Python

Jul 21, 2020

Contents

Welcome to IRI-playbook full node installer!

NOTE: **Please consider switching to Hornet-playbook <https://github.com/nuriel77/hornet-playbook>**

To install your node go to *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

This Installer Includes

- Fully automated installation
- Configuration of firewall and security
- All services running in Docker containers
- Automatically configure the java memory limits based on your system's RAM
- IOTA Peer Manager to manage neighbors
- `iric`, a menu-driven utility to help manage the node
- Serve IOTA Peer Manager and Graphs password protected via HTTPS (secure)
- IRI metrics and graphs, created by Chris Holliday's IOTA Exporter.
- IOTA Caddy PoW middleware, thanks to Luca Moser https://github.com/luca-moser/iotacaddy/blob/master/IOTA.md.

Please star the playbook's repository on github: iri-playbook

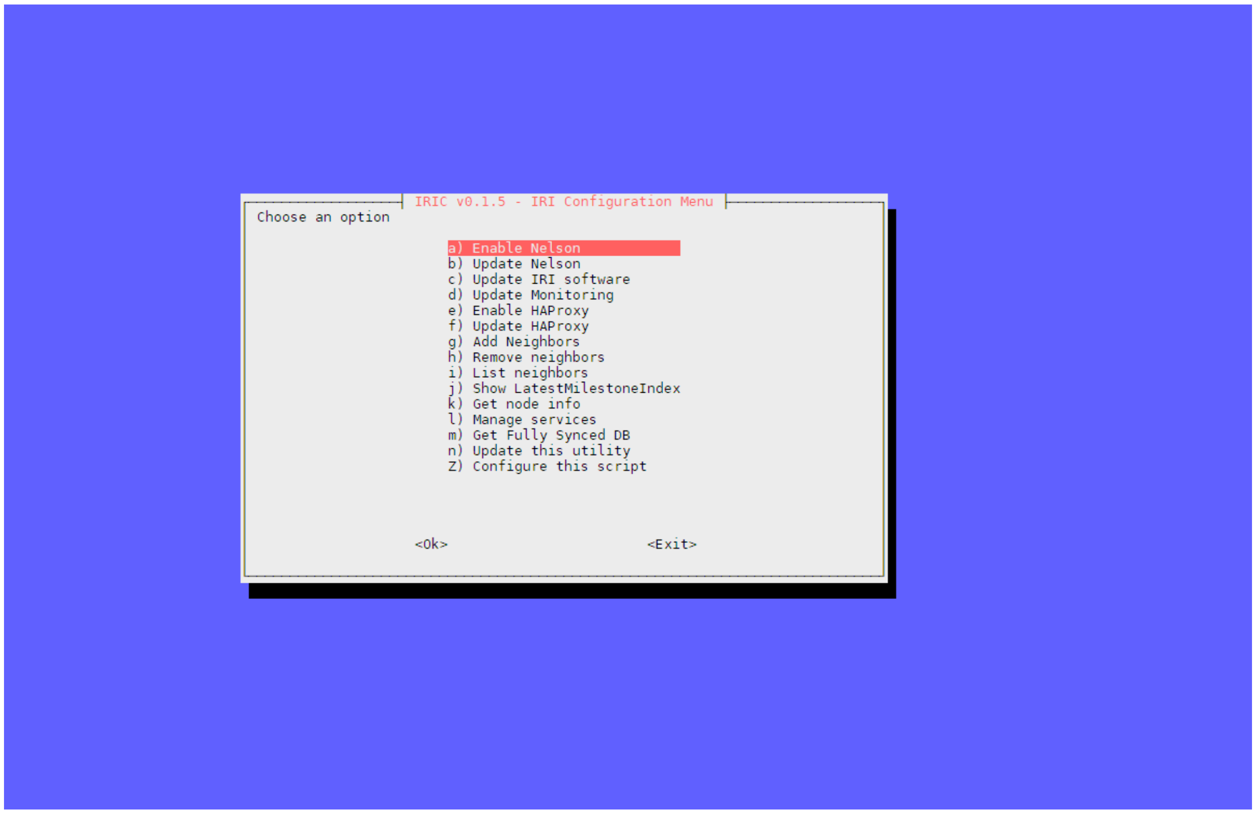
1.1 Introduction

The iri-playbook is a full fledged IOTA node installer including monitoring and operational tools to manage an IRI node.

It was first released in December 2017 and has since helped thousands of node operators get up and running within minutes.

The playbook installer includes a comprehensive set of tools and services to help operate your node on the IOTA network.

Note: The playbook installation provides a handy tool to help manage the full node's services:



```
IRIC v0.1.5 - IRI Configuration Menu
Choose an option

                a) Enable Nelson
                b) Update Nelson
                c) Update IRI software
                d) Update Monitoring
                e) Enable HAProxy
                f) Update HAProxy
                g) Add Neighbors
                h) Remove neighbors
                i) List neighbors
                j) Show LatestMilestoneIndex
                k) Get node info
                l) Manage services
                m) Get Fully Synced DB
                n) Update this utility
                Z) Configure this script


             <Ok>                   <Exit>
```

The installer aims at providing a simple way for anyone to be able to install and operate their IRI node.

The wiki includes some information about operations and configurations that might come in handy.

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

For the fully automated installation visit *Getting Started Quickly*.

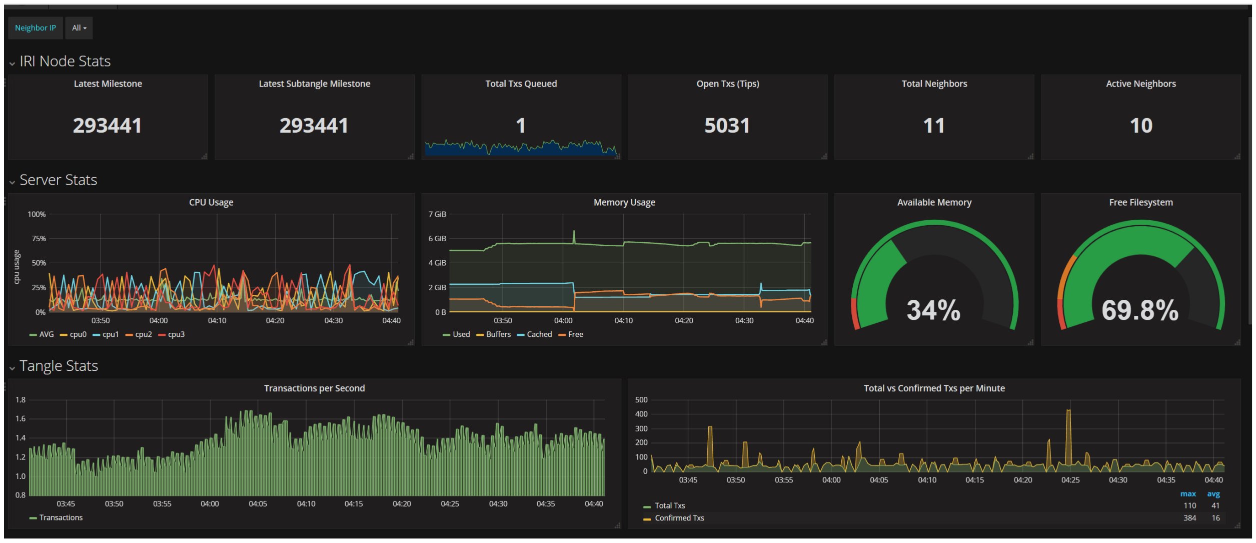
The git repository includes all the code of the automated installation using Ansibe Playbook.

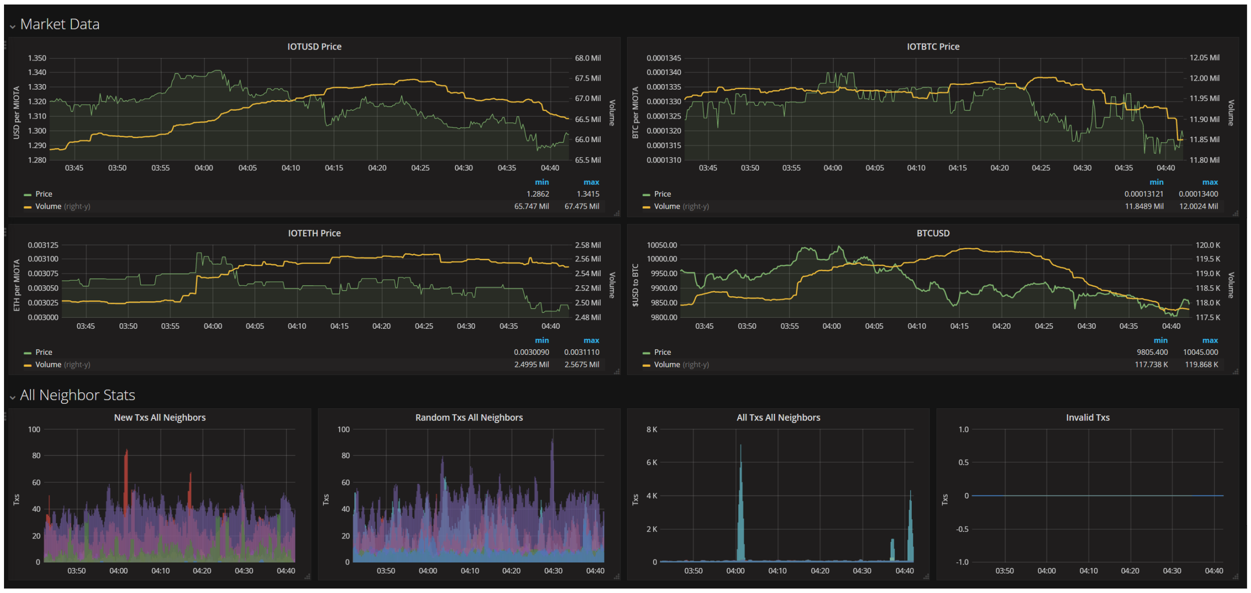
The automated installed installs IRI and IOTA peer manager, a web GUI with which you can view your neighbors, add or remove neighbors, view the sync and much more.

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

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

Warning: Your server's installation of Ubuntu, Debian 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!

Note: To install multiple nodes you will have to follow the manual installation steps. Please refer to the "Installation" chapter.

1.3.1 Run the Installer!

Please consider switching to Hornet-playbook <https://github.com/nuriel77/hornet-playbook>

Step 1

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

For **Debian** or **Ubuntu** users: you may need to install `curl` and `screen`. You can do that by running: `sudo apt-get install screen curl -y`.

Step 2

Enter into a "screen" session by typing: `screen -S iota`. If connection to the server is lost during the installation you can re-attach to the session by running `screen -r iota`.

