Docker Support

How-to Instructions and Examples

Contents

*__NOTE__: This document is subject to change without notice.

Description

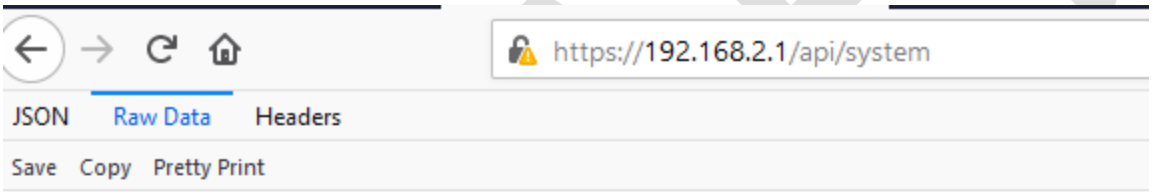
A Docker support feature is available in mPower software version 5.4 or higher on **Conduit 300 (MTCDT3AC)** devices. Docker is a client-server application technology which manages containers. Containerization is an efficient way to encapsulate applications and its dependencies in a lightweight, portable environment. The device UI provides a view of what's currently installed and running with regards to Docker on the device.

But in order to perform key Docker tasks such as building images, removing images, creating containers, and other functions, you must log into your device over SSH/debug console, move to the user_data directory, and execute them at the command prompt.

This document provides an overview of the Docker capability and how-to instructions regarding Docker configuration. Review the setup for some typical use cases such as running a simple Docker container, running a Docker container with publishing ports, example of Docker Compose, Microsoft Azure IoT Edge example, and Amazon AWS IoT Greengrass example (using a Lambda function).

Docker Capability

The system uses the API collection endpoint of docker (BOOLEAN value) under capabilities to identify if the Docker feature is enabled. If enabled, the value is true. This is available in: **/api/system/capabilities/docker**




```
{
   "code" : 200,
   "result" : {
      "accessoryCards" : [],
      "apiVersion" : "5.3.0-79-gaa0f0a4",
      "capabilities" : {
         "adc" : false,
         "battery" : false,
         "bluetooth" : true,
         "cell" : true,
         "cellWwan" : true,
         "din" : false,
         "docker" : true,
         "dout" : false,
         "ethSwitch" : true,
         "externalSerialPort" : false,
         "gpio" : false,
         "gps" : true,
         "lora" : false,
         "loraNetworkServer" : true,
         "nodeRed" : true,
         "rs232" : false,
         "rs422" : false,
         "rs485" : false,
         "serial" : false,
         "supercap" : false,
         "tpm" : true,
         "wifi" : true
      },
```


When Docker is available in the system, the **dockerd** service starts as soon as the system boots.

```
root@mtcpmhs:/var/config/home/admin# ps -ALL | grep dock
 4076  4076 ?        00:00:06 dockerd
 4076  4144 ?        00:00:03 dockerd
 4076  4145 ?        00:00:00 dockerd
 4076  4147 ?        00:00:00 dockerd
 4076  4177 ?        00:00:00 dockerd
 4076  4178 ?        00:00:00 dockerd
 4076  4216 ?        00:00:00 dockerd
 4076  4221 ?        00:00:03 dockerd
 4076  4222 ?        00:00:01 dockerd
 4076  4318 ?        00:00:02 dockerd
root@mtcpmhs:/var/config/home/admin# ps -aux | grep dock
root      4076  1.0  2.6 922264 54792 ?        Sl   13:44   0:18 /usr/bin/dockerd --registry-mirror=http://localhost:5000 --insecure-registry=http://localhost:5000 --raw-logs
root      4207  1.0  0.8 878532 17440 ?        Ssl  13:44   0:16 containerd --config /var/run/docker/containerd/containerd.toml --log-level info
root     16121  0.0  0.0   2816   608 ttyS1    S+   14:12   0:00 grep dock
root@mtcpmhs:/var/config/home/admin#
```

For the testing and troubleshooting purposes, it is possible to disable Docker.

Disable docker by executing the command below (rename the services that are responsible for the Docker feature). In this case, the system will consider that Docker is not supported and no changes (particularly to the Firewall) will be made.

```
$ sudo -s
# mv /usr/bin/dockerd /usr/bin/dockerd.orig
# mv /usr/bin/docker /usr/bin/docker.orig
# reboot -f
```

To restore Docker, execute the commands below and reboot the system:

```
$ sudo -s
# mv /usr/bin/dockerd.orig /usr/bin/dockerd
# mv /usr/bin/docker.orig /usr/bin/docker
# reboot -f
```

Docker and the changes in the system firewall rules and IP tables.

Docker adds its own firewall rules on daemon start and can **add/remove/modify** rules while the docker daemon is running.

While implementing the Docker support in mPower, we followed the recommendations from - the official Docker web site on how to tune iptables rules with the Docker daemon working in iptables mode - https://docs.docker.com/network/iptables/

Docker adds rules into two tables on start, ***nat*** and ***filter***. Also, the daemon creates four custom chains: **DOCKER-USER**, **DOCKER**, **DOCKER-ISOLATION-STAGE-1,** and **DOCKER-ISOLATION-STAGE-2**.

In the **filter** table, Docker changes rules in the FORWARD chain only. The Docker changes do NOT affect **INPUT** and **OUTPUT** chain rules.

```
root@mtcpmhs:/var/config/home/admin# iptables -L -v -n
Chain INPUT (policy DROP 6 packets, 921 bytes)
 pkts bytes target     prot opt in     out     source               destination
  861 85569 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
 2693  431K USER_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
 2693  431K KEEP_STATE_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  697 49760 DOS_PREVENTION  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  697 49760 DNS_SERVER_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   42  2793 DHCP_SERVER_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   42  2793 DHCP_CLIENT_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   42  2793 TRUSTED_IP_UDP_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   42  2793 HTTP_LAN_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   42  2793 HTTPS_LAN_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    6   921 NODERED_LAN_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    6   921 SSH_LAN_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    6   921 ICMP_LAN_INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP 307 packets, 35051 bytes)
 pkts bytes target     prot opt in     out     source               destination
  307 35051 DOCKER-USER  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  307 35051 DOCKER-ISOLATION-STAGE-1  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED
    0     0 DOCKER     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  docker0 docker0  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  861 85569 ACCEPT     all  --  *      lo      0.0.0.0/0            0.0.0.0/0
 2323  837K USER_OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  180 32549 KEEP_STATE_OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 DNS_OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 DHCP_SERVER_OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 DHCP_CLIENT_OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 APP_MANAGER_OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 FTP_OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 WAN_MANAGEMENT_OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 ICMP_OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

In the **nat** table, Docker changes the PREROUTING, OUTPUT and POSTROUTING chains.

```
root@mtcpmhs:/var/config/home/admin# iptables -t nat -L -v -n
Chain PREROUTING (policy ACCEPT 837 packets, 72155 bytes)
 pkts bytes target     prot opt in     out     source               destination
  908 76896 USER_PREROUTE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  908 76896 LOOPBACK_PREROUTE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  526 36296 DOCKER     all  --  *      *       0.0.0.0/0            0.0.0.0/0            ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT 523 packets, 36062 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 692 packets, 49074 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DOCKER     all  --  *      *       0.0.0.0/0           !127.0.0.0/8          ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT 71 packets, 4246 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      !docker0  172.17.0.0/16        0.0.0.0/0
  723 51241 USER_POSTROUTE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  723 51241 LOOPBACK_POSTROUTE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  723 51241 MTR_POSTROUTE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  723 51241 WAN_IPSEC_POSTROUTE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  723 51241 WAN_MASQ_POSTROUTE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Use only the **DOCKER-USER** chain. All other chains are intensively used by docker daemon during runtime. The docker team recommends using the **DOCKER-USER** chain instead of the ***FORWARD*** chain to avoid any confusion with rules.

FIREWALL Implementation and Changes to Support Docker

The firewall can be executed multiple times during the device's runtime. The firewall has different command-line options to tell it what to do. The firewall can change ipv6, ipv4, or both sets of rules. It can change the logging level and other details. The firewall's main limitation is that it can only modify iptables rules for log details. In all other cases, the firewall clears all iptables rules and recreates all chains and rules from scratch.

The Docker and firewall integration was added for ipv4 iptables mode only.

The device API provides the ability to enable or disable Docker at firewall runtime: https://gitlab.multitech.net/multitech/mtsDeviceAPI/

Refer to the **dockerEnabled** endpoint in **/api/firewall/summary**


```
/api/firewall/summary
▼ result:
    commissionMode:          false
  ▶ customDiagnostic:        {…}
  ▶ ddns:                    {…}
  ▶ dhcpClients:             […]
    dhcpClients6:            []
  ▶ dhcpServers:             […]
    dhcpServers6:            []
    dnsEnabled:              true
    dockerEnabled:           true
  ▶ filters:                 […]
  ▶ firewall:                {…}
  ▶ gps:                     {…}
```


Docker configurations examples

This document provides the most common Docker use cases for IoT applications.

Before testing Docker containers, check: 1) your internet connection, 2) the current date-time, and 3) that the device has enough disk space.

```
root@mtcdt3hs:~# date
Mon Jan 11 13:09:09 EET 2021
root@mtcdt3hs:~#
root@mtcdt3hs:~#
root@mtcdt3hs:~# ping google.com
PING google.com (172.217.16.14) 56(84) bytes of data.
64 bytes from mil02s06-in-f14.1e100.net (172.217.16.14): icmp_seq=1 ttl=119 time=21.5 ms
^C
--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 21.503/21.503/21.503/0.000 ms
root@mtcdt3hs:~#
root@mtcdt3hs:~#
root@mtcdt3hs:~#
root@mtcdt3hs:~#
root@mtcdt3hs:~# df -h
Filesystem                  Size  Used Avail Use% Mounted on
none                        962M  8.0K  962M   1% /dev
overlay                     3.1G   15M  2.9G   1% /
tmpfs                       993M  536K  992M   1% /run
tmpfs                       993M  1.1M  992M   1% /var/volatile
/dev/mmcblk1p6               27M  767K   24M   4% /var/oem
/dev/mapper/mapped_config    29M  5.3M   21M  21% /var/config_redundant
/dev/mapper/mapped_config2   29M  5.2M   21M  21% /var/config
cgroup                      993M     0  993M   0% /sys/fs/cgroup
```

Useful Docker Commands:

 # docker --help

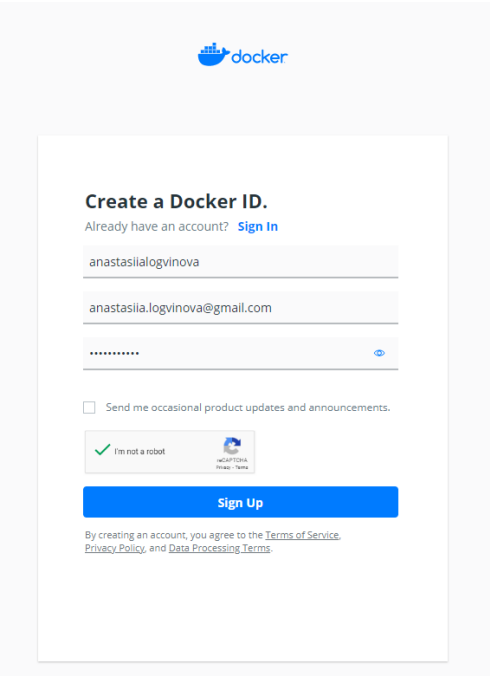
docker login

docker container ls

```
# docker logs -f <container id>

# docker kill <container ID or container Name>

# docker rm <container ID or container Name>

# docker images

# docker rmi <image ID or container Name>

# docker volume ls

# docker system prune --all
```

Sign up for a Docker account

You must have a Docker account to work with Docker. Please sign up here: https://hub.docker.com/signup



You will receive an email notification to confirm your email. As soon as you confirm it, your account will be active and you can start working with Docker.

In the device console, log into docker by executing the command

 # docker login

```
root@mtcpmhs:/var/config/home/admin# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: anastasiialogvinova
Password:
WARNING! Your password will be stored unencrypted in /home/root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@mtcpmhs:/var/config/home/admin#
```

