Operativni Sistemi i Računarstvo u Oblaku

II Semestar – 2023/24 – Vježbe

Sedmica 5

Handout za Vježbe

Agenda:

- Linux za sistemsku administraciju
- Docker
- Pokretanje servisa uz pomoć Docker-a

Kontakt:

Narcisa.hadzajlic@size.ba

Linux za sistemsku adminsitraciju

- Sistemski Logovi
- Komande za manipulaciju procesa.
- Komande za procese
- Administracija grupa i korisnika
- Varijable okruženja
- Monitoranje korisnika

Sistemski Logovi

Logovi se nalaze na putanji „**/var/log/“**. Čitanje logova u Linuxu je važan postupak za praćenje različitih događaja i problema na sistemu. Postoje različiti logovi koje Linux sistem generiše, a neki od najčešćih su sljedeći:

- Syslog - glavni dnevnik događaja na sistemu koji sadrži sve važne događaje vezane uz sistem, uključujući kernel poruke, poruke aplikacija, poruke sistema sigurnosti.
- Auth.log - dnevnik koji sadrži informacije o prijavi i autentikaciji korisnika na sistemu, uključujući i neuspjele prijave.
- Apache access i error logovi - dnevnik koji sadrži informacije o pristupu web browseru, uključujući korisničke zahtjeve i greške.
- Mail.log - dnevnik koji sadrži informacije o e-pošti, uključujući slanje, primanje i greške.
- Cron.log - dnevnik koji sadrži informacije o agendama na sistemu, uključujući i pokretanje skripti i programskih alata na rasporedu.

Otvorite prozor terminala i ukucajte naredbu cd /var/log i onda naredbu ls i vidjet ćete log dnevnike smještene u ovom direktoriju.

Za čitanje logova u Linuxu možete koristiti naredbu "tail", koja prikazuje zadnjih nekoliko linija dnevnika. Primjerice, "tail -f /var/log/syslog" prikazat će zadnjih 10 linija syslog datoteke i nastaviti s prikazivanjem novih linija koje se dodaju u dnevnik dok se ne prekine naredba (CTRL+C). Ako želite pregledati cijelu datoteku, možete koristiti naredbu "less", koja omogućuje pomicanje naprijed i natrag kroz dnevnik.

Komande za održavanje sistema

- shutdown: Koristi se za isključivanje ili restartovanje sistema.
- reboot: Koristi se za ponovno pokretanje sistema.
- halt: Koristi se za zaustavljanje sistema.
- init: Koristi se za prelazak na određeni runlevel ili za pokretanje sistemskih servisa.

Komande za manipulaciju procesa

- systemctl: Koristi se za upravljanje sistemskim servisima.
- ps: Koristi se za prikazivanje trenutno pokrenutih procesa.
- top: Koristi se za prikazivanje aktivnih procesa i njihovih resursa.
- kill: Koristi se za prekid izvršavanja procesa.

Administracija grupa i korisnika

- Useradd Koristi se za kreiranje novog korisničkog naloga.
- Groupadd Koristi se za kreiranje nove korisničke grupe.
- Userdel Koristi se za brisanje korisničkog naloga.
- Groupdel Koristi se za brisanje korisničke grupe.
- Usermod Koristi se za izmjenu postavki postojećeg korisničkog naloga.

Specificne lokacije

1. /etc/passwd Datoteka koja sadrži informacije o korisničkim nalozima.
2. /etc/group Datoteka koja sadrži informacije o korisničkim grupama.
3. /etc/shadow Datoteka koja sadrži šifrirane lozinke korisničkih naloga

---

Varijable okruženja

Environment variables (varijable okruženja) su vrijednosti koje su dostupne u okruženju operativnog sistema i koriste se za razne potrebe aplikacija i sistema. One sadrže podatke kao što su putanja do datoteka, korisnička imena, postavke sistema, i drugo.

U Linuxu, neke od uobičajenih okružnih varijabli uključuju:

- HOME - putanja do mape matičnog direktorija trenutno prijavljenog korisnika.
- PATH - popis mapa u kojima operativni sustav traži izvršne datoteke
- USER - korisničko ime trenutno prijavljenog korisnika.
- SHELL - putanja do ljuske koju koristi trenutni korisnik.
- LANG - postavka jezika sustava.

Naredba "**env**" prikazuje sve okružne varijable na vašem sistemu.

