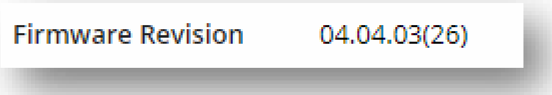
Dockerized Python application on Wago Controller

1. Hardware/Software in use:

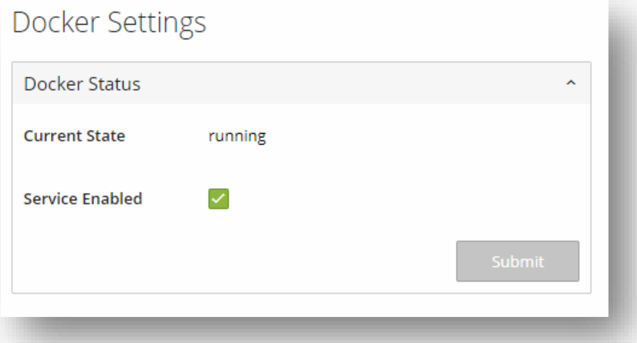
- Wago Edge Controller (752-8303/8000-002)
- FW 04.04.03(23)
- Docker version 20.10.21
- Python version 3.7

Note: For most of our controllers, we use an armv7 architecture. Ensure that the docker image you're pulling from supports this architecture to work on our controllers.



Firmware Revision 04.04.03(26)

Ensure docker is running via the WBM of the controller

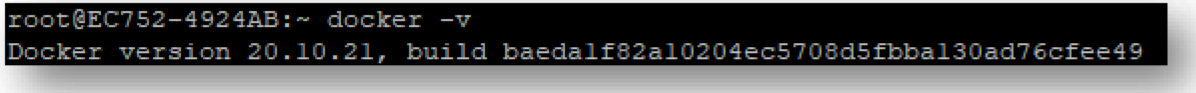


Docker Settings

Docker Status

Current State running

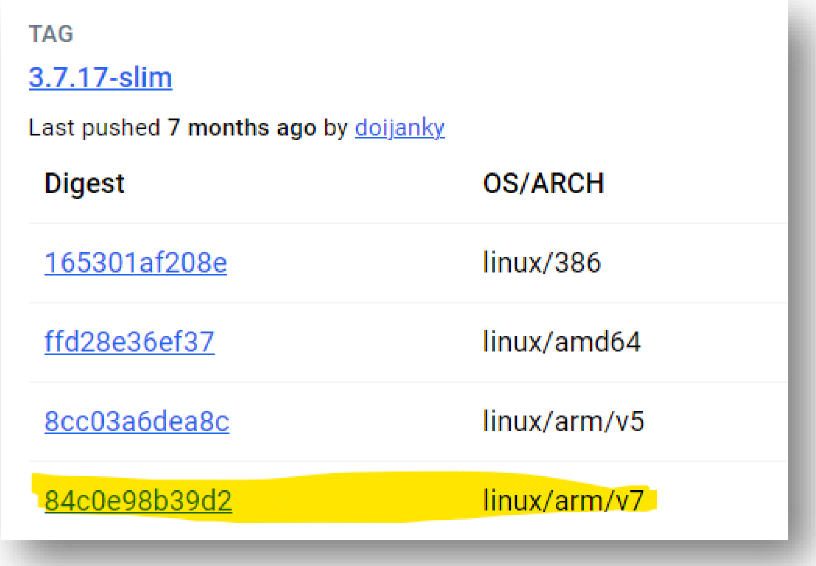
Service Enabled ☑



```
root@EC752-4924AB:~ docker -v
Docker version 20.10.21, build baeda1f82a10204ec5708d5fbba130ad76cfee49
```

Many of our Wago controllers have docker built in and are capable of running containers.

This demo just shows the hardware that I've used for testing.