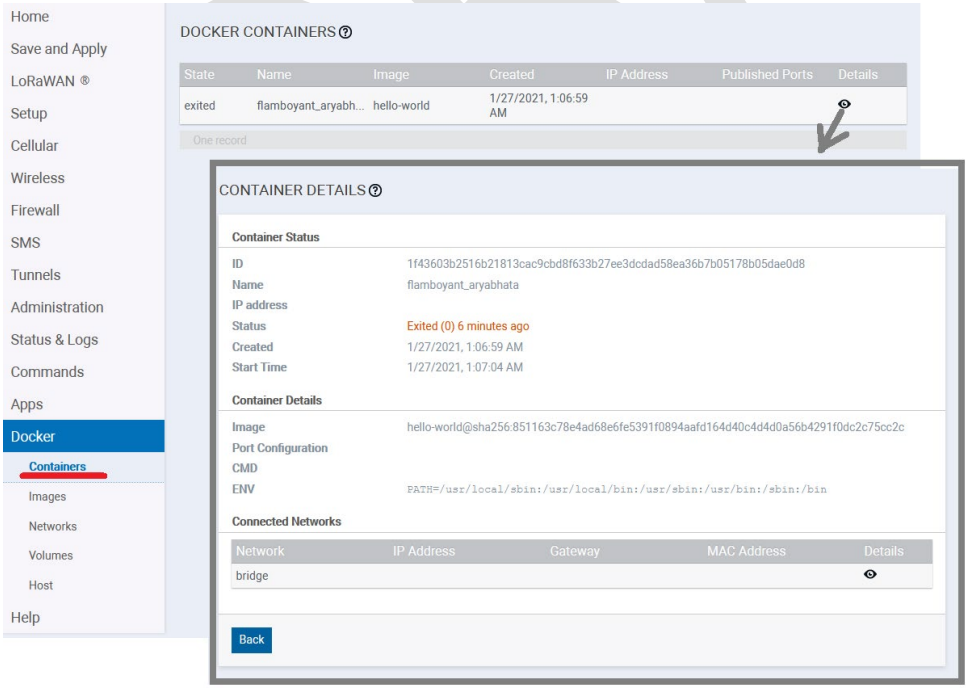
Simple Docker container

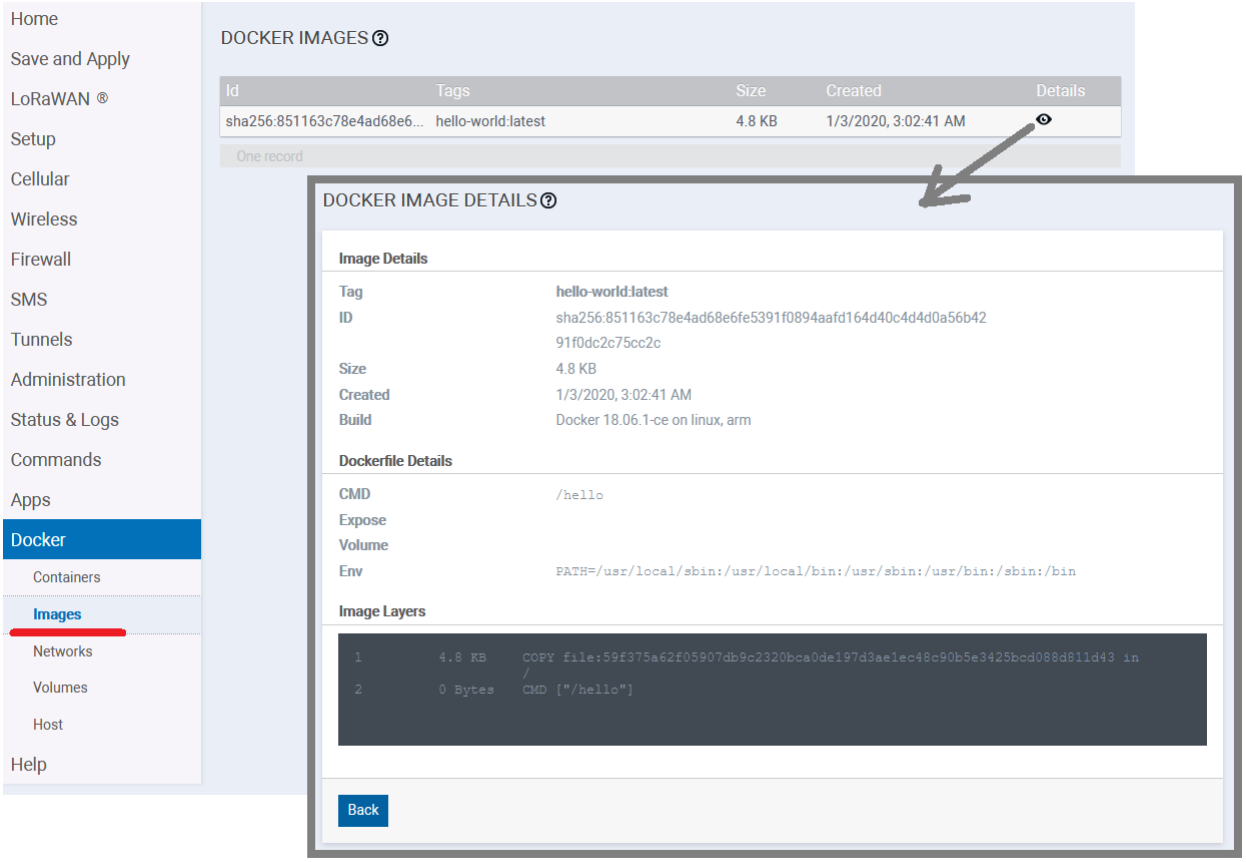
To check that Docker is working, run the **hello-world** example. https://hub.docker.com/_/hello-world. This docker image is from the docker hub.

1. Log in to docker `# docker login`
2. Execute the command `# docker run hello-world`
3. The system will not find the hello-world locally. But it will instead pull and install hello-world app, provided there is an Internet connection.

```
root@mtcpmhs:/var/config/home/admin# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: anastasiialogvinova
Password:
WARNING! Your password will be stored unencrypted in /home/root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@mtcpmhs:/var/config/home/admin# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
4ee5c797bcd7: Pull complete
Digest: sha256:31b9c7d48790f0d8c50ab433d9c3b7e17666d6993084c002c2ff1ca09b96391d
Status: Downloaded newer image for hello-world:latest
[11856.051771] docker0: port 1(veth83e76c3) entered blocking state
[11856.072220] docker0: port 1(veth83e76c3) entered disabled state
[11856.098640] device veth83e76c3 entered promiscuous mode
[11857.002710] cgroup: runc (24195) created nested cgroup for controller "memory" which has incomplete hierarchy support. Nested cgroups may change behavior in the future.
[11857.037345] cgroup: "memory" requires setting use_hierarchy to 1 on the root
[11859.254726] eth0: renamed from vethc73ad81
[11859.299477] docker0: port 1(veth83e76c3) entered blocking state
[11859.305570] docker0: port 1(veth83e76c3) entered forwarding state

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm32v7)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[11860.491198] docker0: port 1(veth83e76c3) entered disabled state
[11860.500814] vethc73ad81: renamed from eth0
[11860.629874] docker0: port 1(veth83e76c3) entered disabled state
[11860.667043] device veth83e76c3 left promiscuous mode
[11860.672067] docker0: port 1(veth83e76c3) entered disabled state
root@mtcpmhs:/var/config/home/admin#
```

As soon as the application is installed, the system displays it on the **Containers** and **Images** pages under the **Docker** menu in the device UI.





4. To delete the image, you must first delete containers that use the image by executing this command:

```
# docker rm container-name
```

Then you can delete the image by executing this command:

```
# docker rmi image-name
```

```
root@mtcpmhs:/var/persistent/docker# docker rm festive_pare
festive_pare
root@mtcpmhs:/var/persistent/docker# docker rm  flamboyant_aryabhata
flamboyant_aryabhata
root@mtcpmhs:/var/persistent/docker# docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:31b9c7d48790f0d8c50ab433d9c3b7e17666d6993084c002c2ff1ca09b96391d
Deleted: sha256:851163c78e4ad68e6fe5391f0894aafd164d40c4d4d0a56b4291f0dc2c75cc2c
Deleted: sha256:2536d8d4e4b1baa6515d44eb77a1402d6be0a533e7d191c51cb8428ba5ece3f4
root@mtcpmhs:/var/persistent/docker#
```

Docker container with publishing ports

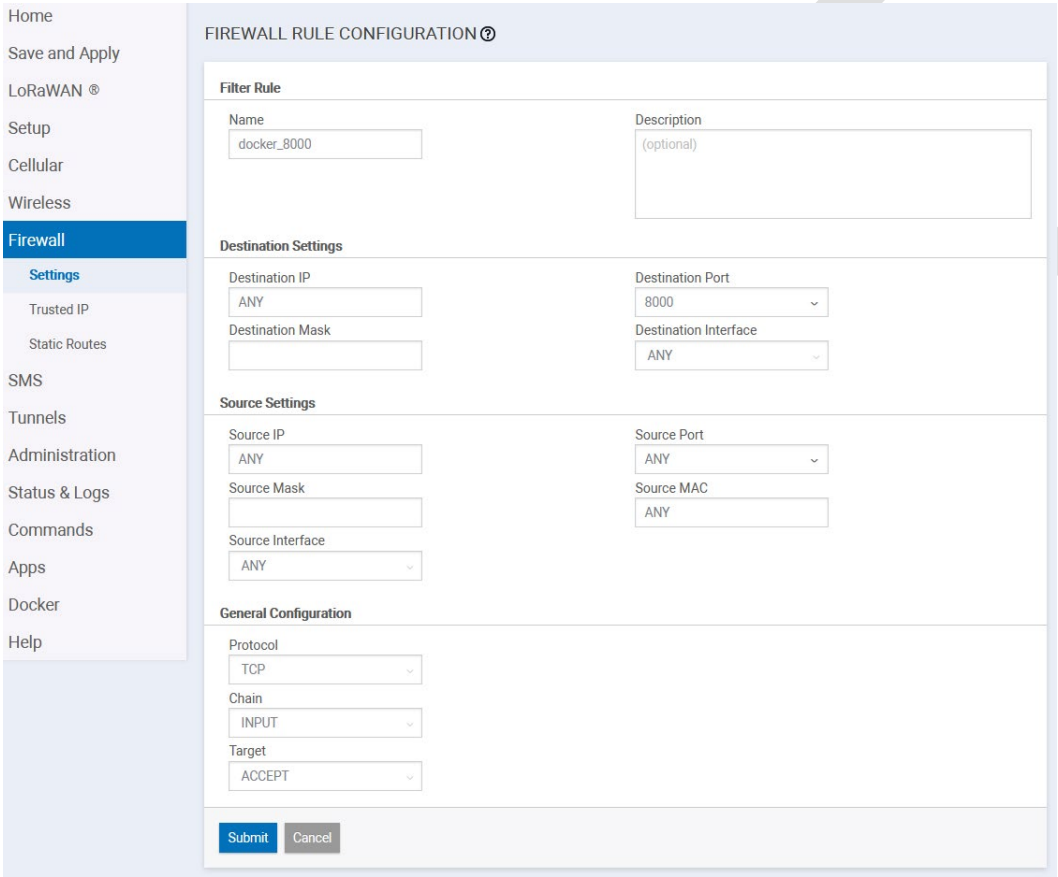
In this example, we use the Ubuntu image and default python simple HTTP server to host and add access to the local file system via HTTP protocol.

We can do that using **host network mode** and default **bridge network mode.**

Host Network Mode

The container runs on port 8000. If we start the container in the host network mode, we need to make a change to the firewall settings. Add an **ACCEPT** rule (**target**) to the **INPUT chain** for the **port 8000.**

Add the rule, and click **Save and Apply**, to save the changes.



Follow the steps below:

1. Download the image:

```
# docker pull ubuntu
```

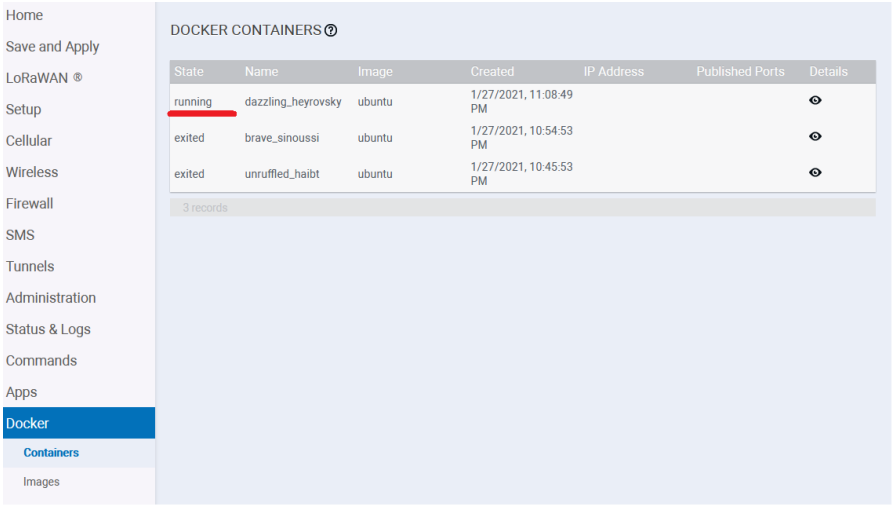
```
root@mtcpmhs:/var/persistent/docker# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
f5b689817f39: Pull complete
8b105a146cef: Pull complete
22943c6e232d: Pull complete
Digest: sha256:703218c0465075f4425e58fac086e09e1de5c340b12976ab9eb8ad26615c3715
Status: Downloaded newer image for ubuntu:latest
root@mtcpmhs:/var/persistent/docker#
```

2. Start the container in the **host network mode:**

```
# docker run --network host -it ubuntu /bin/bash
```

You enter the container and can execute commands there. To exit, the container type **exit**.