Step 3

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

```
sudo bash -c "bash <(curl -s https://raw.githubusercontent.com/nuriel77/iri-playbook/
↪feat/docker/fullnode_install.sh)"
```

Note: 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:

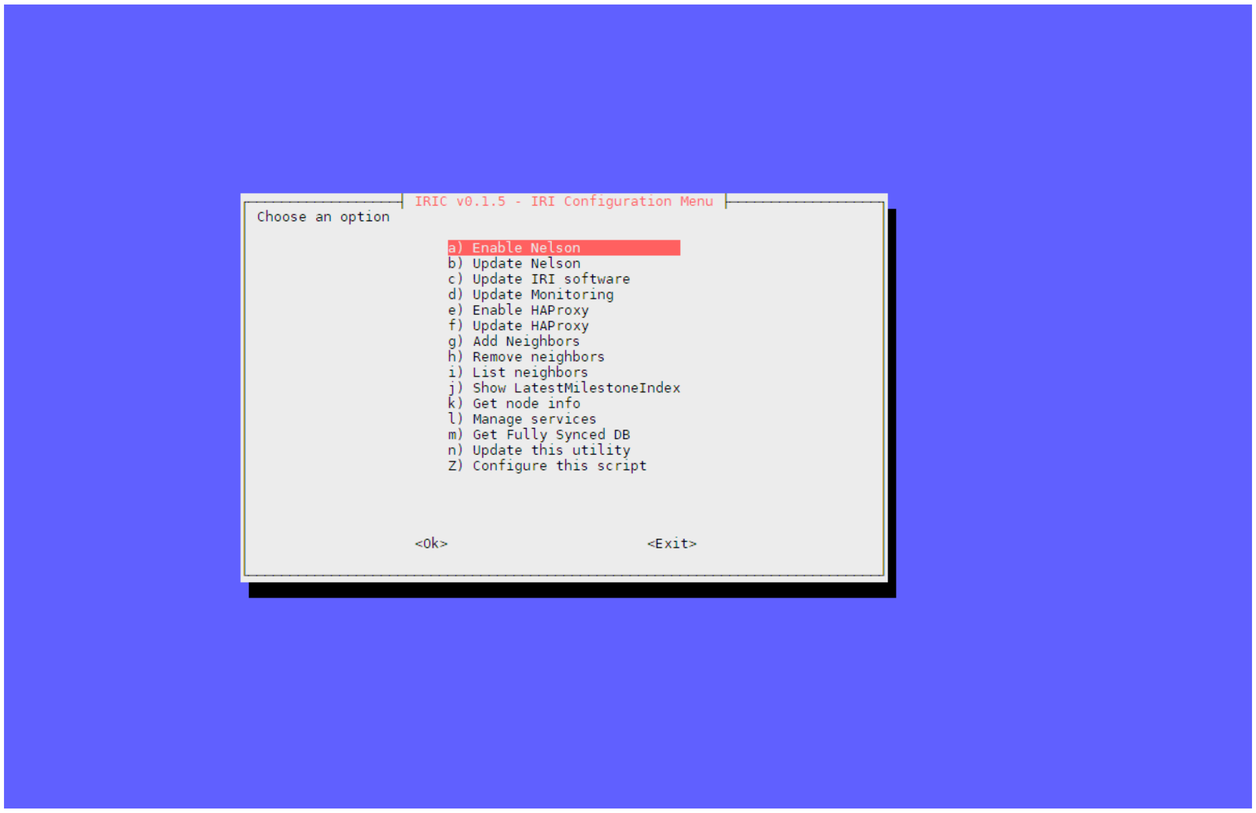
```
https://your-ip:5555/dashboard/db/iota?refresh=30s&orgId=1
```

and Peer Manager via:

```
https://your-ip:8811
```

- You can log in using the username and password you have chosen during the start of the installtion.
- 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 on how to install valid certificates (certbot).
- Please consider hardening the security of your node and periodically upgrading all system packages. 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*.

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 or Debian, 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 username and password you've configured during installation:

```
https://your-ip:8811
```

Accessing Monitoring Graphs

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

```
https://your-ip:5555
```

1.4 The Requirements

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

1.4.1 Virtual Private Server

Having a Virtual Private Server (VPS) at a hosting company 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 service with pay-per-hour or monthly contract. Be aware that some VPS hosting such as SSDNodes had reports by fullnode operators that their contracts have been suspended due to running "crypto" software (IRI).

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

For better performance, at least 6GB RAM and 4 cores are necessary (for example when running consul and consul-haproxy-template for load balancing)

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.

The installer was tested on Ubuntu (LTS only), CentOS and Debian versions:

- 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
- Debian 9.5 x86_64 image for OpenStack
- Debian 10. x86_64 image for OpenStack
- CentOS 7.x x86_64 Generic Cloud Image or CentOS Minimal ISO
- CentOS 8.x x86_64 Minimal ISO

As mentioned above only LTS versions of Ubuntu are supported (e.g. 18.04 and not 18.10)

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, Debian 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 VPS hosting provide a terminal (either GUI application or web-based terminal). Using the terminal you can login to your VPS's command line.

You most 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).

1.5 Installation

Please see *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:

- *Enabling Multiple Loadbalancers*
- *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 *Configuring my server with HTTPS* and look for the Let's encrypt link.

1.6.1 Enabling Multiple Loadbalancers

A new addition to the Dockerized version is the ability to configure Consul (service discovery) to automatically configure HAProxy.

This can work for a single node or multi node cluster (if you have followed the instructions to install multiple nodes at once).

The feature is still under development and considered experimental. You will need some knowledge about HAProxy, Consul and running HTTP/API calls.

For further read please see the documentation here

1.6.2 Controlling IRI

Check status:

```
systemctl status iri
```

Stop:

```
systemctl stop iri
```

Start:

```
systemctl start iri
```

Restart:

```
systemctl restart iri
```

1.6.3 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
```

1.6.4 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, unless using HAProxy in which case the default port is 14267 TCP)

You can check if IRI is "listening" on the ports when you run:

`lsof -Pni|grep "iri"`.

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)
```

(note that when running IRI dockerized you will see docker and not java as the process name)

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 15600
2. IRI is listening for API (or wallet connections) on a local interface (not accessible from "outside") no. 14265

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 have a neighbor connect to my TCP port: `tcp://10.50.0.14:15600`.

If you are behind a home or office router, you will probably have to forward ports from your router to this IP (as this IP will be "internal" and the router have thexternal IP).

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).

1.6.5 Checking IRI Full Node Status

If you prefer usin the command line read further. Otherwise, I recommend using the `iric` tool.

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).

1.6.6 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*.

1.6.7 Adding or Removing Neighbors

I strongly recommend to use `iric` in order to add or remove neighbors. It has the advantage that it makes sure the neighbors you've added or removed persist even after IRI restarts.

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
  --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://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 Neigbors

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://4.3.2.1:4321 -n tcp://[2a01:a0a0:c0c0:1234::1]:15600 -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`!!!

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 options `-f /etc/default/iri` will remove the neighbors from 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`!!!

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]:15600"]}' 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/Debian**:

```
/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.

1.6.8 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/Debian**:

```
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*.

The topics described below are for advanced users or users who want to expose IRI's API directly to the internet (testing API being one example). Only do so if you know what you are doing!

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.

By using the `--remote true` 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 true` option can be specified in the configuration file:

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

Edit the file and find the line:

```
OPTIONS=""
```

and add `--remote` 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/Debian** 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/Debian** 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/Debian**:

```
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)

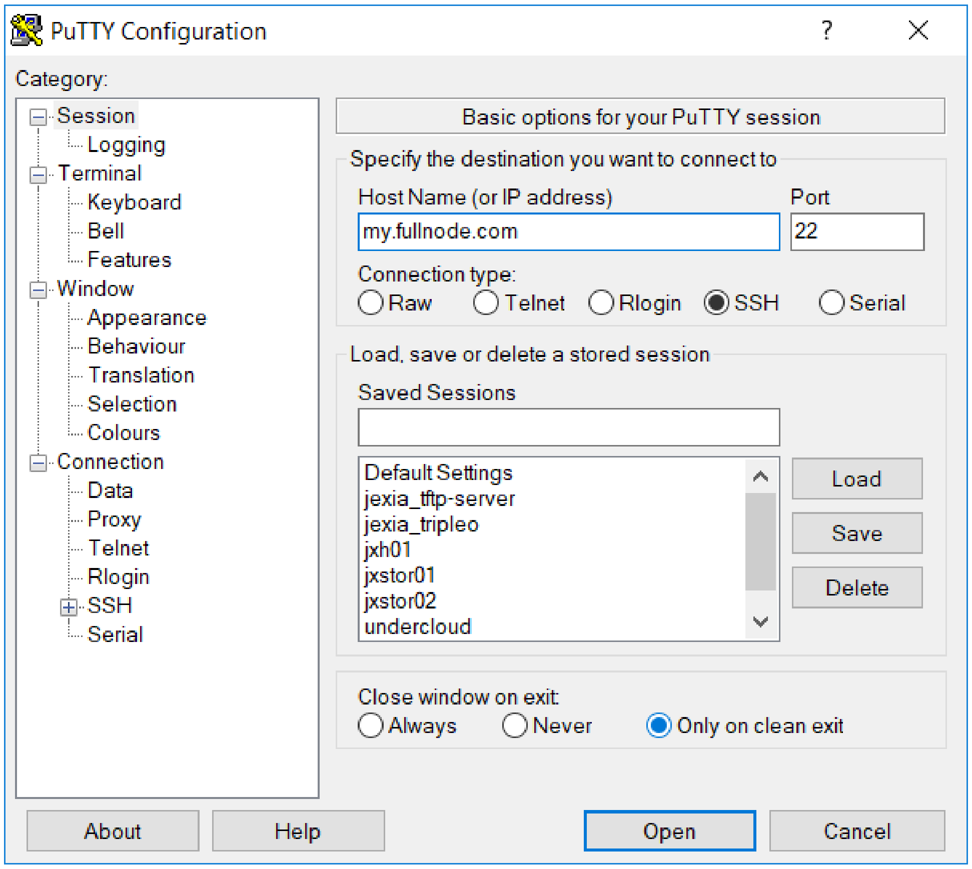
A default installation would have IRI API listening on TCP port 14265.

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.



Specify the destination you want to connect to

Host Name (or IP address): my.fullnode.com

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

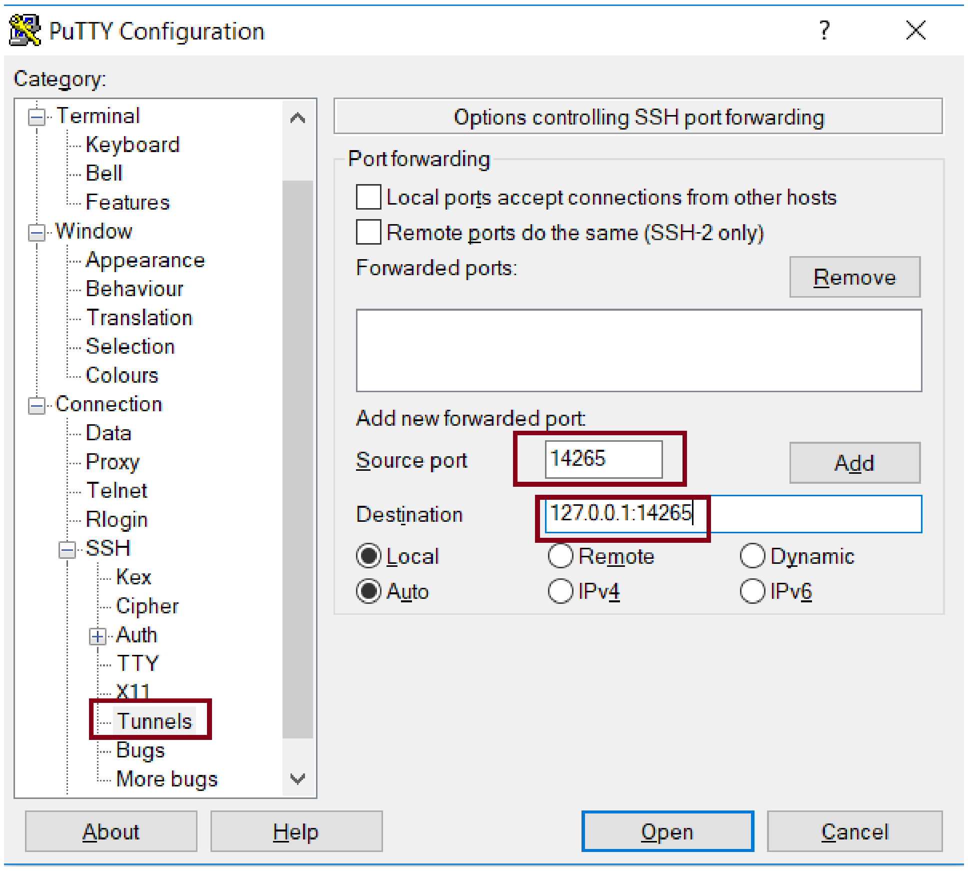
Load, save or delete a stored session

Saved Sessions

Default Settings
jexia_tftp-server
jexia_tripleo
jxh01
jxstor01
jxstor02
undercloud

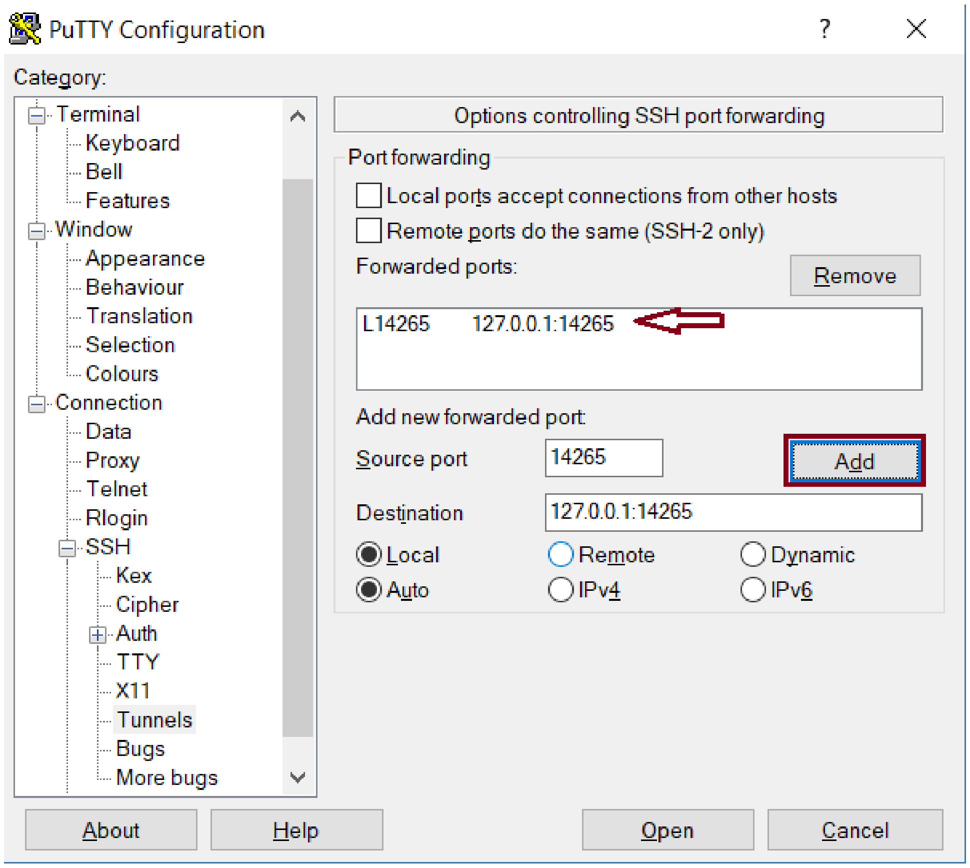
Close window on exit: Always Never [✓ Only on clean exit]

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.



[ ] Local ports accept connections from other hosts
[ ] Remote ports do the same (SSH-2 only)
Source port 14265
Destination 127.0.0.1:14265
[✓ Local] Remote Dynamic
[✓ Auto] IPv4 IPv6

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



☐ Local ports accept connections from other hosts
☐ Remote ports do the same (SSH-2 only)

Forwarded ports:

```
L14265      127.0.0.1:14265
```

Add new forwarded port:

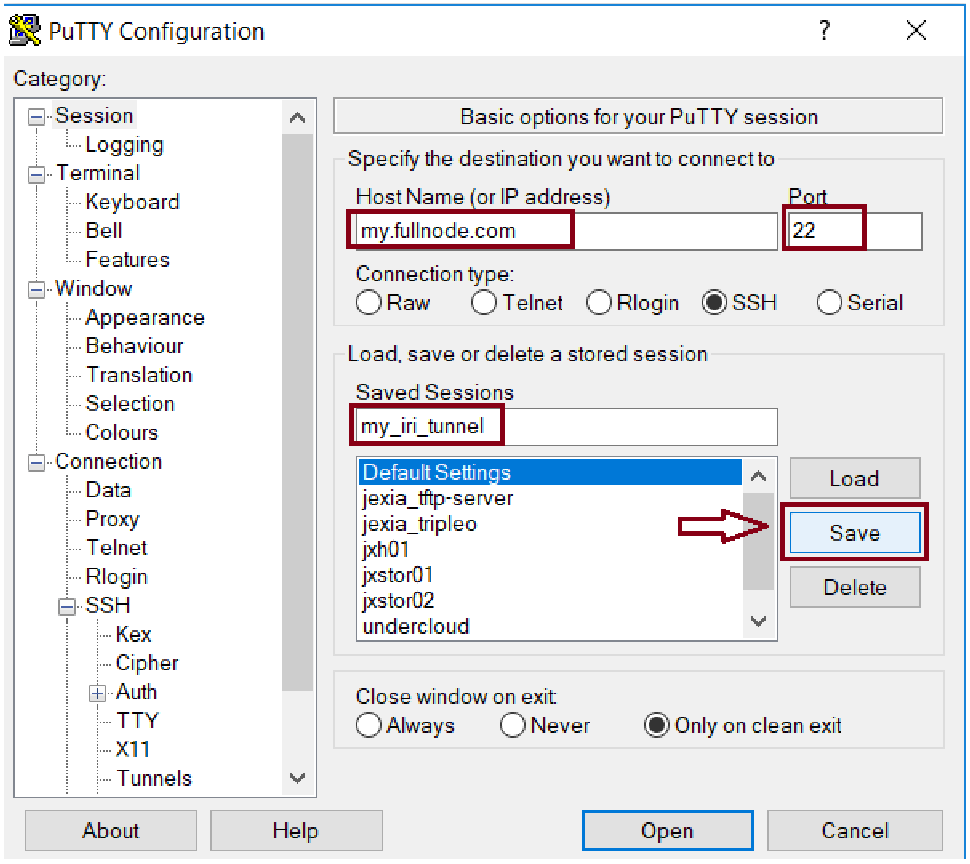
Source port: 14265

Destination: 127.0.0.1:14265

[✓ Local] Remote Dynamic

[✓ Auto] IPv4 IPv6

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.


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): my.fullnode.com

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: my_iri_tunnel

Default Settings
jexia_tftp-server
jexia_tripleo
jxh01
jxstor01
jxstor02
undercloud

Close window on exit: Always Never [✓ Only on clean exit]

5. 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: `http://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

I will take a moment to remind the reader: the preferred method to connect to the API is via using the HAProxy (use the `iric` tool to make sure it is enabled).

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/Debian** 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/Debian: /etc/default/iri
CentOS: /etc/sysconfig/iri
```

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

```
Ubuntu/Debian: /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:

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

Grafana Database file:

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

Prometheus configuration file:

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

IOTA-Prom-Exporter configuration file:

```
/var/lib/iota-prom-exporter/config.js
```

Alert Manager configuration file:

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

HAProxy configuration file:

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

1.9 Maintenance

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

1.9.1 Upgrade IRI

Please use the tool `iric` in order to upgrade IRI's version when a new release is out.

Latest IRI release is available here.

Sometimes it might take a while until the image is pushed to dockerhub, in which case you will see a release is already on Github but not yet available to upgrade your node. It normally takes a few minutes to get built and available on dockerhub.

If a new version has been announced, a new docker image for IRI should be made available and can be pulled via `iric` (upgrade IRI).

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 and Debian 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 partitition 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 Docker

The new release of IRI-playbook introduces a valuable feature where all services are run inside of Docker containers.

Note: Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

Source

1.10.1 Docker Installation

During the installation phase of the playbook, you are provided a selection menu where some options can be selected. One of those options is whether to install Docker on your server (enabled by default). Docker is the service which controls and managed containers on your server.

1.10.2 Services Management

All the services installed by the playbook are "containerized" and controlled by Docker. Systemd (systemctl ...) drop-in files have been configured allowing control of the services as any other service running on the host (e.g. `systemctl status iri` or `journalctl -u iri -e`).

1.10.3 Docker Network

During the installation, the playbook has created a Docker network on the server to be used by IRI and peripheral services. This abstracts the service's networking from the host's network and only required ports are exposed on the host (or publicly where needed).

1.10.4 Service's Configurations and Data Directories

Services' configuration files are normally located on paths on the host itself and mounted "read-only" to the appropriate container.

Data directories such as IRI's database are mounted from the host into the container as "read-write".

Even if the container gets permanently removed, the data directory remains on the host.

1.10.5 Docker Images

In order for docker to run services in containers it requires an image per container. For example, IOTA provides an official IRI Docker image which gets automatically updated when a new version of IRI is released.

When you upgrade IRI (using `iric` for example) the new image/version is downloaded to your server, IRI restarted and that's about it.

You can easily rollback to the older version.

Docker images are shipped in the format: `image-name:tag`, where by default, if no tag is specified, `latest` is the default.

It is important to maintain a tag/version to help identify the version of software of each container's image. Unfortunately, in somecases, that was not possilbe due to upstream image not maintaining versioned tags.

TODO: Provide some commands to help manage docker images and containers, cleanup old images etc.

1.11 Load Balancer

The new release of IRI-playbook Docker version introduces a valuable feature: a highly-available load balancer based on HAProxy and Consul as the registry backend.

`HAProxy` is installed by default on all IRI Playbook installations. When HAProxy is used on a single node – not in a cluster – it doesn't really function as a load balancer, but simply as a reverse proxy for IRI's API port. Nevertheless, it includes rate-limiting and other security policies out-of-the-box.

`Consul` is used in combination with `Consul-template` to provide HAProxy with a dynamic registry of backend nodes. This means that you can register multiple IRI nodes to Consul to benefit from HAProxy's load balancing capability. Consul holds the database containing a registry of all the nodes you want added to the load balancer. In addition, Consul runs periodic health-checks on the nodes to ensure they are performing well.

Please consider the following:

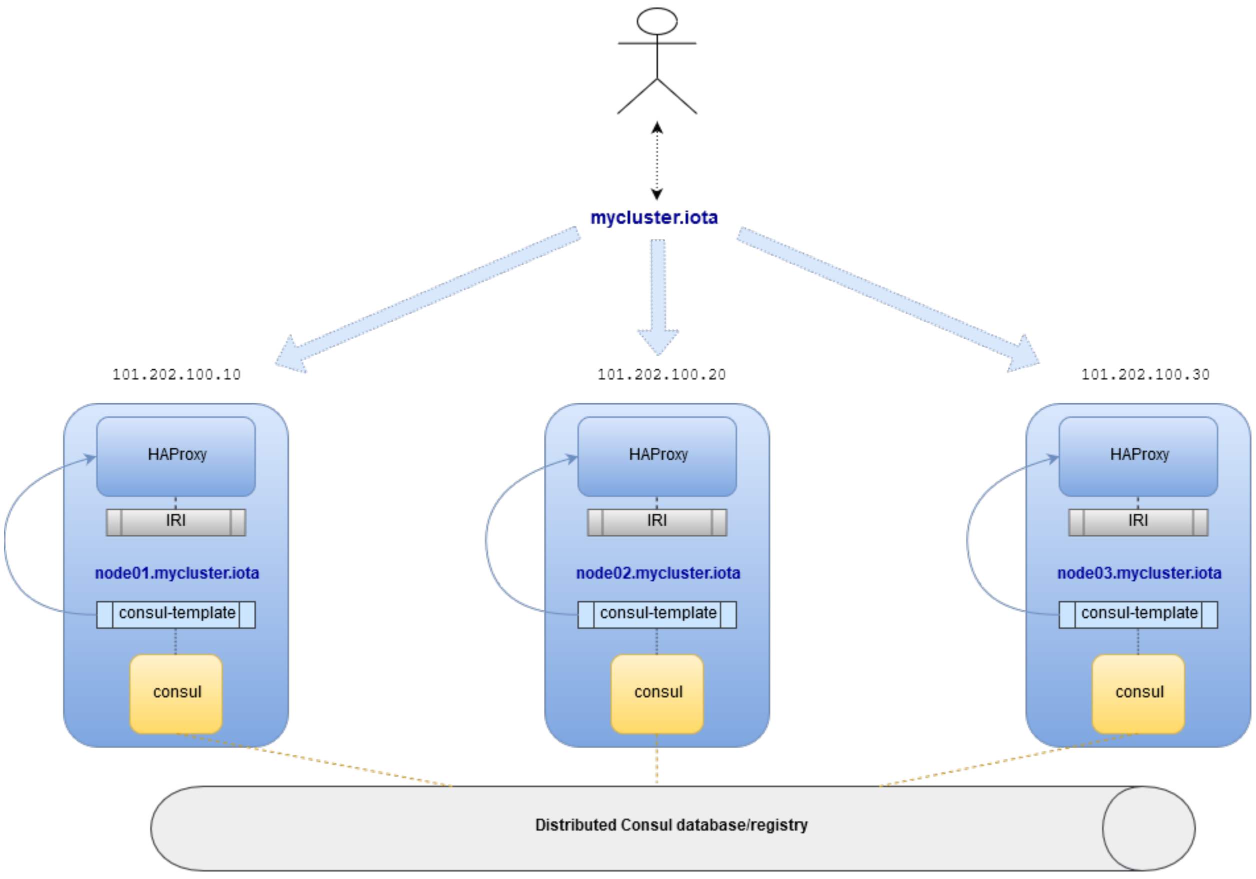
Note: It is strongly recommended **NOT** to use the load balancer feature to register unknown nodes. Please make sure you only use the load balancer feature for your own cluster of nodes. There is no way to check whether unknown nodes are up to no good.

Disclaimer: I take no responsibility for any problems that might arise due to ignoring the recommendation above.

1.11.1 Overview

For command examples for Consul, please refer to this document.

The highly-available setup diagram:



Consul

The IRI docker playbook allows to deploy nodes with both HAProxy and Consul. If your deployment consisted of multiple nodes, Consul nodes will forms a cluster, offering a distributed configuration database for HAProxy.

Consul nodes share a key/value database, distributed services and health-checks registry. A service, in our case, is an IRI node. The health-check associated with a service can be a simple bash script that run some validations on the service.

Each IRI node registers itself with its local Consul daemon. Subsequently, it becomes visible to all other Consul daemons. When a new node is registered, Consul initiates health-checks to that node. If the node is considered healthy, it will become available across the entire cluster on all the instances of HAProxy. Health checks run periodically, and if a failure is detected, that nodes become unavailable on all HAProxy nodes until it recovers.

Let’s recap:

- HAProxy is a load balancer/proxy.
- Consul acts as a distributed key/value database, service registry (discovery) and able to initiate health-checks.
- Consul and HAProxy are installed by default, on a single or multiple node installation.
- Consul forms a cluster when installed on multiple nodes, offering distributed highly available registry.
- HAProxy configures its routes (aka backends) based on Consul’s registry.
- Based on Consul’s health-checks, backends become registered or de-registered from the load balancers (HAProxy).
- Because Consul is distributed, HAProxy on all nodes get configured similarly.

Server and Agent

Consul is able to run in two modes: server and agent. Server mode in itself is also an agent, but is also able to become a cluster member forming a quorum with other server-mode Consul nodes. You can deploy 3 or more consul servers to form a highly available, fault-tolerant cluster (3 or more nodes are required to achieve quorum: always use an odd number equal to or larger than 3).

In the most basic setup, the IRI Docker Playbook will install all IRI nodes with Consul in server mode. This will form a very robust cluster.

An important note about Consul: when you register a service, the node on which you have registered the service becomes its “parent”. If this node becomes unavailable, so does any service that has been registered on it.

What this means is: the **recommended** way to add new nodes to your cluster is by installing them via the playbook: adding them into the `inventory-multi` file. This will configure everything out-of-the-box and is less error prone than any manual method.

If you cannot do that, and simply want to add already existing nodes to your cluster, you have two options:

Option 1

Install Consul agent on such nodes manually, copying the `/etc/consul/conf.d/main.json` and editing it as required (IP addresses etc). Manually open required firewall ports for Consul to communicate with other Consul members. Register the node in its own Consul agent (will become available on the load balancers)

Option 2

Register the new nodes to any of the existing Consul nodes. The downside being that if that Consul node goes down, the node you have registered on it will also be considered unavailable.

HAProxy

With the Consul role enabled during an IRI playbook installation, the HAPRoxy default configuration file is replaced with a template. The template’s values are then populated by the `Consul-template` service. The Consul-template service watches Consul for any changes in service and/or health-check registry. When a new service is registered and health check passes, Consul-template automatically edits HAPRoxy’s configuration template and issues a hot-reload. The same happens if a service is deregistered, updated or health-check fails.

1.11.2 Configuration Options

Options

There are two options:

1. Single IRI node/load balancer
2. Three or more IRI nodes/load balancers

For the first option – a single IRI node is required – installed with the IRI Playbook Docker version. HAProxy, Consul and Consul-template will also be installed on it. This node registers its own IRI to its Consul and is available to receive API calls. It is also possible to manually register new services (IRI nodes) to it.

The disadvantage of a single node is being a single-point-of-failure (SPoF). If it is down, all the instances that are registered on it become unreachable through it.

For the second options, the recommendation is to use at least three nodes in order to form a minimum requirement for creating quorum. In addition, an odd number of nodes should be used. With an odd number of nodes it is possible to reach consensus on who are the active nodes in the cluster.

Deployment of the mutli-node setup requires following the manual installation instructions of the IRI Docker playbook.

Centralized Management for Multi-nodes

At time of writing, there is no centralized management interface for administration of all nodes in the cluster via a centralized panel.

Some functionalities such as enabling HTTPS (+letsencrypt) is centralized via `iric`: it is possible to enable HTTPS for all nodes and request a certificate. Requirements are:

Have a fully qualified domain name for each node Have a single qualified node for the cluster (with multiple A records, each pointing to a node in the cluster).

For example:

```
101.202.100.10 node01.mycluster.iota
101.202.100.20 node02.mycluster.iota
101.202.100.30 node03.mycluster.iota
```

Load balancer's address:

```
101.202.100.10 mycluster.iota
101.202.100.20 mycluster.iota
101.202.100.30 mycluster.iota
```

In DNS settings the TTL for the load balancer's fully-qualified-domain-name record is best set to 1 minute.

High Availability

High availability is possible to achieve when installing a minimum of 3 cluster nodes: it is very unlikely that more than one node at the same time fails. If one node fails, there are still 2 nodes operational.

In this documentation we don't get into configuration of a virtual IP and keepalived, which is one option for keeping a working/accessible IP on one of the nodes (keepalived uses VRRP protocol and makes sure the virtual IP is always configured on at least one active node).

We'll be using simple DNS multiple A records. For example, the hostname `mycluster.iota`. It is possible to configure in any DNS panel multiple A records for this hostname. An A record points a hostname (or hostnames) to an IP address.

As seen in the previous section (Centralized Management for Multi-nodes), we provide an example of these 3 nodes:

```
101.202.100.10 node01.mycluster.iota
101.202.100.20 node02.mycluster.iota
101.202.100.30 node03.mycluster.iota
```

The load balancer's fully-qualified-domain-name is configured with three A records:

```
101.202.100.10 mycluster.iota
101.202.100.20 mycluster.iota
101.202.100.30 mycluster.iota
```

In DNS settings the TTL for the load balancer's fully-qualified-domain-name record is best set to 1 minute.

We configure `mycluster.iota` with 3 A records, each record with each node's IP address pointing to the hostname `mycluster.iota`. A TTL (time-to-live) can usually be configured: it defines the time a DNS server should cache the record. For our scenario it is recommended to set it low (e.g. 60 seconds).

DNS uses round-robin by default (the associated IP address to the hostname are rotated in each reply from the DNS server).

1.11.3 Quick Installation Example for Multiple Nodes

Below are the quick installation command for both Ubuntu and CentOS.

Prerequisites:

- At least 3 strong nodes (2 CPUs, 4GB RAM at least, though doube than that is recommended for smooth operation).
- Configure each node with a unique fully-qualified-domain-name in DNS.
- For sharing an SSL certificate for the load balancer, a single fully-qualified-domain-name in DNS pointing to all the node's IP addresses (multiple A records).
- Centos 7.5, Ubuntu 18.04 and Debian 9.5 have been tested for this setup.

Note: The configuration examples below are to be perform on one of the nodes in the cluster. Pick one of the nodes to be the "installer" node and make sure it is the first one listed in the inventory file (more about the inventory file below).

Ubuntu/Debian

Become root via:

```
sudo su
```

(The recommended way is to prepend `sudo` to each command, but for sake of the tutorial and installation steps you are fine becoming root).

Install updates:

```
apt update -qqy --fix-missing -y && apt-get upgrade -y && apt-get clean && apt-get␣
↪autoremove -y --purge
```

Check if reboot file exists. If it does, issue a reboot:

```
test -f /var/run/reboot-required && reboot
```

Install Ansible on **Ubuntu**:

```
apt-get upgrade -y && apt-get clean && apt-get update -y && apt-get install software-
↪properties-common -y && apt-add-repository ppa:ansible/ansible -y && apt-get update␣
↪-y && apt-get install ansible git screen nano -y
```

Install Ansible on **Debian**:

```
echo 'deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main' >> /etc/apt/
↪sources.list && apt-key adv --keyserver keyserver.ubuntu.com --recv-keys␣
↪93C4A3FD7BB9C367 && apt-get update -y && apt-get install ansible git screen nano -y
```

CentOS

Update packages:

```
yum update -y
```

Ensure Selinux is enabled (this is recommended to run on each node in the cluster before running the playbook):

```
grep '^SELINUX=enforcing' /etc/selinux/config || sed -i 's/SELINUX=.*$/
↪SELINUX=enforcing/' /etc/selinux/config && echo "Selinux enabled, rebooting..." &&␣
↪reboot
```

Install some useful packages:

```
yum install epel-release -y && yum install ansible git nano vim screen curl lsof␣
↪tcpdump yum-utils bind-utils nc -y
```

If hasn't been done already, check if the node needs a reboot due to new kernel packages:

```
needs-restarting  -r
```

This command will let you know if a reboot is required. You can reboot the node by issuing the `reboot` command.

The next steps are the same for both CentOS and Ubuntu

Ubuntu/Debian and CentOS

Clone the repository, specifically the dockerized version:

```
cd /opt && git clone -b "feat/docker" https://github.com/nuriel77/iri-playbook.git &&␣
↪cd iri-playbook
```

Configure a username and password and add some configuration options.

NOTE make sure to configure your own username and password before pasting this command!

