Docker introduction

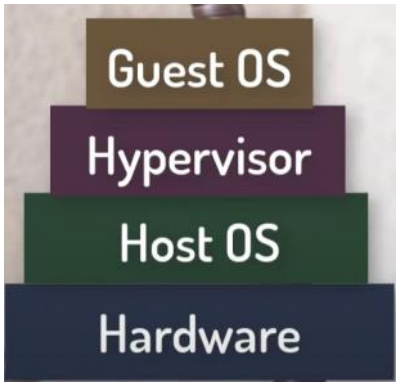
Wednesday, March 2, 2022 4:10 PM

Docker is basically an ecosystem where we can create containers and run containers

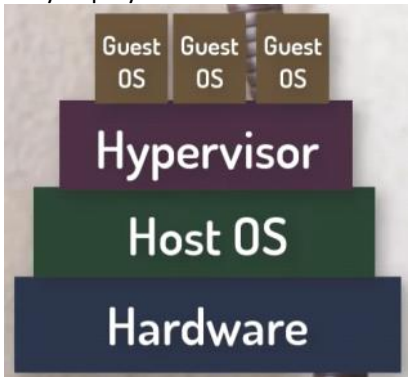
Why do we need docker?

In general , in any company we are having four stages in development of software product.

- From Development phase we send our application to the testing phase and there if we are able to clear testing phase we send it to the operation team where they deploy it as a product or the testing phase sends it back to the development phase to resolve the errors.
- Normally when we create a web application we use frameworks to help in the process, for example : we use Django while using python as the language and Spring boot while we use java as the language.
- So now we are having framework , application , dependencies and OS level features to generate an application but the problem comes in the testing where dependencies and OS level feature versions varies.
- For example : we may face a situation where project runs on our system but it won't runs on the other system, this main issue is due to change in dependencies and OS level features. For this we have introduced docker.



- We have to ship framework, dependencies and libraries from development phase to the testing phase. But we could not shift the OS details because it is resource costly for that we have find a solution called hypervisor (Virtual machines). In virtual machine when we create any project we can pack and send the software application so that other person on his virtual machine can utilize it without any error. Because here OS is not system dependent , it is common everywhere. So now when you try to send a software to the testing team, you must not send the files or application you must send the image of the virtual OS , so the can run it in their system as an instance and utilize in the testing phase without downloading the external dependencies because it is already installed on the image. And the same send to the operational phase and finally deploy the software on the servers.



- Hypervisor is still great but hypervisor is not preferred because we are creating a software by creating a virtual climate to work on using virtual machine. Now suppose think we are working on multiple applications which work on different climate and then to solve it we have to create multiple guest OS and hence it is resource greedy. So the container concept have been emerged.

- Instead of hypervisor we use docker. The difference is hypervisor need to create multiple OS on it to create an individual applications. But using dockers we can create multiple applications on the same OS but in different containers. These containers can have it's own dependencies , extensions etc. And after developing the application as we need to send it to the testing phase similar to the Hypervisor we create an image here. And send it to testing phase, they create multiple containers (instances) and test them and send that image to the production phase.

What Is container?

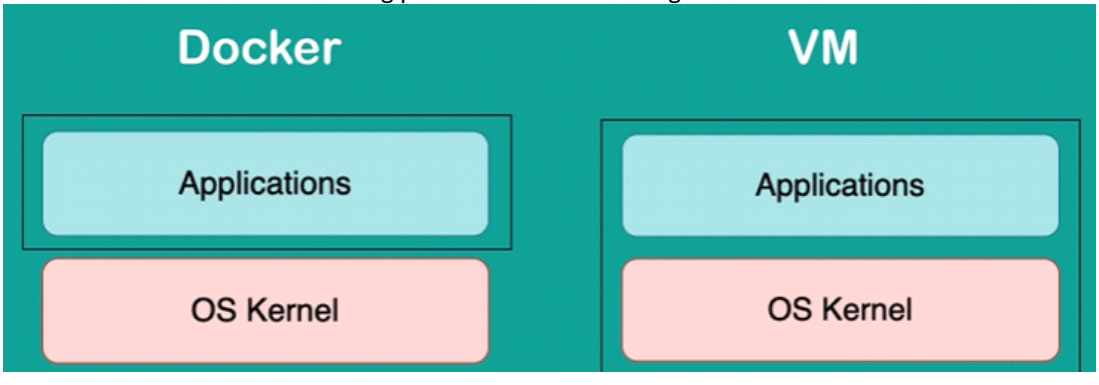
- A way to package application with all the necessary dependencies and configuration, it is portable and easily shared and moved around. Makes development and deployment more efficient.
- Container resides in container repository for each company there will be private depository or we can store in in public repository like docker hub.
- If we want to use container we no need to include any separate operating system because container itself contains an own isolated environment. So we no need to download any dependencies or packages individually it will be auto downloaded if we have container image executed. Containers packaged with all needed configuration.
- Command which runs the docker file "cmd docker run postgres:version"
- Run same app with different versions using containers.
- If there is any misunderstanding in getting rules while making the application between the developer team and operation team in generic process without the containers then there would be problem while deployment. After having containers all those jar files configuration and dependencies are encapsulated and kept in container and no environmental configuration needed on server except Docker Runtime, we can directly deploy.
- Here developers and operation team work together to package the application in a container.
- In general container is a layer of images, here images are mostly Linux based images because they are small in size, application image will be on top usage (postgres). Alpine will be the least used image which is also a Linux base image.
- Dockers have multiple layers , so when we run the docker command multiple layers individually downloaded and only layers which are not already downloaded in local system, take time to download.

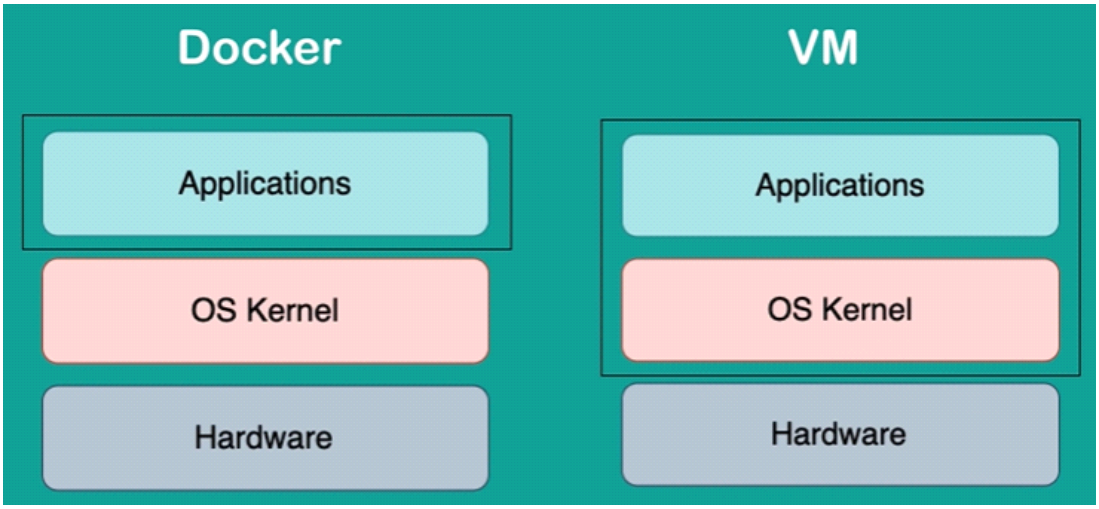
Docker Image:

- It is the actual package which packs the configuration , PostgreSQL and the script. And it is an artifact that can be moved from any system to other system.

Docker Container:

- It is the one which actually start the application, here the image is taken and container environment is created in the system where it contains the dependencies , extensions and packages already imported which are necessary to run the environment.
- In shortcut this is the running phase of the Docker image.





Docker vs Virtual machine

- Docker virtualizes the application layer
- Virtual machine virtualizes the application layer and OS Kernel
- Docker images are smaller compared to the Virtual Machine images
- Docker containers start and run much faster
- VM of any OS can run on any OS host

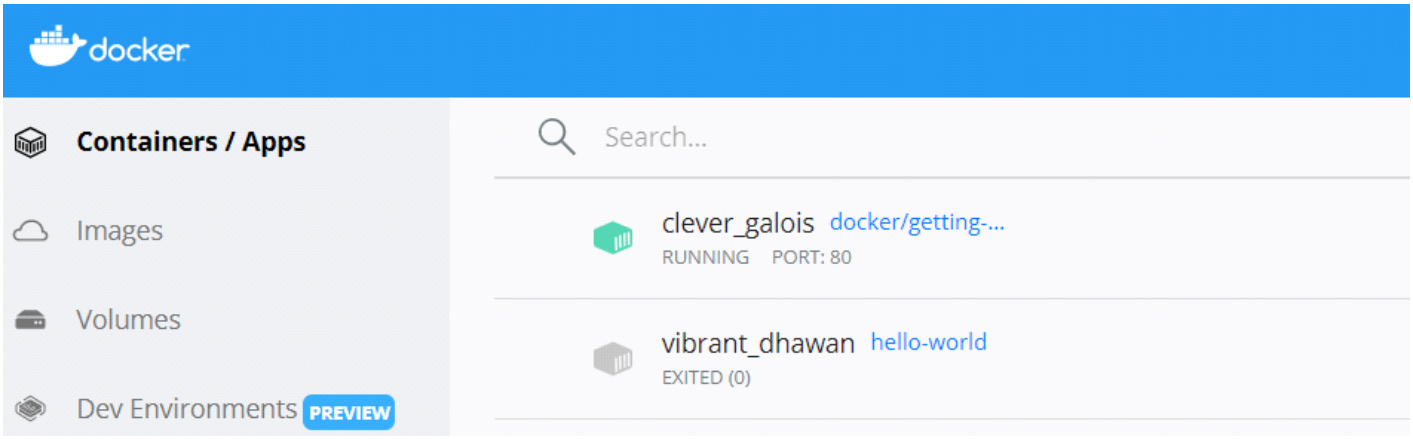
We could not run a Linux based docker image application on the windows kernel hardware

Docker installation:

- https://youtu.be/3c-iBn73dDE
- Refer this link and download docker but not docker toolbox and current system that you are using is windows 10. Docker will support all the versions which are >=10
- If you want to check the windows version then run> winver then we would get the version of the windows.
- And then you can open your windows power shell
- And execute the following command

```
PS C:\Users\JEEVAN SAI> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:97a379f4f88575512824f3b352bc03cd75e239179eea0fecc38e597b2209f49a
Status: Downloaded newer image for hello-world:latest

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/
```

- If we get such output , we can say that docker is installed in the system and check the corresponding docker image that is generated in docker desktop GUI


docker
Containers / Apps
Images
Volumes
Dev Environments
PREVIEW
Search...
clever_galois
docker/getting-...
RUNNING
PORT: 80
vibrant_dhawan
hello-world
EXITED (0)

Basic Docker Commands

03 March 2022 07:56

- Container is a running environment for image
- Container is binded for the port5000
- Container has it's independent file system which is independent to the host system.
- "docker pull redis" this command is used to pull the redis image into our system
- We can execute this command in our windows power shell to create a container in the docker.
- If we try to run this command without opening docker desktop we would have such error as it is not running in background.

```
PS C:\Users\JEEVAN SAI> docker pull redis
Using default tag: latest
error during connect: This error may indicate that the docker daemon is not running.: Post "http://%2F%2F.%2Fpipe%2Fdock
er_engine/v1.24/images/create?fromImage=redis&tag=latest": open //./pipe/docker_engine: The system cannot find the file
specified.
```

- To check all the existing images in the docker we need to use the following command

```
PS C:\Users\JEEVAN SAI> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    2b4cba85892a   10 hours ago   72.8MB
hello-world   latest    feb5d9fea6a5   5 months ago   13.3kB
```

- The image version is also mentioned in the details (TAG)
- If we want to create a container , we need to run the image in the container. For that , we need to do the following command in the powershell

```
PS C:\Users\JEEVAN SAI> docker run redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
f7a1c6dad281: Pull complete
c5f81eaec564: Pull complete
2be237d3defa: Pull complete
1640a11de2e5: Pull complete
9138edee9512: Pull complete
c62664237d8c: Pull complete
Digest: sha256:94a9137659d8e103a41e6d635b8a63d2d341dac6171ed8ea16849962ea67d2ae
Status: Downloaded newer image for redis:latest
1:C 04 Mar 2022 06:37:26.610 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 04 Mar 2022 06:37:26.610 # Redis version=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 04 Mar 2022 06:37:26.610 # Warning: no config file specified, using the default config. In order to specify a con
fig file use redis-server /path/to/redis.conf
1:M 04 Mar 2022 06:37:26.611 * monotonic clock: POSIX clock_gettime
1:M 04 Mar 2022 06:37:26.612 * Running mode=standalone, port=6379.
1:M 04 Mar 2022 06:37:26.612 # Server initialized
1:M 04 Mar 2022 06:37:26.612 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condi
tion. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl
 vm.overcommit_memory=1' for this to take effect.
1:M 04 Mar 2022 06:37:26.613 * Ready to accept connections
```

- Container is running environment of the image.
- If we want to get the list of all running docker containers , we need to use the following command

```
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS      NAMES
83b0dc43ce47   redis     "docker-entrypoint.s…"   40 seconds ago   Up 37 seconds   6379/tcp   gracious_lewin
```

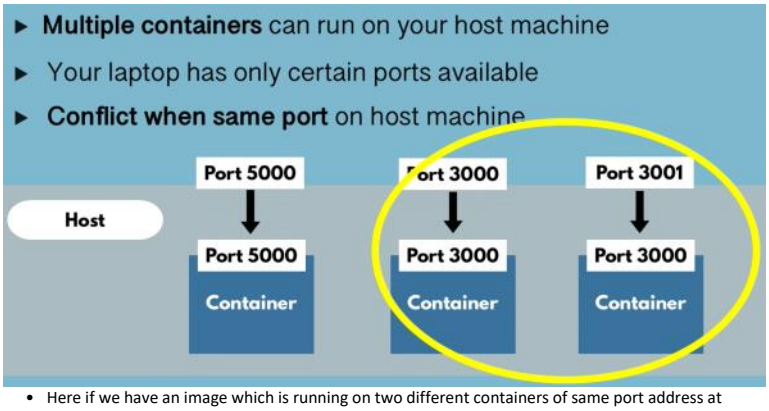
- If we want to halt the container that we have started to execute using docker run command we can simply use the ctrl + c, if this not worked use docker stop command it will definitely work.
- If we want to start the container and return the container id where it is created an runs, we use the following command. Here d is detached mode.

```
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS      NAMES
83b0dc43ce47   redis     "docker-entrypoint.s…"   40 seconds ago   Up 37 seconds   6379/tcp   gracious_lewin
PS C:\Users\JEEVAN SAI> docker run -d redis
498080545bf70db2987be2caa59fa2eccd4b09ee30490a5fabd2975d0312236a
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS      NAMES
498080545bf7   redis     "docker-entrypoint.s…"   9 seconds ago        Up 6 seconds        6379/tcp   focused_agnesi
83b0dc43ce47   redis     "docker-entrypoint.s…"   About a minute ago   Up About a minute   6379/tcp   gracious_lewin
```

- If we want to restart or stop the container of specific container id then use the following commands

```
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS      NAMES
498080545bf7   redis     "docker-entrypoint.s…"   9 seconds ago        Up 6 seconds        6379/tcp   focused_agnesi
83b0dc43ce47   redis     "docker-entrypoint.s…"   About a minute ago   Up About a minute   6379/tcp   gracious_lewin
PS C:\Users\JEEVAN SAI> docker stop 498080545bf7
498080545bf7
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS      NAMES
83b0dc43ce47   redis     "docker-entrypoint.s…"   3 minutes ago   Up 2 minutes   6379/tcp   gracious_lewin
PS C:\Users\JEEVAN SAI> docker start 498080545bf7
498080545bf7
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS         PORTS      NAMES
498080545bf7   redis     "docker-entrypoint.s…"   About a minute ago   Up 5 seconds   6379/tcp   focused_agnesi
83b0dc43ce47   redis     "docker-entrypoint.s…"   3 minutes ago        Up 3 minutes   6379/tcp   gracious_lewin
PS C:\Users\JEEVAN SAI> docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED             STATUS                          PORTS      NAME
S
498080545bf7   redis         "docker-entrypoint.s…"   2 minutes ago       Up 23 seconds                   6379/tcp   focu
sed_agnesi
83b0dc43ce47   redis         "docker-entrypoint.s…"   3 minutes ago       Up 3 minutes                    6379/tcp   grac
ious_lewin
c3976c9aa3fd   redis         "docker-entrypoint.s…"   5 minutes ago       Exited (0) 4 minutes ago                   daza
ling_mclean
ff915f4220ee   redis         "docker-entrypoint.s…"   19 minutes ago      Exited (0) 5 minutes ago                   upbe
at_poincare
bc16927d9dfb   hello-world   "/hello"                 About an hour ago   Exited (0) About an hour ago               conf
ident_cray
4dbf7c8e8667   ubuntu        "bash"                   2 hours ago         Exited (130) 2 hours ago                   happ
y_leavitt
e109696f4778   hello-world   "/hello"                 2 hours ago         Exited (0) 2 hours ago                     fros
ty_bartik
```

- Here "docker ps -a " shows the history of what dockers run previously. Which is helpful to easily identify the recently used container id.
- We can run two containers of same image of different versions and execute at the same port id without clash.



- Here if we have an image which is running on two different containers of same port address at the container side. And here the image is same but it is of different versions , so it is kept in different containers, but it will have the same port address. When we try to run on our host machine , we need to keep it on local host , so if we do that binding then there will not be any issue. They mapping can be given through command line.
- We are having these two containers running currently

```
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS      NAMES
cec56dbb2f20   redis:6.2.6    "docker-entrypoint.s…"   About a minute ago   Up About a minute   6379/tcp   stoic_cray
498080545bf7   redis          "docker-entrypoint.s…"   10 minutes ago       Up 8 minutes        6379/tcp   focused_agnesi
```

- Now we need to map them to the host ports

```
PS C:\Users\JEEVAN SAI> docker run -p6000:6379 redis
1:C 04 Mar 2022 07:07:02.502 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 04 Mar 2022 07:07:02.502 # Redis version=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 04 Mar 2022 07:07:02.502 # Warning: no config file specified, using the default config. In order to specify a config
 file use redis-server /path/to/redis.conf
1:M 04 Mar 2022 07:07:02.503 * monotonic clock: POSIX clock_gettime
1:M 04 Mar 2022 07:07:02.504 * Running mode=standalone, port=6379.
1:M 04 Mar 2022 07:07:02.504 # Server initialized
1:M 04 Mar 2022 07:07:02.504 # WARNING overcommit_memory is set to 0! Background save may fail under low memory conditio
n. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.ov
ercommit_memory=1' for this to take effect.
1:M 04 Mar 2022 07:07:02.505 * Ready to accept connections
```

- Here we can see , this step is used to map 6000 host address to the already assigned 6379 container address.
- Now check it , the mapping done

```
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS          PORTS                                       NAMES
49c36c25d138   redis          "docker-entrypoint.s…"   About a minute ago   Up 59 seconds   0.0.0.0:6000->6379/tcp, :::6000->6379/tcp   jovial_thompson
cec56dbb2f20   redis:6.2.6    "docker-entrypoint.s…"   4 minutes ago        Up 4 minutes    6379/tcp                                    stoic_cray
498080545bf7   redis          "docker-entrypoint.s…"   13 minutes ago       Up 11 minutes   6379/tcp                                    focused_agnesi
```

- So instead of using run directly , if we are using multiple versions of the same image then we must use the following syntax to run "docker run -p6000:6379 redis". Then 6000 host address is mapped to the container address.
- In similar way let's create another redix and remove the other redix without mapping.
- Now we have to execute the second mapping from the container to the host address. Host address should not be common, container address may be common.
- Now if we try to map 6379 to the 6000 then check we would get some error , so we are mapping it with 6001 instead of 6000.

```
[~]$ docker run -p6000:6379 redis:4.0
docker: Error response from daemon: driver failed programming external connectivity on endpoint sleepy_visvesvaraya
24020c6befbe0447048 7a6c): Bind for 0.0.0.0:6000 failed: port is already allocated.
ERRO[0000] error getting events from daemon: net/http: request canceled
[~]$ docker run -p6001:6379 redis:4.0
1:C 26 Oct 12:15:00.705 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 26 Oct 12:15:00.705 # Redis version=4.0.14, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 26 Oct 12:15:00.705 # Warning: no config file specified, using the default config. In order to specify a config
1:M 26 Oct 12:15:00.706 * Running mode=standalone, port=6379.
1:M 26 Oct 12:15:00.706 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/soma
1:M 26 Oct 12:15:00.706 # Server initialized
1:M 26 Oct 12:15:00.706 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will cr
Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add
in the setting after a reboot. Redis must be restarted after THP is disabled.
1:M 26 Oct 12:15:00.706 * Ready to accept connections
^C1:signal-handler (1572092102) Received SIGINT scheduling shutdown...
1:M 26 Oct 12:15:02.828 # User requested shutdown...
1:M 26 Oct 12:15:02.828 * Saving the final RDB snapshot before exiting.
1:M 26 Oct 12:15:02.832 * DB saved on disk
1:M 26 Oct 12:15:02.832 # Redis is now ready to exit, bye bye...
[~]$ docker run -p6001:6379 redis:4.0
```

```
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS          PORTS                                       NAMES
49c36c25d138   redis          "docker-entrypoint.s…"   About a minute ago   Up 59 seconds   0.0.0.0:6000->6379/tcp, :::6000->6379/tcp   jovial_thompson
cec56dbb2f20   redis:6.2.6    "docker-entrypoint.s…"   4 minutes ago        Up 4 minutes    6379/tcp                                    stoic_cray
498080545bf7   redis          "docker-entrypoint.s…"   13 minutes ago       Up 11 minutes   6379/tcp                                    focused_agnesi
PS C:\Users\JEEVAN SAI> docker run -p6001:6379 redis:6.2.6
1:C 04 Mar 2022 07:12:20.506 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 04 Mar 2022 07:12:20.506 # Redis version=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 04 Mar 2022 07:12:20.506 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/
1:M 04 Mar 2022 07:12:20.507 * monotonic clock: POSIX clock_gettime
1:M 04 Mar 2022 07:12:20.509 * Running mode=standalone, port=6379.
1:M 04 Mar 2022 07:12:20.509 # Server initialized
1:M 04 Mar 2022 07:12:20.509 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.ove
 and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 04 Mar 2022 07:12:20.510 * Ready to accept connections
```

Now we have created two mappings and let's element the container operations without mappings.
-p[host address]:[container address]
Now check we created two separate mappings

```
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS                                       NAMES
e1e3e821ae32   redis:6.2.6    "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:6001->6379/tcp, :::6001->6379/tcp   condescending_kare
49c36c25d138   redis          "docker-entrypoint.s…"   7 minutes ago        Up 7 minutes        0.0.0.0:6000->6379/tcp, :::6000->6379/tcp   jovial_thompson
cec56dbb2f20   redis:6.2.6    "docker-entrypoint.s…"   10 minutes ago       Up 10 minutes       6379/tcp                                    stoic_cray
498080545bf7   redis          "docker-entrypoint.s…"   19 minutes ago       Up 17 minutes       6379/tcp                                    focused_agnesi
PS C:\Users\JEEVAN SAI> docker stop cec56dbb2f20
cec56dbb2f20
PS C:\Users\JEEVAN SAI> docker stop 498080545bf7
498080545bf7
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                       NAMES
e1e3e821ae32   redis:6.2.6    "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes    0.0.0.0:6001->6379/tcp, :::6001->6379/tcp   condescending_kare
49c36c25d138   redis          "docker-entrypoint.s…"   7 minutes ago    Up 7 minutes    0.0.0.0:6000->6379/tcp, :::6000->6379/tcp   jovial_thompson
```

Pruning unused Docker Objects

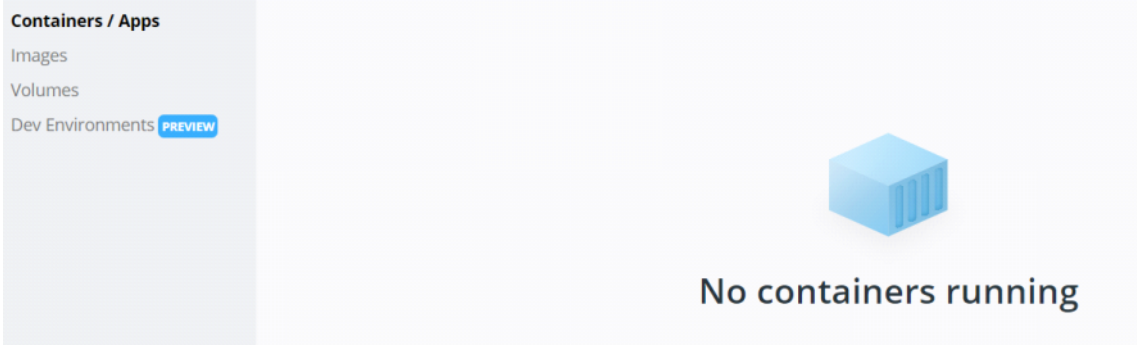
- Docker takes a conservative approach to cleaning up unused objects such as images, containers, volumes and networks these are not removed unless we say to do it. Which in turn take more disk space due to docker. So docker provided prune command.