3. You can see the running container on the **Containers** page under **Docker**:



4. Execute the commands in the container (including updating or installing packages):

```
# apt-get update && apt-get upgrade -y
```

```
asht@mtcpmhs:/var/persistent/docker# docker run --network host -it ubuntu /bin/ba
root@mtcpmhs:/# apt-get update && apt-get upgrade -y
Get:1 http://ports.ubuntu.com/ubuntu-ports focal InRelease [265 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports focal-updates InRelease [114 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports focal-backports InRelease [101 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports focal-security InRelease [109 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports focal/main armhf Packages [1227 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports focal/multiverse armhf Packages [141 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports focal/restricted armhf Packages [10.8 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports focal/universe armhf Packages [10.9 MB]
Get:9 http://ports.ubuntu.com/ubuntu-ports focal-updates/multiverse armhf Packages [5434 B]
Get:10 http://ports.ubuntu.com/ubuntu-ports focal-updates/universe armhf Packages [723 kB]
Get:11 http://ports.ubuntu.com/ubuntu-ports focal-updates/restricted armhf Packages [14.6 kB]
Get:12 http://ports.ubuntu.com/ubuntu-ports focal-updates/main armhf Packages [777 kB]
Get:13 http://ports.ubuntu.com/ubuntu-ports focal-backports/universe armhf Packages [4306 B]
Get:14 http://ports.ubuntu.com/ubuntu-ports focal-security/universe armhf Packages [465 kB]
Get:15 http://ports.ubuntu.com/ubuntu-ports focal-security/main armhf Packages [387 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports focal-security/multiverse armhf Packages [934 B]
Get:17 http://ports.ubuntu.com/ubuntu-ports focal-security/restricted armhf Packages [8245 B]
Fetched 15.3 MB in 24s (629 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  apt libapt-pkg6.0 libc-bin libc6
4 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 4622 kB of archives.
After this operation, 2048 B disk space will be freed.
Get:1 http://ports.ubuntu.com/ubuntu-ports focal-updates/main armhf libc6 armhf 2.31-0ubuntu9.2 [2133 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports focal-updates/main armhf libc-bin armhf 2.31-0ubuntu9.2 [493 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports focal-updates/main armhf libapt-pkg6.0 armhf 2.0.4 [751 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports focal-updates/main armhf apt armhf 2.0.4 [1245 kB]
Fetched 4622 kB in 3s (1340 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
(Reading database ... 4115 files and directories currently installed.)
Preparing to unpack .../libc6_2.31-0ubuntu9.2_armhf.deb ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (Can't locate Term/ReadLine.pm in @INC (you may need to install the Term::ReadLine module) (@INC contains: /etc/perl /usr/local/lib/arm-linux-gnueabihf/perl/5.30.0 /usr/local/share/perl
sr/lib/arm-linux-gnueabihf/perl5/5.30 /usr/share/perl5 /usr/lib/arm-linux-gnueabihf/perl/5.30 /usr/share/perl/5.30 /usr/local/lib/site_perl /usr/lib/arm-linux-gnueabihf/perl-base) at /usr/share/
onf/FrontEnd/Readline.pm line 7.)
debconf: falling back to frontend: Teletype
Unpacking libc6:armhf (2.31-0ubuntu9.2) over (2.31-0ubuntu9.1) ...
Setting up libc6:armhf (2.31-0ubuntu9.2) ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (Can't locate Term/ReadLine.pm in @INC (you may need to install the Term::ReadLine module) (@INC contains: /etc/perl /usr/local/lib/arm-linux-gnueabihf/perl/5.30.0 /usr/local/share/perl
sr/lib/arm-linux-gnueabihf/perl5/5.30 /usr/share/perl5 /usr/lib/arm-linux-gnueabihf/perl/5.30 /usr/share/perl/5.30 /usr/local/lib/site_perl /usr/lib/arm-linux-gnueabihf/perl-base) at /usr/share/
onf/FrontEnd/Readline.pm line 7.)
debconf: falling back to frontend: Teletype
(Reading database ... 4115 files and directories currently installed.)
Preparing to unpack .../libc-bin_2.31-0ubuntu9.2_armhf.deb ...
Unpacking libc-bin (2.31-0ubuntu9.2) over (2.31-0ubuntu9.1) ...
Setting up libc-bin (2.31-0ubuntu9.2) ...
(Reading database ... 4115 files and directories currently installed.)
Preparing to unpack .../libapt-pkg6.0_2.0.4_armhf.deb ...
Unpacking libapt-pkg6.0:armhf (2.0.4) over (2.0.2ubuntu0.2) ...
Setting up libapt-pkg6.0:armhf (2.0.4) ...
(Reading database ... 4115 files and directories currently installed.)
Preparing to unpack .../archives/apt_2.0.4_armhf.deb ...
Unpacking apt (2.0.4) over (2.0.2ubuntu0.2) ...
Setting up apt (2.0.4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
root@mtcpmhs:/#
```

```
# apt-get install -y python2
```

```
root@mtcpmhs:/# apt-get install -y python2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  file libexpat1 libmagic-mgc libmagic1 libpython2-stdlib libpython2.7-minimal
  libpython2.7-stdlib libreadline8 libsqlite3-0 libssl1.1 mime-support
  python2-minimal python2.7 python2.7-minimal readline-common xz-utils
Suggested packages:
  python2-doc python-tk python2.7-doc binutils binfmt-support readline-doc
The following NEW packages will be installed:
  file libexpat1 libmagic-mgc libmagic1 libpython2-stdlib libpython2.7-minimal
  libpython2.7-stdlib libreadline8 libsqlite3-0 libssl1.1 mime-support python2
  python2-minimal python2.7 python2.7-minimal readline-common xz-utils
0 upgraded, 17 newly installed, 0 to remove and 0 not upgraded.
Need to get 5727 kB of archives.
After this operation, 25.2 MB of additional disk space will be used.
Get:1 http://ports.ubuntu.com/ubuntu-ports focal-updates/universe armhf libpython2.7-minimal armhf 2.7.18-1~20.04 [335 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports focal-updates/universe armhf python2.7-minimal armhf 2.7.18-1~20.04 [1084 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports focal/universe armhf python2-minimal armhf 2.7.17-2ubuntu4 [27.5 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports focal-updates/main armhf libssl1.1 armhf 1.1.1f-1ubuntu2.1 [1082 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports focal/main armhf mime-support all 3.64ubuntu1 [30.6 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports focal/main armhf libexpat1 armhf 2.2.9-1build1 [53.3 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports focal/main armhf readline-common all 8.0-4 [53.5 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports focal/main armhf libreadline8 armhf 8.0-4 [109 kB]
Get:9 http://ports.ubuntu.com/ubuntu-ports focal-updates/main armhf libsqlite3-0 armhf 3.31.1-4ubuntu0.2 [467 kB]
Get:10 http://ports.ubuntu.com/ubuntu-ports focal-updates/universe armhf libpython2.7-stdlib armhf 2.7.18-1~20.04 [1813 kB]
Get:11 http://ports.ubuntu.com/ubuntu-ports focal-updates/universe armhf python2.7 armhf 2.7.18-1~20.04 [248 kB]
Get:12 http://ports.ubuntu.com/ubuntu-ports focal/universe armhf libpython2-stdlib armhf 2.7.17-2ubuntu4 [7072 B]
Get:13 http://ports.ubuntu.com/ubuntu-ports focal/universe armhf python2 armhf 2.7.17-2ubuntu4 [26.5 kB]
Get:14 http://ports.ubuntu.com/ubuntu-ports focal/main armhf libmagic-mgc armhf 1:5.38-4 [218 kB]
Get:15 http://ports.ubuntu.com/ubuntu-ports focal/main armhf libmagic1 armhf 1:5.38-4 [68.3 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports focal/main armhf file armhf 1:5.38-4 [22.4 kB]
Get:17 http://ports.ubuntu.com/ubuntu-ports focal-updates/main armhf xz-utils armhf 5.2.4-1ubuntu1 [80.9 kB]
Fetched 5727 kB in 4s (1321 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package libpython2.7-minimal:armhf.
(Reading database ... 4115 files and directories currently installed.)
Preparing to unpack .../00-libpython2.7-minimal_2.7.18-1~20.04_armhf.deb ...
Unpacking libpython2.7-minimal:armhf (2.7.18-1~20.04) ...
Selecting previously unselected package python2.7-minimal.
Preparing to unpack .../01-python2.7-minimal_2.7.18-1~20.04_armhf.deb ...
Unpacking python2.7-minimal (2.7.18-1~20.04) ...
Selecting previously unselected package python2-minimal.
Preparing to unpack .../02-python2-minimal_2.7.17-2ubuntu4_armhf.deb ...
Unpacking python2-minimal (2.7.17-2ubuntu4) ...
Selecting previously unselected package libssl1.1:armhf.
Preparing to unpack .../03-libssl1.1_1.1.1f-1ubuntu2.1_armhf.deb ...
Unpacking libssl1.1:armhf (1.1.1f-1ubuntu2.1) ...
Selecting previously unselected package mime-support.
Preparing to unpack .../04-mime-support_3.64ubuntu1_all.deb ...
Unpacking mime-support (3.64ubuntu1) ...
Selecting previously unselected package libexpat1:armhf.
Preparing to unpack .../05-libexpat1_2.2.9-1build1_armhf.deb ...
Unpacking libexpat1:armhf (2.2.9-1build1) ...
```

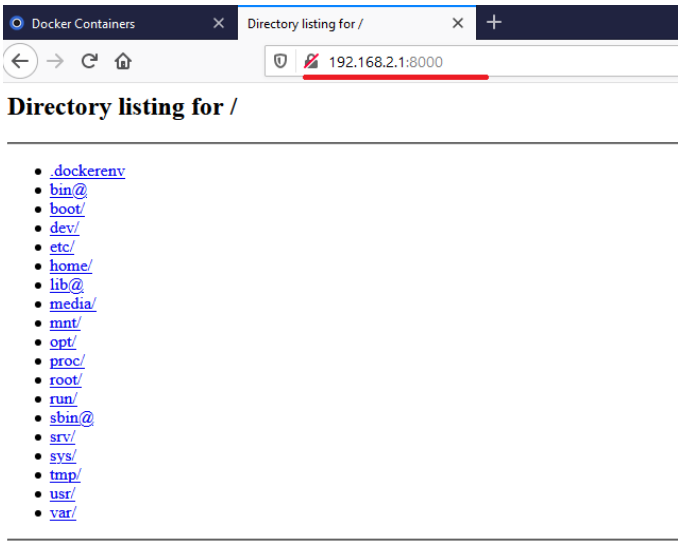
```
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (Can't locate Term/ReadLine.pm in @INC (you may need to install the Term::ReadLine module) (@INC contains: /etc/perl /usr/local/lib/arm-linux-gnueabihf/perl/5.30.0
sr/lib/arm-linux-gnueabihf/perl5/5.30 /usr/share/perl5 /usr/lib/arm-linux-gnueabihf/perl/5.30 /usr/share/perl/5.30 /usr/local/lib/site_perl /usr/lib/arm-linux-gnueabihf/per
onf/FrontEnd/Readline.pm line 7.)
debconf: falling back to frontend: Teletype
Setting up libsqlite3-0:armhf (3.31.1-4ubuntu0.2) ...
Setting up libmagic1:armhf (1:5.38-4) ...
Setting up file (1:5.38-4) ...
Setting up xz-utils (5.2.4-1ubuntu1) ...
update-alternatives: using /usr/bin/xz to provide /usr/bin/lzma (lzma) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/lzma.1.gz because associated file /usr/share/man/man1/xz.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/unlzma.1.gz because associated file /usr/share/man/man1/unxz.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzcat.1.gz because associated file /usr/share/man/man1/xzcat.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzmore.1.gz because associated file /usr/share/man/man1/xzmore.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzless.1.gz because associated file /usr/share/man/man1/xzless.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzdiff.1.gz because associated file /usr/share/man/man1/xzdiff.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzcmp.1.gz because associated file /usr/share/man/man1/xzcmp.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzgrep.1.gz because associated file /usr/share/man/man1/xzgrep.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzegrep.1.gz because associated file /usr/share/man/man1/xzegrep.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzfgrep.1.gz because associated file /usr/share/man/man1/xzfgrep.1.gz (of link group lzma) doesn't exist
Setting up readline-common (8.0-4) ...
Setting up libreadline8:armhf (8.0-4) ...
Setting up libpython2.7-stdlib:armhf (2.7.18-1~20.04) ...
Setting up python2.7 (2.7.18-1~20.04) ...
Setting up libpython2-stdlib:armhf (2.7.17-2ubuntu4) ...
Setting up python2 (2.7.17-2ubuntu4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
```

```
# cd /

# python2 -m SimpleHTTPServer 8000
```

```
root@mtcpmhs:/# python2 -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
```

For this example to work via http://192.168.2.1:8000/, you must add the one accept rule mentioned earlier to the INPUT chain. **Note:** the IP address (192.168.2.1) depends on your network settings**.**

4. Enter 192.168.2.1:8000 in the address bar of your browser to display the directory list:

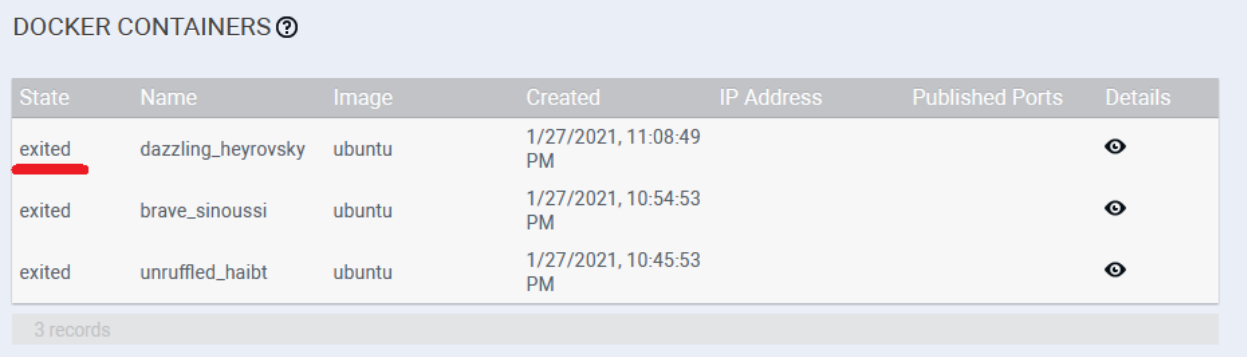


5. As soon as you stop the application (**ctrl-C**), the directory will not be accessible.

6. To exit the container, execute the command: **exit**.

```
root@mtcpmhs:/# exit
exit
root@mtcpmhs:/var/config/home/admin#
```

The container status changes to exited on the **Containers** page



State	Name	Image	Created	IP Address	Published Ports	Details
exited	dazzling_heyrovsky	ubuntu	1/27/2021, 11:08:49 PM			
exited	brave_sinoussi	ubuntu	1/27/2021, 10:54:53 PM			
exited	unruffled_haibt	ubuntu	1/27/2021, 10:45:53 PM			

3 records

Bridge Network Mode

When you run Docker in the bridge network mode, you do not have to add Firewall rules manually.

Remove or disable the **INPUT ACCEPT rule** for the **port 8000**. Click **Save and Apply** to save the changes.

Follow the steps below:

1. Download the image:

```
# docker pull ubuntu
```

If the ubuntu image is already downloaded, skip this step.

2. Start container in the **bridge network mode.** Execute the command to expose the hosted port:

```
# docker run -p 0.0.0.0:8000:8000/tcp -it ubuntu /bin/bash
```

```
root@mtcpmhs:/var/config/home/admin#
root@mtcpmhs:/var/config/home/admin#
root@mtcpmhs:/var/config/home/admin# docker run -p 0.0.0.0:8000:8000/tcp -it ubuntu /bin/bash
root@778987ce8800:/#
```

You enter the container and can execute commands there. To exit the container, type **exit**.

