Flex4apps

v0.1

Till Witt, Johannes Berg, Alex Nowak

Apr 13, 2018

Contents:

CHAPTER 1

Introduction

1.1 Management summary

Goal of this page is to provide a manual and scripts to

- get an reliable open source infrastructure for your business / environment
- reproducibly running in less than 1 hour
- at a cost of less 20 € month (lets see how this changes)

1.2 About the Flex4Apps

The convergence of cloud, communication and IoT infrastructure plus the trend towards virtual applications (e.g. migrating software to the cloud) create new challenges for application developers and infrastructure providers. The resulting systems are complex with dynamic resources hiding possible problems. This creates a requirement for flexible monitoring and optimisation methods. The Flex4Apps project addresses the challenges of monitoring and optimising large, distributed, cyber-physical systems. The goal of the project is to provide a solution to manage the high data volumes and complexity of system monitoring whilst disturbing the target system as little as possible.

The demonstrator is setup using the f4a.me domain.

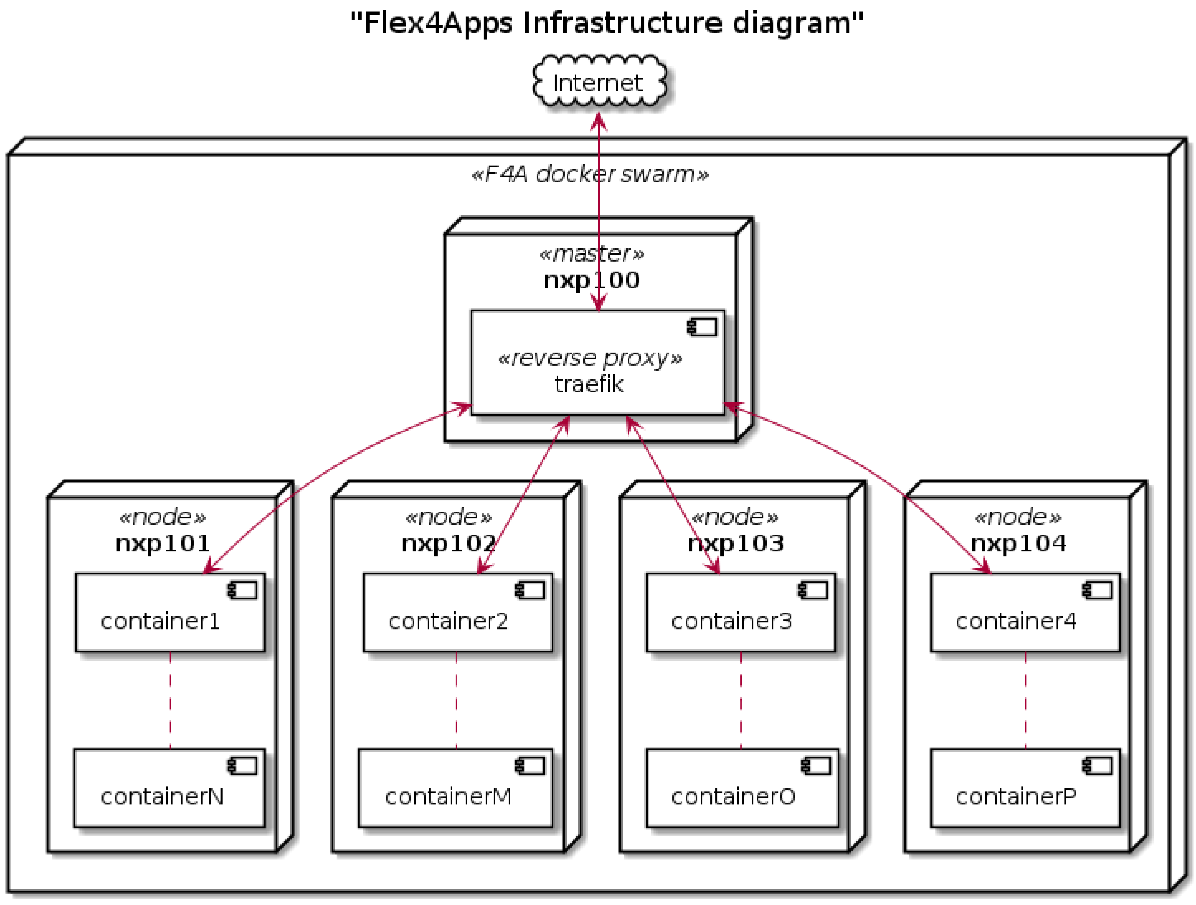
1.3 High level architecture

Goal of the project is not to worry about "hardware" while operating an infrastructure. Currently servers have to be rented / purchased. They need to be maintained and in case something breaks down, the operation of the overlaying service / system is impacted.