TAG

3.7.17-slim

Last pushed **7 months ago** by doijanky

Digest	OS/ARCH
165301af208e	linux/386
ffd28e36ef37	linux/amd64
8cc03a6dea8c	linux/arm/v5
84c0e98b39d2	linux/arm/v7

Dockerized Python application on Wago Controller

2. Dockerfile and Python program

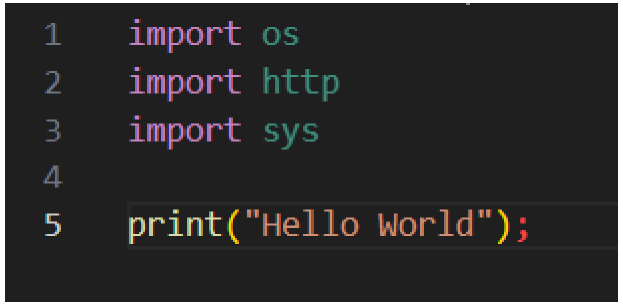
To dockerize your python application, you'll need to have:

1. Python application
2. Dockerfile

- For more information on Dockerfiles:

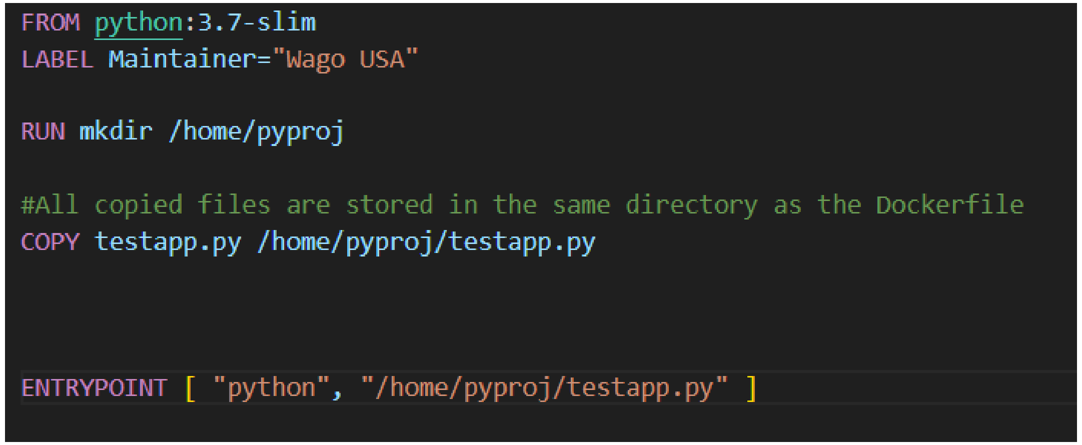
  https://docs.docker.com/reference/dockerfile/

testapp.py

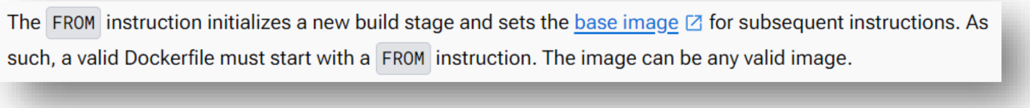


```
import os
import http
import sys

print("Hello World");
```

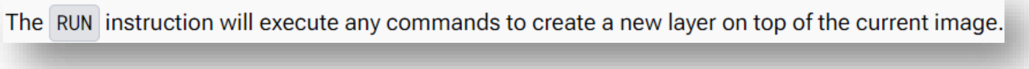
Dockerfile



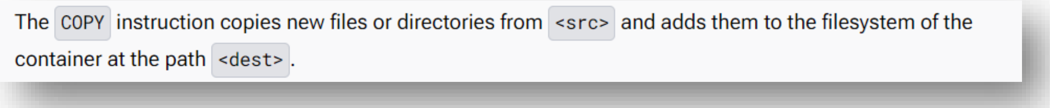
```
FROM python:3.7-slim
LABEL Maintainer="Wago USA"

RUN mkdir /home/pyproj

#All copied files are stored in the same directory as the Dockerfile
COPY testapp.py /home/pyproj/testapp.py



ENTRYPOINT [ "python", "/home/pyproj/testapp.py" ]
```



