Python

Jun 04, 2018

Contents

For a “click-‘n-go” installation (recommended) see *Getting Started Quickly*.

CHAPTER 1

In this installation

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

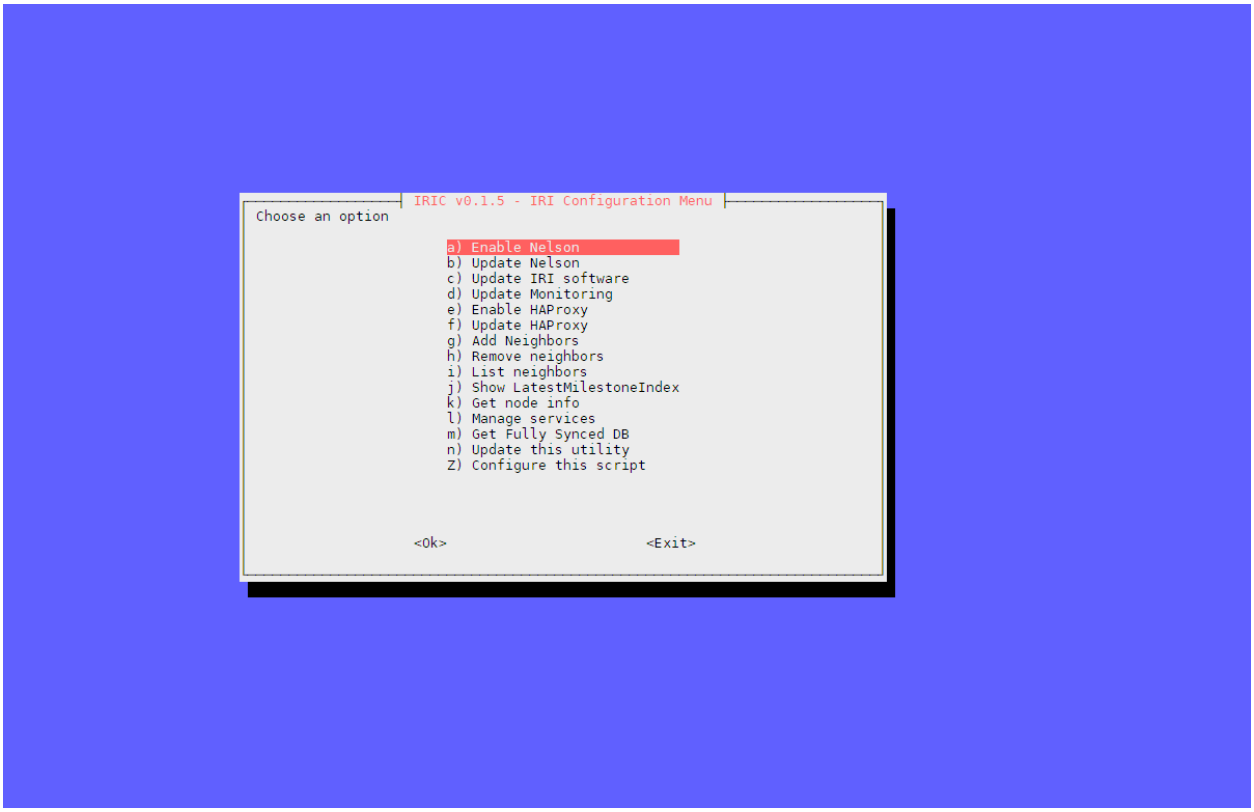
Feel free to star this repository: iri-playbook

1.1 Introduction

My first tutorial I wrote around August 2017. Due to the exponential growth of the community and users who want to run their own full node, I thought it is a good time to write a new, more comprehensive tutorial.

Not only a tutorial: the iri-playbook is a full fledged IOTA node installer including a comprehensive installer.

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



1.1.1 Why Another Tutorial?

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

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

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

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

Good luck!

1.2 Overview

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

The recommended way to install a full node is by using the fully automated installation *Getting Started Quickly*. Please consider using this method rather than the manual method, as it is less error-prone.

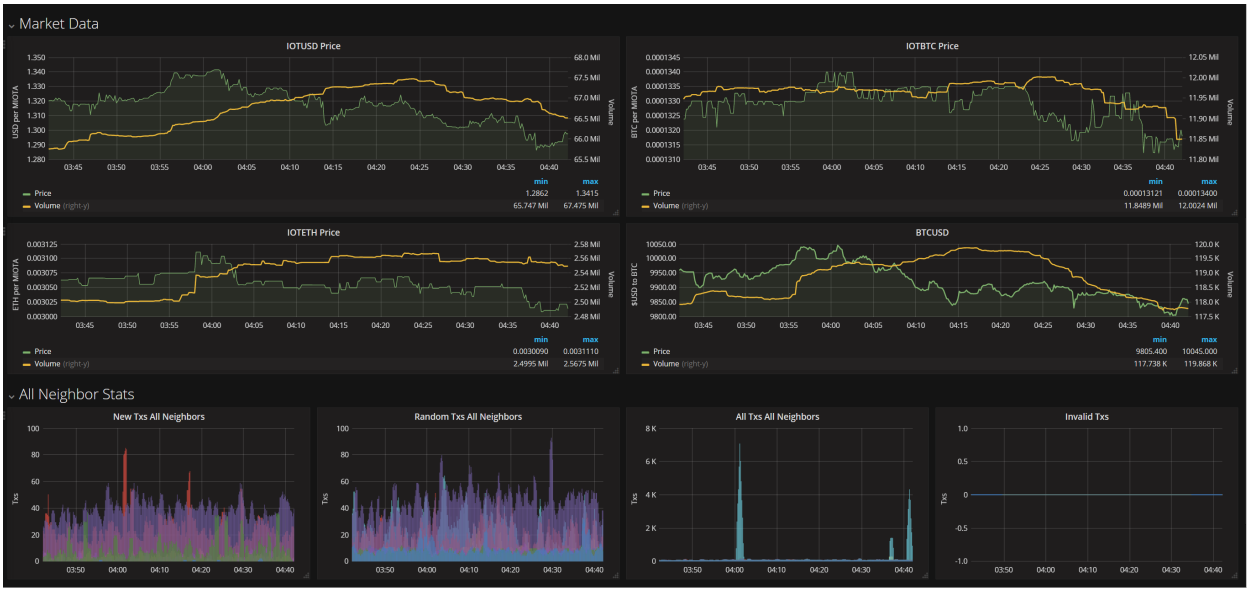
The git repository includes all the code of the automated installation using Ansibe Playbook.

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

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

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

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

1.3.1 Run the Installer!

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

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

```
bash <(curl -s https://raw.githubusercontent.com/nuriel77/iri-playbook/master/
↪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:

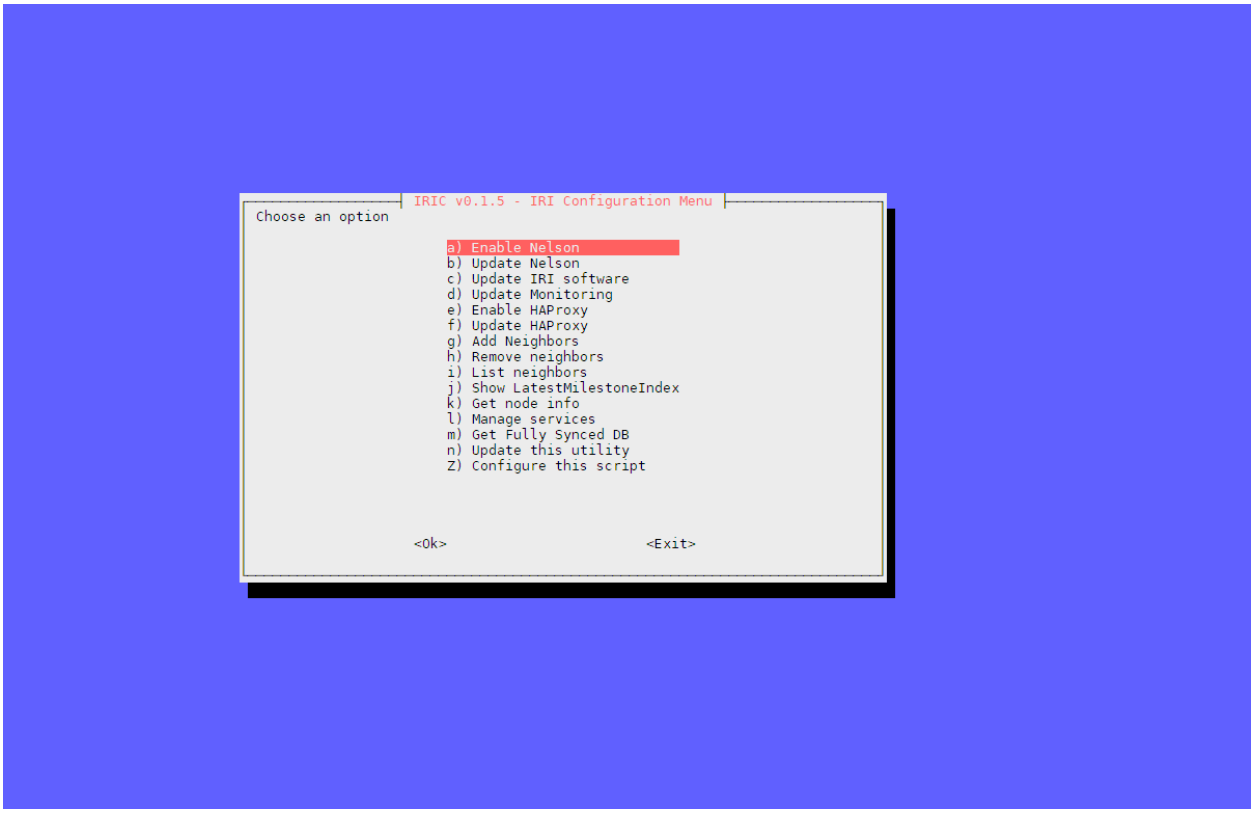
```
http://your-ip:5555/dashboard/db/iota?refresh=30s&orgId=1
```

and Peer Manager via:

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

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

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



DONATIONS

Making this installer happen, supporting and maintaing it takes much effort and time. Nevertheless, it is done happily in order to contribute and help the community.

If you want to leave a donation you can use this address:

