pimp-my-box Documentation

Release 0.1

PyroScope Project

Jun 07, 2021

CONTENTS

The playbooks in the `pimp-my-box` repository will install `rTorrent-PS`, `pyrocore`, and related software onto any remote dedicated server or VPS with `root` access, running Debian or a Debian-like OS. Read *Overview* to learn more.

To get in contact and share your experiences with other users of PyroScope, join the rtorrent-community channel pyroscope-tools on Gitter.

This is also the way to resolve any problems with or questions about your configuration and software installation. *Always* look into the *Trouble-Shooting Guide* as a first measure, which is often the fastest way to get back to a working system. That guide also explains how to efficiently report your problem when you cannot fix it yourself.

Warning: If you update an existing installation, you need *rTorrent-PS 1.1* or later for the contained configuration to work. If you use an older version, call

```
echo >>~rtorrent/rtorrent/rtorrent.d/.rcignore 05-rt-ps-columns.rc
```

when you get errors about `ui.column.render` on startup. Or simply follow *Upgrade the rTorrent-PS Version*.

CHAPTER

ONE

FULL CONTENTS

1.1 Overview

The playbooks in the `pimp-my-box` repository will install `rTorrent-PS`, `pyrocore`, and related software onto any remote dedicated server or VPS with `root` access, running Debian or a Debian-like OS. And nobody prevents you from treating your own workstation as 'remote', so it can be used for a local install too.

`Ansible` is used to describe the installation process. It is a tool that allows you to perform a complete setup remotely on one or any number of *target hosts*, from the comfort of your own workstation. The setup is described in so called *playbooks*, before executing them you just have to add a few values like the name of your target host.

This is in many ways superior to the usual *"call a bash script to set up things once and never be able to update them again"*, since you can run this setup repeatedly to either fix problems, or to install upgrades and new features added to this repository.

Additionally, if your host crashes and cannot be repaired for some reason, restoring the software and its configuration is a breeze and typically done in under an hour. You just need proper backups of crucial data, like the rTorrent session directory. The same works for moving from one hosting provider to another, just copy your data via rsync to your new host, to an identical setup.

The playbooks contained in here install the following components:

- Security hardening of your server.
- rTorrent-PS with UI enhancements, colorization, and some added features.
- PyroScope command line tools (pyrocore) for rTorrent automation.

Optionally:

- FlexGet, the best feed reader and download automation tool there is.
- ruTorrent web UI, served by Nginx over HTTPS and run by PHP FPM.

Each includes a default configuration, so you end up with a fully working system.

The Ansible playbooks and related commands have been tested on Ubuntu Trusty, Xenial, and Bionic, but should run equally well on Debian Stretch and Buster, including RasPi variants – the recommended distribution is Ubuntu Server LTS 64bit (i.e. release 18.04 Bionic at the time of this writing). They should work on other platforms too, especially when they're Debian derivatives, but you might have to make some modifications.

Files are mostly installed into the user accounts `rtorrent` and `rutorrent`, and only a few global configuration files are affected. If you run this against a host with an existing installation, make sure that there are no conflicts (Ansible's `--check` and `--diff` options come in very handy here).

To get in contact and share your experiences with other users of PyroScope, join the rtorrent-community channel pyroscope-tools on Gitter.

This is also the way to resolve any problems with or questions about your configuration and software installation. *Always* look into the *Trouble-Shooting Guide* as a first measure, which is often the fastest way to get back to a working system. That guide also explains how to efficiently report your problem when you cannot fix it yourself.

1.2 Basic Installation

Here's the steps you need to follow to get a working installation on your target host. This might seem like an arduous process, but if you're accustomed to a *Linux* command prompt and ideally also *Ansible*, it boils down to these steps:

- Create your working directory.
- Call the Ansible installer script, if you don't have it available yet.
- Enable a SSH sudo account on your deployment target.
- Create and edit two Ansible config files.
- Run the playbook and wait a bit.
- Enjoy your working and secure seedbox.

