environment variables is missing or has invalid value
    - The check fails
    - The check success

That's all!

A few examples:

```
#!/bin/bash

#
# Directory presence check
#
# @author Krzysztof Wesołowski
# @url https://iwa-ait.org
#

if [[ ! "${DIR}" ]]; then
    echo "DIR parameter is missing"
    exit 1
fi

if [[ ! -d "${DIR}" ]]; then
    echo "Failed asserting that directory at '${DIR}' is present"
```

(continues on next page)

(continued from previous page)

```
    exit 1
fi

echo "'${DIR}' directory is present"
exit 0
```

```
#!/usr/bin/env python3

"""
<sphinx>
load-average
------------

Checks if the load average is not below specified number

Parameters:

- max_load (unit: processor cores, example: 5.0, default: 1)
- timing (default: 15. The load average time: 1, 5, 15)
</sphinx>
"""

import os
import sys
import inspect

path = os.path.dirname(os.path.abspath(inspect.getfile(inspect.currentframe()))) + '/./../'
sys.path.insert(0, path)

from infracheck.infracheck.checklib.loadavg import BaseLoadAverageCheck


class LoadAverageAuto(BaseLoadAverageCheck):
    def main(self, timing: str, max_load: float):
        current_load_average = self.get_load_average(timing)

        if current_load_average > max_load:
            return False, "Load {:.2f} exceeds allowed max. load of {:.2f}. Current load: {:s}".format(
                current_load_average, max_load, self.get_complete_avg()
            )

        return True, "Load at level of {:.2f} is ok, current load: {:s}".format(
            current_load_average, self.get_complete_avg())


if __name__ == '__main__':
    app = LoadAverageAuto()
    status, message = app.main(
        timing=os.getenv('TIMING', '15'),
        max_load=float(os.getenv('MAX_LOAD', 1))
    )

    print(message)
    sys.exit(0 if status else 1)

```

CHAPTER 7

Cache and freshness

It can be harmful to the server to run all checks on each HTTP endpoint call, so the application is running them periodically every X seconds specified by **–refresh-time** switch or **REFRESH_TIME** environment variable (in docker)

7.1 Refresh time

If you use an official docker image, then you can set an environment variable.

Example: check once a day (good for domains whois check).

```
REFRESH_TIME=86400
```

From CLI you can set **–refresh-time=86400**

7.2 Wait time

Some checks could call external APIs, those can have limits. A good example is a *domain-expiration* check which is using whois. Set **–wait=60** to for example wait 60 seconds before each check - where check is a single entry on the list of checks.

7.3 Customizing check freshness time per check

Beside the global setting of **refresh time** there could be a per-check setting called "results_cache_time".

Example of caching the check result for at least 300 seconds

```
{
    "type": "swap-usage-max-percent",
    "results_cache_time": "300",
```

(continues on next page)

(continued from previous page)

```
    "input": {
        "max_allowed_percentage": 0
    }
}
```

CHAPTER 8

From authors

Project was started as a part of RiotKit initiative, for the needs of grassroot organizations such as:

- Fighting for better working conditions syndicalist (International Workers Association for example)
- Tenants rights organizations
- Various grassroot organizations that are helping people to organize themselves without authority

RiotKit Collective