Slurp'it
Beyond network discovery

Installation manual

Slurp’it

slurpit.io

07 October 2024

Site
https://www.slurpit.io

Knowledge base
https://slurpit.io/knowledge-base

Slurp'it
Beyond network discovery

Index

Introduction

This is the installation manual for Slurp’it, it contains the common steps to get the different parts of Slurp’it up and running. Do note that these are not the only ways to run the software, or necessarily the best way for your specific environment. Consult your local sys admins for recommendations.

In this manual we will use docker to run the different images and expect docker to be installed on the different machines.

Slurp'it
Beyond network discovery

1. Docker

1.1 Info
Docker is used to create and run scalable and easy to manage images.

The available images for Slurp'it can be found at
https://gitlab.com/slurpit.io/images/container_registry

The warehouse, scraper and scanner are both python applications and portal is a web service.

1.2 Requirements
Portal needs to be able to access both the scraper and scanner on port 80. Make sure PORTAL_WAREHOUSE_URL has the correct hostname

Warehouse also needs to be able to access Portal on port 80. Make sure WAREHOUSE_CALLBACK_SCANNER_URL and WAREHOUSE_CALLBACK_SCANNER_URL have the correct hostname.

1.3 Compose
Docker compose is a very basic way of getting the different images up and running.
In the Git project is an example Compose file which contains all the basic settings to run the application. In case you want to make changes to the compose file it's better to do this in the override file, when you remove -EXAMPLE from the name it will automatically become active. In here you can then make all the changes so when you pull the latest git updates it won't conflict with the original compose file.

1.4 Example docker-compose.yml
On the https://gitlab.com/slurpit.io/images repository is an example of a docker-compose.yml file available.

1.5 Start the application
Just run up.sh or up.bat

1.6 Stop the application
Just run down.sh or down.bat

1.7 Install docker on Windows Server
We had a couple requests on how to install Docker on Windows Server. For this reason we will add seperate instructions for this in the manual. On Windows Desktop you can easily install Docker desktop and follow the steps in the installer. On Windows Server the steps are a bit different because we have to install the Linux kernel.

 # Windows Server update
Make sure your server is up-to-date.

Run powershell as Administrator and execute the following steps

 # Enable the containers feature
Install-WindowsFeature -Name Containers

 # Enable the Windows Subsystem for Linux

dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

 # Enable Virtual Machine feature
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

Restart your machine.

 # Enable Linux subsystem
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux

 # Set WSL to version 2
wsl --set-default-version 2

Restart your machine to complete the WSL install and update to WSL 2.

 # Download the latest ubuntu
wsl -install -d Ubuntu

 # Install Docker desktop and choose during the installation to use WSL.
https://www.docker.com/products/docker-desktop

Slurp'it
Beyond network discovery

2. Podman

2.1 Info

Podman is like docker, but maintained and supported by RedHat. It's used like docker to create and run scalable and easy to manage images. But we can recommend if you have limited experience with Podman to download our OVA or use Docker since Podman is a bit more difficult to start with.
https://slurpit.io/download

The available images for Slurp'it can be found at
https://gitlab.com/slurpit.io/images/container_registry

The scraper and scanner are both python applications and portal is a web service.

2.2 Requirements

Portal needs to be able to access both the scraper and scanner on port 80. Make sure PORTAL_WAREHOUSE_URL has the correct hostname

Warehouse also needs to be able to access Portal on port 80. Make sure WAREHOUSE_CALLBACK_SCANNER_URL and WAREHOUSE_CALLBACK_SCANNER_URL have the correct hostname.

2.3 Compose

Podman compose is by default not installed, you can download it from:
https://github.com/containers/podman-compose

Podman compose is an easy way of getting the images up and running.

To get started with podman-compose, we created a separate folder in the GIT project called podman. In here are the file up and down to easily start and stop the application. The compose file is quite similar to the docker one, except that we have to map the volumes to :Z to make it work with SeLinux.

If you want to run the Slurp'it application offline in Podman, then you should read the chapter about offline installation in this manual.