And once you got it working, moving to or adding another machine is easy and almost no work, just add that host and run the playbooks for it.

Note that this cannot be an Ansible or Linux shell 101, so if these topics are new for you refer to the usual sources like The Debian Administrator's Handbook, The Linux Command Line, The Art of Command Line, and the Ansible Documentation.

1.2.1 Checking Out the Code

To work with the playbooks, you need a local copy of them. Unsurprisingly, you also need `git` installed for this, to create that local copy (a/k/a *clone* the repository).

Executing these commands *on your workstation* takes care of that:

```
which git || sudo apt-get install git
mkdir -p ~/src; cd $_
git clone "https://github.com/pyroscope/pimp-my-box.git"
cd "pimp-my-box"
```

1.2.2 Installing Ansible

Ansible **must** be installed on the workstation from where you control your target hosts. This can also be the target host itself, if you don't have a local computer running on Linux, Mac OSX, or Windows 10 with WSL installed. In that case, use a personal account on that machine, or create an `ansible` one – any part of the documentation that refers to 'the workstation' then means that account.

See the Ansible Installation Documentation for how to install it using the package manager of your platform. Make sure you get the right version that way, the playbooks are tested using Ansible *2.9.5*. Read *Update to Ansible2 on Your Workstation* when you have an existing workdir from before February 2020.

The *recommended* way to install Ansible is to put it into your home directory. The following commands just require Python 3.5+ to be installed on your system. The installation is easy to get rid of (everything is contained within a single directory). When you have no `~/.ansible.cfg` yet (which you very likely do not), one is added.

Enter / copy+paste this command into a shell prompt on your workstation, within the 'pimp-my-box' directory!

```
./scripts/install_ansible.sh
```

1.2.3 Providing SSH Access for Ansible

For a dedicated server, the first step is to create an account *Ansible* can use to perform its work. Log into your server as `root` and call these commands:

```
account=setup
groupadd $account
useradd -g $account -G $account,users -c "Ansible remote user" \
        -s /bin/bash --create-home $account
eval chmod 0750 ~$account
passwd -l $account
```

Calling the following command as `root` on the *target host* will grant password-less sudo to the new account:

```
# Give password-less sudo permissions to the "setup" user
echo >/etc/sudoers.d/setup "setup ALL=(ALL) NOPASSWD:ALL"
```

In case you prefer password-protected sudo, leave out the `NOPASSWD:`, and also set a password using `passwd setup`.

The `setup` account must allow login using the `id_rsa` key, or another key you create on your *workstation*. See here for establishing working SSH access based on a pubkey login, if you've never done that before.

Finally, the last snippet of SSH configuration goes into `~/.ssh/config` of your *workstation* account, add these lines providing details on how to connect to your target host via SSH (and replace the text in `ALL_CAPS` by the correct value):

```
Host my-box
    HostName IP_ADDRESS_OR_DOMAIN_OF_TARGET
    Port 22
    User setup
    IdentityFile ~/.ssh/id_rsa
    IdentitiesOnly yes
```

Now to test that you did everything right, call the below `ssh` command on your *workstation*, and verify that you get the output as shown:

```
$ ssh my-box "sudo id"
uid=0(root) gid=0(root) groups=0(root)
```

In case you're asked for a password, enter the one you've set on the `setup` account.

1.2.4 Setting Up Your Environment

Now with Ansible installed and able to connect via SSH, you next need to configure the target host (by default named `my-box`) and its specific attributes (the so-called *host vars*). There is an example in host_vars/rpi/main.yml for a default *Raspberry Pi* setup which is used as a template.

To create the necessary files, call this command:

```
./scripts/add_host.sh
```

If you already have an Ansible inventory (i.e. `hosts` file), your configured editor will open it – else a suitable default is created. Make sure you add your target's name to the `[box]` group, if it's missing.

Next the editor will open with `main.yml`, fill in the values as described in the first few lines of the file. In a final step, you need to enter the `sudo` password of your target server.

