

ads-tec IIT GmbH

IRF3000 IRF1000

Application Note - Docker



Version	Date	Editor	Changes
1.0	13.09.2022	SnSz	Initial Version
1.1	14.09.2022	SnPr	Small improvements

Table of Contents

1	Introduction	3
2	Prerequisites	4
2.1	CPU Architecture.....	4
2.2	Storage.....	4
3	Getting started	5
3.1	Docker Installation.....	5
3.2	Device Preparations	5
3.3	Docker Usage.....	7
3.3.1	Linux Workstation.....	7
3.3.2	Windows Workstation.....	8
3.3.3	Generic Usage	8
4	Advanced Topics.....	10
4.1	Security	10
4.1.1	Rootless Docker	10
4.1.2	Separate Namespace.....	10
4.2	Networking.....	10
4.3	Docker Compose.....	12
4.3.1	Installation	12
4.3.2	Usage of Docker Compose	13
4.4	Private Registries	14
5	Example Applications.....	16
5.1	NGINX	16
5.1.1	Prerequisites	16
5.1.2	Start NGINX	16
5.2	Portainer.....	17
5.2.1	Prerequisites	17
5.2.2	Architecture of Portainer.....	17
5.2.3	Installation of the Portainer Server	17
5.2.4	Installation of Portainer Agents.....	19
6	Troubleshooting	24
6.1	Date-Time Error.....	24
6.1.1	Error Description	24
6.1.2	Solution	24
6.2	Certificate signed by unknown authority	25
6.2.1	Error Description	25
6.2.2	Solution	25

1 Introduction

Docker is a software platform that provides the ability to package and run applications in an isolated environment called a container. A container is a lightweight piece of software that contains everything to run an application. This means the software provided by a container is independent of the software running on the IRF300/IRF1000 operating system. Any piece of software can thus be made to be run on the IRF3000/IRF1000. In addition, encapsulation makes it possible to run multiple container instances at once.

With the current firmware version of the IRF1000 and IRF3000 ads-tec is providing a Docker 20.10.17 environment. The docker environment can be accessed and controlled using default Docker Tools, for example the Docker Command Line Interface. This enables to package and run own applications on the devices.

Examples of these applications can be:

- A custom database storage that saves data and provides it through a connected custom webserver.
- A custom database storage that saves data and sends it to another provided endpoint.
- Protocol converters that convert data to a different protocol.

2 Prerequisites

There are a few things that need to be clarified before the use of the Docker software platform on the IRF3000 and IRF1000 so that there is no problem in a later point of the document.

2.1 CPU Architecture

The CPU Architecture describes a hurdle that must be overcome. Only Docker images that are built for the related processor architecture can be run.

Information regarding the processor architectures of the devices:

- IRF3000: ARM64 ARMv8
- IRF1000: ARMHF ARMv7 (Compatible to Raspberry Pi 1-3)

For running Docker containers, it must therefore be ensured that images that come from a registry like the official Docker Hub are available in the respective processor architecture.

2.2 Storage

Embedded devices are usually limited in the amount of RAM and flash storage. This is also the case with our devices. Before a Docker environment is set up, attention should be paid to the storage consumption of the environment. Otherwise, data loss could occur.

Therefore information regarding the storage of the devices is provided:

Product	RAM	Flash Storage (for Docker)
IRF3000	4 GB	6.6 GB
IRF1000	512 MB	3.1 GB

Storage intensive applications should be handled carefully otherwise a permanent destruction of the integrated eMMC chips due to too many write cycles can happen.

3 Getting started

To get started you need a development workstation like a Windows or Linux based PC. This chapter will guide you through the complete installation process using the Docker Command Line Interface as an example.

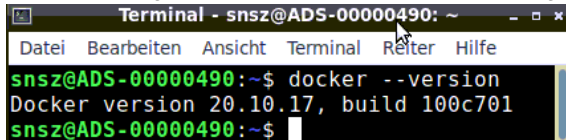
3.1 Docker Installation

To run Docker, you need an up-to-date Docker installation on your system.

To download and install Docker you can use the following link:

<https://docs.docker.com/engine/install/binaries>

With a working Docker CLI installation, the following command should return the Docker Version:

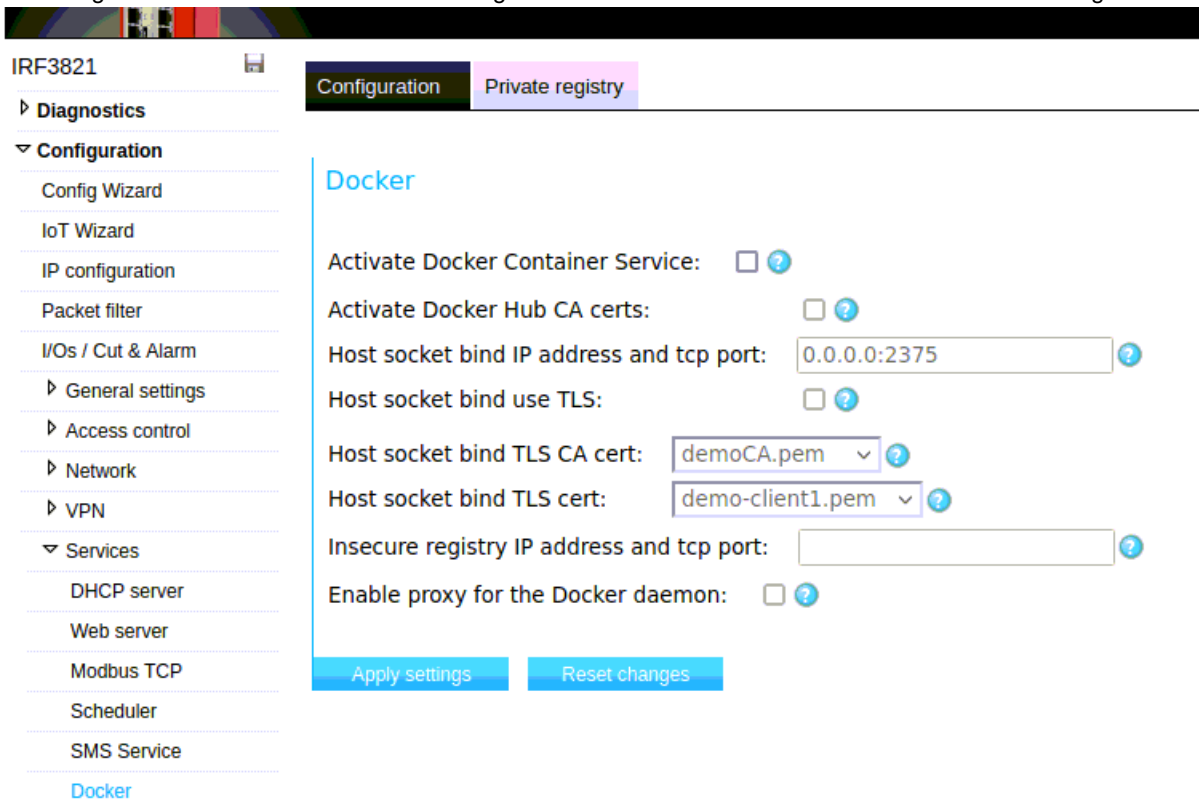


```
Terminal - snsz@ADS-00000490: ~
Datei Bearbeiten Ansicht Terminal Fenster Hilfe
snasz@ADS-00000490:~$ docker --version
Docker version 20.10.17, build 100c701
snasz@ADS-00000490:~$
```

3.2 Device Preparations

This chapter provides the knowledge how to setup Docker on the IRF3000/IRF1000 through the web interface.

The configuration of Docker is located at “Configuration > Services > Docker” as can be seen in the image below.