```
cat <<EOF >/opt/iri-playbook/group_vars/all/z-iri-override.yml
fullnode_user: iotaadmin
fullnode_user_password: 'AllTangle81'
install_docker: true
install_nginx: true
lb_bind_address: 0.0.0.0
memory_autoset: True
EOF
```

Chmod the file:

```
chmod 600 group_vars/all/z-iri-override.yml
```

To enable Consul for this installation:

```
grep -qir "^consul_enabled: [yes|true]" group_vars/all/z-consul-override.yml >/dev/
↪null 2>&1 || echo "consul_enabled: yes" >> group_vars/all/z-consul-override.yml
```

And configure remote api port:

```
grep -qir "^api_port_remote: [yes|true]" group_vars/all/z-consul-override.yml >/dev/
↪null 2>&1 || echo "api_port_remote: yes" >> group_vars/all/z-consul-override.yml
```

Copy the example `inventory-multi.example` to `inventory-multi`:

```
cp inventory-multi.example inventory-multi
```

At this point you should edit the `inventory-multi` file manually. Here's an example how it might look after you have it configured... Make sure you have a unique fully-qualified-domain-name for each node, as this will allow you to obtain a common SSL certificate (letsencrypt) for all nodes:

NOTE that for Ubuntu and Debian you have to remove the # on the line of the `ansible_python_interpreter` !!!

```
# Example configuration of multiple hosts
[fullnode]
node01.myhost.com ip=110.200.10.20
node02.myhost.com ip=110.200.10.21
node03.myhost.com ip=110.200.10.22

[fullnode:vars]
# Enable this line for Ubuntu and Debian only!
ansible_python_interpreter=/usr/bin/python3

# Only set this line if you didn't ssh to the servers previously
# from the node where you are about to run the playbook from:
ansible_ssh_common_args='-o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no'
```

Run the installation:

```
ansible-playbook -i inventory-multi site.yml -v -e images_from_master=yes
```

If your connection to the server breaks (network timeout), you can return to the server and re-run the command above (make sure to be in the `/opt/iri-playbook` directory and run it as root).

NOTE that the option `images_from_master=yes` has been added. This ensures that all Docker images are initially pulled to the main installation node and then copied to the other nodes. Sometimes having 3 nodes pull the same image from Dockerhub at the same time is rather slow.

SSL Certificate

If you want to install letsencrypt (free SSL certificate) there are the following requirements:

You need to have a single fully-qualified-domain-name pointing to the IP addresses of each node (multiple A records) Each node should have its own unique fully-qualified-domain-name (hopefully you have configured these names in the `inventory-multi` file)

All the nodes must share the same SSL certificate for the given fully-qualified-domain-name. This will be done by `iric`.

You can use `iric` to enable HTTPS for your node. Because you have the `inventory-multi` file configured, `iric` will make sure to enable HTTPS on all the nodes of the cluster. You will also be able to request for a certificate.

It is not necessary to request for a certificate for nginx, as nginx is serving services that are unique for each node (unlike the load-balancer).

Reconfiguring Node's Registry

The playbook has automatically registered each node into Consul. The file which has been used can be found in `/etc/consul/service_this_node.json`. This file can be edited and re-loaded into Consul.

For more information on how to work with Consul on the load balancers, please refer to this document.

1.12 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.

1.12.1 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.

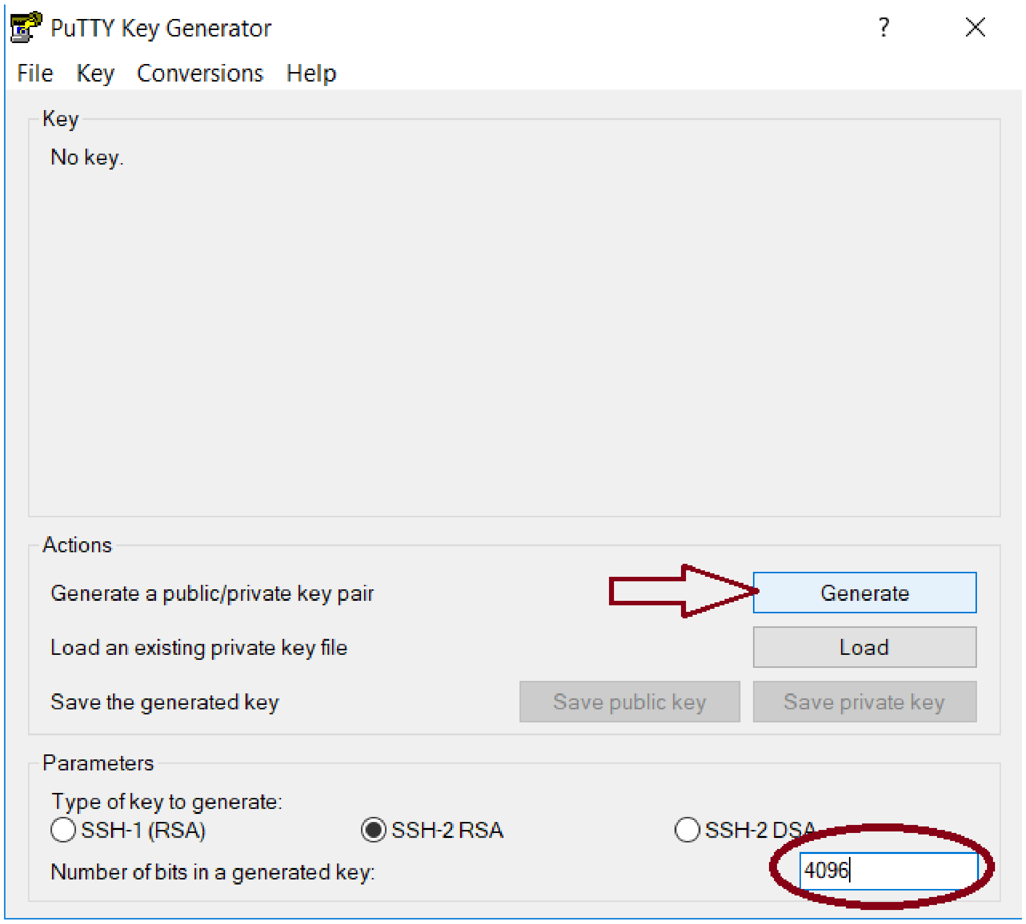
Creating SSH Keys

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`:



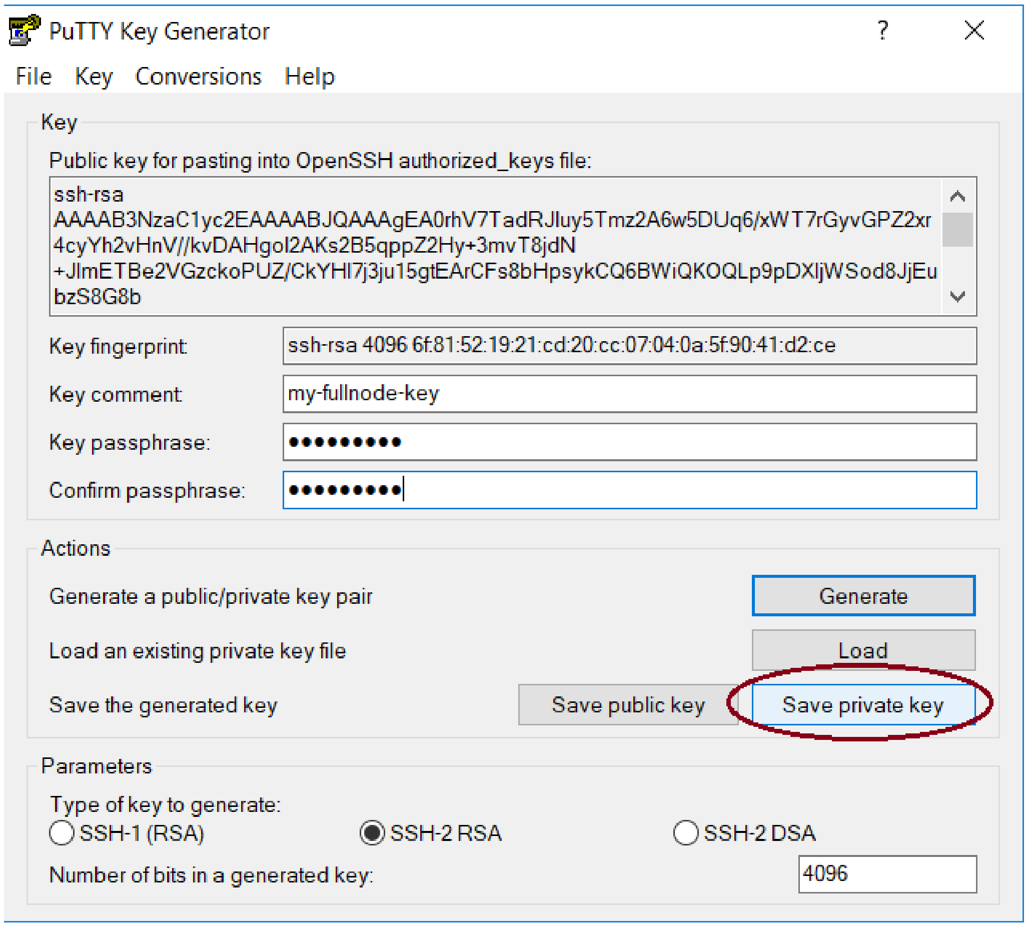
No key.

Type of key to generate: [ ] SSH-1 (RSA) [✓ SSH-2 RSA] [ ] SSH-2 DSA

Number of bits in a generated key: 4096

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.



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAgEA0rhV7TadRJluy5Tmz2A6w5DUq6/xWT7rGyvGPZ2xr
4cyYh2vHnV//kvDAHgol2AKs2B5qppZ2Hy+3mvT8jdN
+JlmETBe2VGzckoPUZ/CkYHl7j3ju15gtEArCFs8bHpsykCQ6BWiQKOQLp9pDXljWSod8JjEu
bzS8G8b
```

Key fingerprint:	ssh-rsa 4096 6f:81:52:19:21:cd:20:cc:07:04:0a:5f:90:41:d2:ce
Key comment:	my-fullnode-key
Key passphrase:	●●●●●●●●●
Confirm passphrase:	●●●●●●●●●

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

Parameters

Type of key to generate:

SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

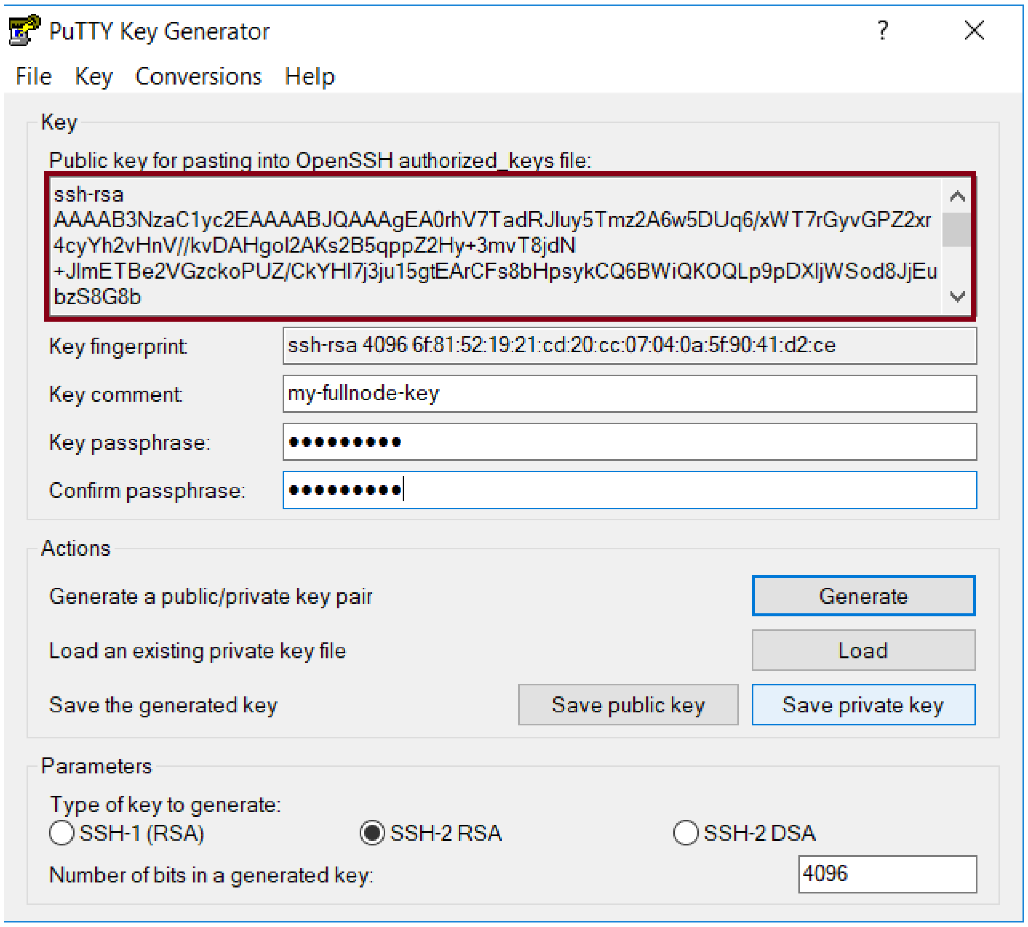
Number of bits in a generated key: 4096

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.



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAgEA0rhV7TadRJluy5Tmz2A6w5DUq6/xWT7rGyvGPZ2xr
4cyYh2vHnV//kvDAHgol2AKs2B5qppZ2Hy+3mvT8jdN
+JlmETBe2VGzckoPUZ/CkYHl7j3ju15gtEArCFs8bHpsykCQ6BWiQKOQLp9pDXljWSod8JjEu
bzS8G8b
```

Key fingerprint: ssh-rsa 4096 6f:81:52:19:21:cd:20:cc:07:04:0a:5f:90:41:d2:ce

Key comment: my-fullnode-key

Key passphrase: ●●●●●●●●●

Confirm passphrase: ●●●●●●●●●

Generate a public/private key pair

Load an existing private key file

Save the generated key

Type of key to generate: SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 4096

See *Using Nano to Edit Files* to learn how to use nano.

6. Set correct permissions:

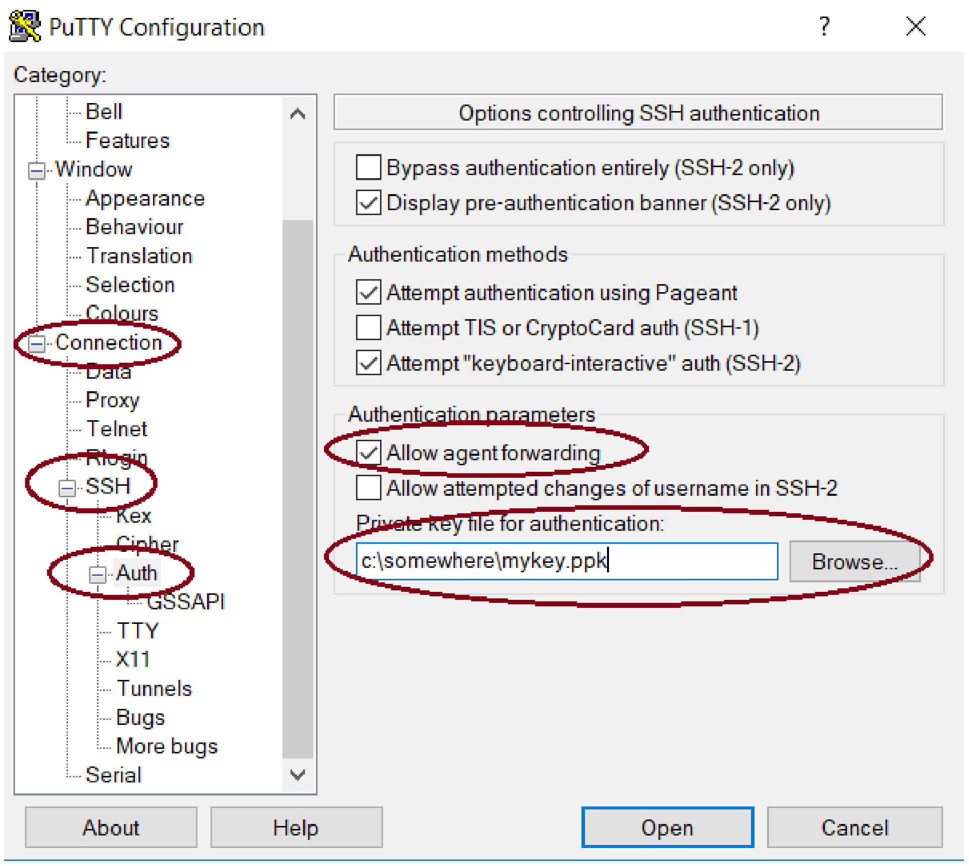
```
chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys
```

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:



Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

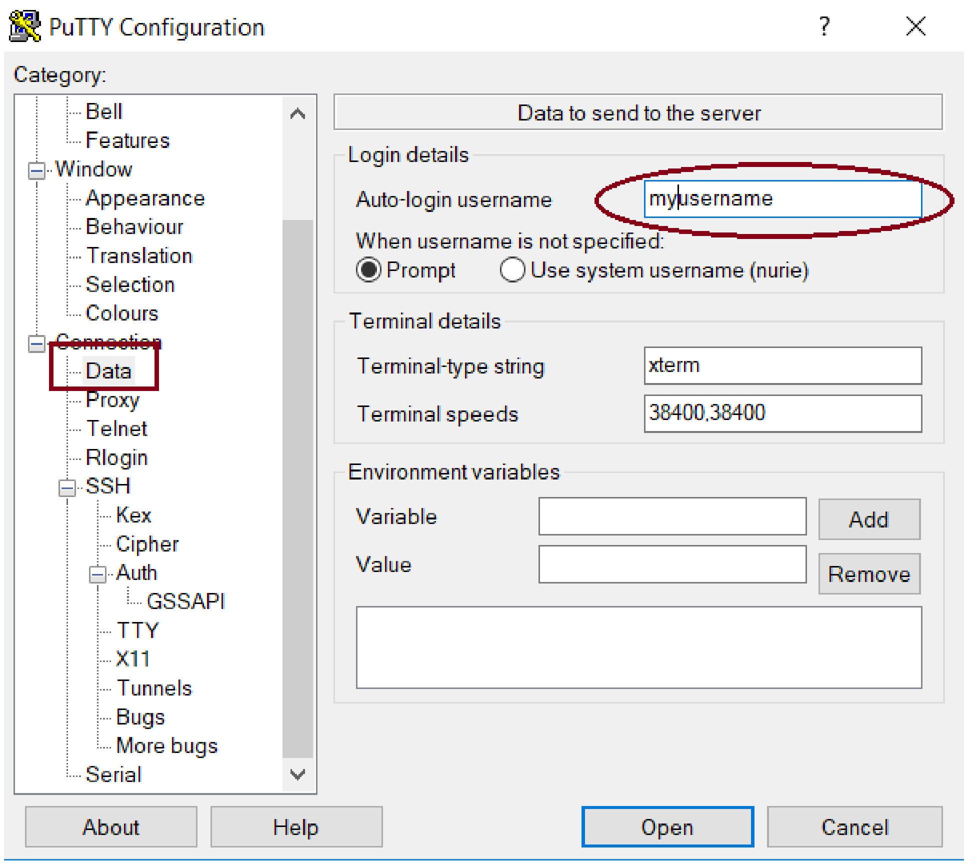
Authentication parameters

- [✓] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

c:\somewhere\mykey.ppk

2. Next open the "Data" option and set the username you've created on the server:



Login details

Auto-login username: myusername

When username is not specified: [✓ Prompt] Use system username (nurie)

Terminal details

Terminal-type string: xterm

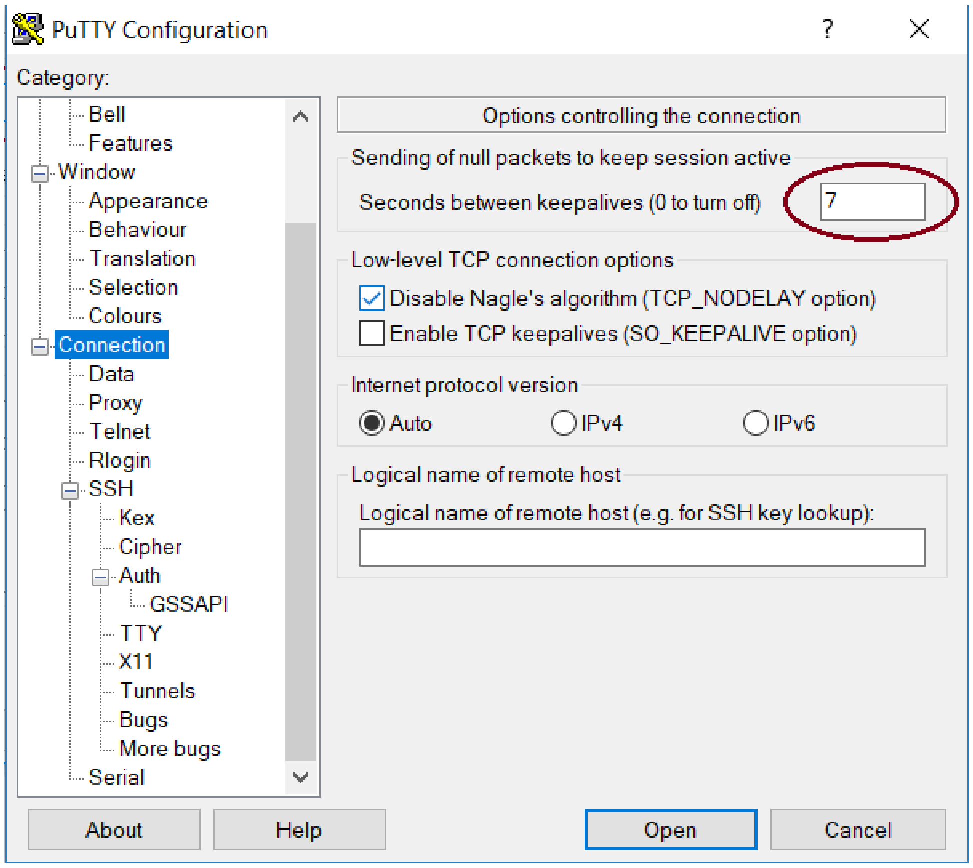
Terminal speeds: 38400,38400

Environment variables

Variable:

Value:

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



Options controlling the connection

Sending of null packets to keep session active

Seconds between keepalives (0 to turn off) 7

Low-level TCP connection options

[✓] Disable Nagle's algorithm (TCP_NODELAY option)

[ ] Enable TCP keepalives (SO_KEEPALIVE option)

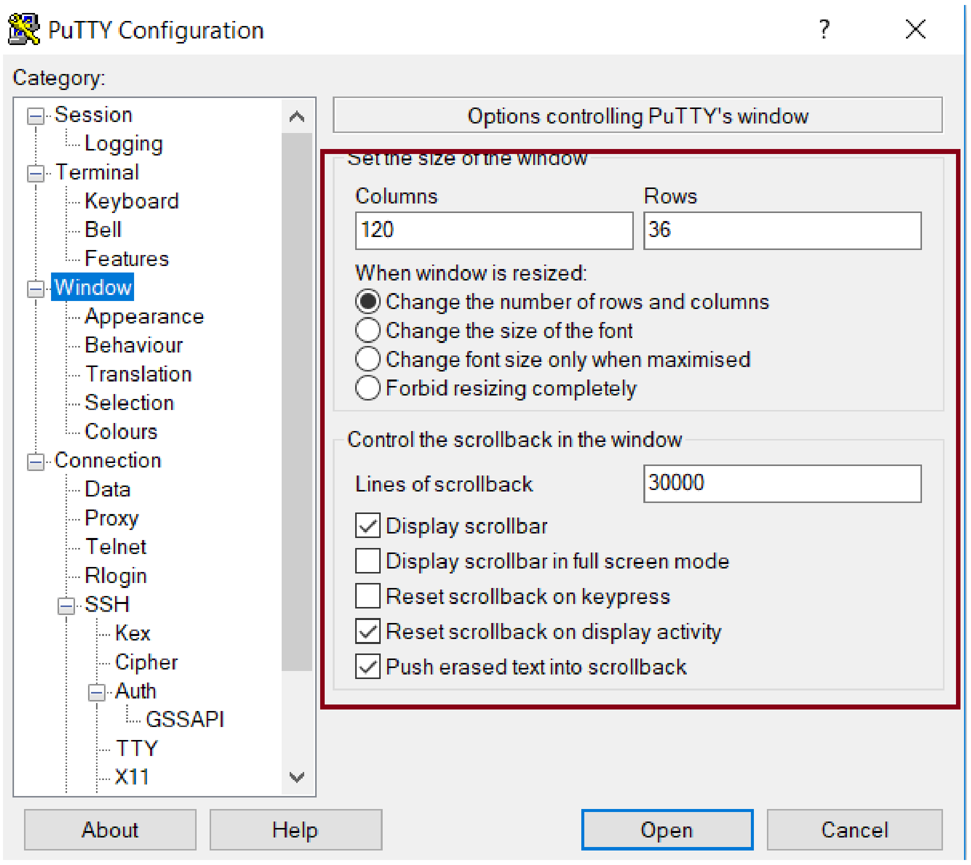
Internet protocol version

[● Auto] ○ IPv4 ○ IPv6

Logical name of remote host

Logical name of remote host (e.g. for SSH key lookup):

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



Options controlling PuTTY's window

Set the size of the window

Columns: 120

Rows: 36

When window is resized:

● Change the number of rows and columns

○ Change the size of the font

○ Change font size only when maximised

○ Forbid resizing completely

Control the scrollback in the window

Lines of scrollback: 30000

[✓] Display scrollbar

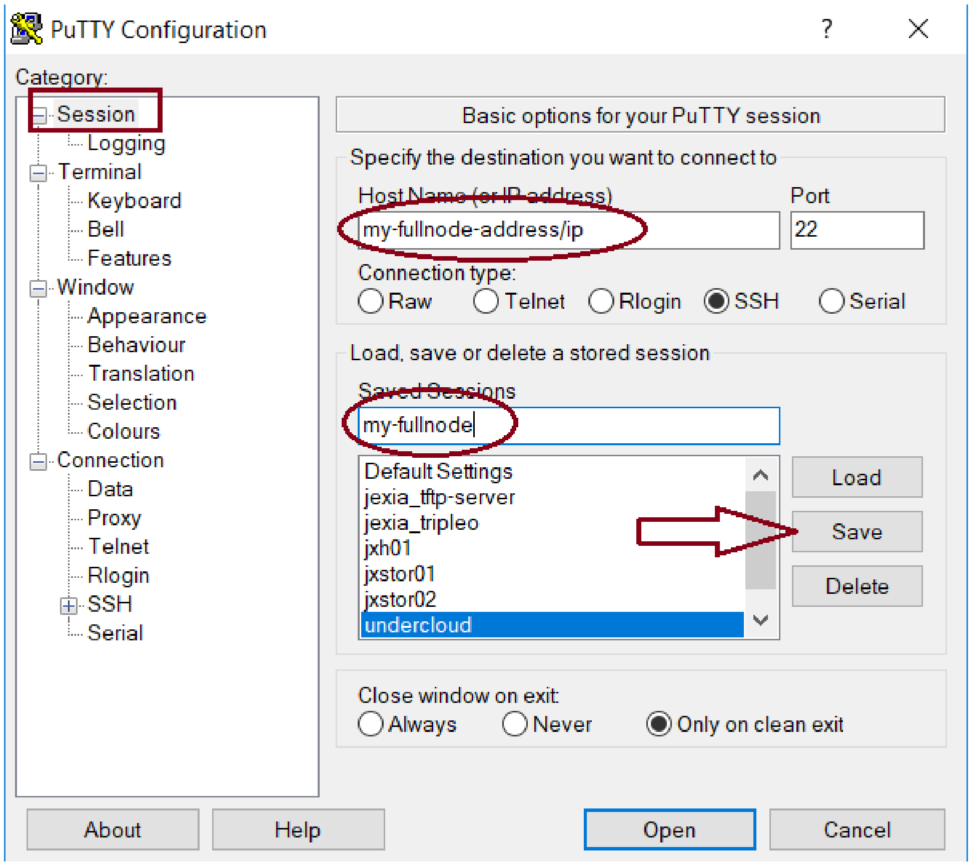
[ ] Display scrollbar in full screen mode

[ ] Reset scrollback on keypress

[✓] Reset scrollback on display activity

[✓] Push erased text into scrollback

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:



Host Name (or IP address): my-fullnode-address/ip
Port: 22
Connection type: Raw [✓ SSH] Telnet Rlogin Serial
Saved Sessions: my-fullnode
Default Settings
jexia_tftp-server
jexia_tripleo
jxh01
jxstor01
jxstor02
undercloud
Close window on exit: Always Never [✓ Only on clean exit]

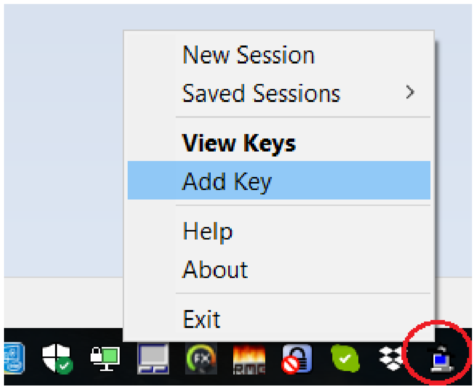
6. Now, or any time you open Putty, you can select this saved session and click "Open". This should connect you to the server. You will be asked to provide the SSH key password (not the user's password from the server!)

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.

From now on, every time you connect to the server using Putty you will not be asked to enter the password.

1.12.2 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 pagent (exit paget), 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
```