Afterwards, you have these files in your working directory: `hosts`, `host_vars/my-box/main.yml`, and `host_vars/my-box/secrets.yml`. If you don't understand what is done here, read the Ansible documentation again, specifically the "Getting Started" page.

Now we can check your setup and that Ansible is able to connect to the target and do its job there. For this, call the command as shown after the `$`, and it should print what OS you have installed on the target(s), like shown in the example.

```
$ ansible box -i hosts -m setup -a "filter=*distribution*"
my-box | success >> {
    "ansible_facts": {
        "ansible_distribution": "Ubuntu",
        "ansible_distribution_major_version": "14",
        "ansible_distribution_release": "trusty",
        "ansible_distribution_version": "14.04"
    },
    "changed": false
}
```

If anything goes wrong, add `-vvvv` to the `ansible` command for more diagnostics, and also check your `~/.ssh/config` and the Ansible connection settings in your `host_vars`. If it's a connection problem, try to directly call `ssh -vvvv my-box` and if that succeeds, also make sure you can become `root` via `sudo su -`. If not, read the resources linked at *the start of this chapter*, and especially the SSH Essentials.

1.2.5 Using the System Python Interpreter

By default, Python 2.7.13 is installed because that version handles SSL connections according to current security standards; the version installed in your system often does not. This has an impact on e.g. FlexGet's handling of `https` feeds.

If you want to use the system's Python interpreter, add these variables to your host vars:

```
pyenv_enabled: false
python_bin: /usr/bin/python2
venv_bin: /usr/bin/virtualenv
```

Doing so is recommended on *Xenial* (has 2.7.12), *Jessie* (2.7.9), or *Stretch* (2.7.13).

1.2.6 Running the Playbook

To execute the playbook, call `ansible-playbook -i hosts site.yml`. The initial installation will take a while, so be patient.

If your Linux release isn't supported with a pre-built package, you'll see a message like the following:

```
WARNING - No DEB package URL defined for '‹platform›', you need to install /opt/
↪rtorrent manually!
```

In that case, compile a binary yourself. If you want to run a *rTorrent-PS* version that is not yet released to GitHub Releases, do the same.

If you added more than one host into the `box` group and want to only address one of them, use `ansible-playbook -i hosts -l ‹hostname› site.yml`. Add (multiple) `-v` to get more detailed information on what each task does.

1.2.7 Starting rTorrent

As mentioned before, after successfully running the Ansible playbook, a fully configured setup is found on the target. So to start rTorrent, log in as the `rtorrent` user and start this command:

```
tmux -2u new -n rT-PS -s rtorrent "~/rtorrent/start; exec bash"
```

To detach from this session (meaning rTorrent continues to run), press `Ctrl-a` followed by `d`.

If you get `rtorrent: command not found` when calling above `tmux` command, then a pre-built Debian package is not available for your OS distribution and you need to build from source (see previous section). You can check explicitly with the following command:

```
$ dpkg -l rtorrent-ps
dpkg-query: no packages found matching rtorrent-ps
```

1.2.8 Changing Configuration Defaults

Customizing Your Setup

A good way to provide customizations is writing your own playbooks. Create a separate project in your own git repository. In that project, you can provide your versions of existing files, add your own helper scripts, and so on. Model it after this repository, and consult the *Ansible* documentation. You can reuse your inventory, by passing `-i ../pimp-by-box/hosts` to the playbook calls, or by setting the `ANSIBLE_INVENTORY` environment variable.

As described in this and the following sections, some key config files are designed to be replaced in this way. Just be aware that once you copy them, you also have to manage them yourself, and merge with changes made to the master in this repo!

Handling Top-Level Config Files

Once created, the file `rtorrent.rc` is only overwritten when you provide `-e force_cfg=yes` on the Ansible command line. This gives you the opportunity to easily refresh the main configuration in `rtorrent.rc` from this repository. But it *also* gives you the option to provide your own custom version by other means, e.g. your own additional playbook, without having that version constantly overwritten.