`docker image prune`	This allows to clean up un used images, by default it only cleans up dangling images. Dangling images mean which is not tagged and is not referenced by any container
`docker image prune -a`	To remove all images which are not used by existing containers

- We can escape the prompt using -f or --force flag
- Even we can give filtering using following command
- `$ docker image prune -a --filter "until=24h"`
- To remove all the stopped containers we have to use the following commands
- `$ docker container prune`
- `$ docker container prune --filter "until=24h"`
- The following command helps us to remove the containers that we have created it will delete all the containers even if it is running state or existed state.

```
PS C:\Users\JEEVAN SAI> docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED             STATUS                         PORTS     NAMES
e1e3e821ae32   redis:6.2.6   "docker-entrypoint.s…"   53 minutes ago      Exited (0) 46 minutes ago                conde
scending_kare
49c36c25d138   redis         "docker-entrypoint.s…"   58 minutes ago      Exited (0) 46 minutes ago                jovia
l_thompson
cec56dbb2f20   redis:6.2.6   "docker-entrypoint.s…"   About an hour ago   Exited (0) 51 minutes ago                stoic
_cray
498080545bf7   redis         "docker-entrypoint.s…"   About an hour ago   Exited (0) 50 minutes ago                focus
ed_agnesi
83b0dc43ce47   redis         "docker-entrypoint.s…"   About an hour ago   Exited (0) About an hour ago             graci
ous_lewin
c3976c9aa3fd   redis         "docker-entrypoint.s…"   About an hour ago   Exited (0) About an hour ago             dazzl
ing_mclean
ff915f4220ee   redis         "docker-entrypoint.s…"   About an hour ago   Exited (0) About an hour ago             upbea
t_poincare
bc16927d9dfb   hello-world   "/hello"                 2 hours ago         Exited (0) 2 hours ago                   confi
dent_cray
4dbf7c8e8667   ubuntu        "bash"                   3 hours ago         Exited (130) 3 hours ago                 happy
_leavitt
e109696f4778   hello-world   "/hello"                 3 hours ago         Exited (0) 3 hours ago                   frost
y_bartik
PS C:\Users\JEEVAN SAI> docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
e1e3e821ae32a445d16173ec69477d4b71110f3608226b6502ad97a5b2bdc679
49c36c25d13881ef266b07a96cb53f99fe8c0d7a14c48336ccea1b8214386df7
cec56dbb2f2034cdc8f1a2dcc0f214babee391ea026a99bcb588f1ea9732c1f3
498080545bf70db2987be2caa59fa2eccd4b09ee30490a5fabd2975d0312236a
83b0dc43ce47d4af13e2371c92152137bd15ad1c96ea27368c31111a0ef52f1e
c3976c9aa3fd50ae98f21c89f60abd70dcd4a3190e4ff5beed31fb4ba4307a87
ff915f4220ee1a1e68587d7f4d71ca367c2f112a57d2bacf258afaf54b951b89
bc16927d9dfb5c642655ab6bc01ef39a2aa4ff7c8fa40e11f53217bec09224cb
4dbf7c8e8667676eed2228b65ae19f9b001085a8dd07f8e85a18cc02be308366
e10696f4778d8ef1eed0183e76f552ffe46b10d7109c171bc3ca825b7d7aa35

Total reclaimed space: 7B
PS C:\Users\JEEVAN SAI> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

- All the Containers are deleted



- Remember , after clearing the container we can delete the image. But not the vice versa. So now previously we have cleared the containers and now we are going to clear the images.

```
PS C:\Users\JEEVAN SAI> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    2b4cba85892a   12 hours ago   72.8MB
redis         6.2.6     0e403e3816e8   20 hours ago   113MB
redis         latest    0e403e3816e8   20 hours ago   113MB
hello-world   latest    feb5d9fea6a5   5 months ago   13.3kB
PS C:\Users\JEEVAN SAI> docker image prune -a
WARNING! This will remove all images without at least one container associated to them.
Are you sure you want to continue? [y/N] y
Deleted Images:
untagged: hello-world:latest
untagged: hello-world@sha256:97a379f4f88575512824f3b352bc03cd75e239179eea0fecc38e597b2209f49a
deleted: sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412
deleted: sha256:e07ee1baac5fae6a26f30cabfe54a36d3402f96afda318fe0a96cec4ca393359
untagged: redis:6.2.6
untagged: redis:latest
untagged: redis@sha256:94a9137659d8e103a41e6d635b8a63d2d341dac6171ed8ea16849962ea67d2ae
deleted: sha256:0e403e3816e890f6edc35de653396a5f379084e5ee6673a7608def32caec6c90
deleted: sha256:aadcd8c698693e711a59da1ea17d6272e82978b706e31749aab6da9ffbcb05b9
deleted: sha256:9ec29b6942383e968e59a8f3db90620636d0d922b55fa0a35b8153d565350936
deleted: sha256:061f675ca55acabf23b66dceceebca4d749e47bb5a23e07e6734923b619fb310
deleted: sha256:58bfa500e04e12abd7ab08c8f7eeb6722dc5c37f2062e7815b24a571ef10ad19
deleted: sha256:88a52cb94016f778cdafceea920e5a3dd423e606e8ae9e1c4888f03d82b8e752
deleted: sha256:1401df2b50d5de5a743b7bac3238ef3b7ce905ae39f54707b0ebb8eda3ab10bc
untagged: ubuntu:latest
untagged: ubuntu@sha256:8ae9bafbb64f63a50caab98fd3a5e37b3eb837a3e0780b78e5218e63193961f9
deleted: sha256:2b4cba85892afc2ad8ce258a8e3d9daa4a1626ba380677cee93ef2338da442ab
deleted: sha256:68a85fa9d77ecac87de23805c4be8766bda08a86787e324036cbcf84b62640fa

Total reclaimed space: 185.4MB
PS C:\Users\JEEVAN SAI> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Prune volumes

- Volumes can be used by one or more containers, and take up space on the Docker host.
- Volumes are never removed automatically, because to do so could destroy data.

```
$ docker volume prune

WARNING! This will remove all volumes not used by at least one container.
```

```
$ docker volume prune --filter "label!=keep"
```

Prune Everything

```
$ docker system prune

WARNING! This will remove:
        - all stopped containers
        - all networks not used by at least one container
        - all dangling images
        - all build cache
Are you sure you want to continue? [y/N] y
```

To also prune volumes, add the `--volumes` flag:

```
$ docker system prune --volumes

WARNING! This will remove:
        - all stopped containers
        - all networks not used by at least one container
        - all volumes not used by at least one container
        - all dangling images
        - all build cache
Are you sure you want to continue? [y/N] y
```

If we want to get logs of any image use this following command

```
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                       NAMES
2c504f8e785c   redis     "docker-entrypoint.s…"   54 seconds ago   Up 47 seconds   0.0.0.0:6000->6379/tcp, :::6000->6379/tcp   affectionate_mcclintock
PS C:\Users\JEEVAN SAI> docker logs affectionate_mcclintock
1:C 04 Mar 2022 08:24:30.688 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 04 Mar 2022 08:24:30.688 # Redis version=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 04 Mar 2022 08:24:30.688 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 04 Mar 2022 08:24:30.689 * monotonic clock: POSIX clock_gettime
1:M 04 Mar 2022 08:24:30.690 * Running mode=standalone, port=6379.
1:M 04 Mar 2022 08:24:30.690 # Server initialized
1:M 04 Mar 2022 08:24:30.690 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory
= 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 04 Mar 2022 08:24:30.691 * Ready to accept connections
```

Here we are creating our own named image which is the same type of the previous

```
PS C:\Users\JEEVAN SAI> docker run -p6001:6379 --name redis-older redis
1:C 04 Mar 2022 08:29:25.950 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 04 Mar 2022 08:29:25.950 # Redis version=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 04 Mar 2022 08:29:25.950 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 04 Mar 2022 08:29:25.951 * monotonic clock: POSIX clock_gettime
1:M 04 Mar 2022 08:29:25.954 * Running mode=standalone, port=6379.
1:M 04 Mar 2022 08:29:25.954 # Server initialized
1:M 04 Mar 2022 08:29:25.954 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory
= 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 04 Mar 2022 08:29:25.955 * Ready to accept connections
```

```
PS C:\Users\JEEVAN SAI> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS                                       NAMES
169541678ef9   redis     "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:6001->6379/tcp, :::6001->6379/tcp   redis-older
2c504f8e785c   redis     "docker-entrypoint.s…"   6 minutes ago        Up 6 minutes        0.0.0.0:6000->6379/tcp, :::6000->6379/tcp   affectionate_mcclintock
```

Now we can use that name we created and do our operations instead of container_id

```
PS C:\Users\JEEVAN SAI> docker logs redis-older
1:C 04 Mar 2022 08:29:25.950 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 04 Mar 2022 08:29:25.950 # Redis version=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 04 Mar 2022 08:29:25.950 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 04 Mar 2022 08:29:25.951 * monotonic clock: POSIX clock_gettime
1:M 04 Mar 2022 08:29:25.954 * Running mode=standalone, port=6379.
1:M 04 Mar 2022 08:29:25.954 # Server initialized
1:M 04 Mar 2022 08:29:25.954 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory
 = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 04 Mar 2022 08:29:25.955 * Ready to accept connections
```

- **If we want to navigate into a particular image and navigate in that container . Use this command.**

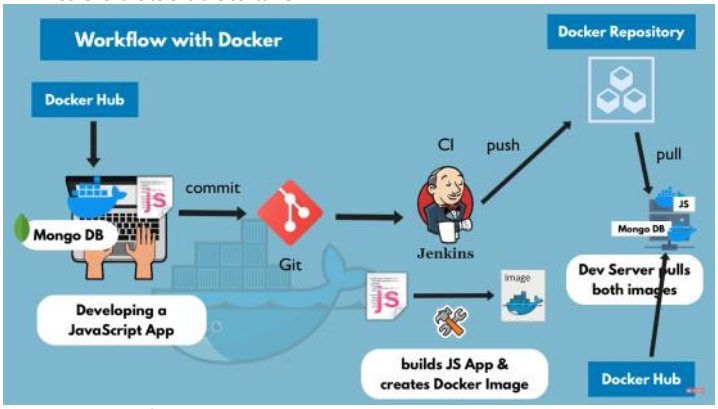
```
PS C:\Users\JEEVAN SAI> docker exec -it redis-older /bin/bash
root@169541678ef9:/data#
```

We can check what environmental variables are set into

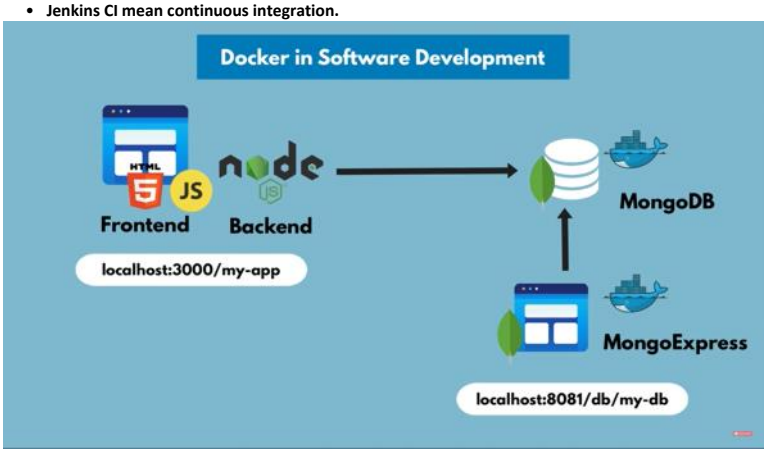
```
PS C:\Users\JEEVAN SAI> docker exec -it redis-older /bin/bash
root@169541678ef9:/data# env
HOSTNAME=169541678ef9
REDIS_DOWNLOAD_SHA=5b2b8b7a50111ef395bf1c1d5be11e6e167ac018125055daa8b5c2317ae131ab
PWD=/data
HOME=/root
REDIS_VERSION=6.2.6
GOSU_VERSION=1.14
TERM=xterm
REDIS_DOWNLOAD_URL=http://download.redis.io/releases/redis-6.2.6.tar.gz
SHLVL=1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
_=/usr/bin/env
root@169541678ef9:/data#
```

We couldn't execute every Linux command here but we can run most of the docker commands

- **Docker start doesn't work on the images , it works on the containers.**
- **Docker run is to create a new container**
- **Docker start is to restart the container**
- **Docker end is to end the container**

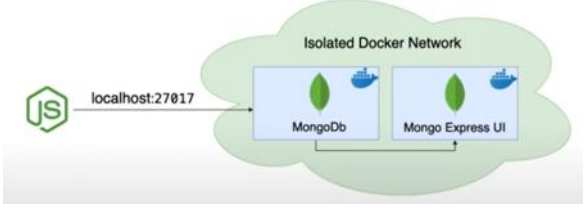


- **Here initially if we assume that we are working on a project where we use mango dB as the database and JavaScript as the front end logic then when we want to send it to the texter first we commit in the git repository and the files get build in the Jenkins and create a docker image , which is send to the docker repository to store which is retrieved by the company servers to test . Here a docker image will be splitted into two containers one is mango DB and other is JS and if there is no errors it will be sent to docker hub.**
- **Jenkins CI mean continuous integration.**



- **Here as we need to use mango DB as our database we need to pull mango and mango Express into our system through docker containers and use it locally. And after pulling we need to connect it to the front end.**
- **To connect we do the following, for establishing communication between mangodb and mango express there is no necessity in doing external communication because they both are in the same isolated docker network. And if front end like JavaScript want to communicate with the database then it communicate through local host address now at the development stage the isolated docker network and node js is in separate files. But when we send to testing in a package it will be send as an image where NodeJS mangodb and mango express will be in the same isolated docker network.**

HOST



- **Now we need to create a network that is used by mango db and mango express. Use the following command.**

```
[~]$ docker network create mongo-network
70eb8e9f1c1eac66312912d30422e4acf3ade947736b0ca833734b030798a712
[~]$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ef884144b3e9        bridge              bridge              local
484f006515ef        docker_default      bridge              local
fa524e351841        host                host                local
70eb8e9f1c1e        mongo-network       bridge              local
96afbbc49358        none                null                local
[~]$
```

- **And now we need to create a container to build the network for mango dB and create a port for JavaScript communication. And in the below step we are creating the user name and password for mango db.**

```
[simple-js-app]$ docker run -d \
> -p 27017:27017 \
> -e MONGO_INITDB_ROOT_USERNAME=admin \
> -e MONGO_INITDB_ROOT_PASSWORD=password \
> --name mongodb \
> --net mongo-network \
> mongo
```

- **Up to now we have created mango dB now we need to connect it to mango express.**
- **In default mango dB runs on 27017:27017**
- **Mango dB runs on 8081:8081**
- **Host address : container address**

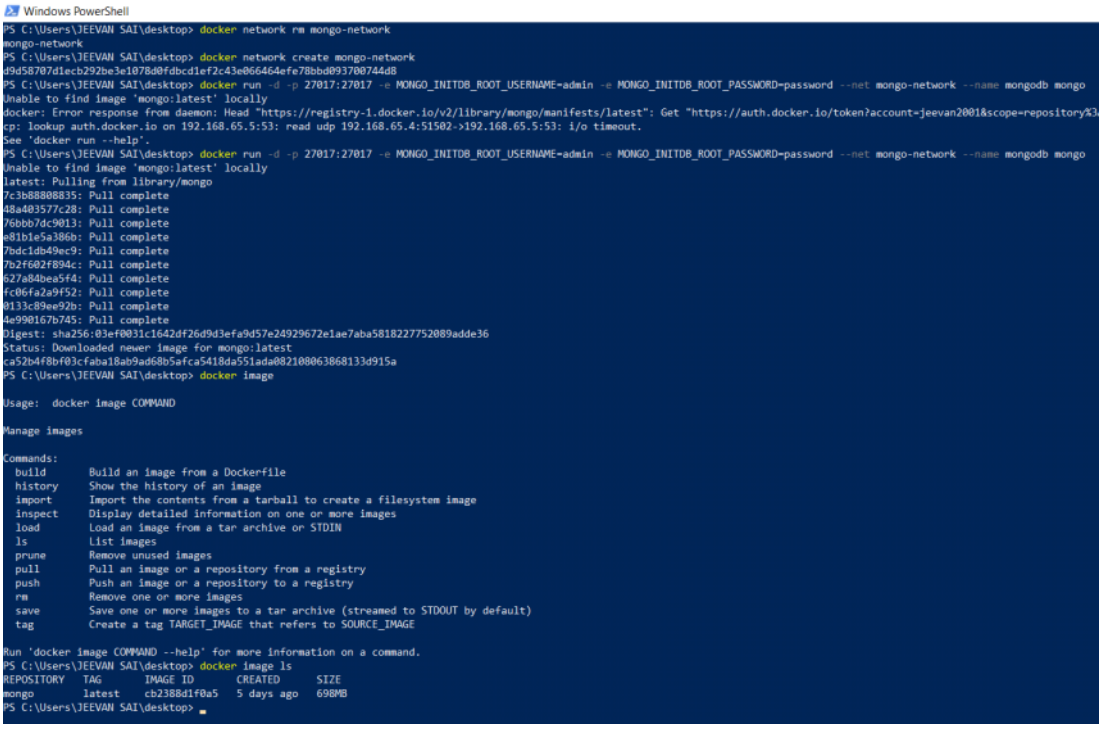
```
[simple-js-app]$ docker run -d \
> -p 8081:8081 \
> -e ME_CONFIG_MONGODB_ADMINUSERNAME=admin \
> -e ME_CONFIG_MONGODB_ADMINPASSWORD=password \
> --net mongo-network \
> --name mongo-express \
> -e ME_CONFIG_MONGODB_SERVER=mongodb \
> mongo-express
```

m

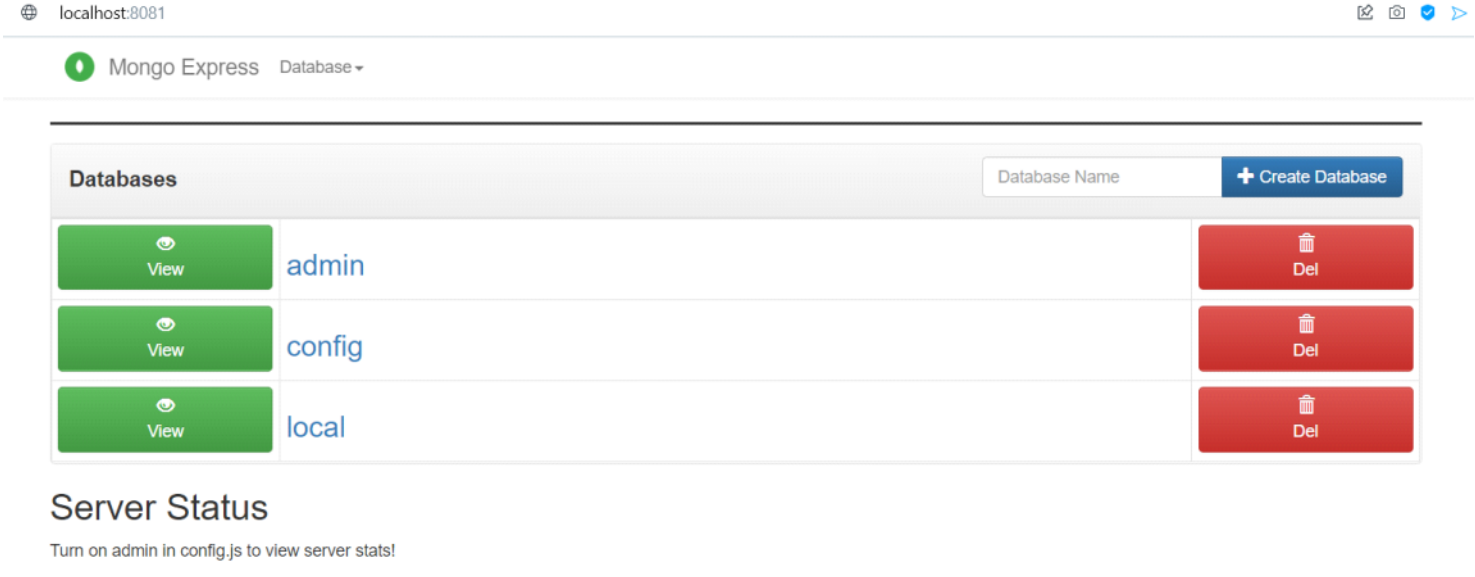
- **Now we need to run mango-express and that mango express runs on 8081:8081. we are providing user name and password of mango db there and utilizing the already created network that is mango network and name the current container as mango-express. Connecting the mango express to the mango db server name the whole container as mango-express**
- **The last line in both the commands is the name of the image that we want to create**

```
# commands

## create docker network
docker network create mongo-network

## start mongodb
docker run -d \
-p 27017:27017 \
-e MONGO_INITDB_ROOT_USERNAME=admin \
-e MONGO_INITDB_ROOT_PASSWORD=password \
--net mongo-network \
--name mongodb \
mongo

## start mongo-express
docker run -d \
-p 8081:8081 \
-e ME_CONFIG_MONGODB_ADMINUSERNAME=admin \
-e ME_CONFIG_MONGODB_ADMINPASSWORD=password \
-e ME_CONFIG_MONGODB_SERVER=mongodb \
--net mongo-network \
--name mongo-express \
mongo-express
```

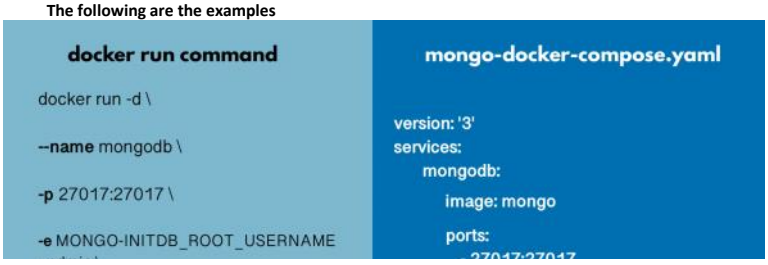
- **When I try to run the above commands we get the following output**
- **First we have to create a mongo network to communicate between mongodb and mongo express**
- **Now we need to start the mongodb container and then mongo express**
- **We get the following outputs**

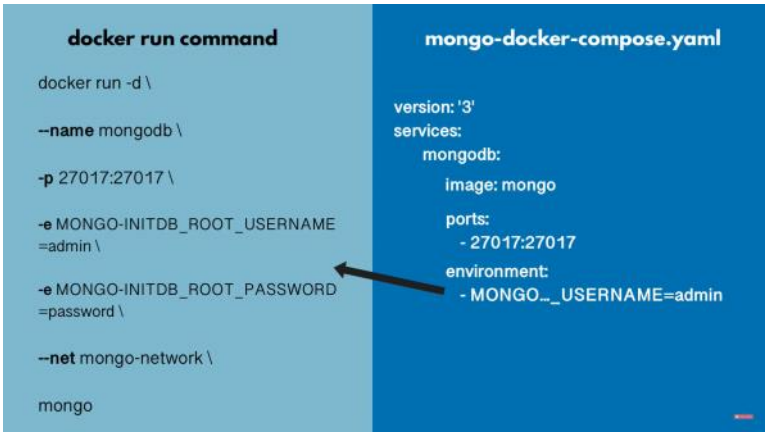


- **Now if we open the browser and go to localhost:8081 we can observe mongodb has been generated locally.**