```
CSSFHHDBUQDGAUGYUHTENLBJ9JMTUFFLYLJZKTLRZVLLDCZZOQHOUXJOVDKXOLXGCJEMXJOULDIKADBHWMGVALMAUW
```

And star the repository: iri-playbook

Thanks!

Connection Lost

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

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

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

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

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

Accessing Peer Manager

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

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

Accessing Monitoring Graphs

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

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

Big thanks to Chris Holliday's amazing tool for node monitoring

1.4 The Requirements

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

1.4.1 Virtual Private Server

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

I will not get into where or how to purchase a VPS (virtual private server). There are many companies offering a VPS for good prices.

The basic recommendation is to have one with at least 4GB RAM, 2 cores and minimum 30GB harddrive (SSD preferably).

Note: At time of writing (December 2017) many users are experiencing out-of-memory errors with 4GB RAM. This should be remedied by next snapshot.

1.4.2 Operating System

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

This tutorial/installer was tested on:

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

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

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

1.4.3 Accessing the VPS

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

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

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

1.4.4 System User

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

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

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

1.5 Installation

If you have little to no experience with Linux, I recommend you use the *Getting Started Quickly*.

To prepare for running the automated "playbook" from this repository you require some basic packages. First, it is always a good practice to check for updates on the server.

> **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 add the sites certificate as an exception. If you want valid certificates you can refer to *Configuring my server with HTTPS* and search for the "Let's Encrypt" link.

1.5.1 Update System Packages

For **Ubuntu** we type:

```
apt-get update
```

and for **CentOS**:

```
yum update
```

This will search for any packages to update on the system and require you to confirm the update.

Reboot Required?

Sometimes it is required to reboot the system after these updates (e.g. kernel updated).

For **Ubuntu** we can check if a reboot is required. Issue the command `ls -l /var/run/reboot-required`:

```
# ls -l /var/run/reboot-required
-rw-r--r-- 1 root root 32 Dec  8 10:09 /var/run/reboot-required
```

If the file is found as seen here, you can issue a reboot (`shutdown -r now` or simply `reboot`).

For **Centos** we have a few options how to check if a reboot is required.

One of these options requires to install `yum-utils`:

```
yum install yum-utils -y
```

Once installed, we can run `needs-restarting -r`:

```
# needs-restarting  -r
Core libraries or services have been updated:
  systemd -> 219-42.el7_4.4
  glibc -> 2.17-196.el7_4.2
  linux-firmware -> 20170606-56.gitc990aae.el7
  gnutls -> 3.3.26-9.el7
  glibc -> 2.17-196.el7_4.2
  kernel -> 3.10.0-693.11.1.el7

Reboot is required to ensure that your system benefits from these updates.

More information:
https://access.redhat.com/solutions/27943
```

As you can see, a reboot is required (do so by issuing a `reboot` or `shutdown -r now`)

1.5.2 Installing Ansible

Ansible is an awesome software used to automate configuration and/or deployment of services. This repository contains what Ansible refers to as a "Playbook" which is a set of instructions on how to configure the system.

This playbook installs required dependencies, the IOTA IRI package and IOTA Peer Manager. In addition, it configures firewalls and places some handy files for us to control these services.

To install Ansible on **Ubuntu** I refer to the official documentation:

```
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 nano -y
```

For **CentOS**, simply run:

```
yum install ansible git nano -y
```

You will notice I've added 'git' which is required (at least on CentOS it doesn't have it pre-installed as in Ubuntu). In addition, I've added 'nano' which is helpful for beginners to edit files with (use vi or vim if you are adventurous).

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

1.5.3 Cloning the Repository

To clone, run:

```
cd /opt && git clone https://github.com/nuriel77/iri-playbook.git && cd iri-playbook
```

This will pull the repository to the directory in which you are and move you into the repository's directory.

1.5.4 Configuring Values

In these two variable files you will find some configuration parameters for the installation. You can edit those using "nano" (see Note below).

```
group_vars/all/iri.yml
```

and

```
group_vars/all/iotapm.yml
```

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

Configure Memory Limits

In **group_vars/all/iri.yml**:

The options `iri_java_mem` and `iri_init_java_mem` in the configuration files can determine what are the memory usage limits for IRI.

Depending on how much RAM your server has, you should set these accordingly.

For example, if your server has 4096MB (4GB memory), a good setting would be:

```
iri_java_mem: 3072
iri_init_java_mem: 256
```

Just leave some room for the operating system and other processes. You will also be able to tweak this after the installation, so don't worry about it too much.

Note: For the click-'n-go installation, these values are automatically configured. You can choose to auto-configure those values: When running the playbook (later in this guide) you can add `-e "memory_autoset=true"` to the ansible-playbook command.

Set Access Password

This user name and password are used for all web-based authentications (e.g. Peer Manager, Monitoring Graphs).

Create a new variable file called **group_vars/all/z-override.yml** and set a user and a (strong!) password of your choice:

```
iotapm_nginx_user: someuser
iotapm_nginx_password: 'put-a-strong-password-here'
```

You can always add new users after the installation has finished:

```
htpasswd /etc/nginx/.htpasswd newuser
```

Replace 'newuser' with the user name of your choice. You will be prompted for a password.

To remove a user from authenticating:

```
htpasswd -D /etc/nginx/.htpasswd username
```

Note: This username and password will also be used for Grafana (monitoring graphs)

Configure Multiple Fullnodes

You can skip this section and proceed to "Running the Playbook" below if you are only installing on a single server.

The nice thing about Ansible's playbooks is the ability to configure multiple nodes at once.

You can have hundreds of fullnodes installed simultaneously!

To configure multiple hosts you need to use their IP addresses or hostnames (hostnames must resolve to their respective IP).

Edit the file `inventory`. Here's an example of how we would list four hosts, using hostname and/or IP:

```
[fullnode]
localhost        ansible_connection=local
iota01.tangle.io ansible_user=john
iota02.tangle.io ansible_user=root
10.20.30.40      ansible_ssh_port=9922
```

A requirement is that you can SSH access these servers from the server you are working on. Please check *Configuring Multiple Nodes for Ansible* for more information.

1.5.5 Running the Playbook

Two prerequisites here: you have already installed Ansible and cloned the playbook's repository.

By default, the playbook will run locally on the server where you've cloned it to. You can run it:

```
ansible-playbook -i inventory site.yml
```

Or, for more verbose output add the *-v* flag:

```
ansible-playbook -i inventory -v site.yml
```

This can take a while as it has to install packages, download IRI and compile it. Hopefully this succeeds without any errors (create a git Issue if it does, I will try to help).

Final Steps

Please go over the *Post Installation* chapters to verify everything is working properly and start adding your first neighbors!

Also note that after having added neighbors, it might take some time to fully sync the node, or read below the "Fully Synchronized Database Download" section.

If you installed *monitoring* and *IOTA Peer Manager* you should be able to access those:

```
Peer Manager: http://your-external-ip:8811
Grafana: http://your-external-ip:5555
```

Use the username and password from `group_vars/all/z-override.yml` if you set it there previously.

If you followed the Getting Started Quickly guide, you configured a password during the installation, and you can use user `iotapm`.

To configure an email for alerts see *Sending Alert Notifications*.

Fully Synchronized Database Download

In order to get up to speed quickly you can download a fully sycned database. Please check *Where can I get a fully synced database to help kick start my node*

1.5.6 Installing Only IOTA Peer Manager or Monitoring

It is possible to install individual components from the playbook. For example, if you already have installed IRI following a different guide/method, you can use this playbook to install the full node monitoring graphs or IOTA Peer Manager.

Overview

- IOTA Peer Manager is a GUI to help monitor, add and remove neighbors: IOTA Peer Manager.
- The full node monitoring includes monitoring and graphs for IRI and your node: IOTA Exporter.

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

- IOTA Peer Manager doesn't require to be served via a webserver. It is however the recommeneded method, unless you want to use SSH tunnel.
- At this stage, the full node monitoring graphs require to be served via a webserver (nginx), which will be installed via this playbook.

Warning: By installing either Peer Manager and/or the full node monitorting, the firewall will be configured and enabled. It is strongly discouraged to run a server without the firewall enabled. Therefore, this playbook does not support running without a firewall.

Updates

In order to install IOTA Peer Manager or fullnode monitoring, some packages and updates are required.

For **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 -y
```

For **CentOS**:

```
yum install git ansible curl -y
```

Installation

Clone this playbook to `/opt`:

```
cd /opt && git clone https://github.com/nuriel77/iri-playbook.git && cd iri-playbook
```

This assumes that you haven't already cloned the repository to this location. If you have, you should enter the `/opt/iri-playbook` directory and run a `git pull`.

Some parameters require configuration before the installation. Both IOTA Peer Manager and the fullnode monitoring need to know on which port to access IRI API.

This is usually port 14265.

Note that in those two steps we are configurinig the variables files directly. Please consider using an override-file to only edit those parameters you need. This will avoid conflicts when updating new versions of the playbook. See *How to override playbook variables*.

1. Edit `edit group_vars/all/iri.yml` and make sure the `iri_api_port:` option points to the correct IRI API port. In addition, ensure that `iri_udp_port` and `iri_tcp_port` match the ports your IRI is using for neighbor peering.
2. Edit `group_vars/all/iotapm.yml`. Find `install_nginx: true` and set it to `false` if you don't want to install nginx to serve these services via webserver. If you choose to install nginx, leave it as `true` (if you already have nginx installed, just leave it as `true`).

As mentioned earlier: currently, the fullnode monitoring depends on nginx being installed.

3. In the same file `group_vars/all/iotapm.yml`, if using nginx, edit `iotapm_nginx_user` and `iotapm_nginx_password`. These will set the user and password with which you will be able to access Peer Manager and/or the fullnode monitoring graphs.

- To install **IOTA Peer Manager only**, run:

```
ansible-playbook -i inventory -v site.yml --tags=iri_firewalld,iri_ufw,iri_ssl,iotapm_
↪role
```

- To install **full node monitoring only**, run:

```
ansible-playbook -i inventory -v site.yml --skip-tags=iotapm_npm --tags=deps,iri_
↪firewalld,iri_ufw,iri_ssl,iotapm_deps,monitoring_role
```

- To install **both Peer Manager and fullnode monitoring**, run:

```
ansible-playbook -i inventory -v site.yml --tags=deps,iri_firewalld,iri_ufw,iri_ssl,
↪iotapm_role,monitoring_role
```

Access

To access the **fullnode monitoring graphs**, point your browser to `http://YOUR-IP:5555` and use the username and password you've configured earlier to log in.

To access the **IOTA Peer Manager** (assuming you've installed nginx), point your browser to `http://YOUR-IP:8811` and use the username and password you've configured earlier to log in.

If you haven't install nginx and want to access IOTA Peer Manager, it is not configured to be accessible externally by default. It would pose a security risk to your server running it exposed and not locked with a password. As an alternative you can use a SSH tunnel to bind to it (port 8011). See *2. Tunneling IRI API for Wallet Connection*.

1.5.7 Install Nelson

It is possible to install Nelson as part of this installation.

Warning: Nelson is still at beta stage.

Nelson depends on IRI being installed and running. Please check `/opt/iri-playbook/group_vars/all/nelson.yml` and configure to match your environment.

If you installed using the Getting Started Quickly chapter, you can just proceed to the installation below.