However, additional files in `rtorrent.d/` or the `_rtlocal.rc` include file are the intended places to make customizations, so it's advisable to always add `-e force_cfg=yes` on updates.

The `_rtlocal.rc` file, which is included by `rtorrent.rc` *after* the standard configuration includes in `rtorrent.d/`, is never overwritten. So it's easy and safe to provide your own version of `_rtlocal.rc` from a custom playbook. Or apply customizations manually, by editing `~rtorrent/rtorrent/_rtlocal.rc` – these will not be reverted by updating from the repository.

A typical use of `_rtlocal.rc` is to change how long log files are kept uncompressed, here the default of 2 days is reduced to just one:

```
pyro.log_archival.days.set = 1
```

Adding Drop-In Files

Another way to customize rTorrent is to use the `~/rtorrent/rtorrent.d` directory. Just place any file with a `.rc` extension there, and it will be loaded on the next restart. This is ideally suited for custom playbooks, which can just add new files to extend the default configuration.

That directory also contains most of the extra rTorrent configuration that comes with `pimp-my-box`. For example, by default terminating rTorrent via `^Q` gets disabled in the `disable-control-q.rc` file, replacing it by `^X q=`, which you won't type by accident.

To restore the rTorrent default, run this command as the `rtorrent` user (or put the line into that file via *Ansible*):

```
echo >>~/rtorrent/rtorrent.d/.rcignore "disable-control-q.rc"
```

Then restart rTorrent.

Optional Applications & More

See *Optional Features* on how to activate add-ons like ruTorrent, and *Advanced Topics* for more details about the box installation and its features.

1.3 Optional Features

1.3.1 Enabling Optional Applications

To activate the optional applications, add these settings to your `host_vars`:

- `flexget_enabled: yes` for *FlexGet*.
- `rutorrent_enabled: yes` for *ruTorrent*.

Read the following sections for details.

1.3.2 Installing FlexGet

After setting `flexget_enabled: yes`, run the playbook again.

FlexGet is just installed ready to be used, for full operation a configuration file located in `~/.config/flexget/config.yml` must be added (see the FlexGet cookbook). A cronjob is provided too (called every 11 minutes), but only starts to actually call FlexGet *after* you add that configuration file.

Look into the files `~/.config/flexget/flexget.log` and `~/.config/flexget/flexget-cron.log` to diagnose any problems.

Hint: Running FlexGet with Python 3

If your target host has Python 3.6+ installed (i.e. runs Bionic or Buster), you can use that for the FlexGet venv by changing `venv_bin` in your `host_vars`:

```
venv_bin: "python3 -m venv"
```

Note that the latest versions of FlexGet *require* Python 3, i.e. you'll get an older release if you stick to Python 2.

On older releases of Ubuntu (Xenial), you can install Python 3.7 or higher from the DeadSnakes PPA. For your convenience, the basic installation on Ubuntu already adds Python 3.7 from there.

Make sure to include the minor version in your configuration then:

```
venv_bin: "python3.7 -m venv"
```

Also, check *beforehand* if the `ssl` module is supported:

```
python3.7 -m ssl
```

If you get an error message that `_ssl` is not available, you cannot use that Python build for `https` RSS feeds.

1.3.3 Installing and Updating ruTorrent

The ruTorrent web UI is an optional add-on, and you have to activate it by setting `rutorrent_enabled` to `yes` and providing a `rutorrent_www_pass` value, usually in your `host_vars/my-box/main.yml` and `host_vars/my-box/secrets.yml` files, respectively. Then run the playbook again.

Alternatively to the self-signed certificate that is created for Nginx, you can also copy a certificate you got from other sources to the paths `/etc/nginx/ssl/cert.key` and `/etc/nginx/ssl/cert.pem`. See this blog post if you want *excessive* detail on secure HTTPS setups.

After the second run, ruTorrent is available at `https://my-box.example.com/rutorrent/` (use you own domain or IP in that URL).

