
Python

Oct 15, 2019

Contents

1	In this installation	3
1.1	Introduction	3
1.2	Overview	4
1.3	Getting Started Quickly	6
1.4	The Requirements	9
1.5	Installation	10
1.6	Post Installation	10
1.7	Full Node Remote Access	16
1.8	Files and Locations	23
1.9	Maintenance	24
1.10	Security Hardening	25
1.11	Fullnode IRI Configuration Utility	37
1.12	Troubleshooting	38
1.13	FAQ	44
1.14	Uninstall	48
1.15	Command Glossary	48
1.16	Appendix	50
1.17	Disclaimer	61
1.18	Donations	61



Quick installation via *Getting Started Quickly*.

Watch [this video](#) on how simple it is to install a node using the IRI playbook! (credits to discord user TangleAid)

CHAPTER 1

In this installation

- Automate the installation
- Take care of firewalls
- Automatically configure the java memory limit based on your system's RAM
- Configuration tools
- Explain how to connect a wallet to your full node
- Install IOTA Peer Manager
- Serve IOTA PM and Graphs password protected via HTTPS
- Optionally install [Nelson](#).
- Install monitoring graphs. Big thanks to Chris Holliday's [IOTA Exporter](#).
- Email alert notifications manager

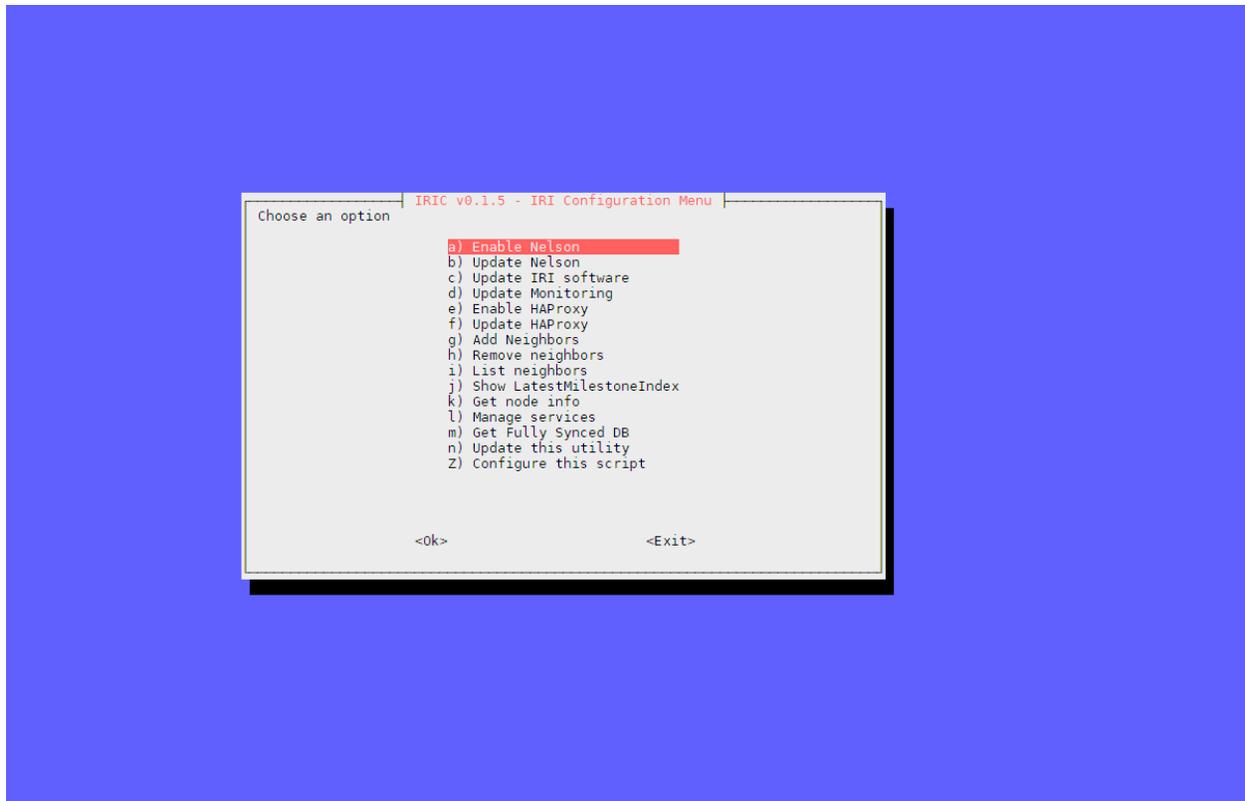
Feel free to star this repository: [iri-playbook](#)

1.1 Introduction

Due to the exponential growth of the community and users who want to run their own full node, I thought it is a good time to write a new, more comprehensive tutorial.

Not only a tutorial: the [iri-playbook](#) is a full fledged IOTA node installer including a comprehensive installer.

Note: Checkout the new addition to the [playbook](#): a handy tool to help manage the full node's services:



1.1.1 Why Another Tutorial?

I am hoping this tutorial, together with the installer will come in handy for those who possess less or almost no skills with Linux. And indeed, this tutorial focuses on Linux – as suggested by many other tutorials (and justifiably), Linux is the best way to go.

I found that many tutorials lack some basic system configuration and explanations thereof. For example, running IRI as an unprivileged user, configuring firewalls, how to connect to it remotely and so on. While the installer takes care of most of those things, the documentation covers further tweakings, maintenance and upgrades.

A copy-paste tutorial is awesome, and as it so often happens, the user can miss on some basic technical explanation about the setup. While it is impossible to include a crash-course of Linux for the purpose of this tutorial, I will try to explain some basic concepts where I find that many users had troubles with.

Feel free to comment, create issues or contact me on IOTA's Discord chat application (nuriel77) for advice and information.

Good luck!

1.2 Overview

Using the documentation you will be able to setup a full node on a Linux system (Ubuntu or CentOS).

Use this link to install your node *Getting Started Quickly*.

The [git repository](#) includes all the code of the automated installation using Ansible Playbook.

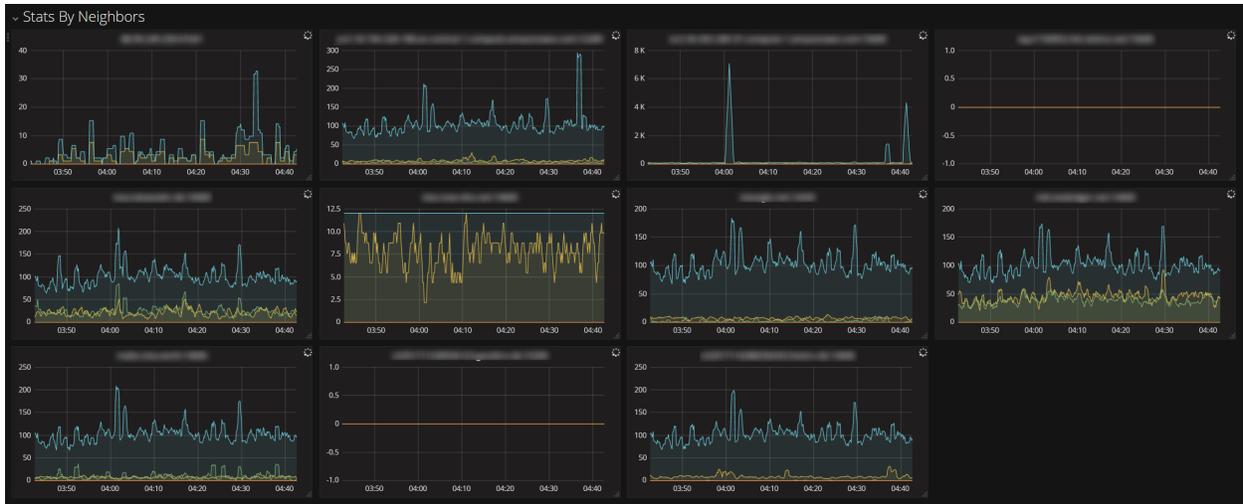
The automated installation installs IRI and IOTA peer manager, a web GUI with which you can view your neighbors, add or remove neighbors, view the sync and much more.

In addition, it will install IOTA node monitoring/graphs (grafana).

For help and/or feedback you can create an issue on the git repository, or try to contact me on IOTA's Discord chat app (nuriel77).

Here are some screenshots from Chris Holliday's [IOTA Exporter](#)., which is installed by default with this installer:





1.3 Getting Started Quickly

NOTE This is the original playbook. It will be deprecated on September 30, 2019. Only available version will be the dockerized version.

You can skip most of the information in this tutorial should you wish to do so and go straight ahead to install the full node.

If you haven't already, just make sure your server matches the *The Requirements*.

Warning: Your server's installation of Ubuntu or CentOS must be a clean one, i.e. no pre-installed cpanel, whcms, plesk and so on. This installer might BREAK any previously installed web-server. It is meant to be installed on a clean system!

1.3.1 Run the Installer!

For **CentOS** users: you may need to install `curl`. You can do that by running: `sudo yum install curl -y`.

For **Ubuntu/Debian** if `curl` is missing you can install it by running: `sudo apt-get install curl -y`.

The installation command will pull a script and kick off the installation. Make sure you read the warning above!

IRI Playbook Installation Command

run:

```
GIT_OPTIONS="-b feat/docker" bash <(curl -s https://raw.githubusercontent.com/nuriel77/iri-playbook/feat/docker/fullnode_install.sh)
```

Documentation for this version can be [found here](#).

Additional Information

- If during the installation you are requested to reboot the node, just do so and re-run the commands above once the node is back.
- A successful installation will display some information when it is done, e.g. the URLs where you can access the graphs and IOTA Peer Manager.

By default you can access the graphs at:

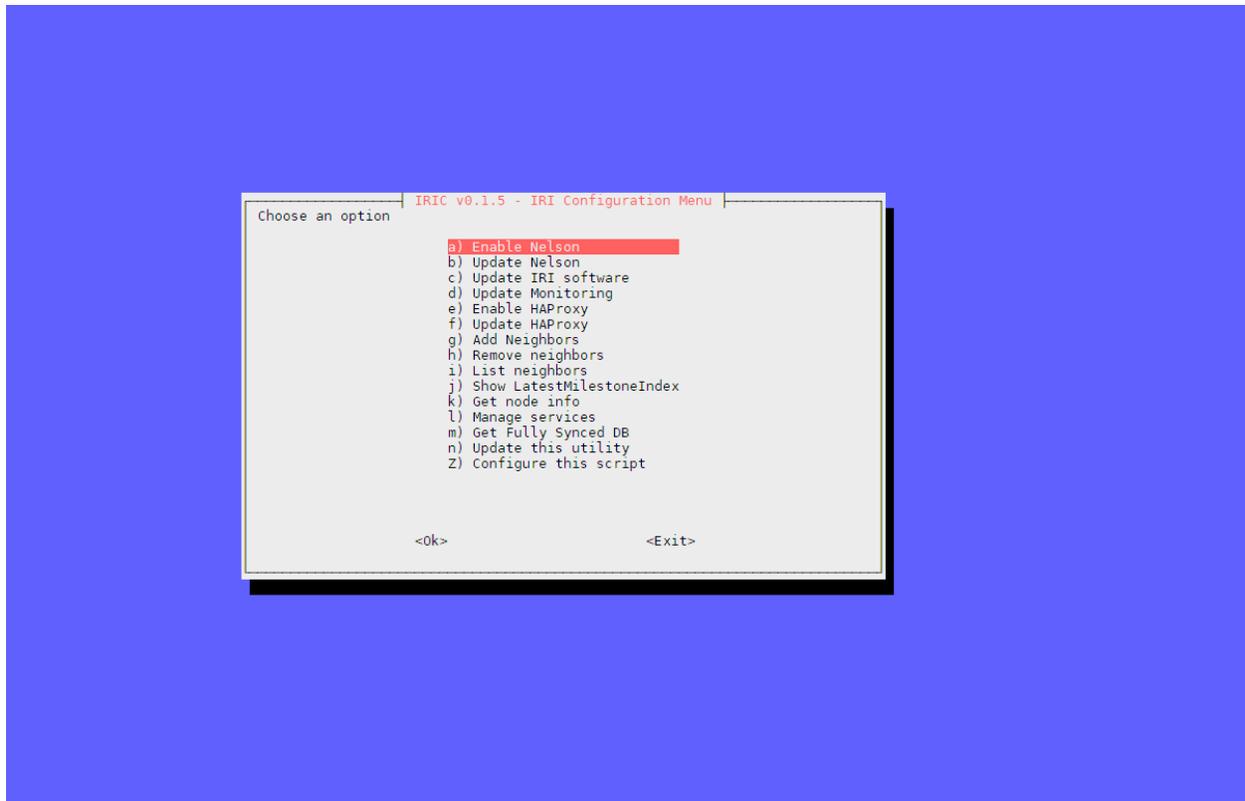
```
http://your-ip:5555/dashboard/db/iota?refresh=30s&orgId=1
```

and Peer Manager via:

```
http://your-ip:8811
```

- You can use the user `iotapm` and the password you've configured during the installation.
- You should be redirected to a HTTPS URL (this has been added recently). This is a self-signed certificate: you will get a warning from the browser. You can add the certificate as an exception and proceed. In the 'appendix' chapter there's some information how to install valid certificates (certbot).
- Please consider hardening the security of your node. Any server is a target for attacks/brute forcing. Even more so if you are going to list your node publicly. See *Security Hardening*.
- You can proceed to the *Post Installation* for additional information on managing your node.
- To configure an email for alerts see *Sending Alert Notifications*.
- Visit the project's page at the [IOTA Ecosystem](#)

Note: Checkout the new addition to the playbook: a handy tool to help manage the full node's services:



Connection Lost

If you lost connection to your server during the installation, don't worry. It is running in the background because we are running it inside a "screen" session .

You can always "reattach" back that session when you re-connect to your server:

```
screen -r -d iota
```

Note: Pressing arrow up on the keyboard will scroll up the command history you've been running. This saves some typing when you need to run the same command again!

Warning: Some VPS providers might be depending on Network Block Devices (for example Scaleway). If using Ubuntu, you need to configure `ufw` prior to running the installer. See: <https://gist.github.com/georgkreimer/7a02af49604da91c5e3605b08b2872ec>

Accessing Peer Manager

You can access the peer manager using the user 'iotapm' and the password you've configured during installation:

```
http://your-ip:8811
```

Accessing Monitoring Graphs

You can access the Grafana IOTA graphs using 'iotapm' and the password you've configured during the installation

```
http://your-ip:5555
```

Big thanks to Chris Holliday's amazing tool for [node monitoring](#)

1.4 The Requirements

- *Virtual Private Server*
- *Operating System*
- *Accessing the VPS*
- *System User*

1.4.1 Virtual Private Server

This is probably the best and most common option for running a full node.

There are many companies offering a VPS for good prices. Make sure not to take a VPS platform which is based on Virtuozzo or OpenVZ. Performance is not best and I personally don't like the fact the hosting company can see what processes I am running on my private server.

Also, a good advice is not to take a contract for a year, but try to find hosting services with pay-per-hour or monthly contract. Some hostings such as SSDNodes are not recommended: some fullnode operators claimed their contracts have been cancelled due to running "crypto" software (IRI).

The basic recommendation is to have one with at least 4GB RAM, 2 cores and minimum 60GB harddrive (SSD preferably).

1.4.2 Operating System

When you purchase a VPS you are often given the option which operating system (Linux of course) and which distribution to install on it.