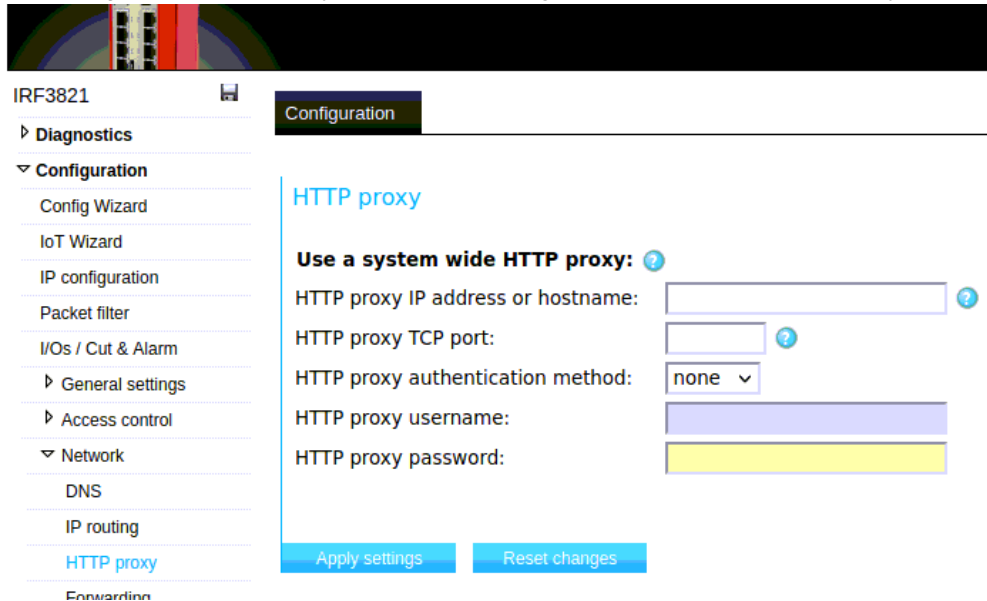


The various fields are briefly explained below.

- The Docker-Service can be started and stopped with the “Activate Docker Container Service” Checkbox. This checkbox is Docker’s global on/off switch on the IRF3000/IRF1000.
- Additional CA certificates are required for communication with the public Docker hub Registry. There is an additional checkbox “Activate Docker Hub CA certs” to provide these certificates. In the default state, no registry is trusted for security reasons.
- The IP address on which the Docker socket is created can be selected using the “Host socket bind IP address and tcp port” field. This allows restricting access to only one specified ethernet port.
- In addition, the connection can be encrypted through TLS. The next three fields are available for this.
- **The use of TLS in the production system is highly recommended!**

- The field “Insecure Registry IP address and TCP port” is available to test a registry that does not support a TLS encrypted connection. **It is highly recommended not to use this on a production system!**
- If a proxy is required for the Docker Daemon (Docker Service on the device), this can be activated using the checkbox “Enable proxy for the Docker daemon”. If the checkbox is enabled, the system wide proxy is used, which can be defined as follows.

The Configuration for the Proxy is located at “Configuration > Network > HTTP Proxy”



IRF3821

Configuration

Diagnosics

Configuration

Config Wizard

IoT Wizard

IP configuration

Packet filter

I/Os / Cut & Alarm

General settings

Access control

Network

DNS

IP routing

HTTP proxy

Forwarding

HTTP proxy

Use a system wide HTTP proxy: ?

HTTP proxy IP address or hostname: ?

HTTP proxy TCP port: ?

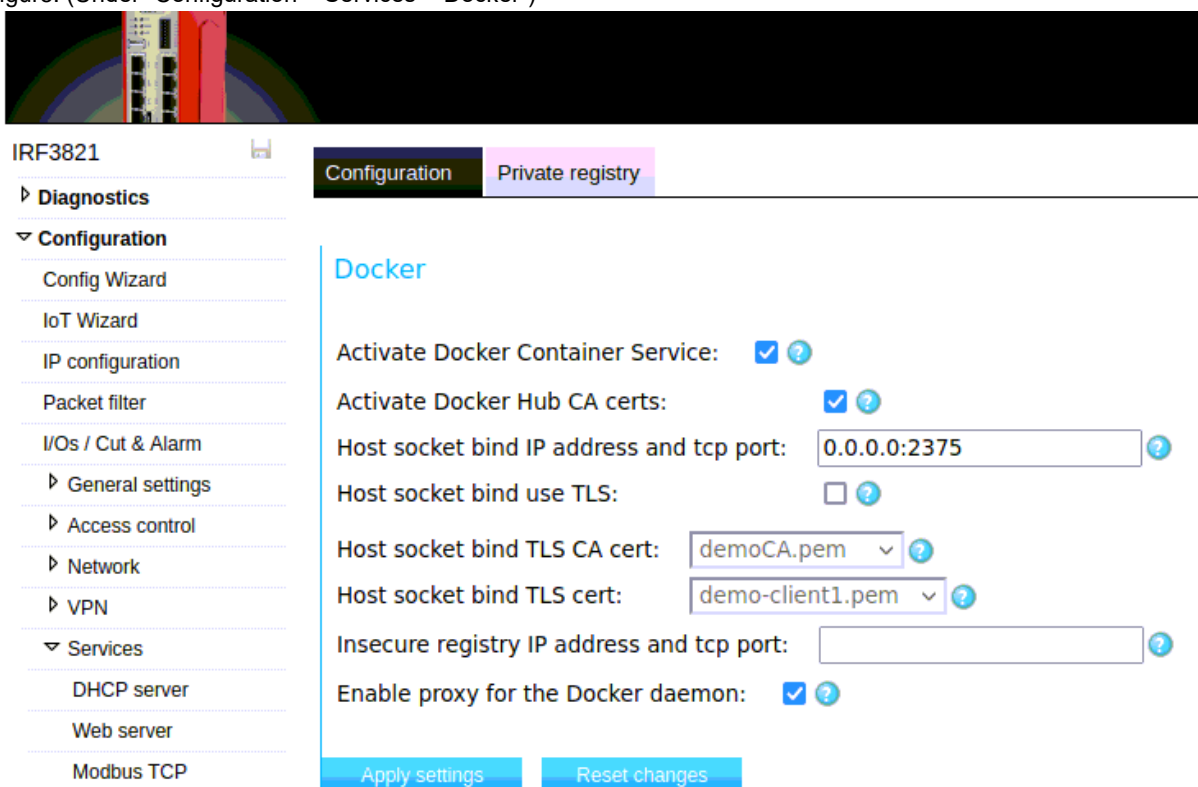
HTTP proxy authentication method: none

HTTP proxy username:

HTTP proxy password:

Apply settings Reset changes

To enable docker and pull an image from Dockerhub as a test, the configuration must look like the following figure. (Under “Configuration > Services > Docker”)



IRF3821

Configuration Private registry

Diagnosics

Configuration

Config Wizard

IoT Wizard

IP configuration

Packet filter

I/Os / Cut & Alarm

General settings

Access control

Network

VPN

Services

DHCP server

Web server

Modbus TCP

Docker

Activate Docker Container Service: ☒ ?

Activate Docker Hub CA certs: ☒ ?

Host socket bind IP address and tcp port: 0.0.0.0:2375 ?

Host socket bind use TLS: ☐ ?

Host socket bind TLS CA cert: demoCA.pem ?

Host socket bind TLS cert: demo-client1.pem ?

Insecure registry IP address and tcp port: ?

Enable proxy for the Docker daemon: ☒ ?

Apply settings Reset changes

Since the device is not directly connected to the Internet, the proxy (a system with internet access) is defined and activated with this configuration.

For further explanations, reference is made to what was previously described.

3.3 Docker Usage

This chapter describes how to use the Docker Command Line Interface on the IRF3000/IRF1000.

After successfully starting the Docker service, the IP address of the device needs to be found out. This can be done in different ways, one of these ways is using the web interface.

To find out the current IP address of the device, the following page under “Configuration > IP configuration” can be used.

The screenshot shows the web interface of the IRF3821 device. The left sidebar contains a navigation menu with the following items: Diagnostics, Configuration (expanded), Config Wizard, IoT Wizard, IP configuration (selected), Packet filter, I/Os / Cut & Alarm, General settings, Access control, Network, VPN, Services, IoT, System, and Information. The main content area is titled 'IP configuration' and is divided into two sections: WAN and LAN. The WAN section has 'Operational mode' set to 'IP router', 'IP assignment' set to 'DHCP', and checkboxes for 'DNS via DHCP' and 'Gateway via DHCP'. The LAN section has 'IP assignment' set to 'static', 'IP address' set to '192.168.0.254', and 'Subnet mask' set to '255.255.255.0'. The bottom of the sidebar shows 'User: admin'.

In the test case, the device is connected to the development workstation via LAN. The IP address in this case is “192.168.0.254”.