Installation

- If you installed this playbook before Nelson was added you need to update the git repository. Run:

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

- To install Nelson, run:

```
cd /opt/iri-playbook && ansible-playbook -i inventory -v site.yml --tags=nelson_role -
↪e "nelson_enabled=true"
```

You can stop, start and restart nelson via `systemctl (start|stop|restart) nelson`.

Join the `#nelson-peering` channel on IOTA's Discord if you have questions regarding Nelson.

Upgrade Nelson Version

Run the upgrade command:

```
cd /opt/iri-playbook && ansible-playbook -i inventory -v site.yml --tags=nelson_role -
↪e "upgrade_nelson=true" -e "nelson_enabled=true"
```

View Status/Logs and configuration

- To view nelson status run: `systemctl status nelson`.
- To view nelson logs run: `journalctl -u nelson`.

Or `journalctl --no-pager -n50 -u nelson` to view 50 last lines of Nelson's log.

- Nelson's configuration file can be found here: `/etc/nelson/nelson.ini`.
- Nelson's data directory can be found here: `/var/lib/nelson/data`.

1.5.8 Install Field

Please visit Carriota Field on Github to learn more about what it is.

Field has been added to the playbook as an optional add-on. The recommended way to install it is using the `iric` configuration tool.

You might want to upgrade the `iric` tool (there's an option for that in the menu) if you are missing the option to enable Field.

Note: The playbook also installs field_exporter to show stats on Grafana. If you already have Field installed and don't have the field_exporter installed yet: make sure you have the latest `iric` and then choose to update field (proceed with the update when asked).

Manual Installation

The variables file for Field is in `/opt/iri-playbook/group_vars/all/field.yml`. If you want to change any of the variables you can copy the file to: `/opt/iri-playbook/group_vars/all/z-field.yml` and edit this file. It will override anything in the original file when running the playbook.

Another option is to override variables adding `-e some_var=someval` for any variable you want to override on the Ansible command line.

The manual procedure to install Field:

```
cd /opt/iri-playbook && git pull && ansible-playbook -i inventory -v site.yml --
↪tags=prometheus_config_file,field_exporter,field_role -e field_enabled=yes
```

This will result in Field installed and configured. You should check the configuration file at `/etc/field/field.ini` to configure your payout address and node's name.

Control Field

To restart Field run:

```
systemctl restart field
```

To stop Field run:

```
systemctl stop field
```

To view the logs:

```
journalctl -u field
```

And use SHIFT-g to skip to the end of the logs.

Again, the recommended way to enable, upgrade and manage Field is via the `iric` tool that ships with the playbook.

Note: For more information and support with Carriota Field please join IOTA's Discord and find "#carriota-field" channel.

1.6 Post Installation

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

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

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

See examples in the chapters below:

- *Controlling IRI*
- *Controlling IOTA Peer Manager*
- *Checking Ports*
- *Checking IRI Full Node Status*
- *Connecting to IOTA Peer Manager*
- *Adding or Removing Neighbors*
- *Install IOTA Python libs*

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

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

link.

1.6.1 Controlling IRI

Check status:

```
systemctl status iri
```

Stop:

```
systemctl stop iri
```

Start:

```
systemctl start iri
```

Restart:

```
systemctl restart iri
```

1.6.2 Controlling IOTA Peer Manager

Check status:

```
systemctl status iota-pm
```

Stop:

```
systemctl stop iota-pm
```

Start:

```
systemctl start iota-pm
```

Restart:

```
systemctl restart iota-pm
```

1.6.3 Checking Ports

IRI uses 3 ports by default:

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

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

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

Here is the output you should expect:

```
# lsof -Pni|egrep "iri|iotapm"
java     2297     iri   19u  IPv6  20331      0t0  UDP *:14600
java     2297     iri   21u  IPv6  20334      0t0  TCP *:14600 (LISTEN)
java     2297     iri   32u  IPv6  20345      0t0  TCP 127.0.0.1:14265 (LISTEN)
node     2359 iotapm   12u  IPv4  21189      0t0  TCP 127.0.0.1:8011 (LISTEN)
```

What does this tell us?

1. `*:<port number>` means this port is listening on all interfaces - from the example above we see that IRI is listening on ports TCP and UDP no. 14600
2. IRI is listening for API (or wallet connections) on a local interface (not accessible from "outside") no. 14265
3. Iota-PM is listening on local interface port no. 8011

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

Here's how to check your IP address:

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

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

(continues on next page)

(continued from previous page)

```
    inet6 fe80::c5f4:d95b:ba52:865c/64 scope link
       valid_lft forever preferred_lft forever
```

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

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

I could tell neighbors to connect to my UDP port: `udp://10.50.0.14:14600` or to my TCP port: `tcp://10.50.0.14:14600`.

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

1.6.4 Checking IRI Full Node Status

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

For example, we can run:

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

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

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

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

1.6.5 Connecting to IOTA Peer Manager

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

1.6.6 Adding or Removing Neighbors

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

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

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

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

nbctl script

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

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

Add or remove full node neighbors.

optional arguments:
  -h, --help            show this help message and exit
  --neighbors NEIGHBORS, -n NEIGHBORS
                        Neighbors to process. Can be specified multiple times.
  --remove, -r          Removes neighbors
  --add, -a             Add neighbors
  --list, -l            List neighbors
  --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 udp://1.2.3.4:12345 -n tcp://4.3.2.1:4321 -f /etc/default/iri
```

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

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

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

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

Note that the last options `-f /etc/default/iri` will also add the neighbors to the configuration file, but **make sure** you are pointing to the correct file. For example, in CentOS it is `/etc/sysconfig/iri`, on other guides it is locted 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 udp://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 locted 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":["udp://neighbor-ip:port", "udp://neighbor-ip:port", "udp://
↪[2a01:a0a0:c0c0:1234::1]:14600"]}' http://localhost:14265
```

to remove:

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

to list:

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

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

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

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

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

On **Ubuntu**:

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

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

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

1.6.7 Install IOTA Python libs

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

On **Ubuntu**:

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

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

```
reattach -h
```

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

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

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

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

1.7 Full Node Remote Access

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

1.7.1 1. Exposing IRI Port Externally

IRI has a command-line argument ("option") `--remote`. 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` option, IRI will "listen" on the external interface/IP.

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

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

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

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

Edit the file and find the line:

```
OPTIONS=""
```

and add `--remote` to it:

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

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` option in the configuration file) you will need to allow the port in the firewall.

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

On **CentOS**:

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

On **Ubuntu** without rate limiting:

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

On **Ubuntu** with rate limiting:

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

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

Allowing the port manually

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

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

On **Ubuntu**:

```
ufw allow 14265/tcp
```

And to add rate limits:

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

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

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

1.7.2 2. Tunneling IRI API for Wallet Connection

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

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

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

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

What do you need to "forward" the IRI API?

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

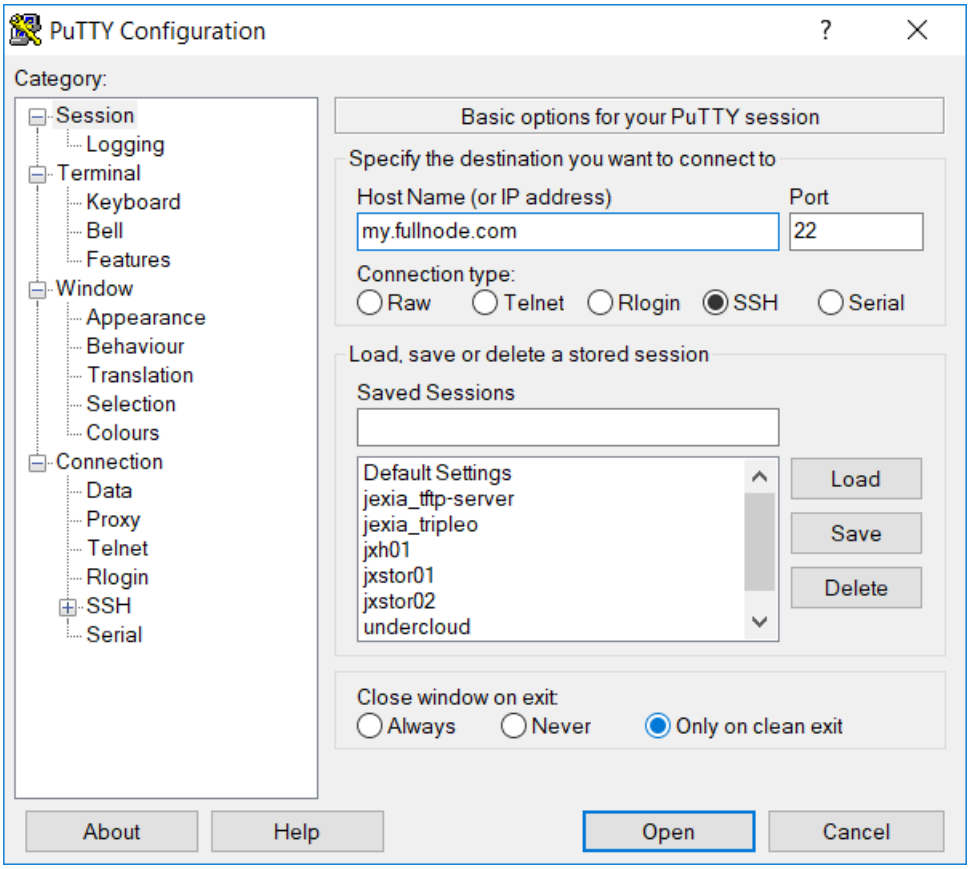
A default installation would have IRI API listening on TCP port 14265.

