Chapter 8

Additional containers and services in docker-compose.yaml

faasd supports deploying both functions via its REST API, and a series of additional services which are managed by `faasd` itself. This can be used to add something like a storage or caching capability to your functions. We'll explore how to configure the OpenFaaS core services and how to add Grafana for creating a dashboard and InfluxDB, which is a time series database.

Whilst faasd uses a file named `docker-compose.yaml`, it does not use Docker or Compose, only its format and spec.

faasd supports a sub-set of directives in the compose format including:

- bind-mounting volumes - for storage and persistence
- setting a specific runtime user, so that bind-mounting can be accessed by apps
- exposing services through ports
- adding Linux capabilities
- setting a dependency order

See the `docker-compose.yaml` file located at `/var/lib/faasd` for examples.

Once you deploy a core service, it's up to you how you want to interact with it.

- You can use it only from your functions using the service name for discovery such as influxdb, to allow functions to store data
- You can expose it only on the local host and use an inlets or SSH tunnel to forward it back to your local machine or another network

When it comes to exposing something like Grafana or Prometheus, think twice about how you are going to do that.

By default it has no TLS or authentication, so perhaps it's best if you deploy Caddy, and add a basic-auth rule in front of it, or perhaps only access it via a tunnel on your local machine.

Exposing core services

The OpenFaaS stack is made up of several core services including NATS and Prometheus.

Edit `/var/lib/faasd/docker-compose.yaml` and change:

You can expose a core service like the gateway on all interfaces by using the format `FROM_PORT:TO_PORT`. If you do this on an Internet-facing host, anyone with the host's IP address will be able to connect to the service.

```
  gateway:
    ports:
       - "8080:8080"
```

Expose Prometheus only to `127.0.0.1`.

You can expose a service to only the loopback of the faasd host with the following:

```
  prometheus:
    ports:
       - "127.0.0.1:9090:9090"
```

Expose Prometheus to only the OpenFaaS bridge device `openfaas0`, so that functions and core services can access it:

```
  prometheus:
    ports:
       - "10.62.0.1:9090:9090"
```

Most services should either not be exposed, or only exposed on the OpenFaaS bridge device.

Bind mount a folder:

Note that the source of every folder starts with `/var/lib/faasd/` so if you want to mount a `grafana` folder, it must be created at `/var/lib/faasd/grafana/`

```
  grafana:
...
    volumes:
      - type: bind
        source: ./grafana/
        target: /etc/grafana/provisioning/
```

You may also need to chown the local folder with a user like `1000` and also override the user in the container's section.

In the example above, that would mean running:

```
chown -R 1000:1000 /var/lib/faasd/grafana
```

Run as a specific user:

```
  postgresql:
    user: "1000"
```

If you do not have user `1000` in `/etc/passwd` you can add a system user which cannot log in for this purpose.

```
groupadd --gid 1000 faasd

useradd --uid 1000 --system \
  --no-create-home \
  --gid 1000 faasd
```

All containers for the core services and functions are added to a Linux bridge device called `openfaas0`, you can use its IP address to reference core services from functions. By default the address is `10.62.0.1`, but you can look it up with `ifconfig` or `ip addr`.

If using this value from your function, it is advisable to add an environment variable in your stack.yml file and read that at runtime, instead of hard-coding it.

```
functions:
  read-db:
    environment:
      postgres_host: 10.62.0.1
```

stack.yml example

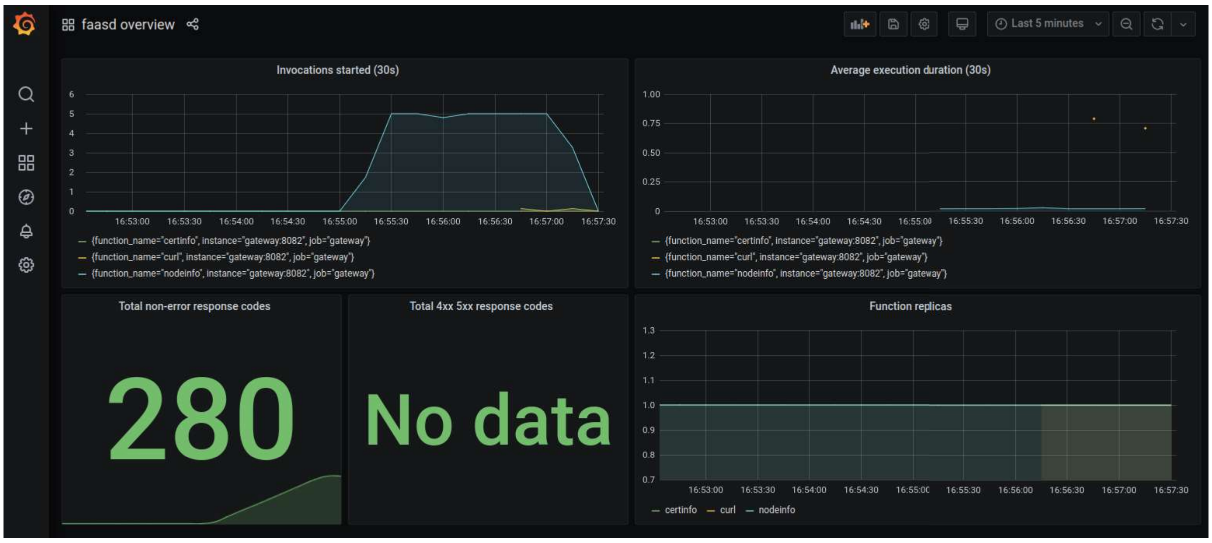
```
process.env.postgres_host
```

Example of accessing the variable in your code.