3.3.1 Linux Workstation

To use the Docker service on the IRF3000/IRF1000 an environment variable must be set. To make setting the environment variable easier, the Docker Command Line Interface should be useable by a normal user. This can be configured using the following link: <https://docs.docker.com/engine/install/linux-postinstall/#manage-docker-as-a-non-root-user>

Then the environment variable DOCKER_HOST can be set to the IP address of the IRF3000/IRF1000. Setting the environment variable has the effect that all Docker Command Line Interface commands are redirected to the specified DOCKER_HOST.

```
Terminal - snsz@ADS-00000490: ~
Datei Bearbeiten Ansicht Terminal Reiter Hilfe
snasz@ADS-00000490:~$ export DOCKER_HOST=192.168.0.254:2375
snasz@ADS-00000490:~$
```

Please continue the instructions at chapter 3.3.3 Generic Usage.

3.3.2 Windows Workstation

Todo

3.3.3 Generic Usage

In this Chapter the common usage of the Docker Command Line Interface gets described.

The following command can be used to validate whether the use of the IRF3000/IRF1000 as a Docker host worked.

```
snsz@ADS-00000490:~$ export DOCKER_HOST=192.168.0.254:2375
snsz@ADS-00000490:~$ docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Docker Buildx (Docker Inc., v0.8.2-docker)
  compose: Docker Compose (Docker Inc., v2.6.1)
  extension: Manages Docker extensions (Docker Inc., v0.2.7)
  sbom: View the packaged-based Software Bill of Materials (SBOM) for an image (Anchore Inc., 0.6.0)
  scan: Docker Scan (Docker Inc., v0.17.0)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 20.10.17
 Storage Driver: btrfs
   Build version: Btrfs v5.7
   Library Version: 102
 Logging Driver: json-file
 Cgroup Driver: none
 Cgroup Version: 1
 Plugins:
  volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runtime.v1.linux runc io.containerd.runc.v2
 Default Runtime: runc
 Init Binary: docker-init
 containerd version:
 runc version:
 init version: de40ad0
 Security Options:
  rootless
 Kernel Version: 5.4.117
 Operating System: OpenWRT 1.1.0
 OSType: linux
 Architecture: aarch64
 CPUs: 4
 Total Memory: 3.731GiB
 Name: IRF3821-AX00000000
 ID: E6SY:HMMW:OMC3:7I25:EAXU:GU2I:CJ46:RKNS:I2BV:SZMH:7MO4:QLCX
 Docker Root Dir: /usr/local/docker/.local/share/docker
 Debug Mode: false
 HTTP Proxy: 192.168.0.40:8080
 HTTPS Proxy: 192.168.0.40:8080
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false
```

WARNING: API is accessible on http://0.0.0.0:2375 without encryption.
Access to the remote API is equivalent to root access on the host. Refer

to the 'Docker daemon attack surface' section in the documentation for more information: <https://docs.docker.com/go/attack-surface/>
 WARNING: Running in rootless-mode without cgroups. To enable cgroups in rootless-mode, you need to boot the system in cgroup v2 mode.

On the one hand, this can be used to find out whether the forwarding from the Docker Command Line Interface to the IRF3000/IRF1000 worked and, on the other hand, information from the Docker Service on the IRF3000/IRF1000 is displayed.

With the following command, a so-called "Alpine" container can be pulled from Dockerhub on the IRF3000/IRF1000. The Alpine Docker Image is based on Alpine Linux and contains only 5MB in size.

```
snsz@ADS-00000490:~$ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
9b18e9b68314: Pull complete
Digest: sha256:bc41182d7ef5ffc53a40b044e725193bc10142a1243f395ee852a8d9730fc2ad
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

With the command "docker images" can be validated if the image was pulled correctly.

```
snsz@ADS-00000490:~$ docker images
REPOSITORY    TAG       IMAGE ID      CREATED        SIZE
alpine        latest    a6215f271958 4 weeks ago    5.29MB
```

The following command starts the "Alpine" container in interactive mode, which contains a volume mount and a port definition.

```
snsz@ADS-00000490:~$ docker run -it -v /:/app -p 8080:8080 alpine /bin/sh
/ # ls
app  dev  home  media  opt  root  sbin  sys  usr
bin  etc  lib   mnt    proc  run   srv   tmp  var
```

More information can be found in the official docker documents: <https://docs.docker.com/>

4 Advanced Topics

4.1 Security

The Docker service allows third-party software to run on the IRF3000/IRF1000. For this reason, the protection of the base system is an important point. To ensure the IRF3000/IRF1000 is safe, two protection mechanisms are implemented on the device. These two mechanisms are explained below.

4.1.1 Rootless Docker

To ensure the security of the IRF3000/IRF1000 despite the Docker service, it is integrated in rootless mode. The Docker service therefore has no root rights within the operating system. To enable the best possible use of the Docker service, the IRF3000/IRF1000 provides functions such as the network for Docker.

Please also be aware of the limitations that Docker Rootless brings with it. You can read about them at the following URL: <https://docs.docker.com/engine/security/rootless/#known-limitations>

More information about the rootless mode can be found in the official Docker documentation: <https://docs.docker.com/engine/security/rootless>

4.1.2 Sandboxed Filesystem

To provide another layer of security, the Docker service is also located in a separate root filesystem. This enables the entire Docker environment to be run encapsulated from the IRF3000/IRF1000 System.

4.2 Networking

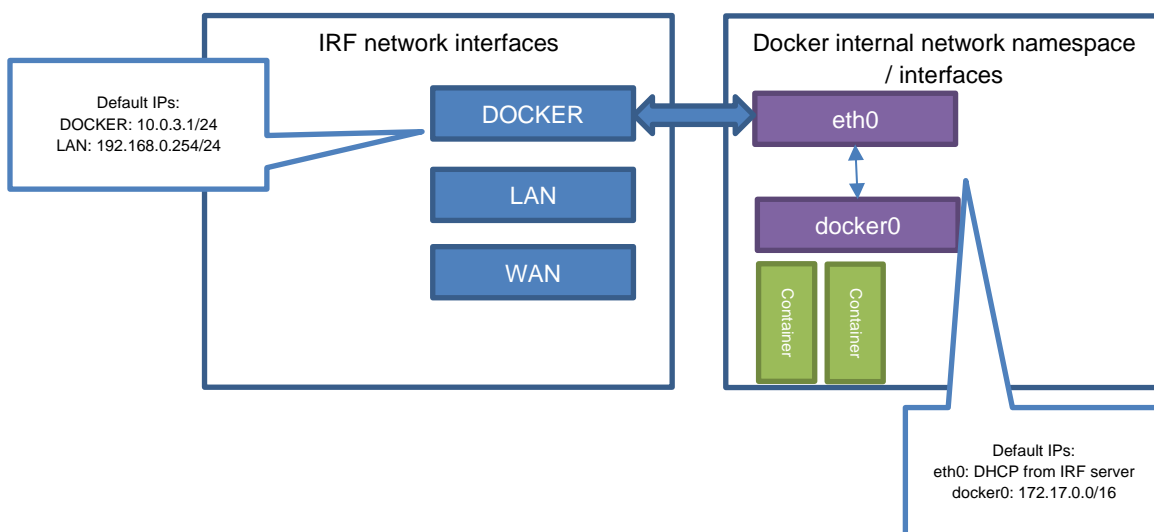
4.2.1 Overview

Docker rootless on the IRFs is running the docker host network concept by default.

(<https://docs.docker.com/network/host/>). A virtual network interface (DOCKER) is representing the connection to the Docker service within the IRF operating system.

The Docker internal network namespace contains one primary network interface (eth0) which will get its IP from the main DHCP server of the IRF like an external network component. By default the DOCKER interface on the IRF is on 10.0.3.1/24 and eth0 within the docker namespace will get the first IP in the configured DHCP range: 10.0.3.2. Docker is using an internal virtual ethernet bridge (docker0) with an internal DHCP server which will supply IPs in the network 172.17.0.0/16 to all containers. The network interfaces of the containers get bridged onto this docker0 bridge interface. The connection between eth0 and docker0 within the namespace is routed.

All network configuration of the docker network namespace is done by the docker CLI.



4.2.2 Network configuration

The operating system of the IRF3000/IRF1000 starts an internal bridge when the Docker service is enabled. The IP address of the internal bridge is set to “10.0.3.1/24” by default. This default value can be changed in the IP configuration of the device under “Configuration > IP configuration”:

IP address:	<input type="text" value="192.168.0.254"/>
Subnet mask:	<input type="text" value="255.255.255.0"/>
NAT (Masquerading):	<input type="checkbox"/> ?
WWAN:	
Dialmode:	<input type="text" value="disabled"/> ?
DOCKER:	
IP address:	<input type="text" value="10.0.3.1"/>
Subnet mask:	<input type="text" value="255.255.255.0"/>
NAT (Masquerading):	<input type="checkbox"/> ?

There is also a DHCP server that distributes the IP addresses for the Docker containers. This DHCP server is started by default, when the Docker service gets enabled. This DHCP server can be further configured on the web interface under “Configuration > Services > DHCP server”.

The configuration options can be seen in the figure below.

IRF3821

Configuration State

Diagnosics

Configuration

Config Wizard

IoT Wizard

IP configuration

Packet filter

I/Os / Cut & Alarm

General settings

Access control

Network

VPN

Services

DHCP server

Web server

Modbus TCP

Scheduler

SMS Service

Docker

GPS service

IoT

System

Information

DHCP server

Activate DHCP server: ☒ ?

On following interfaces: ☐ LAN ☐ WAN ☒ DOCKER

Interface: LAN

Starting IP address:

Ending IP address:

DHCP lease time: (seconds)

Interface: WAN

Starting IP address:

Ending IP address:

DHCP lease time: (seconds)

Interface: DOCKER

Starting IP address:

Ending IP address:

DHCP lease time: (seconds)

Docker and the associated containers automatically get an IP address from the DHCP Servers when they are started. (The configuration shown in the picture is created by starting the Docker service in the default configuration.). Please note that there are two DHCP servers running in this scenario. The first DHCP server is

the one shown here running on the OS of the IRF. The second one is running inside the docker network namespace and can be configured by the docker tools only.

4.3 Docker Compose

Docker Compose is a tool that can be used to configure multiple containers at once. Within a YAML configuration file different Docker containers can be defined. It is possible here to connect different containers using internal Docker networking. Thus, the configuration with the help of Docker Compose is perfectly suited for distributed systems, such as databases with linked web servers.

4.3.1 Installation

For the installation of Docker Compose, refer to the following Docker Docs websites:

- Windows: <https://docs.docker.com/compose/install>
- Linux: <https://docs.docker.com/compose/install/linux>

4.3.2 Usage of Docker Compose

To illustrate the use of Docker Compose, a YAML file from the ads-tec internal test system is shown.