Docker swarm is one solution to achieve this abstraction layer. For this approach three or more (virtual) servers are required. RAM and CPU power is key, SSDs help to improve the performance.

Even though docker runs on many platforms we decided to operate based on an Ubuntu 16.04 LTS. Windows is also possible, but has some drawbacks.

Docker only achieves the independence with regards to the stateless execution of services like webservers, mailservers and other services. To provide persistent storage a file server / cluster and a database server is needed.



CHAPTER 2

Automatic server updates & reboots

A cron job can take care of that:

```
echo "0 3 * * * (apt-get update & apt-get -y upgrade && apt autoclean -y && apt␣
↪autoremove -y)" >> mycron && \
echo "0 4 * * 4 reboot" >> mycron && \
crontab -u root mycron && \
rm mycron && \
crontab -l
```

Make sure you have the right timezone set:

```
timedatectl set-timezone Europe/Berlin
```

To check the status and list the timezones:

```
timedatectl status
timedatectl list-timezones
```

CHAPTER 3

State of the architecture

3.1 XaaS vs self-hosted

3.2 server infrastructure

3.3 file system

3.4 database

3.5 applications

3.6 our prefered choice

CHAPTER 4

Docker

4.1 What is Docker

to be described

4.1.1 Step 1 - setting up the servers

Nowadays the servers are usually preinstalled or an installation process can be kicked off via web interface. For the F4A usecase we chose Ubuntu 16.04 LTS (Long term support).

First we should ensure that the system is up-to-date and secure. This is done by kicking off the advanced packaging tool (apt). Within this process we can directly install the docker server component. All steps are done by issueing the following command:

```
apt-get update && apt-get upgrade -y && apt install -y docker.io
```

As docker is still being developed, certain functionality still changes. This tutorial has been created using the following docker version (you can find our yours by executing `` `docker version` ``):

```
root@nxp100:~# docker version
Client:
 Version:      1.13.1
 API version:  1.26
 Go version:   go1.6.2
 Git commit:   092cba3
 Built:        Thu Nov  2 20:40:23 2017
 OS/Arch:      linux/amd64

Server:
 Version:      1.13.1
 API version:  1.26 (minimum version 1.12)
 Go version:   go1.6.2
 Git commit:   092cba3
 Built:         Thu Nov  2 20:40:23 2017
 OS/Arch:       linux/amd64
 Experimental:  false
root@nxp100:~#
```

4.1.2 Step 2 - initiate a swarm

Setting up the docker swarm. A swarm is a group of computers:

```
docker swarm init --advertise-addr 89.144.27.100
```

getting the feedback:

```
Swarm initialized: current node (r3t3pu7rd74njml1afsf2uoev) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token <some secret token displayed here> \
    89.144.27.100:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the␣
↪instructions.
```

If you dont remember the token etc - just run:

```
docker swarm join-token worker
```