Slurp'it
Beyond network discovery

3. Enviroment variables

3.1 Portal

3.1.1 PORTAL_BASE_URL

Default: http://localhost
Type: string

If you want the portal to remotely accessably, change localhost with the hostname of the machine

3.1.2 PORTAL_WAREHOUSE_URL

Default: http://slurpit-warehouse
Type: string

The portal needs to be able to communicate with the warehouse, if the warehouse runs on a different machine it will need to know the host on which it runs.
Communication goes through a rest api, so part 80 needs to be reachable

3.1.3 PORTAL_ ENVIRONMENT

Default: production
Type: string

In case you want to debug something on something thats going wrong in the portal app, you can set this variable to `development`. Generally this should stay on `production`

3.1.4 TZ

Default: Europe/Amsterdam
Type: string

Timezone for the image

3.1.5 PORTAL_SAFEKEY

Type: string

This is a value you should use and change. The safekey is used to decrypt/encrypt sensitive information.

3.1.6 DATABASE

Default the database is on the image itself, the password is used to create the database locally the first time you run it. In case you want to host the database yourself, mariadb needs to be installed somewhere and the following environment variables need to be changed:

- PORTAL_DB_HOSTNAME – default: localhost
- PORTAL_DB_NAME – default: portal
- PORTAL_DB_PORT – default: 3306
- PORTAL_DB_USERNAME – default: root
- PORTAL_DB_PASSWORD – default: qDlqHrK1VBoq!#ac

3.1.7 Backup

Backups are made daily at midnight, backups are kept for the last 7 days and 1 for the last 32 weeks by default.
These settings can be changed using the following docker compose settings:

- PORTAL_BACKUP, default true
- PORTAL_BACKUP_DAYS, default 7
- PORTAL_BACKUP_WEEKS, default 32

If you are constantly running out of disk space, you might want to consider lowering these settings.
Backups are stored in the /backup/files.
To manually create a backup you can run /backup/backup.sh.
To for example restore the backup from January first 2024 run: /backup/restore.sh 20240101. After you restore the backup you need to restart the container, because there is a chance that the database is missing changelogs, which run at the start.

3.2 Warehouse

3.2.1 WAREHOUSE_DEBUG

Default: false
Type: boolean

If set to true, it will generate extra logging information

3.2.2 WAREHOUSE_CALLBACK_SCANNER_URL

Default: http://slurpit-portal/callback/scanner
Type: string

Callback for where found devices should be processed, if the portal runs on a different host, change the slurpit-portal part with that hostname. Note that if this url does not point towards the portal it will not work.

3.2.3 WAREHOUSE_CALLBACK_SCANNER_TOKEN

Type: string

In case the callback_scanner_url has some sort of authentication bearer token

3.2.4 WAREHOUSE_CALLBACK_SCRAPER_URL

Default: http://slurpit-portal/callback/scraper
Type: string

Callback for the notification that the scraper finished running, if the portal runs on a different host, change the slurpit-portal part with that hostname. Note that if this url does not point towards the portal it will not work.

3.2.5 WAREHOUSE_CALLBACK_SCRAPER_TOKEN

Type: string

In case the callback_scraper_url has some sort of authentication bearer token

3.2.6 WAREHOUSE_MONGODB_LOCAL

Default: true
Type: boolean

When set to false, it will not run the mongodb locally, use this when you host the mongodb on a separate system

3.2.7 WAREHOUSE_MONGODB _HOST

Default: 127.0.0.1:27017
Type: string

The address where the mongodb server is hosted

3.2.8 WAREHOUSE_MONGODB _DBNAME

Default: slurpit
Type: string

The database name used by mongodb

3.2.9 WAREHOUSE_MONGODB _USERNAME

Default: slurpit
Type: string

The database username used for login by mongodb. If used with warehouse_mongodb_local=true, the database is created using this username.

3.2.10 WAREHOUSE_MONGODB _PASSWORD

Default: o1LdqfT1II0W!Wy3
Type: string

The database password used for login by mongodb. If used with warehouse_mongodb_local=true, the database is created using this password.

3.2.11 TZ

Default: Europe/Amsterdam
Type: string

Timezone for the image

3.2.12 Backup

Backups are made daily at midnight, backups are kept for the last 7 days and 1 for the last 32 weeks by default.
These settings can be changed using the following docker compose settings:

- WAREHOUSE_BACKUP, default true
- WAREHOUSE_BACKUP_DAYS, default 7
- WAREHOUSE_BACKUP_WEEKS, default 32

If you are constantly running out of disk space, you might want to consider lowering these settings.
Backups are stored in the /backup/files.
To manually create a backup you can run /backup/backup.sh.
To for example restore the backup from January first 2024 run: /backup/restore.sh 20240101. After you restore the backup you need to restart the container, because there is a chance that the database is missing changelogs, which run at the start.