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

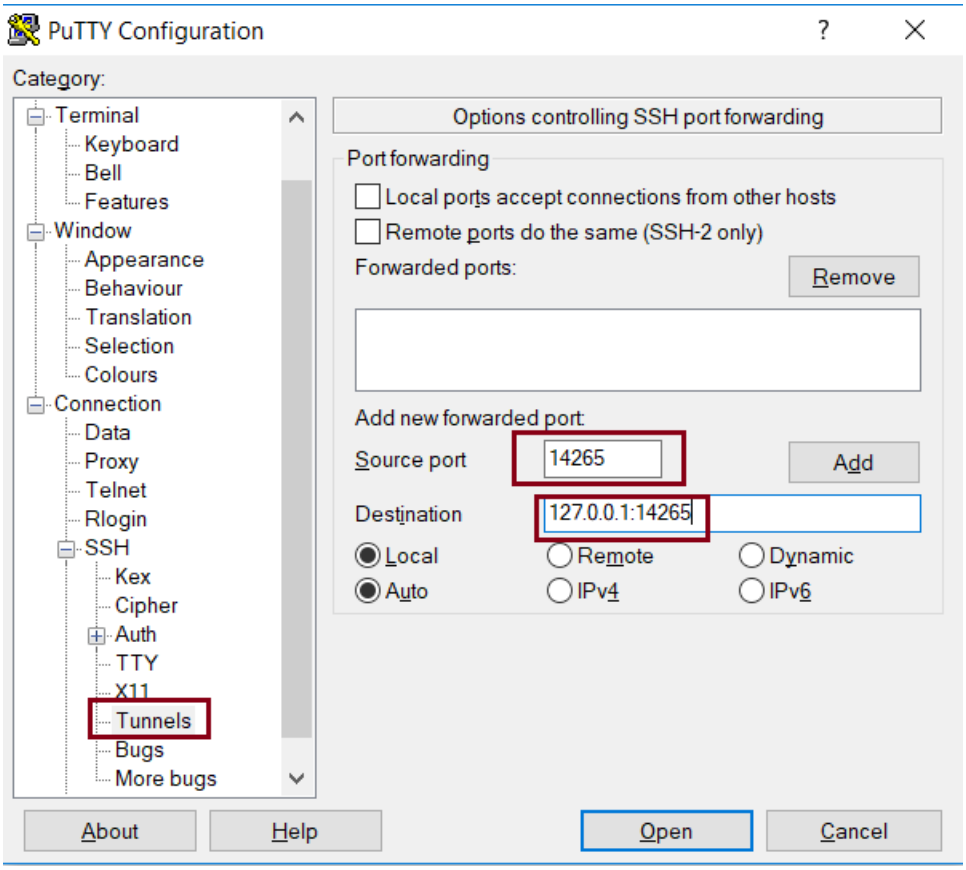
For Windows desktop/laptop

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

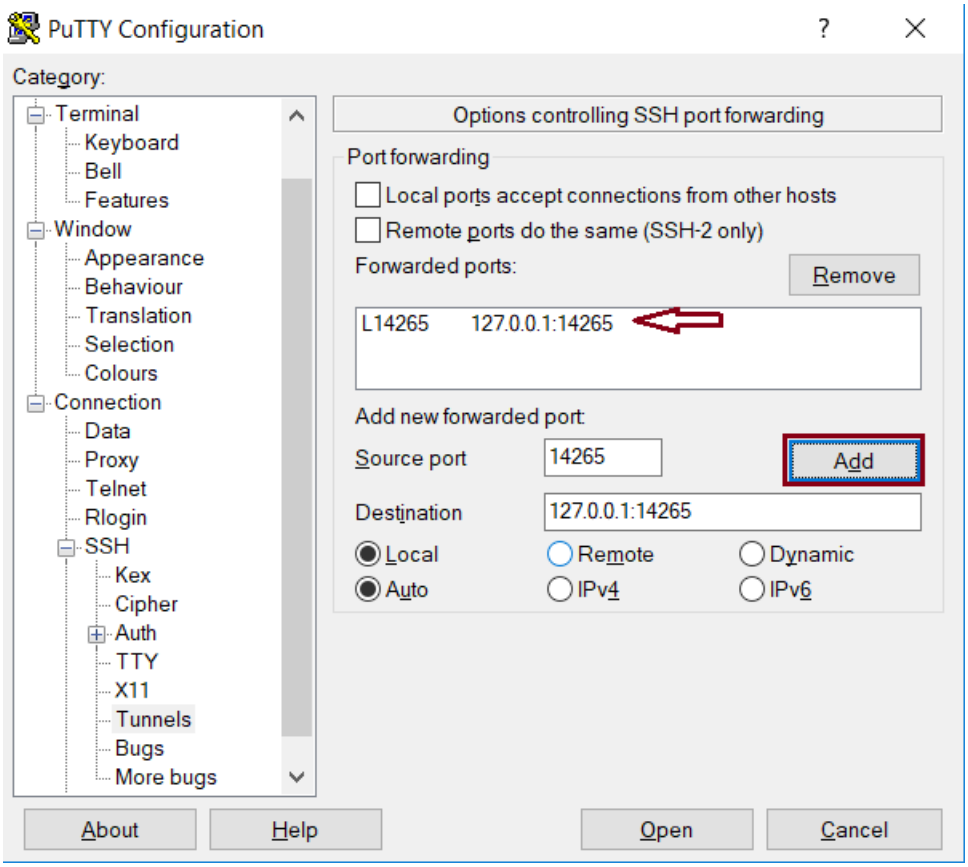
1. Open putty and create a new session name. Start by entering the node's address and SSH port.



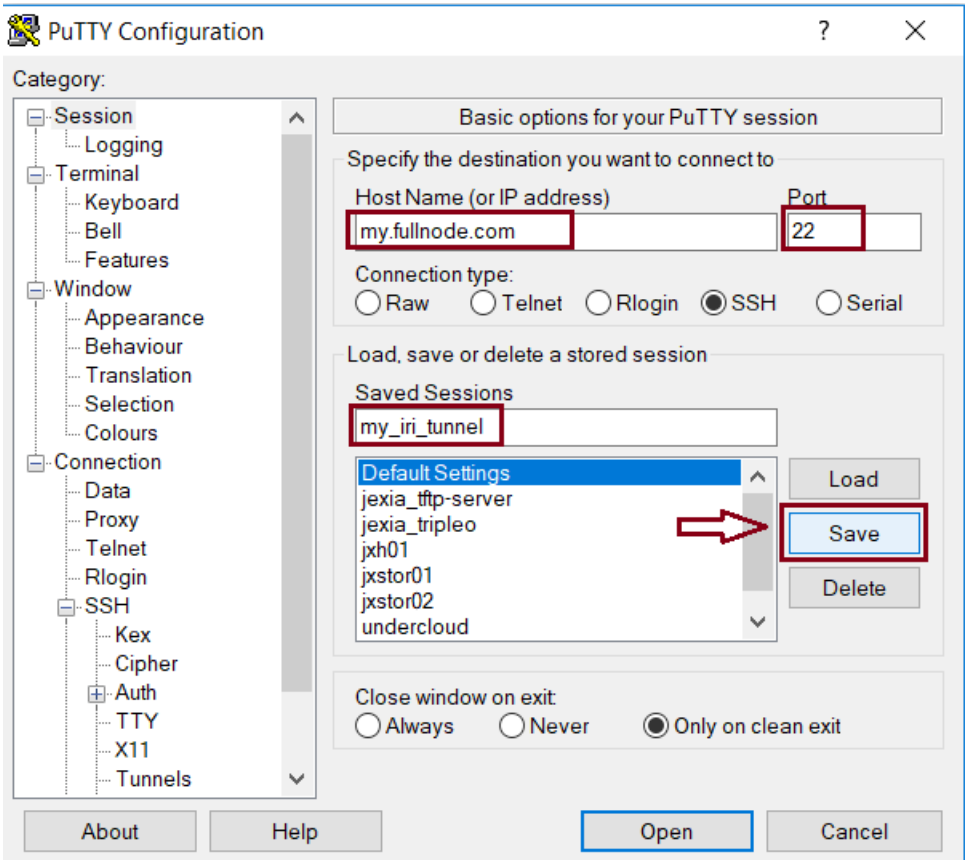
2. On the menu on the left choose 'Tunnels'. Then fill in the Source port and Destination as shown in the image below. The destination is comprised of the IP address and the port. We use 127.0.0.1:14265, as this is by default where we want to forward the port from.



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



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



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

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

```
REMOTE_LIMIT_API="removeNeighbors, addNeighbors, interruptAttachingToTangle,␣
↪attachToTangle, getNeighbors"
```

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

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

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

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

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

1.8 Files and Locations

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

IRI configuration file (changes require iri to restart):

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

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

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

IRI installation path:

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

IRI database:

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

Grafana configuration file:

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

Grafana Database file:

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

Prometheus configuration file:

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

IOTA-Prom-Exporter configuration file:

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

Alert Manager configuration file:

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

HAProxy configuration file:

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

Nelson configuration file:

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

Field configuration file:

```
/etc/field/field.ini
```

Field Exporter configuration file:

```
/opt/prometheus/field_exporter/config.js
```

1.9 Maintenance

1.9.1 Upgrade IRI

Latest IRI release is available here.

If a new version has been announced, you can follow this guide to get the new version.

In the following example we assume that the new version is **1.4.2.4**.

Note: The foundation might announce additional information in tandem with upgrades, for example whether to use the `--rescan` flag, remove older database etc. If required, additional options can be specified under

the `OPTIONS=""` value in the configuration file (`/etc/default/iri` for Ubuntu or `/etc/sysconfig/iri` for CentOS). The database folder is in `/var/lib/iri/target/mainnetdb` and can be removed using `systemctl stop iri && rm -rf /var/lib/iri/target/mainnet*`.

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

To update manually:

Make sure you are running all the commands as 'root' (run `sudo su` first). Then, download new IRI to the directory:

```
export IRIVER=1.4.2.4 ; curl -L "https://github.com/iotaledger/iri/releases/download/v
↪${IRIVER}/iri-${IRIVER}.jar" --output "/var/lib/iri/target/iri-${IRIVER}.jar"
```

Then update the IRI configuration file in place using `sed`:

In **Ubuntu**:

```
sed -i 's/^IRI_VERSION=.*$/IRI_VERSION=1.4.2.4/' /etc/default/iri
```

In **CentOS**:

```
sed -i 's/^IRI_VERSION=.*$/IRI_VERSION=1.4.2.4/' /etc/sysconfig/iri
```

This will update the version line to match, e.g.:

```
IRI_VERSION=1.4.2.4
```

This requires a iri **restart**: `systemctl restart iri`.

To verify the new version is loaded:

```
ps aux|grep iri-1.4.2.4|grep -vq grep && echo found
```

Of course, replace the version with the one you expect to see.

This should output `found` if okay.

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*, or update manually using the following instructions:

A new feature has been added to read extra metrics from IRI using ZeroMQ. ZMQ has to be enabled in IRI first **if you haven't done it already**:

```
grep -q ^ZMQ_ENABLED /var/lib/iri/iri.ini || echo "ZMQ_ENABLED = true" >>/var/lib/iri/
↪iri.ini && systemctl restart iri
```

After about 10-30 seconds (depending on how long it takes IRI to restart) you should be able to see the ZMQ port listening for connections:

```
lsof -Pni:5556
```

Output should look similar to:

```
java      5192        iota    47u  IPv6 38464889      0t0  TCP *:5556 (LISTEN)
```

Next we can update iota-prom-exporter and the respective Grafana dashboard:

```
cd /opt/iri-playbook && git pull && ansible-playbook -i inventory -v site.yml --
↪tags=iri_ssl,prometheus_config,monitoring_deps,iota_prom_exporter,grafana_config -e␣
↪overwrite=yes
```

Now you should be able to open Grafana and see the new row of metrics (ZMQ).

If you encounter errors when running the command, depending on the error, please refer to *HTTP Error 401 Unauthorized When Running Playbook* or *How to Handle Git Conflicts*.