This tutorial/installer was tested on:

- Ubuntu 16.04 (amd64) Server Cloud Image (Xenial)
- Ubuntu 17.04 (amd64) Server Cloud Image (Zesty)
- Ubuntu 18.04 (amd64) Server Cloud Image (Bionic)
- Ubuntu 16.04, 17.10 and 18.04 (amd64) Server image ISO
- CentOS 7.4 x86_64 Generic Cloud Image or CentOS Minimal ISO

Note: This installation does not support operating systems with pre-installed panels such as cpanel, whcms, plesk etc. If you can, choose a "bare" system.

Warning: Some VPS providers provide a custom OS installation (Ubuntu or CentOS) with additional software installed (LAMP, cpanel etc). These images will not work nicely with the installer. In some cases, VPS providers modify images and might deliver operating systems that will be incompatible with this installer.

1.4.3 Accessing the VPS

Once you have your VPS deployed, most hosting provide a terminal (either GUI application or web-based terminal). With the terminal you can login to your VPS's command line. You probably received a password with which you can login to the server. This can be a 'root' password, or a 'privileged' user (with which you can access 'root' privileges).

The best way to access the server is via a Secure Shell (SSH). If your desktop is Mac or Linux, this is native on the command line. If you use Windows, I recommend installing [Putty](#)

There are plenty of tutorials on the web explaining how to use SSH (or SSH via Putty). Basically, you can use a password login or SSH keys (better).

1.4.4 System User

Given you are the owner of the server, you should either have direct access to the 'root' account or to a user which is privileged. It is often recommended to run all commands as the privileges user, prefixing the commands with 'sudo'. In this tutorial I will leave it to the user to decide.

If you accessed the server as a privileged user, and want to become 'root', you can issue a `sudo su -`. Otherwise, you will have to prefix most commands with `sudo`, e.g.

```
sudo apt-get install something
```

1.5 Installation

Please visit [Getting Started Quickly](#).

1.6 Post Installation

At time of writing, the database is quite large (10GB+). In order to help your node catch up to speed it is recommended to download a fully synced database copy. Please refer to [Where can I get a fully synced database to help kick start my node](#) on how to get this done.

We can run a few checks to verify everything is running as expected. First, let's use the `systemctl` utility to check status of `iri` (this is the main full node application)

Using the `systemctl status iri` we can see if the process is Active: `active (running)`.

See examples in the chapters below:

- [Controlling IRI](#)
- [Controlling IOTA Peer Manager](#)
- [Checking Ports](#)
- [Checking IRI Full Node Status](#)
- [Connecting to IOTA Peer Manager](#)

- [Adding or Removing Neighbors](#)
- [Install IOTA Python libs](#)

Note: See [Maintenance](#) for additional information, for example checking logs and so on. Also, you can refer to [Command Glossary](#) for a quick over view of most common commands.

Warning: All web pages served by this installer will be served on HTTPS with self-signed certificates. The browser will issue a warning when you connect for the first time. You can proceed and accept the certificate as an exception. If you want valid certificates you can refer to [serverHTTPS](#) and look for the Let's encrypt

link.

1.6.1 Controlling IRI

Check status:

```
systemctl status iri
```

Stop:

```
systemctl stop iri
```

Start:

```
systemctl start iri
```

Restart:

```
systemctl restart iri
```

1.6.2 Controlling IOTA Peer Manager

Check status:

```
systemctl status iota-pm
```

Stop:

```
systemctl stop iota-pm
```

Start:

```
systemctl start iota-pm
```

Restart:

```
systemctl restart iota-pm
```

1.6.3 Checking Ports

IRI uses 2 ports by default:

1. TCP neighbor peering port
2. TCP API port (this is where a light wallet would connect to or iota peer manager)

You can check if IRI and iota-pm are “listening” on the ports if you run:

```
lsof -Pni|egrep "iri|iotapm".
```

Here is the output you should expect:

```
# lsof -Pni|egrep "iri|iotapm"
java      2297    iri      21u  IPv6  20334      0t0  TCP *:15600 (LISTEN)
java      2297    iri      32u  IPv6  20345      0t0  TCP 127.0.0.1:14265 (LISTEN)
node      2359  iotapm   12u  IPv4  21189      0t0  TCP 127.0.0.1:8011 (LISTEN)
```

What does this tell us?

1. *:<port number> means this port is listening on all interfaces - from the example above we see that IRI is listening on port TCP no. 15600
2. IRI is listening for API (or wallet connections) on a local interface (not accessible from “outside”) no. 14265
3. Iota-PM is listening on local interface port no. 8011

Now we can tell new neighbors to connect to our IP address.

Here’s how to check your IP address:

If you have a static IP - which a VPS most probably has - you can view it by issuing a `ip a`. For example:

```
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8950 qdisc pfifo_fast state UP qlen_
   ↪1000
   link/ether fa:16:3e:d6:6e:15 brd ff:ff:ff:ff:ff:ff
   inet 10.50.0.24/24 brd 10.50.0.255 scope global dynamic eth0
       valid_lft 83852sec preferred_lft 83852sec
   inet6 fe80::c5f4:d95b:ba52:865c/64 scope link
       valid_lft forever preferred_lft forever
```

See the IP address on `eth0`? (10.50.0.24) this is the IP address of the server.

Yes - for those of you who’ve noticed, this example is a **private** address. But if you have a VPS you should have a public IP.

I could tell neighbors to connect to my tcp port: `tcp://10.50.0.14:15600`.

Note that the playbook installation automatically configured the firewall to allow connections to these ports. If you happen to change those, you will have to allow the new ports in the firewall (if you choose to do so, check google for iptables or firewalld commands).

1.6.4 Checking IRI Full Node Status

The tool `curl` can issue commands to the IRI API.

For example, we can run:

```
curl -s http://localhost:14265 -X POST -H 'X-IOTA-API-Version: someval' -H 'Content-
↳Type: application/json' -d '{"command": "getNodeInfo"}' | jq
```

The output you will see is JSON format. Using `jq` we can, for example, extract the fields of interest:

```
curl -s http://localhost:14265 -X POST -H 'X-IOTA-API-Version: someval' -H 'Content-
↳Type: application/json' -d '{"command": "getNodeInfo"}' | jq '.
↳latestSolidSubtangleMilestoneIndex, .latestMilestoneIndex'
```

Note: If you've just started up your IRI node (or restarted) you will see a matching low number for both `latestSolidSubtangleMilestoneIndex` and `latestMilestoneIndex`. This is expected, and after a while (10-15 minutes) your node should start syncing (given that you have neighbors).

1.6.5 Connecting to IOTA Peer Manager

For IOTA Peer Manager, this installation has already configured it to be accessible via a webserver. See *Peer Manager Behind WebServer with Password*.

1.6.6 Adding or Removing Neighbors

In order to add neighbors you can either use the `iota` Peer Manager or the command-line.

To use the command line you can use the script `nbctl` that was shipped with this installation.

If you don't have `nbctl` installed you can get it by running:

```
wget -O /usr/bin/nbctl https://raw.githubusercontent.com/nuriel77/iri-playbook/master/
↳roles/iri/files/nbctl && chmod +x /usr/bin/nbctl
```

nbctl script

You can run `nbctl` with `-h` to get help on all the options:

```
# nbctl -h
usage: nbctl [-h] [--neighbors NEIGHBORS] [--remove] [--add] [--list]
           [--file FILE] [--host HOST] [--api-version API_VERSION]

Add or remove full node neighbors.

optional arguments:
  -h, --help                show this help message and exit
  --neighbors NEIGHBORS, -n NEIGHBORS
                           Neighbors to process. Can be specified multiple times.
  --remove, -r              Removes neighbors
  --add, -a                 Add neighbors
  --list, -l                List neighbors
```

(continues on next page)

(continued from previous page)

```
--file FILE, -f FILE Configuration file to update
--host HOST, -i HOST IRI API endpoint. Default: http://localhost:15265
--api-version API_VERSION, -x API_VERSION
                        IRI API Version. Default: 1.4
```

```
Example: nbctl -a -n tcp://1.2.3.4:12345 -n tcp://4.3.2.1:4321 -f /etc/default/iri
```

The nice thing about `nbctl` is that it communicates with IRI to add/remove neighbors and also updates the configuration file.

Updating the configuration file is important - if you restart IRI it will start with the neighbors listed in the configuration file.

- The script will connect by default to IRI API on `http://localhost:14265`.
- If you need to connect to a different endpoint you can specify that using `-i http://my-node-address:port`.
- `nbctl` also has the ability to configure the configuration file for you!

Listing Neighbors

If you want to list neighbors, simply run:

```
nbctl -l
```

To show only the addresses and ports, run:

```
nbctl -l | jq -r '.neighbors[] | "\(.address)\(.connectionType) "'
```

Adding Neighbors

To add one or more neighbors use the `-a` option and specify the neighbors using `-n neighbors-address`, once or multiple times, e.g.:

```
nbctl -a -n tcp://1.2.3.4:12345 -n tcp://4.3.2.1:4321 -n tcp://
↪[2a01:a0a0:c0c0:1234::1]:14600 -f /etc/default/iri
```

Note that the last options `-f /etc/default/iri` will also add the neighbors to the configuration file, but **make sure** you are pointing to the correct file. For example, in CentOS it is `/etc/sysconfig/iri`, on other guides it is located in `/home/iota/node/iota.ini!!!`

In the example above note the IPv6 address: it is encapsulated in square brackets. This is the correct syntax for IPv6 addresses.

Removing Neighbors

To remove one or more neighbors use the `-r` option and specify the neighbors using `-n neighbors-address`, once or multiple times, e.g.:

```
nbctl -r -n tcp://1.2.3.4:12345 -n tcp://4.3.2.1:4321 -f /etc/default/iri
```

Note that the last option `-f /etc/default/iri` will also add the neighbor(s) to the configuration file. **Make sure** you are pointing to the correct file. For example, in CentOS it is `/etc/sysconfig/iri`, on other guides it is located in `/home/iota/node/iota.ini`!!!

Using curl

If you don't have `nbctl` script you can to run a `curl` command, e.g. to add:

```
curl -H 'X-IOTA-API-VERSION: 1.4' -d '{"command":"addNeighbors",
  "uris":["tcp://neighbor-ip:port", "tcp://neighbor-ip:port", "tcp://
  ↪[2a01:a0a0:c0c0:1234::1]:14600"]}' http://localhost:14265
```

to remove:

```
curl -H 'X-IOTA-API-VERSION: 1.4' -d '{"command":"removeNeighbors",
  "uris":["tcp://neighbor-ip:port", "tcp://neighbor-ip:port"]}' http://localhost:14265
```

to list:

```
curl -H 'X-IOTA-API-VERSION: 1.4' -d '{"command":"getNeighbors"}' http://
  ↪localhost:14265
```

Note: Adding or remove neighbors is done “on the fly” with `curl`, so you will also have to add (or remove) the neighbor(s) in the configuration file of IRI.

The reason to add it to the configuration file is that after a restart of IRI, any neighbors added with the peer manager will be gone.

On **CentOS** you can add neighbors to the file:

```
/etc/sysconfig/iri
```

On **Ubuntu**:

```
/etc/default/iri
```

Edit the `IRI_NEIGHBORS=""` value as shown in the comment in the file.

Note: See *Using Nano to Edit Files* for instructions on how to use `nano` for editing files.

1.6.7 Install IOTA Python libs

You can install the official `iota.libs.py` to use for various python scripting with IOTA and the `iota-cli`.

On **Ubuntu**:

```
apt-get install python-pip -y && pip install --upgrade pip && pip install pyota
```

You can test with the script that shipped with this installation (to reattach pending transactions):

```
reattach -h
```

On **CentOS** this is a little more complicated, and better install `pyota` in a “virtualenv”:

```
cd ~
yum install python-pip gcc python-devel -y
virtualenv venv
source ~/venv/bin/activate
pip install pip --upgrade
pip install pyota
```

Now you can test by running the reattach script as shown above.

Note: Note that if you log in back to your node you will have to run the `source ~/venv/bin/activate` to switch to the new python virtual environment.

1.7 Full Node Remote Access

Update: the recommended way to enable remote access to IRI API port (e.g. for wallets) is via HAProxy. Please refer to *Running IRI API Port Behind HAProxy*.

1.7.1 1. Exposing IRI Port Externally

IRI has a command-line argument (“option”) `--remote true`. Here’s an explanation on what it does:

By default, IRI’s API port will listen on the local interface (127.0.0.1). This prevents any external connections to it.

Using the `--remote` option, IRI will “listen” on the external interface/IP.

We are going to have to edit the configuration file to enable this option and restart IRI. Follow the next steps.

Note: To edit files you can use `nano` which is a simple editor. See *Using Nano to Edit Files* for instructions.

The `--remote` option can be specified in the configuration file:

- on **CentOS** `/etc/sysconfig/iri`
- on **Ubuntu** `/etc/default/iri`

Edit the file and find the line:

```
OPTIONS=""
```

and add `--remote true` to it:

```
OPTIONS="--remote true"
```

Save the file and exit, then restart iri: `systemctl restart iri`

After IRI initializes, you will see (by issuing `lsof -Pni|grep java`) that the API port is listening on your external IP.

You can follow the instructions below on how to enable access to the port on the firewall.

Note: By default, this installation is set to **not** allow external communication to this port for security reasons. Should you want to allow this, you need to allow the port in the firewall.

Expose IRI API Port in Firewall

Allowing the port via the playbook

If you followed the steps above (enabling the `--remote true` option in the configuration file) you will need to allow the port in the firewall.

You can do this using the playbook which as a bonus also adds rate limiting.

On **CentOS**:

```
cd /opt/iri-playbook && git pull && ansible-playbook -i inventory -v site.yml --
↳tags=iri_firewalld -e api_port_remote=yes
```

On **Ubuntu** without rate limiting:

```
cd /opt/iri-playbook && git pull && ansible-playbook -i inventory -v site.yml --
↳tags=iri_ufw -e api_port_remote=yes
```

On **Ubuntu** with rate limiting:

```
cd /opt/iri-playbook && git pull && ansible-playbook -i inventory -v site.yml --
↳tags=iri_ufw -e api_port_remote=yes -e ufw_limit_iri_api=yes
```

Note: Rate limiting in ubuntu is using ufw which is a very simple wrapper to the iptables firewalls. It only allows one value of max 6 connections per 30 seconds. This might prevent doing PoW on your node if you choose to expose attachToTangle.

Allowing the port manually

On **CentOS** we run the command (which also adds rate limiting):

```
firewall-cmd --remove-port=14265/tcp --zone=public --permanent && firewall-cmd --
↳zone=public --permanent --add-rich-rule='rule port port="14265" protocol="tcp"
↳limit value=30/m accept' && firewall-cmd --reload
```

On **Ubuntu**:

```
ufw allow 14265/tcp
```

And to add rate limits:

```
ufw limit 14265/tcp comment 'IRI API port rate limit'
```

Note: Rate limiting via ufw on ubuntu is very simple in that it only allows a value of 6 hits per 30 seconds. This can be a problem if you want to enable PoW – attachToTangle on your node.

Now you should be able to point your (desktop's) light wallet to your server's IP:port (e.g. 80.120.140.100:14265).

1.7.2 2. Tunneling IRI API for Wallet Connection

Another option for accessing IRI and/or the iota-pm GUI is to use a SSH tunnel.

SSH tunnel is created within a SSH connection from your computer (desktop/laptop) towards the server.

The benefit here is that you don't have to expose any of the ports or use the `--remote true` flag. You use SSH to help you tunnel through its connection to the server in order to bind to the ports you need.

Note: For IOTA Peer Manager, this installation has already configured it to be accessible via a webserver. See [Peer Manager Behind WebServer with Password](#)

What do you need to “forward” the IRI API?