4.1.3 Step 3 - preparing the domain

register any domain you like. Just make sure that the domain names are pointing to all the server IPs you have. With that load balancing / failover is possible:

```
f4a.me.                        86400    IN      SOA     nsa3.schlundtech.de. mail.
↪tillwitt.de. 2017112808 43200 7200 1209600 86400
f4a.me.                        86400    IN      NS      nsa3.schlundtech.de.
f4a.me.                        86400    IN      NS      nsb3.schlundtech.de.
f4a.me.                        86400    IN      NS      nsc3.schlundtech.de.
f4a.me.                        86400    IN      NS      nsd3.schlundtech.de.
f4a.me.                          600    IN      MX      10 mail.f4a.me.
*.f4a.me.          600 IN       A       89.144.24.15
*.f4a.me.                600    IN      A       89.144.27.100
*.f4a.me.                600    IN      A       89.144.27.101
*.f4a.me.                600    IN      A       89.144.27.102
*.f4a.me.                600    IN      A       89.144.27.103
nxp100.f4a.me.                 600      IN      A       89.144.27.100
nxp101.f4a.me.                 600      IN      A       89.144.27.101
nxp102.f4a.me.                 600      IN      A       89.144.27.102
nxp103.f4a.me.                 600      IN      A       89.144.27.103
nxp104.f4a.me.                 600      IN      A       89.144.24.15
```

4.2 Docker commands

4.2.1 docker run vs docker compose

Advantages of docker run are that the command is easy to issue, just a copy & paste to the servers command line. Downside is, that the commands get quite long and adding line breaks introduces another possible fault. If you want to correct a running service you need to remove it first and then reissue it.

Advantages of using a docker-compose.yml is that they are usually easy to edit. Disadvantage is that you have to create them on the server first then issue the command to start them - so one additional step. But the biggest advantage is that they can be re-executed on existing services which will lead to a service update.

4.2.2 Examples

starting a generic web application with docker run:

```
docker service create \
    --name demo \
    --label "traefik.port=80" \
    --network traefik-net \
    kitematic/hello-world-nginx
```

Thats all - and the service is running.

To create the same via docker-compose.yml:

```
version: "3"

services:
  nginx:
    image: kitematic/hello-world-nginx
    networks:
      - traefik-net
    deploy:
      labels:
        - traefik.port=80
        - "traefik.frontend.rule=Host:demo.f4a.me"
networks:
  traefik-net:
   external: true
```

Then you need to issue the following command:

```
docker stack deploy --compose-file docker-compose.yml demo
```

4.2.3 Conclusion

To quickly test a service - docker run is nice. But to maintain a production environment docker-compose files are strongly recommended.

4.3 Docker registry

Running your own registry:

```
docker service create \
   --name backoffice \
   --network traefik-net \
   --label "traefik.port=5000" \
   --label 'traefik.frontend.auth.basic=flex4apps:$apr1$G9e4rgPu$jbn2AAk2F.OeGnRVFnIR/
↪1' \
   --mount type=bind,src=/swarm/volumes/registry,dst=/var/lib/registry \
   registry:2
```

4.3.1 Pushing to private registry

The local image needs to be taged and then pushed:

```
docker tag phabricator_image registry.f4a.me/phabricator
docker push registry.f4a.me/phabricator
```

Run that image::

docker service create –name demo –label "traefik.port=80" -e "GITURL=https://secret@gogs.tillwitt.de/NXP/homomorphic-encryption-demo.git" flex4apps:GQfgCEsjkHC7LRf3Q9PkW4L6onDLtu@backoffice.f4a.me/homomorphic_img

4.3.2 Query the registry

Get the overview of all images:

```
https://registry.f4a.me/v2/_catalog
```

Get all tags of an image:

```
https://registry.f4a.me/v2/henc/tags/list
```

4.3.3 Private repository viewer

4.3.4 Alternatives

CHAPTER 5

Gluster

5.1 What is gluster

GlusterFS is a distributed file system that presents storage elements from multiple servers as a unified file system. The various servers, also known as cluster nodes, form a client-server architecture over TCP/IP.

5.2 Step 1 - join computers into a gluster

For each swarm the the `/etc/hosts` needs to be updated. Otherwise there will be no connection. The following command appends the IPs of our swarm nodes to the host file. They need to be adapted to match your system setup:

```
echo 89.144.27.100 gluster0 >> /etc/hosts
echo 89.144.27.101 gluster1 >> /etc/hosts
echo 89.144.27.102 gluster2 >> /etc/hosts
```

create a data where gluster shall store its operational data:

```
mkdir -p /data/gluster
```

If not done already you can install gluster via the following command:

```
apt-get update && apt-get install -y glusterfs-server && service glusterfs-server␣
↪status
```