You can see a new container with a **running** state on the **Docker Containers** page:

DOCKER CONTAINERS

State	Name	Image	Created	IP Address	Published Ports	Details
running	sweet_payne	ubuntu	1/27/2021, 11:17:25 PM	172.17.0.2	8000:8000/tcp	
exited	dazzling_heyrovsky	ubuntu	1/27/2021, 11:08:49 PM			
exited	brave_sinoussi	ubuntu	1/27/2021, 10:54:53 PM			
exited	unruffled_haibt	ubuntu	1/27/2021, 10:45:53 PM			

4 records

3. Execute the commands below in the container (including updating or installing packages):

```
# apt-get update && apt-get upgrade -y
```

```
root@778987ce8800:/# apt-get update && apt-get upgrade -y
Get:1 http://ports.ubuntu.com/ubuntu-ports focal InRelease [265 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports focal-updates InRelease [114 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports focal-backports InRelease [101 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports focal-security InRelease [109 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports focal/restricted armhf Packages [10.8 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports focal/universe armhf Packages [10.9 MB]
Get:7 http://ports.ubuntu.com/ubuntu-ports focal/multiverse armhf Packages [141 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports focal/main armhf Packages [1227 kB]
Reading package lists... Done
E: Release file for http://ports.ubuntu.com/ubuntu-ports/dists/focal-updates/InRelease is not valid yet (invalid for another 1h 43min 46s). Updates for this repository will not be applied.
E: Release file for http://ports.ubuntu.com/ubuntu-ports/dists/focal-backports/InRelease is not valid yet (invalid for another 18min 6s). Updates for this repository will not be applied.
E: Release file for http://ports.ubuntu.com/ubuntu-ports/dists/focal-security/InRelease is not valid yet (invalid for another 40min 13s). Updates for this repository will not be applied.
root@778987ce8800:/#
```

```
# apt-get install -y python2

# cd /

# python2 -m SimpleHTTPServer 8000
```

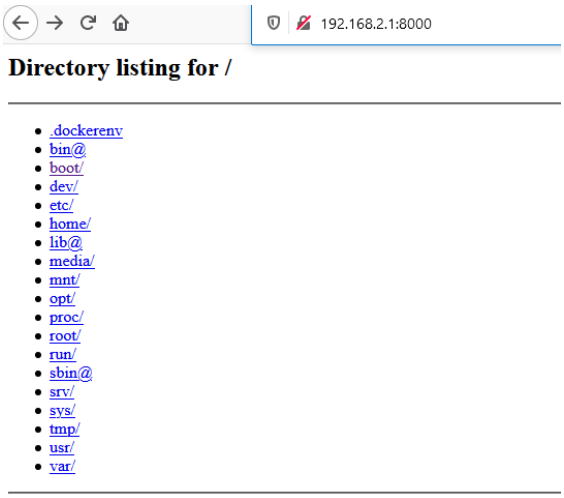
```
root@778987ce8800:/# cd /
root@778987ce8800:/# python2 -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
```

If everything is configured properly, you will see the following iptables rule:

```
# iptables -S | grep 8000

-A DOCKER -d 172.17.0.2/32 ! -i docker0 -o docker0 -p tcp -m tcp --dport 8000 -j ACCEPT
```

```
iptables -S | grep 8000
-A DOCKER -d 172.17.0.2/32 ! -i docker0 -o docker0 -p tcp -m tcp --dport 8000 -j ACCEPT
```

4. Enter 192.168.2.1:8000 in the address bar of your browser to see the directory list:



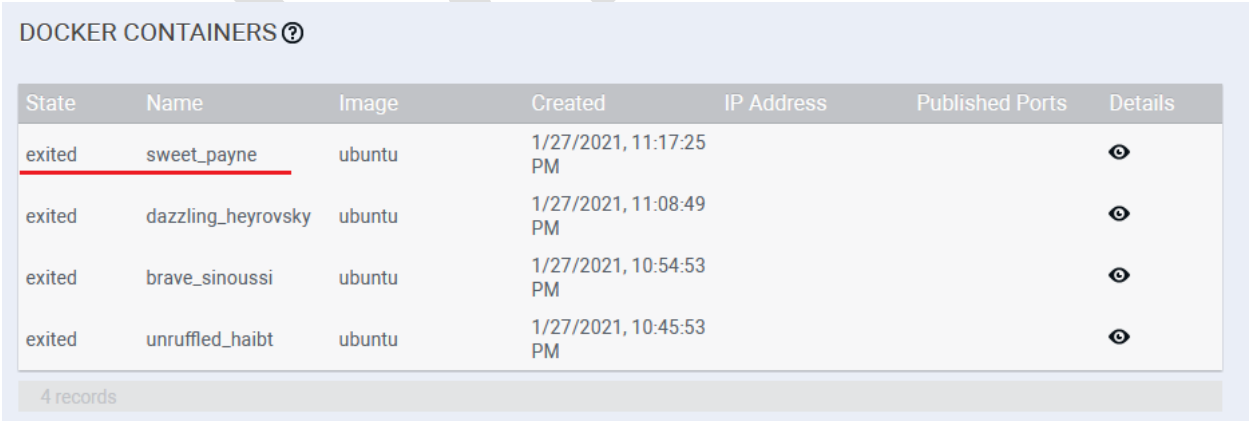
5. To stop the container, type ctrl+c

```
root@a3fe209b5c33:/# python2 -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
192.168.2.242 - - [27/Jan/2021 21:22:59] "GET /var/log/faillog HTTP/1.1" 200 -
192.168.2.242 - - [27/Jan/2021 21:23:05] "GET / HTTP/1.1" 200 -
 ^CTraceback (most recent call last):
  File "/usr/lib/python2.7/runpy.py", line 174, in _run_module_as_main
    "__main__", fname, loader, pkg_name)
  File "/usr/lib/python2.7/runpy.py", line 72, in _run_code
    exec code in run_globals
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 235, in <module>
    test()
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 231, in test
    BaseHTTPServer.test(HandlerClass, ServerClass)
  File "/usr/lib/python2.7/BaseHTTPServer.py", line 610, in test
    httpd.serve_forever()
  File "/usr/lib/python2.7/SocketServer.py", line 231, in serve_forever
    poll_interval)
  File "/usr/lib/python2.7/SocketServer.py", line 150, in _eintr_retry
    return func(*args)
KeyboardInterrupt
root@a3fe209b5c33:/#
```

6. To exit the container, type **exit**.

```
root@a3fe209b5c33:/# exit
exit
root@mtcpmhs:/var/config/home/admin#
```

The status of the container will change to **exited** on the **Docker Containers** page.



Docker Compose Example: Set Up and Run WordPress

This example is based on: https://docs.docker.com/compose/wordpress/ that demonstrates how to use Docker Compose to run WordPress in an isolated environment built with Docker containers.

The original compose file contains a reference to a MySQL image for the Intel platform only. We should adjust the script for the arm32 platform. To do that, we have to replace **image: mysql:5.7** with **image: beercan1989/arm-mysql:5.7** in the **docker-compose.yml** file**.**

1. **Execute the following commands on the device to define the project:**

```
# cd
# mkdir my_wordpress
# cd my_wordpress/
# touch docker-compose.yml
# vi docker-compose.yml
```

2. Copy and paste the compose snippet from the example to the **docker-compose.yml.** Make sure to change the Intel Platform to arm32 (see line is in RED in the example below)

```
version: '3.3'

services:
   db:
     image: beercan1989/arm-mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}
```

```
root@mtcpmhs:/var/persistent/docker/volumes# cd
root@mtcpmhs:~# mkdir my_wordpress
root@mtcpmhs:~# cd my_wordpress/
root@mtcpmhs:~/my_wordpress# touch docker-compose.yml
root@mtcpmhs:~/my_wordpress# vi docker-compose.yml
root@mtcpmhs:~/my_wordpress# vi docker-compose.yml
root@mtcpmhs:~/my_wordpress# cat docker-compose.yml
version: '3.3'

services:
   db:
     image: beercan1989/arm-mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}

root@mtcpmhs:~/my_wordpress#
```

3. **To build the project, execute this command:**

```
# docker-compose up -d
```

It will take several minutes to complete.

```
root@mtcpmhs:~/my_wordpress# docker-compose up -d
Creating network "my_wordpress_default" with the default driver
Creating volume "my_wordpress_db_data" with default driver
Pulling db (beercan1989/arm-mysql:5.7)...
5.7: Pulling from beercan1989/arm-mysql
def78db8dd6e: Downloading [================================>                ]  25.49MB/38.19MB
69ae3b359f19: Download complete
65eaeb5577ed: Download complete
05712d47a012: Download complete
a9e8f88b7af9: Download complete
ebbcb8fe246c: Download complete
2c9d1d9de513: Download complete
28d350a17cc5: Downloading [============>                                     ]  19.25MB/75.83MB
fdc451b52cb9: Download complete
4ee4091a5f46: Download complete
da149758a458: Download complete
5c0052ed8d72: Download complete
01a19a8b5b9a: Download complete
```

```
6775825d6dc5: Pull complete
4c649dcaa656: Pull complete
4398312b35b2: Pull complete
Digest: sha256:25ae92cd273f1bc2f72e7d4120f68b5c17e4cf3bd7539fe0469614dac3bdc5dc
Status: Downloaded newer image for wordpress:latest
Creating my_wordpress_db_1 ... done
Creating my_wordpress_wordpress_1 ... done
root@mtcpmhs:~/my_wordpress#
```

When the project is built, you can start working with the Word Press.

Below you can see what changes are made in the system after the project has been built.

4. Check that the containers are up:

```
# docker ps
```

```
root@mtcpmhs:~/my_wordpress# docker ps
CONTAINER ID        IMAGE                   COMMAND                  CREATED             STATUS              PORTS                    NAMES
4403c452c6ab        wordpress:latest        "docker-entrypoint.s…"   3 minutes ago       Up 2 minutes        0.0.0.0:8000->80/tcp     my_wordpress_wordpress_1
c1ca7289bdea        beercan1989/arm-mysql:5.7   "/opt/docker-arm-mys…"   3 minutes ago       Up 3 minutes        3306/tcp                 my_wordpress_db_1
root@mtcpmhs:~/my_wordpress#
```

5. Check that the Docker images are present on the **Docker Images** page:

DOCKER IMAGES

Id	Tags	Size	Created	Details
sha256:a4c74df97920d7bc4...	wordpress:latest	411.1 MB	1/21/2021, 10:04:22 PM	
sha256:83203d49d0d151cfe...	beercan1989/arm-mysql:5.7	433.3 MB	9/19/2019, 8:30:33 PM	

2 records

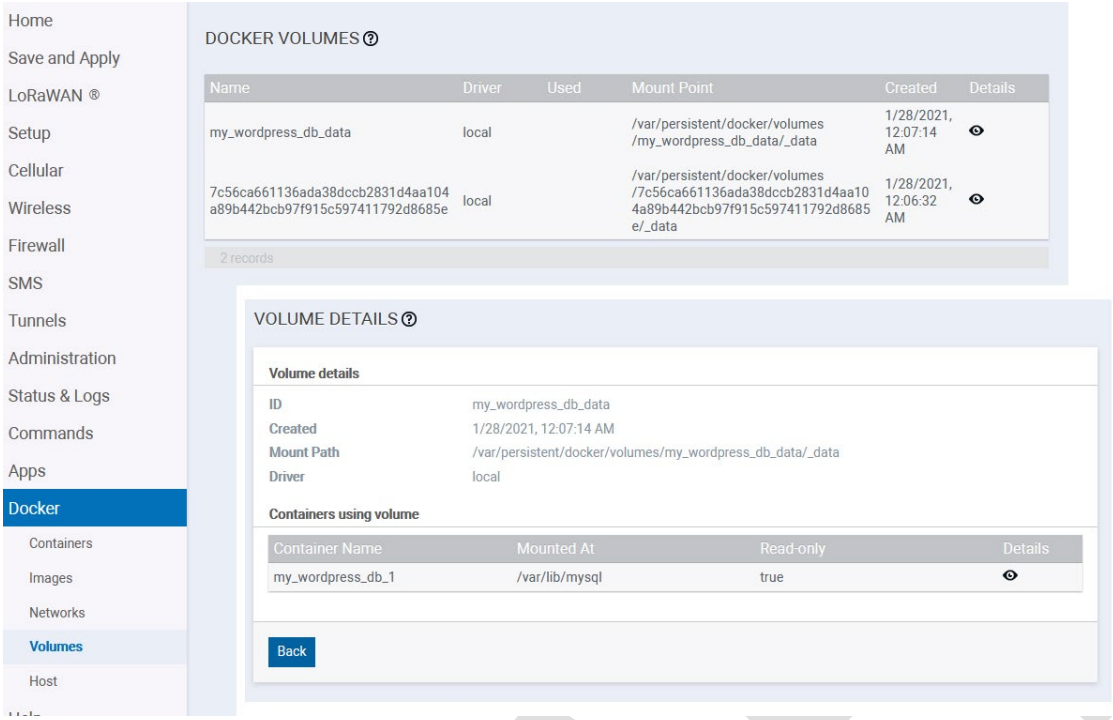
You can also see the containers on the **Docker Containers** page:

DOCKER CONTAINERS

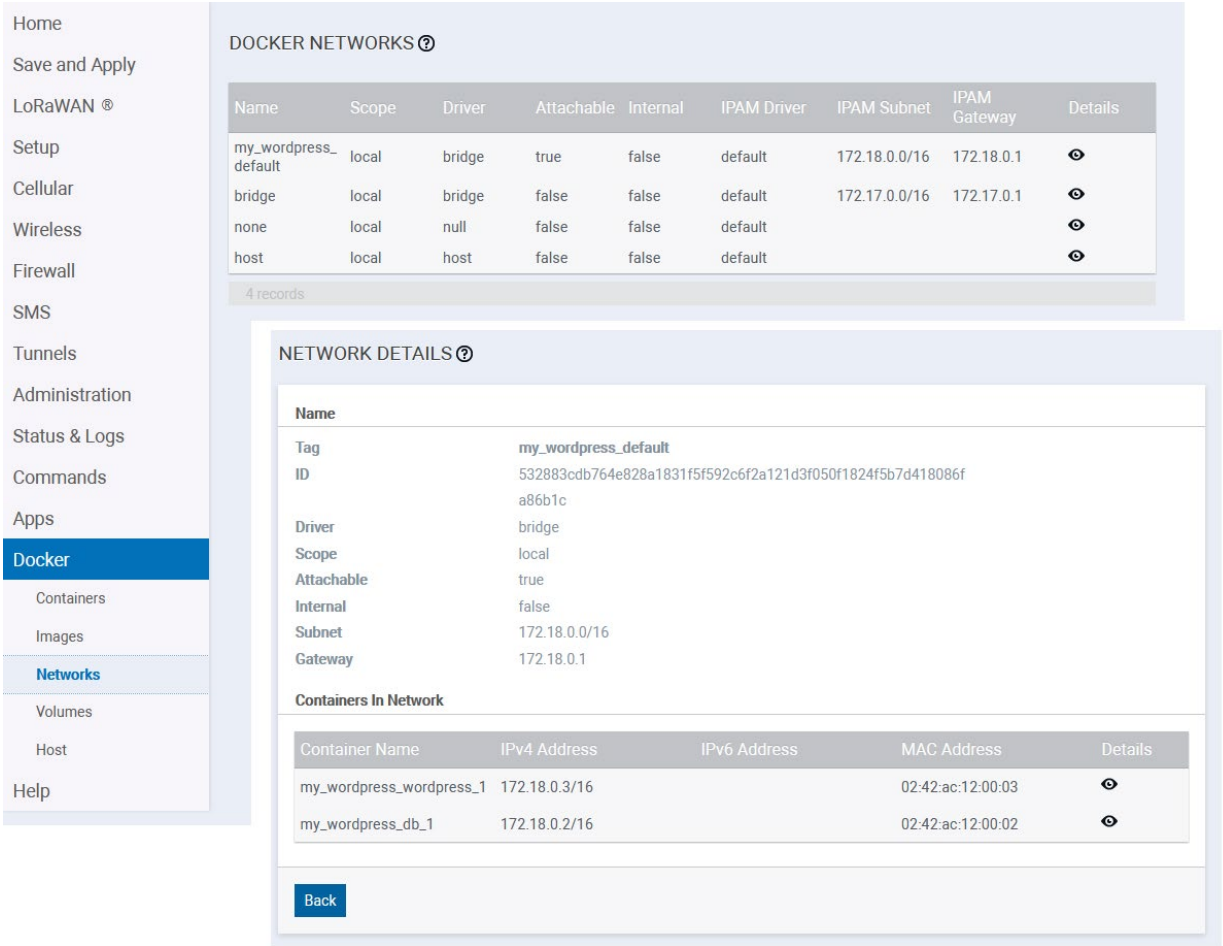
State	Name	Image	Created	IP Address	Published Ports	Details
running	my_wordpress_wo...	wordpress:latest	1/28/2021, 12:05:28 AM	172.18.0.3	8000:80/tcp	
running	my_wordpress_db_1	beercan1989/arm-...	1/28/2021, 12:04:43 AM	172.18.0.2		