- Your server's IP
- The SSH port (22 by default in which case it doesn't need specifying)
- The port on which IRI API is listening
- The port on which you want to access IRI API on (let's just leave it the same as the one IRI API is listening on)

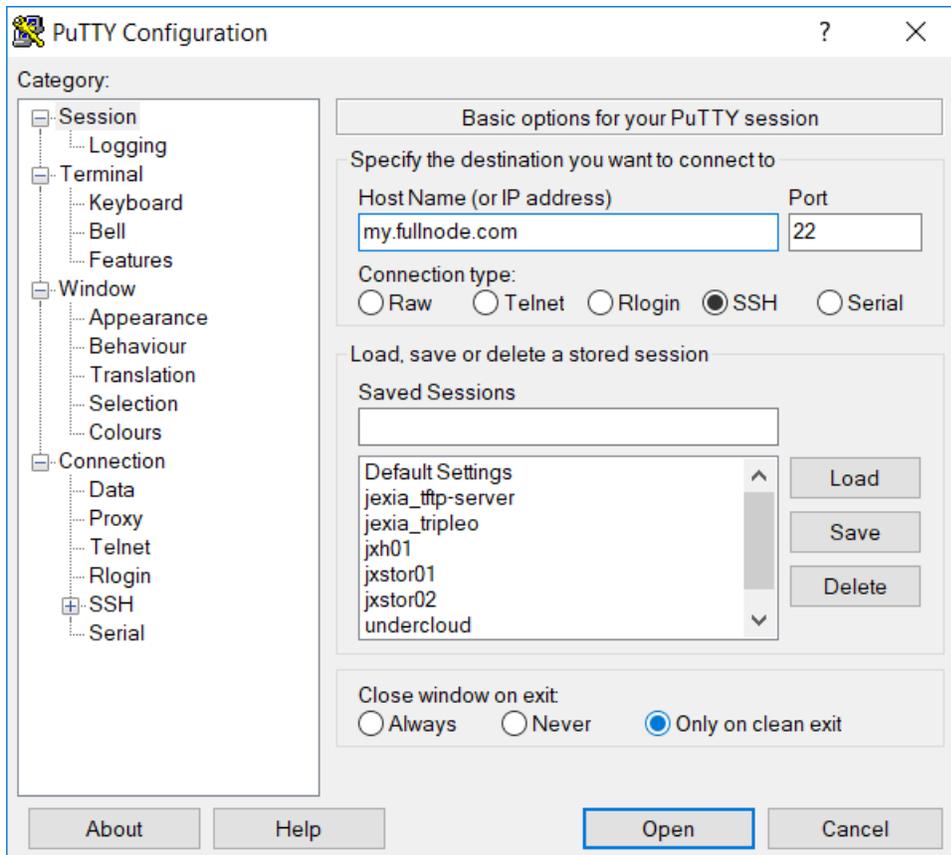
A default installation would have IRI API listening on TCP port 14265.

Note: In order to create the tunnel you need to run the commands below **from** your laptop/desktop and not on the server where IRI is running.

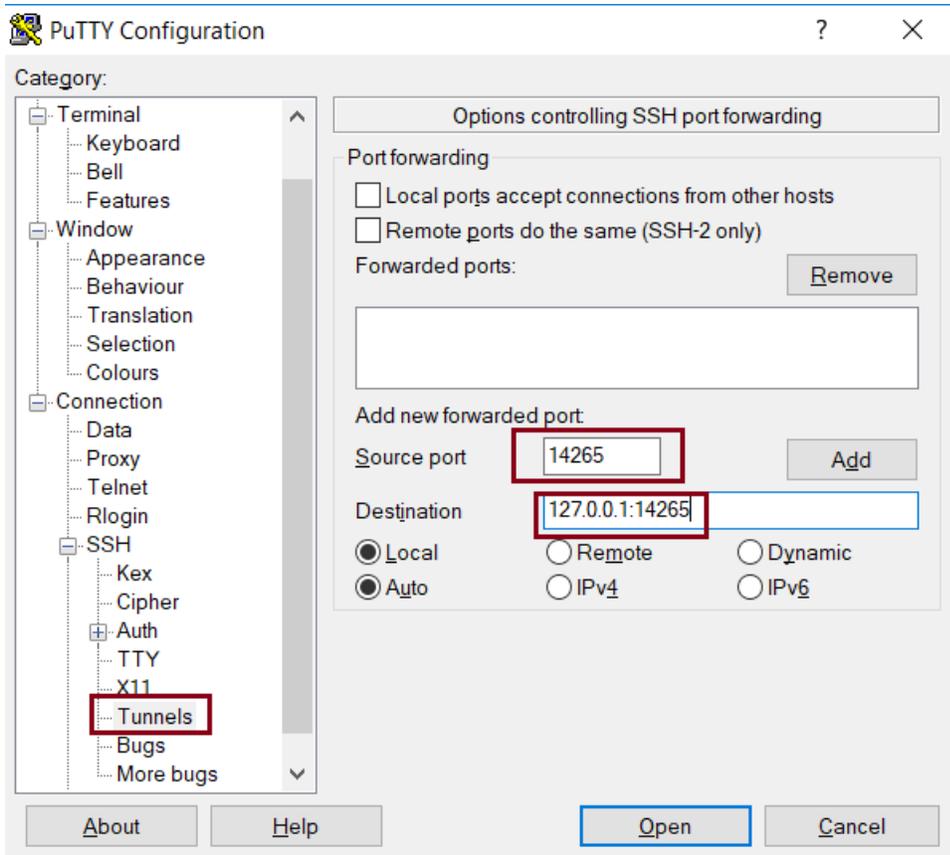
For Windows desktop/laptop

You can use Putty to create the tunnel/port forward. This can be done for any port on the server. Here we are going to forward the IRI API port from the server to your local machine.

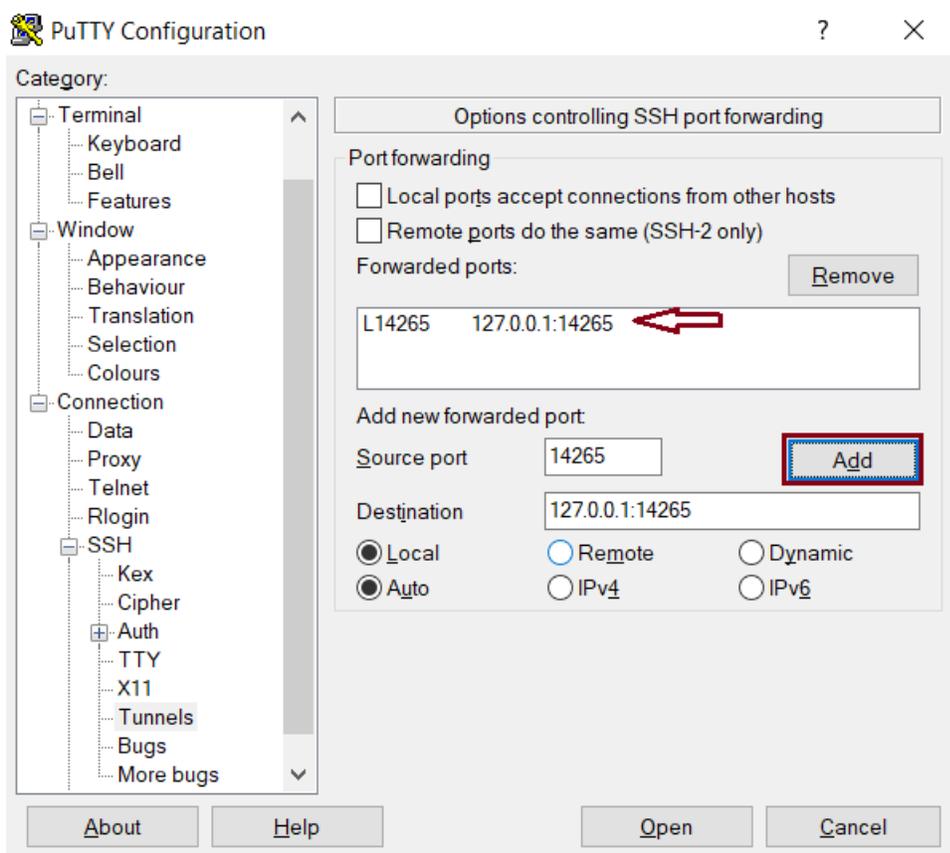
1. Open putty and create a new session name. Start by entering the node's address and SSH port.



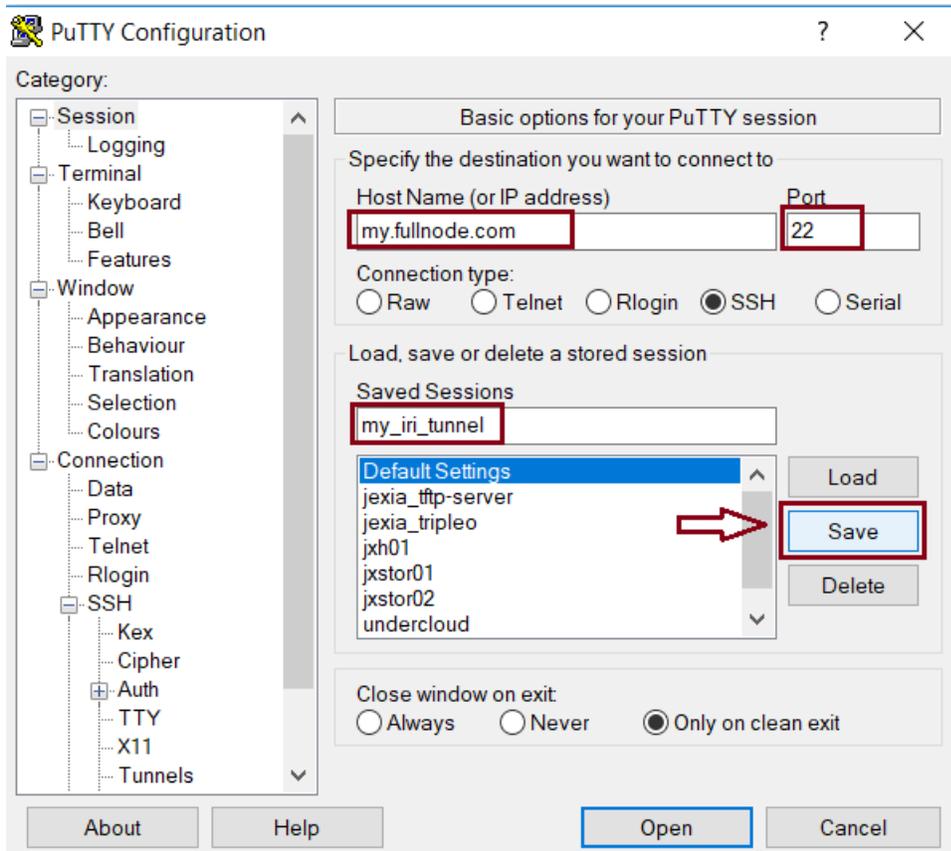
2. On the menu on the left choose 'Tunnels'. Then fill in the Source port and Destination as shown in the image below. The destination is comprised of the IP address and the port. We use 127.0.0.1:14265, as this is by default where we want to forward the port from.



3. Next click 'Add'. You will see that the configuration has been added to the 'Forwarded ports' area.



4. Back in the 'Session' menu, enter a name with which you want to save this configuration/session, last check that the node's address and port are correct, and click 'Save'. The session will be added to the list.



- To open the session and start the port forwarding, all you have to do is to load the session and click 'Open'. To test that the port is being forwarded you can open the browser and point it to `http://localhost:14265`. This should reply something in the lines of `error: Invalid API Version`. If this is the case, your API port is being forwarded successfully. You can edit the wallet's node configuration and point it to this address to start using your full node!

For any type of bash command line (Mac/Linux/Windows bash)

Here is the tunnel we would have to create (run this on our laptop/desktop)

```
ssh -p <ssh port> -N -L <iota-pm-port>:localhost:<iota-pm-port> <user-name>@<server-
ip>
```

Which would look like:

```
ssh -p 22 -N -L 14265:localhost:14265 root@<your-server-ip>
```

Should it ask you for host key verification, reply 'yes'.

Once the command is running you will not see anything, but you can connect with your wallet. Edit your wallet's "Edit Node Configuration" to point to a custom host and use `http://localhost:14265` as address.

To stop the tunnel simply press `Ctrl-C`.

You can do the same using the IRI API port (14265) and use a light wallet from your desktop to connect to `http://localhost:14265`.

1.7.3 Peer Manager Behind WebServer with Password

This installation also configured a webserver (nginx) to help access IOTA Peer Manager. It also locks the page using a password, one which you probably configured earlier during the installation steps.

The IOTA Peer Manager can be accessed if you point your browser to: `https://your-server-ip:8811`.

Note: The port 8811 will be configured by default unless you changed this before the installation in the variables file.

1.7.4 Limiting Remote Commands

There's an option in the configuration file which works in conjunction with the `--remote true` option:

```
REMOTE_LIMIT_API="removeNeighbors, addNeighbors, interruptAttachingToTangle, ↵
↵attachToTangle, getNeighbors"
```

When connecting to IRI via an external IP these commands will be blocked so that others cannot mess with the node's configuration.

Below we describe how to edit these commands, if necessary.

Note: To edit files you can use `nano` which is a simple editor. See *Using Nano to Edit Files* for instructions.

- On **CentOS** edit the file `/etc/sysconfig/iri`
- On **Ubuntu** edit the file `/etc/default/iri`.

This option excludes the commands in it for the remote connection. This is to protect your node. If you make changes to this option, you will have to **restart IRI**: `systemctl restart iri`.

1.8 Files and Locations

Here's a list of files and locations that might be useful to know:

IRI configuration file (changes require iri to restart):

```
Ubuntu: /etc/default/iri
CentOS: /etc/sysconfig/iri
```

IOTA Peer Manager configuration file (changes require iota-pm restart):

```
Ubuntu: /etc/default/iota-pm
CentOS: /etc/sysconfig/iota-pm
```

IRI installation path:

```
/var/lib/iri/target
```

IRI database:

```
/var/lib/iri/target/mainnet*
```

Grafana configuration file:

Python

```
/etc/grafana/grafana.ini
```

Grafana Database file:

```
/var/lib/grafana/grafana.db
```

Prometheus configuration file:

```
/etc/prometheus/prometheus.yaml
```

IOTA-Prom-Exporter configuration file:

```
/opt/prometheus/iota-prom-exporter/config.js
```

Alert Manager configuration file:

```
/opt/prometheus/alertmanager/config.yml
```

HAProxy configuration file:

```
/etc/haproxy/haproxy.cfg
```

Nelson configuration file:

```
/etc/nelson/nelson.ini
```

1.9 Maintenance

- *Upgrade IRI*
- *Upgrade IOTA Monitoring*
- *Check Database Size*
- *Check Logs*
- *Replace Database*

1.9.1 Upgrade IRI

Latest IRI release is available [here](#).

If a new version has been announced, you can follow this guide to get the new version or use the menu-driven tool `iric` to get the latest IRI version.

If using `iric`, make sure to update it to the latest version before using it to upgrade IRI.

In the following example we assume that the new version is **1.6.0**.

Note: The foundation might announce additional information in tandem with upgrades, for example whether to use the `--rescan` flag, remove older database etc. If required, additional options can be specified under the `OPTIONS=""` value in the configuration file (`/etc/default/iri` for Ubuntu or `/etc/sysconfig/iri` for CentOS). The database folder is in `/var/lib/iri/target/mainnetdb` and can be removed using `systemctl stop iri && rm -rf /var/lib/iri/target/mainnet*`.

You can update IRI using the `iric` tool: *Fullnode IRI Configuration Utility*. Make sure that there are no additional manual steps to be taken if any are announced by the Foundation.