3.3 Scanner

3.3.1 SCANNER_DEBUG

Default: false
Type: boolean

If set to true, it will generate extra logging information

3.3.2 SCANNER_POOLSIZE

Default: 4
Type: integer

Number of snmp calls running at the same time. This value basically controls how many threads are started. Adjust according to your networks size/machine capabilities.

3.3.3 SCANNER_TIMEOUT

Default: 10
Type: integer

Timeout in seconds for both scanning for devices and for snmp calls

3.3.4 SCANNER_WAREHOUSE_URL

Default: http://slurpit-warehouse
Type: string

The url of where the warehouse is located

3.3.5 SCANNER_RATELIMITER

Default: 0
Type: integer

Number of snmp login requests per second the scanner can make to devices within your network. With 0 there is no limit

3.3.6 TZ

Default: Europe/Amsterdam
Type: string

Timezone for the image

3.3.7 SCANNER_ARP_ENABLED

Default: true
Type: Boolean

Whether the scanner should use arp to find devices

3.3.8 SCANNER_ICMP_ENABLED

Default: true
Type: Boolean

Whether the scanner should ping to find devices

3.3.9 SCANNER_TCP_ENABLED

Default: false
Type: Boolean

Whether the scanner should scan port 22 to find devices

3.4 Scraper

3.4.1 SCRAPER_DEBUG

Default: false
Type: boolean

If set to true, it will generate extra logging information

3.4.2 SCRAPER_POOLSIZE

Default: 4
Type: integer

Number of calls running at the same time. This value basically controls how many threads are started. Adjust according to your networks size/machine capabilities.

3.4.3 SCRAPER_TIMEOUT

Default: 20
Type: integer

Timeout in seconds for running commands on the different nodes

3.4.4 SCRAPER_WAREHOUSE_URL

Default: http://slurpit-warehouse
Type: string

The url of where the warehouse is located

3.4.5 SCRAPER_COMMAND_LATENCY_ENABLED

Default: false
Type: Boolean
Docs: https://pynet.twb-tech.com/blog/netmiko-send-command-timing.html

SCRAPER_COMMAND_LATENCY is an entirely timing based solution. When enabled it uses the send_command_timing function from Netmiko. It does not look for nor care about any pattern in the output. It basically tries to intelligently guess whether the device is done outputting. This can be usefull for devices which experience a big latency.

3.4.6 SCRAPER_COMMAND_LATENCY_DELAY

Default: 2.0
Type: integer

This feature will only work when you have enabled the SCRAPER_COMMAND_LATENCY. Once you are changing the Latency Delay the Scraper will sleep longer. Here the Scraper is trying to ensure that all available data has been read. Consequently, it waits a bit longer to ensure nothing new arrives.

Once the scraper does his last read, it then checks if this result is empty or not. If there is data (i.e. it is not empty), then it returns back to the main reading loop. Basically, there is more output data to process so let's keep reading.

If you are running into cases where you are not capturing all the data you require, then you should increase this delay time. Default is 2 seconds.

3.4.7 SCRAPER_LOGIN_RETRY

Default: 1
Type: integer

Number of times ssh will try to login before giving up

3.4.8 SCRAPER_LOGIN_RETRY_DELAY

Default: 0.1
Type: float

Time in seconds between each login retry

3.4.9 SCRAPER_RATELIMITER

Default: 0
Type: integer

Number of ssh login requests per second the scraper can make to devices within your network.
With 0 there is no limit

3.4.10 SCRAPER_RATELIMIT_COMMANDS

Default: false
Type: boolean

If true, this will include commands in the SCRAPER_RATELIMITER

3.4.11 TZ

Default: Europe/Amsterdam
Type: string

Timezone for the image

Slurp'it
Beyond network discovery

4. Offline

If you want to run on machines separated from the internet, you will need to save the 4 slurpit images on a machine which does have access to the internet and then copy them to the machine which you want to run the images on. Both machines will need to have docker to save and load the images. To save images use:

Docker

- docker pull slurpit/warehouse:latest
- docker save slurpit/warehouse:latest -o slurpit-warehouse.tar
- docker pull slurpit/scanner:latest
- docker save slurpit/scanner:latest -o slurpit-scanner.tar
- docker pull slurpit/scraper:latest
- docker save slurpit/scraper:latest -o slurpit-scraper.tar
- docker pull slurpit/portal:latest
- docker save slurpit/portal:latest -o slurpit-portal.tar

