Testing with Docker

Tom Buskens

June 22, 2019

Contents

1 Docker CI

In our repositories we wanted to use a CI tool to build our docker images with an automated test build. To accomplish this we used the GitLab CI/CD Runners running in a docker image. Because we wanted to use the docker build tools we need to run the dind image, which stands for Docker in Docker. This environment is totally save and you cannot access the host docker system because the runners are not running in a privileged environment but still using the host docker socket. When this is all setup, we can begin using our runner and continue creating our CI file.

```
Running with gitlab-runner 11.11.2 (ac2a293c)
  on Shared-01 P-JLjMku
ERROR: Preparation failed: Cannot connect to the Docker daemon at
unix:///var/run/docker.sock. Is the docker daemon running?
(executor_docker.go:969:0s)
Will be retried in 3s ...
ERROR: Preparation failed: Cannot connect to the Docker daemon at
unix:///var/run/docker.sock. Is the docker daemon running?
(executor_docker.go:969:0s)
Will be retried in 3s ...
ERROR: Preparation failed: Cannot connect to the Docker daemon at
unix:///var/run/docker.sock. Is the docker daemon running?
(executor_docker.go:969:0s)
Will be retried in 3s ...
ERROR: Job failed (system failure): Cannot connect to the Docker daemon
at unix:///var/run/docker.sock. Is the docker daemon running?
(executor_docker.go:969:0s)
```

Figure 1: Running the Gitlab Runner without the Docker socket mounted.

```
version: '3'
services:
  gitlab-runner1:
    image: gitlab/gitlab-runner:latest
    container_name: GitLabRunner1
    restart: always
    volumes:
      - ./config:/etc/gitlab-runner
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
        dockernet:
            ipv4_address: 192.168.177.48
networks:
    dockernet:
        external: true
```

Figure 2: The correct docker-compose.yml setup for the Gitlab Runner.

2 Docker build

Now we have setup our Gitlab Runner, we can use this to build our custom docker images with. To do this we need to create a .gitlab-ci.yml which contains the docker commands to run the test. First we build the image and if you want to, you can create a script to test the features of your image. This is something we didn't implement yet because all of our images are build for an Raspberry Pi, but the VPS server containing the Gitlab Runner is using a AMD64 processer and doesn't have the ina sensor, so we can not run the Raspberry Pi image.

```
image: docker:stable

variables:
  # For non-Kubernetes executors, we use tcp://docker:2375/
  DOCKER_HOST: tcp://docker:2375/
  # When using dind, it's wise to use the overlayfs driver for
  # improved performance.
  DOCKER_DRIVER: overlay2

services:
  - docker:dind

before_script:
  - docker info

build:
  stage: build
  script:
    - docker build -t registry.twizzel.net/hva/sfs-power-monitoring/prometheus-powermeter-build .
    - docker ps
```

Figure 3: Our .gitlab-ci.yml to build and run the image.

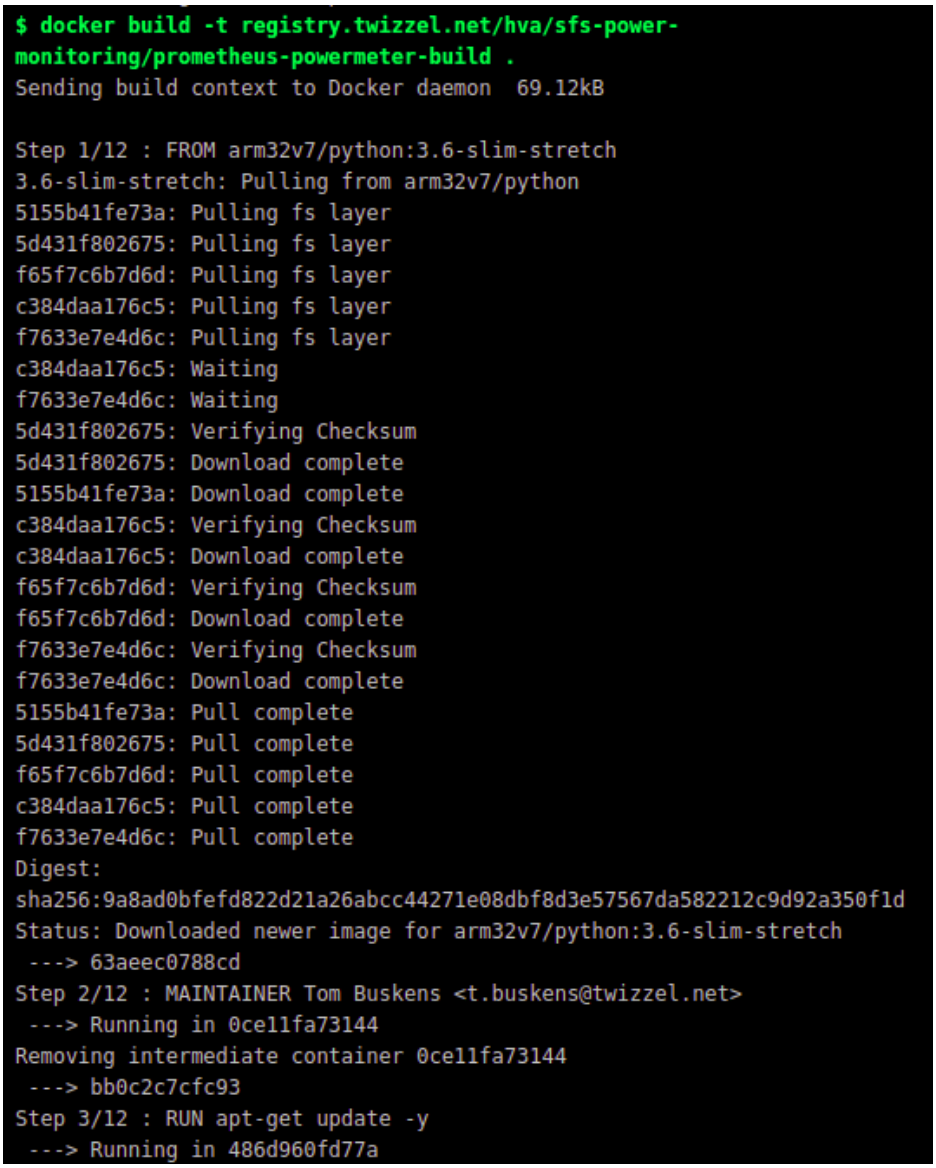

Figure 4: First build of our prometheus-powermeter image.

```
$ docker run registry.twizzel.net/hva/sfs-power-monitoring/prometheus-
powermeter-build
Traceback (most recent call last):
  File "metric_ina.py", line 61, in <module>
    inamodule = INA219(0.1, 2.0)
  File "/usr/local/lib/python3.6/site-packages/ina219.py", line 106, in
__init__
    self._i2c = I2C.get_i2c_device(address=address, busnum=busnum)
  File "/usr/local/lib/python3.6/site-packages/Adafruit_GPIO/I2C.py",
line 63, in get_i2c_device
    busnum = get_default_bus()
  File "/usr/local/lib/python3.6/site-packages/Adafruit_GPIO/I2C.py",
line 55, in get_default_bus
    raise RuntimeError('Could not determine default I2C bus for
platform.')
RuntimeError: Could not determine default I2C bus for platform.
ERROR: Job failed: exit code 1
```

Figure 5: Docker run works but fails because of a missing I2C bus.