1.9.2 Upgrade IOTA Monitoring

IOTA Prometheus Monitoring is used by Grafana which are the awesome graphs about the full node.

You can update the monitoring using the `iric` tool: *Fullnode IRI Configuration Utility*.

1.9.3 Check Database Size

You can check the size of the database using `du -hs /var/lib/iri/target/mainnetdb/`, e.g.:

```
# du -hs /var/lib/iri/target/mainnetdb/
4.9G    /var/lib/iri/target/mainnetdb/
```

Note: To check free space on the system's partitions use `df -h`. If one of the partitions' usage exceeds 85% you should consider a cleanup. Don't worry about the `/boot` partition though.

1.9.4 Check Logs

Follow the last 50 lines of the log (iri):

```
journalctl -n 50 -f -u iri
```

For `iota-pm`:

```
journalctl -n 50 -f -u iota-pm
```

Click 'Ctrl-C' to stop following and return to the prompt.

Alternatively, omit the `-f` and use `--no-pager` to view the logs.

1.9.5 Replace Database

If you want to re-download a fully synced database please refer to *Where can I get a fully synced database to help kick start my node* on how to get this done.

1.10 Security Hardening

In the following chapter some advice and tutorials on how to secure your Linux system.

It includes disabling SSH root access, switching SSH port, creating SSH keys and more.

1.10.1 SSH Key Access

On most servers, password authentication is allowed by default making the server more susceptible to SSH password brute forcing. Switching to SSH key access only is a first good step in making your server more secure.

Once SSH keys authentication is configured (using a user other than root), it is safe to disable password authentication and root SSH access.

Note: Most VPS providers provide a terminal/console access to the server. This is NOT SSH, and can be used to recover access to your server if you get locked out SSH (e.g. configuration error, missing SSH keys, firewall lockout etc.)

Overview

We are going to:

- Create a user with sudo rights (if none exists)
- Explain how to create SSH keys (using puttygen - this is for Windows users)
- Allow for SSH key access using the aforementioned user

This guide is focused on using Putty as a SSH client. If you are using Mac, the process of creating a user on the fullnode server and setting SSH access is the same.

Access User

The first step is to ensure you have a user on the system other than root. Then, grant this user “sudo” privileges.

The following commands assume that you are currently operating as user `root` (verify with `whoami`).

If you already have a user with sudo privileges you can skip this part.

1. Create the user, you can choose a name, and a home directory:

```
useradd -m -d /home/myusername myusername
```

2. Set a password for the new user:

```
passwd myuser
```

3. Add the user to the “sudoers”:

```
echo "myuser ALL=(ALL) NOPASSWD:ALL" >/etc/sudoers.d/myuser && chmod 440 /etc/sudoers.  
→d/myuser
```

4. Check the user is configured properly, run the following commands:

```
su - myuser  
sudo su  
whoami
```

The above should result in `root`. This means that the new user can become root.

Note: It is worth mentioning that a slightly more secure approach would be to add the user to group `wheel`. The difference is that if you add the user to group `wheel`, each time you try to become root you will have to enter the user's password.

Should you want to use this approach, skip step 3 and run `usermod -aG wheel myuser` instead. If you already performed step 3, you can simply remove the file `/etc/sudoers.d/myuser`.

At this point you should be able to SSH into your server using the new user + password. For example: `ssh myuser@myfullnode` for cli, or use Putty.

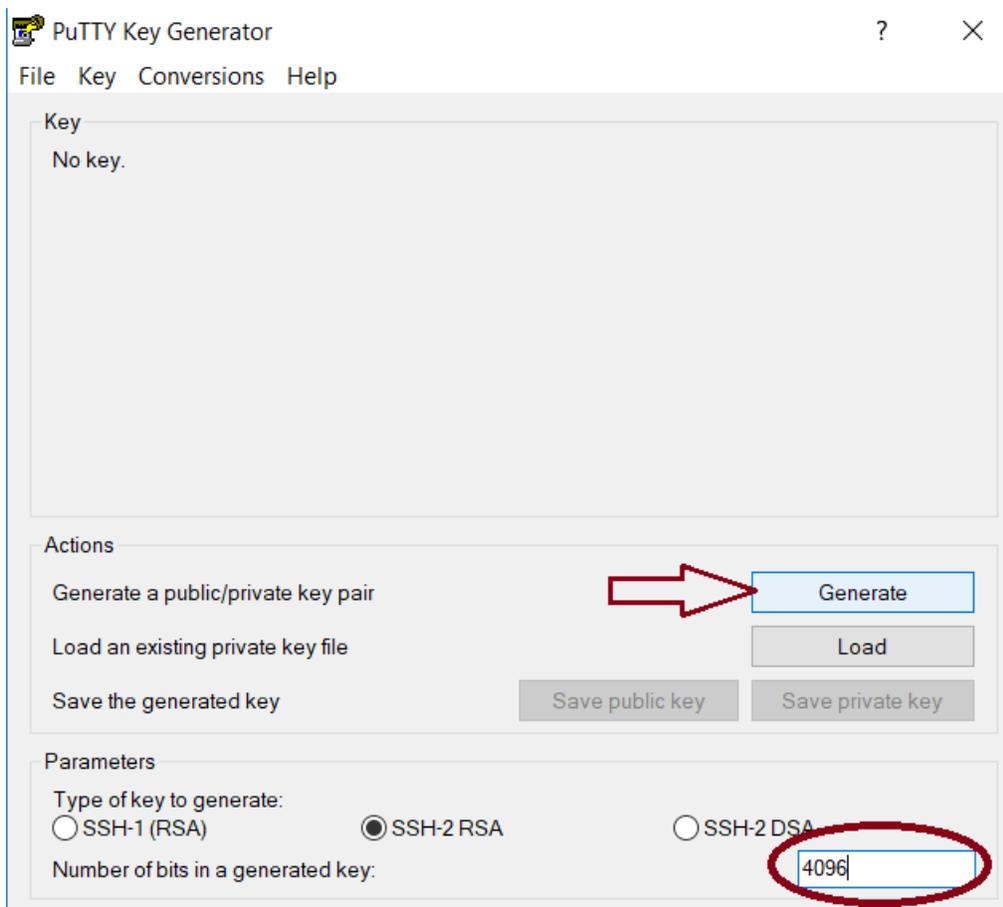
Creating SSH Keys

You can download **Putty** for Windows [here](#). Install the MSI ("Windows Installer") package.

The installer includes: `putty`, `puttygen` and `pagent`.

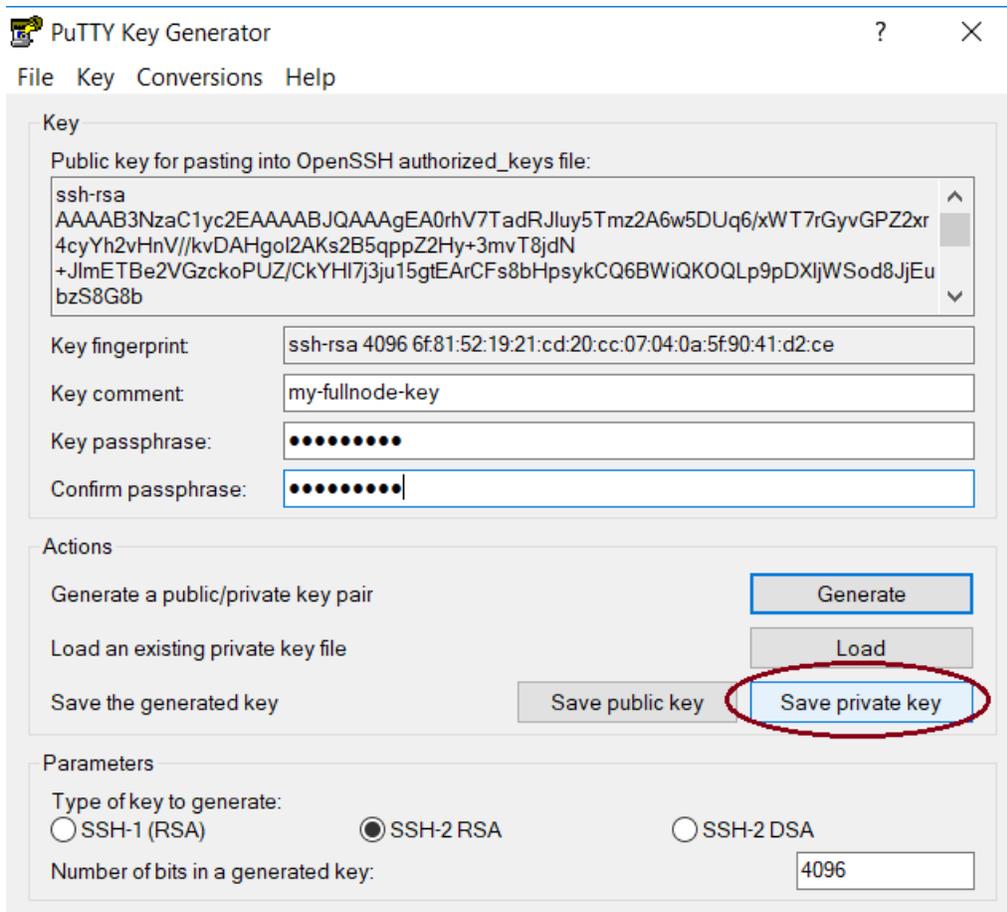
The first step is to create SSH keys. A SSH key pair consists of a private key and a public key (**never** share your private key with anyone and keep it safe!).

1. In Windows, open the application called PuTTYgen. Set the number of bits to 4096 and click generate:



2. Once the key is generated, fill in the comment, choose a (strong) password and click "Save private key". Don't close Puttygen yet!

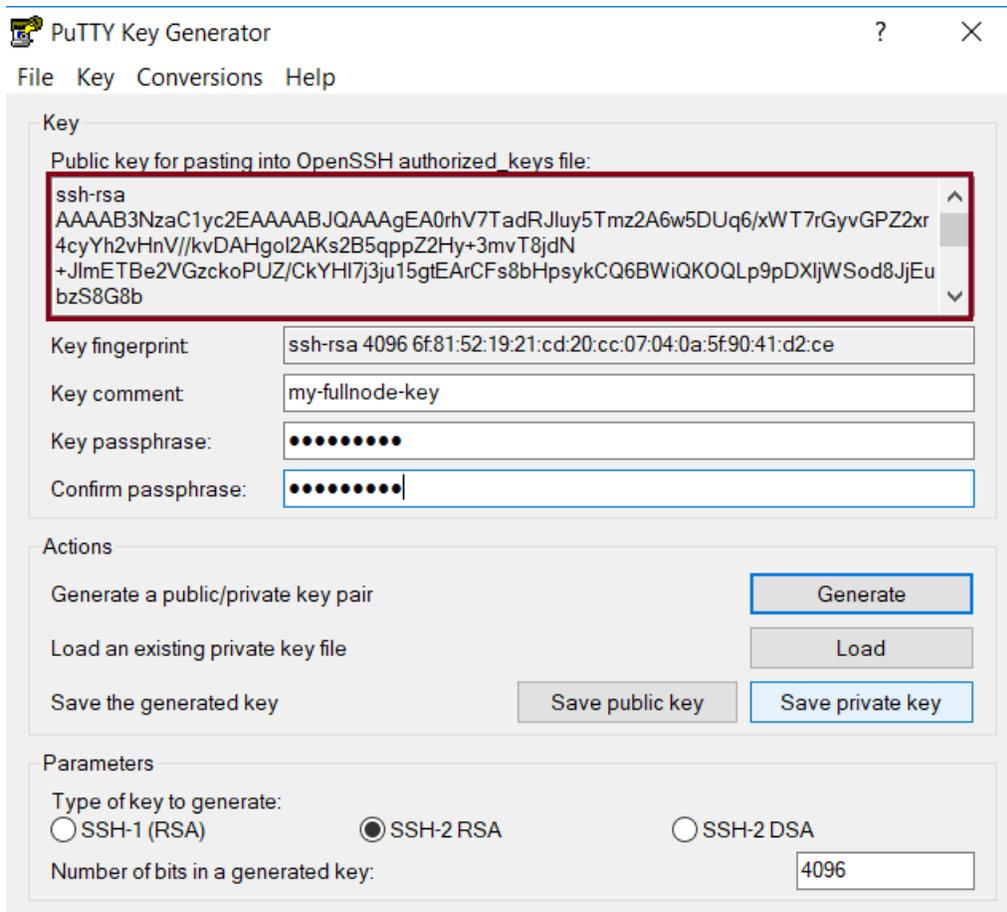
Remember where you save the key to. We are going to use it in the following steps.



3. On the server, make sure you are operating as the user you've created earlier (`whoami` to verify, or `su - myuser` to switch to the user).
4. Create the ssh folder:

```
mkdir -p ~/.ssh
```

5. Select and copy the entire ssh public key from puttygen (see image below). Then, using nano or vi editor, add the public SSH key to a new file `~/.ssh/authorized_keys` on the server.



See [Using Nano to Edit Files](#) to learn how to use nano.

6. Set correct permissions:

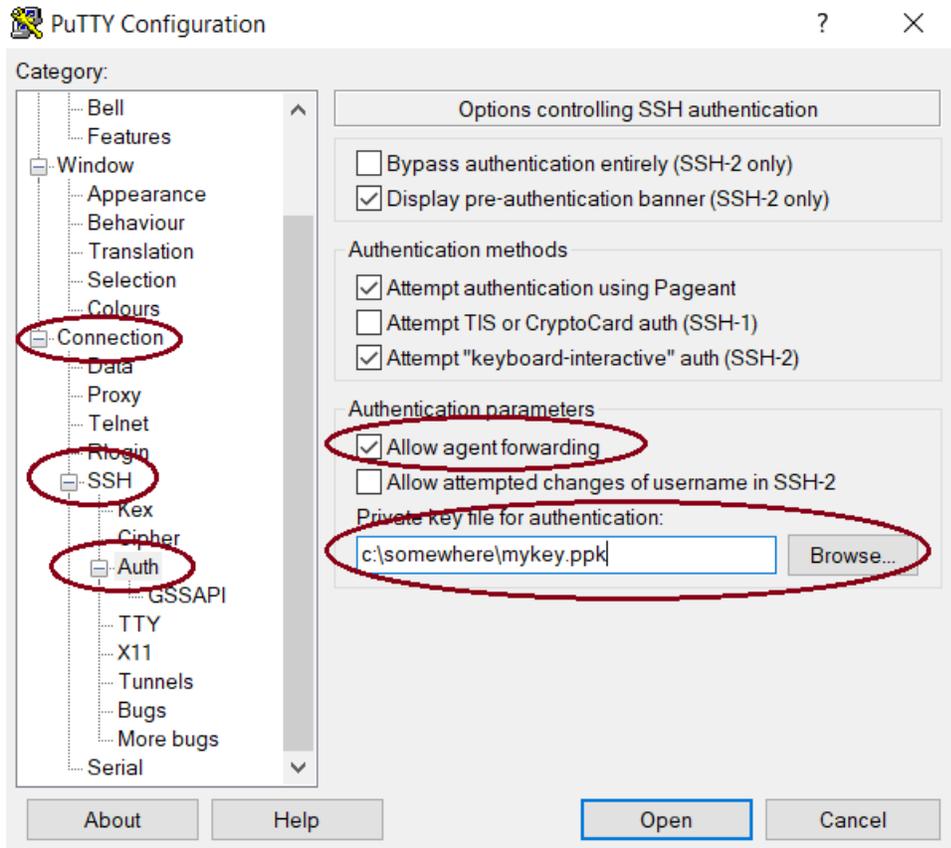
```
chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys
```

Now you should be able to access the server using the SSH keys.