Now we need to join the nodes to the gluster:

```
gluster peer probe gluster0 && gluster peer probe gluster1 && gluster peer probe␣
↪gluster2
```

And check if everything went well:

```
gluster peer status
```

This should return something like:

```
root@nxp100:~# gluster peer status
Number of Peers: 4

Hostname: gluster1
Uuid: 32e5a4ac-bd12-...
State: Peer in Cluster (Connected)

Hostname: gluster2
Uuid: 681007c5-ad57-...
State: Peer in Cluster (Connected)
```

5.3 Step 2 - create a sync volume

we need to create a volume for data syncing. (This only needs to be executed on one node):

```
gluster  volume create datapoint replica 3 transport tcp  \
          gluster0:/data/gluster \
          gluster1:/data/gluster \
          gluster2:/data/gluster \
          force
```

and then start the volume (This only needs to be executed on one node):

```
gluster volume start datapoint
```

last but not least we mount it (this needs to be done on each node!):

```
mkdir -p /data/shared
mount.glusterfs localhost:/datapoint /data/shared
```

The gluster is ready for use now

5.4 Experience

Strange shutdown of Cluster

gluster peer status gluster volume info gluster volume heal datapoint

apparently node was down.

CHAPTER 6

Traefik

6.1 What is traefik

Traefik is a Reverse Proxy Server + Load Balancer that facilitates the (automated) deployment of Docker Containers.

6.2 Setup procedure

Within this project we use v1.4.4 / roquefort

Hints for the setup[1]

```
--mkdir -p /docker/traefik

docker network create --opt encrypted --driver overlay traefik-net

docker network create --driver overlay traefik-net
```

```
# PULL UPDATE & LAUNCH DOCKER
#     git pull && docker stack deploy --compose-file docker-compose.yml demo

# REMOVE STACK
#     docker stack rm demo

# ADD BASIC AUTH and ESCAPE FOR docker-compose usage
# htpasswd -bBn user password | sed 's/\$/\$\$/g' #escape for docker-compose usage

version: "3"

services:
  nginx:
    image: kitematic/hello-world-nginx
```

[1] DDD Paul https://dddpaul.github.io/blog/2016/11/07/traefik-on-docker-swarm/

```
    networks:
      - traefik-net
    environment:
      - test=noContent
    deploy:
      labels:
        - traefik.port=80
#         - "traefik.frontend.auth.basic=witt:$$2y$$05$
↪$kOFY7071ilbnpiJNDaIO9e1WeuhHnKtp9Adrevz4r8wJ3b3X1XuqW"
#         - "traefik.frontend.auth.basic=ich:$$2y$$05$$jTZv0re2cXmiGrzRxW./8Ofse.6g/
↪AEChvbMGdqYKIMqsr8xW/c"
#         - "traefik.frontend.auth.basic=user:$$2y$$05$$IRrTxLpG7ICzroI8Pb5P4.
↪p2rMXGqyeeZM857BJxTFzP5q9W4RYuS"
        - "traefik.frontend.rule=Host:demo.f4a.me"
networks:
  traefik-net:
   external: true
```

To run it just on one machine:

    docker network create traefik-net

6.3 Basic auth support

create a file for authentication, so no need for listing the users in the call:

```
touch .htpasswd
htpasswd -bB .htpasswd username password
```

CHAPTER 7

Applications

7.1 What are applications

On the created system applications can be launched. Templates for applications are called images. Running applications are called containers.

7.2 Generic demonstrator

7.2.1 Docker file

```
# PULL UPDATE & LAUNCH DOCKER
#     git pull && docker stack deploy --compose-file docker-compose.yml demo

# REMOVE STACK
#     docker stack rm demo

# ADD BASIC AUTH and ESCAPE FOR docker-compose usage
# htpasswd -bBn user password | sed 's/\$/\$\$/g' #escape for docker-compose usage

version: "3"

services:
  nginx:
    image: kitematic/hello-world-nginx
    networks:
      - traefik-net
    environment:
      - test=noContent
    deploy:
      labels:
        - traefik.port=80
#         - "traefik.frontend.auth.basic=witt:$$2y$$05$
↪$kOFY7071ilbnpiJNDaIO9e1WeuhHnKtp9Adrevz4r8wJ3b3X1XuqW"
```

```
#         - "traefik.frontend.auth.basic=ich:$$2y$$05$$jTZv0re2cXmiGrzRxW./8Ofse.6g/
↪AEChvbMGdqYKIMqsr8xW/c"
#         - "traefik.frontend.auth.basic=user:$$2y$$05$$IRrTxLpG7ICzroI8Pb5P4.
↪p2rMXGqyeeZM857BJxTFzP5q9W4RYuS"
        - "traefik.frontend.rule=Host:demo.f4a.me"
networks:
  traefik-net:
   external: true
```