To *update to a new version* of ruTorrent, first add the desired version as `rutorrent_version` to your variables – that version has to be available on GitHub Releases. Then move the old installation tree away:

```
cd ~rutorrent
mv ruTorrent-master _ruTorrent-master-$(date "+%Y-%m-%d-%H%M").bak
tar cfz _profile-$(date "+%Y-%m-%d-%H%M").bak profile
```

Finally, rerun the playbook to install the new version. In case anything goes wrong, you can move back that backup you made initially.

1.4 Advanced Topics

1.4.1 Activating Firewall Rules

If you want to set up firewall rules using the Uncomplicated Firewall (UFW) tool, then call the playbook using this command:

```
# See above regarding adding the '-l' option to select a single host
ansible-playbook -i hosts site.yml -t ufw -e ufw=true
```

This will install the `ufw` package if missing, and set up all rules needed by apps installed using this project. Note that activating the firewall is left as a manual task, since you can make a remote server pretty much unusable when SSH connections get disabled by accident – only a rescue mode or virtual console can help to avoid a full reinstall then, if you have no physical access to the machine.

So to activate the firewall rules, use this in a `root` shell on the *target host*:

```
egrep 'ssh|22' /lib/ufw/user.rules /etc/ufw/user.rules
# Make sure the output contains
#   ### tuple ### limit tcp 22 0.0.0.0/0 any 0.0.0.0/0 in
```

(continues on next page)

(continued from previous page)

```
# followed by 3 lines starting with '-A'.

ufw enable  # activate the firewall
ufw status verbose  # show all the settings
```

1.4.2 Using Ansible for Remote Management

The setup work to get *Ansible* controlling your machines is not just for installing things, you can also simplify daily management tasks.

First, define this function to make calls to `rtorrent` tools easier:

```
rtansible() {
    if test -z "$1"; then
        echo >&2 "usage: rtansible TARGET CMD [ARGS...]"
        return 1
    fi
    local target="${1:?Missing a group/host name as 1st arg}"; shift
    local cmd="${1:?Missing a remote command as 2nd arg}"; shift
    ansible "$target" -f4 -a "sudo -i -u rtorrent -- $cmd" "$@"
}
```

Then consider this example, which prints the number of items loaded into rTorrent, for all hosts in the `box` group of your inventory:

```
$ rtansible box \
    '~rtorrent/bin/rtxmlrpc view.size @/dev/null default' -o \
    | sort
my-box | success | rc=0 | (stdout) 42
my-box2 | success | rc=0 | (stdout) 123
```

Similarly, this shows the Linux release you're using:

```
$ rtansible box 'lsb_release -cs' -o | sort
my-box | CHANGED | rc=0 | (stdout) bionic
```

Another example is updating the `pyrocore` installation from git, like this:

```
rtansible box '~rtorrent/.local/pyroscope/update-to-head.sh'
rtansible box '~rtorrent/bin/pyroadmin --version' -o | sort
```

This is especially useful if you control more than one host.

1.4.3 Using the bash Download Completion Handler

The default configuration adds a *finished* event handler that calls the `~rtorrent/bin/_event.download.finished` script. That script in turn just calls any existing `_event.download.finished-*.sh` script, which allows you to easily add custom completion behaviour via your own playbooks.

The passed parameters are `hash`, `name`, and `base_path`; the completion handler ensures the session state is flushed, so you can confidently read the session files associated with the provided hash.

Be aware that you cannot call back into rTorrent via XML-RPC within an event handler, because that leads to a deadlock. If you need to do that, call your script directly using `execute.bg` or one of its variants.

Alternatively detach yourself from the process that rTorrent created for the event, so the event handler finishes as far as rTorrent is concerned.

In any of these cases, be aware that things run concurrently and can go horribly wrong, if you don't take care of race conditions and such. A command queue like nq can help here, by accepting commands than run in the backgropund, but strictly serialized in the order they are added.