```
version: "3.9"
services:
  alpine_ssh1:
    image: 10.20.131.150:5000/alpine-with-sshd-irf3
    ports:
      - 2222:22
    expose:
      - 22
    networks:
      - alpine-network
    volumes:
      - /tmp:/tmp
  alpine_ssh2:
    image: 10.20.131.150:5000/alpine-with-sshd-irf3
    ports:
      - 2223:22
    expose:
      - 22
    networks:
      - alpine-network
    volumes:
      - /tmp:/tmp
networks:
  alpine-network:
    name: alpine-network
    driver: bridge
```

The Image alpine-with-sshd image is an Alpine Linux container which contains an SSH server that runs when the container starts.

Within the Docker Compose files, you can define the properties that will be passed to the container at startup.

There are configurations for the port, the internal network “alpine-network” and volumes that are shared with the underlying host system (/tmp).

To start up the Docker Compose setup, the following command is sufficient:

```
snsz@ADS-00000490:~$ docker compose -f docker-compose-irf3.yml up
[+] Running 11/11
alpine_ssh2 Pulled
alpine_ssh1 Pulled
  $HASH_SUMS Pull complete
  $HASH_SUMS Pull complete
...
[+] Running 3/3
Network alpine-network          Created
Container docker_files-alpine_ssh1-1 Created
Container docker_files-alpine_ssh2-1 Created
Attaching to docker_files-alpine_ssh1-1, docker_files-alpine_ssh2-1
docker_files-alpine_ssh1-1 | ssh-keygen: generating new host keys: RSA DSA ECDSA ED25519
docker_files-alpine_ssh1-1 | Server listening on 0.0.0.0 port 22.
docker_files-alpine_ssh1-1 | Server listening on :: port 22.
docker_files-alpine_ssh2-1 | ssh-keygen: generating new host keys: RSA DSA ECDSA ED25519
docker_files-alpine_ssh2-1 | Server listening on 0.0.0.0 port 22.
docker_files-alpine_ssh2-1 | Server listening on :: port 22.
```

The following commands can be used to connect to the respective SSH servers in the Docker containers.

```
snsz@ADS-00000490:~$ ssh -p 2222 fatt@192.168.0.254
fatt@192.168.0.254's password:
welcome to Alpine!
```

The Alpine wiki contains a large amount of how-to guides and general information about administrating Alpine systems.
See <<http://wiki.alpinelinux.org/>>.

You can setup the system with the command: setup-alpine

You may change this message by editing /etc/motd.

```
0aea0eb4e827:/$ touch /tmp/test
0aea0eb4e827:/$ ls /tmp
test
```

```
snsz@ADS-00000490:~$ ssh -p 2223 fatt@192.168.0.254
fatt@192.168.0.254's password:
Welcome to Alpine!
```

The Alpine wiki contains a large amount of how-to guides and general information about administrating Alpine systems.
See <<http://wiki.alpinelinux.org/>>.

You can setup the system with the command: setup-alpine

You may change this message by editing /etc/motd.