2 records

You can also check the volume **Docker** uses on the **Docker Volumes** page:



You can also see the docker network interfaces that were created on the **Docker Networks** page:




```
# ifconfig -a | grep br-
```

```
root@mtcpmhs:~/my_wordpress# ifconfig -a | grep br-
br-532883cdb764 Link encap:Ethernet  HWaddr 02:42:9E:8C:D9:6B
```

```
# ifconfig br-532883cdb764
```

```
root@mtcpmhs:~/my_wordpress# ifconfig br-532883cdb764
br-532883cdb764 Link encap:Ethernet  HWaddr 02:42:9E:8C:D9:6B
          inet addr:172.18.0.1  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:628 errors:0 dropped:0 overruns:0 frame:0
          TX packets:728 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1042729 (1018.2 KiB)  TX bytes:400887 (391.4 KiB)
```

6. Docker adds the changes to the **nat** and **filter** firewall rules. To see the changes, execute the following commands:

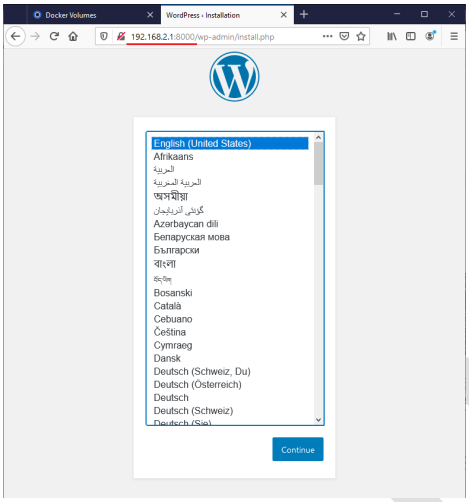
```
# iptables -t nat -S | grep br-
```

```
root@mtcpmhs:~/my_wordpress# iptables -t nat -S | grep br-
-A POSTROUTING -s 172.17.0.0/16 ! -o br-532883cdb764 -j MASQUERADE
-A DOCKER -i br-532883cdb764 -j RETURN
```

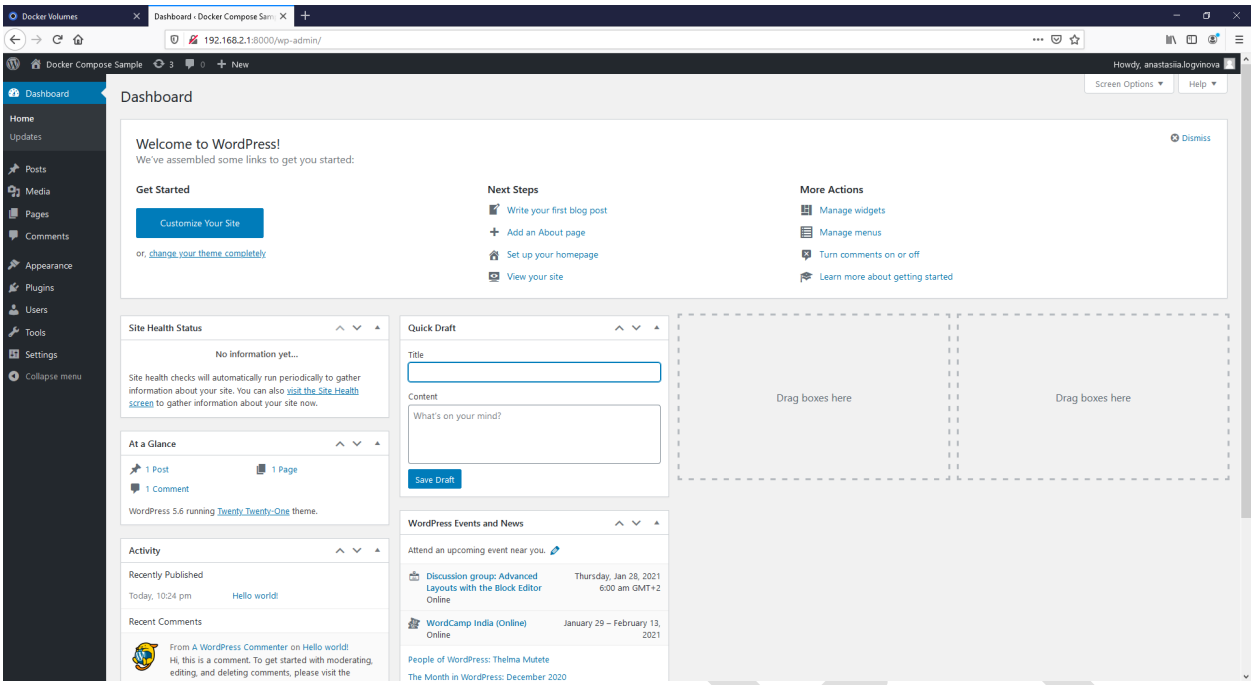
```
# iptables -t filter -S | grep br-
```

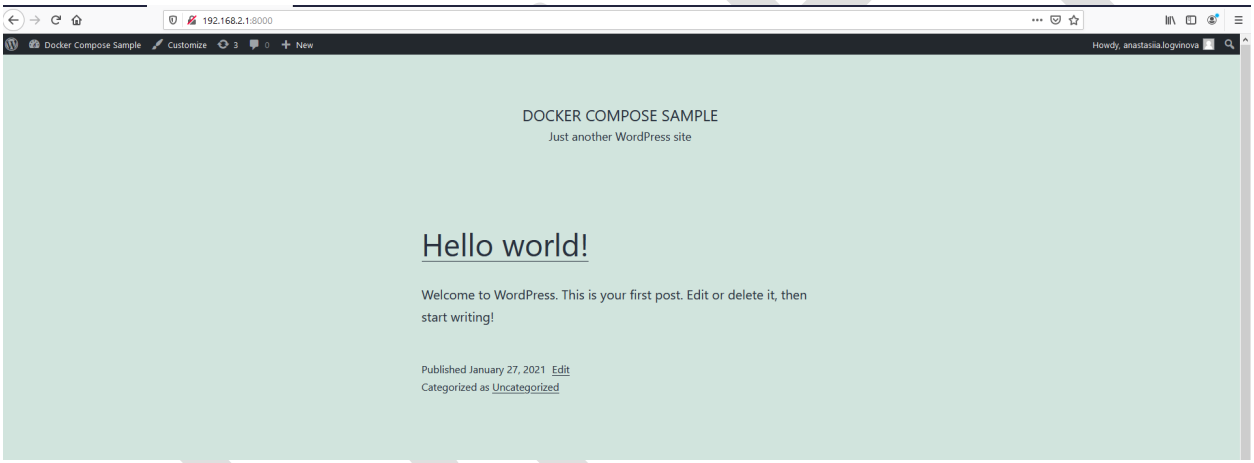
```
root@mtcpmhs:~/my_wordpress# iptables -t filter -S | grep br-
-A FORWARD -o br-532883cdb764 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o br-532883cdb764 -j DOCKER
-A FORWARD -i br-532883cdb764 ! -o br-532883cdb764 -j ACCEPT
-A FORWARD -i br-532883cdb764 -o br-532883cdb764 -j ACCEPT
-A DOCKER-ISOLATION-STAGE-1 -i br-532883cdb764 ! -o br-532883cdb764 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-2 -o br-532883cdb764 -j DROP
```

7. Go to http://192.168.1.85:8000/. The WordPress installation page should open:



8. Finish the WordPress installation and the application dashboard appears:


Docker Volumes
Dashboard ‹ Docker Compose Sam
192.168.2.1:8000/wp-admin/
Docker Compose Sample
New
Howdy, anastasiia.logvinova
Dashboard
Home
Updates
Posts
Media
Pages
Comments
Appearance
Plugins
Users
Tools
Settings
Collapse menu
Screen Options
Help
Dashboard
Welcome to WordPress!
We've assembled some links to get you started:
Dismiss
Get Started
Customize Your Site
or, change your theme completely
Next Steps
Write your first blog post
Add an About page
Set up your homepage
View your site
More Actions
Manage widgets
Manage menus
Turn comments on or off
Learn more about getting started
Site Health Status
No information yet...
Site health checks will automatically run periodically to gather information about your site. You can also visit the Site Health screen to gather information about your site now.
Quick Draft
Title
Content
What's on your mind?
Save Draft
Drag boxes here
Drag boxes here
At a Glance
1 Post
1 Page
1 Comment
WordPress 5.6 running Twenty Twenty-One theme.
WordPress Events and News
Attend an upcoming event near you.
Discussion group: Advanced Layouts with the Block Editor
Online
Thursday, Jan 28, 2021
6:00 am GMT+2
WordCamp India (Online)
Online
January 29 – February 13, 2021
People of WordPress: Thelma Mutete
The Month in WordPress: December 2020
Activity
Recently Published
Today, 10:24 pm
Hello world!
Recent Comments
From A WordPress Commenter on Hello world!
Hi, this is a comment. To get started with moderating, editing, and deleting comments, please visit the

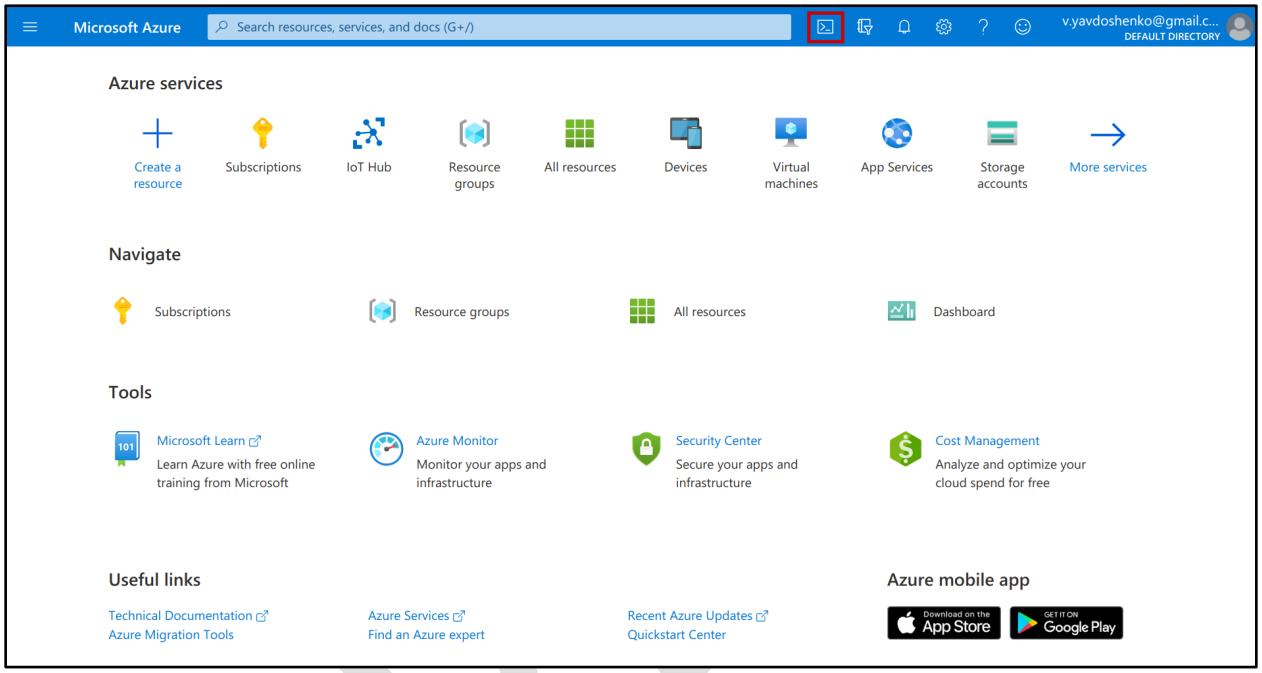

192.168.2.1:8000
Docker Compose Sample
Customize
New
Howdy, anastasiia.logvinova
DOCKER COMPOSE SAMPLE
Just another WordPress site
Hello world!
Welcome to WordPress. This is your first post. Edit or delete it, then start writing!
Published January 27, 2021 Edit
Categorized as Uncategorized