In a future version of faasd, you will be able to access any additional services you add using their name, so `postgresql` or `prometheus`.

Adding a Grafana Dashboard

Grafana is a visualisation tool for viewing time-series data from sources like InfluxDB and Prometheus. You can use it to monitor your OpenFaaS functions and see how long they take to execute, along with how many return a failure HTTP code.



faasd overview
Last 5 minutes
Invocations started (30s)
Average execution duration (30s)
{function_name="certinfo", instance="gateway:8082", job="gateway"}
{function_name="curl", instance="gateway:8082", job="gateway"}
{function_name="nodeinfo", instance="gateway:8082", job="gateway"}
Total non-error response codes
280
Total 4xx 5xx response codes
No data
Function replicas
certinfo
curl
nodeinfo

Figure 8.1: Grafana dashboard with a light load-test

Grafana dashboard with a light load-test

Make a data directory for Grafana:

```
sudo mkdir -p /var/lib/faasd/grafana
```

Edit `/var/lib/faasd/docker-compose.yaml` and add:

```
  grafana:
    image: docker.io/grafana/grafana:latest
    environment:
      - GF_AUTH_ANONYMOUS_ORG_ROLE=Admin
      - GF_AUTH_ANONYMOUS_ENABLED=true
      - GF_AUTH_BASIC_ENABLED=false
    volumes:
      # we assume cwd == /var/lib/faasd
      - type: bind
```

```
        source: ./grafana/
        target: /etc/grafana/provisioning/
    cap_add:
      - CAP_NET_RAW
    depends_on:
      - prometheus
    ports:
      - "3000:3000"
```

By default this will expose Grafana on the IP of the host, you can avoid this and use 127.0.0.1:3000 access Grafana:

```
    ports:
      - "127.0.0.1:3000:3000"
```

Restart faasd with:

```
sudo systemctl daemon-reload \
  && sudo systemctl restart faasd
```

Then check Grafana's logs:

```
sudo journalctl -t openfaas:grafana
```

Once you start Grafana, add a Datasource for Prometheus:

- Name: `faasd`
- URL: `http://prometheus:9090`

Now get your `dashboard.json` from your purchased bundle and import it via the Dashboards page, the URL should be `http://localhost:3000/dashboard/import`

You are looking for the field "Import via panel json" which is where you paste in the data. The dashboard will now appear under your Dashboards list.

Adding a database - Postgresql

Postgresql is described as "The World's Most Advanced Open Source Relational Database" and can be hosted on your faasd instance.

Add the following to docker-compose.yaml:

```
  postgresql:
    image: docker.io/library/postgres:11
    environment:
      PGDATA: /var/lib/postgresql/data/pgdata
      POSTGRES_PASSWORD_FILE: /run/secrets/postgres-passwd
      POSTGRES_USER: postgres
    volumes:
      - type: bind
        source: ./postgresql/pgdata
        target: /var/lib/postgresql/data/pgdata
      - type: bind
        source: ./postgresql/run
        target: /var/run/postgresql
      - type: bind
```

```
        source: ./postgresql/postgres-passwd
        target: /run/secrets/postgres-passwd
    user: "1000"
    ports:
      - "10.62.0.1:5432:5432"
```

Now create the data directories, so that your instance is stateful and can be restarted without dataloss:

```
# For databases
sudo mkdir -p /var/lib/faasd/postgresql/pgdata

# For the run lock
sudo mkdir -p /var/lib/faasd/postgresql/run

# For the password
TOKEN=$(head -c 12 /dev/urandom | shasum| cut -d' ' -f1)
echo $TOKEN > /var/lib/faasd/postgresql/postgres-passwd

# Set permissions
chown -R 1000:1000 /var/lib/faasd/postgresql/
```

Now restart faasd:

```
sudo systemctl daemon-reload \
  && sudo systemctl restart faasd
```

You should see the logs appear with:

```
sudo journalctl -t openfaas:postgresql

# Or, if it failed on:
sudo journalctl -t openfaas:faasd
```

This configuration only exposes port 5432 to your functions, your host will be `10.62.0.1` and the port will be 5432.

Adding a database - InfluxDB

You can add InfluxDB for access from your functions in the same way as Grafana.

Edit `/var/lib/faasd/docker-compose.yaml` and add:

```
  influxdb:
    image: docker.io/library/influxdb:1.7
    environment:
      - INFLUXDB_ADMIN_USER=admin
    volumes:
      # we assume cwd == /var/lib/faasd
      - type: bind
        source: ./influxdb/
        target: /var/lib/influxdb
    user: "1000"
    cap_add:
      - CAP_NET_RAW
```