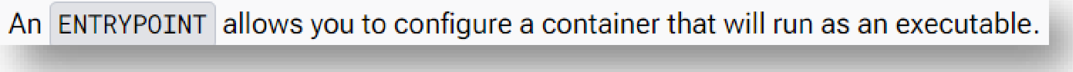
The `FROM` instruction initializes a new build stage and sets the base image for subsequent instructions. As such, a valid Dockerfile must start with a `FROM` instruction. The image can be any valid image.



The `RUN` instruction will execute any commands to create a new layer on top of the current image.



The `COPY` instruction copies new files or directories from `<src>` and adds them to the filesystem of the container at the path `<dest>`.



An `ENTRYPOINT` allows you to configure a container that will run as an executable.

Dockerized Python application on Wago Controller

3. Place the docker file and python application in PLC.

Using FTP, I've created a directory called Docker and within that directory, I have my Dockerfile and testapp.py

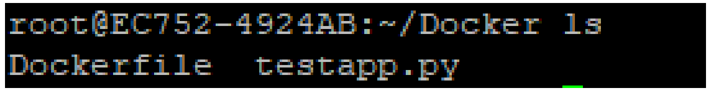
4. Docker Build

Using putty or whatever SSH utility you choose, go to the folder created that has your python script and dockerfile.

When inside the folder, use your docker build command. For this I use:

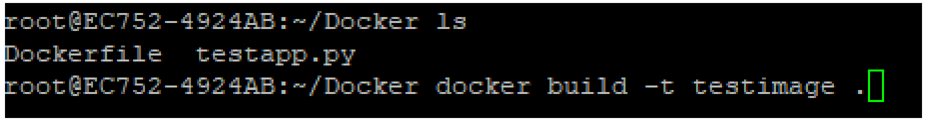
docker build -t testimage .

Files inside controller



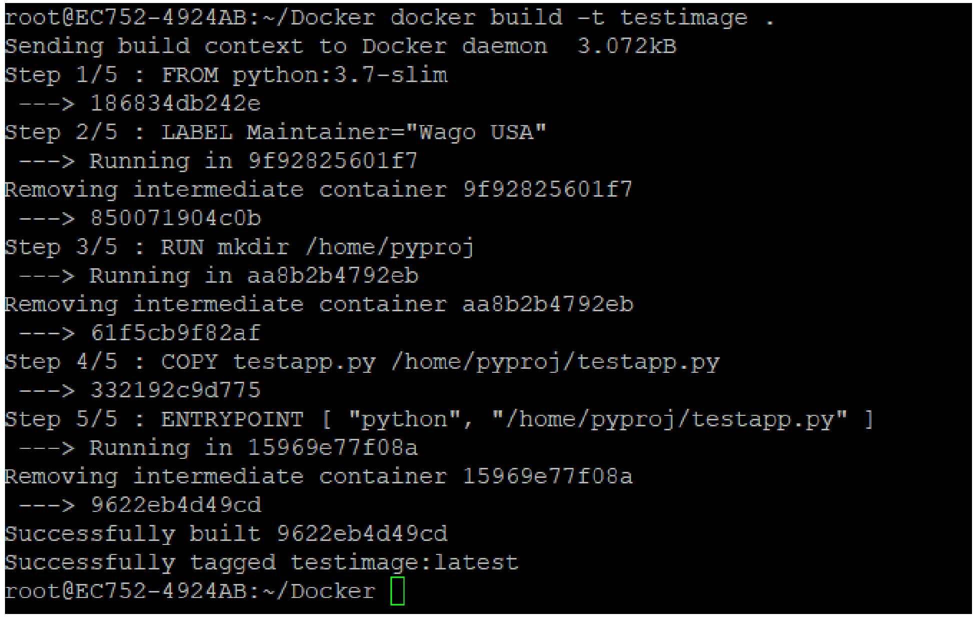
```
root@EC752-4924AB:~/Docker ls
Dockerfile  testapp.py
```