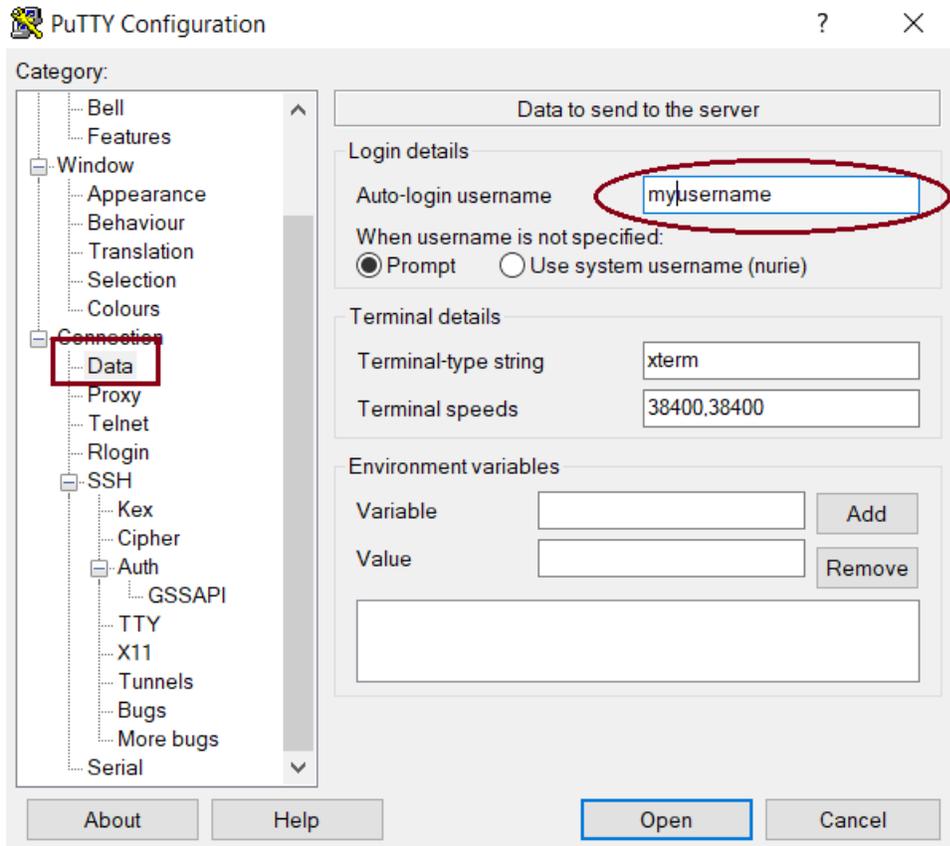
You can close Puttygen. If needed, you can always run puttygen again, create a new key, load an existing key (it will require the password you've configured with it), replace the password or copy the public key from it.

Access Using the SSH Keys

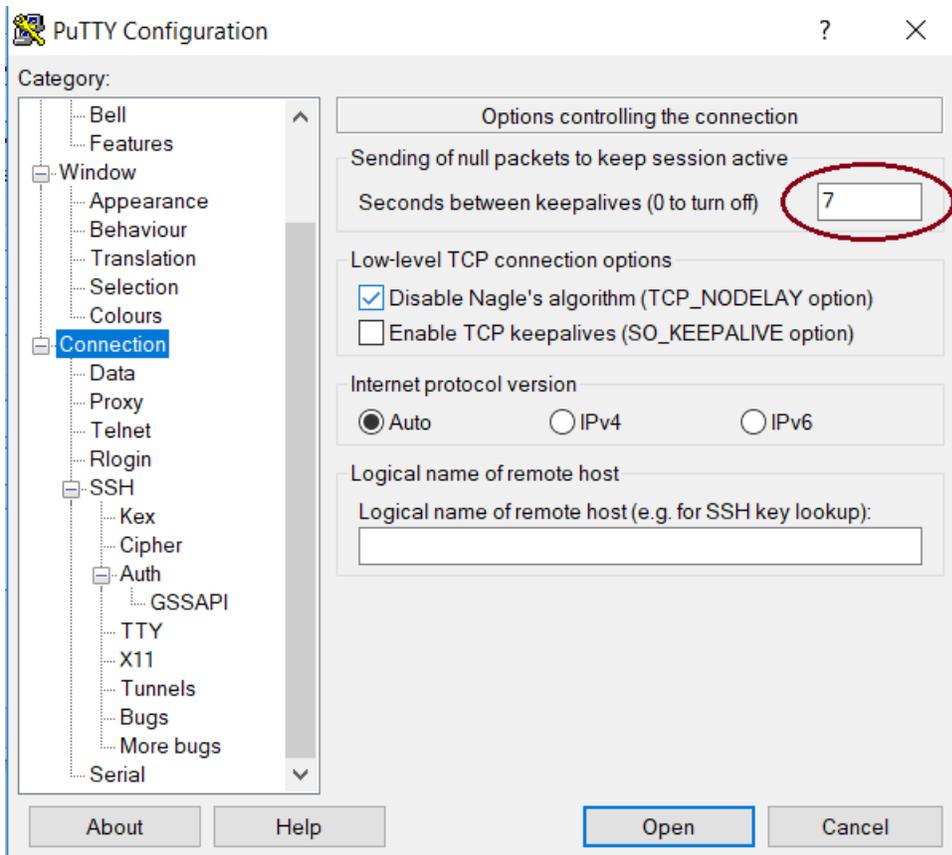
1. Open the application `Putty`. On the left side you will have a tree browser. Open “Connection”, “SSH”, and “Auth”. Configure as shown in the image below, browse the file system to select the private ssh key you've created earlier:



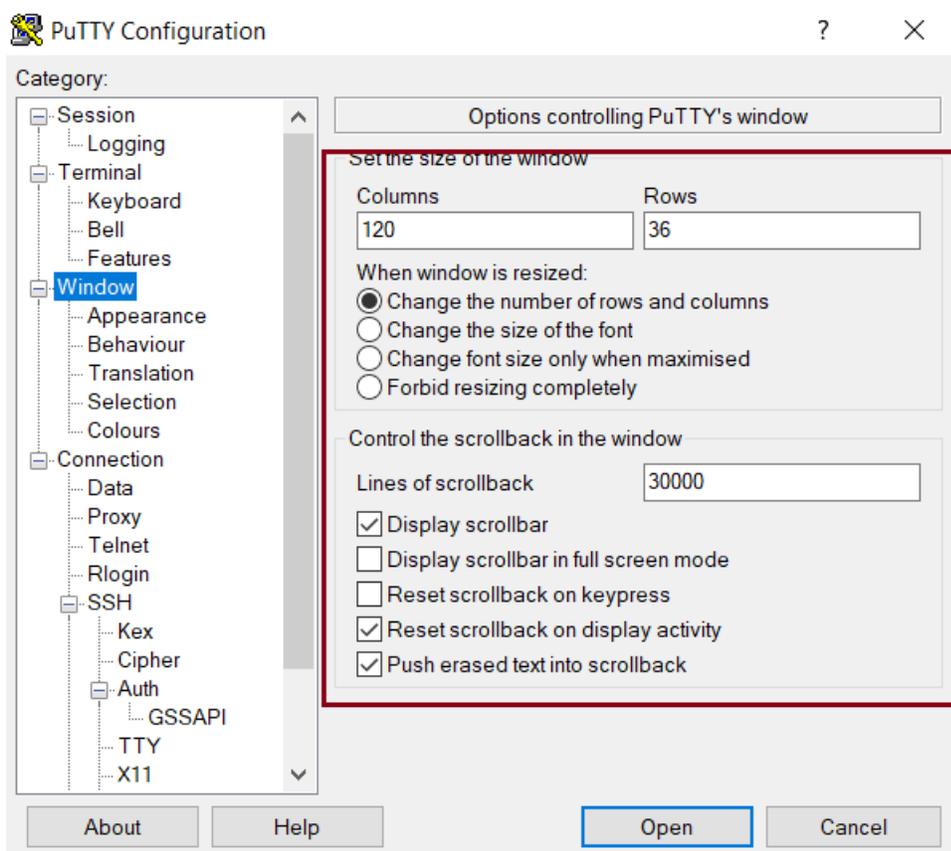
2. Next open the “Data” option and set the username you’ve created on the server:



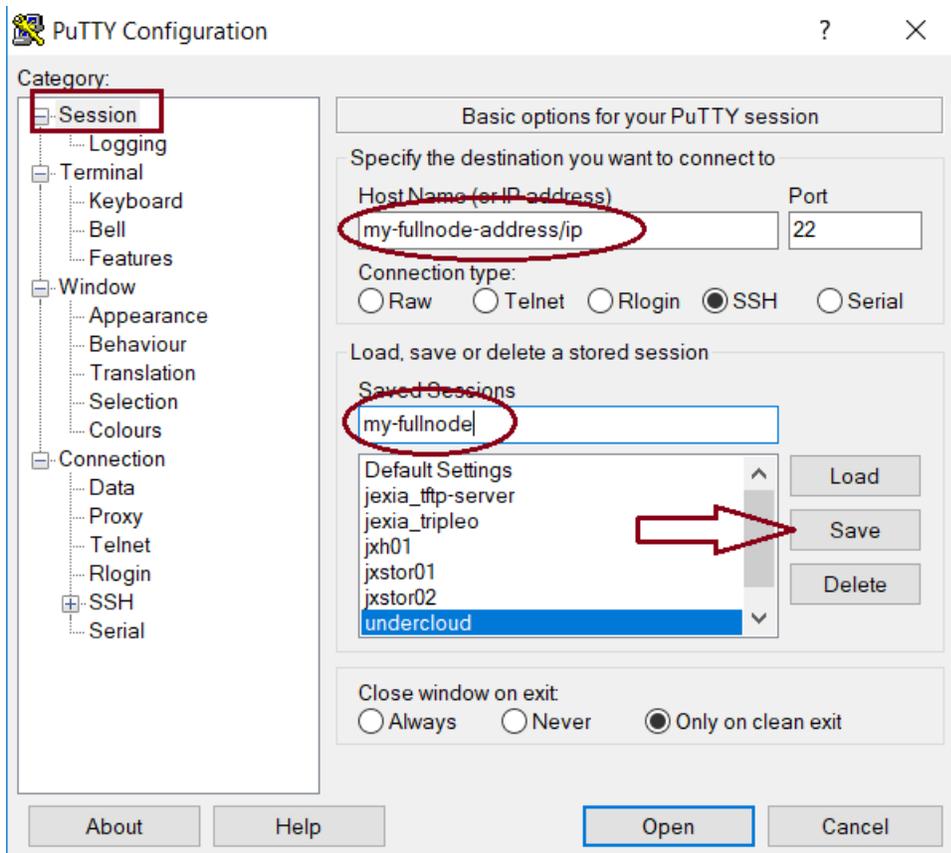
3. Then, on the “Connection”, set a keepalive value (5 or 7 is fine):



4. On the “Window”, set the Columns, Rows and Lines of scrollbar as shown here:



5. Now go to “Session” and set on the top your servers IP address (or hostname). In Saved Sessions choose a name and click save:



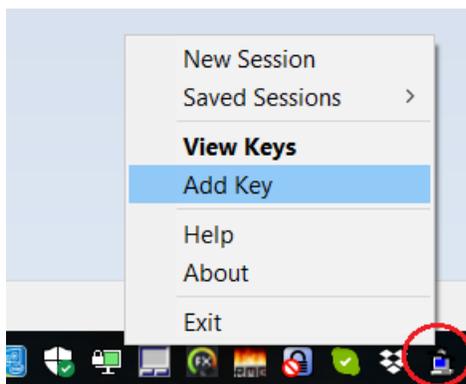
- Now, or any time you open Putty, you can select this saved session and click “Open”. This should connect you to the server. You will be asked to provide the SSH key password (not the user’s password from the server!)

Below is explained how to load the SSH private key to pagent – in which case you will not have to repeatedly enter the key’s password every time you connect to the server.

Adding SSH Key to Pagent

Pagent is a utility that was delivered with Putty. It loads the SSH private key into memory and allows you to connect to the server without having to enter the key’s password every time.

Once you open pagent you will find its icon on the task bar’s icons. Right clicking it opens a menu where you can select “Add keys”:



Browse the filesystem to select your private key. Enter the password, and that’s about it.

Now, everytime you connect to the server using Putty you should not be asked to enter the password again.

1.10.2 Disabling Password Authentication

In this part we will disable SSH password authentication to the server, thereby making it less susceptible to password brute forcing.

You need to run the following commands as user `root`, either by becoming root i.e. `sudo su` or prepend `sudo` to the commands e.g. `sudo systemctl restart sshd`.

Warning: Only follow these steps if you've successfully completed the previous chapter and can access your server using SSH keys!

Disable

Disable SSH password authentication:

```
grep -q "^PasswordAuthentication" /etc/ssh/sshd_config && sed -i 's/^
↪PasswordAuthentication.*/PasswordAuthentication no/g' /etc/ssh/sshd_config || echo
↪"PasswordAuthentication no" >> /etc/ssh/sshd_config
```

Restart SSH daemon:

```
systemctl restart sshd
```

If you want to test this, you need to make sure you unload the SSH keys from pageant (exit pageant), and manually connect to the server from Putty (not via the saved session - because the saved session has the keys already configured in it).

If all okay, you will be refused and not able to connect without SSH keys.

Enable

If you wish to re-enable the password authentication, run:

```
sed -i 's/^PasswordAuthentication.*/PasswordAuthentication yes/g' /etc/ssh/sshd_config
```

And restart sshd daemon:

```
systemctl restart sshd
```

1.10.3 Disabling SSH Root Access

Disabling SSH root access to your server makes it less likely to be hacked. In the previous steps you might have already enabled SSH key only access. That already means, that if root doesn't have any SSH keys configured, nobody will be able to access root via SSH.

Nonetheless, it is a good practice to disable the root account from being accessible via SSH.

Disable:

Python

```
sed -i 's/^PermitRootLogin.*/PermitRootLogin no/g' /etc/ssh/sshd_config
```

And restart ssh daemon:

```
systemctl restart sshd
```

To re-enabled root access:

```
sed -i 's/^PermitRootLogin.*/PermitRootLogin yes/g' /etc/ssh/sshd_config
```

And restat sshd daemon as shown above.

1.10.4 Using Alternative SSH Port

SSH by default uses port 22. That means that any hacker trying to force his way into a server will try hammering this port. One of the most simple ways to get rid of those attempts and make them useless is switching to an alternative port number.

There are more than 65k ports on a Linux system to choose from. No hacker is going to bother and try to find the SSH port if he doesn't find it on 22 to begin with. They much rather save time and skip to a different server where SSH is on port 22.

To change the port, a few things have to be done. The most important step is to ensure that you have some terminal/console access provided to you by your hosting provider. This is important in case you lock yourself out. Then you can still access the server and revert or fix any faults.

Warning: I'd like to repeat this again: make sure you have a terminal or console access to your server provided by the hosting provider. It is very important in case something in the configuration goes wrong and you cannot access using SSH anymore.

Firewall

Choose a port number (let's say 9922) and allow it through the firewall.

The following command have to be run as user root or by prefixing the commands with sudo e.g. `sudo yum install policycoreutils-python`.

CentOS

Run:

```
firewall-cmd --add-port=9922/tcp --zone=public --permanent && firewall-cmd --reload
```

And tell Selinux we want to use this port:

```
semanage port -a -t ssh_port_t -p tcp 9922
```

If the command gets an error that semanage was not found, make sure to install it and re-run it afterwards:

```
yum -y install policycoreutils-python
```

Ubuntu

Run:

```
ufw allow 9922/tcp
```

SSH Daemon

Edit the file `/etc/ssh/sshd_config` and find the line with `# Port 22`.