Here is a non-trivial example that goes to `~/bin/_event.download.finished-jenkins.sh`, and triggers a Jenkins job for any completed item:

```
#! /bin/bash
#
# Called in rTorrent event handler

set -x

infohash="${1:?You MUST provide the infohash of the completed item!}"
url="http://localhost:8080/job/event.download.finished/build?delay=0sec"
json="$(python -c "import json; print json.dumps(dict(parameter=dict(name='INFOHASH',␣
↪value='$infohash')))")"

http --ignore-stdin --form POST "$url" token=C0mpl3t3 json="$json"
```

You need to add the related `event.download.finished` job and `rtorrent` user to Jenkins of course. The user's credentials must be added to `~rtorrent/.netrc`, like this:

```
machine localhost
    login rtorrent
    password YOUR_PWD
```

Make sure to call `chmod 0600 ~/.netrc` after creating the file.

To check that everything is working, download something and check the build history of your Jenkins job – if nothing seems to happen, look into `~/rtorrent/log/execute.log` to debug.

The fact that *Jenkins* runs in its own separate process means your job can make free use of `rtxmlrpc` and `rtcontrol` to change things in *rTorrent*.

1.4.4 Extending the Nginx Site

The main Nginx server configuration includes any `/etc/nginx/conf.d/rutorrent-*.include` files, so you can add your own locations in addition to the default `/rutorrent` one. The main configuration file is located at `/etc/nginx/sites-available/rutorrent`.

Use a `/etc/nginx/conf.d/upstream-*.conf` file in case you need to add your own `upstream` definitions.

1.4.5 Implementation Details

Location of Configuration Files

- `/home/rtorrent/rtorrent/rtorrent.rc` – Main *rTorrent* configuration file; to update it from this repository use `-e force_cfg=yes`, see *Basic Installation* for details.
- `/home/rtorrent/rtorrent/_rtlocal.rc` – *rTorrent* configuration include for custom modifications, this is *never* overwritten once it exists.
- `/home/rtorrent/.pyroscope/config.ini` – `pyrocore` main configuration.

- `/home/rtorrent/.pyroscope/config.py` – `pyrocore` custom field configuration.
- `/home/rtorrent/.config/flexget/config.yml` – *FlexGet* configuration.
- `/home/rutorrent/ruTorrent-master/conf/config.php` – *ruTorrent* configuration.
- `/home/rutorrent/profile/` – Dynamic data written by *ruTorrent*.
- `/etc/nginx/sites-available/rutorrent` – *NginX* configuration for the *ruTorrent* site.
- `/etc/php5/fpm/pool.d/rutorrent.conf` or `/etc/php/7.0/fpm/pool.d/rutorrent.conf` – PHP worker pool for *ruTorrent*.

Location of Installed Software

- `/home/rtorrent/.local/profile.d/` — Directory with shell scripts that get sourced in `~rtorrent/.bash_aliases`.
- `/home/rtorrent/.local/pyenv/` — Unless you chose to use the system's *Python*, the interpreter used to run `pyrocore` and `flexget` is installed here.
- `/home/rtorrent/.local/pyroscope` — Virtualenv for `pyrocore`.
- `/home/rtorrent/.local/flexget` — Virtualenv for `flexget`.
- `/home/rutorrent/ruTorrent-master` — *ruTorrent* code base.

Secure Communications

All internal RPC is done via Unix domain sockets.

- `/var/run/php-fpm-rutorrent.sock` — *NginX* sends requests to PHP using the *php-fpm* pool `rutorrent` via this socket; it's owned by `rutorrent` and belongs to the `www-data` group.
- `/var/torrent/.scgi_local` — The XMLRPC socket of rTorrent. It's group-writable and owned by `rtorrent.rtorrent`; ruTorrent talks directly to that socket (see issue #9 for problems with using /RPC2).

1.5 Trouble-Shooting Guide

1.5.1 Reporting Problems

If you have any trouble during *pimp-my-box* installation and configuration, or using any of the commands from the documentation, join the rtorrent-community channel pyroscope-tools on Gitter. You can also ask questions on platforms like Reddit or Stack Exchange.