```
d56442bec757:~$ ls /tmp/
test
```

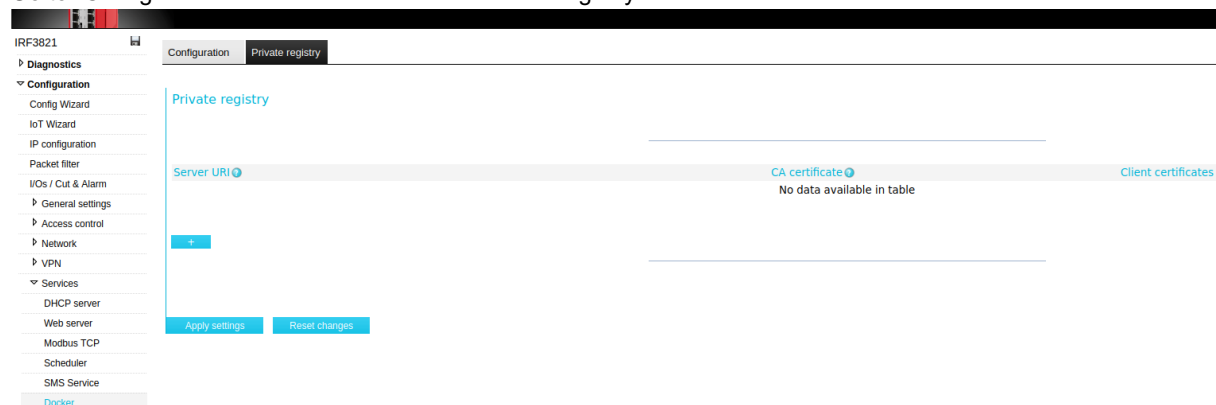
Here the connection process to the SSH Servers in the Docker Containers can be seen. In addition, the “volume mount” is illustrated using an example. Here the first container creates a file in the /tmp folder. The second container can also see the file because of the volume mount.

Of course, Docker Compose can be used to implement much more at this point! This is only an example and serves for illustrational purposes.

4.4 Private Registries

A private registry can be seen as a privately hosted repository for docker images. As a counterexample Dockerhub is a publicly hosted registry.

Go to “Configuration > Services > Docker > Private registry”



Various things can be configured here.

- Server URI: The Server URI for the Device the Registry is running. (IP-Format or DNS-Name with optional port)
- CA certificate (for explanation refer to IRF2000_IF10000_Application_Note_VPN_Zertifikate_mit_XCA_DE)
- Client certificates (for explanation refer to IRF2000_IF10000_Application_Note_VPN_Zertifikate_mit_XCA_DE)

The certificates can be uploaded to the IRF3000/IRF1000 in the Menu “Configuration > General settings > Certificates”

The screenshot shows the configuration interface for an IRF3821 device. The left sidebar contains a navigation menu with the following items: Diagnostics, Configuration (expanded), Config Wizard, IoT Wizard, IP configuration, Packet filter, I/Os / Cut & Alarm, General settings (expanded), System data, Date & time, User interface, Certificates (selected), SCEP, Access control, Network, VPN, and Services. The main content area is titled 'Configuration' and displays the 'Certificates' section. Under 'Trusted root certification authorities:', there are two entries: 'DEMO-CN/emailAddress=democa@ads-tec.de (demoCA.pem)' and 'Big-LinX Signing Certificate (idascepca.pem)'. Below this, the 'Device certificates' section shows two entries: 'DEMO-CN1/emailAddress=demo1@ads-tec.de (demo-client1.pem)' and 'DEMO-CN2/emailAddress=demo2@ads-tec.de (demo-client2.pem)'.

IRF3821

Configuration

▸ Diagnostics

▾ Configuration

- Config Wizard
- IoT Wizard
- IP configuration
- Packet filter
- I/Os / Cut & Alarm
- ▾ General settings
 - System data
 - Date & time
 - User interface
 - Certificates**
 - SCEP
- Access control
- Network
- VPN
- Services

Certificates

Trusted root certification authorities:

Certificate

- DEMO-CN/emailAddress=democa@ads-tec.de (demoCA.pem)
- Big-LinX Signing Certificate (idascepca.pem)

Device certificates

Certificate

- DEMO-CN1/emailAddress=demo1@ads-tec.de (demo-client1.pem)
- DEMO-CN2/emailAddress=demo2@ads-tec.de (demo-client2.pem)

For further explanation for creating and providing those certificates refer to the Application Note:
IRF2000_IF10000_Application_Note_VPN_Zertifikate_mit_XCA_DE

5 Example Applications

Various examples and their installation are shown within this chapter.

5.1 NGINX

NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server.

The Installation process of NGINX is shown within this chapter.

5.1.1 Prerequisites

- Docker should be configured right: for correct configuration refer to Chapter 3.
 - Docker Hub CA Certs: required for Docker Hub communication
 - Enable proxy for the Docker Daemon: required if the IRF3000/IRF1000 is not directly connected to the Internet.
- Docker should be up and running (refer to Chapter 3)
- Docker should be reachable by the Command Line Interface (refer to Chapter 3)

5.1.2 Start NGINX

With the following command the NGINX web server can be started (in the background (-d)). The internal Docker Container Port 80 is routed to the external port 8080 in this case (-p 8080:80).

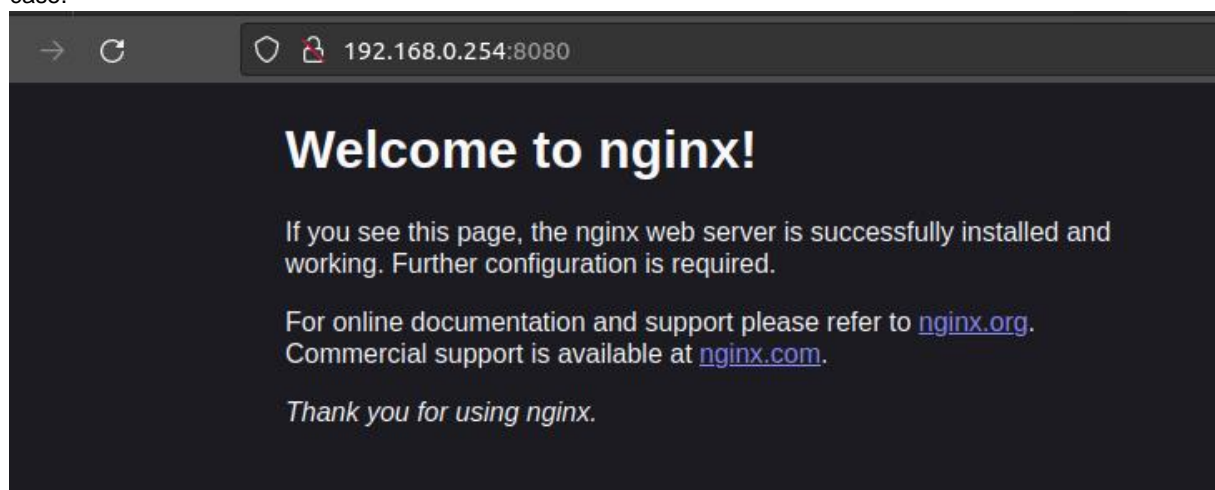
```
snsz@ADS-00000490:~$ docker run --rm -d -p 8080:80 --name web nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
5b1423465504: Pull complete
1cdde8b981f2: Pull complete
6c0b05f215c0: Pull complete
004f1937a10a: Pull complete
fd61d71c75fe: Pull complete
717bf61a04cf: Pull complete
Digest: sha256:b95a99feebf7797479e0c5eb5ec0bdfa5d9f504bc94da550c2f58e839ea6914f
Status: Downloaded newer image for nginx:latest
ae295948560741d9f09b55f8faad38d4dc41e6c1e7badfd609f960f9672f8475
```

After the successful “docker pull” command the NGINX web server gets started.

To see if the web server is up and running the following command can be sent.

```
snsz@ADS-00000490:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
NAMEs
ae2959485607   nginx    "/docker-entrypoint...." 2 minutes ago  Up 2 minutes  0.0.0.0:8080->80/tcp, :::8080->80/tcp    web
```

Now it can be tested whether the web server is reachable. The IP address of the IRF3000 is 192.168.0.254 in this case.



Please note that it is not possible to run a service on a TCP port which is already in use by the IRFs main OS, like TCP port 80.

5.2 Portainer

Portainer is a lightweight Web interface management platform for Docker/Swarm, Kubernetes and Nomad. The goal of the application is to easily manage different Docker environments. We recommend Portainer to run on the IRF as primary Web frontend for Docker administration.

5.2.1 Prerequisites

- Docker should be configured right: for correct configuration refer to Chapter 3.
 - Docker Hub CA Certs: required for Docker Hub communication
 - Enable proxy for the Docker Daemon: required if the IRF3000/IRF1000 is not directly connected to the Internet.
- Docker should be up and running (refer to Chapter 3)
- Docker should be reachable by the Command Line Interface (refer to Chapter 3)

5.2.2 Architecture of Portainer

The fundamental architecture of Portainer consists of Portainer server and Portainer agents (or edge agents). The Portainer server provides a web interface for managing the Portainer agents. The Portainer agent communicates with the Portainer server. The Portainer server can manage the resources of the Portainer agents (or edge agents).

For more information, please refer to <https://docs.portainer.io/start/architecture>

5.2.3 Installation of the Portainer Server

Export DOCKER_HOST environment variable:

```
snsz@ADS-00000490:~$ export DOCKER_HOST=192.168.0.254:2375
```

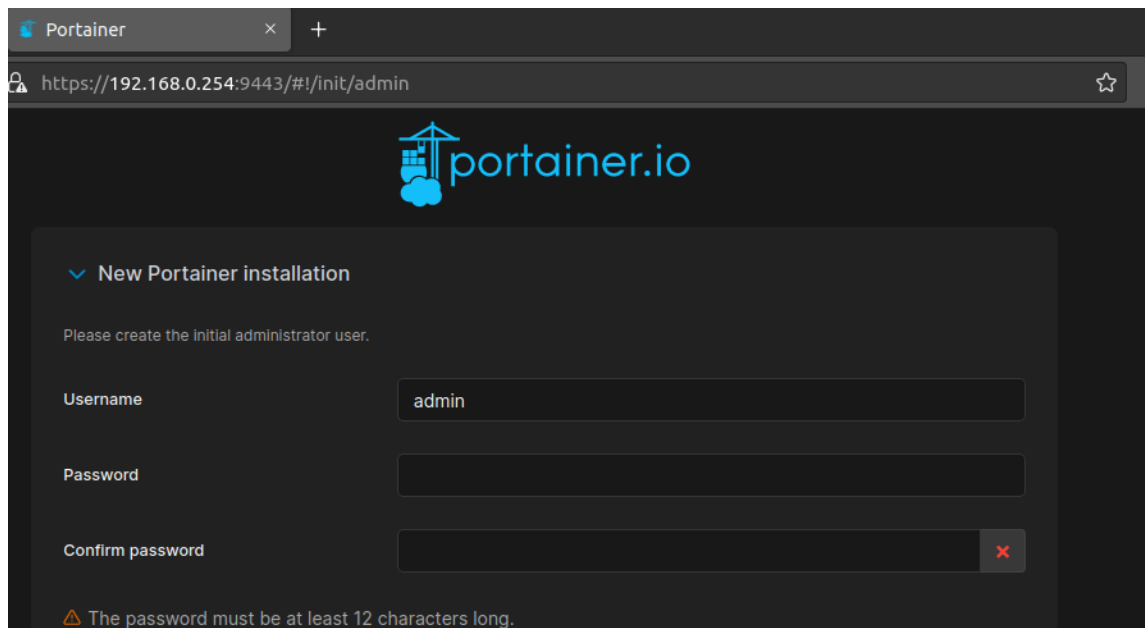
Create volume for Portainer Server database creation:

```
snsz@ADS-00000490:~$ docker volume create portainer_data
```

Download and install the Portainer Server:

```
snsz@ADS-00000490:~$ docker run -d -p 8080:8000 -p 9443:9443 --name portainer --restart=always
-v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-
ce:latest
Unable to find image 'portainer/portainer-ce:latest' locally
latest: Pulling from portainer/portainer-ce
772227786281: Pull complete
96fd13befc87: Pull complete
c4ae3071bd43: Pull complete
09555252dba0: Pull complete
Digest: sha256:444ade51d69d7fca889c7aa14525c459dba313a0e7ca79aee985e6c0749427de/var/run/docker.sock:/v
ar/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest
```

After successful installation of the Portainer Server it can be accessed through the Web Browser. The IRF3000 in this case has the IP address 192.168.0.254. The Portainer is accessible through https:



Portainer

https://192.168.0.254:9443/#/init/admin

portainer.io

▼ New Portainer installation

Please create the initial administrator user.

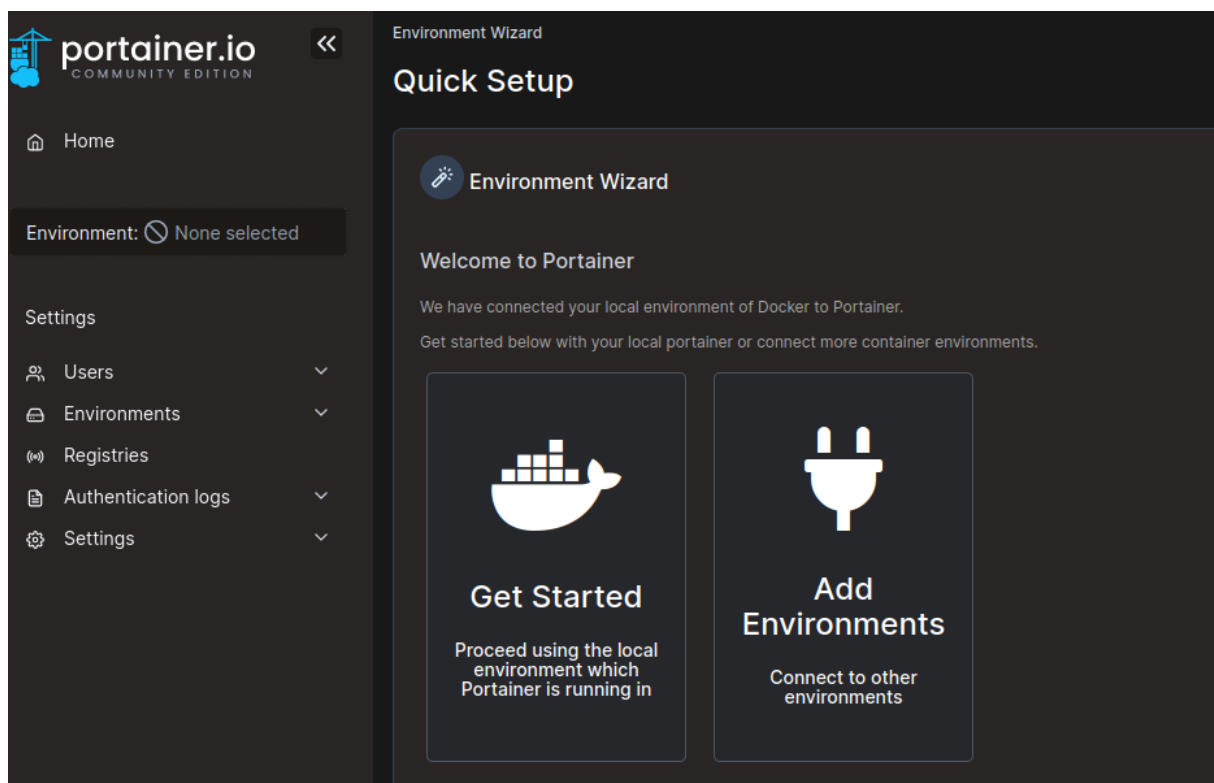
Username: admin

Password:

Confirm password:

⚠ The password must be at least 12 characters long.

After the creation of the “admin” user you will be referred to the Portainer Environment Wizard.



portainer.io
COMMUNITY EDITION

Home

Environment: None selected

Settings

- Users
- Environments
- Registries
- Authentication logs
- Settings

Environment Wizard


Quick Setup

Environment Wizard

Welcome to Portainer


We have connected your local environment of Docker to Portainer.

Get started below with your local portainer or connect more container environments.



Get Started

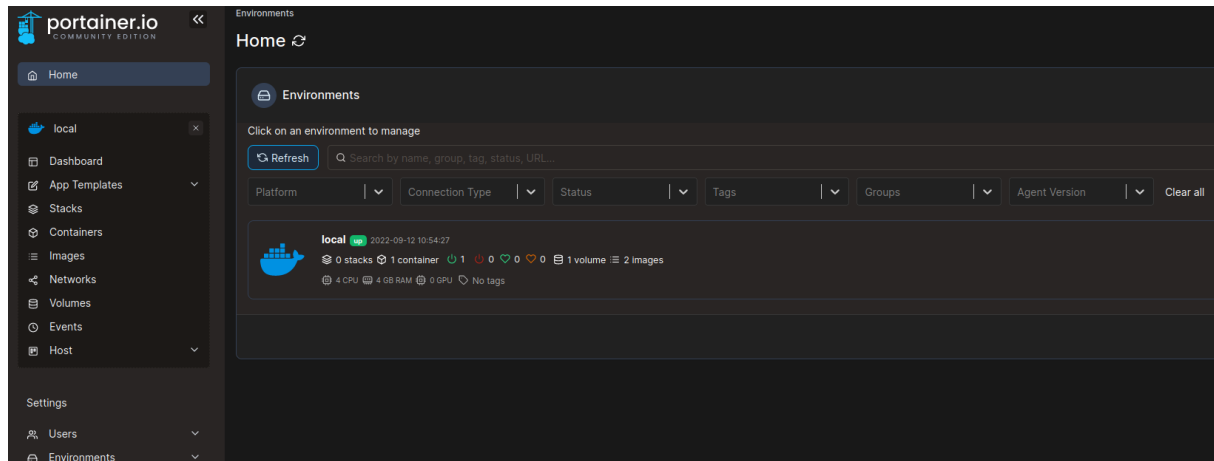
Proceed using the local environment which Portainer is running in



Add Environments

Connect to other environments

If “Get Started” is pressed, it will take you to the Portainer's home page:



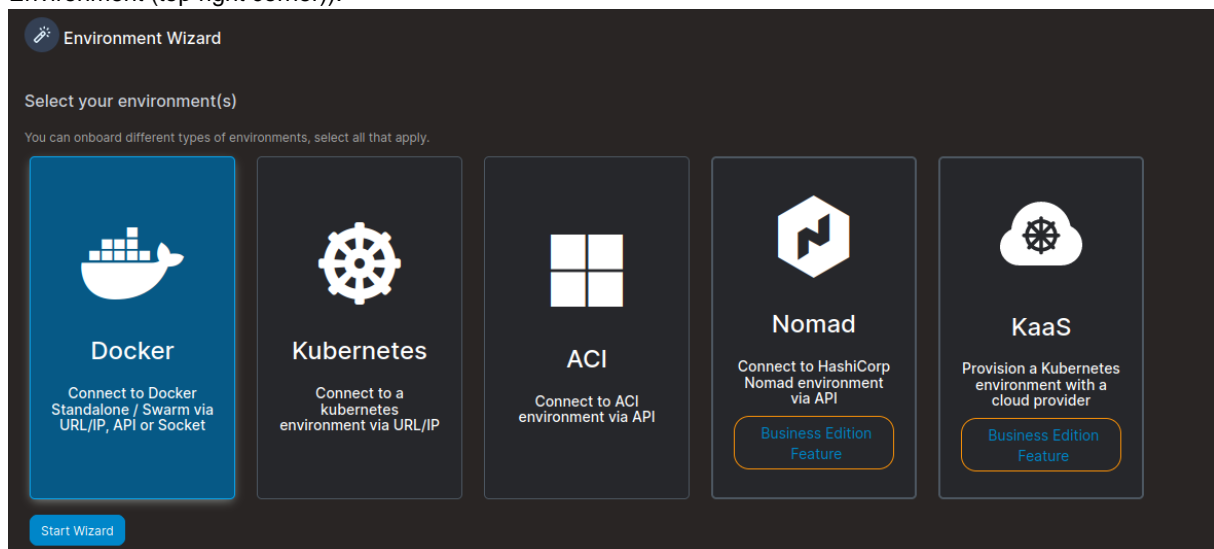
The local environment can be seen on the homepage. Now the local environment can be conveniently configured via the web interface. On the left side there are options such as „Containers“, „Images“, „Networks“,... provided by the Docker Service.

5.2.4 Installation of Portainer Agents.

This chapter covers the installation of Portainer Agents. Since there are differences between Rootless Docker and Rootful Docker when installing Portainer Agents, both ways are shown as examples.

For example, once a rootful Docker is used on a development workstation. On the other hand, an IRF3000/IRF1000 is used to install a Portainer Agent.


If „Add Environments“ is pressed, different Environments can be selected (Also through Environment > Add Environment (top right corner)):




5.2.4.1 Development Workstation as a Portainer Agent (Rootful Docker)

In this test case the development workstation is configured with a Docker environment and added as a Portainer agent. Therefore, Docker must be selected in this case.


Connect to your Docker environment




Agent



API



Socket



Edge Agent

Linux

Windows

CLI script for installing agent on your environment with Docker Swarm:

```


docker network create \
--driver overlay \
portainer_agent_network


docker service create \
--name portainer_agent \
--network portainer_agent_network \
-p 9001:9001/tcp \
--mode global \
--constraint 'node.platform.os == linux' \
--mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
--mount type=bind,src=/var/lib/docker/volumes,dst=/var/lib/docker/volumes \
portainer/agent:2.15.0


```

Copy command

Name*

 Name is required

Environment address* 

 This field is required

The commands shown in the picture must be executed on the development workstation (rootful docker). If this is successful the fields „Name“ and „Environment address“ can be entered.

In the example case it looks like this (The development workstation has the IP address “192.168.0.40“):

Linux

Windows

CLI script for installing agent on your environment with Docker Swarm:

```


docker network create \
--driver overlay \
portainer_agent_network

docker service create \
--name portainer_agent \
--network portainer_agent_network \
-p 9001:9001/tcp \
--mode global \
--constraint 'node.platform.os == linux' \
--mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
--mount type=bind,src=/var/lib/docker/volumes,dst=/var/lib/docker/volumes \
portainer/agent:2.15.0

```

Copy command

Name*

Environment address* 

More settings

Metadata

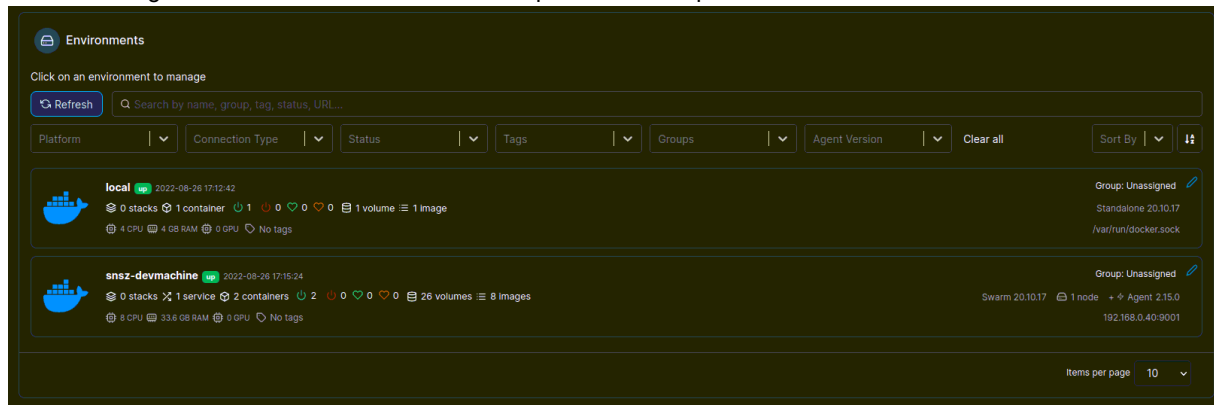
Group

Tags

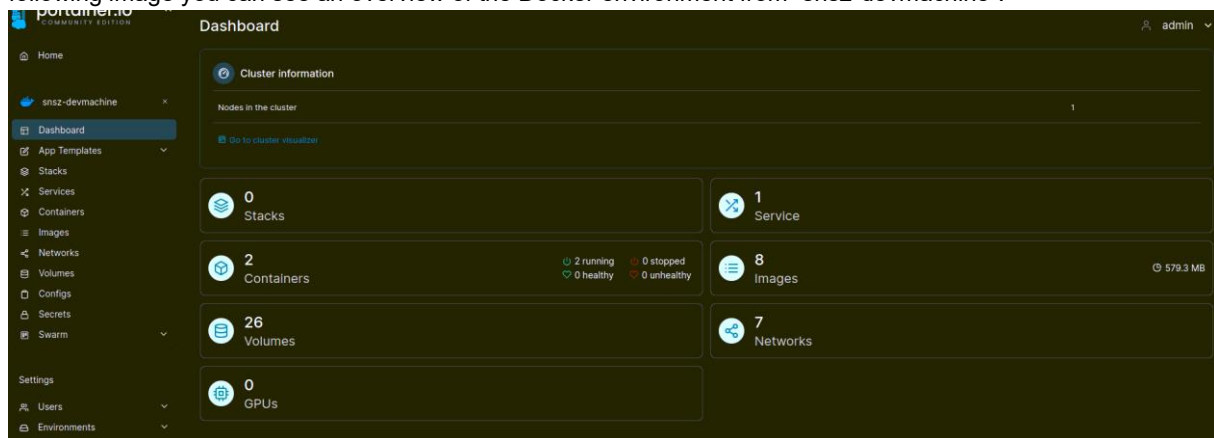
Connect

After clicking “Connect” the specified Agent machine is added to the Portainer server.

After returning to “Home” it can be seen that the specified development machine is now listed.



When clicking on one of the devices “local” or “snsz-devmachine” different configurations can be made. Within the following image you can see an overview of the Docker environment from “snsz-devmachine”:



5.2.4.2 IRF3000/IRF1000 as a Portainer Agent (Rootless Docker)

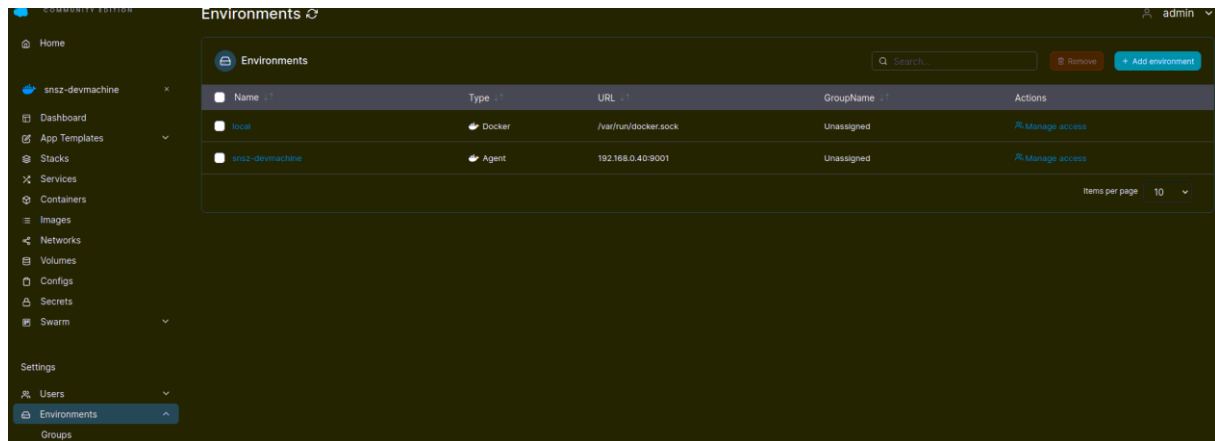
Since Rootless Docker runs by default on the IRF3000/IRF1000 there are some restrictions. For example, the overlay network driver used in the previous chapter cannot be used because it is not supported by Docker Rootless. This may change in the future.

For now, the default startup command of the Portainer Agent for the IRF3000/IRF1000 changes. A Portainer agent can be run on the IRF3000/IRF1000 using the following commands (in this case the IRF3000/IRF100 has the IP address 192.168.0.253):

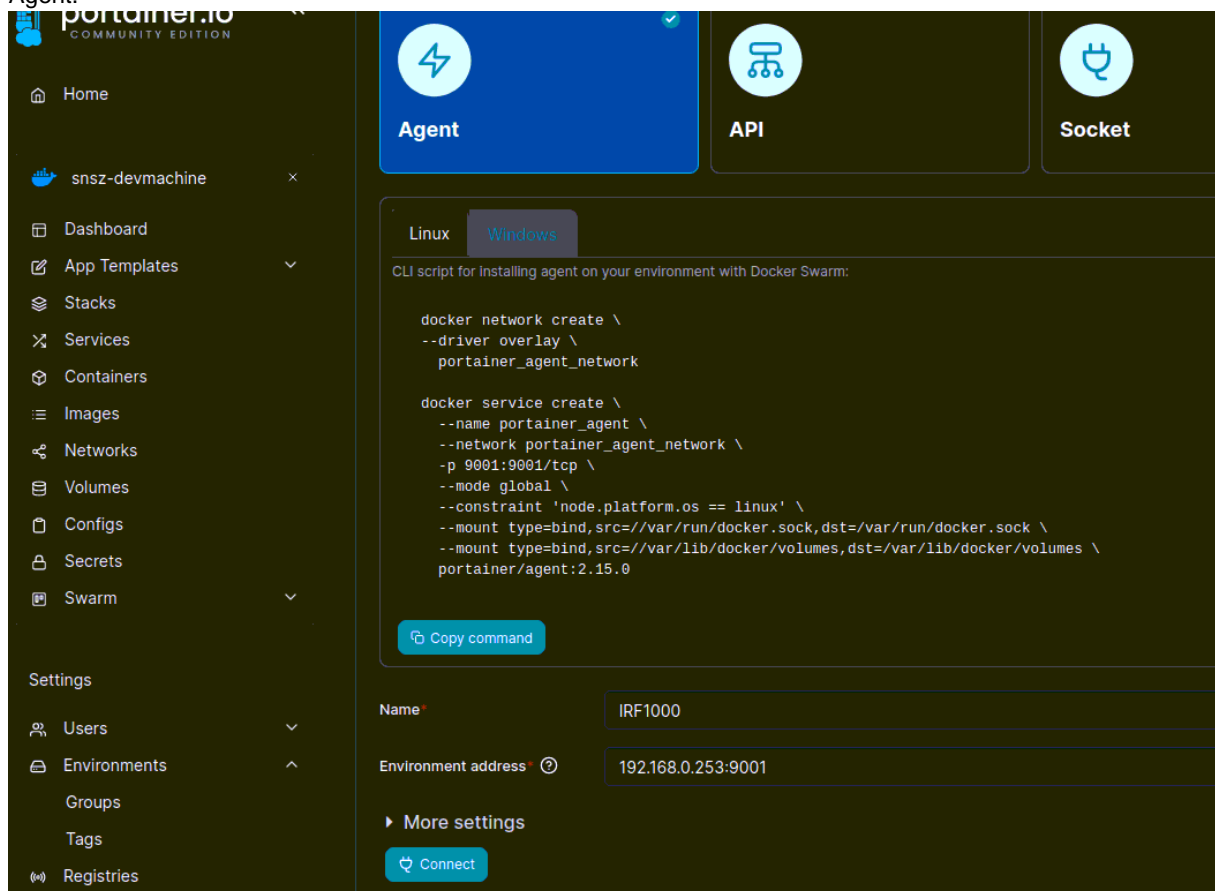
```
snsz@ADS-00000490:~$ export DOCKER_HOST=192.168.0.253:2375
```

