Linux 2 DEVOPS 2020
Lektion 10

Idag

- Andra sätt att virtualisera
- Dockers
- Infrastructure as code och Ansible

Containers

- Inte en hel virtuell dator, utan bara en viss miljö i ett virtuellt operativsystem
- Körs på ett operativsystem och delar oftast detta operativsystems kärna, samt bibliotek och en del binärer
- Kan betraktas som en abstraktion för applikationslagret i OSI-modellen
- Exempel: Docker, LXC, Solaris Containers

Containers

”[Containers] may look like real computers from the point of view of programs running in them. A computer program running on an ordinary operating system can see all resources (connected devices, files and folders, network shares, CPU power, quantifiable hardware capabilities) of that computer. However, programs running inside of a container can only see the container's contents and devices assigned to the container. ” (Wikipedia)

VMs vs containers

Bilder stulna från backblaze.com

VMs vs containers

Tabell stulen från backblaze.com

Övning 1

- Ni har en fysisk server som ni vill använda till att snabbt sätta upp och köra testmiljöer för er applikation som körs under Linux.
- Lista fördelar med att köra containers respektive virtuella servrar på denna server.

Övning 1, exempel

- Containers:
  - Behöver mindre kraft för varje container än för en virtuell server
  - Går fortare att få igång testkörning för applikationer
  - Kräver mindre minne
- Virtuella servrar:
  - Tillgång till hela operativsystemet
  - Kan ha helt olika operativsystem

Docker

- Docker container engine
- Kan köra mjukvara i containers
- ”OS level virtualization”
- Containers är fristående och i princip isolerade från varandra, men kan kommunicera med varandra
- Alla containers har samma operativsystem

Docker intro

https://www.youtube.com/watch?v=V9AKvZZCWLc

Installera Docker

- Ubuntu 18: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04
- Motsvarande för Ubuntu 20: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04

Installera Docker

- Se till att få in GPG-nyckeln för Dockers repository

```
sudo apt install apt-transport-https ca-certificates curl
software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -
```

- Lägg till Dockers repository till de repositories servern använder

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"

sudo apt update

apt-cache policy docker-ce
```

Installera Docker

- Installera Docker

```
sudo apt install docker-ce
```

- Kontrollera att den snurrar

```
sudo systemctl status docker
```

- Testa

```
sudo docker run hello-world
sudo docker info
```

Övning 2

- Installera docker.
- Kör den förpreparerade ”Hello world” som test.

  docker run hello-world

Övning 2

```
sudo docker run hello-world
```

Fler Docker-kommandon

- Ta reda på vilka ”images” du har att köra på servern

```
docker image ls
docker images
```

- Ta reda på vilka containers som körs på servern

```
docker ps
```

Fler Docker-exempel

- Köra bash, dvs få en miljö att köra sina program i
  - Interaktivt

    ```
    docker run -it bash
    ```
  - Skicka in kommando

    ```
    docker run -t bash <någonting>
    ```

Övning 3

- Använd bash i en docker-container för att köra en egen enkel ”hello world”

Övning 3

```
docker run -t bash echo ”hej världen”
```

Docker

- Viktiga kommandon
  - run
  - info
  - ps
  - config
  - container
  - images
  - pull

Docker

- Förbindelse mellan host och container
- Containernamnet hittar man med docker ps
- Kopiera filer till container:

  docker cp <fil> <container>:<path>
- Kopiera filer från container:

  docker cp <container>:<path/fil> <fil>

Övning 4

- Gör ett enkelt skript som räknar från 1 till 10 med en sekunds paus för varje steg
- Starta bash i en docker-container
- Kopiera scriptet till din container (lägg det t ex under /tmp)
- Kör scriptet i din container
- Observera intressanta saker kring sökvägar och att scriptet i containtern bara finns så länge som containern finns

Övning 4

```
sudo docker run -it bash
```

```
sudo docker cp ./mycounter.sh fa381cde7ee1:/tmp
```

script som kommer strula lite:

```
#!/bin/bash

for i in  {1..10}
do
echo $i;
sleep 1;
done
```

Docker image

- Skapa en image-fil, typiskt med namnet Dockerfile
  - Viktigt att ha med ett ”FROM” som man bygger utifrån, t ex en Linux-version eller en programmiljö
- Bygg en image med docker build
- Nu finns den lokalt och du kan köra din image

Dockerfile, exempel 1

```
FROM alpine
CMD ["echo", "Hello StackOverflow!"]
```

Exempel stulet från
https://riptutorial.com/docker/example/10772/helloworld-dockerfile

Dockerfile, exempel 2

```
FROM python:3

# set a directory for the app
WORKDIR /usr/src/app

# copy all the files to the container
COPY . .

# install dependencies
RUN pip install --no-cache-dir -r
requirements.txt

# define the port number the container
should expose
EXPOSE 5000

# run the command
CMD ["python", "./app.py"]
```

Exempel stulet från https://docker-curriculum.com/

Bygg en image

- Bygg en image med docker build

  `docker build -t <namn> <path>`

  Exempel: `sudo docker build -t hello .`

Övning 5

- Plocka hem Alpine att använda som bas för en egen enkel image
- Testa att köra Alpine i docker
- Skapa en Dockerfile för en image ”hello” som är din egen implementation av ett ”hej världen”
- Bygg och kör din image

Övning 5

Dockerfile :

```
FROM alpine
CMD ["echo", "Hello again world!"]
```

```
sudo docker build -t hello .
```

```
sudo docker run hello
```

Övning 6

- Skapa nu en Dockerfile för en image ”counter” som kör ditt räkneskript (det som räknar från 1 till 10) under bash
- Bygg och kör din image

Övning 6

Dockerfile :

```
FROM bash
COPY . .
CMD ["bash", "./mycounter.sh"]
```

```
sudo docker build -t counter .
```

```
sudo docker run counter
```

Docker

- Finns många grunduppsättningar (images) att utgå från
- Deploy av egna program genom egna images som använder sig av dessa som bas
- Väldigt snabb deploy när konfigurationen väl är gjord

Infrastructure as code (IaC)

- Datacenter / kluster med maskiner som definieras genom logisk konfiguration, inte kopplat till hårdvara
- Mycket skalbara miljöer som dessutom förmodas hålla ned kostnaden för infrastruktur
- Exempel: Chef, Puppet, Ansible

Infrastructure as code (IaC)

”Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools. The IT infrastructure managed by this process comprises both physical equipment, such as bare-metal servers, as well as virtual machines, and associated configuration resources.” (Wikipedia)

Infrastructure as code (IaC)

- Deskriptiv modell för att hantera på infrastruktur
  - Fysiska och virtuella maskiner
  - Nätverk och nätverkstopologi
  - Lastbalansering
- Versionshantering för den uppbyggda infrastrukturen
- Egentligen samma principer som för att hantera applikationer inom DevOps

Infrastructure as a Service (IaaS)

- Abstraherar tillgång till fysiska resurser
- Ger tillgång till nätverk, servrar etc
- API:er av olika slag för att allokera och använda resurser

Infrastructure as a Service (IaaS)

”Infrastructure as a service (IaaS) are online services that provide high-level APIs used to dereference various low-level details of underlying network infrastructure like physical computing resources, location, data partitioning, scaling, security, backup etc.” (Wikipedia)

IaC och IaaS

- IaC är instruktionerna för servicen
- IaaS är ett sätt att leverera servicen

Ansible

- Verktyg för IaC
- Konfigurationshantering
- Automatisering av återkommande uppgifter
- Öppen källkod + RedHat-moduler
- https://www.ansible.com/

Ansible

”Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code.[2] It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows. It includes its own declarative language to describe system configuration.” (Wikipedia)

Ansible

- Inventory i textfiler
- ssh som normal inloggning
- Köra kommandon på flera noder parallellt
- Playbooks för orkestrering

Ansible, exempel

Exempel från www.ansible.com

Tillbakablick, reflektion, kommentarer ...