There might be a `#` before `Port` (or not). In any case, make sure to remove the `#` and any trailing spaces. Set the new port number:

```
Port 9922
```

Save the file and restart sshd daemon:

```
systemctl restart sshd
```

Your current SSH connection will not drop. But you should be able to see SSHD listening on the new port:

```
lsof -Pni|grep sshd
```

Next, configure your putty session (click “Load” when selecting your saved session, change the port number and click “Save”, then “Open”).

1.11 Fullnode IRI Configuration Utility

`iric` is a new tool I started working on in February 2018. It is currently under development, but ready to use should you find it helpful.

Many new users who have little experience with Linux have recently installed a fullnode using the playbook. Lacking experience with Linux makes it hard to manage the fullnode.

Some users find it exciting and want to learn more about Linux. The playbook’s documentation includes some basics such as configuring firewalls, security, checking logs, managing services etc.

For those who don’t have the time to take a dive into Linux, I started working on `iric` – a utility to help manage a full node. Its aim is to include tasks related to management of the fullnode.

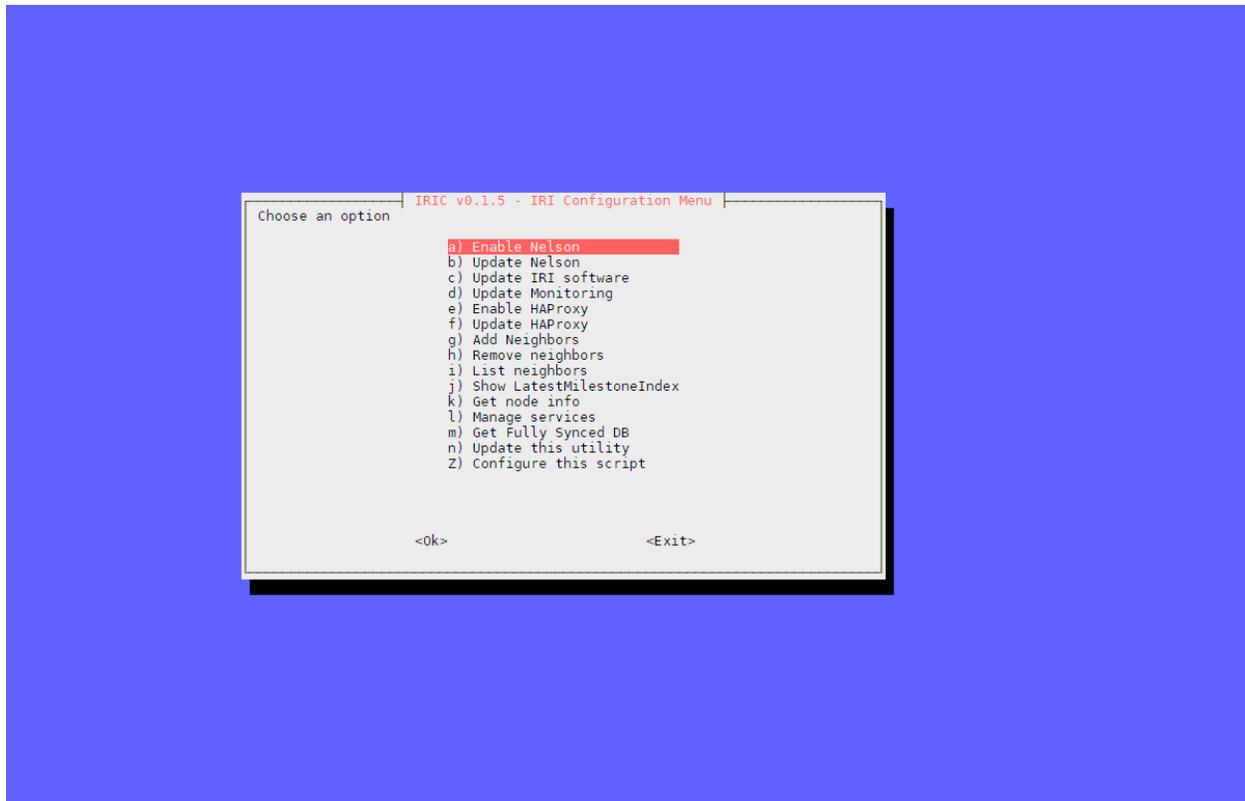
Instead of having to copy & paste long commands from the documentation, one can choose to use this menu-driven utility.

Feel free to use it. Feedback is most welcome.

If you don’t have `iric` installed (older playbook installations) you can add it by running:

```
cd /opt/iri-playbook/ && git pull && ansible-playbook -i inventory site.yml -v --  
↪tags=scripts,nbctl_config
```

To run the utility, type: `iric`. This will open it up:



Not much to explain here. Enjoy!

1.12 Troubleshooting

1.12.1 Upload Logs to Pastebin

Sometimes it helps to share your logs with someone who can help figure out problems.

First make sure you have the pastebin tool installed.

On **CentOS**: `yum install -y fpaste`

On **Ubuntu**: `apt-get install -y pastebinit`

Below are examples for Ubuntu and CentOS how to upload various files. You can tweak parameters as required.

Note that the long `sed` commands are there to hide IP addresses.

The command will return a URL link which you can share, that will open the logs in the browser.

**** DO NOT COPY PASTE BLINDLY, edit commands as required before execution! ****

Ubuntu Logs

Here are a few examples. You can change the log file name if required.

The two `sed` commands can be added in between any command to hide IP addresses:

```
sed 's/\([0-9]\{1,3\}\.\)\{3,3\}[0-9]\{1,3\}/x.x.x.x/g'|sed -r 's#:\[.*\]: ([0-9]+)#:\[xxxx:xxxx:xxxx:xxxx\]:\1#g'
```

```
# Example uploading last 200 lines of main syslog, hide IPv4 and IPv6 addresses
tail -200 /var/log/syslog | sed 's/\([0-9]\{1,3\}\.\.\)\{3,3\}[0-9]\{1,3\}/x.x.x.x/g
↵'|sed -r 's#:\[.*\]:([0-9]+)#:\[xxxx:xxxx:xxxx:xxxx\]:\1#g'| pastebinit -b_
↵pastebin.com -P

# Example uploading iri-playbook log
cat /tmp/iri-playbook-201801061902.log | pastebinit -b pastebin.com -P

# Example uploading last 200 lines of iota-pm service log
journalctl -u iota-pm --no-pager -n 200 | pastebinit -b pastebin.com -P

# Example uploading last 200 lines of iri service log
journalctl -u iri --no-pager -n 200 | pastebinit -b pastebin.com -P
```

CentOS Logs

Here are a few examples. You can change the log file name if required.

The two sed commands can be added in between any command to hide IP addresses:

```
sed 's/\([0-9]\{1,3\}\.\.\)\{3,3\}[0-9]\{1,3\}/x.x.x.x/g'|sed -r 's#:\[.
*\]:([0-9]+)#:\[xxxx:xxxx:xxxx:xxxx\]:\1#g'.
```

```
# Example uploading last 200 lines of main syslog, hide IPv4 and IPv6 addresses
tail -200 /var/log/messages | sed 's/\([0-9]\{1,3\}\.\.\)\{3,3\}[0-9]\{1,3\}/x.x.x.x/g
↵'|sed -r 's#:\[.*\]:([0-9]+)#:\[xxxx:xxxx:xxxx:xxxx\]:\1#g'| fpaste -P "yes"

# Example uploading iri-playbook log
cat /tmp/iri-playbook-201801061902.log | fpaste -P "yes"

# Example uploading last 200 lines of iota-pm service log
journalctl -u iota-pm --no-pager -n 200 | fpaste -P "yes"

# Example uploading last 200 lines of iri service log
journalctl -u iri --no-pager -n 200 | fpaste -P "yes"
```

1.12.2 How to Handle Git Conflicts

This is by no means a git tutorial, and the method suggested here has nothing to do with how one should be using git.

Background

It is simply the case that updates are applied to configuration files over time. A user might have configured values that might later conflict with new updates.

I was looking for a quick solution for users who are not familiar with Linux or git. One idea was to rename all the variable files adding the extension `.example` and using those as the “source”.

The other solution is the one I am presenting here.

Backup My Changes

If you run a `git pull` and receive a message about conflicts, e.g.:

```
error: Your local changes to the following files would be overwritten by merge:
    somefile
Please, commit your changes or stash them before you can merge.
Aborting
```

This means you've applied changes in files which have already been updated upstream.

The fastest answer is to use `git stash` to stash all the changes you've made:

```
git stash
```

This should allow you to run `git pull` without any errors. After that you can use `git stash apply` to get your changes back.

It is recommended not to edit the variable files in order to avoid such conflicts. You can better create “override” files [How to override playbook variables](#)

A longer route would be to identify those files which are in conflict:

```
git status
```

And view the changes you've applied:

```
git diff
```

You can run the following command which will backup the files you've changed and allow to pull the updated versions:

```
mkdir -p /tmp/my-changes && for f in $(git status|grep modified|awk {'print $3'});do
  ↪cp $f /tmp/my-changes/ ; git checkout -- $f ;done
```

This will copy any conflicting file into the directory `/tmp/my-changes`.

At this point you will not have any conflicts and be able to run `git pull`.

Apply Changes

The next step is to identify the changes. You can view the files that have been backed up using `ls -l /tmp/my-changes`.

For each file in that directory find its corresponding (new) updated file: `find -name filename`.

To view the differences run `diff /tmp/my-changes/my-old-file my-newfile`. The command's output might not be the prettiest; you can choose to handle the conflicts manually.

Once you are done applying your changes, you can proceed to run the playbook command you were about to apply.

1.12.3 HTTP Error 401 Unauthorized When Running Playbook

This is how the error would look like:

```
TASK [monitoring : create prometheus datasource in grafana]_
↪*****
fatal: [localhost]: FAILED! => {"changed": false, "connection": "close", "content": "
↪{"message\": \"Basic auth failed\"}", "content_length": "31", "content_type": "
↪application/json; charset=UTF-8", "date": "Fri, 29 Dec 2017 10:40:13 GMT", "json":
↪{"message": "Basic auth failed"}, "msg": "Status code was not [200, 409]: HTTP_
↪Error 401: Unauthorized", "redirected": false, "status": 401, "url": "http://
↪localhost:3000/api/datasources"}
    to retry, use: --limit @/opt/iri-playbook/site.retry

PLAY RECAP_
↪*****
```

This can happen for a number of reasons. It is most probably a password mismatch between what the playbook sees in `group_vars/all/iotapm.yml` under the value `iotapm_nginx_password` and perhaps the `iotapm_nginx_user` too.

Solution A

Try to correct this by checking the password which is currently configured in grafana:

```
grep ^admin /etc/grafana/grafana.ini
```

The result should look like:

```
admin_user = iotapm
admin_password = hello123
```

You can try to override the password when running the playbook, appending it to the end of the ansible command, e.g.:

```
ansible-playbook -i inventory -v site.yml --tags=monitoring_role -e iotapm_nginx_
↪password=hello123
```

Solution B

If Solution A doesn't work, there's a way to force-reset the password.

This solution also works if you haven't installed Grafana via this tutorial and cannot login.

1. Stop grafana-server:

```
systemctl stop grafana-server
```

2. Delete grafana's database:

```
rm -f /var/lib/grafana/grafana.db
```

3. Edit `/etc/grafana/grafana.ini`, set correct values for `admin_user` and `admin_password`.

4. Start grafana-server:

```
systemctl start grafana-server
```

Now you should be able to login to grafana.

1.12.4 Error Starting up Nelson After Upgrade

Checking nelson logs can reveal startup errors (e.g. `journalctl -u nelson --no-pager -n40`)

If you get an error that looks like this when starting up nelson:

```
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]: 20:57:40.241
↳16600::NODE terminating...
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]: Unhandled Rejection at:
↳Promise Promise {
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]: <rejected> Error:
↳"toString()" failed
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]: at stringSlice (buffer.
↳js:560:43)
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]: at Buffer.toString
↳(buffer.js:633:10)
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]: at FSReqWrap.
↳readFileAfterClose [as oncomplete] (fs.js:506:23) } reason: Error: "toString()"
↳failed
Jan 29 20:57:40 vmi111112.contaboserver.net nelson[3178]: at stringSlice (buffer.
↳js:560:43)
```

The nelson database might have become corrupt. You can remove it and it will re-create:

```
rm -rf /var/lib/nelson/data/neighbors.db
```

Start up nelson, and check the status again:

```
systemctl start nelson
```

Status:

```
systemctl status nelson
```

1.12.5 Error Starting or Restarting IRI

Examples of errors:

Hostname can't be null

If you get this message in the logs:

```
java.lang.IllegalArgumentException: hostname can't be null
```

It is most likely you have a typo in one (or more) of the neighbors in your configuration file, or the entire line is invalid.

Make sure all neighbors adhere to the format examples:

```
tcp://some-node.myserver.com:15600
tcp://10.20.30.40:14600
tcp://[2xxx:7xx:aaaf:111:2222:ff:ffff:xxxx]:12345
```

1.12.6 Fix Nginx

If you've tried to enable HTTPS (Let's Encrypt) via an automated script supporting Nginx and your Nginx is no longer working, follow these instructions on how to restore it:

```
wget -O /etc/nginx/sites-enabled/default https://gist.githubusercontent.com/nuriel77/
↪e847aa6dbb360d277a0313c983e35721/raw/a68e4528fe07a429284cc19b923d72d62a25d2c9/
↪default
```

And then restart nginx:

```
systemctl restart nginx
```

You can verify it is working via:

```
systemctl status nginx
```

It should be active.

1.12.7 Cannot Connect with Trinity to the Node

There are several things that could prevent Trinity from establishing a connection to your node.

Most importantly, you need to make sure you have configured the node with a valid SSL certificate and enabled HAProxy. This can be done using `iric` (enable HAProxy and then Enable HTTPS / Certificate). Make sure the process completes successfully.

A simple validation to see if your node is still serving a valid certificate is to open the URL on the browser, for example: `https://mynode.io:14267`. If you get a green padlock and no security warning, all should be fine (ignore the fact that the page shows “403 Forbidden”, that is expected when a browser is talking to the IRI port).

No Green Padlock

If you don't get the green padlock that indicates that the certificate is invalid. A good place to start is to issue the following command to see which certificate is configured on HAProxy:

```
grep "bind 0.0.0.0.*ssl" /etc/haproxy/haproxy.cfg
```

You should see something like:

```
bind 0.0.0.0:14267 ssl crt /etc/letsencrypt/live/cluster0.x-vps.com/haproxy.pem
```

Note the `/etc/letsencrypt/live/DOMAINNAME` <- the domain name should match the one that points to your node's IP address.

If there is a different certificate configured (e.g. `/etc/ssl/private/fullnode.crt.key`) you will have to re-run the process in `iric` to configure HTTPS. If this issue is recurring without you having done anything to modify the configuration, please contact `nuriel77` on discord.

Secure Connection Failed

If you don't get the green padlock and see a message in the browser containing the words: “Secure Connection Failed” and/or “SSL_ERROR_RX_RECORD_TOO_LONG”, your node was probably not configured with HTTPS. Please re-run the process in `iric` to configure HTTPS. If this issue is recurring without you having done anything to modify the configuration, please contact `nuriel77` on discord.

1.12.8 Apt cache Error When Upgrading Iric

On Ubuntu, when trying to upgrade `iric` you get an error containing something like:

```
fatal: [localhost]: FAILED! => {"changed": false, "msg": "Failed to update apt cache:
↳"}
```

Solution is to upgrade the Grafana repositories via this command:

```
cd /opt/iri-playbook && git pull && ansible-playbook -i inventory -v site.yml --
↳tags=monitoring_deps
```

Afterwards try to run the `iric` upgrade once more.