```
snsz@ADS-00000490:~$ docker run -d -p 9001:9001 --name portainer_agent --restart always -v /var/run/docker.sock:/var/run/docker.sock -v /usr/local/docker/.local/share/docker/volumes:/var/lib/docker/volumes portainer/agent:2.15.0 5055ffa6764e0989204c9dac06d19647dab6f1abb1aac174d3de728054b0dd85
```

A new environment can be created in the upper right corner by clicking on Environment. This can be seen in the following image:

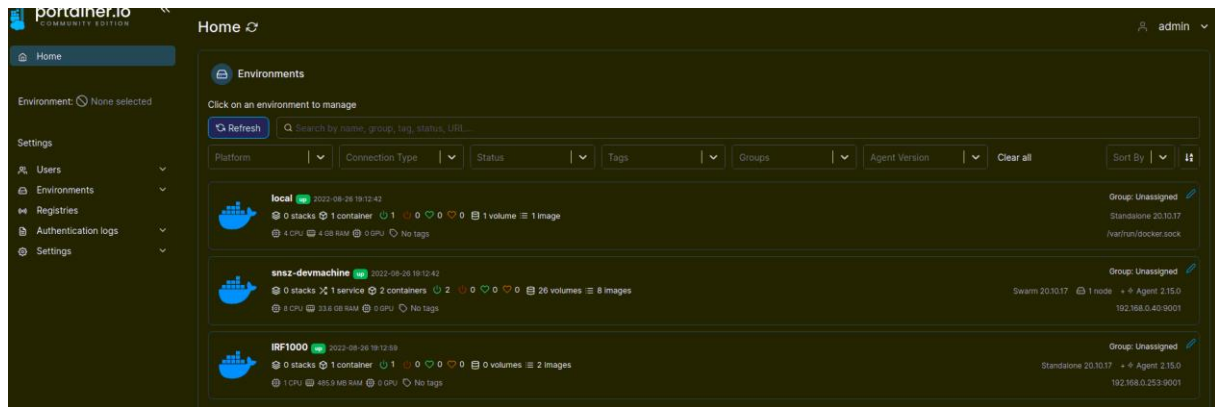


Once this is selected, “Docker” can be chosen again and the IRF3000/IRF1000 can be added as an Portainer Agent.



Once the name and IP address of the IRF3000/IRF1000 is given, it can be added by clicking “Connect”.

As before, the added device can now be seen on the Portainer Server homepage:



6 Troubleshooting

In the troubleshooting chapter explains common problems and how to solve them.

6.1 Date-Time Error

6.1.1 Error Description

Error response from daemon: Get "<https://registry-1.docker.io/v2/>": x509: certificate has expired or is not yet valid: current time 2022-08-25T09:42:19Z is before 2022-09-12T06:26:10Z.

This error occurs because the device system clock is set to a time before the issued certificate is valid.

6.1.2 Solution

Select "Configuration > General Settings > Date & time":

IRF3821

Configuration State

Diagnosics

Configuration

Config Wizard

IoT Wizard

IP configuration

Packet filter

I/Os / Cut & Alarm

General settings

System data

Date & time

User interface

Certificates

SCEP

Access control

Network

VPN

Services

IoT

System

Information

User: admin

Date & time

Date & Time: Sat Aug 27 11:33:29 CEST 2022

Time zone: Europe/Berlin

Enable timeserver synchronization (NTP): ☐

1. NTP server: pool.ntp.org

2. NTP server: de.pool.ntp.org

3. NTP server: ptbtime1.ptb.de

Enable NTP time server relay: ☐

Manual setting of date & time :

Date (day/month/year): 27 / 08 / 2022

Time (hour/minute/second): 09 / 33 / 29

Apply settings Reset changes

Now you can choose the time manually or obtain it via the NTP server.

6.2 Certificate signed by unknown authority

6.2.1 Error Description

Docker: Error response from daemon: Get "<https://registry-1.docker.io/v2/>": x509 certificate signed by unknown authority.

6.2.2 Solution

- Check if the certificates are valid
- Check whether the certificates are present