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 paritions use `df -h` If one of the paritions' usage exceeds 85% you should consider a cleanup. Don't worry about the /boot paritition 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

At any time you can remove the existing database and start sync all over again. This is required if you know your database is corrupt (don't assume, use the community's help to verify such suspicion) or if you want your node to sync more quickly.

To remove an existing database:

1. stop IRI: `systemctl stop iri`.
2. delete the database: `rm -rf /var/lib/iri/target/mainnet*`
3. start IRI: `systemctl start iri`

If you want to import an already existing database, check *Where can I get a fully synced database to help kick start my node*.

1.10 Security Hardening

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

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

1.10.1 SSH Key Access

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

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

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

Overview

We are going to:

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

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

Access User

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

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

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

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

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

2. Set a password for the new user:

```
passwd myuser
```

3. Add the user to the "sudoers":

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

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

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

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

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

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

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

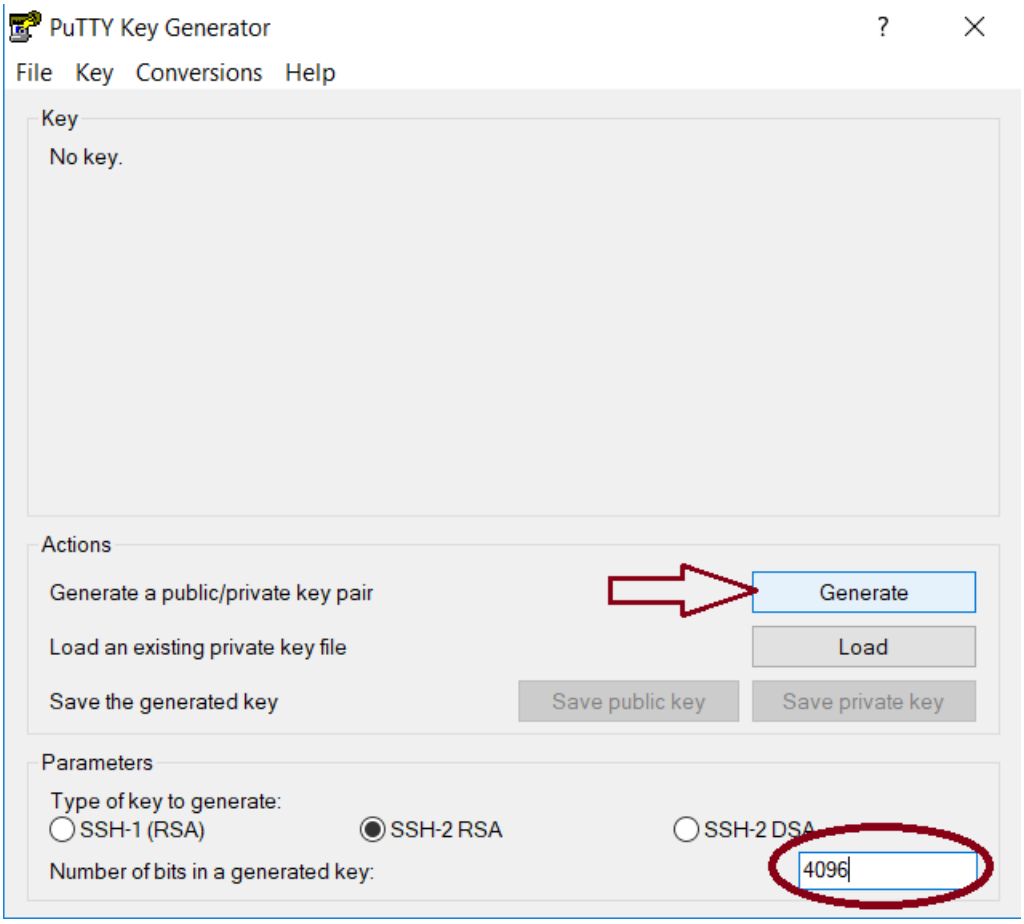
Creating SSH Keys

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

The installer includes: putty, puttygen and pagent.

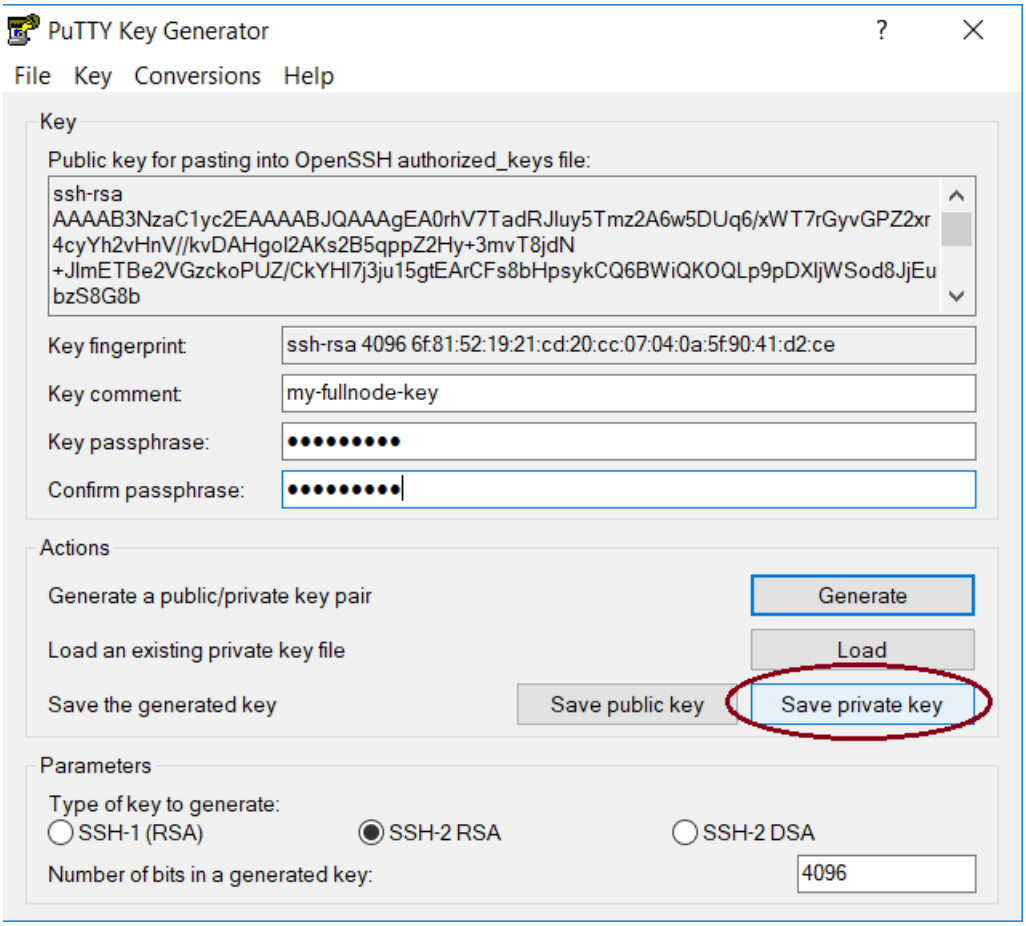
The first step is to create SSH keys. A SSH key pair consists of a private key and a public key (**never** share your private key with anyone and keep it safe!).

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



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

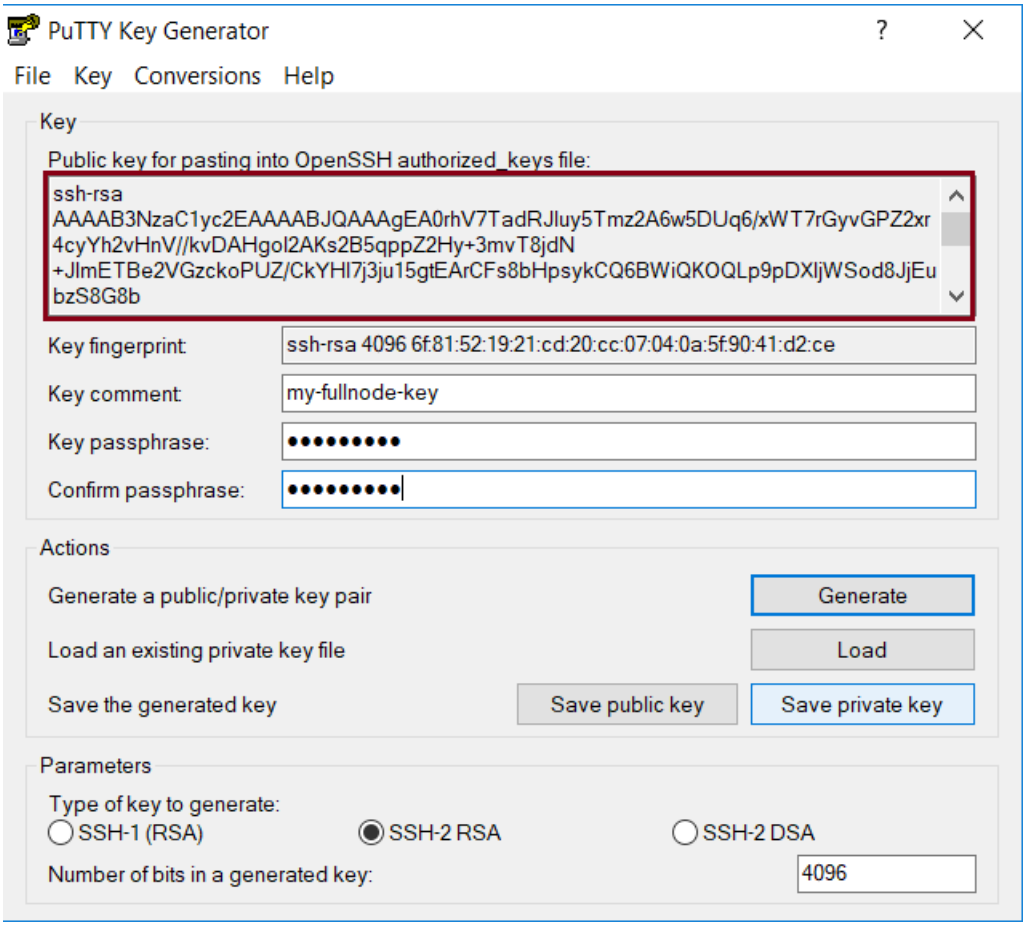
Remember where you save the key to. We are going to use it in the following steps.