- **Instead of running docker run command we have other alternative that is docker compose . The following are the examples**





Mango.yaml (docker compose file)

```yaml
version: '3'
services:
  mongodb:
    image: mongo
    ports:
     - 27017:27017
    environment:
     - MONGO_INITDB_ROOT_USERNAME=admin
     - MONGO_INITDB_ROOT_PASSWORD=password
  mongo-express:
    image: mongo-express
    ports:
     - 8080:8081
    environment:
     - ME_CONFIG_MONGODB_ADMINUSERNAME=admin
     - ME_CONFIG_MONGODB_ADMINPASSWORD=password
     - ME_CONFIG_MONGODB_SERVER=mongodb
```

To run the docker yaml file use the following command

```
[my-app]$ docker-compose -f mongo.yaml up
Creating network "myapp_default" with the default driver
Creating myapp_mongodb_1
Creating myapp_mongo-express_1
Attaching to myapp_mongo-express_1, myapp_mongodb_1
mongo-express_1  | Waiting for mongodb:27017...
mongo-express_1  | /docker-entrypoint.sh: connect: Connection refused
mongo-express_1  | /docker-entrypoint.sh: line 14: /dev/tcp/mongodb/27017: Connection refused
mongodb_1        | about to fork child process, waiting until server is ready for connections.
mongodb_1        | forked process: 27
mongodb_1        | 2019-11-08T16:47:28.102+0000 I  CONTROL  [main] ***** SERVER RESTARTED *****
mongodb_1        | 2019-11-08T16:47:28.108+0000 I  CONTROL  [main] Automatically disabling TLS
fy --sslDisabledProtocols 'none'
mongodb_1        | 2019-11-08T16:47:28.115+0000 I  CONTROL  [initandlisten] MongoDB starting :
64-bit host=887a135686a6
mongodb_1        | 2019-11-08T16:47:28.115+0000 I  CONTROL  [initandlisten] db version v4.2.1
mongodb_1        | 2019-11-08T16:47:28.115+0000 I  CONTROL  [initandlisten] git version: edf6d
mongodb_1        | 2019-11-08T16:47:28.116+0000 I  CONTROL  [initandlisten] OpenSSL version: O
mongodb_1        | 2019-11-08T16:47:28.116+0000 I  CONTROL  [initandlisten] allocator: tcmallo
mongodb_1        | 2019-11-08T16:47:28.116+0000 I  CONTROL  [initandlisten] modules: none
mongodb_1        | 2019-11-08T16:47:28.116+0000 I  CONTROL  [initandlisten] build environment:
mongodb_1        | 2019-11-08T16:47:28.116+0000 I  CONTROL  [initandlisten]     distmod: ubunt
mongodb_1        | 2019-11-08T16:47:28.116+0000 I  CONTROL  [initandlisten]     distarch: x86_
mongodb_1        | 2019-11-08T16:47:28.116+0000 I  CONTROL  [initandlisten]     target_arch: x
mongodb_1        | 2019-11-08T16:47:28.116+0000 I  CONTROL  [initandlisten] options: { net: {
ls: { mode: "disabled" } }, processManagement: { fork: true, pidFilePath: "/tmp/docker-entrypo
{ destination: "file", logAppend: true, path: "/proc/1/fd/1" } }
```

You can check from the previous code executed we got the output as a new docker network created in our system.

```
[~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
[~]$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
c39dae23a354        bridge              bridge              local
484f006515ef        docker_default      bridge              local
fa524e351841        host                host                local
70eb8e9f1c1e        mongo-network       bridge              local
dcb502d78173        myapp_default       bridge              local
96afbbc49358        none                null                local
[~]$
```

- **These are the two containers that are created by the previous code**

```
[~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                     NAMES
31ce821760f7        mongo-express       "tini -- /docker-e..."   2 days ago          Up 3 minutes        0.0.0.0:8080->8081/tcp    myapp_mongo-express_1
887a135686a6        mongo               "docker-entrypoint..."   2 days ago          Up 3 minutes        0.0.0.0:27017->27017/tcp  myapp_mongodb_1
[~]$
```

- **Now if we open what are the containers that are created we see these two are created by the yaml file**

```
[my-app]$ docker-compose -f mongo.yaml down
Stopping myapp_mongodb_1 ... done
Stopping myapp_mongo-express_1 ... done
Removing myapp_mongo-express_1 ... done
Removing myapp_mongodb_1 ... done
Removing network myapp_default
[my-app]$
```

- **And if we are done with the work and if we want to down the containers and network we need to use this command. Or if we want to drop the container we can use the drop command of container.**

r

- **Data volume provides persistence to the data stored in the database. In general , when we restart the data , the data that is stored previously is lost. But using this volume concept the data will be persistence.**

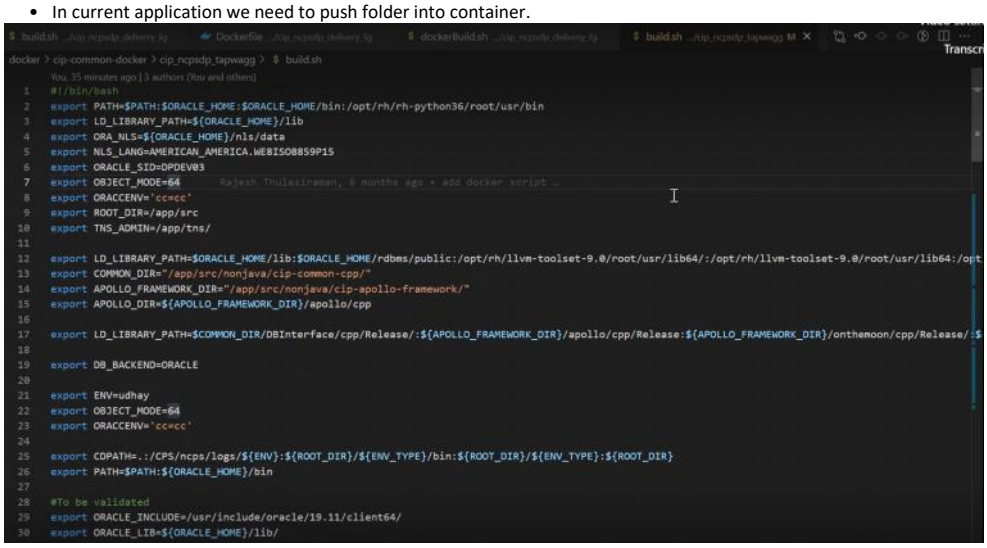
Mentor's meet

Wednesday, March 16, 2022 10:37 PM

- For developer's they no need to bother about operating system of docker because there will be no connection between system operating system and docker's inner OS. Because for the developer the docker is just an image.
- When we use the virtual machine first we need to setup the environment to deploy the application so we need to think about the Operating system. But when coming to docker it is OS will be handled inside the docker container because there is no need to have same OS for docker container and system OS.
- YAML components
  - Image : defines the image we want to import
  - Image policy: defines we must always import the latest version.
  - Command is primarily executed when we run the container.
  - Restart Policy is where we need to mention what we need to do at restart or we need to run it continuously or if there is any pattern.
  - Node selector: defines the size of the node.
- We use docker Kubernetes with azure cloud.
- Here we compile the source code in the container and get the binary code out of it. And use another container to add the dependencies and files of the application. And send for deployment.
- Volumes are used to mount already created files or folders into container or local workspace.
- '..' represent path of the just above directory.
- Cd .. : mean change directory to the parent directory.
- Cd - : is used to go to last visited path.
- For example: when image want to run it uses all the user defined volumes and environmental variables and run as a container.
- Base image: it is an image from which we create a container.
- Less size images are preferable. If it is of large size then it occupies more and more storing and execution space.

Real Time application

- A python application which takes input as raw data and convert into parquet file format. And we have some CPP applications that use parquet files for processing. Now we need to containerize the python code and need to copy the raw data (ra-da copier). Ra-da copiers have dependencies on other libraries such as apollo framework. The parquet file that we discussed is also a framework.
- Apollo framework is responsible for converting oracle data into parquet files. We use apollo API for this process. Apollo libraries generally created in CPP. So we could not access directly in python. Through on the moon API, we get the following functionality.
- Python <-> On the Moon API <-> Apollo framework
- Apollo library also depends on another library called common CPP.
- In existing application we are having two containers.
  - Compiling and building libraries
  - Actually running application.
- Apollo libraries are pre compiled. So we need to import them through repositories or artifactory. And download common CPP libraries and push them to containers. And finally launch them through script.
- We need apollo library and common CPP libraries to run the python application.
- In current application we need to push folder into container.



- This is the sample build.sh code.

```
#To clone the ndx-common-cpp code
if [ -d $ROOT_DIR/nonjava/cip-common-cpp/.git ]; then
    echo "Code Checkout of common cpp"
    cd $ROOT_DIR/nonjava/cip-common-cpp
    git pull
    git checkout $CIP_COMMON_BRANCH
    git pull origin $CIP_COMMON_BRANCH
    git branch
```

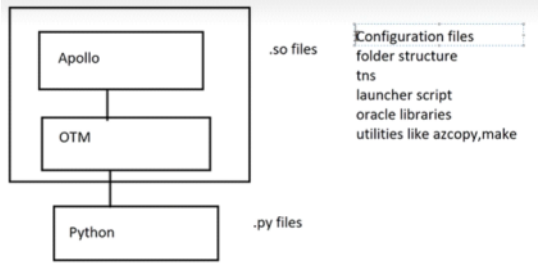
- The above operation is downloading the file from git repository to the local.
- The above image is we are checking that, if we are having the following directory already available on our local system. -d is denoting directory. If we found the folder is already created then redirect the path to common CPP folder and then the git pull command downloads documents at that path from git server to the local. Git checkout is where we are checking our release version. And git pull origin is used to update at that specific branch.
- Now after we get all the files and all the editing is done we need to push into the container , the following code is used for that.

```
cd /app/src/nonjava/cip-cps-data-processing/nonjava/production/trendalignmentprocessing/cpp/src
source scl_source enable devtoolset-9
source scl_source enable llvm-toolset-9.0
make -j 6 CPSFG=1 -f ../build/x86/Linux/Makefile
result=$?

if [[ $result -eq 0 ]]; then
    echo "Now copying outputs to bin & lib"
    cp /app/src/nonjava/cip-cps-data-processing/nonjava/production/trendalignmentprocessing/cpp/Release_fg/PWTA_Aggregation /app/bin

    for library in $(ldd /app/bin/PWTA_Aggregation | grep -v "oracle" | cut -d '>' -f 2 | awk '{print $1}')
    do
        [ -f "${library}" ] && cp --verbose "${library}" /app/common_libs
    done
fi

exit ${result}
```

- ".So" files are used to connect python with CPP files.

Aggregation

- We have an application called aggregation . It is pure CPP code. All the work and processing done using CPP code. And applications such as collecting data from the customer will be also written on CPP code.
- Docker is used for deployment and for faster production the projects. We deploy CPP applications into containers.
- Work is mainly based on CPP application and docker is just a minor part of that.



- Apollo and OTM reads from parquet files which are generated by python application instead of oracle database.

Source

release/22.3.2 ··· CIP CPS Data Processing / python / **rawdata** / Browse Filter

Source	Description	Size	Last Modified
..			
__init__.py	HCO - force python scripts delivery	93 B	26 Nov 2021
__main__.py	CPSS-7572 - Add memory consumed and elapsed time in logs	2.93 KB	29 Mar 2022
process.py	Max Workers parameter bugFix	11.83 KB	15 Feb 2022
rwd_rawdaily.py	CPSS-7760 - rawdata copier BugFix seniority adjustment	22.84 KB	21 Feb 2022
rwd_tbsk.py	CPSS-7760 - rawdata copier BugFix seniority adjustment	17.83 KB	21 Feb 2022
utils.py	HCO - force python scripts delivery	4.24 KB	26 Nov 2021

- This is the path we need to containerize, we need to push all the files into a container.

```
PS > data > vignesh > CPS > docker > $ build.sh
3   ART_URL="https://artifactory.adlm.nielseniq.com:443/artifactory"
4   APOLLO_VERSION=22.2.2
5   curl -sSf -H "${ART_API}" -O "${ART_URL}/cip-generic-prod-apollo-local/${APOLLO_VERSION}/apollo.tar"
6
7   GIT_USER=momo9005
8   GIT_PASSWORD=MzAzMDAzMTc0OTU2OiTLIrV9dBDH9rZqhLMtnlQA73L8
9   CODE_BRANCH=dev/22.3.0
10
11  CODE_URL=${CODE_URL:=https://${GIT_USER}:${GIT_PASSWORD}@adlm.nielseniq.com/bitbucket/scm/cd/cip-cps-data-processing.git}
12
```

```
__main__.py M   utils.py M   rwd_tbsk.py M   rwd_rawdaily.py M   $ build.sh /CPS/.../docker   rawdata_1035_98.log   $ dockerBuild.sh M
CPS > data > vignesh > CPS > docker > $ build.sh
 6
 7  GIT_USER=momo9005
 8  GIT_PASSWORD=MzAzMDAzMTc0OTU2OiTLIrV9dBDH9rZqhLMtnlQA73L8
 9  CODE_BRANCH=dev/22.3.0
10
11  CODE_URL=${CODE_URL:=https://${GIT_USER}:${GIT_PASSWORD}@adlm.nielseniq.com/bitbucket/scm/cd/cip-cps-data-processing.git}
12
13  mkdir -p ./src
14  WHEREAMI=$(pwd)
15  if [ -d ./src/cip-cps-data-processing.git ]; then
16      echo "Code checkout"
17      cd ./src/cip-cps-data-processing
18      git pull
19      git checkout $CODE_BRANCH
20      git pull origin $CODE_BRANCH
21      git branch
22  else
23      echo "code clone"
24      mkdir ./src/cip-cps-data-processing
25      git init ./src/cip-cps-data-processing
26      cd ./src/cip-cps-data-processing
27      git remote add -f origin $CODE_URL
28      git config core.sparseCheckout true
29      echo "python" >> .git/info/sparse-checkout
30      git checkout $CODE_BRANCH
31
32  fi
33  cd $WHEREAMI
34  docker build -t python_test .
```

- The code URL helps us to download the files from the bitbucket to the local.
- GIT_Password : NTYyNjY5MzExMDg3OikulSF7s12dxVHsrguDiWFnde4W
- Git_Username: saka2003
- Mkdir makes a new directory
- Whereami is storing the current working location. That is pwd.
- Git pull download the changes from the git repository to local repository.
- Git checkout moves to particular branch and go into it.
- Git pull origin code branch is used to update the directory with latest changes.
- Git branch is used to print the current branch, generally we do it for debugging purpose.
- Git init is used to initialize the directory. And it tells that we need to use the following directory to store the files from the git.
- Git remote is used to modify the directory by authenticating.
- If we want to download only the python director use line 28 and 29.
- And after downloading python locally we are creating a image named python_test with the new changes that we made.

```
CPS > data > vignesh > CPS > docker > $ build.sh
1  #!/bin/bash
2  ART_API="X-JFrog-Art-Api:AKCp5dKiDXqVDeRRBVaof6dBoA72njDrLq4ai8mZV9vtp3pgc5r8LuPCTDDAU7o88yDM67ur9"
3  ART_URL="https://artifactory.adlm.nielseniq.com:443/artifactory"
4  APOLLO_VERSION=22.2.2
5  curl -sSf -H "${ART_API}" -O "${ART_URL}/cip-generic-prod-apollo-local/${APOLLO_VERSION}/apollo.tar"
```

- The following lines are to download the apollo and OTM API.
- To execute the .sh file that we have before

```
[ncpsapp@dayrhecdpq01 cip-cps-data-processing]$ cd /CPS/data/vignesh/CPS/docker/
[ncpsapp@dayrhecdpq01 docker]$ ll
total 380636
-rw-r--r--. 1 ncpsapp dba      2525 Mar 31 10:06 apollo.conf
-rw-r--r--. 1 ncpsapp dba 389744640 Mar 31 09:31 apollo.tar
-rwxr-xr-x. 1 ncpsapp dba      1096 Mar 31 07:46 build.sh
-rw-r--r--. 1 ncpsapp dba      1851 Apr  1 02:51 Dockerfile
-rw-r--r--. 1 ncpsapp dba      3343 Apr  1 05:55 Dockerfile_1
-rw-r--r--. 1 ncpsapp dba       619 Apr  1 05:50 Dockerfile_2
-rw-r--r--. 1 ncpsapp dba      2055 Apr  1 12:43 go.sh
drwxr-xr-x. 3 ncpsapp dba        37 Apr  1 01:18 src
[ncpsapp@dayrhecdpq01 docker]$ rm apollo.tar
[ncpsapp@dayrhecdpq01 docker]$ ./build.sh
```

While building an image, a Docker File is created automatically . But when we want to change the name of that when we create it, do the following.

```
docker build -t python_test . -f Dcokerfile_1
```

- "ll" represents number of files under the sub folder.

```
[ncpsapp@dayrhecdpq01 docker]$ cd src
[ncpsapp@dayrhecdpq01 src]$ ll
total 0
drwxr-xr-x. 4 ncpsapp dba 32 Apr  1 01:18 cip-cps-data-processing
[ncpsapp@dayrhecdpq01 src]$ cd cip-cps-data-processing/l
[ncpsapp@dayrhecdpq01 cip-cps-data-processing]$ l       l
total 4
drwxr-xr-x. 15 ncpsapp dba 4096 Apr  1 01:18 python
[ncpsapp@dayrhecdpq01 cip-cps-data-processing]$ ll python/
total 56
drwxr-xr-x. 2 ncpsapp dba   62 Apr  1 01:18 calibration
drwxr-xr-x. 2 ncpsapp dba 4096 Apr  1 01:18 common
drwxr-xr-x. 2 ncpsapp dba   57 Apr  1 01:18 datamodel
drwxr-xr-x. 3 ncpsapp dba   18 Apr  1 01:18 docs
drwxr-xr-x. 2 ncpsapp dba   34 Apr  1 01:18 jenkins
drwxr-xr-x. 2 ncpsapp dba 4096 Apr  1 01:18 legacy
-rw-r--r--. 1 ncpsapp dba 1072 Apr  1 01:18 LICENSE.md
-rw-r--r--. 1 ncpsapp dba  897 Apr  1 01:18 Makefile
drwxr-xr-x. 2 ncpsapp dba 4096 Apr  1 01:18 pfw_extractor
drwxr-xr-x. 2 ncpsapp dba 4096 Apr  1 01:18 pfw_tdf_extractor
drwxr-xr-x. 2 ncpsapp dba 4096 Apr  1 01:18 rawdata
drwxr-xr-x. 2 ncpsapp dba   78 Apr  1 01:18 rawdata_tcu
-rw-r--r--. 1 ncpsapp dba  144 Apr  1 01:18 README.md
-rw-r--r--. 1 ncpsapp dba  584 Apr  1 12:12 requirements.txt
-rw-r--r--. 1 ncpsapp dba  205 Apr  1 01:18 setup.cfg
-rw-r--r--. 1 ncpsapp dba 2281 Apr  1 01:18 setup.py
-rw-r--r--. 1 ncpsapp dba 1329 Apr  1 01:18 sonar-project.properties
drwxr-xr-x. 2 ncpsapp dba   78 Apr  1 01:18 ta_restftbsk
drwxr-xr-x. 2 ncpsapp dba 4096 Apr  1 01:18 tests
-rw-r--r--. 1 ncpsapp dba   47 Apr  1 01:18 venv.cfg
drwxr-xr-x. 2 ncpsapp dba   24 Apr  1 01:18 version
[ncpsapp@dayrhecdpq01 cip-cps-data-processing]$
```

".so" mean shared object. We can use those libraries to access the parquet files. These files implement apollo and on the moon.

- Add command typically unzip the files and saves at mentioned location.

```
CPS > data > vignesh > CPS > docker > Dockerfile_1 > ADD
32      yum -y localinstall llvm-toolset-9.0* && \
33      ln -s /usr/lib64/libtinfo.so.5 /usr/lib64/libtinfo.so && \
34      ln -s /usr/lib/oracle/19.11/client64/lib/libocci.so.19.1 /usr/lib/oracle/19.11/client64/lib/libocci.so && \
35      rm -rf /app/llvm-toolset-9.0* && \
36      yum clean all && \
37      rm -rf /var/cache/yum && \
38      mkdir -p /app/lib/ && \
39      cp /opt/rh/llvm-toolset-9.0/root/usr/lib64/libLLVM-9.so /app/lib/ && \
40      rm -rf /opt/rh/llvm-toolset-9.0/*
41
42
43  COPY ./src/cip-cps-data-processing/python/requirements.txt requirements.txt
44  RUN pip3 install -r requirements.txt
45
46  USER root
47  RUN mkdir -p /app/apollo /app/python /app/bin /app/logs /app/tns /app/apollo/session /app/apollo/cache /app/apollo/metrics && chmod +777 -R /app/*
48
49  USER cip
50  ADD --chown=cip:cip apollo.tar /app/
51  COPY --chown=cip:cip ./src/cip-cps-data-processing/python /app/python
52  COPY --chown=cip:cip ./go.sh /app/bin
53  COPY --chown=cip:cip --from=azcopy /usr/bin/azcopy /app/bin/azcopy
54  COPY --chown=cip:cip ./apollo.conf /app/apollo/
55  RUN chmod +x /app/bin/go.sh
56  ENV TNS_ADMIN="/app/tns"
57
58  # USER root
59  #ENTRYPOINT [ "/app/bin/go.sh" ]
```

- And here "--chown" is changing the ownership of the file to CIP. And copying the local files to the containers.
- Go.sh and app bin line is the launcher script.
- Azcopy line is utility , we are copying utilities from local to the container.
- And we are copying apollo configuration file to the container.
- "Chmod +x" gives the executing permission to the go.sh script
- TNS file consists data about database server.
- For example,

```
DSPCNAZ1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 10.255.149.4)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = DSPUNAXA)
    )
  )
```

- We are giving IP address, port and server and service name, for every environment we must have such details in TNS. Which is required to connect to the oracle to the server.
- If we want to connect Linux server to oracle server we need this TNS file(tnsnames.ora)
- And for folder structure we create many folders using this command.

```
USER root
RUN mkdir -p /app/apollo /app/python /app/bin /app/logs /app/tns /app/apollo/session /app/apollo/cache /app/apollo/metrics && chmod +777 -R /app/*
```

- "chmod +777" is used to give all permissions
- All the above code is discussed under DockerFile
- In go script we will give the details of path and where to save the log script.
- `#./go.sh rawdata 1035 98 US`
- First number is shop id
- Second number is cycle number
- Third is from which country we are going to get.
- Output of the raw data is going to the ADLS (Cloud Storage)
- About we use azcopy because we copy the files of the local storage or container to the cloud with the help of ADLS
- The following is used to install azcopy in docker file

```
FROM alpine as azcopy
RUN apk add --no-cache wget \
 && wget https://aka.ms/downloadazcopy-v10-linux -O /tmp/azcopy.tgz \
 && export BIN_LOCATION=$(tar -tzf /tmp/azcopy.tgz | grep "/azcopy") \
 && tar -xzf /tmp/azcopy.tgz $BIN_LOCATION --strip-components=1 -C /usr/bin
```