Microsoft Azure IoT Edge example

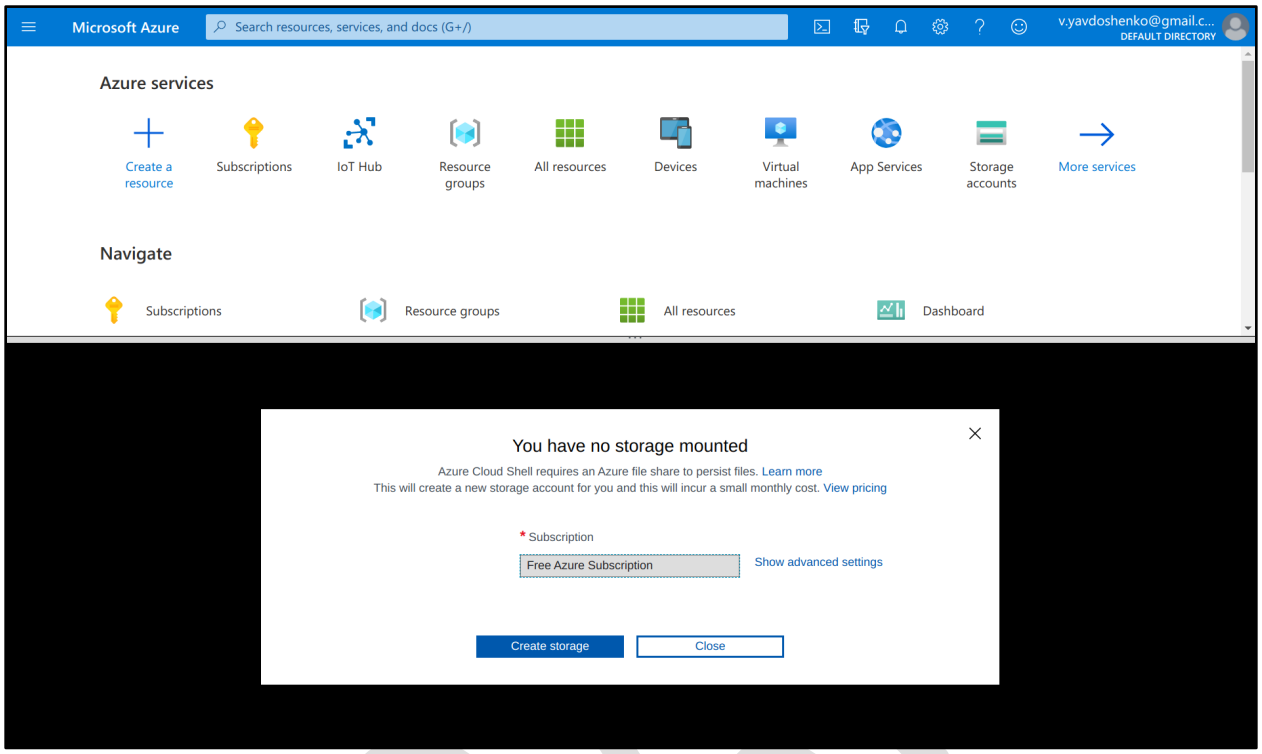
Important: Before proceeding, you must create a Microsoft Account, an Azure subscription, and storage to execute the steps below.

1. Log into: https://portal.azure.com/#home

2. Go to the console.



3. Mount storage (the system suggests to do that automatically)



4. In the **Azure Bash console,** execute the commands below:

Note: Use your own unique names for iotEdgeRes, iotEdgeHubUniqueName, iotEdgeDevice to avoid names collisions.

```
$ az extension add --name azure-iot
$ az group create --name iotEdgeRes --location westus2
$ az iot hub create --resource-group iotEdgeRes --name iotEdgeHubUniqueName --sku F1 --partition-count 2
$ az iot hub device-identity create --hub-name iotEdgeHubUniqueName --device-id iotEdgeDevice --edge-enabled
$ az iot hub device-identity connection-string show --device-id iotEdgeDevice --hub-name iotEdgeHubUniqueName
```

The last command shows **the connection string** required to run Docker on the device.

Here is an example of the Azure console output:

```
$ az extension add --name azure-iot
```

```
$ az group create --name iotEdgeRes --location westus2
```

```
Bash
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

anastasiia@Azure:~$ az extension add --name azure-iot
anastasiia@Azure:~$ az group create --name iotEdgeRes --location westus2
{
  "id": "/subscriptions/dbb25f92-ddb7-4690-aa9e-5cf639514d21/resourceGroups/iotEdgeRes",
  "location": "westus2",
  "managedBy": null,
  "name": "iotEdgeRes",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
anastasiia@Azure:~$
```

```
$ az iot hub create --resource-group iotEdgeRes --name iotEdgeHubTest1 --sku F1 --partition-count 2
```

```
anastasiia@Azure:~$ az iot hub create --resource-group iotEdgeRes --name iotEdgeHubTest1 --sku F1 --partition-count 2

{- Finished ..
  "etag": "AAAACQF3zuQ=",
  "id": "/subscriptions/dbb25f92-ddb7-4690-aa9e-5cf639514d21/resourceGroups/iotEdgeRes/providers/Microsoft.Devices/IotHubs/iotEdgeHubTest1",
  "identity": {
    "type": "None"
  },
  "location": "westus2",
  "name": "iotEdgeHubTest1",
  "properties": {
    "authorizationPolicies": null,
    "cloudToDevice": {
      "defaultTtlAsIso8601": "1:00:00",
      "feedback": {
        "lockDurationAsIso8601": "0:00:05",
        "maxDeliveryCount": 10,
        "ttlAsIso8601": "1:00:00"
      },
      "maxDeliveryCount": 10
    },
    "comments": null,
    "enableFileUploadNotifications": false,
    "eventHubEndpoints": {
      "events": {
        "endpoint": "sb://ihsuprodmwhres020dednamespace.servicebus.windows.net/",
        "partitionCount": 2,
        "partitionIds": [
          "0",
          "1"
        ],
        "path": "iothub-ehub-iotedgehub-7708039-625813a5b5",
        "retentionTimeInDays": 1
      }
    },
    "features": "None",
    "hostName": "iotEdgeHubTest1.azure-devices.net",
    "ipFilterRules": [],
    "locations": [
      {
        "location": "West US 2",
        "role": "primary"
      },
      {
        "location": "West Central US",
        "role": "secondary"
      }
    ],
    "messagingEndpoints": {
      "fileNotifications": {
        "lockDurationAsIso8601": "0:01:00",
        "maxDeliveryCount": 10,
        "ttlAsIso8601": "1:00:00"
      }
    },
    "minTlsVersion": null,
    "privateEndpointConnections": null,
    "provisioningState": "Succeeded",
    "publicNetworkAccess": null,
    "routing": {
      "endpoints": {
        "eventHubs": [],
        "serviceBusQueues": [],
        "serviceBusTopics": [],
        "storageContainers": []
      },
      "enrichments": null,
      "fallbackRoute": {
        "condition": "true",
        "endpointNames": [
          "events"
        ],
        "isEnabled": true,
        "name": "$fallback"
      },
      "routes": []
    },
    "state": "Active",
    "storageEndpoints": {
      "$default": {
        "authenticationType": null,
        "connectionString": "",
        "containerName": "",
        "sasTtlAsIso8601": "1:00:00"
      }
    }
  },
  "resourcegroup": "iotEdgeRes",
  "sku": {
    "capacity": 1,
    "name": "F1",
    "tier": "Free"
  },
  "subscriptionid": "dbb25f92-ddb7-4690-aa9e-5cf639514d21",
  "tags": {},
  "type": "Microsoft.Devices/IotHubs"
}
anastasiia@Azure:~$
```

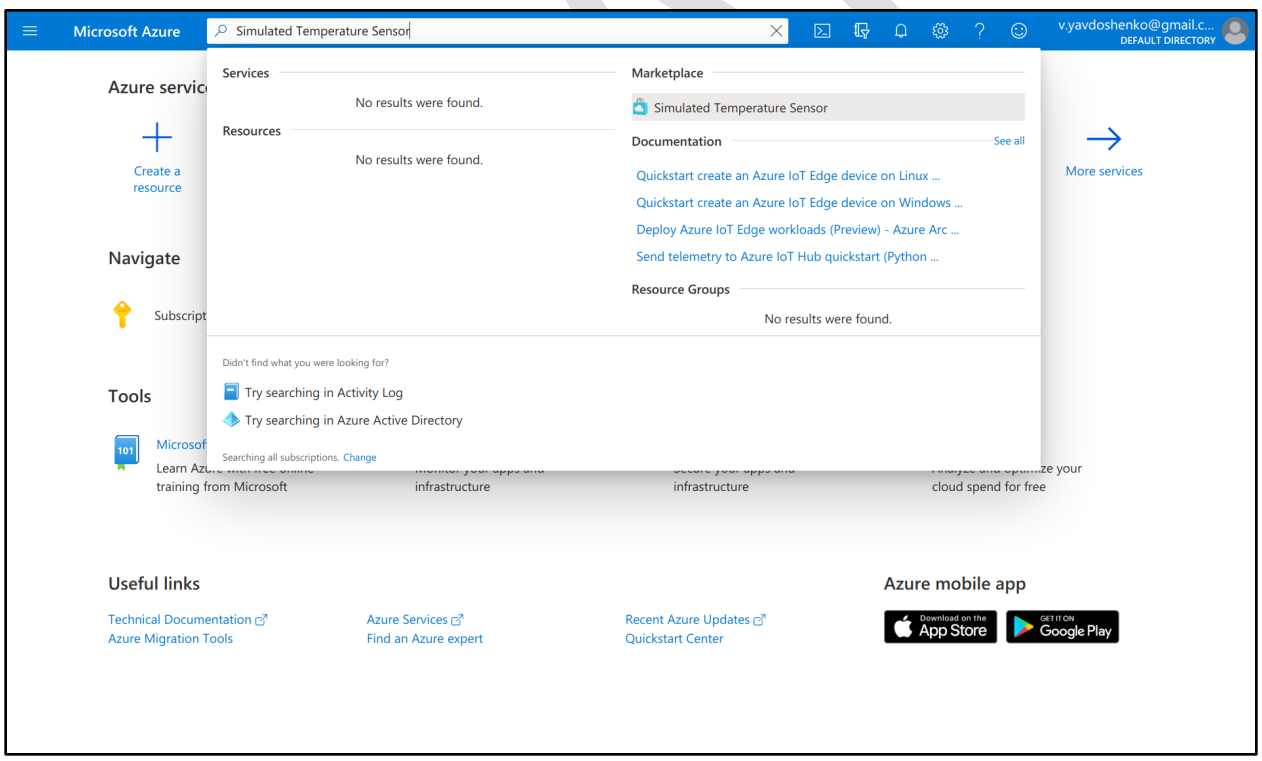
```
$ az iot hub device-identity create --hub-name iotEdgeHubTest1 --device-id
iotEdgeDevice --edge-enabled
```

```
anastasiia@Azure:~$ az iot hub device-identity create --hub-name iotEdgeHubTest1 --device-id iotEdgeDevice --edge-enabled
{
  "authentication": {
    "symmetricKey": {
      "primaryKey": "NHt8OGLp4UGpzX27Pl6+TkBa5WHBaVIOMl+egE+VJ6Y=",
      "secondaryKey": "J7emJs+RwIBRdxvSjOWk32aucDGmubYsAeVK/05BgFc="
    },
    "type": "sas",
    "x509Thumbprint": {
      "primaryThumbprint": null,
      "secondaryThumbprint": null
    }
  },
  "capabilities": {
    "iotEdge": true
  },
  "cloudToDeviceMessageCount": 0,
  "connectionState": "Disconnected",
  "connectionStateUpdatedTime": "0001-01-01T00:00:00",
  "deviceId": "iotEdgeDevice",
  "deviceScope": "ms-azure-iot-edge://iotEdgeDevice-637475275181753756",
  "etag": "NjczODE0NzU1",
  "generationId": "637475275181753756",
  "lastActivityTime": "0001-01-01T00:00:00",
  "parentScopes": [],
  "status": "enabled",
  "statusReason": null,
  "statusUpdatedTime": "0001-01-01T00:00:00"
}
anastasiia@Azure:~$
```

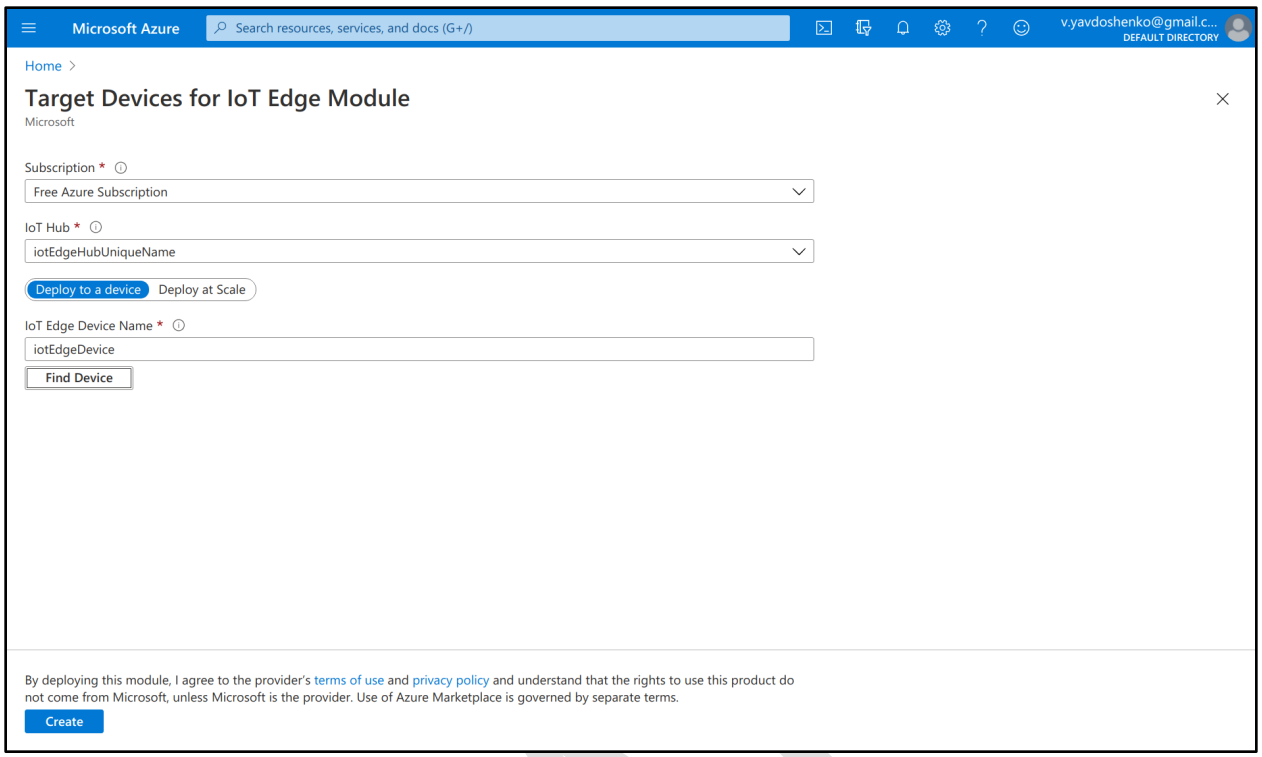
```
$ az iot hub device-identity connection-string show --device-id iotEdgeDevice --hub-name iotEdgeHubTest1
```

```
anastasiia@Azure:~$ az iot hub device-identity connection-string show --device-id iotEdgeDevice --hub-name iotEdgeHubTest1
{
  "connectionString": "HostName=iotEdgeHubTest1.azure-devices.net;DeviceId=iotEdgeDevice;SharedAccessKey=NHt8OGLp4UGpzX27Pl6+TkBa5WHBaVIOMl+egE+VJ6Y="
}
anastasiia@Azure:~$
```

5. Deploy Simulated Temperature Sensor as end device tied to your gateway device.



6. Set manually **iotEdgeDevice** or use Find Device. Leave all settings as default. Click **Create.**



7. Run Docker on the device.

For this example, we use a Docker project developed for your device and testing purposes. This application is stored on the Docker portal: **yavdoshenko/iotedge:arm32v7** (**NOTE:** these details will change soon.)