Simple demo

The following application is already working with the current setup.

To launch an easy demonstrator, lets instantiate a webserver and make it available at demo.f4a.net:

```
docker service create \
    --name demo \
    --label "traefik.port=80" \
    --network traefik-net \
    kitematic/hello-world-nginx
```

Generating a new user with password run:

```
htpasswd -nbm flex4apps password
```

or go to: http://www.htaccesstools.com/htpasswd-generator/

The output will be something like:

```
flex4apps:$apr1$XqnUcSgR$39wlPxxyyxPxXZjFb34wo.
```

Example for traefik label usage below. If single quotes are in the password they would need to be escaped.

To do that close the quoting before it, insert the escaped single quote, and re-open the quoting: `` `'first part'\''second part'` `` But I dont even know if md5 password contain single quotes.

How to start the demo service:

```
docker service create \
    --name demopw \
    --label "traefik.port=80" \
    --label 'traefik.frontend.auth.basic=myName:$apr1$a7R637Ua$TvXp8/lgky5MDLGLacI1e1
↪' \
    --network traefik-net \
    kitematic/hello-world-nginx
```

7.3 grafana

7.3.1 What is grafana

7.3.2 Setting it up

create the service like this:

```
docker service create \
  --name=grafana \
  --network traefik-net \
  --label "traefik.port=3000" \
  --mount type=bind,src=/swarm/volumes/grafana,dst=/var/lib/grafana \
  -e "GF_SECURITY_ADMIN_PASSWORD=someSecretPassword" \
  grafana/grafana
```

7.4 portainer

start portainer as a service we first need to create a data directory:

```
mkdir -p /docker/portainer
```

To start the container itself:

```
docker service create \
--name "portainer" \
--constraint 'node.role == manager' \
--network "traefik-net" --replicas "1" \
--mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
--mount type=bind,src=/docker/portainer,dst=/data \
--label "traefik.frontend.rule=Host:portainer.f4a.me" \
--label "traefik.backend=tool-portainer" \
--label "traefik.port=9000" \
--label "traefik.docker.network=traefik-net" \
--reserve-memory "20M" --limit-memory "40M" \
--restart-condition "any" --restart-max-attempts "55" \
--update-delay "5s" --update-parallelism "1" \
portainer/portainer \
-H unix:///var/run/docker.sock
```

7.5 Phabricator

The database user needs to be able to create databases

7.5.1 install

1. change your password:

   ```
   sed -i 's/<some secret>/yourPassword/g' Dockerfile
   ```

1. build the image with:

   ```
   docker build -t phabricator_image .
   ```

2. tag and push it to the registry:

   ```
   docker tag phabricator_image registry.f4a.me/phabricator
   docker push registry.f4a.me/phabricator
   ```

3. deploy it from the registry by executing the docker-compose:

```
docker-compose up
```

