Docker

Differences between various os

- What is Difference between Ubuntu, Fedora, Suse, redhat, centos ,etc?
- They all contain same Linux os kernel which interacts with hardware and some extra software
- These extra softwares make each os different
- These software are
  - Different ui
  - Drivers, File Managers, Developer tools,

What are containers?

- Containers are completely isolated environments.
- They have their own processes, network interfaces, mounts ,etc just like vms except they all share same OS kernel.
- Docker uses LXC Containers

Types of name spaces

- PID
- Mount
- IPC
- CGroup
- network

- wget sudo tar -zxf rootfs.tar.gz

```
ramadevivalluru2@sivaubuntu:~$ ls -al rootfs
total 84
drwxr-xr-x 21 root             root             4096 Jan  1  1970 .
drwxr-xr-x  6 ramadevivalluru2 ramadevivalluru2 4096 Dec  8 13:00 ..
drwxr-xr-x  2 root             root             4096 Oct 15  2016 bin
drwxr-xr-x  2 root             root             4096 Sep 12  2016 boot
drwxr-xr-x  2 root             root             4096 Oct 15  2016 dev
drwxr-xr-x 56 root             root             4096 Dec  8 07:19 etc
drwxr-xr-x  2 root             root             4096 Sep 12  2016 home
drwxr-xr-x  9 root             root             4096 Sep 24  2016 lib
drwxr-xr-x  2 root             root             4096 Sep 23  2016 lib64
drwxr-xr-x  2 root             root             4096 Sep 23  2016 media
drwxr-xr-x  2 root             root             4096 Sep 23  2016 mnt
drwxr-xr-x  2 root             root             4096 Sep 23  2016 opt
drwxr-xr-x  2 root             root             4096 Sep 23  2016 proc
drwx------  2 root             root             4096 Dec  8 07:17 root
drwxr-xr-x  3 root             root             4096 Sep 23  2016 run
drwxr-xr-x  2 root             root             4096 Sep 23  2016 sbin
drwxr-xr-x  2 root             root             4096 Sep 23  2016 srv
drwxr-xr-x  2 root             root             4096 Sep 12  2016 sys
drwxrwxrwt  2 root             root             4096 Sep 24  2016 tmp
drwxr-xr-x 10 root             root             4096 Oct 15  2016 usr
drwxr-xr-x 11 root             root             4096 Oct 15  2016 var
```

Using chroot

- **Chroot** allows us to restrict a process' view of the file system

```
ramadevivalluru2@sivaubuntu:~$ sudo chroot rootfs /bin/bash
root@sivaubuntu:/# ls -al /
total 84
drwxr-xr-x 21 root root 4096 Jan  1  1970 .
drwxr-xr-x 21 root root 4096 Jan  1  1970 ..
drwxr-xr-x  2 root root 4096 Oct 15  2016 bin
drwxr-xr-x  2 root root 4096 Sep 12  2016 boot
drwxr-xr-x  2 root root 4096 Oct 15  2016 dev
drwxr-xr-x 56 root root 4096 Dec  8 07:19 etc
drwxr-xr-x  2 root root 4096 Sep 12  2016 home
drwxr-xr-x  9 root root 4096 Sep 24  2016 lib
drwxr-xr-x  2 root root 4096 Sep 23  2016 lib64
drwxr-xr-x  2 root root 4096 Sep 23  2016 media
drwxr-xr-x  2 root root 4096 Sep 23  2016 mnt
drwxr-xr-x  2 root root 4096 Sep 23  2016 opt
drwxr-xr-x  2 root root 4096 Sep 23  2016 proc
drwx------  2 root root 4096 Dec  8 07:17 root
drwxr-xr-x  3 root root 4096 Sep 23  2016 run
drwxr-xr-x  2 root root 4096 Sep 23  2016 sbin
drwxr-xr-x  2 root root 4096 Sep 23  2016 srv
drwxr-xr-x  2 root root 4096 Sep 12  2016 sys
drwxrwxrwt  2 root root 4096 Sep 24  2016 tmp
drwxr-xr-x 10 root root 4096 Oct 15  2016 usr
drwxr-xr-x 11 root root 4096 Oct 15  2016 var
root@sivaubuntu:/# which python
/usr/local/bin/python
root@sivaubuntu:/# /usr/bin/python -c 'print "Hello, container world!"'
Hello, container world!
root@sivaubuntu:/#
```

```
$ # outside of the chroot
$ top
```

```
$ sudo chroot rootfs /bin/bash
root@localhost:/# mount -t proc proc /proc
root@localhost:/# ps aux | grep top
1000     24753  0.1  0.0 156636  4404 ?        S+   22:28   0:00 top
root     24764  0.0  0.0  11132   948 ?        S+   22:29   0:00 grep top
```

- We can see the process outside of chrooted process

Running a process in separate namespace using unshare

```
$ sudo unshare -p -f --mount-proc=$PWD/rootfs/proc \
    chroot rootfs /bin/bash
root@localhost:/# ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.0  20268  3240 ?        S    22:34   0:00 /bin/bash
root          2  0.0  0.0  17504  2096 ?        R+   22:34   0:00 ps aux
root@localhost:/#
```

Running a new process inside existing namespace

```
$ # From the host, not the chroot.
$ ps aux | grep /bin/bash | grep root
...
root     29840  0.0  0.0  20272  3064 pts/5    S+   17:25   0:00 /bin/bash
```

- The kernel exposes namespaces under /proc/(PID)/ns as files. In this case, /proc/29840/ns/pid is the process namespace we're hoping to join.

```
$ sudo ls -l /proc/29840/ns
total 0
lrwxrwxrwx. 1 root root 0 Oct 15 17:31 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx. 1 root root 0 Oct 15 17:31 mnt -> 'mnt:[4026532434]'
lrwxrwxrwx. 1 root root 0 Oct 15 17:31 net -> 'net:[4026531969]'
lrwxrwxrwx. 1 root root 0 Oct 15 17:31 pid -> 'pid:[4026532446]'
lrwxrwxrwx. 1 root root 0 Oct 15 17:31 user -> 'user:[4026531837]'
lrwxrwxrwx. 1 root root 0 Oct 15 17:31 uts -> 'uts:[4026531838]'
```

- The nsenter command provides a wrapper around setns to enter a namespace.

- We'll provide the namespace file, then run the unshare to remount /proc and chroot to setup a chroot.

- This time, instead of creating a new namespace, our shell will join the existing one.

```
$ sudo nsenter --pid=/proc/29840/ns/pid \
    unshare -f --mount-proc=$PWD/rootfs/proc \
    chroot rootfs /bin/bash
root@localhost:/# ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.0  20272  3064 ?        S+   00:25   0:00 /bin/bash
root          5  0.0  0.0  20276  3248 ?        S    00:29   0:00 /bin/bash
root          6  0.0  0.0  17504  1984 ?        R+   00:30   0:00 ps aux
```