1.12.9 Can't Load Spent-Addresses-db Folder

This error may appear after having upgraded a node to 1.6.0-RELEASE:

```
com.iota.iri.IRI$IRILauncher - Welcome to IRI 1.6.0-RELEASE
Jan 12 00:45:26 ubuntu-4gb-nbg1-dc3-2 iri[5181]: Exception in thread "main" com.iota.
↳iri.service.spentaddresses.SpentAddressesException: Can't load spent-addresses-db_
↳folder
```

Solution should be to download a fully synced DB.

1.13 FAQ

1.13.1 How to override playbook variables

You might have noticed that many Ansible commands in the documentation use `-e somevar=value` to specify variables.

This variable declaration takes precedence over any other pre-defined variables.

An easy approach to override variables in the files found in `group_vars/all/` path is to override them.

The reason is that if you edit any of these files you risk a conflict when updates are pulled from the `iri-playbook` repository.

Overriding file variables

The files in `group_vars/all/` are read in alphabetic order.

For example: you have a file called `aaa.yaml` with the variable `test_var`:

```
test_var: 1234
```

and you have a file called `bbb.yaml`, also with the variable `test_var`:

```
test_var: abcd
```

When the playbook runs, it first reads the file `aaa.yaml` and then `bbb.yaml`. `test_var` ends up with the value `abcd`.

Best practice is to create a file starting with the letter z, for example `zzz-myenvironment.yaml` and in it define all the variables you want.

1.13.2 How to tell if my node is synced

You can check if your node is synced by looking at `iota-pm` GUI. Check if Latest Mile Stone Index and Latest Solid Mile Stone Index are equal:

Latest Milestone Index:

299371 (JLHZCLLXQQV9NGFSPS9YQZFUKEYBFZATCGVHJFZRTMMUDJANZGEB9BDKVMQSHAFBMMSZVH9MJUYWCZ9999)

Latest Solid Milestone Index:

299371 (JLHZCLLXQQV9NGFSPS9YQZFUKEYBFZATCGVHJFZRTMMUDJANZGEB9BDKVMQSHAFBMMSZVH9MJUYWCZ9999)

Tips: 3013

Transactions to Request: 49

Another option is to run the following command on the server's command line (make sure the port matches your IRI API port):

```
curl -s http://localhost:14265 -X POST -H 'X-IOTA-API-Version: 1' -H 'Content-Type: application/json' -d '{"command": "getNodeInfo"}' | jq '.latestSolidSubtangleMilestoneIndex, .latestMilestoneIndex'
```

This will output 2 numbers which should be equal.

Note: Above command will fail if you don't have `jq` installed. See below how to install it.

You can install `jq`:

Ubuntu: `apt-get install jq -y`

Centos: `yum install jq -y`

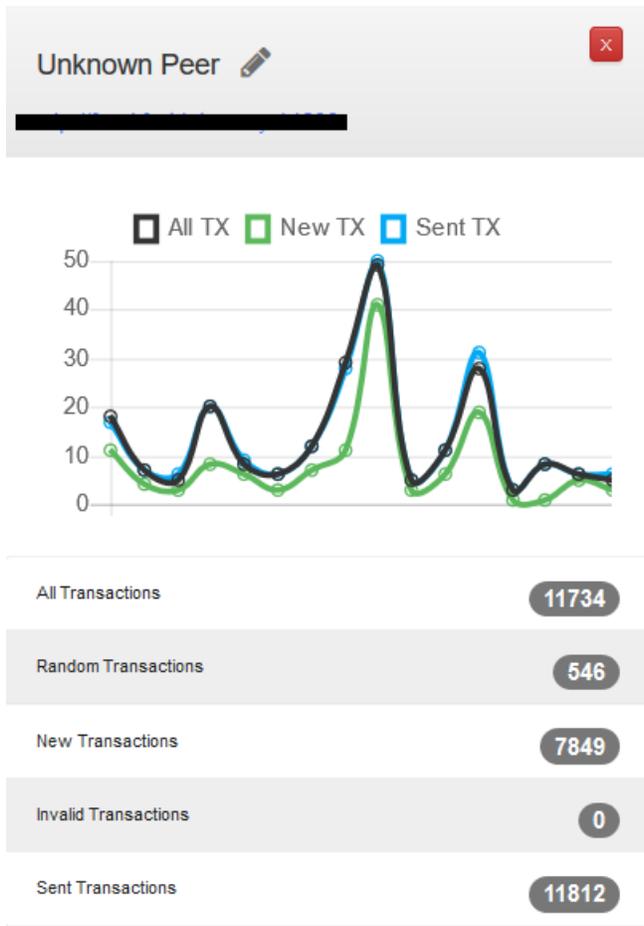
Alternatively, use python:

```
curl -s http://localhost:14265 -X POST -H 'X-IOTA-API-Version: 1' -H 'Content-Type: application/json' -d '{"command": "getNodeInfo"}' | python -m json.tool | egrep "latestSolidSubtangleMilestoneIndex|latestMilestoneIndex"
```

If you have problems getting in sync after a very long time, consider downloading a fully synced database as described here: *Where can I get a fully synced database to help kick start my node*

1.13.3 How do I tell if I am syncing with my neighbors

You can use IOTA Peer Manager. Have a look at the neighbors boxes. They normally turn red after a while if there's no sync between you and their node. Here's an example of a healthy neighbor, you can see it is also sending new transactions (green line) and the value of New Transactions increases in time:



1.13.4 Where to get the latest milestone index from

It used to be possible via the botbox on Slack. And since Slack is no longer in use, you can get it by running:

```
curl -s https://x-vps.com/lmsi | jq .
```

This is a value which is based on querying approximately 100 full nodes.

At time of writing, we are still waiting for the official botbox to be added to IOTA's Discord chat application.

1.13.5 Why is latestSolidSubtangleMilestoneIndex always behind latestMilestoneIndex

This is probably the most frequently asked question.

At time of writing, and to the best of my knowledge, there is not one definitive answer. There are probably various factors that might keep the Solid milestone from ever reaching the latest one and thus remaining not fully synced.

I have noticed that this problem exacerbates when the database is relatively large (5GB+). This is mostly never a problem right after a snapshot, when things run much smoother. This might also be related to ongoing "bad" spam

attacks directed against the network.

Some things to try:

- Check your IRI logs. Some case in the past have shown a component failing (e.g. ZMQ) which caused milestone to get stuck. The logs might help identify errors. You can use `iric` to view logs (Manage Service->IRI->View log). If you don't have `iric` you can install it *Fullnode IRI Configuration Utility*.
- If there's nothing seen in IRI logs (no errors), check other services.
- *How to get my node swap less*
- *Where can I get a fully synced database to help kick start my node*
- Finding "healthier" neighbors. This one is actually often hard to ascertain – who is "healthy", probably other fully synced nodes.

1.13.6 How to get my node swap less

You can always completely turn off swap, which is not always the best solution. Using less swap (max 1GB) can be helpful at times to avoid some OOM killers (out-of-memory).

As a simple solution you can change the "swappiness" of your linux system. I have a 8GB 4 core VPS, I lowered the swappiness down to 1. You can start with a value of 10, or 5. Run these two commands:

```
echo "vm.swappiness = 1" >>/etc/sysctl.conf
```

and:

```
sysctl -p
```

You might need to restart IRI in order for it to adapt to the new setting. Try to monitor the memory usage using `free -m`, swap in particular, e.g.:

```
free -m
      total        used         free       shared  buff/cache   available
Mem:    7822        3331           692         117       3798       4030
Swap:   3815           1        3814
```

You'll see that in this example nothing is being used. If a large "used" value appears for Swap, it might be a good idea to lower the value and restart IRI.

1.13.7 Where can I get a fully synced database to help kick start my node

For the sake of the community, I regularly create a copy of the database snapshot files. NOTE: The current database only contains the snapshot files. The full DB is no longer available since IRI version 1.6.0.

You can use the `iric` tool to download and install the database *Fullnode IRI Configuration Utility*, or update manually using the following instructions:

- The full command will only work if you've installed your full node using this tutorial/playbook.

```
systemctl stop iri && rm -rf /var/lib/iri/target/{mainnetdb*,mainnet.snapshot*} &&
↪ mkdir -p /var/lib/iri/target && cd /var/lib/iri/target && wget -O - https://x-vps.
↪ com/iota.db.tgz | tar zxv && chown iri.iri /var/lib/iri -RL && systemctl start iri
```

NOTE If there has been a corruption in the `spent-addresses-db` directory, you might also like to first run the command `rm -rf /var/lib/iri/target/spent-addresses-*` before running the above command.

1.13.8 I try to connect the light wallet to my node but get connection refused

Note: Please use Trinity to connect to nodes. Usage of the light wallet is discouraged.

There are commonly two reasons for this to happen:

If your full node is on a different machine from where the light wallet is running from, there might be a firewall between, or, your full node is not configured to accept external connections.

See *Full Node Remote Access*

1.14 Uninstall

It is possible to remove the services and configuration files installed by the playbook.

Warning: It is not possible to remove everything installed by the playbook. For example, some packages might have already been installed by the user prior to running the playbook. In addition, enabling of firewalls, main nginx file configuration files and some additional essentials are not reverted/removed.

1. In order to run the uninstaller, please become root via `sudo su -`.
2. Run:

```
cd /opt/iri-playbook && ansible-playbook -i inventory site.yml --tags=uninstall -e_  
↪uninstall_playbook=yes
```

1.15 Command Glossary

This is a collection of most command commands to come in handy.

1.15.1 Check IRI's node status

```
curl -s http://localhost:14265 -X POST -H 'X-IOTA-API-Version: someval' -H 'Content-  
↪Type: application/json' -d '{"command": "getNodeInfo"}' | jq
```

Same as above but extract the milestones

```
curl -s http://localhost:14265 -X POST -H 'X-IOTA-API-Version: 1' -H 'Content-  
↪Type: application/json' -d '{"command": "getNodeInfo"}'|python -m json.tool|egrep  
↪"latestSolidSubtangleMilestoneIndex|latestMilestoneIndex"
```

1.15.2 Add neighbors

This is the nbctl script that shipped with this installation (use it with `-h` to get help):

```
nbctl -a -n tcp://1.2.3.4:12345 -n tcp://4.3.2.1:4321
```

1.15.3 Remove neighbors

This is the nbctl script that shipped with this installation (use it with -h to get help):

```
nbctl -r -n tcp://1.2.3.4:12345 -n tcp://4.3.2.1:4321
```

1.15.4 Check iri and iota-pm ports listening

```
lsof -Pni|egrep "iri|iotapm"
```

1.15.5 Check all ports on the node

```
lsof -Pni
```

1.15.6 Opening a port in the firewall

In CentOS:

```
firewall-cmd --add-port=14265/tcp --zone=public --permanent && firewall-cmd --reload
```

In Ubuntu:

```
ufw allow 14265/tcp
```

1.15.7 Checking memory usage per application

This is the ps_mem script that shipped with this installation. If you don't have it you can see total memory usage using `free -m`.

```
ps_mem
```

1.15.8 Checking system load and memory usage

All Linux systems have `top`, but there's a nicer utility called `htop`.

You might need to install it:

```
On Ubuntu: apt-get install htop -y  
On CentOS: yum install htop -y
```

Then run `htop`

Note: If 'htop' is not available in CentOS you need to install 'epel-release' and try again, i.e. 'yum install epel-release -y'

1.16 Appendix

This chapter includes additional configuration options and/or general systems configuration.

It is meant for more advanced usage.

1.16.1 Using Fully Qualified Domain Name for my server

This requires that you have set up DNS service to point a fully qualified domain name to your server's IP address.

For example, `x-vps.com` points to 185.10.48.110 (if you simply `ping x-vps.com` you will see the IP address).

Instead of using the ports e.g. 8811 and 5555 with IP combination, we can use a FQDN, e.g. `pm.example.com` to reach peer manager on our server.

Note that by doing this you will not be able to re-use a let's encrypt free certificate. Unless you request a certificate per domain, or purchase a star certificate.

In this chapter we are going to configure nginx to serve IOTA Peer Manager and Grafana on port 443 (HTTPS), while using a fully qualified domain name.

You should be able to create subdomains for your main domain name. For example, if your FQDN is "example.com", you can create in your DNS service an entry for:

```
pm.example.com
```

and:

```
grafana.example.com
```

Here's what you have to change:

For Peer Manager, edit the file `/etc/nginx/conf.d/iotapm.conf`:

```
upstream iotapm {
    server 127.0.0.1:8011;
}

server {
    listen 443 ssl http2;
    server_name pm.example.com;
    server_tokens off;

    include /etc/nginx/conf.d/ssl.cfg;

    auth_basic "Restricted";
    auth_basic_user_file /etc/nginx/.htpasswd;

    location / {
        proxy_pass http://iotapm;
    }
}
```

Of course, don't forget to replace `pm.example.com` with your own FQDN e.g. `pm.my-fqdn.com`.

Now, test nginx is okay with the change:

```
nginx -t
```

Output should look like this:

```
# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Then, reload nginx configuration:

```
systemctl reload nginx
```

You should be able to point your browser to `https://pm.my-fqdn.com` and see the Peer Manager.

For **Ubuntu** you will have to allow http port in ufw firewall:

```
ufw allow http
```

For **Centos**:

```
firewall-cmd --add-service=https --permanent --zone=public && firewall-cmd --reload
```

The same can be done for grafana `/etc/nginx/conf.d/grafana.conf`:

```
upstream grafana {
    server 127.0.0.1:3000;
}

server {
    listen 443 ssl http2;
    server_name grafana.example.com;
    server_tokens off;

    include /etc/nginx/conf.d/ssl.cfg;

    location / {
        proxy_pass http://grafana;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Again, test nginx: `nginx -t` and reload nginx: `systemctl reload nginx`.

Now you should be able to point your browser to `https://grafana.my-fqdn.com`.

1.16.2 Reverse Proxy for IRI API (wallet)

If you read the two chapters above about configuring nginx to support FQDN or HTTPS you might be wondering whether you should reverse proxy from the web server to IRI API port (for wallet connections etc).

`iri-playbook` installs HAProxy with which you can reverse proxy to IRI API port and benefit from logging and security policies. In addition, you can add a HTTPS certificate. IOTA's Trinity wallet requires nodes to have a valid SSL certificate.

See [Running IRI API Port Behind HAProxy](#) on how to enable HAProxy for wallet via reverse proxy and how to enable HTTPS(SSL) for it.

1.16.3 Sending Alert Notifications

Since release v1.1 a new feature has been introduced to support alerting.

Warning: This is considered an advanced feature. Configuration hereof requires some basic Linux and system configuration experience.

Note: To edit files you can use `nano` which is a simple editor. See [Using Nano to Edit Files](#) for instructions.

TL;DR version

1. Edit the file `/opt/prometheus/alertmanager/config.yml` using `nano` or any other editor.
2. Find the following lines:

```
# Send using postfix local mailer
# You can send to a gmail or hotmail address
# but these will most probably be put into junkmail
# unles you configure your DNS and the from address
- name: email-me
  email_configs:
  - to: root@localhost
    from: alertmanager@test001
    html: '{{ template "email.tmpl" . }}'
    smarthost: localhost:25
    send_resolved: true
```

3. Replace the email address in the line: `- to: root@localhost` with your email address.
4. Replace the email address in the line `from: alertmanager@test001` with your node's name, e.g: `alertmanager@fullnode01`.
5. Save the file (in `nano` CTRL-X and confirm 'y')
6. Restart `alertmanager`: `systemctl restart alertmanager`

Note

Emails generated by your server will most certainly end up in junk mail. The reason being that your server is not configured as verified for sending emails.