The following code is mandatory to make a container

```
RUN yum-config-manager --enable software_collections && \
    yum-config-manager --enable ol7_latest ol7_optional_latest && \
    yum -y install oracle-softwarecollection-release-el7 scl-utils && \
    yum -y install devtoolset-9 git makedepend vim wget openssl-devel uuid-devel libuuid libuuid-devel ksh gcc-c++ && \
    yum -y install bzip2-devel libffi-devel && \
    cd /opt && wget https://www.python.org/ftp/python/3.8.5/Python-3.8.5.tgz && tar xzf Python-3.8.5.tgz && cd Python-3.8.5 && ./configure --enable-optimizations && ma
    yum -y install epel-release python3-pip && \
    pip3 install Cython numpy cx_Oracle && \
    mkdir /app && \
    cd /app && \
    curl -sSf -H "X-JFrog-Art-Api:AKCp5dKiDXqVDeRRBVaof6dBoA72njDrLq4aiBm2V9vtp3pgc5r8LuPCTDDAU7o88yDM67ur9" -O "https://artifactory.adlm.nielseniq.com:443/artifactory,
    curl -sSf -H "X-JFrog-Art-Api:AKCp5dKiDXqVDeRRBVaof6dBoA72njDrLq4aiBm2V9vtp3pgc5r8LuPCTDDAU7o88yDM67ur9" -O "https://artifactory.adlm.nielseniq.com:443/artifactory,
    curl -sSf -H "X-JFrog-Art-Api:AKCp5dKiDXqVDeRRBVaof6dBoA72njDrLq4aiBm2V9vtp3pgc5r8LuPCTDDAU7o88yDM67ur9" -O "https://artifactory.adlm.nielseniq.com:443/artifactory,
    curl -sSf -H "X-JFrog-Art-Api:AKCp5dKiDXqVDeRRBVaof6dBoA72njDrLq4aiBm2V9vtp3pgc5r8LuPCTDDAU7o88yDM67ur9" -O "https://artifactory.adlm.nielseniq.com:443/artifactory,
    yum -y localinstall llvm-toolset-9.0* && \
    ln -s /usr/lib64/libtinfo.so.5 /usr/lib64/libtinfo.so && \
    ln -s /usr/lib/oracle/19.11/client64/lib/libocci.so.19.1 /usr/lib/oracle/19.11/client64/lib/libocci.so && \
    mkdir /app/src && mkdir /app/bin && mkdir /app/common_libs && chmod 777 -R /app/* && \
    rm -rf /app/llvm-toolset-9.0* && \
    yum clean all && \
    rm -rf /var/cache/yum
```

.so files helps us to do operations of apollo and OTM

```
ADD --chown=cip:cip apollo.tar /app/
COPY --chown=cip:cip ./src/cip-cps-data-processing/python /app/python
```

The following two lines copy apollo and python folders into the container

- go.sh (Launcher script)

```
python3 -m $@
```

-m is module name

```
mkdir -p ./src
WHEREAMI=$(pwd)
if [ -d ./src/cip-cps-data-processing.git ]; then
    echo "Code checkout"
        cd ./src/cip-cps-data-processing
    git pull
        git checkout $CODE_BRANCH
    git pull origin $CODE_BRANCH
    git branch
else
        echo "code clone"
        mkdir ./src/cip-cps-data-processing
        git init ./src/cip-cps-data-processing
    cd ./src/cip-cps-data-processing
    git remote add -f origin $CODE_URL
    git config core.sparseCheckout true
    echo "python" >> .git/info/sparse-checkout
    git checkout $CODE_BRANCH

fi
```

- Here git init mean we are telling git to monitor the changes that are happened inside the following directory.
- Git pull is used to update the local repository changes.
- -p in mkdir is used to check if the following directory is present or not , if not create the directory using mkdir path
- -p in expose helps us to expose the path
- -t helps us to create and tag the images in current build context in build process.
- Git remote add is used to add the remote URL to the git , to declare from where we must pull the repository that we have discussed previously with git pull.
- Git sparse checkout command is used to check the URL that we declared is exists or not, if exists the initiation part of remote URL will be completed. Or else we will get an error.
- And echo "python">> sparsecheckout line, we are telling git , consider only this folder to be up to date, rather than considering the whole repository. After declaring it we are moving into the particular code branch.
- Build.sh is used for building the script and go.sh is a launcher script.

```
CPS > data > vignesh > CPS > docker > Dockerfile > ...
1  FROM alpine as azcopy
2  RUN apk add --no-cache wget \
3   && wget https://aka.ms/downloadazcopy-v10-linux -O /tmp/azcopy.tgz \
4   && export BIN_LOCATION=$(tar -tzf /tmp/azcopy.tgz | grep "/azcopy") \
5   && tar -xzf /tmp/azcopy.tgz $BIN_LOCATION --strip-components=1 -C /usr/bin
```

Line 1: We are using base image as alpine , alpine is light weight base image and naming that build as azcopy, azcopy is used to upload the files into ADLS (Azure data lake storage)
Line 2: RUN command is used to run the command in the container, we are adding wget into the container with out considering cache.
Line 3: And using that wget we can download files from the internet, and save that file locally under temp/azcopy.tgz , -O denotes saving the file
Line 4: Now as we don't want to unzip whole tar file , we need to find a folder under that tar called azcopy and store that folder location , which is used to extract in line 5
Line 5: we are unzipping

Options of zipping:

Options:
-c : Creates Archive
-x : Extract the archive
-f : creates archive with given filename
-t : displays or lists files in archived file
-u : archives and adds to an existing archive file
-v : Displays Verbose Information
-A : Concatenates the archive files
-z : zip, tells tar command that creates tar file using gzip
-j : filter archive tar file using tbzip
-W : Verify a archive file
-r : update or add file or directory in already existed .tar file

Line 4: tzf is used to displays or lists files in archived file and creates a tar file using gzip and creates an archive with given filename and name it as azcopy
Line 5: And in line 5 we try to extract the archive and declare the zip and creates an archive with file

name, the operations goes from right to left . xzf => 1)f 2)z 3)x
In line 5, we are giving a bin location of azcopy to the tar and create the archive with the given filename. declare the file as tar using 'z' . And x is used to extract the archive and copy it (-c) to the user/bin

- Whenever we start Docker Component 'From' it is a new build stage. New layer is created and the old layers will be cleared as we are into the new from command.

```
49  USER cip
50  ADD --chown=cip:cip apollo.tar /app/
51  COPY --chown=cip:cip ./src/cip-cps-data-processing/python /app/python
52  COPY --chown=cip:cip ./go.sh /app/bin
53  COPY --chown=cip:cip --from=azcopy /usr/bin/azcopy /app/bin/azcopy
54  COPY --chown=cip:cip ./apollo.conf /app/apollo/
55  RUN chmod +x /app/bin/go.sh
56  ENV TNS_ADMIN="/app/tns"
57
58  ENTRYPOINT [ "/app/bin/go.sh" ]
```

Here in line 53, a from command is used to import the previously saved and executed build statements. And copy the azcopy file that is previously extracted and saved under user/bin/azcopy to the app/bin/azcopy.

We are having the sales data or raw data in a database (Data validation) , and place that into another database (Data processing).
Rawda application main work is to transfer the data from Data validation(do clean up activity) database to the Data processing database.

When we run run_python.ksh launch python module for us. Which is present under CPS/data/vignesh/CPS/

Run_python.ksh script will be called from the UI, the UI gives the input for the script

The export variables that we export from run_python.ksh will be available for the application that we are going to call. If we want to use some enviroment variables in our application then we declare and export them in ksh script file.
run_python.ksh

```
11  export HIVE_DP_BATCH_CONNECT=cip-us-cert-ncp-hms-lb.azure-cip-np.nielsencsp.net:9081
12  export HIVE_DP_DATABASE_NAME=dp_dpcnaz1
13  export HIVE_DV_BATCH_CONNECT=cip-us-cert-ncp-hms-lb.azure-cip-np.nielsencsp.net:9081
14  export HIVE_DV_DATABASE_NAME=dv_dvcnaz1
15  export TCU_BATCH_CONNECT=cpssys/CPSSYS@DPCNAZ1
16  export TNS_ADMIN=/CPS/data/vignesh/tns
17  export AZACCOUNT=retcipcuscertg2sa
18  export AZCONTAINER=cip-shared-data-ncpsdp
```

The variables that we exported in run_python.ksh will be used under CPS/data/vigneshCPS/src/cip-cops.data-processing/python/rawdata/__main__.py

Run_python.ksh

```
export APOLLO_DIR="/CPS/data/vignesh/CPS/apollo/"
export PYTHONPATH=${APOLLO_DIR}:${PYTHONPATH}
#. ${APOLLO_DIR}/apollo_init.sh
export LD_LIBRARY_PATH=${APOLLO_DIR}:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=${APOLLO_DIR}/thrift/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=${APOLLO_DIR}/parquet/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=${APOLLO_DIR}/gtest/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=${APOLLO_DIR}/boost/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=${APOLLO_DIR}/redis/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=${APOLLO_DIR}/re2/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=${APOLLO_DIR}/azure-sdk/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=${APOLLO_DIR}/curl/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=${APOLLO_DIR}/bifrost/Release:${LD_LIBRARY_PATH}
```

We are having these LD_LIBRARY_PATH to store all the folders that are required for our application and export that variable or storage location to the OS, to perform the application. Whenever application needs the libraries , the OS goes to the LD_LIBRARY_PATH and fetch the required library. So , we are storing all the required libraries and paths in LD_LIBRARY_PATH.

Here we are calling application from ksh file, in which we can use current exported variables.

```
52  source /CPS/ncps/dppocna/ncpsdp/python/venv/bin/activate
53
54  if [[ ${1} != *.py* ]]; then
55      python -m $@
56  else
57      python ${DIR_BATCH_BIN}/$@
58  fi
59  pStatus=$?
60  deactivate
61  exit $pStatus
```

To manage all the containers of Docker , we use Kubernetes. Kubernetes is used for orchestration. In Kubernetes , we can do load balancing , according to the number of users are working or using the application. If there are more users then add more containers. All these operations are done dynamically, we need to code it once, and then Kubernetes will take care of that operation. When we are running the container , it will run in the node. And the group of nodes is called cluster. So to manage those all nodes we use cluster as a whole and manage the nodes using the cluster.

Instead of managing individual containers , we manage a group of containers in Kubernetes. We have a concept call pod in Kubernetes. Inside a pod we can run a group of containers.

In Kubernetes , we use Yaml file , this is the basic thing. Yaml file is typically a configuration file. And it contains the options like image name , image pull policy like how frequent we have to run the image. We can declare how much memory we need, and volume also can be specified in the Yaml file.

Raw data application reads the data in oracle and writes the data in parquet files and do further processing.

All the images are stored in azure container repository. Docker and Kubernetes search for images in azure container repository instead of docker hub.

Files are stored locally on the server in general and when we want to use volume and persist the data we can store the files in a cloud called netapp . And to access that we must have a volume claim so we defined the claimName attribute under persistentVolumeClaim. So we can mount the storage on the netapp.

tail -f logfilename.log is used to see the log file changes in real time
To check the logs of the python application , we need to go to the following directory to check that. These logs are made by python application itself, in our case these logs are made by raw data application. The logs that we created will be in our directory that we created for storing logs.

```
udpcnaz1@mue2oelcdpc4001(/CPS/data/vignesh/jeevan)$./run_python.ksh  rawdata
5 98 US
udpcnaz1@mue2oelcdpc4001(/CPS/data/vignesh/jeevan)$clear
udpcnaz1@mue2oelcdpc4001(/CPS/data/vignesh/jeevan)$cd /aksdata
udpcnaz1@mue2oelcdpc4001(/aksdata)$ls
dpcnaz1  DPCNAZ1  DPDNAZ1
udpcnaz1@mue2oelcdpc4001(/aksdata)$cd DPCNAZ1/
/aksdata/DPCNAZ1
udpcnaz1@mue2oelcdpc4001(/aksdata/DPCNAZ1)$ls
config  cpsfgdelivery  cpspf       cpssp   tns
cpsagg  cpsintegrity   cpsrawdata  cpstcu
udpcnaz1@mue2oelcdpc4001(/aksdata/DPCNAZ1)$cd cpsrawdata/
/aksdata/DPCNAZ1/cpsrawdata
udpcnaz1@mue2oelcdpc4001(/aksdata/DPCNAZ1/cpsrawdata)$ls
data  logs  tmp
udpcnaz1@mue2oelcdpc4001(/aksdata/DPCNAZ1/cpsrawdata)$cd logs
/aksdata/DPCNAZ1/cpsrawdata/logs
udpcnaz1@mue2oelcdpc4001(/aksdata/DPCNAZ1/cpsrawdata/logs)$ls -lrt
total 452
-rw-rw-r--. 1 cipapp hive  4428 Apr  7 02:25 cps-dpcnaz1-rawdata-service1035-
8-ccUS-pod0-20220407021721.log
-rw-rw-r--. 1 cipapp hive 19828 Apr  7 02:49 cps-dpcnaz1-rawdata-service1035-
8-ccUS-pod0-20220407024357.log
-rw-r--r--. 1 cipapp hive 19828 Apr  7 02:57 cps-dpcnaz1-rawdata-service1035-
8-ccUS-pod0-20220407025315.log
-rw-r--r--. 1 cipapp hive 19828 Apr  7 04:25 cps-dpcnaz1-rawdata-service1035-
8-ccUS-pod0-20220407041845.log
-rw-r--r--. 1 cipapp hive 19828 Apr  7 05:16 cps-dpcnaz1-rawdata-service1035-
8-ccUS-pod0-20220407050900.log
```

- The above image is the process to check the application logs , ls -lrt is used to visit all the files in current directory with the additional information like , permissions that the file have and owner and date of creation etc.
- Now under this we can open our own log file so we can see the application logs.
- In our application , we copy the data from Data validation to the data processing.
- We store the parquet files in the DV hive location.
- The data which is in the local is copied to the azure storage with the help of ADLS (Azure data lake storage)

```
| rwd_tbsk : partiton - P247 or P202007 : Transaction Count=0
| /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-tbsk-1035-p247.parq

| /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-tbsk-1035-p247.parq

| rwd_tbsk : partiton - P248 or P202008 : Transaction Count=0
| /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-tbsk-1035-p248.parq

| /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-tbsk-1035-p248.parq

| rwd_tbsk : partiton - P249 or P202009 : Transaction Count=0
| /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-tbsk-1035-p249.parq
```

- Here in the application logs , we can see some transactions happening in individual partitions. While transferring files from Data validation to data processing we convert the raw data into parquet files , where it is named as cube , cube mean a data file and tbsk mean total basket. In nielseniq we measure the rawdata in two ways , raw daily and total basked.
- Transaction count 0 mean , there is no data to copy from DV to DP in current partition.
- Transaction count represents number of rows are copied from DV to DP.
- As said, the application of rawdata is not only to copy the data , it does some other tasks such as the following picture. It have some virtual formula's and based on that , we change the actual rawdata into a new data.

```
| alter_resrawdacolumns | New columns to be added in RESRAWDACOLUMNS: []
| alter_hive_columns    | New columns to be added to hive: set()
| process               | Keep Banded Packs Flag  : Y
| process               | Rawdaily Sr Adj Flag    : 0
| process               | Seniority Adj Active    : 0
| process               | Virtual Formulas :
| process               |     ['DT4009', '(CASE WHEN RAWDA.DT5662>0 THEN (RAWDA.DT5650*RAWD

| process               |     ['DT4010', '(CASE WHEN RAWDA.DT5663>0 THEN (RAWDA.DT5650*RAWD

| process               |     ['DT4011', '(CASE WHEN RAWDA.DT5664>0 THEN (RAWDA.DT5650*RAWD

| process               |     ['DT4012', '(CASE WHEN RAWDA.DT5665>0 THEN (RAWDA.DT5650*RAWD

| process               |     ['DT4013', 'NVL(RAWDA.DT5131,1)']
| process               |     ['DT4001', 'RAWDA.DT5650']
| process               |     ['DT4002', 'RAWDA.DT5650 * RAWDA.DT5675']
| process               |     ['DT4003', '(CASE WHEN (RAWDA.DT5662>0 OR RAWDA.DT5663>0 OR R

| process               |     ['DT4004', '(CASE WHEN RAWDA.DT5662>0 THEN RAWDA.DT5650 END)'

| process               |     ['DT4005', '(CASE WHEN RAWDA.DT5663>0 THEN RAWDA.DT5650 END)'

| process               |     ['DT4006', '(CASE WHEN RAWDA.DT5664>0 THEN RAWDA.DT5650 END)'

| process               |     ['DT4007', '(CASE WHEN RAWDA.DT5665>0 THEN RAWDA.DT5650 END)'

| process               |     ['DT4008', '(CASE WHEN (RAWDA.DT5662>0 OR RAWDA.DT5663>0 OR R
ND)']
| process               | self.rawdaily_dv_partitions = ['P202007', 'P202008', 'P202009', '
202105', 'P202106', 'P202107']
| process               | self.rawdaily_dp_partitions = ['P247', 'P248', 'P249', 'P250', 'P
```

- Similarly , we copy data from raw daily other than tbsk as the following example, it also have its own individual transaction count.

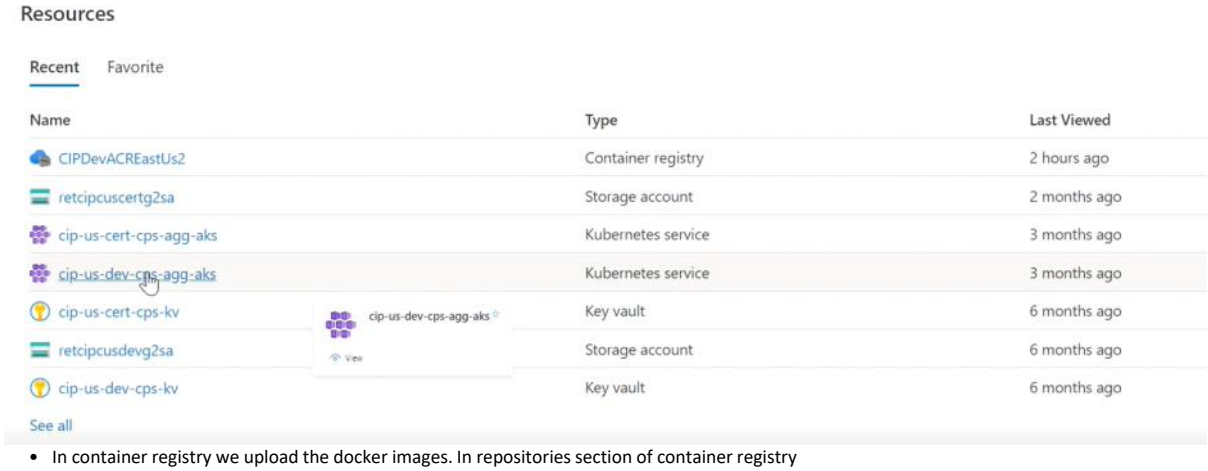
```
process               | self.rawdaily_dv_partitions = ['P202007', 'P202008', 'P202009', '
02105', 'P202106', 'P202107']
process               | self.rawdaily_dp_partitions = ['P247', 'P248', 'P249', 'P250', 'P

dump_rawdaily         | rwd_rawdaily : partition - P202007 or P247 : Transaction Count=0
dump_rawdaily         | /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-rawdaily-1035-p247

dump_rawdaily         | /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-rawdaily-1035-p247

dump_rawdaily         | rwd_rawdaily : partition - P202008 or P248 : Transaction Count=0
dump_rawdaily         | /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-rawdaily-1035-p248

dump_rawdaily         | /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-rawdaily-1035-p248
```

- Here for example P202008 mean we are retrieving the data of 8th week of 2020 and copying it.
- And at last of copying all the data , we transfer the data from local to the ADLS as shown in the picture.

```
parquet Deleting local copy...
2022/05/30 11:11:03 | INFO  | Mem: 0.387 GB | rwd_rawdaily.py      | dump_rawdaily        | rwd_rawdaily : partition - P202107 or P259 : Transaction Count=76
8839
2022/05/30 11:11:03 | INFO  | Mem: 0.387 GB | rwd_rawdaily.py      | dump_rawdaily        | /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-rawdaily-1035-p259.
parquet Uploading to adls...
2022/05/30 11:11:06 | INFO  | Mem: 0.387 GB | rwd_rawdaily.py      | dump_rawdaily        | /app/tmp/dp_dpcnaz1/rawdata/service_1035/cube-rawdaily-1035-p259.
parquet Deleting local copy...
2022/05/30 11:11:06 | INFO  | Mem: 0.134 GB | common_utils.py      | wrapper              | Peak memory usage : 190296 KB (or) 185.84 MB (or) 0.18 GB
2022/05/30 11:11:06 | INFO  | Mem: 0.134 GB | common_utils.py      | wrapper              | Elapsed              : 0:04:13.702454
```

- We will have general details at the last , such as time required to complete the pod and peak memory usage to complete the task.

These are some services that are being provided by the azure

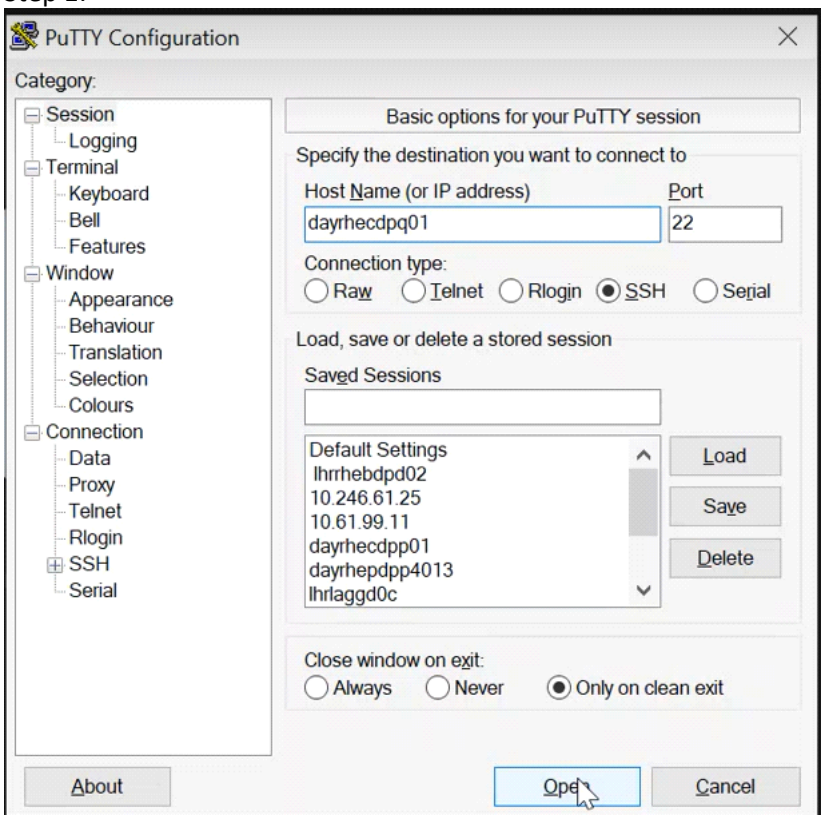


- In container registry we upload the docker images. In repositories section of container registry we have all the docker images that we can use, and even we can upload one.
- Storage account stores all the output of the application, output is the end product of the application process. Parquet files are being stored inside this storage account.
- In netapp storage we write log files
- The details regarding parquet is stored in hive address.
- The local data is copied into azure storage.
- We are modifying the data into parquet files , where you can see in the log it is stored in the form of cube_tbsk_serviceid_partitionnumber.parq , where cube is the file type and tbsk is the total basket which refers to the CPS.
- If we have transaction count greater than 0 then we get to copy the data into parquet file (DV to DP).
- Netapp is used to store the log files and azure is used to store most of the processes.