8. Execute the command:

```
sudo docker run -it -d --rm --privileged -e connectionString='connection string'
yavdoshenko/iotedge:arm32v7
```

where

- **'connection string'** is a value that is retrieved from Azure
- **yavdoshenko/iotedge:arm32v7** is the application that is stored on Docker

Note: The system will start downloading the application. It will take up to 10 minutes to complete.

```
docker run -it -d --rm --privileged -e connectionString='HostName=iotEdgeHubTest1.azure-
devices.net;DeviceId=iotEdgeDevice;SharedAccessKey=NHt8OGLp4UGpzX27Pl6+TkBa5WHBaVIOMl+egE+VJ6Y
=' yavdoshenko/iotedge:arm32v7
```

```
admin@mtcpmhs:~$ sudo -s
Password:
root@mtcpmhs:/var/config/home/admin# docker run -it -d --rm --privileged -e connectionString='HostName=iotEdgeHubTest1.azure-devices.net;DeviceId=iotEdgeDevice;SharedAccessKey=NHt8OGLp4UGpzX27P16+TkBa5WHBaVIOM
1+egE+VJ6Y=' yavdoshenko/iotedge:arm32v7
Unable to find image 'yavdoshenko/iotedge:arm32v7' locally
arm32v7: Pulling from yavdoshenko/iotedge
74cafcd4ef02: Pull complete
b11c6b5e0a0a: Pull complete
92494e4cae3e: Pull complete
b20432c553ae: Pull complete
e42c2bc03c1b: Pull complete
9979ff0656bf: Pull complete
2e716e52c45c: Pull complete
fe26abc2cde7: Pull complete
930dacb08ca6: Pull complete
4d60ad284572: Pull complete
a400bdab31bd: Pull complete
f7f079e6a112: Pull complete
662c905eac69: Pull complete
9a7f34497413: Pull complete
4913d3f73e43: Pull complete
Digest: sha256:5416442d11463a849659ac0501de7c53c4a9b7c95793d407be2f205c5b6139d2
Status: Downloaded newer image for yavdoshenko/iotedge:arm32v7
0401c0e577b7c6a90035bd031e111c919c775947f570b5009733fa10f3dd15d7
root@mtcpmhs:/var/config/home/admin#
```

9. Make sure Docker is running:

 # docker ps

```
root@mtcpmhs:/var/config/home/admin# docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                          NAMES
0401c0e577b7        yavdoshenko/iotedge:arm32v7    "bash edge_dind.sh"      About a minute ago  Up About a minute   2375/tcp, 15580-15581/tcp      hardcore_poitras
```

10. Open the shell inside the container by executing the command below:
 # docker exec -it <CONTAINER ID> /bin/bash
Replace <CONTAINER ID> with your CONTAINER ID.

```
docker exec -it 0401c0e577b7 /bin/bash
```

```
root@mtcpmhs:/var/config/home/admin# docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                          NAMES
0401c0e577b7        yavdoshenko/iotedge:arm32v7    "bash edge_dind.sh"      About a minute ago  Up About a minute   2375/tcp, 15580-15581/tcp      hardcore_poitras
root@mtcpmhs:/var/config/home/admin# docker exec -it 0401c0e577b7 /bin/bash
root@0401c0e577b7:/#
```

It will take **at least 10-15 minutes** to start all the containers.

11. Check that all containers are running by executing the command:

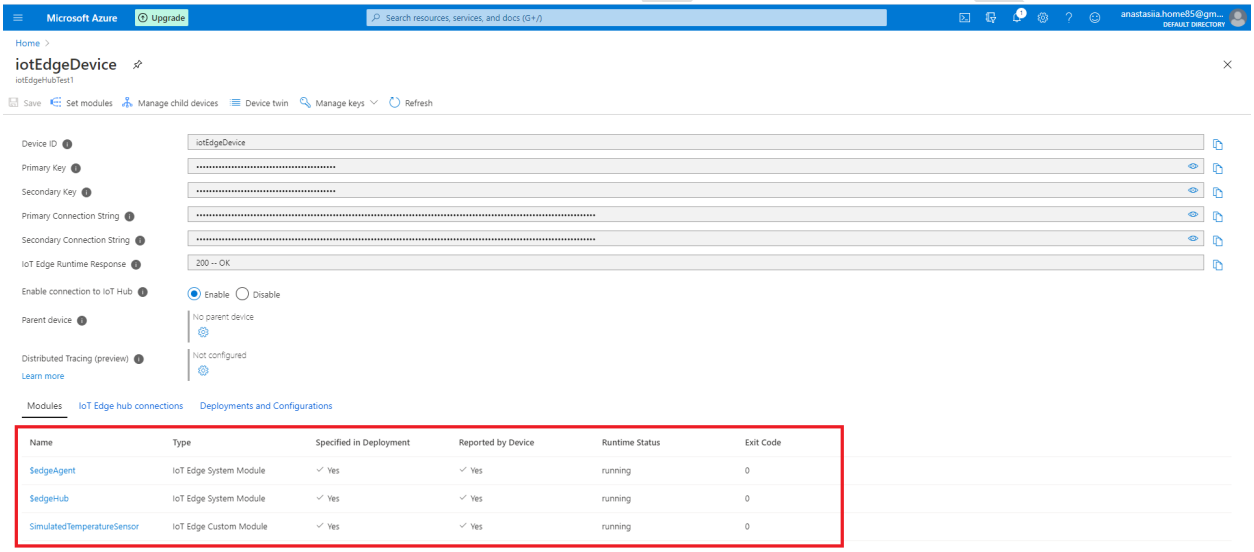
docker ps

```
root@0401c0e577b7:/# docker ps
CONTAINER ID        IMAGE                                                    COMMAND                  CREATED             STATUS              PORTS                                                                                    NAMES
4f5315475add        mcr.microsoft.com/azureiotedge-hub:1.0                   "/bin/sh -c 'echo \"$…"  58 minutes ago      Up 58 minutes       0.0.0.0:443->443/tcp, 0.0.0.0:5671->5671/tcp, 0.0.0.0:8883->8883/tcp     edgeHub
837049a5f8f6        mcr.microsoft.com/azureiotedge-simulated-temperature-sensor:1.0  "/bin/sh -c 'echo \"$…"  About an hour ago   Up 59 minutes                                                                                SimulatedTempe
ratureSensor
7cf6366649ec        mcr.microsoft.com/azureiotedge-agent:1.0                 "/bin/sh -c 'exec /a…"   About an hour ago   Up About an hour                                                                             edgeAgent
root@0401c0e577b7:/#
```

12. Execute the command **iotedge logs SimulatedTemperatureSensor** in the container.

```
root@0401c0e577b7:/# iotedge logs SimulatedTemperatureSensor
[2021-02-01 11:50:07 +00:00]: Starting Module
SimulatedTemperatureSensor Main() started.
Initializing simulated temperature sensor to send 500 messages, at an interval of 5 seconds.
To change this, set the environment variable MessageCount to the number of messages that should be sent (set it to -1 to send unlimited me
ssages).
[Information]: Trying to initialize module client using transport type [Amqp_Tcp_Only].
[Information]: Successfully initialized module client of transport type [Amqp_Tcp_Only].
        02/01/2021 11:53:24> Sending message: 1, Body: [{"machine":{"temperature":21.143373745211107,"pressure":1.0163337178088603},"ambie
nt":{"temperature":20.544170913307077,"humidity":25},"timeCreated":"2021-02-01T11:53:24.0544942Z"}]
        02/01/2021 11:53:31> Sending message: 2, Body: [{"machine":{"temperature":22.138618737756563,"pressure":1.1297160587317603},"ambie
nt":{"temperature":20.63045244064669,"humidity":24},"timeCreated":"2021-02-01T11:53:31.4604466Z"}]
        02/01/2021 11:53:37> Sending message: 3, Body: [{"machine":{"temperature":22.466279297236483,"pressure":1.1670444769003587},"ambie
nt":{"temperature":20.893090744220228,"humidity":24},"timeCreated":"2021-02-01T11:53:36.9891627Z"}]
        02/01/2021 11:53:42> Sending message: 4, Body: [{"machine":{"temperature":22.49040239280574,"pressure":1.1697926776614136},"ambien
t":{"temperature":21.414174670313567,"humidity":24},"timeCreated":"2021-02-01T11:53:42.3307136Z"}]
        02/01/2021 11:53:47> Sending message: 5, Body: [{"machine":{"temperature":22.424605501198492,"pressure":1.1622968292504612},"ambie
nt":{"temperature":20.779154610484444,"humidity":24},"timeCreated":"2021-02-01T11:53:47.8040022Z"}]
        02/01/2021 11:53:53> Sending message: 6, Body: [{"machine":{"temperature":23.508785927206645,"pressure":1.2858110549982253},"ambie
nt":{"temperature":20.588870028540896,"humidity":24},"timeCreated":"2021-02-01T11:53:53.1512012Z"}]
        02/01/2021 11:53:58> Sending message: 7, Body: [{"machine":{"temperature":24.52302882879648,"pressure":1.4013577146730167},"ambien
t":{"temperature":20.51493886998619,"humidity":25},"timeCreated":"2021-02-01T11:53:58.4456857Z"}]
        02/01/2021 11:54:03> Sending message: 8, Body: [{"machine":{"temperature":25.511893862398296,"pressure":1.5140132248301854},"ambie
nt":{"temperature":20.870424664286162,"humidity":24},"timeCreated":"2021-02-01T11:54:03.7595064Z"}]
        02/01/2021 11:54:08> Sending message: 9, Body: [{"machine":{"temperature":26.4133082510267,"pressure":1.6167060032815226},"ambient
":{"temperature":21.25442565966138,"humidity":26},"timeCreated":"2021-02-01T11:54:08.9483019Z"}]
        02/01/2021 11:54:14> Sending message: 10, Body: [{"machine":{"temperature":26.350697397650546,"pressure":1.609573121251328},"ambie
```

13. Go to the Azure portal and check the state of your device.
    The Runtime Status should say running.



Amazon AWS IoT Greengrass example using Lambda

This example requires an AWS account. You first must configure AWS and IoT Greengrass. Please refer to the Greengrass developer manual for more details on Greengrass:

https://docs.aws.amazon.com/greengrass/latest/developerguide/gg-gs.html

AWS Lambda is an event-driven, serverless computing platform provided by AWS. To run on an AWS IoT Greengrass core, a Python Lambda function requires the AWS IoT Greengrass Core SDK for Python.

NOTE: *Before you create and deploy your Lambda function, make sure you understand your code design and the specific costs that will result from it. Amazon will charge you every time the Lambda function is called.*

1. Sign into the AWS Console:
   https://console.aws.amazon.com/console/home
2. Open the IoT Greengrass page:
   https://us-east-2.console.aws.amazon.com/greengrass/home
3. Create a group:
   https://us-east-2.console.aws.amazon.com/iot/home?#/greengrass/create/group
4. Use the default creation settings.
5. Add the name. For example, iotGroup.
6. Set the core function name. For example, iotGroup_Core.
7. Push the button: Create Group and Core
8. ***Important note:*** Please download the keys on the next page. This is a tar.gz archive.
9. Be careful with the platform choice. Refer to the Supported platforms and requirements page:
   https://docs.aws.amazon.com/greengrass/v1/developerguide/what-is-gg.html#gg-platforms

Also, see the AWS IoT Greengrass downloads page
https://docs.aws.amazon.com/console/greengrass/gg-core-download

Now, we can prepare the device:

1. Copy the downloaded archive to the device:
   ```
   $ scp ./5853c3300f-setup.tar.gz admin@192.168.1.85:/home/admin/
   ```

2. On the device, enter the following commands:
   ```
   # mkdir /greengrass
   # tar xvzf /home/admin/5853c3300f-setup.tar.gz -C /greengrass/
   # cd /greengrass/certs
   # curl -o root.ca.pem https://www.amazontrust.com/repository/AmazonRootCA1.pem
   ```

3. Pull the official AWS Greengrass docker image onto the device:
   ```
   # docker pull amazon/aws-iot-greengrass:1.10.2-alpine-armv7l
   ```

4. To avoid hardlink/softlink protection error, do the following on the device:
   ```
   # sysctl fs.protected_hardlinks=1
   # sysctl fs.protected_symlinks=1
   ```

5. Start the docker container (to put it in the background by adding -d key after run):

```
# docker run --rm --init -it --name aws-iot-greengrass \
--entrypoint /greengrass-entrypoint.sh \
-v /greengrass/certs:/greengrass/certs \
-v /greengrass/config:/greengrass/config \
-p 8883:8883 amazon/aws-iot-greengrass:1.10.2-alpine-armv7l
```

```
root@mtcdt3hs:~# docker run --rm --init -it --name aws-iot-greengrass \
> --entrypoint /greengrass-entrypoint.sh \
> -v /greengrass/certs:/greengrass/certs \
> -v /greengrass/config:/greengrass/config \
> -p 8883:8883 amazon/aws-iot-greengrass:1.10.2-alpine-armv7l
Unable to find image 'amazon/aws-iot-greengrass:1.10.2-alpine-armv7l' locally
1.10.2-alpine-armv7l: Pulling from amazon/aws-iot-greengrass
ad20c9452290: Pull complete
ea9b170b8dae: Pull complete
d104e3f776ba: Pull complete
e6ded7e03a63: Pull complete
2f3c3e1b3711: Pull complete
894ff470a364: Pull complete
Digest: sha256:b25da6b5f2bcd676402e59a74099cfc5ad9d5b0cc58504f748fd8981ff44a304
Status: Downloaded newer image for amazon/aws-iot-greengrass:1.10.2-alpine-armv7l
[25374.621522] docker0: port 1(veth8fbf09a) entered blocking state
[25374.639200] docker0: port 1(veth8fbf09a) entered disabled state
[25374.663104] device veth8fbf09a entered promiscuous mode
[25376.931605] eth0: renamed from veth34dc736
[25376.965967] docker0: port 1(veth8fbf09a) entered blocking state
[25376.971946] docker0: port 1(veth8fbf09a) entered forwarding state
grep: /greengrass/ggc/deployment/group/group.json: No such file or directory
Setting up greengrass daemon
Validating hardlink/softlink protection
Waiting for up to 1m10s for Daemon to start

Greengrass successfully started with PID: 12
```

Important note: Use version 1.10.2 for this case. Do not use the latest version, 1.11.0. This connection displays on the AWS dashboard:
https://us-east-2.console.aws.amazon.com/iot/home?#/dashboard
It may take a minute or two.