7.5.2 Docker file

```
# download base image ubuntu 16.10
FROM php:7.1-apache
RUN apt-get update
RUN apt-get -y install git

WORKDIR /repo
RUN git clone https://github.com/phacility/libphutil.git
RUN git clone https://github.com/phacility/arcanist.git
RUN git clone https://github.com/phacility/phabricator.git

RUN a2enmod rewrite
RUN chown -R www-data /repo
RUN chgrp -R www-data /repo


# mysqli extension
RUN docker-php-ext-install -j$(nproc) mysqli

# GD extension
RUN apt-get -y install libfreetype6-dev libjpeg62-turbo-dev libpng-dev
RUN docker-php-ext-configure gd --with-freetype-dir=/usr/include/ --with-jpeg-dir=/
↪usr/include/

RUN pecl install apcu
RUN apt-get install -y python3-pygments
RUN apt-get install -y sendmail

WORKDIR /repo/phabricator

# apache config
RUN sed -i 's#DocumentRoot /var/www/html#DocumentRoot /repo/phabricator/
↪webroot\nRewriteEngine on\nRewriteRule ^(.*)$          /index.php?__path__=$1  [B,L,
↪QSA]\n<Directory "/repo/phabricator/webroot">\n  Require all granted\n</Directory>#g
↪' /etc/apache2/sites-enabled/000-default.conf

# ssl reverse proxy - preabmle
RUN echo \<?php >> support/preamble.php
RUN echo \$_SERVER[\'REMOTE_ADDR\'] = \$_SERVER[\'HTTP_X_FORWARDED_FOR\']\; >>␣
↪support/preamble.php
RUN echo \$_SERVER[\'HTTPS\'] = true\; >> support/preamble.php

# phabricator configuration
RUN ./bin/config set mysql.host mariadb.f4a.me
RUN ./bin/config set mysql.user phabricator
RUN ./bin/config set mysql.pass ...
RUN ./bin/config set mysql.port 3306
RUN ./bin/config set phabricator.base-uri https://phabricator.f4a.me

RUN ./bin/config set phpmailer.smtp-host home.tillwitt.de
RUN ./bin/config set phpmailer.smtp-protocol TLS
RUN ./bin/config set phpmailer.smtp-port 587
RUN ./bin/config set phpmailer.smtp-user notify
RUN ./bin/config set phpmailer.smtp-password ...


ADD start.sh .
RUN chmod 755 start.sh

ENTRYPOINT ./start.sh && /bin/bash
```

7.5.3 Docker compose

```
# PULL UPDATE & LAUNCH DOCKER
#     git pull && docker stack deploy --with-registry-auth --compose-file docker-
↪compose.yml phabricator

# REMOVE STACK
#     docker stack rm phabricator

# ADD BASIC AUTH and ESCAPE FOR docker-compose usage
# htpasswd -bBn user password | sed 's/\$/\$\$/g' #escape for docker-compose usage

version: "3.1"


services:
   web:
     image: registry.f4a.me/phabricator
     networks:
       - traefik-net
#     ports:
#        - "80:80"
     deploy:
      labels:
        - traefik.port=80
        - "traefik.frontend.rule=Host:phabricator.f4a.me"

#   db:
#     image: mysql:5.7
#     volumes:
#       - ./data:/var/lib/mysql
#     restart: always
#     environment:
#       MYSQL_ROOT_PASSWORD: root
#       MYSQL_DATABASE: phabricator
#       MYSQL_USER: phabricator
#       MYSQL_PASSWORD: phabricator

networks:
  traefik-net:
   external: true
```

7.6 MariaDB

7.6.1 What is MariaDB

MariaDB is free and open source relational database system. It was created as a :fork: from MySQL after Oracle started releasing new functionality not as open source anymore and due to the high support cost of MySQL.

7.6.2 How to set it up

As usual make sure that the path for data volume exists:

```
mkdir -p /swarm/volumes/mariadb
```

The initiate the docker service:

```
docker service create \
      --name mariadb \
      --publish 3306:3306 \
      --network traefik-net \
      --mount type=bind,src=/swarm/volumes/mariadb,dst=/var/lib/mysq \
      --label "traefik.port=3306" \
      -e MYSQL_ROOT_PASSWORD=someSecretPassword \
      mariadb:latest
```

7.7 PhpMyAdmin

7.7.1 What is PhpMyAdmin

7.7.2 How to set it up

The following command will start up PhpMyAdmin:

```
docker service create \
    --name phpmyadmin \
    --label "traefik.port=80" \
    --network traefik-net \
    -e ALLOW_ARBITRARY=1 \
    nazarpc/phpmyadmin
```

7.8 gogs

7.8.1 What is gogs

todo

7.8.2 How to set it up

Pull image from Docker Hub.

very strange installation. First need to use –publish 3000:3000 and connect direct for install. Then remove instance and also remove published port. This is certainly something I need to review.

create the data volume for gogs:

```
mkdir -p /swarm/volumes/gogs
```

start the service:

```
docker service create \
    --name gogs \
    --mount type=bind,src=/swarm/volumes/gogs,dst=/data \
    --label "traefik.port=3000" \
    --network traefik-net \
    gogs/gogs
```

7.9 Drone

7.9.1 About the software

A continuous integration server which is open source, and tightly integrates with open source git platforms like gogs or services like github.