You can, alternatively, try to send emails to your gmail account if you have one (or any other email account).

You will find examples in the `/opt/prometheus/alertmanager/config.yml` on how to authenticate.

For more information about `alertmanager`'s configuration consult the [documentation](#).

Configuration

The monitoring system has a set of default alerting rules. These are configured to monitor various data of the full node.

For example:

- CPU load high
- Memory usage high
- Swap usage high
- Disk space low
- Too few or too many neighbors
- Inactive neighbors
- Milestones sync

Prometheus is the service responsible for collecting metrics data from the node's services and status.

Alert Manager is the service responsible for sending out notifications.

Configuration Files

It is possible to add or tweak existing rules:

Alerts

The alerting rules are part of Prometheus and are configured in `/etc/prometheus/alert.rules.yml`.

Note: Changes to Prometheus's configuration requires a restart of prometheus.

Notifications

The configuration file for alertmanager can be found in `/opt/prometheus/alertmanager/config.yml`.

This is where you can **set your email address and/or slack channel** (not from iota!) to where you want to send the notifications.

The email template used for the emails can be found in `/opt/prometheus/alertmanager/template/email.tmpl`.

Note: Changes to Alert Manager configuration files require a restart of alertmanager.

Controls

Prometheus can be controlled via `systemctl`, for example:

```
To restart: systemctl restart prometheus
To stop: systemctl stop prometheus
Status: systemctl status prometheus
Log: journalctl -u prometheus
```

The same can be done with `alertmanager`.

For more information see [Documentation Prometheus Alertmanager](#)

1.16.4 Restart IRI On Latest Subtangle Milestone Stuck

A trigger to restart IRI restart when the Latest Subtangle Milestone Stuck is stuck has been added to `alertmanager`.

If you don't have alert manager or had it installed before this feature was introduced, see *Upgrading the Playbook to Get the Feature*.

Warning: This feature is disabled by default as this is not considered a permanent or ideal solution. Please, first try to download a fully synced database as proposed in the [faq](#), or try to find “healthier” neighbors.

Enabling the Feature

Log in to your node and edit the `alertmanager` configuration file: `/opt/prometheus/alertmanager/config.yml`.

You will find the following lines:

```
# routes:
# - receiver: 'executor'
# match:
#   alertname: MileStoneNoIncrease
```

Remove the `#` comments, resulting in:

```
routes:
- receiver: 'executor'
  match:
    alertname: MileStoneNoIncrease
```

Try not to mess up the indentation (should be 2 spaces to begin with).

After having applied the changes, save the file and restart `alertmanager`: `systemctl restart alertmanager`.

What will happen next is that the service called `prom-am-executor` will be called and trigger a restart to IRI when the Latest Subtangle Milestone is stuck for more than 30 minutes.

Note: This alert-trigger is set to only execute if the Latest Subtangle Milestone is stuck and not equal to the initial database milestone.

Disabling the Feature

A quick way to disable this feature:

```
systemctl stop prom-am-executor && systemctl disable prom-am-executor
```

To re-enable:

```
systemctl enable prom-am-executor && systemctl start prom-am-executor
```

Configuring the Feature

You can choose to tweak some values for this feature, for example how long to wait on stuck milestones before restarting IRI:

Edit the file `/etc/prometheus/alert.rules.yml`, find the alert definition:

```
# If latest subtangle milestone doesn't increase for 30 minutes
- alert: MileStoneNoIncrease
  expr: increase(iota_node_info_latest_subtangle_milestone[30m]) == 0
    and iota_node_info_latest_subtangle_milestone != 243000
  for: 1m
  labels:
    severity: critical
  annotations:
    description: 'Latest Subtangle Milestone increase is {{ $value }}'
    summary: 'Latest Subtangle Milestone not increasing'
```

The line that denotes the time: `increase(iota_node_info_latest_subtangle_milestone[30m]) == 0` – here you can replace the `30m` with any other value in the same format (e.g. `1h`, `15m` etc...)

If any changes to this file, remember to restart prometheus: `systemctl restart prometheus`

Upgrading the Playbook to Get the Feature

If you installed the playbook before this feature was release you can still install it.

1. Enter the `iri-playbook` directory and pull new changes:

```
cd /opt/iri-playbook && git pull
```

If this command breaks, it means that you have conflicting changes in one of the configuration files. See [How to Handle Git Conflicts](#) on how to apply new changes (or hit me up on Discord or github for assistance: @nuriel77)

2. **WARNING**, this will overwrite changes to your monitoring configuration files if you had any manually applied! Run the playbook's monitoring role:

```
ansible-playbook -i inventory -v site.yml --tags=monitoring_role -e overwrite=true
```

3. **If** the playbook fails with 401 authorization error (probably when trying to run prometheus grafana datasource), you will have to re-run the command and supply your web-authentication password together with the command:

```
ansible-playbook -i inventory -v site.yml --tags=monitoring_role -e overwrite=true -e_
↪iota_pm_nginx_password="mypassword"
```

1.16.5 Configuring Multiple Nodes for Ansible

Using the Ansible playbook, it is possible to configure multiple full nodes at once.

How does it work?

Basically, following the manual installation instructions should get you there: *Installation*.

This chapter includes some information on how to prepare your nodes.

Overview

The idea is to clone the `iri-playbook` repository onto one of the servers/nodes, configure values and run the playbook.

The node from where you run the playbook will SSH connect to the rest of the nodes and configure them. Of course, it will also become a full node by itself.

SSH Access

For simplicity, let's call the node from where you run the playbook the "master node".

In order for this to work, you need to have SSH access to all nodes from the master node. This guide is based on user `root` access. There is a possibility to run as a user with privileges and become root, but we will skip this for simplicity.

Assuming you already have SSH access to all the nodes (using password?) let's prepare SSH key authentication which allows you to connect without having to enter a password each time.

Make sure you are root `whoami`. If not, run `sudo su -` to become root.

Create New SSH Key

Let's create a new SSH key:

```
ssh-keygen -b 2048 -t rsa
```

You will be asked to enter the path (allow the default `/root/.ssh/id_rsa`) and password (for simplicity, just click 'Enter' to use no password).

Output should look similar to this:

```
# ssh-keygen -b 2048 -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:tCmiLASAsDLPahH3hcI0s0TKDCXg/QwQukVQZCHL3Ok root@test001
The key's randomart image is:
+---[RSA 2048]-----+
|#%/..                |
|@%*=O.                |
|X*O*.                 |
|+*. + .o              |
```

(continues on next page)

(continued from previous page)

```

|o.oE.o. S      |
|.o . . .      |
|. o            |
|. .           |
|              |
+-----[SHA256]-----+

```

The generated key is the default key to be used by SSH when authenticating to other nodes (`/root/.ssh/id_rsa`).

Copy SSH Key Identity

Next, we copy the public key to the other nodes:

```
ssh-copy-id -i /root/.ssh/id_rsa root@other-node-name-or-ip
```

Given that you have root SSH access to the other nodes, you will be asked to enter a password, and possibly a question about host authenticity.

Output should look like:

```

# ssh-copy-id root@other-node-name-or-ip
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host 'node-name (10.10.1.1)' can't be established.
ECDSA key fingerprint is SHA256:4QAhCxlDhXR2bWes4uSVGl7ZAKiVXqgNT7geWAS043M.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
↳any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
↳now it is to install the new keys
root@other-node-name-or-ip's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'root@other-node-name-or-ip'"
and check to make sure that only the key(s) you wanted were added.

```

Perform the authentication test, e.g `ssh 'root@other-node-name-or-ip'`. This should work without a password.

Run the `ssh-copy-id -i /root/.ssh/id_rsa root@other-node-name-or-ip` for each node you want to configure.

Once this is done you can use Ansible to configure these nodes.

1.16.6 Using Nano to Edit Files

Nano is a linux editor with which you can easily edit files. Of course, this is nothing like a graphical editor (e.g. notepad) but it does its job.

Most Linux experts use `vi` or `vim` which is much harder for beginners.

First, ensure you have nano installed:

- On **Ubuntu**: `apt-get install nano -y`
- On **CentOS**: `yum install nano -y`

Next, you can use nano to create a new file or edit an existing one. For example, we want to create a new file `/tmp/test.txt`, we run:

```
nano /tmp/test.txt
```

Nano opens the file and we can start writing. Let's add the following lines:

```
IRI_NEIGHBORS="tcp://just-testing.com:13000 tcp://testing:15600"
```

Instead of writing this, you can copy paste it. Pasting can be done using right mouse click or **SHIFT-INSERT**.

To save the file you can click **F3** or, to exit and save you can click **CTRL-X**, if any modifications it will ask you if to save the file.

After having saved the file, you can run `nano /tmp/test.txt` again in order to edit the existing file.

Note: Please check [Nano's Tutorial](#) for more information.

1.16.7 Running IRI API Port Behind HAProxy

The IRI API port can be configured to be accessible via HAProxy. The benefits in doing so are:

- Logging
- Whitelist/blacklisting
- Password protection
- Rate limiting per IP, or per command
- Denying invalid requests

To get it configured and installed you can use `iric` or run:

```
cd /opt/iri-playbook && git pull && ansible-playbook -i inventory -v site.yml --  
→tags=iri_ssl,loadbalancer_role -e '{"lb_bind_addresses": ["0.0.0.0"]}' -e_  
→overwrite=yes
```

Please read this **important information**:

The API port will be accessible on **14267** by default.

Note that if you have previously enabled IRI with `--remote` option or `API_HOST = 0.0.0.0` you can disable those now. HAProxy will take care of that.

In addition, the **REMOTE_LIMIT_API** in the configuration files are no longer playing any role. HAProxy has taken control over the limited commands.

To see the configured denied/limited commands see `group_vars/all/lb.yml` or edit `/etc/haproxy/haproxy.cfg` after installation. The regex is different from what you have been used to.

Rate Limits

HAProxy enables rate limiting. In some cases, if you are loading a seed which has a lot of transactions on it, HAProxy might block too many requests.

One solution is to increase the rate limiting values in `/etc/haproxy/haproxy.cfg`. Find those lines and set the number accordingly:

```
# dynamic stuff for frontend + raise gpc0 counter
tcp-request content track-sc2 src
acl conn_rate_abuse sc2_conn_rate gt 250
acl http_rate_abuse sc2_http_req_rate gt 400
acl conn_cur_abuse sc2_conn_cur gt 21
```

Don't forget to restart HAProxy afterwards: `systemctl restart haproxy`.

Enabling HTTPS for HAProxy

To enable HTTPS for haproxy find the option in the main menu of `iric`. It will enable HAProxy to serve the IRI API on port 14267 with HTTPS (Warning: this will override any manual changes you might have applied to `/etc/haproxy/haproxy.cfg` previously).

The option will also allow you to configure all the services served via nginx to use the certificate (Grafana, IOTA Peer Manager etc)

Note: If you previously used a script to configure Let's Encrypt with Nginx and your Nginx is no longer working, please follow the instructions at [Fix Nginx](#)

1.16.8 Installation Options

This is an explanation about the select-options provided by the fully automated installer.

Nelson

Nelson is a software which enabled auto-peering for IRI (finding neighbors automatically).

If Nelson is not used, neighbors have to be manually maintained (default).

You can read more about it [here](#).

HAproxy

HAProxy is a proxy/load-balancer. In the context of this installation it can be enabled to serve the IRI API port.

You can read more about it here: [Running IRI API Port Behind HAProxy](#).

Monitoring

The monitoring refers to installation of:

- Prometheus (metrics collector)
- Alertmanager (trigger alerts based on certain rules)
- Grafana (Metrics dashboard)
- Iota-prom-exporter (IRI full node metrics exporter for Prometheus)

It is recommended to install those to have a full overview of your node's performance.

ZMQ Metrics

IRI can provide internal metrics and data by exposing ZeroMQ port (locally by default). If enabled, this will allow the `iota-prom-exporter` to read this data and create additional graphs in Grafana (e.g. transactions confirmation rate etc).

1.16.9 Upgrade IRI and Remove Existing Database

(option #3 from the [IOTA Snapshot Blog](#))

A snapshot of the database normally involves a new version of IRI. This is also the case in the upcoming snapshot of April 29th, 2018.

Here are the steps you should follow in order to get a new version of IRI and remove the old database:

Run the following commands as user `root` (you can run `sudo su` to become user `root`).

1. Stop IRI:

```
systemctl stop iri
```

2. Remove the existing database:

```
rm -rf /var/lib/iri/target/mainnet*
```

3. Run `iric` the command-line utility. Choose “Update IRI Software”. This will download the latest version and restart IRI.

If you don't have `iric` installed, you can refer to this chapter on how to upgrade IRI manually [Upgrade IRI](#).

1.16.10 Upgrade IRI and Keep Existing Database

(option #2 from the [IOTA Snapshot Blog](#))

If you want to keep the existing database, the instructions provided by the IF include steps to compile the RC version (v1.4.2.4_RC) and apply a database migration tool.

To make this process easy, I included a script that will automate this process. This script works for both CentOS and Ubuntu (but **only** for `iri-playbook` installations).

You will be asked if you want to download a pre-compiled IRI from my server, or compile it on your server should you choose to do so.

Please read the warning below and use the following command (as root) in order to upgrade to 1.4.2.4_RC and keep the existing database:

```
bash <(curl -s https://x-vps.com/get_iri_rc.sh)
```

Warning: This script will only work with installations of the `iri-playbook`. I provide this script to assist, but I do not take any responsibility for any damages, loss of data or breakage. By running this command you agree to the above and you take full responsibility.

For assistance and questions you can find help on IOTA's `#fullnodes` channel (discord).

1.16.11 Upgrade IRI to the Local Snapshot Version

At time of writing the database has grown to 51GB+ (compressed). It becomes harder to share the database copy with node operators.

I decided to help iri-playbook users to upgrade their nodes to the 1.6.0-RC12 version of IRI. (note that it **is not** an official release).

This will allow nodes with smaller HDD space to keep their nodes up and running.

Please run the following command (become root) to get and run the upgrade script:

```
bash <(curl -s https://raw.githubusercontent.com/nuriel77/iri-playbook/master/upgrade-
↳ local-snapshots.sh)
```

This should get the node configured, download the snapshot meta and state files and restart IRI.

Note: The upgrade script does not support the dockerized version of the iri-playbook

Configure Local Snapshot Settings

The default values should work fine. Nevertheless, if you would like to configure the settings `iric` can be used to edit the configuration file:

Run `iric` and select n) Configure Files, then b) IRI INI Config.

Under the line with `; Local Snapshots Settings` you will find the settings for local snapshots configuration.

1.17 Disclaimer

- This wiki is based on the [Ansible-playbook repository](#). Please review *The Requirements* before installing.
- Installer is meant to be installed on a clean OS. I do not take any responsibility for having installed this on a system with already existing software and/or mission critical services.
- You are responsible for the security of your server. This includes using strong passwords and storing them safely. This wiki includes information on security hardening. Please make sure you follow the steps in [ref:securityHardening](#) to improve your full node's security.
- Refer to [Appendix](#) for extra configuration options for your full node.
- You are responsibility for add-on software such as Nelson and Field, support which can be provided on the respective channels on IOTA's Discord.
- I am not associated with the IOTA foundation. I am simply an enthusiastic community member.

1.18 Donations

If you liked this tutorial, and would like to leave a donation you can use this IOTA address:

```
JFYIHZQOPCRSLKIYHTWRSIR9RZELTZKHNZFHGWXAPCQIEBNJSZFIWMSBGAPDKZZZGFNTAHBLGNPRRQIZHDFNPQPPWGC
```

Thanks!