1.12.3 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:

```
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 restart sshd daemon as shown above.

1.12.4 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 65000 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/Debian

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
```

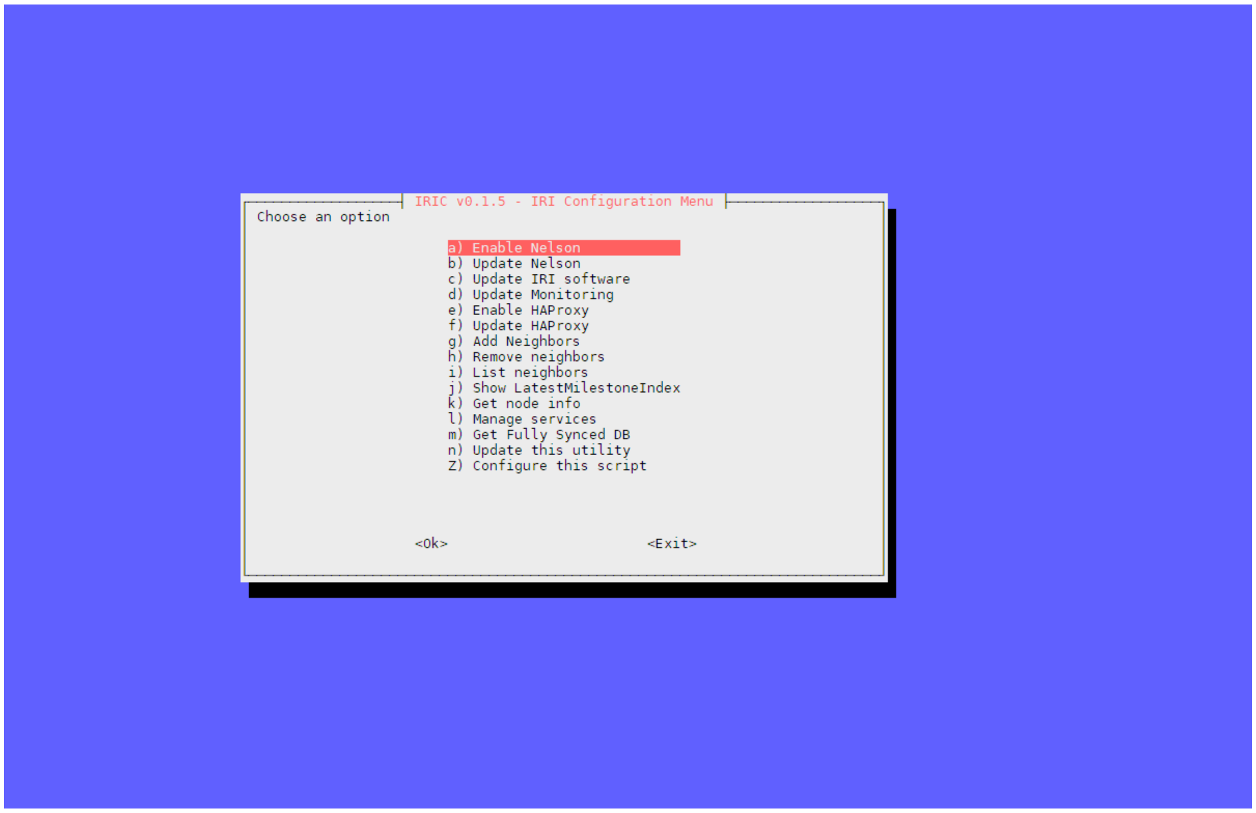
Next, configure your putty session (click “Load” when selecting your saved session, change the port number and click “Save”, then “Open”).

1.13 Fullnode IRI Configuration Utility

`iric` is a tool to help manage services and IRI node related features.

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

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



```
┤ IRIC v0.1.5 - IRI Configuration Menu ├
Choose an option

a) Enable Nelson
b) Update Nelson
c) Update IRI software
d) Update Monitoring
e) Enable HAProxy
f) Update HAProxy
g) Add Neighbors
h) Remove neighbors
i) List neighbors
j) Show LatestMilestoneIndex
k) Get node info
l) Manage services
m) Get Fully Synced DB
n) Update this utility
Z) Configure this script

<Ok>                    <Exit>
```

Tip: Make sure your terminal's window is large enough so you can see the entire menu buttons.