Docker build command



```
root@EC752-4924AB:~/Docker ls
Dockerfile  testapp.py
root@EC752-4924AB:~/Docker docker build -t testimage .
```

Successful Docker build screen



```
root@EC752-4924AB:~/Docker docker build -t testimage .
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM python:3.7-slim
 ---> 186834db242e
Step 2/5 : LABEL Maintainer="Wago USA"
 ---> Running in 9f92825601f7
Removing intermediate container 9f92825601f7
 ---> 850071904c0b
Step 3/5 : RUN mkdir /home/pyproj
 ---> Running in aa8b2b4792eb
Removing intermediate container aa8b2b4792eb
 ---> 61f5cb9f82af
Step 4/5 : COPY testapp.py /home/pyproj/testapp.py
 ---> 332192c9d775
Step 5/5 : ENTRYPOINT [ "python", "/home/pyproj/testapp.py" ]
 ---> Running in 15969e77f08a
Removing intermediate container 15969e77f08a
 ---> 9622eb4d49cd
Successfully built 9622eb4d49cd
Successfully tagged testimage:latest
root@EC752-4924AB:~/Docker
```

The docker build command will look for any file named "Dockerfile" in the present directory and run it . The "-t" flag is to name the image.

Dockerized Python application on Wago Controller

5. Check image and run container

If all goes well, you should have two new images in your machine:

The python base image that you selected via the FROM command and the new image that you built.

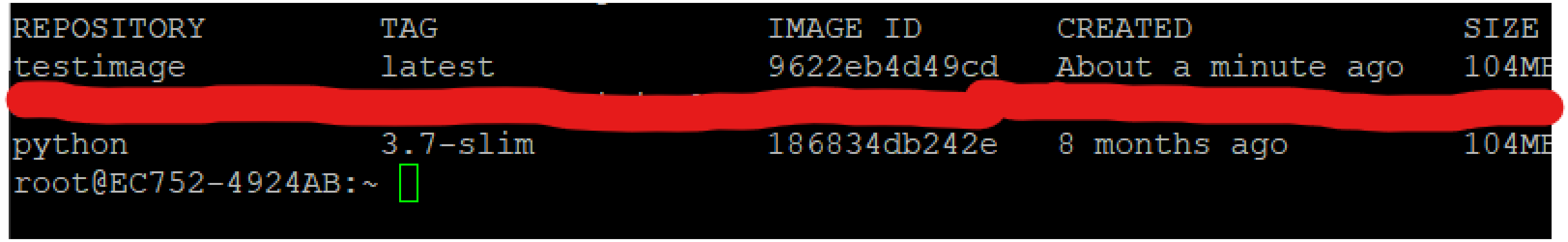
Now we just have to run the container:

docker run –it <image_ID>

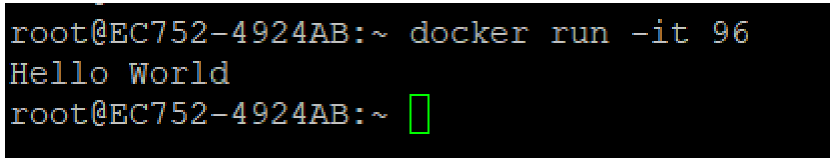
More info on docker run and docker build:

https://docs.docker.com/reference/cli/docker/container/run/

https://docs.docker.com/reference/cli/docker/image/build/



```
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
testimage           latest              9622eb4d49cd        About a minute ago   104MB
[redacted]
python              3.7-slim            186834db242e        8 months ago         104MB
root@EC752-4924AB:~
```



```
root@EC752-4924AB:~ docker run -it 96
Hello World
root@EC752-4924AB:~
```