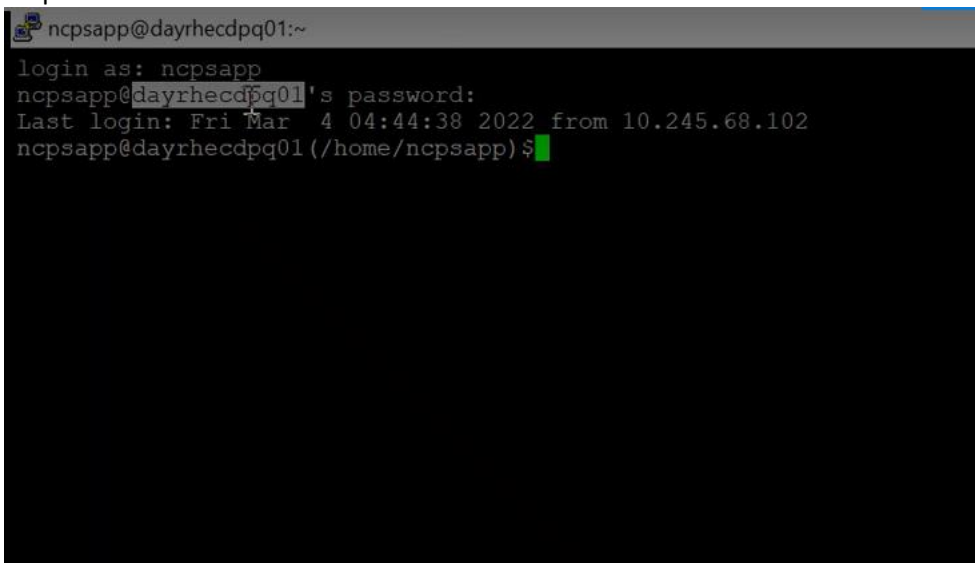
How to connect to the company server

Wednesday, March 16, 2022 10:48 PM

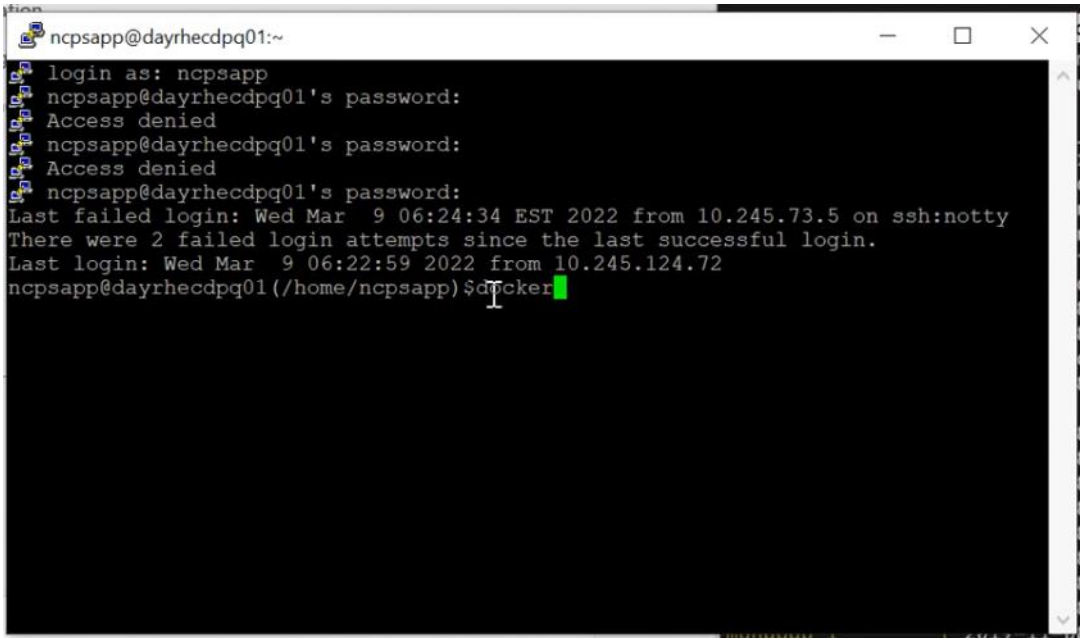
Step 1:



Step 2:



- Password will be dayrhecdpq01
- After able to login , type docker command on the terminal that it is directed as the following



- You have to work in the following directory

```
ncpsapp@dayrhecdpq01(/CPS/data/vignesh)$mkdir jeevan
ncpsapp@dayrhecdpq01(/CPS/data/vignesh)$ls
```

- General commands of Linux don't use this in docker (wq mean write and quit and q mean quit and q! mean overwrite a file)
- Remove file $rm
- To list all the current directory file $ll
- Yaml file is used for Kubernetes.

Kubernetes server details:
mue2oelcdpc4001.azure-cip-np.nielsencsp.net
user name ncpsread
password is mue2oelcdpc4001
su - udpcnaz1
5kyrlBO6

33 27 18.5 9

Docker server details:
dayrhecdpq01
username: ncpsapp
password: dayrhecdpq01

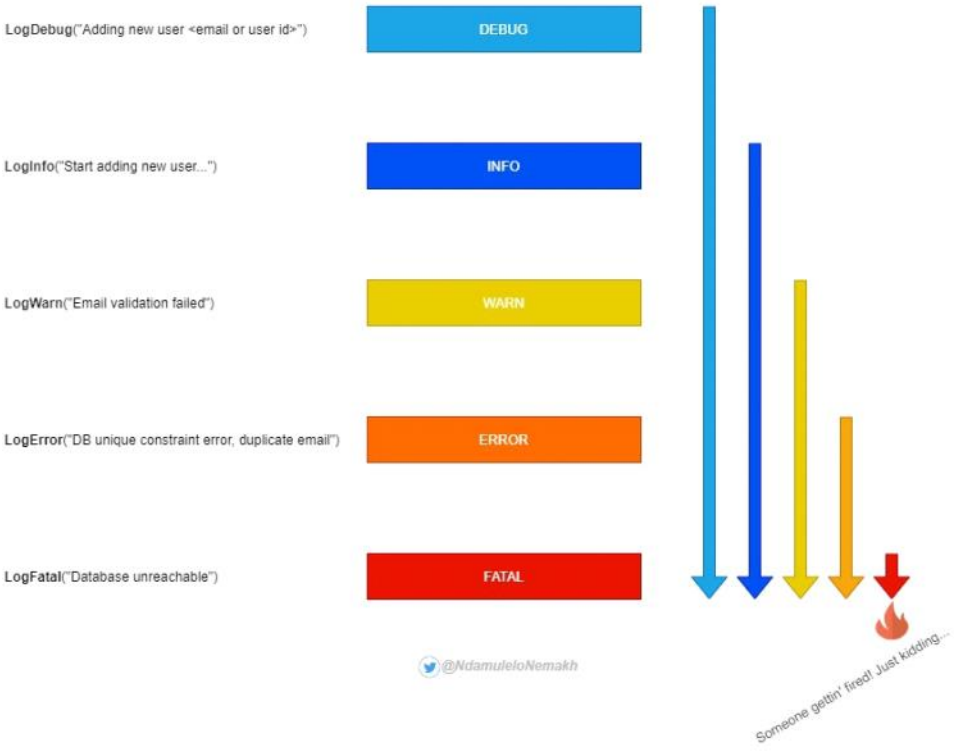
rawdata_<<service>>_<<country_code>>_<<time stamp>>.log

2022/04/01 11:08:39 | INFO | Mem: 0.080 GB | process.py | process | Service ID : 1035

2022/04/01 11:08:39 | INFO | Mem: 0.080 GB | process.py | process | ProdCycle : 98

2022/04/01 11:08:39 | INFO | Mem: 0.080 GB | process.py | process | Country Code : US

Example: New User Sign up API



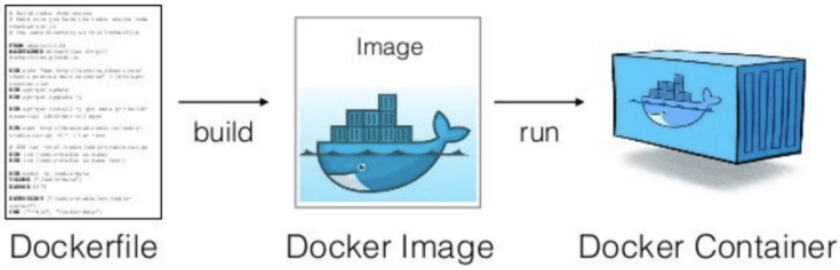
Dockerfile Components

Saturday, April 9, 2022 10:41 AM

[Creating Dockerfile with from, run, cmd, entrypoint, copy, add, volume, expose | in Hindi]

Docker is an open-source software platform that is designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. To create a Docker container we need a docker image and to create a docker image we need a Dockerfile and artifacts with the help of Dockerfile we create the docker image.

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is simply a text file named as Dockerfile(without any extensions) with some commands and rules that Docker uses to create an image. It contains the steps Docker is supposed to follow to package your app.



Dockerfile Commands:

```
FROM

ENV

WORKDIR

ENTRYPOINT

CMD

COPY

ADD

RUN

EXPOSE
```

- All the following Docker file commands run within the container. And RUN commands helps us to run Linux command "inside" the container that we have created.

FROM

```
FROM <image> [AS <name>]
```

FROM is used to define the base image to start the build process. Every Dockerfile must start with the FROM instruction. The idea behind this is that you need a starting point to build your image.

```
FROM ubuntu
```

ENV

```
ENV <key> <value>
```

This command used to set the environment variables that is required to run the project.

ENV sets the environment variables, which can be used in the Dockerfile and any scripts that it calls. These are persistent with the container too and they can be referenced at any time.

```
ENV HTTP_PORT="9000"
```

WORKDIR

```
WORKDIR /path/to/workdir
```

WORKDIR tells Docker that the rest of the commands will be run in the context of the `/app` folder inside the image.

```
WORKDIR /app
```

It will create the **app** directory in the container.

- Run commands run the Linux commands inside the container.

RUN

RUN has 2 forms:

- `RUN <command>` (*shell* form, the command is run in a shell, which by default is `/bin/sh -c` on Linux or `cmd /S /C` on Windows)
- `RUN ["executable", "param1", "param2"]` (*exec* form)

The `RUN` instruction will execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step in the `Dockerfile`.

The RUN command runs within the container at build time.

```
RUN /bin/bash -c 'source $HOME/.bashrc; echo $HOME'
```

- Entry point is the first point of execution after container starts execution , if we run a docker run command then we can pass parameters to the entry point to execute it when the container starts running. By including the parameter.

ENTRYPOINT

ENTRYPOINT has two forms:

- `ENTRYPOINT ["executable", "param1", "param2"]` (*exec* form, preferred)
- `ENTRYPOINT command param1 param2` (*shell* form)

An `ENTRYPOINT` allows you to configure a container that will run as an executable.

ENTRYPOINT sets the command and parameters that will be executed first when a container is run. Any command-line arguments passed to `docker run <image>` will be appended to the ENTRYPOINT command, and will override all elements specified using `CMD`. For example, `docker run <image> bash` we will add the command argument **bash** to the end of the ENTRYPOINT command.

You can override ENTRYPOINT instructions using the `docker run --entrypoint`

```
ENTRYPOINT [ "sh", "-c", "echo $HOME" ]
```

If the ENTRYPOINT isn't specified, Docker will use /bin/sh -c as the default executor.

CMD

The `CMD` instruction has three forms:

- `CMD ["executable","param1","param2"]` (*exec* form, this is the preferred form)
- `CMD ["param1","param2"]` (as *default parameters to ENTRYPOINT*)
- `CMD command param1 param2` (*shell* form)

The main purpose of a CMD is to provide defaults when executing a container. These will be executed after the entry point.

In Dockerfiles, you can define `CMD` defaults that include an executable.

Example:

```
CMD ["executable","param1","param2"]
```

If they omit the executable, you must specify an `ENTRYPOINT` instruction as well.

`CMD ["param1","param2"]` (as default parameters to ENTRYPOINT)

NOTE: There can only be one **CMD** instruction in a `Dockerfile`. If you want to list more than one **CMD**, then only the last **CMD** will take effect.

```
CMD ["bin/ping", "localhost"]
```

Understand how CMD and ENTRYPOINT interact

Both `CMD` and `ENTRYPOINT` instructions define what command gets executed when running a container. There are a few rules that describe their co-operation.

1. Dockerfile should specify at least one of **CMD** or **ENTRYPOINT** commands.
2. **ENTRYPOINT** should be defined when using the container as an executable.
3. **CMD** should be used as a way of defining default arguments for an **ENTRYPOINT** command or for executing an ad-hoc command in a container.
4. **CMD** will be overridden when running the container with alternative arguments.

COPY

COPY has two forms:

- `COPY <src>... <dest>`
- `COPY ["<src>",... "<dest>"]` (this form is required for paths containing whitespace)

The **COPY** command is used to copy one or many local files or folders from source and adds them to the filesystem of the containers at the destination path.

It builds up the image in layers, starting with the parent image, defined using `FROM`.The Docker instruction **WORKDIR** defines a working directory for the **COPY** instructions that follow it.

The `<dest>` is an absolute path, or a path relative to `WORKDIR`, into which the source will be copied inside the destination container.

```
COPY test relativeDir/   # adds "test" to `WORKDIR`/relativeDir
COPY test /absoluteDir/  # adds "test" to /absoluteDir/
```

ADD

ADD has two forms:

- `ADD <src>... <dest>`
- `ADD ["<src>",... "<dest>"]`

The **ADD** command is used to add one or many local files or folders from the source and adds them to the filesystem of the containers at the destination path.

It is Similar to COPY command but it has some additional features:

- If the source is a **local tar** archive in a recognized compression format, then it is automatically unpacked as a directory into the Docker image.
- If the source is a URL, then it will download and copy the file into the destination within the Docker image. However, Docker discourages using **ADD** for this purpose.

```
ADD rootfs.tar.xz /
ADD http://example.com/big.tar.xz /usr/src/things/
```

EXPOSE

```
EXPOSE <port> [<port>/<protocol>...]
```

The EXPOSE command informs the Docker that the container listens on the specified network ports at runtime. You can specify whether the port listens on TCP or UDP, and the default is TCP if the protocol is not specified.

But **EXPOSE will not** allow communication via the defined ports to containers outside of the same network or to the host machine. To allow this to happen you need to *publish* the ports.

The **EXPOSE** command does not actually publish the port. To actually publish the port when running the container, use the `-p` flagon `docker run` to publish and map one or more ports, or the `-P` flag to publish all exposed ports and map them to high-order ports.

These are the flags -p and -P, and they differ in terms of whether you want to publish one or all ports:

- *To actually publish the port when running the container, use the* -p *flag on* docker run *to publish and map one or more ports*
- *he* -P *flag to publish all exposed ports*

```
//Publish host port to container port

docker run -p 80:80/tcp -p 80:80/udp app

//To publish all the ports you define in your Dockerfile
with EXPOSE and bind them to the host machine, you can use
the -P flag.

docker run -P app
```

- The build is run by the Docker daemon, not by the CLI. The first thing a build process does is send the entire context (recursively) to the daemon. In most cases, it's best to start with an empty directory as context and keep your Dockerfile in that directory. Add only the files needed for building the Dockerfile.
- Do not use your root directory, /, as the PATH for your build context, as it causes the build to transfer the entire contents of your hard drive to the Docker daemon.

To tag the image into multiple repositories after the build, add multiple `-t` parameters when you run the `build` command:

```
$ docker build -t shykes/myapp:1.0.2 -t shykes/myapp:latest .
```

- Whenever possible, Docker uses a build-cache to accelerate the docker build process significantly. This is indicated by the CACHED message in the console output.
- **Docker build kit**
  - We must enable the build kit on our docker to avail the following advantages.

- Detect and skip executing unused build stages
- Parallelize building independent build stages
- Incrementally transfer only the changed files in your build context between builds
- Detect and skip transferring unused files in your build context
- Use external Dockerfile implementations with many new features
- Avoid side-effects with rest of the API (intermediate images and containers)
- Prioritize your build cache for automatic pruning

To use the BuildKit backend, you need to set an environment variable `DOCKER_BUILDKIT=1` on the CLI before invoking `docker build`.

- A Dockerfile must begin with a FROM instruction. This may be after parser directives, comments, and globally scoped ARGs. The FROM instruction specifies the Parent Image from which you are building. FROM may only be preceded by one or more ARG instructions, which declare arguments that are used in FROM lines in the Dockerfile.

Docker treats lines that *begin* with `#` as a comment, unless the line is a valid parser directive. A `#` marker anywhere else in a line is treated as an argument. This allows statements like:

```
# Comment
RUN echo 'we are running some # of cool things'
```

Note however, that whitespace in instruction *arguments*, such as the commands following `RUN`, are preserved, so the following example prints ` hello world` with leading whitespace as specified:

```
RUN echo "\
     hello\
     world"
```

- Parser directives are optional, and affect the way in which subsequent lines in a Dockerfile are handled.
- Parser directives are not case-sensitive. However, convention is for them to be lowercase. Convention is also to include a blank line following any parser directives. Line continuation characters are not supported in parser directives.
- Parser directive must be at top of the file otherwise it will treated as comments
- Escape character at the end of the line to go to the new line ` or \

Environment replacement

- Environment variables (declared with the ENV statement) can also be used in certain instructions as variables to be interpreted by the Dockerfile.
- Environment variables are notated in the Dockerfile either with $variable_name or ${variable_name}

- `${variable:-word}` indicates that if `variable` is set then the result will be that value. If `variable` is not set then `word` will be the result.
- `${variable:+word}` indicates that if `variable` is set then `word` will be the result, otherwise the result is the empty string.

- In all cases, word can be any string, including additional environment variables.
- Escaping is possible by adding a \ before the variable: \$foo or \${foo}, for example, will translate to $foo and ${foo} literals respectively.

```
FROM busybox
ENV FOO=/bar
WORKDIR ${FOO}   # WORKDIR /bar
ADD . $FOO       # ADD . /bar
COPY \$FOO /quux # COPY $FOO /quux
```

Environment variables are supported by the following list of instructions in the `Dockerfile`:

- `ADD`
- `COPY`
- `ENV`
- `EXPOSE`
- `FROM`
- `LABEL`
- `STOPSIGNAL`
- `USER`
- `VOLUME`
- `WORKDIR`
- `ONBUILD` (when combined with one of the supported instructions above)

We can declare the environment variables inside the ENV

```
ENV abc=hello
ENV abc=bye def=$abc
ENV ghi=$abc
```

.dockerignore file

Before the docker CLI sends the context to the docker daemon, it looks for a file named `.dockerignore` in the root directory of the context. If this file exists, the CLI modifies the context to exclude files and directories that match patterns in it. This helps to avoid unnecessarily sending large or sensitive files and directories to the daemon and potentially adding them to images using `ADD` or `COPY`.

Here is an example `.dockerignore` file:

```
# comment
*/temp*
*/*/temp*
temp?
```

Rule	Behavior
`# comment`	Ignored.
`*/temp*`	Exclude files and directories whose names start with `temp` in any immediate subdirectory of the root. For example, the plain file `/somedir/temporary.txt` is excluded, as is the directory `/somedir/temp`.
`*/*/temp*`	Exclude files and directories starting with `temp` from any subdirectory that is two levels below the root. For example, `/somedir/subdir/temporary.txt` is excluded.
`temp?`	Exclude files and directories in the root directory whose names are a one-character extension of `temp`. For example, `/tempa` and `/tempb` are excluded.

- **/*.go will exclude all files that end with .go that are found in all directories, including the root of the build context.

Lines starting with `!` (exclamation mark) can be used to make exceptions to exclusions. The following is an example `.dockerignore` file that uses this mechanism:

```
*.md
!README.md
```

All markdown files *except* `README.md` are excluded from the context.

Now consider this example:

```
*.md
README-secret.md
!README*.md
```

All of the README files are included. The middle line has no effect because `!README*.md` matches `README-secret.md` and comes last.

You can even use the `.dockerignore` file to exclude the `Dockerfile` and `.dockerignore` files. These files are still sent to the daemon because it needs them to do its job. But the `ADD` and `COPY` instructions do not copy them to the image.

- The FROM instruction initializes a new build stage and sets the Base Image for subsequent instructions. As such, a valid Docker file must start with a FROM instruction. The image can be any valid image – it is especially easy to start by pulling an image from the Public Repositories.
- ARG is the only instruction that may precede FROM in the Docker file.

An `ARG` declared before a `FROM` is outside of a build stage, so it can't be used in any instruction after a `FROM`. To use the default value of an `ARG` declared before the first `FROM` use an `ARG` instruction without a value inside of a build stage:

```
ARG VERSION=latest
FROM busybox:$VERSION
ARG VERSION
RUN echo $VERSION > image_version
```

RUN COMMAND

```
RUN /bin/bash -c 'source $HOME/.bashrc; \
echo $HOME'
```

Together they are equivalent to this single line:

```
RUN /bin/bash -c 'source $HOME/.bashrc; echo $HOME'
```

To use a different shell, other than '/bin/sh', use the *exec* form passing in the desired shell. For example:

```
RUN ["/bin/bash", "-c", "echo hello"]
```

Note

The *exec* form is parsed as a JSON array, which means that you must use double-quotes (") around words not single-quotes (').

Note

In the *JSON* form, it is necessary to escape backslashes. This is particularly relevant on Windows where the backslash is the path separator. The following line would otherwise be treated as *shell* form due to not being valid JSON, and fail in an unexpected way:

```
RUN ["c:\windows\system32\tasklist.exe"]
```

The correct syntax for this example is:

```
RUN ["c:\\windows\\system32\\tasklist.exe"]
```

RUN is an image build step, the state of the container after a RUN command will be committed to the container image. A Dockerfile can have many RUN steps that layer on top of one another to build the image. CMD is the command the container executes by default when you launch the built image. A Dockerfile will only use the final CMD defined.

CMD

The `CMD` instruction has three forms:

- `CMD ["executable","param1","param2"]` (*exec* form, this is the preferred form)
- `CMD ["param1","param2"]` (as *default parameters to ENTRYPOINT*)
- `CMD command param1 param2` (*shell* form)

There can only be one `CMD` instruction in a `Dockerfile`. If you list more than one `CMD` then only the last `CMD` will take effect.

The main purpose of a `CMD` is to provide defaults for an executing container. These defaults can include an executable, or they can omit the executable, in which case you must specify an `ENTRYPOINT` instruction as well.

- If you want shell processing then either use the shell form or execute a shell directly, for example: CMD ["sh", "-c", "echo $HOME"]. When using the exec form and executing a shell directly, as in the case for the shell form, it is the shell that is doing the environment variable expansion, not docker.

If you use the *shell* form of the `CMD` , then the `<command>` will execute in `/bin/sh -c` :

```
FROM ubuntu
CMD echo "This is a test." | wc -
```

If you want to **run your** `<command>` **without a shell** then you must express the command as a JSON array and give the full path to the executable. **This array form is the preferred format of** `CMD` **.** Any additional parameters must be individually expressed as strings in the array:

```
FROM ubuntu
CMD ["/usr/bin/wc","--help"]
```

If you would like your container to run the same executable every time, then you should consider using `ENTRYPOINT` in combination with `CMD` . See *ENTRYPOINT*.

If the user specifies arguments to `docker run` then they will override the default specified in `CMD` .

> **Note**
>
> Do not confuse `RUN` with `CMD` . `RUN` actually runs a command and commits the result; `CMD` does not execute anything at build time, but specifies the intended command for the image.

LABEL COMMAND

To view an image's labels, use the `docker image inspect` command. You can use the `--format` option to show just the labels;

```
$ docker image inspect --format='' myimage
```

Regardless of the `EXPOSE` settings, you can override them at runtime by using the `-p` flag. For example

```
$ docker run -p 80:80/tcp -p 80:80/udp ...
```

ENV COMMAND

The `ENV` instruction sets the environment variable `<key>` to the value `<value>`. This value will be in the environment for all subsequent instructions in the build stage and can be replaced inline in many as well. The value will be interpreted for other environment variables, so quote characters will be removed if they are not escaped. Like command line parsing, quotes and backslashes can be used to include spaces within values.

Example:

```
ENV MY_NAME="John Doe"
ENV MY_DOG=Rex\ The\ Dog
ENV MY_CAT=fluffy
```

The environment variables set using `ENV` will persist when a container is run from the resulting image. You can view the values using `docker inspect`, and change them using `docker run --env <key>=<value>`.

Or using `ARG`, which is not persisted in the final image:

```
ARG DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get install -y ...
```

Alternative syntax

The `ENV` instruction also allows an alternative syntax `ENV <key> <value>`, omitting the `=`. For example:

```
ENV MY_VAR my-value
```

This syntax does not allow for multiple environment-variables to be set in a single `ENV` instruction, and can be confusing. For example, the following sets a single environment variable (`ONE`) with value `"TWO= THREE=world"`:

ADD

ADD has two forms:

```
ADD [--chown=<user>:<group>] <src>... <dest>
ADD [--chown=<user>:<group>] ["<src>",... "<dest>"]
```

--chown command will only be applicable for Linux based containers of Docker file, when coming to windows --chown will not work.

The `ADD` instruction copies new files, directories or remote file URLs from `<src>` and adds them to the filesystem of the image at the path `<dest>`.