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

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

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



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

6. Set correct permissions:

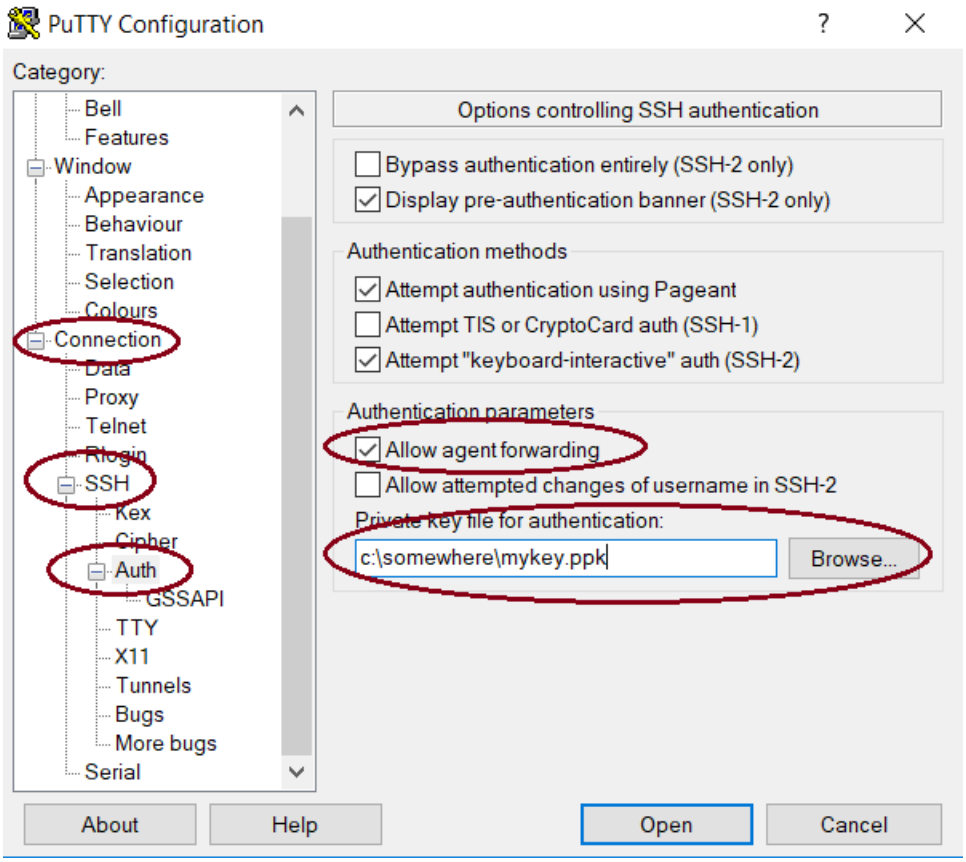
```
chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys
```

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

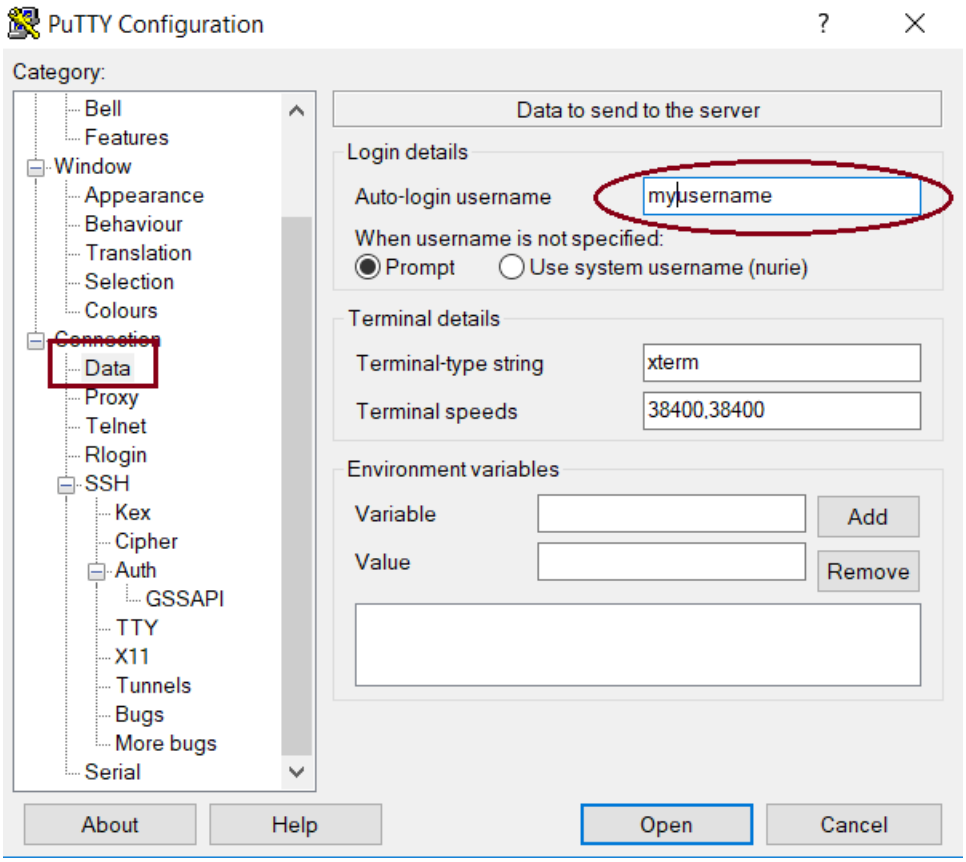
You can close Puttygen. If needed, you can always run puttygen again, create a new key, load an existing key (it will require the password you've configured with it), replace the password or copy the public key from it.

Access Using the SSH Keys

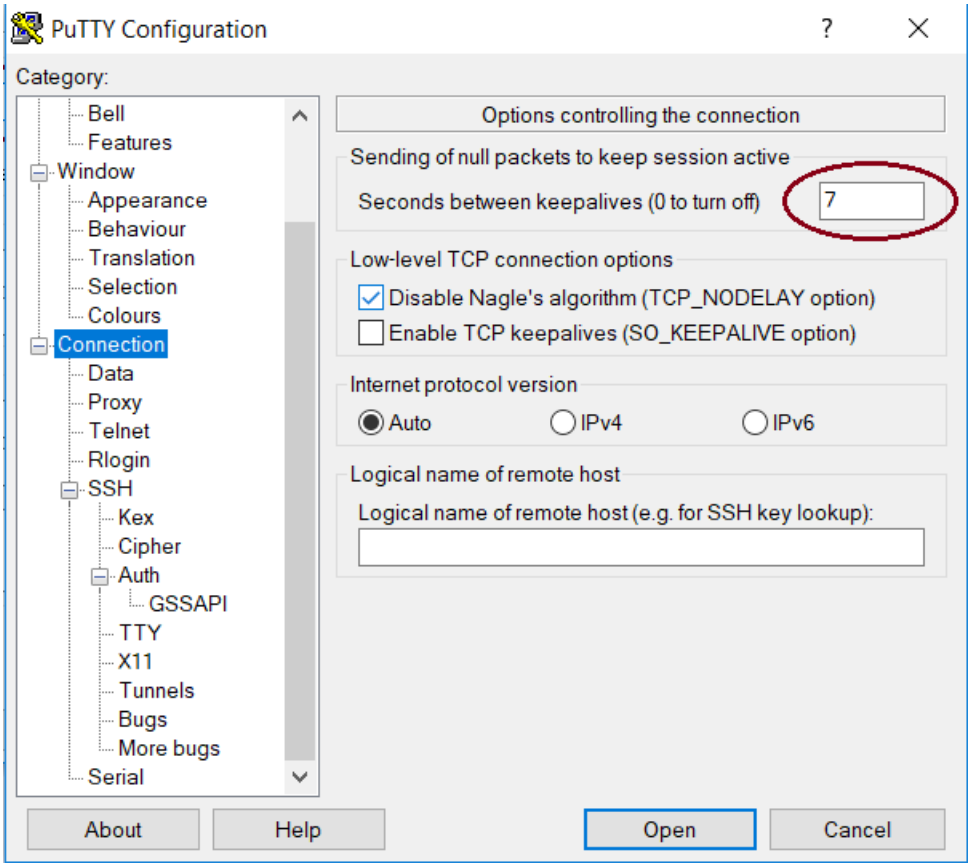
1. Open the application `Putty`. On the left side you will have a tree browser. Open "Connection", "SSH", and "Auth". Configure as shown in the image below, browse the file system to select the private ssh key you've created earlier:



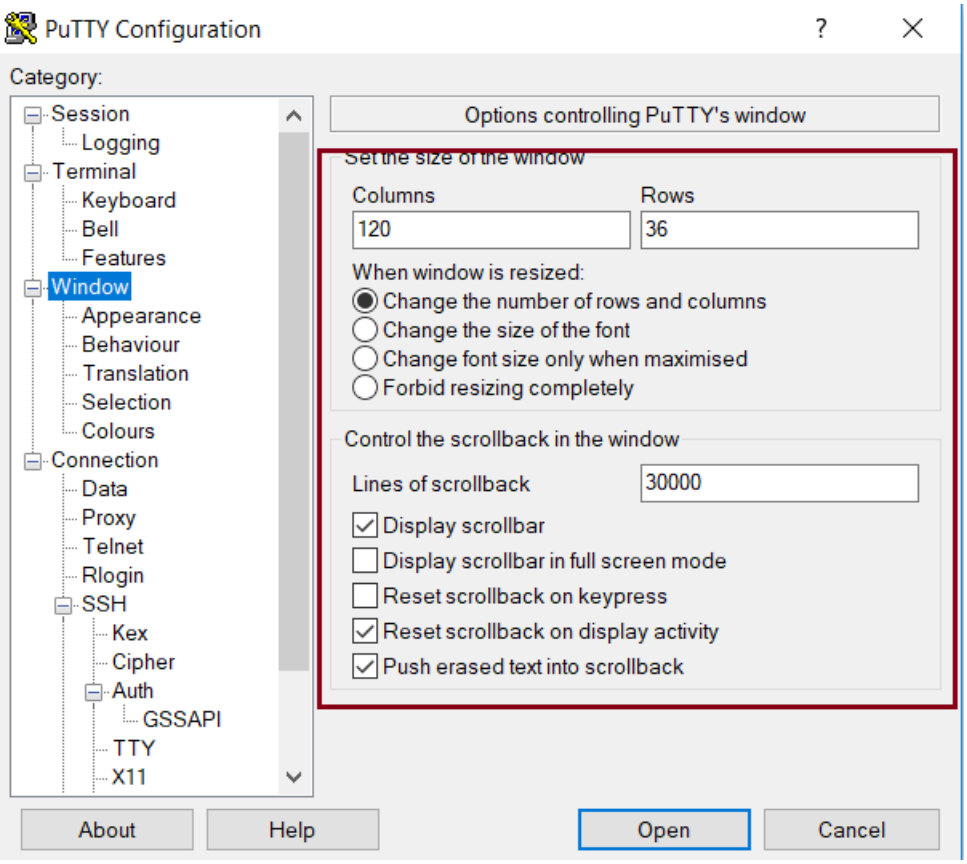
2. Next open the “Data” option and set the username you’ve created on the server:



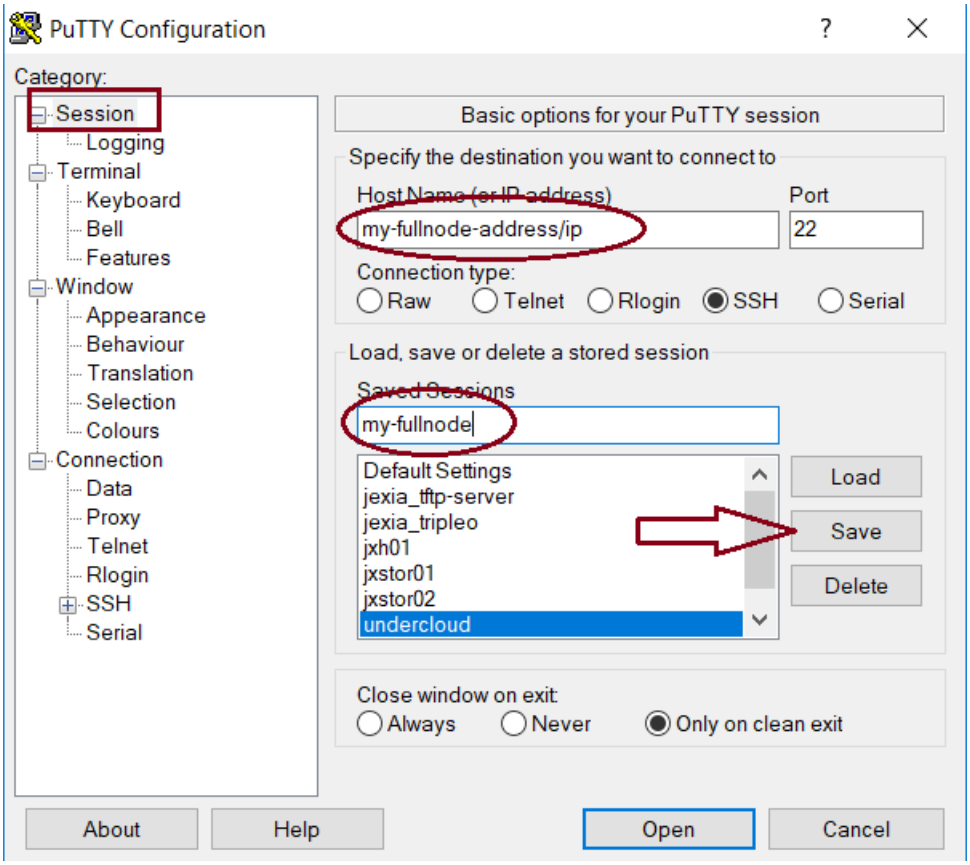
3. Then, on the “Connection”, set a keepalive value (5 or 7 is fine):



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



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



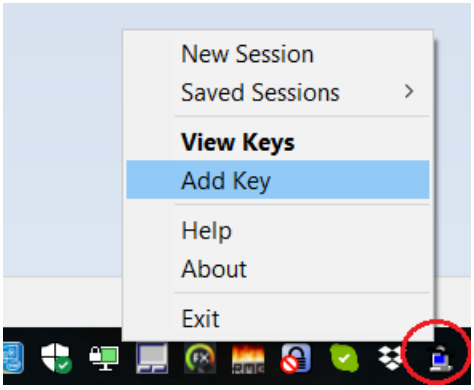
6. Now, or any time you open Putty, you can select this saved session and click "Open". This should connect you to the server. You will be asked to provide the SSH key password (not the user's password from the server!)

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.

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

1.10.2 Disabling Password Authentication

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

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

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

Disable

Disable SSH password authentication:

```
sed -i 's/^PasswordAuthentication.*/PasswordAuthentication no/g' /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
```

1.10.3 Disabling SSH Root Access

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

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

Disable:

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

And restart ssh daemon:

```
systemctl restart sshd
```

To re-enabled root access:

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

And restat sshd daemon as shown above.

1.10.4 Using Alternative SSH Port

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

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

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

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

Firewall

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

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

CentOS

Run:

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

And tell Selinux we want to use this port:

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

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

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

Ubuntu

Run:

```
ufw allow 9922/tcp
```

SSH Daemon

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

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

```
Port 9922
```

Save the file and restart sshd daemon:

```
systemctl restart sshd
```

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

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

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

1.11 Fullnode IRI Configuration Utility

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

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

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

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

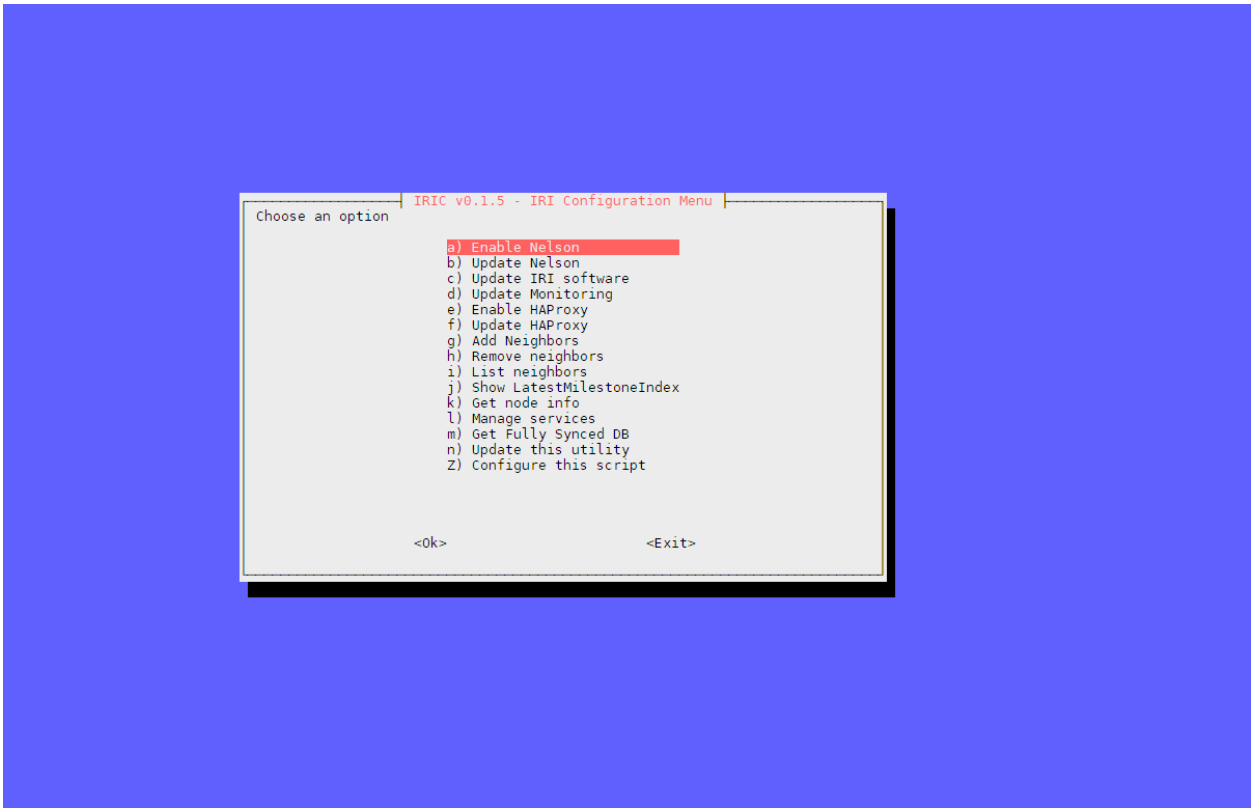
Instead of having to copy & paste long commands from the documentation, one can choose to use this menu-driven utility.

Feel free to use it. Feedback is most welcome.

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

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

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



Not much to explain here. Enjoy!

1.12 Troubleshooting

1.12.1 Upload Logs to Pastebin

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

First make sure you have the pastebin tool installed.

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

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

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

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

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

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

Ubuntu Logs

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

The two `sed` commands can be added in between any command to hide IP addresses: `sed 's/\([0-9]\{1,3\}\.\)\{3,3\}[0-9]\{1,3\}/x.x.x.x/g'|sed -r 's#:\[.*\]:([0-9]+)#:\[xxxx:xxxx:xxxx:xxxx\]:\1#g'`.

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

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


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

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

CentOS Logs

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

The two `sed` commands can be added in between any command to hide IP addresses: `sed 's/\([0-9]\{1,3\}\.\)\{3,3\}[0-9]\{1,3\}/x.x.x.x/g'|sed -r 's#:\[.*\]:([0-9]+)#:\[xxxx:xxxx:xxxx:xxxx\]:\1#g'`.

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

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


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

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

1.12.2 How to Handle Git Conflicts

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

Background

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

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

The other solution is the one I am presenting here.

Backup My Changes

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

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

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

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

```
git stash
```

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

It is recommended not to edit the variable files in order to avoid such conflicts. You can better create "override" files *How to override playbook variables*

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

```
git status
```

And view the changes you've applied:

```
git diff
```

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

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

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

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

Apply Changes

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

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

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

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

1.12.3 HTTP Error 401 Unauthorized When Running Playbook

This is how the error would look like:

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

PLAY RECAP␣
↪*********************************************************************************************
```

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

Solution A

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

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

The result should look like:

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

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

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

Solution B

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

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

1. Stop grafana-server:

```
systemctl stop grafana-server
```

2. Delete grafana's database:

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

3. Edit `/etc/grafana/grafana.ini`, set correct values for `admin_user` and `admin_password`.
4. Start grafana-server:

```
systemctl start grafana-server
```

Now you should be able to login to grafana.

1.12.4 Error Starting up Nelson After Upgrade

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

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

```
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]: 20:57:40.241
↪16600::NODE  terminating...
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]: Unhandled Rejection at:
↪Promise Promise {
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]:   <rejected> Error:
↪"toString()" failed
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]:     at stringSlice (buffer.
↪js:560:43)
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]:     at Buffer.toString
↪(buffer.js:633:10)
Jan 29 20:57:40 vmi111112.shintaboserver.net nelson[3178]:     at FSReqWrap.
↪readFileAfterClose [as oncomplete] (fs.js:506:23) } reason: Error: "toString()"
↪failed
Jan 29 20:57:40 vmi1111112.contaboserver.net nelson[3178]:     at stringSlice (buffer.
↪js:560:43)
```

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

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

Start up nelson, and check the status again:

```
systemctl start nelson
```

Status:

```
systemctl status nelson
```

1.12.5 Error Starting or Restarting IRI

Examples of errors:

Hostname can't be null

If you get this message in the logs:

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

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

Make sure all neighbors adhere to the format examples:

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

1.12.6 Fix Nginx

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

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

And then restart nginx:

```
systemctl restart nginx
```

You can verify it is working via:

```
systemctl status nginx
```

It should be active.

1.13 FAQ

1.13.1 How to override playbook variables

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

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

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

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

Overriding file variables

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

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

```
test_var: 1234
```

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

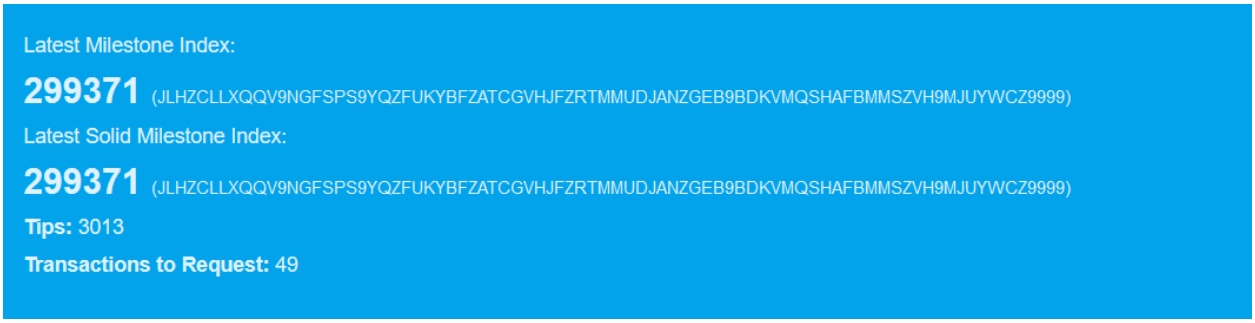
```
test_var: abcd
```

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

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

1.13.2 How to tell if my node is synced

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



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

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

This will output 2 numbers which should be equal.

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

You can install `jq`:

Ubuntu: `apt-get install jq -y`

Centos: `yum install jq -y`

Alternatively, use python:

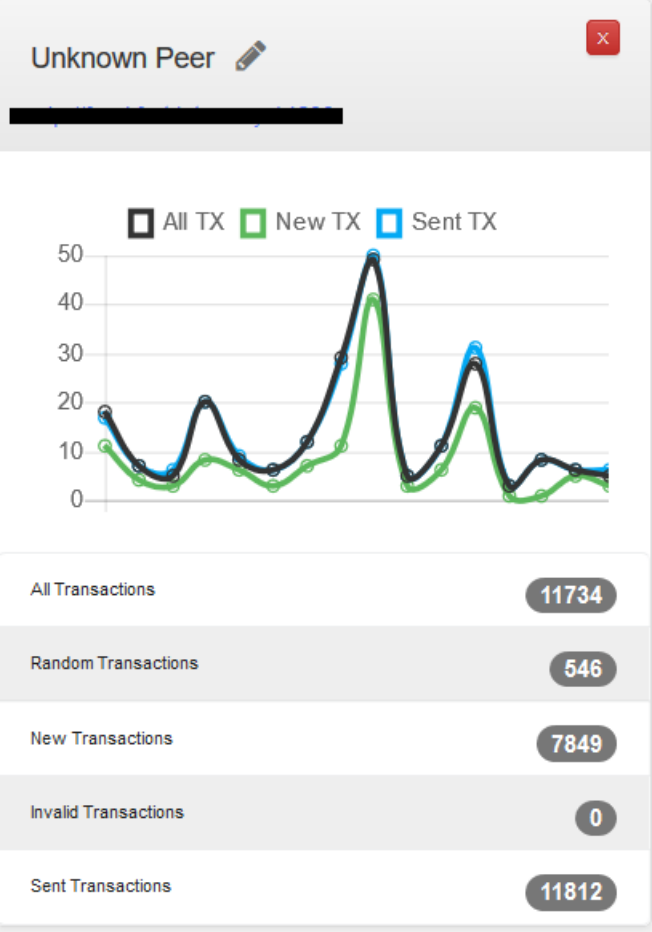
```
curl -s http://localhost:14265 -X POST  -H 'X-IOTA-API-Version: 1' -H 'Content-Type:␣
↪application/json' -d '{"command": "getNodeInfo"}'|python -m json.tool|egrep
↪"latestSolidSubtangleMilestoneIndex|latestMilestoneIndex"
```

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

If the issue still persists, perhaps difficulties syncing are related to this: whyAmISeeingUDPBadLength

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

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



1.13.4 Where to get the latest milestone index from

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

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

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

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

1.13.5 Why is latestSolidSubtangleMilestoneIndex always behind latestMilestoneIndex

This is probably the most frequently asked question.

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

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

attacks directed against the network.

Some things to try:

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

1.13.6 How to get my node swap less

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

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

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

and:

```
sysctl -p
```

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

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

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

1.13.7 What are the revalidate and rescan options for

Here's a brief explanation what each does, courtesy of Alon Elmaliah:

Revalidate "drops" the stored solid milestone "table". So all the milestones are revalidated once the node starts (checks signatures, balances etc). This is used it you take a DB from someone else, or have an issue with solid milestones acting out.

Rescan drops all the tables, except for the raw transaction trits, and re stores the transactions (refilling the metadata, address indexes etc) - this is used when a migration is needed when the DB schema changes mostly.

It is possible to add these options to the IRI configuration file (or startup command):

`--revalidate` or `--rescan`.

If you have used this installation's tutorial / automation, you will find the configuration file in the following location:

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

You will see the OPTIONS variable, so you can tweak it like so:

```
OPTIONS="--rescan"
```

and restart IRI to take effect: `systemctl restart iri`

Note: Once you've restarted the service with the `--rescan` or `--revalidate` options you can remove the option from the configuration file. If it stays in the configuration file, subsequent restarts will use that option again, perhaps when you do not explicitly choose to enable it.

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

For the sake of the community, I regularly create a copy of a fully synced database.

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

NOTE I am providing this database copy to help the community. Making this possible involves increasing costs due to the frequent downloads/bandwidth usage. Please consider donating to help keep this possible:

```
CSSFHHDBUQDGAUGYUHTENLBJ9JMTUFFLYLJZKTLRZVLLDCZZOQHOUXJOVDKXOLXGCJEMXJOULDIKADBHWMGVALMAUW
```

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

```
cd /tmp && wget -O iota.db.tgz https://x-vps.com/iota.db.tgz && systemctl stop iri &&␣
↪rm -rf /var/lib/iri/target/mainnetdb* && mkdir /var/lib/iri/target/mainnetdb/ && pv␣
↪iota.db.tgz | tar xzf - -C /var/lib/iri/target/mainnetdb/ && chown iri.iri /var/lib/
↪iri -R && rm -f /tmp/iota.db.tgz && systemctl start iri
```

Note: There was some debate on the slack channel whether after having imported a foreign database if it is required to run IRI with the `--revalidate` or `--rescan` flags. Some said they got fully synced without any of these.

To shed some light on what these options actually do, you can read about it in *What are the revalidate and rescan options for*

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

There are commonly two reasons for this to happen:

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

See *Full Node Remote Access*

1.14 Uninstall

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

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

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

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

1.15 Command Glossary

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

1.15.1 Check IRI's node status

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

Same as above but extract the milestones

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

1.15.2 Add neighbors

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

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

1.15.3 Remove neighbors

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

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

1.15.4 Check iri and iota-pm ports listening

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

1.15.5 Check all ports on the node

```
lsof -Pni
```

1.15.6 Opening a port in the firewall

In **CentOS**:

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

In **Ubuntu**:

```
ufw allow 14265/tcp
```

1.15.7 Checking memory usage per application

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

```
ps_mem
```

1.15.8 Checking system load and memory usage

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

You might need to install it:

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

Then run `htop`

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

1.16 Appendix

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

It is meant for more advanced usage.

1.16.1 Using Fully Qualified Domain Name for my server

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

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

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

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;
  }
}
```

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** 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.

1.16.2 Configuring my server with HTTPS

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/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** 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

---

1.16.3 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.

1.16.4 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

1.16.5 Restart IRI On Latest Subtangle Milestone Stuck

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

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

Warning: This feature is disabled by default as this is not considered a permanent or ideal solution. Please, first try to download a fully sycned database as proposed in the faq, or try to find "healthier" neighbors.

Enabling the Feature

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

You will find the following lines:

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

Remove the # comments, resulting in:

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

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

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

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

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

Disabling the Feature

A quick way to disable this feature:

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

To re-enable:

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

Configuring the Feature

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

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

```
# If latest subtangle milestone doesn't increase for 30 minutes
- alert: MileStoneNoIncrease
  expr: increase(iota_node_info_latest_subtangle_milestone[30m]) == 0
    and iota_node_info_latest_subtangle_milestone != 243000
  for: 1m
  labels:
    severity: critical
  annotations:
```

(continues on next page)

(continued from previous page)

```
    description: 'Latest Subtangle Milestone increase is {{ $value }}'
    summary: 'Latest Subtangle Milestone not increasing'
```

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

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

Upgrading the Playbook to Get the Feature

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

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

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

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

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

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

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

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

1.16.6 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       |
|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
```

(continues on next page)

(continued from previous page)

```
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out␣
↪any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted␣
↪now it is to install the new keys
root@other-node-name-or-ip's password:

Number of key(s) added: 1

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

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

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

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

1.16.7 Using Nano to Edit Files

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

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

First, ensure you have `nano` installed:

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

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

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

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

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

1.16.8 Running IRI API Port Behind HAProxy

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

- Logging

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

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

```
cd /opt/iri-playbook && git pull && ansible-playbook -i inventory -v site.yml --
↪tags=iri_ssl,loadbalancer_role -e lb_bind_address=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/haroxy.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*

1.16.9 Installation Options

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

Nelson

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

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

You can read more about it here.

Field

Field is a proxy for your IRI node that sends regular statistics to the Field server.

You can read more about it here.

In addition to field, field-exporter is installed which provides metrics about the node's performance in the Field and other stats from the Field server.

You can read more about it here.

HAproxy

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

You can read more about it here: *Running IRI API Port Behind HAProxy*.

Monitoring

The monitoring refers to installation of:

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

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

ZMQ Metrics

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

1.16.10 Upgrade IRI and Remove Existing Database

(option #3 from the IOTA Snapshot Blog)

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

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

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

1. Stop IRI:

```
systemctl stop iri
```

2. Remove the existing database:

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

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

If you don't have `iric` installed, you can refer to this chapter on how to upgrade IRI manually *Upgrade IRI*.

1.16.11 Upgrade IRI and Keep Existing Database

(option #2 from the IOTA Snapshot Blog)

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

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

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

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

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

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

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

1.17 Disclaimer

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

1.18 Donations

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

```
CSSFHHDBUQDGAUGYUHTENLBJ9JMTUFFLYLJZKTLRZVLLDCZZOQHOUXJOVDKXOLXGCJEMXJOULDIKADBHWMGVALMAUW
```

Thanks!