DevOps Tools Day19

Prakash Ramamurthy
prakash.ramamurthy@wipro.com

Agenda

- Docker Services
- Docker Stack
- Docker Swarm
- Docker Secret
- Hands-On Demonstration

Docker Services

Docker Services

Definition

- Application is divided into several pieces called services
- Services are basically containers in production
- Each service runs only one image
- Services can be replicated by replicating containers
- Service port can be exposed by container port mapping

Docker Compose File

Definition

- Pulls the image from registry
- Can create multiple instances of service
- Automatically restart the containers if they fail
- Maps the container port to system port
- Instruct the containers to share common port through load-balanced network
- Defines overlay network with required settings

Docker – Overlay Network

Definition

- Creates distributed network among multiple Docker hosts
- Overlays on host-specific networks
- Allow containers on different hosts to communicate securely
- Transparently handles routing of each packet among containers on different Docker hosts

Docker Stack

Docker Stack

Definition

- Group of interrelated services
- Allow sharing of dependencies
- Services can be orchestrated and scaled together
- Single stack can define and coordinate the function of entire application
- More complex applications can use multiple stcks

Docker Swarm

Docker Swarm

Definition

- Group of machines that are running Docker
- Multiple machines form together a cluster
- Swarm cluster is managed by “manager”
- “workers” joined into swarm together with manager run multiple replicas of services
- Only swarm manager accept commands
- Manager authorize other machines to join swarm as workers
- Manager schedule which service/container to run on which worker node

Docker Secrets

Docker Secrets

Definition

- A piece of data such as password, SSH Private key or SSL certificate
- Should be always encrypted while storing in a container or transmitting over network
- Data is automatically encrypted while in transit or while sitting in container
- Accessible only to those services that are explicitly granted to access it
- Services can access secrets only when they are running tasks
- Provide a layer of abstractions between container and a set of credentials

Hands-On Demonstration

Hands-On Demonstration

Docker Service - Need of Swarm

- Docker Service main command

```
$ docker service
```

- List the existing services (Result in error as Swarm is not initialized)

```
$ docker service ls
```

- Now initialize the Docker Swarm

```
$ docker swarm init
```

- Now you can list the services

```
$ docker service ls
```

Hands-On Demonstration

Docker Service

- Create Docker Service

```
$ docker service create --name sayhello --replicas 2 sayhello
```

- List the docker service

```
$ docker service ls
```

- List the task associated with Docker Service

```
$ docker service ps sayhello
```

- List the container where the tasks are running

```
$ docker container ls
```

Hands-On Demonstration

Docker Service - Scaling

- Inspecting Docker Service

```
$ docker service inspect --pretty sayhello
```

- Scale the Docker Services for replication

```
$ docker service scale sayhello=5
```

- List the tasks to confirm number of tasks

```
$ docker service ps sayhello
```

- Scale the Docker service again

```
$ docker service scale sayhello=3
```

- List the tasks to confirm the number of tasks. You may also list containers.

```
$ docker service ps sayhello
$ docker container ls
```

Hands-On Demonstration

Docker Service – Self Healing

- List the container

```
$ docker container ls
```

- Create a fault by stopping one of the container

```
$ docker container stop 82acda6220ae
```

- Repeat the following command couple of times. Look for failure of service and its recreation as part of self-healing.

```
$ docker service ps sayhello
```

- Check for creation of another container in place of stopped container

```
$ docker container ls -a
```

Hands-On Demonstration

Docker Service – Update

```
$ cat app.py
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visits = redis.incr("counter")
    except RedisError:
        visits = "<i>cannot connect to Redis, counter disabled</i>"

    html = "<h3>Hello {name}!</h3>" \
           "<h1>Adding a small change</h1>" \
           "<b>Hostname:</b> {hostname}<br/>" \
           "<b>Visits:</b> {visits}"
    return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname(), visits=visits)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)
```

Hands-On Demonstration

Docker Service – Update

- Build new image

```
$ docker build -t sayhello_new .
```

- List both the old and new image

```
$ docker image ls
```

- Remove existing service that has no port forwarding

```
$ docker service ls
$ docker service rm sayhello
```

- Create new service with port forwarding

```
$ docker service create --name sayhello2 --replicas 1 --publish 11022:80 sayhello
```

Hands-On Demonstration

Docker Service – Update

- List the new service with sayhello image

```
$ docker service ls
```

- Update the service with new image sayhello_new

```
$ docker service update --image sayhello_new sayhello2
```

- List the services to check for change in image

```
$ docker service ls
```

- Check the output on browser:

http://<IP address>:11022

Hands-On Demonstration

Docker Service – Rollback

- Rollback the service update

```
$ docker service ls
```

- List the services to check for change in image

```
$ docker service ls
```

- Check the output on browser:

http://<IP address>:11022

Hands-On Demonstration

Docker Stack

- Docker Stack Command

```
$ docker stack
```

- Create a Stack of services

```
$ docker stack deploy -c docker-compose-sayhello.yaml sayhello_stack
```

- List the stack

```
$ docker stack ls
```

- List tasks in the stack

```
$ docker stack ps sayhello_stack
```

- List services in the stack

```
$ docker stack services sayhello_stack
```

Hands-On Demonstration

Docker Swarm

- Docker Swarm Command

```
$ docker swarm
```

- To Leave Docker Swarm

```
$ docker swarm leave –force
```

- To initialize Swarm Manager to generate token for workers to join

```
$ docker swarm init --advertise-addr 10.199.0.104
```

- To generate token for workers to join the swarm

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

Hands-On Demonstration

Docker Secret

- Docker Secret Command

```
$ docker secret
```

- Create a Secret

```
$ echo "This is a secret" | docker secret create my_secret_data -

$ docker service create --name webservice --secret="my_secret_data" tomcat:8
```

Hands-On Demonstration

Docker Secret

- List the service, task and container

```
$ docker service ls
$ docker service ps webservice
$ docker container ls
```

- Check the container for the secrets folder

```
$ docker container exec 669235a8eec6 ls -l /run/secrets
```

- Check whether the secrets folder has the secret content

```
$ docker container exec 669235a8eec6 cat /run/secrets/my_secret_data
```

Hands-On Demonstration

Docker Secret

- Check whether this secret content will be part of image

```
$ docker container commit 669235a8eec6 newtomcat
```

- Create container from new image and check /run/secrets/my_secret_data

```
$ docker container run -d newtomcat
$ docker container ls
$ docker container exec 29794907fcb1 ls -l /run/secrets
$ docker container exec 29794907fcb1 cat /run/secrets/my_secret_data
```

Hands-On Demonstration

Docker Secret

- Try removing the secret

```
$ docker secret ls
$ docker secret rm my_secret_data
ERROR
```

- Remove the secret connected with a service

```
$ docker service update --secret-rm="my_secret_data" webservice

$ docker container ls
```

- Check whether the secret is removed from container running the task of service

```
$ docker container exec 40d947afeb3e ls /run/secrets/
```

- ls: cannot access '/run/secrets/': No such file or directory

Hands-On Demonstration

Docker Secret

- Remove the service

```
$ docker service rm webservice
```

- Remove the secret

```
$ docker secret rm my_secret_data
```

Thank You