1.14 Troubleshooting

1.14.1 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/Debian**: `apt-get install -y pastebinit`

Below are examples for Ubuntu, Debian 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/Debian 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 /var/log/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 /var/log/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"
```

1.14.2 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.

1.14.3 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/z-installer-override.yml` under the value `fullnode_user_password` and perhaps the `fullnode_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 fullnode_user_
↪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.

1.14.4 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://[2xxx:7xx:aaaf:111:2222:ff:ffff:xxxx]:12345
```

1.14.5 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.

1.15 FAQ

1.15.1 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`:

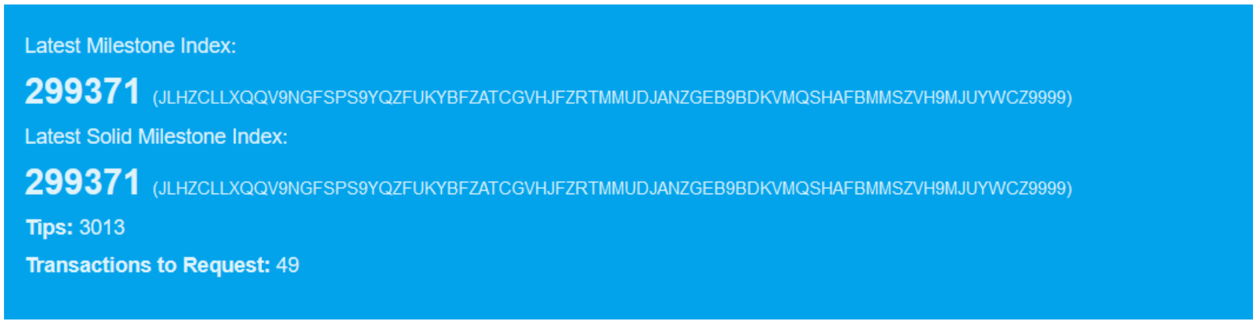
```
test_var: abcd
```

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.

1.15.2 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 (JLHZCLLXQQV9NGFSPS9YQZFUKYBFZATCGVHJFZRTMMUDJANZGEB9BDKVMQSHAFBMMSZVH9MJUYWCZ9999)
Latest Solid Milestone Index:
299371 (JLHZCLLXQQV9NGFSPS9YQZFUKYBFZATCGVHJFZRTMMUDJANZGEB9BDKVMQSHAFBMMSZVH9MJUYWCZ9999)
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/Debian: `apt-get install jq -y`

Centos: `yum install jq -y`

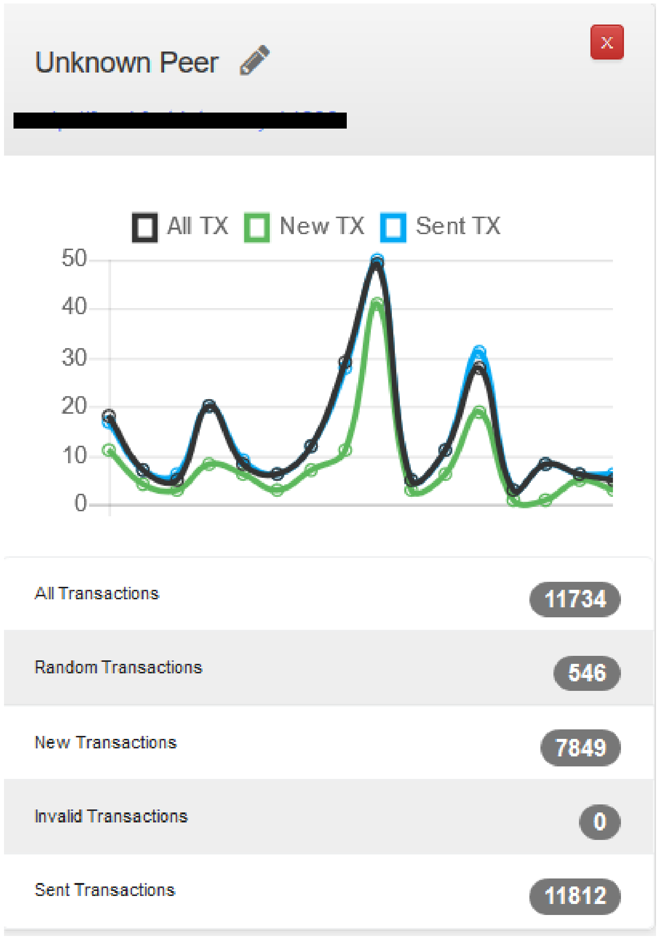
Alternatively, use python:

```
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*

1.15.3 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:



All Transactions 11734

Random Transactions 546

New Transactions 7849

Invalid Transactions 0

Sent Transactions 11812

1.15.4 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.

You can also use `iric` as it has an option in the menu to view the latest milestone index.

1.15.5 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.

1.15.6 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.

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

This service is no longer provided, please consider switching to Hornet

For the sake of the community, I regularly create a copy of a the database snapshot files. I no longer share the fully sycned db 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.

1.16 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
```

1.17 Command Glossary

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

1.17.1 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"
```

1.17.2 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
```

1.17.3 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
```

1.17.4 Check iri and iota-pm ports listening

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

1.17.5 Check all ports on the node

```
lsof -Pni
```

1.17.6 Opening a port in the firewall

In **CentOS**:

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

In **Ubuntu/Debian**:

```
ufw allow 14265/tcp
```

1.17.7 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
```

1.17.8 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/Debian: 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'

1.18 Appendix

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

It is meant for more advanced usage.

1.18.1 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.

In this chapter we are going to configure nginx to serve IOTA Peer Manager and Grafana on port 80, 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 80;
  server_name pm.example.com;
  server_tokens off;

  # Redirect same port from http to https
  # The two lines here under are included in newer
  # versions of the playbook. Omit those if they were
  # not present in your configuration file.
  error_page 497 https://$host:$server_port$request_uri;
  include /etc/nginx/conf.d/ssl.cfg;

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

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

(continues on next page)

(continued from previous page)

```
  }
}
```

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 `http://pm.my-fqdn.com` and see the Peer Manager.

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

ufw allow http

For **Centos**:

firewall-cmd –add-service=http –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 80;
    server_name grafana.example.com;
    server_tokens off;

  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 `http://grafana.my-fqdn.com`.

Note: Using SSL/HTTPS for accessing your panels ensures all traffic and passwords are impossible to “sniff”. The iri-playbook enables HTTPS by default but uses a self-signed certificate.

1.18.2 Configuring my server with HTTPS

Note that you can configure your node with HTTPS via `iric`. This document is kept here for reference and for advanced users.

There are amazing tutorials out there explaining how to achieve this. What is important to realize is that you can either create your own "self-signed" certificates (you become the Certificate Authority which isn't recognized by anyone else), or use valid certificate authorities.

Since a while the IRI Playbook uses own generated self-signed certificate by default. You can replace the certificate and key with your own certificate+key. This can be done here `/etc/nginx/conf.d/ssl.cfg` (this file is included in most configurations).

Let's Encrypt is a free service which allows you to create a certificate per domain name. Other solution would be to purchase a certificates.

By having a "valid" certificate for your server (signed by a trusted authority), you will get the green lock next to the URL in the browser, indicating that your connection is secure.

Your connection will also be encrypted if you opt for a self-signed certificate. However, the browser cannot verify who signed the certificate and will report a certificate error (in most cases you can just accept it as an exception and proceed).

Here is a great tutorial on how to add HTTPS to your **nginx**, choose nginx and the OS version you are using (Ubuntu/Debian/CentOS):

(For iri-playbook installations you can configure the generated certificate and key in /etc/nginx/conf.d/ssl.cfg)

https://certbot.eff.org/

Note: I encourage you to refer to the previous chapter about configuring FQDN for Peer Manager and Grafana. From there you can proceed to adding HTTPS to those configurations.

Note: For **Ubuntu/Debian** you will have to allow https port in ufw firewall:

ufw allow https

For **Centos**:

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

1.18.3 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.

1.18.4 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

1.18.5 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.

1.18.6 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/Debian**: `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 Turorial for more information.

1.18.7 Running IRI API Port Behind HAProxy

On the IRI Dockerized version, IRI is already configured to run behind of HAProxy on port 14267. You do not need to follow the instructions below. They are kept here is reference only.

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 run the following command or 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):

```
cd /opt/iri-playbook && git pull && ansible-playbook -i inventory site.yml -v --
↪tags=iri_ssl,loadbalancer_role -e lb_bind_address=0.0.0.0 -e haproxy_https=yes -e␣
↪overwrite=yes
```

Note that this will apply a default self-signed certificate, but the command is required to enable HTTPS in the first place. If you want to use a valid certificate from a trusted certificate authority you can provide your own certificate + key file manually after running the above command. Alternatively, check the section below for installing a Let's Encrypt certificate which is free:

Let's Encrypt Free Certificate You can install a letsencrypt certificate: one prerequisite is that you have a fully qualified domain name pointing to the IP of your node.

If you already have a domain name, and ran the above command to enable HTTPS, you can run the following script:

```
/usr/local/bin/certbot-haproxy.sh
```

The script will ask you for your email address which is used as an account at Let's Encrypt. It will also ask for the domain name that points to your server's public IP address.

The script will install the required utilities and request the certificate for you. It will proceed to install the certificate with HAProxy and add a cron job to automatically renew the certificate before it expires.

Once the script is finished you can point your browser to https://your-domain-name:14267: you should get a 403 forbidden page. You will be able to see the green lock icon/pad on the left of the URL which means the certificate is valid.

If you need help with this, please find help on Discord #fullnodes channel.

Note: This setup is not fully automated yet via iric. For that reason, please avoid running the HAProxy enable commands as that will overwrite the certificate configuration in haproxy configuration file. If you did that accidentally you can always run the /usr/local/bin/certbot-haproxy.sh once more and it will set the correct configuration file for haproxy.

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*

1.18.8 Installation Options

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

Docker

This installation runs all the services inside Docker containers. If you already have Docker installed on your system you might choose to skip this step.

Nginx

Nginx is a fast and versatile webserver. Its main function in this configuration is to allow access to GUIs in the browser such as IOTA Peer Manager, Prometheus, Grafana and more.

System Dependencies

Although all services are going to run inside of Docker, some additional packages installed on the system are required. If you choose not to install any dependencies, some things might not function as expected and you will have to resolved the dependencies manually.

Firewall

The installation takes care of the firewalls: it ensures the firewall is running and configures the required ports. You can choose not to let the installer configure the firewall should you wish to do this manually.

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*.

IOTA Caddy

The IOTA Caddy is a feature to perform more efficient PoW. All `attachToTangle` requests are proxied from HAProxy to the IOTA Caddy middleware.

Thanks to Luca Moser

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).

1.18.9 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*.

1.18.10 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/Debian (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).

1.19 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 *Security Hardening* to improve your full node's security.
- Refer to *Appendix* for extra configuration options for your full node.
- You take full responsibilty for running add-on software along side this installation/services.
- I am not associated with the IOTA foundation.

1.20 Donations

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

```
JFYIHZQOPCRSLKIYHTWRSIR9RZELTZKHNZFHGWXAPCQIEBNJSZFIWMSBGAPDKZZGFNTAHBLGNPRRQIZHDFNPQPPWGC
```

Thanks!