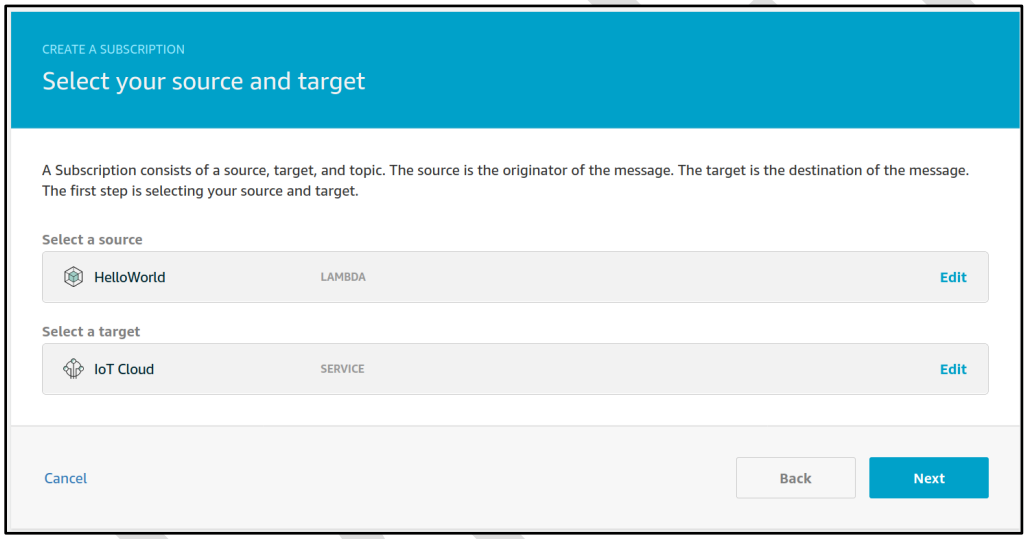
The next step is adding the lambda function on AWS IoT Greengrass.

1. Check the AWS official repository:
   https://github.com/aws/aws-greengrass-core-sdk-python
2. Try this example:
   https://github.com/aws/aws-greengrass-core-sdk-python/blob/master/examples/HelloWorld/greengrassHelloWorld.py
3. The latest release is 1.6.0. Download the archive:
   https://github.com/aws/aws-greengrass-core-sdk-python/archive/v1.6.0.tar.gz
4. Unpack this archive. The above example placed in the example folder.
5. Compress the example file and greengrasssdk folder (this is SDK) into one zip archive.
   ```
   $ zip -r lambda.zip greengrasssdk greengrassHelloWorld.py
   ```
6. Open the Lambda console:
   https://us-east-2.console.aws.amazon.com/lambda/home?/functions
7. Create a function. Author from scratch.
   https://us-east-2.console.aws.amazon.com/lambda/home?/functions#/create/function
8. Set name: HelloWorld

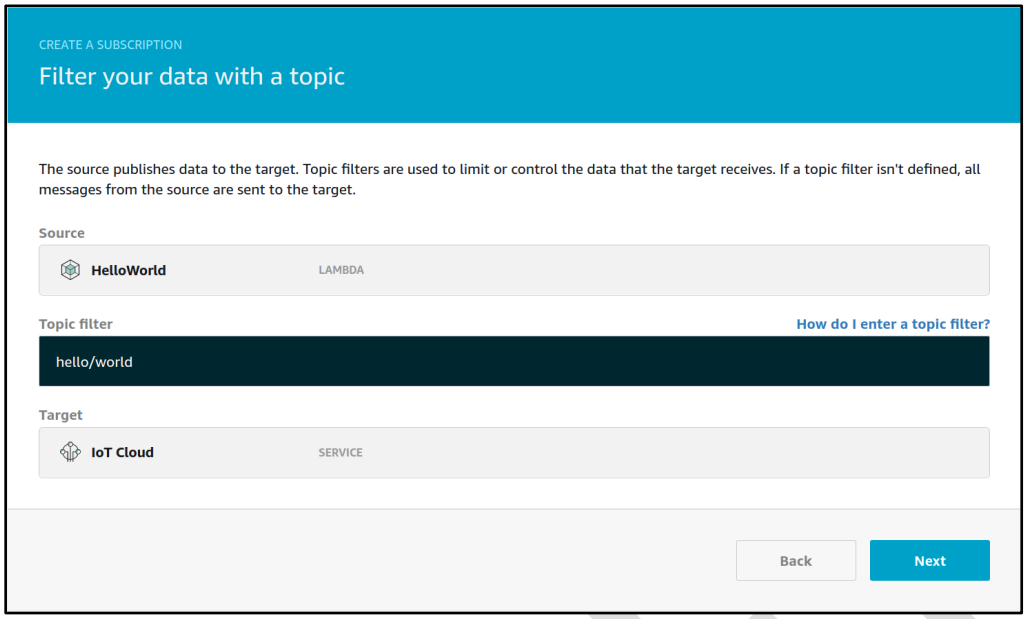
9. Select ***Python 2.7***
10. Push the button: Create Function.
11. Function code -> Actions -> Upload ***zip*** file.
12. Edit handler. Set ***greengrassHelloWorld.function_handler***
13. Actions -> Publish new version.

The next phase involves adding the Lambda function to Greengrass.

1. Select created iotGroup on the Greengrass page. https://us-east-2.console.aws.amazon.com/iot/home#/greengrass/grouphub
2. Lambdas -> Add Lambda -> Use existing Lambda
3. Select HelloWorld and click Next button.
4. Select Version 1 and click Finish.
5. Open Subscriptions page for the iotGroup.
6. Add subscription.
7. Select source and target. Then click Next.



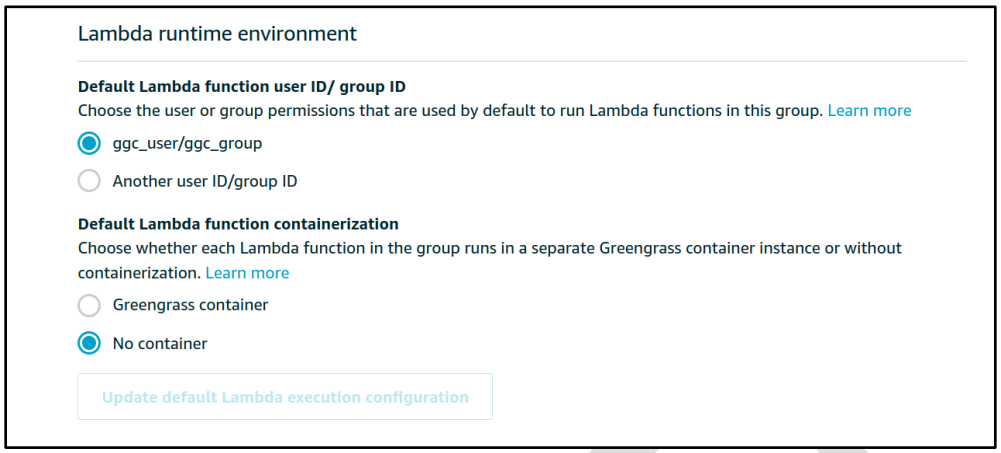
8. Set topic: ***hello/world*** **.** Click Next.



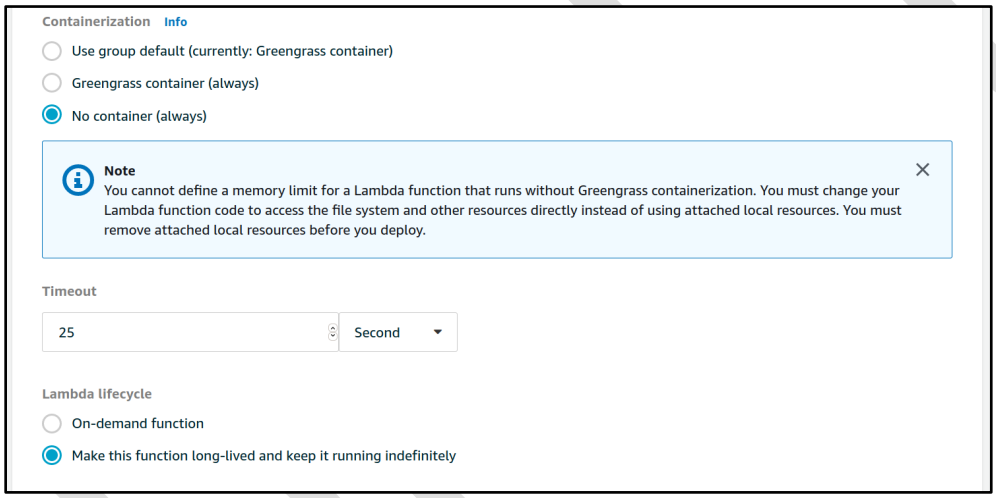
9. Click Finish.

The next phase is deployment. Make sure your Docker container is ready and started.

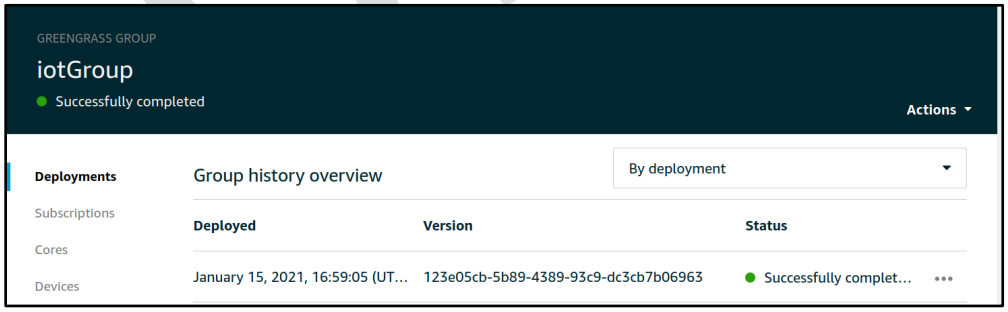
1. Open Settings page for the iotGroup. Change Default Lambda function containerization to No container.



2. Open Lambdas page for the iotGroup. Select your Lambda function. Choose Edit configuration via ellipsis. Change Containerization option to No container. Set timeout and Lambda lifecycle.



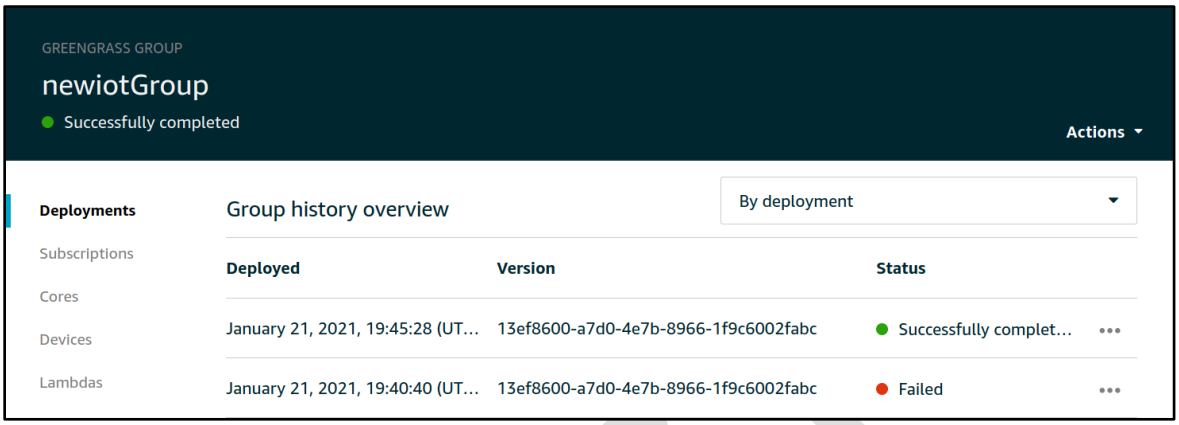
3. Open Deployment page for the iotGroup.
4. Actions -> Deploy. (Use Automatic detection)



5. Return to the device and check the container log.

```
# docker exec aws-iot-greengrass cat /greengrass/ggc/var/log/system/runtime.log
```

NOTE: You may experience a first-time deployment error:



A first-time deployment error can be due to a timeout error in /greengrass/ggc/var/log/system/runtime.log. In this case, you need to redeploy. Make sure to reset deployment from the Actions menu. Reset finishes successfully after the docker container is started.

Finally, test your deployment:

https://us-east-2.console.aws.amazon.com/iot/home?region=us-east-2#/test

1. Set Subscription topic to ***hello/world***
2. Set MQTT payload display to Display payloads as strings (more accurate)
3. Choose Subscribe to topic.