7.9.2 Setting it up

A good installation procedure is available here at http://docs.drone.io/install-for-gogs/. The corresponding commands for F4A are below:

```
docker run \
  --name drone \
  --label "traefik.port=8000" \
  --publish 8000:8000 \
  --publish 9000:9000 \
  -e DRONE_OPEN=true \
  -e DRONE_HOST=drone.f4a.me \
  -e DRONE_GOGS=true \
  -e DRONE_GOGS_URL=https://gogs.tillwitt.de \
  -e DRONE_SECRET=<some secret> \
  drone/drone:0.8

mkdir -p /swarm/volumes/drone

docker service create \
  --name drone \
  --label "traefik.port=8000" \
  --label "traefik.docker.network=traefik-net" \
  --network traefik-net \
  --mount type=bind,src=/swarm/volumes/drone,dst=/var/lib/drone/ \
  --publish 8000:8000 \
  --publish 9000:9000 \
  -e DRONE_OPEN=true \
  -e DRONE_HOST=drone.f4a.me \
  -e DRONE_GOGS=true \
  -e DRONE_GOGS_URL=https://gogs.tillwitt.de \
  -e DRONE_SECRET=<some secret> \
  -e DRONE_ADMIN=witt \
  drone/drone:0.8

docker service create \
  --name drone_agent \
  --mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
  --network traefik-net \
  -e DRONE_SERVER=drone:9000 \
  -e DRONE_SECRET=<some secret> \
  drone/agent:0.8
```

7.9.3 How to:

Once setup, with in this case gogs, you can log into the web interface. After a short sync all repositories should be visible. Activate drone.io for the corresponding repository.

To tell drone.io what to execute you need to add a `.drone.yml` to your repository. Examples are below.

7.9.4 Examples and configuration

example:

```
image: dockerfile/nginx
script:
  - echo hello world

  publish:
    docker:
      registry: registry.f4a.me
      email: witt@f4a.me
      repo: registry.f4a.me/flex4apps/flex4apps/homomorphic-encryption
      file: homomorphic-encryption/Dockerfile
      context: homomorphic-encryption
      tag: latest
      secrets: [ docker_username, docker_password ]
```

7.10 elasticsearch and kibana

7.10.1 What is elasticsearch and kibana

todo

7.10.2 How to set it up - release 2.4

Inspired by https://sematext.com/blog/docker-elasticsearch-swarm/. Issue the following command:

```
docker service create \
   --name esc24 \
   --label "traefik.port=9200" \
   --label 'traefik.frontend.auth.basic=flex4apps:$apr1$G9e4rgPu$jbn2AAk2F.OeGnRVFnIR/
↪1' \
   --network traefik-net \
   --replicas 3 \
   --endpoint-mode dnsrr \
   --update-parallelism 1 \
   --update-delay 60s \
   --mount type=volume,source=esc24,target=/data \
 elasticsearch:2.4 \
   elasticsearch \
   -Des.discovery.zen.ping.multicast.enabled=false \
   -Des.discovery.zen.ping.unicast.hosts=esc24 \
   -Des.gateway.expected_nodes=3 \
   -Des.discovery.zen.minimum_master_nodes=2 \
   -Des.gateway.recover_after_nodes=2 \
   -Des.network.bind=_eth0:ipv4_
```

7.10.3 Release 5.6

Inspired by https://github.com/elastic/elasticsearch-docker/issues/91 and https://idle.run/elasticsearch-cluster

The host systems have to be prepared to run elasticsearch in a docker:

```
echo vm.max_map_count=262144 >> /etc/sysctl.conf && sysctl --system && sysctl vm.max_
↪map_count
```