Podman

- podman pull slurpit/warehouse:latest
- podman save slurpit/warehouse:latest -o slurpit- warehouse.tar
- podman pull slurpit/scanner:latest
- podman save slurpit/scanner:latest -o slurpit-scanner.tar
- podman pull slurpit/scraper:latest
- podman save slurpit/scraper:latest -o slurpit-scraper.tar
- podman pull slurpit/portal:latest
- podman save slurpit/portal:latest -o slurpit-portal.tar

To then load the images on the machine you want to run them on use:

Docker

- docker load -i slurpit-warehouse.tar
- docker load -i slurpit-scanner.tar
- docker load -i slurpit-scraper.tar
- docker load -i slurpit-portal.tar

Podman

- podman load -i slurpit-warehouse.tar
- podman load -i slurpit-scanner.tar
- podman load -i slurpit-scraper.tar
- podman load -i slurpit-portal.tar

After you've loaded the images on the machine, you can use the image as normal, meaning you can use `docker run registry.gitlab.com/slurpit.io/images/scanner:latest` to run the image.

Slurp'it
Beyond network discovery

5. Default login

Username:
admin@admin.com

Password:
12345678

6. Quickstart

Change in the file docker-compose.yml:

1. Four times your Timezone for slurpit-warehouse, slurpit-scraper, slurpit-scanner & slurpit-portal:
   - **TZ: Europe/Amsterdam**

2. The Portal url to visit the web gui:
   - **PORTAL_BASE_URL: http://localhost**

The Portal URL to access the web GUI is currently set to http://localhost. If you prefer to use HTTPS, update the URL to **https://your_domain_name** or **https://your_server_ip** and ensure you have the correct SSL certificates placed in the `certs` folder.

6.1 Windows

To start the containers run up.bat
To stop the containers run down.bat

6.2 macOS or Linux

To start the containers run up.sh
To stop the containers run down.sh

7. Backups

Since the system is based on containers, the application's data and configuration are stored within volume folders. This makes it very easy to backup and restore the application.

7.1 Create backup

Copy the folders **certs** and **db** to safe location where you store your backups.
You can do this while the platform is running, but it's recommended to turn off the application first to not get any corrupt data.

Important
When you are making a backup you have to make sure that you keep the original owner rights in the database folder. Therefor use the -a when you copy the files to a new location, e.g.:

```
cp -a /opt/slurpit/db /opt/backup/db -R
```

and when you tar the folder include the -p option, e.g.:

```
tar -czpf archive.tar.gz /opt/slurpit/db
```

7.2 Restore backup

1. Turn off the software by running down.bat or down.sh
2. Replace the backup folders **certs** and **db**.
3. Turn on the software again up.bat or up.sh

8. Update the containers

8.1 Online environment

When you start Slurpit by using up.sh or up.bat the containers will be automatically updated.
So in case you want to update the application you just have to restart it using the scripts.

8.2 Offline environment

1. Stop the application
2. Download the latest images like specified in **chapter 3**
3. Start the application

Slurp'it
Beyond network discovery

9. SSL certificate

9.1 Add a certificate

To add a certificate:

1. Place the following files in the `certs` folder:
   - `private.key`
   - `certificate.crt`
2. Change the beginning of the following variables to `https` in the `docker-compose.yml` file.

   - PORTAL_BASE_URL
   - WAREHOUSE_CALLBACK_SCANNER_URL
   - WAREHOUSE_CALLBACK_SCRAPER_URL

For example:

```
WAREHOUSE_CALLBACK_SCANNER_URL: https://slurpit-portal/callback/scanner
WAREHOUSE_CALLBACK_SCRAPER_URL: https://slurpit-portal/callback/scraper
PORTAL_BASE_URL: https://192.168.56.1
```

9.2 Replace a certificate

To replace a certificate:

1. Run the `down.sh` or `down.bat` script.
2. Replace the files in the `certs` folder.
3. Run the `up.sh` or `up.bat` script.

9.3 Create a self-signed certificate

If you don't have a certificate yet, follow these steps to create one:

1. Generate a private key:
   openssl genrsa -out private.key 2048

2. Generate a certificate signing request (CSR):
   openssl req -new -key private.key -out certificate.csr

3. Generate a self-signed certificate:
   openssl x509 -req -in certificate.csr -signkey private.key -out certificate.crt

Slurp'it

Beyond network discovery