Multiple `<src>` resources may be specified but if they are files or directories, their paths are interpreted as relative to the source of the context of the build.

To add all files starting with “hom”:

```
ADD hom* /mydir/
```

In the example below, `?` is replaced with any single character, e.g., “home.txt”.

```
ADD hom?.txt /mydir/
```

The `<dest>` is an absolute path, or a path relative to `WORKDIR`, into which the source will be copied inside the destination container.

The example below uses a relative path, and adds “test.txt” to `<WORKDIR>/relativeDir/`:

```
ADD test.txt relativeDir/
```

Whereas this example uses an absolute path, and adds “test.txt” to `/absoluteDir/`

```
ADD test.txt /absoluteDir/
```

- The format of the --chown flag allows for either username and groupname strings or direct integer UID and GID in any combination. Providing a username without groupname or a UID without GID will use the same numeric UID as the GID.
- Valid ADD component syntaxes.

```
ADD --chown=55:mygroup files* /somedir/
ADD --chown=bin files* /somedir/
ADD --chown=1 files* /somedir/
ADD --chown=10:11 files* /somedir/
```

If you build by passing a `Dockerfile` through STDIN (`docker build - < somefile`), there is no build context, so the `Dockerfile` can only contain a URL based `ADD` instruction.
You can also pass a compressed archive through STDIN:
(`docker build - < archive.tar.gz`), the `Dockerfile` at the root of the archive and the rest of the archive will be used as the context of the build.

If your URL files are protected using authentication, you need to use `RUN wget`, `RUN curl` or use another tool from within the container as the `ADD` instruction does not support authentication.

`ADD` obeys the following rules:

- The `<src>` path must be inside the *context* of the build; you cannot `ADD ../something /something`, because the first step of a `docker build` is to send the context directory (and subdirectories) to the docker daemon.
- If `<src>` is a URL and `<dest>` does not end with a trailing slash, then a file is downloaded from the URL and copied to `<dest>`.
- If `<src>` is a URL and `<dest>` does end with a trailing slash, then the filename is inferred from the URL and the file is downloaded to `<dest>/<filename>`. For instance, `ADD http://example.com/foobar /` would create the file `/foobar`. The URL must have a nontrivial path so that an appropriate filename can be discovered in this case (`http://example.com` will not work).
- If `<src>` is a directory, the entire contents of the directory are copied, including filesystem metadata.

> **Note**
>
> The directory itself is not copied, just its contents.

- If `<src>` is a *local* tar archive in a recognized compression format (identity, gzip, bzip2 or xz) then it is unpacked as a directory. Resources from *remote* URLs are **not** decompressed. When a directory is copied or unpacked, it has the same behavior as `tar -x`, the result is the union of:
- If `<src>` is any other kind of file, it is copied individually along with its metadata. In this case, if `<dest>` ends with a trailing slash `/`, it will be considered a directory and the contents of `<src>` will be written at `<dest>/base(<src>)`.
- If multiple `<src>` resources are specified, either directly or due to the use of a wildcard, then `<dest>` must be a directory, and it must end with a slash `/`.
- If `<dest>` does not end with a trailing slash, it will be considered a regular file and the contents of `<src>` will be written at `<dest>`.
- If `<dest>` doesn't exist, it is created along with all missing directories in its path.

COPY

COPY has two forms:

```
COPY [--chown=<user>:<group>] <src>... <dest>
COPY [--chown=<user>:<group>] ["<src>",... "<dest>"]
```

The `COPY` instruction copies new files or directories from `<src>` and adds them to the filesystem of the container at the path `<dest>`.

Multiple `<src>` resources may be specified but the paths of files and directories will be interpreted as relative to the source of the context of the build.

To add all files starting with “hom”:

```
COPY hom* /mydir/
```

In the example below, `?` is replaced with any single character, e.g., “home.txt”.

```
COPY hom?.txt /mydir/
```

The `<dest>` is an absolute path, or a path relative to `WORKDIR`, into which the source will be copied inside the destination container.

The example below uses a relative path, and adds “test.txt” to `<WORKDIR>/relativeDir/`:

```
COPY test.txt relativeDir/
```

Whereas this example uses an absolute path, and adds “test.txt” to `/absoluteDir/`

```
COPY test.txt /absoluteDir/
```

- Valid syntaxes which include chown flag

```
COPY --chown=55:mygroup files* /somedir/
COPY --chown=bin files* /somedir/
COPY --chown=1 files* /somedir/
COPY --chown=10:11 files* /somedir/
```

- We are having the same rules of COPY command for the ADD command also.

ENTRYPOINT

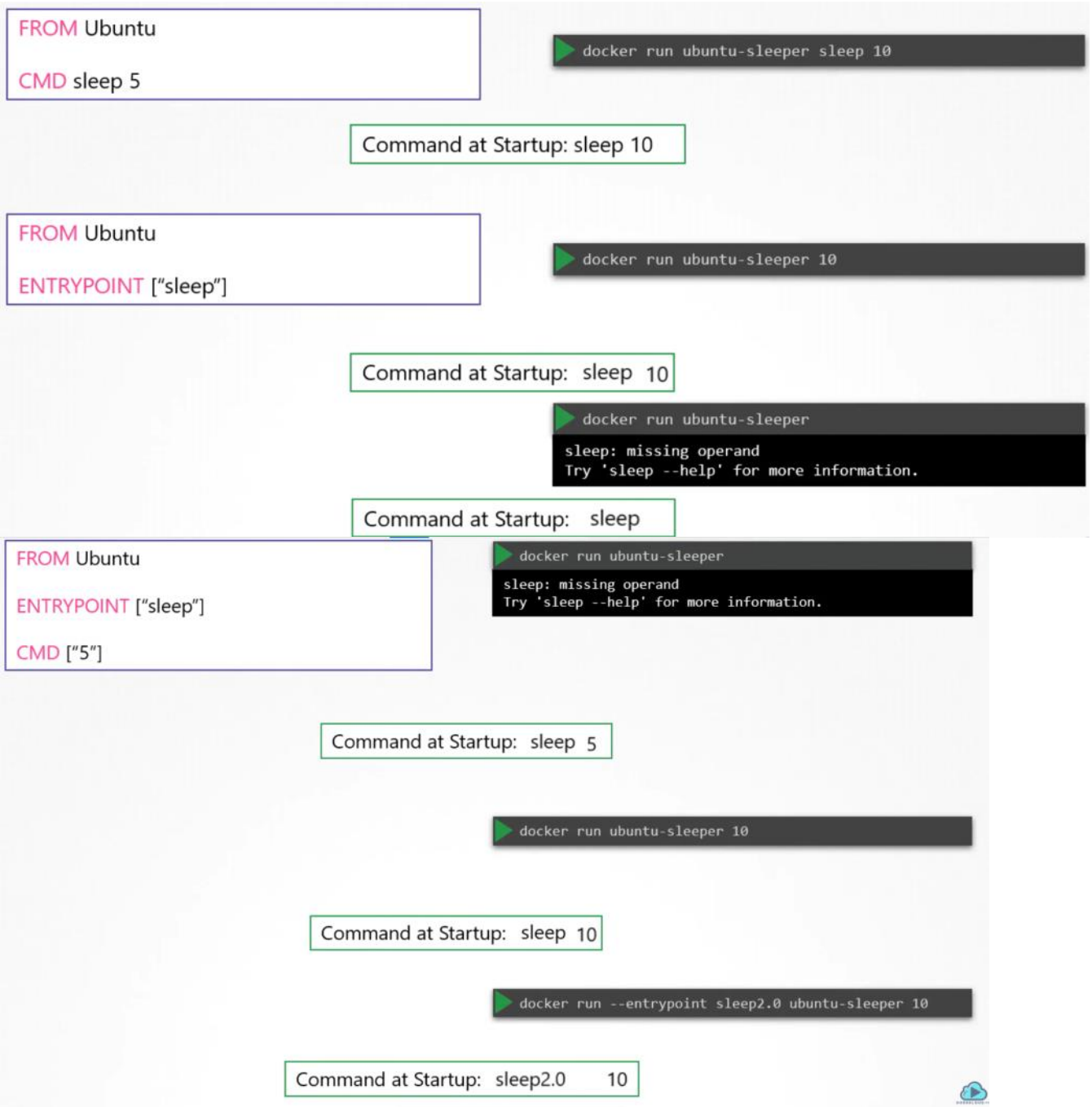
ENTRYPOINT has two forms:

The *exec* form, which is the preferred form:

```
ENTRYPOINT ["executable", "param1", "param2"]
```

The *shell* form:

```
ENTRYPOINT command param1 param2
```



Understand how CMD and ENTRYPOINT interact

Both `CMD` and `ENTRYPOINT` instructions define what command gets executed when running a container. There are few rules that describe their co-operation.

1. Dockerfile should specify at least one of `CMD` or `ENTRYPOINT` commands.
2. `ENTRYPOINT` should be defined when using the container as an executable.
3. `CMD` should be used as a way of defining default arguments for an `ENTRYPOINT` command or for executing an ad-hoc command in a container.
4. `CMD` will be overridden when running the container with alternative arguments.

VOLUME

```
VOLUME ["/data"]
```

The `docker run` command initializes the newly created volume with any data that exists at the specified location within the base image. For example, consider the following Dockerfile snippet:

```
FROM ubuntu
RUN mkdir /myvol
RUN echo "hello world" > /myvol/greeting
VOLUME /myvol
```

This Dockerfile results in an image that causes `docker run` to create a new mount point at `/myvol` and copy the `greeting` file into the newly created volume.

- **Changing the volume from within the Dockerfile**: If any build steps change the data within the volume after it has been declared, those changes will be discarded.
- **JSON formatting**: The list is parsed as a JSON array. You must enclose words with double quotes (`"`) rather than single quotes (`'`).

USER

```
USER <user>[:<group>]
```

or

```
USER <UID>[:<GID>]
```

WORKDIR

```
WORKDIR /path/to/workdir
```

The `WORKDIR` instruction sets the working directory for any `RUN` , `CMD` , `ENTRYPOINT` , `COPY` and `ADD` instructions that follow it in the `Dockerfile` . If the `WORKDIR` doesn't exist, it will be created even if it's not used in any subsequent `Dockerfile` instruction.

The `WORKDIR` instruction can be used multiple times in a `Dockerfile` . If a relative path is provided, it will be relative to the path of the previous `WORKDIR` instruction. For example:

```
WORKDIR /a
WORKDIR b
WORKDIR c
RUN pwd
```

The output of the final `pwd` command in this `Dockerfile` would be `/a/b/c` .

The `WORKDIR` instruction can resolve environment variables previously set using `ENV`. You can only use environment variables explicitly set in the `Dockerfile`. For example:

```
ENV DIRPATH=/path
WORKDIR $DIRPATH/$DIRNAME
RUN pwd
```

The output of the final `pwd` command in this `Dockerfile` would be `/path/$DIRNAME`

If not specified, the default working directory is `/`. In practice, if you aren't building a Dockerfile from scratch (`FROM scratch`), the `WORKDIR` may likely be set by the base image you're using.

Therefore, to avoid unintended operations in unknown directories, it is best practice to set your `WORKDIR` explicitly.

ARG

```
ARG <name>[=<default value>]
```

The `ARG` instruction defines a variable that users can pass at build-time to the builder with the `docker build` command using the `--build-arg <varname>=<value>` flag. If a user specifies a build argument that was not defined in the Dockerfile, the build outputs a warning.

```
[Warning] One or more build-args [foo] were not consumed.
```

A Dockerfile may include one or more `ARG` instructions. For example, the following is a valid Dockerfile:

```
FROM busybox
ARG user1
ARG buildno
# ...
```

Default values

An `ARG` instruction can optionally include a default value:

```
FROM busybox
ARG user1=someuser
ARG buildno=1
# ...
```

If an `ARG` instruction has a default value and if there is no value passed at build-time, the builder uses the default.

You can use an `ARG` or an `ENV` instruction to specify variables that are available to the `RUN` instruction. Environment variables defined using the `ENV` instruction always override an `ARG` instruction of the same name. Consider this Dockerfile with an `ENV` and `ARG` instruction.

```
FROM ubuntu
ARG CONT_IMG_VER
ENV CONT_IMG_VER=v1.0.0
RUN echo $CONT_IMG_VER
```

Then, assume this image is built with this command:

```
$ docker build --build-arg CONT_IMG_VER=v2.0.1 .
```

In this case, the `RUN` instruction uses `v1.0.0` instead of the `ARG` setting passed by the user: `v2.0.1` This behavior is similar to a shell script where a locally scoped variable overrides the variables passed as arguments or inherited from environment, from its point of definition. Unlike an `ARG` instruction, `ENV` values are always persisted in the built image. Consider a docker build without the `--build-arg` flag:

```
$ docker build .
```

Using this Dockerfile example, `CONT_IMG_VER` is still persisted in the image but its value would be `v1.0.0` as it is the default set in line 3 by the `ENV` instruction.

Improve Docker file efficiency

Tuesday, April 12, 2022 12:15 PM

- We can have build context at one directory and Docker file at another directory.

- Piping helps us to save memory for creating the Dockerfile , we can create Dockerfile through pipes and if we create using pipes the file will not persist later in the Disk.

Pipe Dockerfile through `stdin`

Docker has the ability to build images by piping `Dockerfile` through `stdin` with a *local or remote build context*. Piping a `Dockerfile` through `stdin` can be useful to perform one-off builds without writing a Dockerfile to disk, or in situations where the `Dockerfile` is generated, and should not persist afterwards.

> **The examples in this section use here documents for convenience, but any method to provide the `Dockerfile` on `stdin` can be used.**
>
> For example, the following commands are equivalent:

```
echo -e 'FROM busybox\nRUN echo "hello world"' | docker build -
```

```
docker build -<<EOF
FROM busybox
RUN echo "hello world"
EOF
```

> You can substitute the examples with your preferred approach, or the approach that best fits your use-case.

Build an image using a Dockerfile from stdin, without sending build context

Use this syntax to build an image using a `Dockerfile` from `stdin`, without sending additional files as build context. The hyphen (`-`) takes the position of the `PATH`, and instructs Docker to read the build context (which only contains a `Dockerfile`) from `stdin` instead of a directory:

```
docker build -t myimage:latest -<<EOF
FROM busybox
RUN echo "hello world"
EOF
```

- As these files are not copied into the image , we increase our build speed as no files are sent to the daemon. Or we can exclude the files using dockerignore file.
- These types of pipelining will not work for the COPY and ADD command of Dockerfile Components. The above syntax is not correct for these type of commands.

Build from a local build context, using a Dockerfile from stdin

Use this syntax to build an image using files on your local filesystem, but using a `Dockerfile` from `stdin`. The syntax uses the `-f` (or `--file`) option to specify the `Dockerfile` to use, using a hyphen (`-`) as filename to instruct Docker to read the `Dockerfile` from `stdin`:

```
docker build [OPTIONS] -f- PATH
```

The example below uses the current directory (`.`) as the build context, and builds an image using a `Dockerfile` that is passed through `stdin` using a here document.

```
# create a directory to work in
mkdir example
cd example

# create an example file
touch somefile.txt

# build an image using the current directory as context, and a Dockerfile passed th
```

```
touch somefile.txt

# build an image using the current directory as context, and a Dockerfile passed thr
docker build -t myimage:latest -f- . <<EOF
FROM busybox
COPY somefile.txt ./
RUN cat /somefile.txt
EOF
```

Build from a remote build context, using a Dockerfile from stdin

Use this syntax to build an image using files from a remote `git` repository, using a `Dockerfile` from `stdin`. The syntax uses the `-f` (or `--file`) option to specify the `Dockerfile` to use, using a hyphen (`-`) as filename to instruct Docker to read the `Dockerfile` from `stdin`:

```
docker build [OPTIONS] -f- PATH
```

This syntax can be useful in situations where you want to build an image from a repository that does not contain a `Dockerfile`, or if you want to build with a custom `Dockerfile`, without maintaining your own fork of the repository.

The example below builds an image using a `Dockerfile` from `stdin`, and adds the `hello.c` file from the "hello-world" Git repository on GitHub.

```
docker build -t myimage:latest -f- https://github.com/docker-library/hello-world.git <<EOF
FROM busybox
COPY hello.c ./
EOF
```

Exclude with .dockerignore

To exclude files not relevant to the build (without restructuring your source repository) use a `.dockerignore` file. This file supports exclusion patterns similar to `.gitignore` files. For information on creating one, see the .dockerignore file.

Don't install unnecessary packages

To reduce complexity, dependencies, file sizes, and build times, avoid installing extra or unnecessary packages just because they might be "nice to have." For example, you don't need to include a text editor in a database image.

Minimize the number of layers

In older versions of Docker, it was important that you minimized the number of layers in your images to ensure they were performant. The following features were added to reduce this limitation:

- Only the instructions `RUN`, `COPY`, `ADD` create layers. Other instructions create temporary intermediate images, and do not increase the size of the build.
- Where possible, use multi-stage builds, and only copy the artifacts you need into the final image. This allows you to include tools and debug information in your intermediate build stages without increasing the size of the final image.

Sort multi-line arguments

Whenever possible, ease later changes by sorting multi-line arguments alphanumerically. This helps to avoid duplication of packages and make the list much easier to update. This also makes PRs a lot easier to read and review. Adding a space before a backslash (`\`) helps as well.

Here's an example from the `buildpack-deps` image:

```
RUN apt-get update && apt-get install -y \
  bzr \
  cvs \
  git \
  mercurial \
  subversion \
  && rm -rf /var/lib/apt/lists/*
```

Dockerfile instructions

These recommendations are designed to help you create an efficient and maintainable `Dockerfile` .

FROM

Dockerfile reference for the FROM instruction

Whenever possible, use current official images as the basis for your images. We recommend the Alpine image as it is tightly controlled and small in size (currently under 6 MB), while still being a full Linux distribution.

LABEL

Understanding object labels

You can add labels to your image to help organize images by project, record licensing information, to aid in automation, or for other reasons. For each label, add a line beginning with `LABEL` and with one or more key-value pairs. The following examples show the different acceptable formats. Explanatory comments are included inline.

> Strings with spaces must be quoted **or** the spaces must be escaped. Inner quote characters (`"`), must also be escaped.

```
# Set one or more individual labels
LABEL com.example.version="0.0.1-beta"
LABEL vendor1="ACME Incorporated"
LABEL vendor2=ZENITH\ Incorporated
LABEL com.example.release-date="2015-02-12"
LABEL com.example.version.is-production=""
```

An image can have more than one label. Prior to Docker 1.10, it was recommended to combine all labels into a single `LABEL` instruction, to prevent extra layers from being created. This is no longer necessary, but combining labels is still supported.

```
# Set multiple labels on one line
LABEL com.example.version="0.0.1-beta" com.example.release-date="2015-02-12"
```

The above can also be written as:

```
# Set multiple labels at once, using line-continuation characters to break long lines
LABEL vendor=ACME\ Incorporated \
      com.example.is-beta= \
      com.example.is-production="" \
      com.example.version="0.0.1-beta" \
      com.example.release-date="2015-02-12"
```

RUN

Dockerfile reference for the RUN instruction

Split long or complex `RUN` statements on multiple lines separated with backslashes to make your `Dockerfile` more readable, understandable, and maintainable.

Using `apt-get update` alone in a `RUN` statement causes caching issues and subsequent `apt-get instal` instructions fail. For example, say you have a Dockerfile:

```
# syntax=docker/dockerfile:1
FROM ubuntu:18.04
RUN apt-get update
RUN apt-get install -y curl
```

Using `RUN apt-get update && apt-get install -y` ensures your Dockerfile installs the latest package versions with no further coding or manual intervention. This technique is known as "cache busting". You can also achieve cache-busting by specifying a package version. This is known as version pinning, for example:

```
RUN apt-get update && apt-get install -y \
    package-bar \
    package-baz \
    package-foo=1.3.*
```

```
RUN apt-get update && apt-get install -y \
    aufs-tools \
    automake \
    build-essential \
    curl \
    dpkg-sig \
    libcap-dev \
    libsqlite3-dev \
    mercurial \
    reprepro \
    ruby1.9.1 \
    ruby1.9.1-dev \
    s3cmd=1.1.* \
 && rm -rf /var/lib/apt/lists/*
```

In addition, when you clean up the apt cache by removing `/var/lib/apt/lists` it reduces the image size, since the apt cache is not stored in a layer. Since the `RUN` statement starts with `apt-get update`, the package cache is always refreshed prior to `apt-get install`.

CMD

Dockerfile reference for the CMD instruction

The `CMD` instruction should be used to run the software contained in your image, along with any arguments. `CMD` should almost always be used in the form of `CMD ["executable", "param1", "param2"…]`. Thus, if the image is for a service, such as Apache and Rails, you would run something like `CMD ["apache2","-DFOREGROUND"]`. Indeed, this form of the instruction is recommended for any service-based image.

In most other cases, `CMD` should be given an interactive shell, such as bash, python and perl. For example, `CMD ["perl", "-de0"]`, `CMD ["python"]`, or `CMD ["php", "-a"]`. Using this form means that when you execute something like `docker run -it python`, you'll get dropped into a usable shell, ready to go. `CMD` should rarely be used in the manner of `CMD ["param", "param"]` in conjunction with `ENTRYPOINT`, unless you and your expected users are already quite familiar with how `ENTRYPOINT` works.

- Setting and unsetting the environmental variable using RUN command. Unsetting helps us to save the space of the image. Because after setting a environmental variable , we will create a new layer and then when we try to un set the environmental variable in different command , we may dump the data . But the layer persists. To remove this problem we do all the operations in the single command. Then the extra created layer will also be deleted .

```
# syntax=docker/dockerfile:1
FROM alpine
RUN export ADMIN_USER="mark" \
    && echo $ADMIN_USER > ./mark \
    && unset ADMIN_USER
CMD sh
```

```
$ docker run --rm test sh -c 'echo $ADMIN_USER'
```

ADD or COPY

- Dockerfile reference for the ADD instruction
- Dockerfile reference for the COPY instruction

Although `ADD` and `COPY` are functionally similar, generally speaking, `COPY` is preferred. That's because it's more transparent than `ADD`. `COPY` only supports the basic copying of local files into the container, while `ADD` has some features (like local-only tar extraction and remote URL support) that are not immediately obvious. Consequently, the best use for `ADD` is local tar file auto-extraction into the image, as in `ADD rootfs.tar.xz /`.

Because image size matters, using `ADD` to fetch packages from remote URLs is strongly discouraged; you should use `curl` or `wget` instead. That way you can delete the files you no longer need after they've been extracted and you don't have to add another layer in your image. For example, you should avoid doing things like:

```
ADD https://example.com/big.tar.xz /usr/src/things/
RUN tar -xJf /usr/src/things/big.tar.xz -C /usr/src/things
RUN make -C /usr/src/things all
```

And instead, do something like:

```
RUN mkdir -p /usr/src/things \
    && curl -SL https://example.com/big.tar.xz \
    | tar -xJC /usr/src/things \
    && make -C /usr/src/things all
```

For other items (files, directories) that do not require `ADD`'s tar auto-extraction capability, you should always use `COPY`.

ENTRYPOINT

Dockerfile reference for the ENTRYPOINT instruction

The best use for `ENTRYPOINT` is to set the image's main command, allowing that image to be run as though it was that command (and then use `CMD` as the default flags).