The issue the following command to start three instances of elasticsearch:

```
docker service create \
  --replicas 3 \
  --name esc56 \
  --label "traefik.port=9200" \
  --label 'traefik.frontend.auth.basic=flex4apps:$apr1$G9e4rgPu$jbn2AAk2F.OeGnRVFnIR/1
↪' \
  --mount type=volume,source=esc56,target=/data \
  --network traefik-net \
  elasticsearch:5.6.4 bash -c 'ip addr && IP=$(ip addr | awk -F"[ /]*" "/inet .*\/24/
↪{print \$3}") && \
      echo publish_host=$IP && \
      exec /docker-entrypoint.sh -Enetwork.bind_host=0.0.0.0 -Enetwork.publish_host=
↪$IP -Ediscovery.zen.minimum_master_nodes=2 -Ediscovery.zen.ping.unicast.hosts=tasks.
↪esc56'
```

7.11 kibana

7.11.1 What is kibana

todo

7.11.2 How to set it up - release 2.4

Issue the following command:

```
docker service create \
   --name kb56 \
   --label "traefik.port=5601" \
   --label 'traefik.frontend.auth.basic=flex4apps:$apr1$G9e4rgPu$jbn2AAk2F.OeGnRVFnIR/
↪1' \
   --network traefik-net \
   -e "ELASTICSEARCH_URL=http://esc56:9200" \
   kibana:5.6
```

7.12 Next steps

The following applications will be considered for next steps:

- drone.io
- https://codeship.com
- pritunl

CHAPTER 8

Security

CHAPTER 9

Let’s encrypt

Rate limit 20 per week

Where to search for certificates:

https://crt.sh/?q=%25f4a.me

9.1 Docker swarm networks

9.2 Docker secrets

9.3 Docker notary?

CHAPTER 10

Monitoring

prometheus

- chmod 777 for directory needed otherwise "Opening storage failed" err="open DB in /prometheus: open /prometheus/583762017: permission denied"

https://github.com/epasham/docker-repo/blob/master/monitoring/prom-stack/promUp.sh

## node exporter

docker service create –name nodeexporter –mode global –network traefik-net –label com.group="prom-monitoring" –mount type=bind,source=/proc,target=/host/proc –mount type=bind,source=/sys,target=/host/sys –mount type=bind,source=/,target=/rootfs prom/node-exporter:latest –collector.filesystem.ignored-mount-points "^/(sys|proc|dev|host|etc)($|/)"

CHAPTER 11

cheat sheet

11.1 Clean up of containers

remove all exited containers (should be run on each node):

```
docker rm $(docker ps -q -f status=exited)
```

attach bash to a running container:

```
sudo docker exec -i -t containername /bin/bash
```

11.2 Dos and donts

https://community.spiceworks.com/topic/1832873-a-list-of-don-ts-for-docker-containers

1. Don't store data in containers
2. Don't ship your application in two pieces
3. Don't create large images
4. Don't use a single layer image
5. Don't create images from running containers
6. Don't use only the "latest" tag
7. Don't run more than one process in a single container
8. Don't store credentials in the image. Use environment variables
9. Don't run processes as a root user
10. Don't rely on IP addresses

11.3 Updating this documentation

issue the following command to update this documentation:

```
docker run --name sphinxneeds --rm \
      -e "Project=Flex4apps" \
      -e "Author=Till Witt, Johannes Berg, Alex Nowak" \
      -e "Version=v0.1" \
      -v "$(pwd)/compose:/project/compose" \
      -v "$(pwd)/docs:/project/input" \
      -v "$(pwd)/output:/project/output" \
      -i -t tlwt/sphinxneeds-docker


docker run --name buildTheDocs --rm \
      -e "Project=Flex4apps" \
      -e "Author=Till Witt, Johannes Berg, Alex Nowak" \
      -e "Version=v0.1" \
      -v "$(pwd)/compose:/project/compose" \
      -v "$(pwd)/docs:/project/input" \
      -v "$(pwd)/output:/project/output" \
      -i -t sphinx_image
```

CHAPTER 12

glossary

todo this still needing correction

fork to be described ;-)

CHAPTER 13

Indices and tables

- genindex
- modindex
- search

Index