If you are sure there is a bug, then open an issue on *GitHub*. Make sure that nobody else reported the same problem before you, there is a search box you can use (after the **Filter** button). Please note that the *GitHub* issue tracker is not a support platform, use the Gitter channel or Reddit for any questions, as mentioned above.

And ESR's golden oldie How To Ask Questions The Smart Way is still a most valuable resource, too.

Note: Please **describe your problem clearly**, and provide any pertinent information. What are the **version numbers** of software and OS? What did you do? What was the **unexpected result**? If things worked and 'suddenly' broke, **what did you change**?

In the chat, don't ask if somebody is there, just describe your problem. Eventually, someone will notice you – people *do* live in different time zones than you.

Put up any logs on 0bin or any other pastebin service, and **make sure you removed any personal information** you

don't want to be publically known. Copy the pastebin link into the chat window.

The following helps with querying your system environment, e.g. the version of Python and your OS.

1.5.2 Providing Diagnostic Information

Python Diagnostics

Execute the following command to be able to provide some information on your Python installation:

```
deactivate 2>/dev/null; /usr/bin/virtualenv --version; python <<'.'
import sys, os, time, pprint
pprint.pprint(dict(
    version=sys.version,
    prefix=sys.prefix,
    os_uc_names=os.path.supports_unicode_filenames,
    enc_def=sys.getdefaultencoding(),
    maxuchr=sys.maxunicode,
    enc_fs=sys.getfilesystemencoding(),
    tz=time.tzname,
    lang=os.getenv("LANG"),
    term=os.getenv("TERM"),
    sh=os.getenv("SHELL"),
))
.
```

If `enc_fs` is **not** `UTF-8`, then call `dpkg-reconfigure locales` (on Debian type systems) and choose a proper locale (you might also need `locale-gen en_US.UTF-8`), and make sure `LANG` is set to `en_US.UTF-8` (or another locale with UTF-8 encoding).

OS Diagnostics

Similarly, execute this in a shell prompt:

```
uname -a; echo $(lsb_release -as 2>/dev/null); grep name /proc/cpuinfo | uniq -c; \
free -m | head -n2; uptime; \
strings $(which rtorrent) | grep "client version"; \
ldd $(which rtorrent) | egrep "lib(torrent|curses|curl|xmlrpc.so|cares|ssl|crypto)"; \
ps auxw | egrep "USER|/rtorrent" | grep -v grep
```

1.5.3 Common Problems & Solutions

Error in option file: .../05-rt-ps-columns.rc:...: Invalid key

You combined a brand-new *pimp-my-box* with an older version of *rTorrent-PS*.

Solution 1 (preferred)

Upgrade the rTorrent-PS Version to a recent build.

Also make sure your `~/rtorrent/rtorrent.rc` is the newest one with the line...

```
method.insert = pyro.extended, const|value, (system.has, rtorrent-ps)
```

This auto-detects the presence of *rTorrent-PS*, but only works with builds from June 2018 onwards.

Solution 2

Replace this line in `~/rtorrent/rtorrent.rc`...

```
method.insert = pyro.extended, const|value, (system.has, rtorrent-ps)
```

with that one...

```
method.insert = pyro.extended, const|value, 1
```

SSH Error: Host key verification failed

If you get this error, one easy way out is to first enter the following command and then repeat your failing Ansible command:

```
export ANSIBLE_HOST_KEY_CHECKING=False
```

rtorrent: command not found

When you get this error using the `tmux` start command as shown in *Starting rTorrent*, then neither a package nor an explicitly compiled binary of rTorrent is installed on your machine.

See *Running the Playbook* on how to solve this.

1.6 Software Updates

1.6.1 Updating Your System with Changes in The Repository

You have full control over when your system is upgraded with new features and fixes, which implies you are also responsible for that.

Read the commit log from the top to the date / commit SHA you last updated your working directory, to actually know what you're installing. You can also use `git log` or `gitk` on your machine for that, *after* a pull.