Let's start with an example of an image for the command line tool `s3cmd` :

```
ENTRYPOINT ["s3cmd"]
CMD ["--help"]
```

Now the image can be run like this to show the command's help:

```
$ docker run s3cmd
```

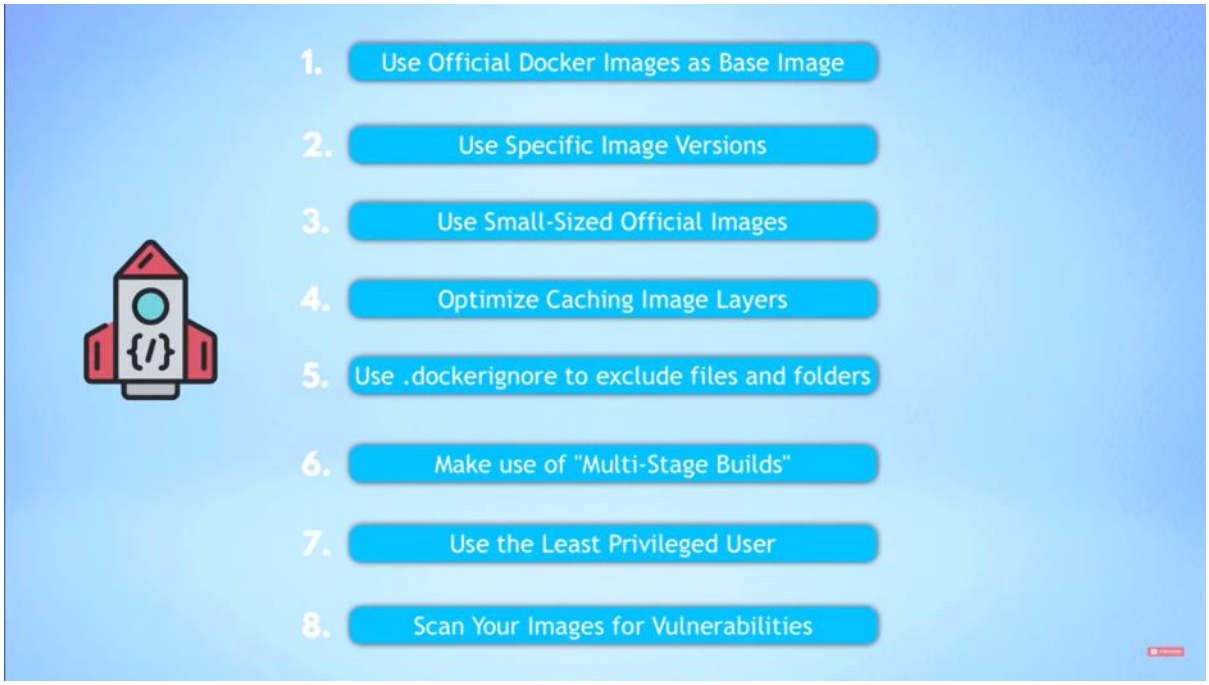
Or using the right parameters to execute a command:

```
$ docker run s3cmd ls s3://mybucket
```

Best Practices of Docker Images

Tuesday, April 12, 2022 2:38 PM

[Top 8 Docker Best Practices for using Docker in Production ☑ - DEV Community]



Docker ln command

Wednesday, April 20, 2022 12:53 PM

ln command with symbolic links in Linux

- The symbolic link is a connection to the specified file or folder in Linux. Also, the provided symbolic link points to the source file or folder.
- There are two types of symbolic links types called hard links and soft links.
- If you delete the original file, the soft link has no value, because it points to a non-existent file. But in the case of hard link, it is entirely opposite. Even if you delete the original file, the hard link will still has the data of the original file.
- Soft links and hard links are similar to the shortcuts in the windows operating system.

ln Command Syntax

The ln command has the following syntax. The default link type for the ln command is a hard link.

```
ls OPTION SOURCE LINK
```

- **OPTION** is used to set the link type or similar.
- **SOURCE** is the source or actual file or folder.
- **LINK** is the newly created link for a file or folder.

Create Soft Link For File

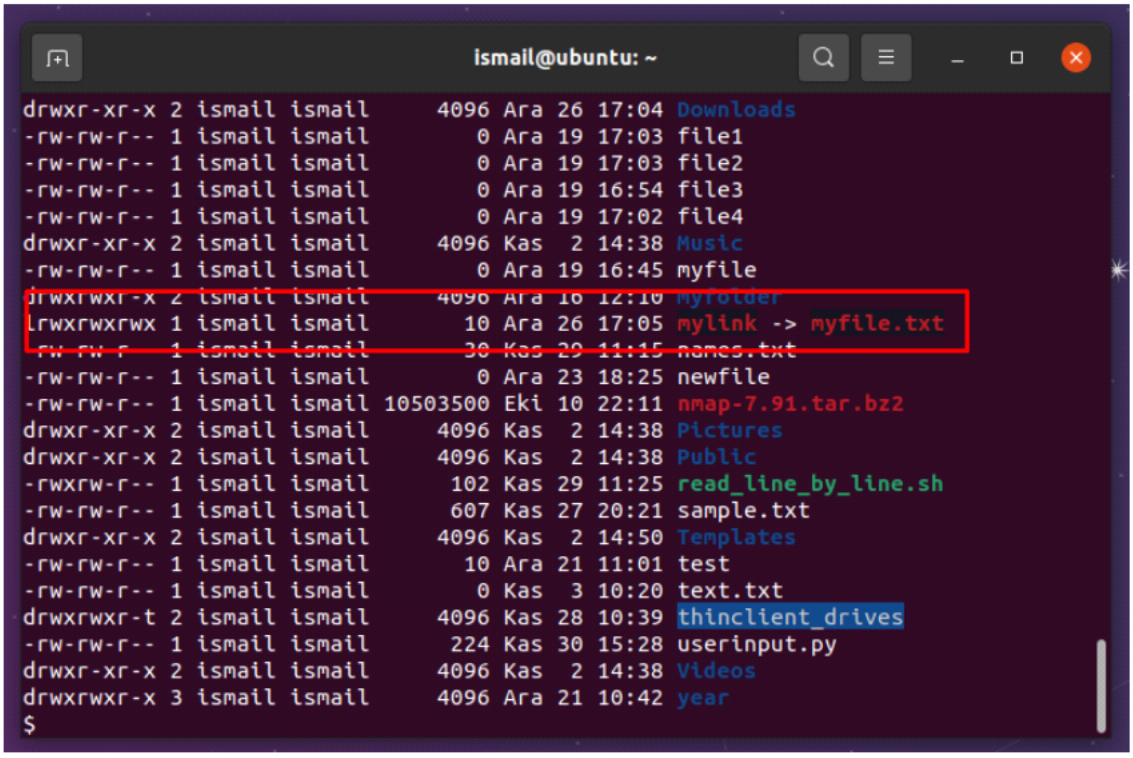
The most popular use case for the ln command is creating the soft link. In order to create a soft link, we will use the **-s** or **–symbolic** option with the ln command. Then we will provide the source which can be a file and the link. In the following example, we will create a soft link for the file named myfile.txt with the name of mylink.

```
$ ln -s myfile.txt mylink
```

```
$ ln -s /home/ismail/myfile.txt /var/mylink
```

- We are creating a soft link for the file named myfile.txt as mylink and that link is located under , var folder of destination. The above provided paths are absolute in nature.
- These links are useful to utilize the files at some location , without importing locally . They access those files by creating links of those files locally and use them for accessing.
- For example, if we have some files locally on the system and we need to utilize those in the containers, instead of copying the files into the container, we use them through these links. This results in saving memory space and easy accessing.

In terminal "**ls -l**" command can be used to list files and their types where the links will be listed and their source also provided with the -> sing. Also, the file type contains the l to describe it as a link type at the start of the permissions.



- Not only files are used to create soft links a file or directory link can be created by using the ln command.

```
$ ln -s /var/lib /home/ismail/lib
```

Overwrite Existing Symbolic Link

In some cases, a file or symbolic can exist already and we may try to create the same name link. We will get the following error.

```
ln: failed to create symbolic link 'mylink': File exists
```

We should overwrite the existing link or file to create a new symbolic link. This requires overwriting the existing link. The -f option is used to overwrite the existing link.

```
$ ln -f -s myfile.txt mylink
```

Create Hard Link For File and Folder

A hard link is different from a soft link where the real file inodes are copied into the link. If you rm this link the original file will be deleted too. Hard link creation does not require a parameter for the ln command.

```
$ ln myfile.txt mylink
```

We can also create a hard link with the full or absolute path like below.

```
$ ln /home/ismail/myfile.txt /var/myfile
```

- The unlink command can delete both soft links and hard links also with an absolute or relational path.

In the following example, we delete the link named `mylink` using the unlink command.

```
$ unlink mylink
```

Find and Delete Broken Symbolic Links

During time symbolic links are created, used, removed even broken. A symbolic link is broken when the destination file or directory is removed or changed. Broken links can be easily found or deleted using the `find` command.

```
$ find /home/ibaydan -xtype l -delete
```

Docker mkdir command

Wednesday, April 20, 2022 1:38 PM

- The mkdir command is used to create directories in our Linux Operating System. Every directory that needs to be created will also contain some files or subdirectories that a user wants. With the help of mkdir command we can create multiple directories with setting the directories with permissions. Without permission we cannot create directories and we will face with an error saying “permission denied”.

The syntax of the Mkdir Command in Linux is given below.

```
mkdir  [OPTION]... DIRECTORY...
```

This command is the basic command to create the directory. We just have to name the directory we want to create.If we want to create multiple directories, we can use following command.

```
mkdir Doc media pics
```

We need to give the name of the directories with a space in between to create multiple directories.

```
mkdir [-m=mode] [-p] [-v] [-Z=context] directory [directory ...]
```

There are plenty of options that can be used with the mkdir command.

-m option: This -m option is used to set the mode of the directory i.e. permissions such as read, write and execute for the created directory.

```
$ mkdir -m a=rwx [directories]
```

-p option: This option enables the user to create parent directories as per the requirement. If the directories exist already, no error will be displayed.

```
$ mkdir -p [directories]
```

- Here while we give permissions , we have used a as an option , here a is the user name. we are giving read write and executing permissions for the user a.

-v, –verbose: This option displays a verbose information of each directory that is created.

```
$ mkdir -v [directories]
```

–version: This option will print the information of the version and exit.

-v, –verbose: This option displays a verbose information of each directory that is created.

```
$ mkdir -v [directories]
```

–version: This option will print the information of the version and exit.

```
$ mkdir –version
```

–help: This option will display the information of the mkdir command. It will list the syntax of the command and also the options used in mkdir command and a brief description of each option.

```
$ mkdir --help
```

-Z option: This option is used to set defaul SELinux rules on a particular directory at the time of creation.

```
$ mkdir -Z [directories]
```

Example #1

```
$ mkdir pics
```

The above command will help to create the directory called pics in the user's current directory.

Example #2

```
$ mkdir ~pics
```

Example #3

To create multiple directories at once use the following command.

```
mkdir sample1 sample2 sample3
```

The above command will create the directories sample1 sample2 and sample3 if they do not exist.

Example #4

```
mkdir a/b
```

The above command will create the directory named b inside the directory a. If the directory **a** does not exist then it will display an error message. Using the –p option will create the parent directory if it does not exist.

Example #5

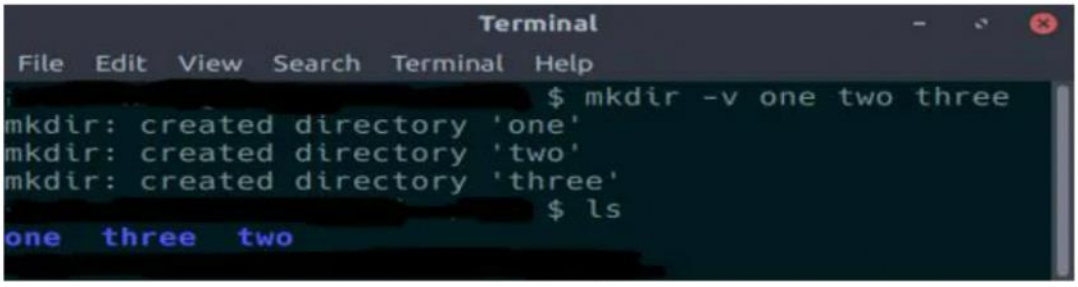
```
rmdir -p a/b
```

It is same as the command shown in example 3. mkdir command will create the directory a if it does not exist and create a directory called b inside the directory a.

- In the above example it is mkdir , it is not rmdir.
- If we create any directory in verbose mode then we would get an detailed output of what directories we have created.

We can create new directories in verbose mode by using the –v option. When you create a new directory using this option it will produce the following verbose output in the screen.

Output:



Docker User Instruction

Wednesday, April 20, 2022 2:03 PM

Docker – USER Instruction

Last Updated : 28 Oct, 2020

By default, a Docker Container runs as a Root user. This poses a great security threat if you deploy your applications on a large scale inside Docker Containers. You can change or switch to a different user inside a Docker Container using the **USER** Instruction. For this, you first need to create a user and a group inside the Container.

Step 1: Create the Dockerfile

You can specify the instructions to create a new user and group and to switch the user both in the *Dockerfile*. For this example, we will simply create an Ubuntu Image and use the bash with a different user other than the Root user.

```
FROM ubuntu:latest
RUN apt-get -y update
RUN groupadd -r user && useradd -r -g user user
USER user
```

In the above *dockerfile*, we have pulled the base Image Ubuntu and updated it. We have created a new group called **user** and a new user inside the group with the same name. Using the **USER** option, we have then switched the user.

Step 2: Build the Docker Image

After creating the *Dockerfile*, we can now create the Docker Image using the Build command.

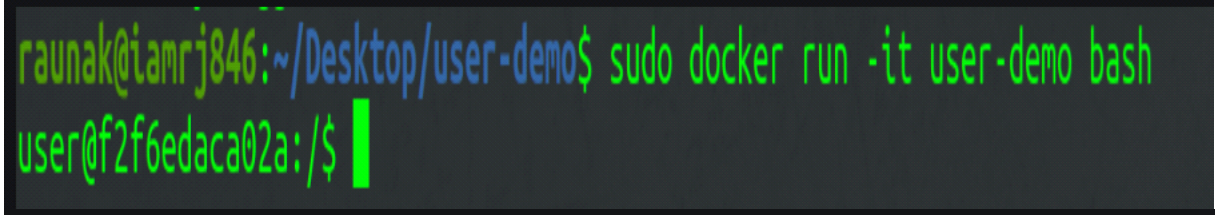
```
sudo docker build -t user-demo .
```

```
raunak@iamrj846:~/Desktop/user-demo$ sudo docker build -t user-demo .
[sudo] password for raunak:
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM ubuntu:latest
 ---> 9140108b62dc
Step 2/4 : RUN apt-get -y update
 ---> Using cache
 ---> 5e55f8bef1ae
Step 3/4 : RUN groupadd -r user && useradd -r -g user user
 ---> Running in 6814b19f78fe
Removing intermediate container 6814b19f78fe
 ---> 4c44b07b179c
Step 4/4 : USER user
 ---> Running in e707e2402ba6
Removing intermediate container e707e2402ba6
 ---> f8d65864ff58
Successfully built f8d65864ff58
Successfully tagged user-demo:latest
raunak@iamrj846:~/Desktop/user-demo$
```

Step 3: Run the Docker Container

Use the Docker Run command to run the Container.

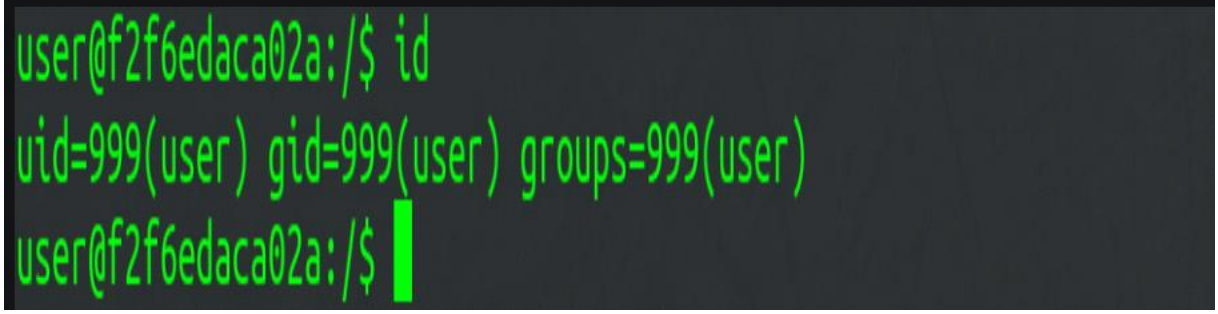
```
sudo docker run -it user-demo bash
```



Step 4: Verify the output

You can now check that the default user and the group have now changed to the one we created in the *Dockerfile* using the **id** command.

```
id
```



Docker yum command

Wednesday, April 20, 2022 2:26 PM

- YUM command (Yellowdog Updater Modified) is the traditional package manager for RedHat based systems.

yum command Basic Usage

The general syntax of YUM command is

```
yum [options] <command> [<args>...]
```

Available commands include **install**, **search**, **query**, etc. `args` can be a package name, a group name, or subcommand(s) specific to the 'command'.

Managing packages using the yum command

Let's now see how we can use the yum command to install/remove/query packages on our RedHat based system.

1. Search and Install packages

Let's install Syncthing – the file-syncing application using the yum command. But you may not know the exact name of the package. It's better to search for the package first. You can use the **search** command of YUM for searching packages.

```
yum search syncthing
```

```
root@li1986-214:~
[14:32:02] # yum search syncthing
Last metadata expiration check: 2:54:30 ago on Thu 26 Nov 2020 11:37:37 AM UTC.
====================== Name Exactly Matched: syncthing ======================
syncthing.x86_64 : Continuous File Synchronization
========================== Name Matched: syncthing ==========================
syncthing-cli.x86_64 : Continuous File Synchronization (CLI)
syncthing-tools.x86_64 : Continuous File Synchronization (server tools)
```

Yum Search Syncthing

In our case, the name of the package is also **syncthing**. Once you know the exact package name, you can use the **install** command of YUM for installing that package.

```
1   yum install syncthing
```

```
root@li1986-214:~
[14:32:08] # yum install syncthing
Last metadata expiration check: 2:59:17 ago on Thu 26 Nov 2020 11:37:37 AM UTC.
Dependencies resolved.
================================================================================
 Package            Architecture    Version                Repository     Size
================================================================================
Installing:
 syncthing          x86_64          1.8.0-2.el8.1          epel          9.4 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 9.4 M
Installed size: 33 M
Is this ok [y/N]:
```

2. List information about a package

To list more information about a package, use the **`info`** command of YUM.

```
root@li1986-214:~
[18:26:47] # yum info syncthing
Last metadata expiration check: 0:18:03 ago on Mon 30 Nov 2020 06:08:52 PM UTC.
Available Packages
Name         : syncthing
Version      : 1.8.0
Release      : 2.el8.1
Architecture : x86_64
Size         : 9.4 M
Source       : syncthing-1.8.0-2.el8.1.src.rpm
Repository   : epel
Summary      : Continuous File Synchronization
URL          : https://syncthing.net
License      : MPLv2.0 and MIT and OFL and CC-BY and BSD and ASL 2.0 and CC0 and ISC
Description  : Syncthing replaces other file synchronization services with something
             : open, trustworthy and decentralized. Your data is your data alone and
             : you deserve to choose where it is stored, if it is shared with some
             : third party and how it's transmitted over the Internet. Using
             : syncthing, that control is returned to you.
             :
             : This package contains the syncthing client binary and systemd
             : services.
```

YUM Info Syncthing

3. List all installed packages using yum

To see the list of installed packages, you can use the **`list installed`** command of YUM.

```
yum list installed
```

```
root@li1986-214:~
[14:41:34] # yum list installed
Installed Packages
NetworkManager.x86_64                1:1.22.8-5.el8_2                  @anaconda
NetworkManager-libnm.x86_64          1:1.22.8-5.el8_2                  @anaconda
NetworkManager-team.x86_64           1:1.22.8-5.el8_2                  @anaconda
NetworkManager-tui.x86_64            1:1.22.8-5.el8_2                  @anaconda
acl.x86_64                           2.2.53-1.el8                      @anaconda
audit.x86_64                         3.0-0.17.20191104git1c2f876.el8   @anaconda
audit-libs.x86_64                    3.0-0.17.20191104git1c2f876.el8   @anaconda
authselect.x86_64                    1.1-2.el8                         @anaconda
authselect-compat.x86_64             1.1-2.el8                         @AppStream
authselect-libs.x86_64               1.1-2.el8                         @anaconda
basesystem.noarch                    11-5.el8                          @anaconda
bash.x86_64                          4.4.19-10.el8                     @anaconda
bind-export-libs.x86_64              32:9.11.13-5.el8_2                @anaconda
binutils.x86_64                      2.30-73.el8                       @BaseOS
biosdevname.x86_64                   0.7.3-2.el8                       @anaconda
brotli.x86_64                        1.0.6-1.el8                       @anaconda
bzip2-libs.x86_64                    1.0.6-26.el8                      @anaconda
```

```
yum list installed | grep vim
```

```
root@li1986-214:~
[14:41:39] # yum list installed | grep vim
vim-common.x86_64                    2:8.0.1763-13.el8                 @AppStream
vim-enhanced.x86_64                  2:8.0.1763-13.el8                 @AppStream
vim-filesystem.noarch                2:8.0.1763-13.el8                 @AppStream
vim-minimal.x86_64                   2:8.0.1763-13.el8                 @anaconda
```

YUM List Installed Vim

If it didn't produce any output, it means that the package is not installed. In that case.