Primjer: "**env | grep USER**" će izlistati sve varijable okruženja koje sadrže riječ "USER" u njihovom imenu.

Također, naredbua "**export**" da bi se postavila vrijednost okružne varijable.

Primjer: "**export MY_VAR=my_value**" će postaviti okružnu varijablu "MY_VAR" na vrijednost "my_value". Ova varijabla će biti dostupna u svim sljedećim procesima koji se pokrenu u trenutnom okruženju.

Lokacija ovih varijabli se nalazi u **/etc/.bashrc**

Docker

Docker je platforma koja se koristi za razvoj, implementaciju i pokretanje aplikacija u izoliranim okruženjima zvanim kontejneri. Ovi kontejneri sadrže sve što je potrebno za pokretanje aplikacije, uključujući kod, vrijeme izvođenja, biblioteke i zavisnosti, osiguravajući konzistentnost u različitim okruženjima

- Docker Images:
    - Docker slike su lagani, samostalni, izvršni paketi koji sadrže sve što je potrebno za pokretanje dijela softvera, uključujući kod, vrijeme izvođenja, biblioteke, ovisnosti i druge konfiguracijske datoteke.
    - Slike su građevni blokovi kontejnera. Kreiraju se pomoću Dockerfiles-a ili povlačenjem unaprijed napravljenih slika iz Docker Hub-a ili drugih spremišta.
    - Slike su nepromjenjive, što znači da su samo za čitanje i ne mogu se mijenjati nakon kreiranja. Bilo kakve modifikacije slike rezultiraju stvaranjem novog sloja slike.

- Containers:
    - Kontejneri su instance Docker slika koje se pokreću kao izolovani procesi na operativnom sistemu domaćina.
    - Oni pružaju konzistentno okruženje za pokretanje aplikacija, osiguravajući da se ponašaju na isti način bez obzira na okruženje u kojem su raspoređene.
    - Kontejneri su lagani, prenosivi i mogu se lako postaviti, skalirati i njima upravljati.
- Dockerfile:
    - Dockerfile je tekstualna datoteka koja sadrži upute za pravljenje Docker slike.
    - Određuje osnovnu sliku koju treba koristiti, postavlja okruženje, kopira datoteke i direktorije u sliku, instalira ovisnosti i definira komande koje će se pokrenuti kada se kontejner pokrene.
    - Dockerfiles omogućavaju programerima da automatiziraju proces kreiranja Docker slika, osiguravajući konzistentnost i reproduktivnost u različitim okruženjima.

Razlike:

- Docker slika vs. kontejner:

Slika je statičan predložak samo za čitanje koji sadrži kod aplikacije i sve njegove zavisnosti, dok je kontejner instanca slike koja se može pokrenuti i koja se pokreće kao proces na operativnom sistemu domaćina.

- Docker slika vs. Dockerfile:

Slika je rezultat izgradnje Dockerfile-a, koji definira konfiguraciju i sadržaj slike. Dockerfile specificira kako napraviti sliku, dok je sama slika upakovani rezultat.

- Kontejner vs. Dockerfile:

Kontejner je instanca slike koja se izvodi kao proces na operativnom sistemu domaćina, dok je Dockerfile tekstualna datoteka koja sadrži uputstva za pravljenje Docker slike. Dockerfile se koristi za kreiranje slike, a slika se zatim koristi za pokretanje kontejnera.

Ukratko, Docker slike su građevni blokovi koji sadrže aplikaciju i njene ovisnosti, kontejneri su instance tih slika koje se mogu pokrenuti, a Dockerfiles se koriste za definiranje konfiguracije i sadržaja Docker slika.

Pokretanje servisa i aplikacija pomoću Docker-a

1. Instalirajte Docker sa oficijelne stranice
2. Otvorite terminal i provjerite da li je Docker ispravno instaliran:

   ```
   docker --version
   ```

3. Preuzmite proizvoljnu jednostavnu Docker sliku sa Docker Hub, spremište Docker slika, kao što je "hello-world" pomoću komande:

   ```
   docker pull hello-world
   ```

4. Pokrenite kontejner nakon konfigurisanja slike:

   ```
   docker run hello-world
   ```

```
PS C:\Users\Comp> docker --version
Docker version 25.0.3, build 4debf41
PS C:\Users\Comp> docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:53641cd209a4fecfc68e21a99871ce8c6920b2e7502df0a20671c6fccc73a7c6
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest

What's Next?
  1. Sign in to your Docker account → docker login
  2. View a summary of image vulnerabilities and recommendations → docker scout quickview hello-world
PS C:\Users\Comp> docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

5. Neke komande kontejnera:

   docker ps: List running containers

   docker ps -a: List all containers (including stopped ones)

   docker stop <container_id>: Stop a running container

   docker rm <container_id>: Remove a stopped container

   docker images: List downloaded Docker images

   docker rmi <image_id>: Remove a Docker image

6. Kreirajte Dockerfile koji definira konfiguracije Docker slike. Na primjer, Dockerfile za jednu Python Flask aplikaciju:

   FROM python:3.9-slim

   WORKDIR /app

   COPY . .

   RUN pip install Flask

CMD ["python", "app.py"]

7. Konfigurišite i pokrenite sliku tako što ćete kreirati direktorijum koji sadrži neku Python Flask program (app.py) i Dockerfile. Navigirajte do tog direktorijuma i u terminalu i napravite Docker sliku pomoću komande:

   docker build -t my-flask-app .

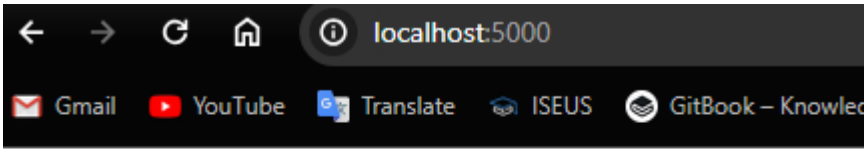
```
PS C:\Users\Comp\Downloads> cd aplikacija
PS C:\Users\Comp\Downloads\aplikacija> docker build -t my-flask-app .
[+] Building 1.2s (9/9) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                         0.0s
 => => transferring dockerfile: 145B                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                           1.1s
 => [internal] load .dockerignore                                                                            0.0s
 => => transferring context: 2B                                                                              0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:0b4b0801ae9ae61bb57bc738b1efbe4e16b927fd581774c8edfed90f0  0.0s
 => [internal] load build context                                                                            0.0s
 => => transferring context: 57B                                                                             0.0s
 => CACHED [2/4] WORKDIR /aplikacija                                                                         0.0s
 => CACHED [3/4] COPY . .                                                                                    0.0s
 => CACHED [4/4] RUN pip install Flask                                                                       0.0s
 => exporting to image                                                                                       0.0s
 => => exporting layers                                                                                      0.0s
 => => writing image sha256:baef5adaa53f12b583830288426d52185832a706c08e1627fc45c869b9cdc275                 0.0s
 => => naming to docker.io/library/my-flask-app                                                              0.0s

View build details: docker-desktop://dashboard/build/default/default/0eu4n6gkhah52zkoip87tu3u6

What's Next?
  1. Sign in to your Docker account → docker login
  2. View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\Comp\Downloads\aplikacija>
```

8. Okrenite uređeni kontejner na osnovu slike i to komandom:

   docker run -d -p 5000:5000 my-flask-app

```
PS C:\Users\Comp\Downloads\aplikacija> docker run -d -p 5000:5000 my-flask-app
3a3032a76d4c03c52c66fa5d127f87f07f392d00824e2f1b10d25199c31b5fc8
PS C:\Users\Comp\Downloads\aplikacija> docker ps
CONTAINER ID   IMAGE          COMMAND           CREATED         STATUS         PORTS                    NAMES
3a3032a76d4c   my-flask-app   "python app.py"   5 seconds ago   Up 4 seconds   0.0.0.0:5000->5000/tcp   kind_ishizaka
PS C:\Users\Comp\Downloads\aplikacija>
```

9. Pristupite aplikaciji koja je pokrenuta u kontejnetu preko web pretraživača i to na adresi http://localhost:5000.



Hello, World!

* Python Flask je razvojni okvir za izradu web aplikacija u Python-u. Dizajniran je tako da bude jednostavan i lak za korištenje, omogućavajući programerima da brzo kreiraju web aplikacije s

minimalnim šablonskim kodom. Flask pruža alate i biblioteke za rukovanje HTTP zahtjevima, usmjeravanje URL-ova i upravljanje sesijama, što ga čini idealnim za izgradnju malih i srednjih web aplikacija i API-ja.

Evo primjer jedne Python Flask aplikacije koja definira jednu rutu ("/") i vraća "Hello, World!" kada se pristupi toj ruti:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```