Call these commands *in your working directory of the pimp-my-box repository* for updating:

```
git pull --ff-only
ansible-playbook -i hosts site.yml
```

Before you start, make sure to read any warnings that might be at the top of the README.

Also *re-read the explanation* of adding `-e force_cfg=yes` and the consequences that has, namely overwriting some configuration files that are normally created only once and then left untouched.

Don't ask *"Should I add this option?"* in support, that is entirely dependent on how *you* manage your system. See above.

1.6.2 Upgrade the Python Version

Updating Python

When you installed *Python* via *pyenv* (i.e. `pyenv_enabled` is still set to `true`), you can update to a new *Python* release by reinstalling the related software.

You first have to update pyenv itself and remove the old version file (this keeps any existing Python version installed). Login as `rtorrent` and call these commands:

```
cd ~/.local/pyenv
git pull --ff-only
for i in plugins/py*; do ( cd $i && git pull --ff-only ); done
mv version version.$(date --rfc-3339=date)
```

To get a list of choices for the new version, login as `rtorrent` and call this:

```
cd ~/.local/pyenv/ && bin/pyenv install -l | egrep ' 3.[6-9]| 2.7'
```

Now select the specific Python version you want installed, by setting the `pyenv_python_version` variable in your `host_vars` or `group_vars`. On older releases like Trusty, you can only go as high as `3.6.10`, because their SSL libraries are too old for newer Python versions.

Then execute the `pyenv` role:

```
ansible-playbook site.yml -i hosts -t pyenv
```

As given, these commands affect all hosts in the `box` group of your inventory.

Updating Virtual Environments

To update all virtualenvs so they're using the new version, just remove them (like shown here) or move them to a backup directory:

```
ansible box -i hosts -a \
    "bash -c 'cd ~rtorrent/.local && rm -rf pyroscope flexget'"
```

Then execute the relevant roles to restore them with the new version setup:

```
ansible-playbook site.yml -i hosts -t cli,fg
```

Note that both `pyrocore` and `flexget` get upgraded to their newest available version by this.

1.6.3 Update to Ansible2 on Your Workstation

Make sure to uninstall the old Ansible version, or move its commands to an extra directory:

```
ls -l ~/bin/ansible*
mkdir -p ~/.local/bin/ansible1
mv ~/bin/ansible* $_
```

Install Ansible 2.9.5, see *Installing Ansible* for details:

```
./scripts/install_ansible.sh
```

Check by calling `ansible --version`, which should show something like this:

```
ansible 2.9.5
  config file = ~/.ansible.cfg
  configured module search path = ['~/.ansible/plugins/modules', '/usr/share/ansible/
↪plugins/modules']
  ansible python module location = ~/.local/venvs/ansible2/lib/python3.6/site-
↪packages/ansible
  executable location = ~/.local/bin/ansible
  python version = 3.6.8 (default, Jan 28 2020, 20:29:43) [GCC 4.6.3]
```

Add this to your `host_vars` file(s), e.g. `host_vars/my-box/main.yml`:

```
ansible_become: true
ansible_python_interpreter: /usr/bin/python3
```

Try to call the playbook in check mode:

```
ansible-playbook site.yml -l my-box -t base --check --diff
```

This might show deprecation warnings, but should run without errors otherwise.

Hint:

Special considerations for Trusty (Ubuntu 14.04)

Python version 3.4 as available on Trusty is too old for Ansible. So set the Python interpreter explicitly to Python2 as follows:

```
ansible_python_interpreter: /usr/bin/python2
```

1.6.4 Upgrade the rTorrent-PS Version

To upgrade the installed `rtorrent-ps` package, execute this command on your workstation:

```
ansible box -i hosts -a "rm /opt/rtorrent/pmb-installed"
```

Then run the playbook to install the new version:

```
ansible-playbook site.yml -i hosts -t rtps
```

Finally connect to your `tmux` session, and stop & restart rTorrent.

CHAPTER

TWO

INDICES & TABLES

- genindex
- modindex
- search