4. Remove a package

To remove a package, use the **remove** command of YUM.

```
yum remove syncthing
```

```
root@li1986-214:~
[16:51:09] # yum remove syncthing
Dependencies resolved.
================================================================================
 Package              Architecture      Version                  Repository      Size
================================================================================
Removing:
 syncthing            x86_64            1.8.0-2.el8.1            @epel           33 M

Transaction Summary
================================================================================
Remove  1 Package

Freed space: 33 M
Is this ok [y/N]:
```

YUM Remove

To remove all unneeded packages that were originally installed as dependencies, use the **autoremove** command

```
yum autoremove
```

5. Upgrade a package using the yum command

To upgrade all the packages that can be upgraded, use the **upgrade** command

```
yum upgrade
```

```
root@li1986-214:~
[19:02:09] # yum upgrade
Last metadata expiration check: 0:55:18 ago on Mon 30 Nov 2020 06:08:52 PM UTC.
Dependencies resolved.
================================================================================
 Package                Arch   Version                              Repo      Size
================================================================================
Installing:
 kernel                 x86_64 4.18.0-193.28.1.el8_2                BaseOS   2.8 M
 kernel-core            x86_64 4.18.0-193.28.1.el8_2                BaseOS    28 M
 kernel-modules         x86_64 4.18.0-193.28.1.el8_2                BaseOS    23 M
Upgrading:
 bind-export-libs       x86_64 32:9.11.13-6.el8_2.1                 BaseOS   1.1 M
 ca-certificates        noarch 2020.2.41-80.0.el8_2                 BaseOS   391 k
 centos-gpg-keys        noarch 8.2-2.2004.0.2.el8                   BaseOS    12 k
 centos-repos           x86_64 8.2-2.2004.0.2.el8                   BaseOS    13 k
 epel-release           noarch 8-9.el8                              epel      22 k
 kernel-headers         x86_64 4.18.0-193.28.1.el8_2                BaseOS   4.0 M
```

YUM Upgrade

To upgrade a specific package, just add the name of the package, for example:

```
1   yum upgrade nftables
```

```
root@li1986-214:~
[19:06:49] # yum upgrade nftables
Last metadata expiration check: 0:58:06 ago on Mon 30 Nov 2020 06:08:52 PM UTC.
Dependencies resolved.
================================================================================
 Package                Architecture Version                    Repository    Size
================================================================================
Upgrading:
 nftables               x86_64       1:0.9.3-12.el8_2.1         BaseOS       311 k
 python3-nftables       x86_64       1:0.9.3-12.el8_2.1         BaseOS        25 k

Transaction Summary
================================================================================
Upgrade  2 Packages

Total download size: 336 k
Is this ok [y/N]:
```

6. Search and Install package groups

Package groups are just multiple packages under a single name. These packages groups can be a whole server GUI, Security Tools, Administration Tools, etc. To see the list of groups, you can use the **`group list`** command of YUM.

```
1   yum group list
```

```
root@li1986-214:~
[15:58:10] # yum group list
Last metadata expiration check: 1:25:09 ago on Thu 26 Nov 2020 02:38:02 PM UTC.
Available Environment Groups:
   Server with GUI
   Server
   Workstation
   KDE Plasma Workspaces
   Virtualization Host
   Custom Operating System
Installed Environment Groups:
   Minimal Install
Available Groups:
   Container Management
   .NET Core Development
   RPM Development Tools
   Development Tools
   Graphical Administration Tools
   Headless Management
   Legacy UNIX Compatibility
   Network Servers
```

To know which packages are there in a group package, just use the **`group info`** command and give the name of the package. For "Security Tools" package, type

```
1   yum group info "Security Tools"
```

> **Note:** You **need to enclose** the Group Package Name which has multiple words in quotes(" ").
> Even if the Group package name is a single word, it is recommended that you use quotes.

that you use quotes.

```
root@li1986-214:~
[16:36:47] # yum group info "Security Tools"
Last metadata expiration check: 2:03:03 ago on Thu 26 Nov 2020 02:38:02 PM UTC.

Group: Security Tools
 Description: Security tools for integrity and trust verification.
 Default Packages:
   scap-security-guide
 Optional Packages:
   aide
   hmaccalc
   openscap
   openscap-engine-sce
   openscap-utils
   scap-security-guide-doc
   scap-workbench
   tpm-quote-tools
   tpm-tools
   tpm2-tools
   trousers
   udica
```

Let's install the Security Tools Group package using the **`group install`** command.

```
1   yum group install "Security Tools"
```

```
root@li1986-214:~
[16:55:08] # yum group install "Security Tools"
Last metadata expiration check: 2:18:47 ago on Thu 26 Nov 2020 02:38:02 PM UTC.
Dependencies resolved.
================================================================================
 Package                  Architecture   Version               Repository      Size
================================================================================
Installing group/module packages:
 scap-security-guide      noarch         0.1.48-7.el8          AppStream      6.9 M
Installing dependencies:
 GConf2                   x86_64         3.2.6-22.el8          AppStream      1.0 M
 libxslt                  x86_64         1.1.32-4.el8          BaseOS         249 k
 openscap                 x86_64         1.3.2-6.el8           AppStream      3.3 M
 openscap-scanner         x86_64         1.3.2-6.el8           AppStream       70 k
 xml-common               noarch         0.6.3-50.el8          BaseOS          39 k
Installing Groups:
 Security Tools

Transaction Summary
================================================================================
Install  6 Packages
```

7. List available or enabled repositories

To list all the available repositories, type

```
1   yum repolist all
```

```
root@li1986-214:~
[16:53:17] # yum repolist all
repo id                         repo name                                          status
AppStream                       CentOS-8 - AppStream                               enabled
AppStream-source                CentOS-8 - AppStream Sources                       disabled
BaseOS                          CentOS-8 - Base                                    enabled
BaseOS-source                   CentOS-8 - BaseOS Sources                          disabled
Devel                           CentOS-8 - Devel WARNING! FOR BUILDROOT USE ONLY!  disabled
HighAvailability                CentOS-8 - HA                                      disabled
PowerTools                      CentOS-8 - PowerTools                              disabled
base-debuginfo                  CentOS-8 - Debuginfo                               disabled
c8-media-AppStream              CentOS-AppStream-8 - Media                         disabled
c8-media-BaseOS                 CentOS-BaseOS-8 - Media                            disabled
centosplus                      CentOS-8 - Plus                                    disabled
centosplus-source               CentOS-8 - Plus Sources                            disabled
cr                              CentOS-8 - cr                                      disabled
epel                            Extra Packages for Enterprise Linux 8 - x86_64     enabled
epel-debuginfo                  Extra Packages for Enterprise Linux 8 - x86_64 - De disabled
epel-modular                    Extra Packages for Enterprise Linux Modular 8 - x86 enabled
epel-modular-debuginfo          Extra Packages for Enterprise Linux Modular 8 - x86 disabled
epel-modular-source             Extra Packages for Enterprise Linux Modular 8 - x86 disabled
epel-playground                 Extra Packages for Enterprise Linux 8 - Playground disabled
```

To list all the enabled repositories, type

```
1   yum repolist enabled
```

```
root@li1986-214:~
[16:53:33] # yum repolist enabled
repo id                 repo name
AppStream               CentOS-8 - AppStream
BaseOS                  CentOS-8 - Base
epel                    Extra Packages for Enterprise Linux 8 - x86_64
epel-modular            Extra Packages for Enterprise Linux Modular 8 - x86_64
extras                  CentOS-8 - Extras
```

8. List dependencies of a package

To list the dependencies of a package, use the **deplist** command.

```
yum deplist syncthing
```

```
root@li1986-214:~
[17:07:52] # yum deplist syncthing
Last metadata expiration check: 2:29:52 ago on Thu 26 Nov 2020 02:38:02 PM UTC.
package: syncthing-1.8.0-2.el8.1.x86_64
  dependency: /bin/sh
   provider: bash-4.4.19-10.el8.x86_64
  dependency: libc.so.6(GLIBC_2.14)(64bit)
   provider: glibc-2.28-101.el8.x86_64
  dependency: libdl.so.2()(64bit)
   provider: glibc-2.28-101.el8.x86_64
  dependency: libdl.so.2(GLIBC_2.2.5)(64bit)
   provider: glibc-2.28-101.el8.x86_64
  dependency: libpthread.so.0()(64bit)
   provider: glibc-2.28-101.el8.x86_64
  dependency: libpthread.so.0(GLIBC_2.2.5)(64bit)
   provider: glibc-2.28-101.el8.x86_64
  dependency: libpthread.so.0(GLIBC_2.3.2)(64bit)
   provider: glibc-2.28-101.el8.x86_64
  dependency: rtld(GNU_HASH)
   provider: glibc-2.28-101.el8.i686
   provider: glibc-2.28-101.el8.x86_64
  dependency: systemd
   provider: systemd-239-31.el8_2.2.i686
   provider: systemd-239-31.el8_2.2.x86_64
```

9. View history of installation/removal of packages

Sometimes, viewing your YUM history is a good idea especially if you want to repeat the installations on a different system. History can be viewed using the **history** command of YUM.

```
root@li1986-214:~
[19:09:37] # yum history
ID     | Command line             | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------
    25 | remove syncthing         | 2020-11-26 16:53 | Removed        |    1
    24 | install syncthing        | 2020-11-26 16:51 | Install        |    1
    23 | remove tigervnc-server   | 2020-11-26 13:50 | Removed        |   62
    22 | install tigervnc-server  | 2020-11-26 13:48 | Install        |   62
    21 | remove tigervnc-server   | 2020-09-26 15:30 | Removed        |   62
    20 | groupremove Server with  | 2020-09-26 15:27 | ?, E, I        |  881 EE
    19 |                          | 2020-09-26 14:46 | Install        |    1
    18 | install tigervnc-server  | 2020-09-26 13:55 | Install        |    1
    17 | groupinstall Server with | 2020-09-26 13:27 | I, U           |  951 EE
    16 | install htop             | 2020-09-24 13:10 | Install        |    1
    15 | install screen           | 2020-09-22 13:08 | Install        |    1
    14 | install epel-release     | 2020-09-22 13:04 | Install        |    1
    13 | reinstall rsyslog        | 2020-09-10 13:52 | R              |    2
    12 | install strace           | 2020-09-10 13:45 | Install        |    1
    11 | install rsyslog          | 2020-09-10 12:25 | Install        |    4
    10 | remove rsyslog           | 2020-09-10 12:18 | Removed        |    4 EE
```

- Yum, command helps all Users and System Administrators to search for information about packages and then install, update, and remove all rpm related files from systems.
- The advantage of using YUM will be automatically performing the dependency resolution in a single command, even its supports to install the packages from various 3rd party repositories without any issue of dependencies
- Yum package installer saves a good amount of internet bandwidth while we

download packages.

Sub-Command	Description
install	Install the specific software packages
update	Packages will be updated to the latest version, also we can use few arguments along with update command ('--exclude', '--security')
remove	Remove a particular package
list	List information about available packages
info	List description & summary information of available packages
clean	Clean up the accumulated things from the yum cache directory
groups	Install a group of software packages
search	It will help to find the packages
localinstall	Install a set of local .rpm extensions packages
upgrade	The upgrade and update will perform the same function, both arguments will update to the latest current version of the package, but the only difference is, the upgrade will delete the obsolete packages.
downgrade	Downgrade the package from higher to lower version
repolist	A list of configured repositories
history	It lists all the latest yum operations and also 'history' command can be used with "list, info, summary, repeat, redo, undo, new" of certain transactions too.
help	It will list the help for all commands or a given command help

1. How do I check if there are any packages available for update?

```
$ sudo yum check-update
```

```
Output:

Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.mirrors.estointernet.in
 * extras: centos.mirrors.estointernet.in
 * updates: centos.mirrors.estointernet.in
base                                                     | 3.6 kB     00:00
extras                                                   | 2.9 kB     00:00
updates                                                  | 2.9 kB     00:00

augeas-libs.x86_64                      1.4.0-9.el7_8.1             updates
bind-export-libs.x86_64                 32:9.11.4-16.P2.el7_8.3     updates
bind-libs.x86_64                        32:9.11.4-16.P2.el7_8.3     updates
bind-libs-lite.x86_64                   32:9.11.4-16.P2.el7_8.3     updates
```

Note:

Using the above command we can see (if there are any available) packages from the enabled repositories for an update. It is always recommended that the installers examine and verify the list of available packages before blindly executing the 'update' sub-command to update your system, as sometimes the new updates/patches may not be compatible with your existing applications or hardware. I have not added the full list to the above output as it has almost 116 packages available to update, however a screenshot with only a few lines to give you an idea about it. Add 'wc -l' to get the total count of the available packages (like 'yum check-update | wc -l).

2. How to install a particular package?

```
$ sudo yum install vsftpd
```

```
Output:

Dependencies Resolved

================================================================================
 Package                Arch                Version                  Repository              Size
================================================================================
Installing:
 vsftpd                 x86_64              3.0.2-27.el7             base                   172 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 172 k
Installed size: 353 k
Is this ok [y/d/N]: y
```

Note:

I have used the above command to install a particular package "vsftpd", once you have executed the command it will ask you for confirmation and enter 'y' to proceed with the installation. Also, you can install the package without confirmation by using '-y' at the end of the above command.

3. How to update a particular package?

```
$ sudo yum update vsftpd -y
```

Note:

Please use the same "vsftpd" package as we know this package already exists in the system. After a couple of days you will be notified by the release of the new updates. This time I just want to update only a particular package, in our example "vsftpd" by using the above command we can update only the concern package from the whole bunch of new releases. To update the entire system, just use the following command:

```
$ sudo yum update -y
```

4. How to exclude a particular package while updating the whole system?

Note:

For a better understanding, first let me show you a list of all the available packages to update using the following command. From this list we can pick a particular package to exclude. To add the line numbers to the following list, I have used the 'cat ' command with the '-n' option. If you want to know more about cat commands in Linux click here

```
$ sudo yum list updates | cat -n
```

```
Output:

 1  Loaded plugins: fastestmirror, langpacks
 2  Loading mirror speeds from cached hostfile
 3   * base: centos.mirrors.estointernet.in
 4   * extras: centos.mirrors.estointernet.in
 5   * updates: centos.mirrors.estointernet.in
 6  Updated Packages
 7  augeas-libs.x86_64                      1.4.0-9.el7_8.1             updates
 8  bind-export-libs.x86_64                 32:9.11.4-16.P2.el7_8.3     updates
 9  bind-libs.x86_64                        32:9.11.4-16.P2.el7_8.3     updates
10  bind-libs-lite.x86_64                   32:9.11.4-16.P2.el7_8.3     updates
11  bind-license.noarch                     32:9.11.4-16.P2.el7_8.3     updates
12  bind-utils.x86_64                       32:9.11.4-16.P2.el7_8.3     updates
13  binutils.x86_64                         2.27-43.base.el7_8.1        updates
14  bpftool.x86_64                          3.10.0-1127.8.2.el7         updates
15  device-mapper.x86_64                    7:1.02.164-7.el7_8.2        updates
16  device-mapper-event.x86_64              7:1.02.164-7.el7_8.2        updates
17  device-mapper-event-libs.x86_64         7:1.02.164-7.el7_8.2        updates
18  device-mapper-libs.x86_64               7:1.02.164-7.el7_8.2        updates
19  firefox.x86_64
```

Note:

In this example I have selected the "device-mapper" package which is the 15th number on the above list to exclude while updating the whole system. This can be done by using one of the following commands.

```
$ sudo yum update --exclude=device-mapper\*        OR        $ sudo yum -x "package_name*" update
```

Note:

To exclude more than one (multi) package in a single command, use one of the following commands.

```
$ sudo yum update --exclude=device-mapper\* --exclude=PackageKit\*
   OR
$ sudo yum -x "package_name1*" -x "package_name2*" update
```

Note:

Regular updates of security fixes can prevent vulnerabilities. Use the following command to install/update only the security fixes. For the below output you can see that we have almost 519 packages that are available for updates in the system, out of that it will fetch only "1 patch" belongs the security fix.

```
$ sudo yum update --security
```

```
Output:

 --> teamd-1.29-1.el7.x86_64 from base removed (updateinfo)
1 package(s) needed (+0 related) for security, out of 519 available
Resolving Dependencies
--> Running transaction check
---> Package openvpn.x86_64 0:2.4.8-1.el7 will be updated
---> Package openvpn.x86_64 0:2.4.9-1.el7 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package              Arch           Version           Repository          Size
================================================================================
Updating:
 openvpn              x86_64         2.4.9-1.el7       epel               524 k

Transaction Summary
================================================================================
Upgrade  1 Package

Total download size: 524 k
Is this ok [y/d/N]:
```

5. How to remove a particular package?

```
$ sudo yum remove vsftpd
```

```
Output:

Dependencies Resolved

================================================================================
 Package                Arch                Version                Repository              Size
================================================================================
Removing:
 vsftpd                 x86_64              3.0.2-27.el7           @base                  353 k

Transaction Summary
================================================================================
Remove  1 Package

Installed size: 353 k
Is this ok [y/N]: y
```

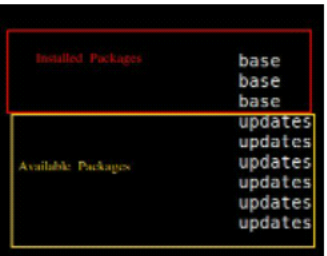
Note:

Using the above command we can completely erase/delete a particular package from our system. Always be careful before removing any packages and do not give auto-confirmation using '-y' with remove sub-command, as sometimes it may remove the dependencies also and it could lead to unexpected results for the packages that still require those dependencies. In case if there is any such situation, it is better you can use 'rpm -e --nodeps package_Name' to overcome the dependency issue. In the above output you can see that I have completely removed the "vsftpd" package.

6. How to get a combined list of all the packages?

```
$ sudo yum list all
```

```
Output:

yp-tools.x86_64                                2.14-5.el7                    base
ypbind.x86_64                                  3:1.37.1-9.el7                base
ypserv.x86_64                                  2.31-12.el7                   base
yum-NetworkManager-dispatcher.noarch           1.1.31-54.el7_8               updates
yum-plugin-aliases.noarch                      1.1.31-54.el7_8               updates
yum-plugin-auto-update-debug-info.noarch       1.1.31-54.el7_8               updates
yum-plugin-changelog.noarch                    1.1.31-54.el7_8               updates
yum-plugin-copr.noarch                         1.1.31-54.el7_8               updates
yum-plugin-fastestmirror.noarch                1.1.31-54.el7_8               updates
yum-plugin-filter-data.noarch
```



Note:

With the above command we can get the combined list of information of all the installed and available packages. You can see the differences in the above output.

```
$ sudo yum list installed
```

```
Output:

yum-metadata-parser.x86_64                     1.1.4-10.el7                  @anaconda
yum-plugin-fastestmirror.noarch                1.1.31-53.el7                 @anaconda
yum-utils.noarch                               1.1.31-53.el7                 @anaconda
zenity.x86_64                                  3.28.1-1.el7                  @anaconda
zip.x86_64                                     3.0-11.el7                    @anaconda
zlib.x86_64
```

Note:

Here you can get the list of all the information on the packages that are currently installed on the system.

```
$ sudo yum list installed "http?-*"
```

```
Output:

Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.mirrors.estointernet.in
 * extras: centos.mirrors.estointernet.in
 * updates: centos.mirrors.estointernet.in
Installed Packages
httpd.x86_64                                   2.4.6-93.el7.centos           @base
httpd-devel.x86_64                             2.4.6-93.el7.centos           @base
httpd-manual.noarch                            2.4.6-93.el7.centos           @base
httpd-tools.x86_64                             2.4.6-93.el7.centos           @base
```

Note:

Using the above command we can list all the installed packages that start with "http" It is called double-quoted glob expression, it can be used to list all the versions of some specific components as these are distinguished by numbers.

7. How to find out detailed information about a specific package?

```
$ sudo yum info vsftpd
```

```
Output:

Available Packages
Name        : vsftpd
Arch        : x86_64
Version     : 3.0.2
Release     : 27.el7
Size        : 172 k
Repo        : base/7/x86_64
Summary     : Very Secure Ftp Daemon
URL         : https://security.appspot.com/vsftpd.html
License     : GPLv2 with exceptions
Description : vsftpd is a Very Secure FTP daemon. It was written completely from
            : scratch.
```

Using the above command we can get detailed information about a particular package like version, release date, size, summary. In the above output, you can see the detailed information about the 'vsftpd' package

8. How to clear the cache information from the cache directory?

```
$ sudo yum clean all
```

```
Output:

Loaded plugins: fastestmirror, langpacks
Cleaning repos: base extras updates
Cleaning up list of fastest mirrors
```

Note:

You can use the above command to clear the cached information from our system. By default yum will store all package information in the "/var/cache/yum" directory and it uses the system disk space until we clean up the accumulated entries in the cache directory. Sometimes non-clear cached data may cause issues when you attempt to install/update the package or there is a possibility of showing the wrong version of the package even after the repository has updated packages. Hence, it is recommended once in a while to clean up the accumulated entries in the yum cache directory.

9. How to install a group of packages?

Note:

Before use, the 'groupinstall' sub-command, let us check the summary of the groups, like how many installed groups, available groups, and available environment groups.

```
$ sudo yum groups summary
```

```
Output:

Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Determining fastest mirrors
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
base                                                     | 3.6 kB  00:00:00
extras                                                   | 2.9 kB  00:00:00
updates                                                  | 2.9 kB  00:00:00
(1/4): extras/7/x86_64/primary_db                        | 194 kB  00:00:01
(2/4): base/7/x86_64/group_gz                            | 153 kB  00:00:01
(3/4): updates/7/x86_64/primary_db                       | 1.3 MB  00:00:02
(4/4): base/7/x86_64/primary_db                          | 6.1 MB  00:00:15
Available Environment Groups: 10
Available Groups: 10
Done
```

Note:

After finding the summary of group information, let us check the list of available group packages using the following command.

```
$ sudo yum grouplist
```

```
Output:

Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
Available Environment Groups:
   Minimal Install
   Compute Node
   Infrastructure Server
   File and Print Server
   Basic Web Server
   Virtualization Host
   Server with GUI
   GNOME Desktop
   KDE Plasma Workspaces
   Development and Creative Workstation
Available Groups:
   Compatibility Libraries
   Console Internet Tools
```

Note:

The above command will list all the available group packages. For the above output, I have attached only a few group names from the list. Now we can use the 'groupinfo' sub-command to get a better idea about the packages that are part of the group.

```
$ sudo yum groupinfo 'Basic Web Server'
```

```
Output:

Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net

Environment Group: Basic Web Server
 Environment-Id: web-server-environment
 Description: Server for serving static and dynamic internet content.
 Mandatory Groups:
   +base
   +core
   +web-server
 Optional Groups:
   +backup-client
   +debugging
```

Note:

As you can see in the above output, the "Basic Web Server" has been bundled with three mandatory groups like (base, core, web-server) and other optional groups. If you want to know each of these group package details, then use the 'groupinfo' argument with these group names. In our demo, we don't need the bundled "Basic Web Server" package. Here we are going to install only the webserver, so let's check the webserver info and install it on our server.

```
$ sudo yum groupinfo 'web-server'
```

```
Output:

Loaded plugins: fastestmirror, langpacks
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net

Group: Web Server
 Group-Id: web-server
 Description: Allows the system to act as a web server, and run Perl and Python web applications.
 Mandatory Packages:
    httpd
 Default Packages:
   +crypto-utils
    httpd-manual
   +mod_fcgid
   +mod_ssl
 Optional Packages:
```

```
$ sudo yum groupinstall 'web-server'
```

```
Output:

Dependencies Resolved

================================================================================
 Package                  Arch               Version                   Repository          Size
================================================================================
Installing for group install "Web Server":
 crypto-utils             x86_64             2.4.1-42.el7              base                78 k
 mod_fcgid                x86_64             2.3.9-6.el7               base                79 k
 mod_ssl                  x86_64             1:2.4.6-93.el7.centos     base               113 k
Installing for dependencies:
 perl-Newt                x86_64             1.08-36.el7               base                64 k

Transaction Summary
================================================================================
Install  3 Packages (+1 Dependent package)
```

Note:

Finally, we have installed the group package using the 'groupinstall' sub-command. In the above output you can see the name of the installed packages.

10. How to find the package names?

```
$ sudo yum search samba
```

```
Output:

Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
============================================= N/S matched: samba ==============================
kdenetwork-fileshare-samba.x86_64 : Share files via samba
pcp-pmda-samba.x86_64 : Performance Co-Pilot (PCP) metrics for Samba
samba-client.x86_64 : Samba client programs
samba-client-libs.i686 : Samba client libraries
samba-client-libs.x86_64 : Samba client libraries
samba-common.noarch : Files used by both Samba servers and clients
```

Note:

This 'search' argument is very useful, for ex: if you don't remember or don't know the package name, then you can use the above command to search all the available packages to match with the name you specified.

11. How to install a .rpm extension package using yum?

```
$ sudo yum localinstall epel-release-latest-7.noarch.rpm
```

```
Output:

Loaded plugins: fastestmirror, langpacks
Examining epel-release-latest-7.noarch.rpm: epel-release-7-12.noarch
Marking epel-release-latest-7.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package epel-release.noarch 0:7-12 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch             Version          Repository                                Size
================================================================================
Installing:
 epel-release           noarch           7-12             /epel-release-latest-7.noarch             24 k

Transaction Summary
================================================================================
Install  1 Package
```

Note:

Sometimes certain packages won't be available in our yum repositories, so we may need to download the rpm related package and install it in a traditional method of using rpm installation like "rpm -ivh xyz-1.2.3.rpm" but this package will demand dependencies and those dependencies are available in yum repository, so naturally, first we do install the dependencies using yum command then install the original rpm files. To overcome the dual attempt we can use 'localinstall' sub-command to install both dependencies and the rpm package in a single yum command. You can see the details in the above output.

12. How to upgrade the system with the latest releases?

```
$ sudo yum upgrade -y
```

Note:

The beauty is we can use both 'update and upgrade' sub-command to achieve the latest version of packages in our system, but there is a small difference between them. The 'upgrade' sub-command will remove all the obsolete packages during the update of the system whereas 'update' doesn't. You can use the above command only in the same family like if you want to upgrade your version from 7.4 to 7.8, but to upgrade 7.4 to 8.1 then this command is not recommended. It is better to upgrade to a different family to make a fresh installation of the 8.1 OS.

13. How to downgrade a package?

Note:

For a better understanding first I will update the "http-parser" package from version (2.7.1-8.el7 to 2.7.1-8.el7_7.2), then will downgrade the sub-command to revert to the original version. Sometimes, there will be compatibility issues with existing applications or hardware after the new updates of few packages, in that case we may force to revert/downgrade those packages into the previous version.

```
$ sudo yum list installed | grep http-parser
```

```
Output:

http-parser.x86_64                      2.7.1-8.el7               @anaconda
```

```
$ sudo yum update http-parser
```

```
Output:

Dependencies Resolved

================================================================================
 Package              Arch              Version                  Repository   Size
================================================================================
Updating:
 http-parser          x86_64            2.7.1-8.el7_7.2          updates      29 k

Transaction Summary
================================================================================
Upgrade  1 Package
```

```
$ sudo yum downgrade http-parser-2.7.1-8.el7
```

```
Output:

Dependencies Resolved

================================================================================
 Package              Arch              Version                  Repository   Size
================================================================================
Downgrading:
 http-parser          x86_64            2.7.1-8.el7              base         29 k

Transaction Summary
================================================================================
Downgrade  1 Package
```

14. How to view the repository details/information?

```
$ sudo yum repolist
```

```
Output:

Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
repo id                                repo name                            status
base/7/x86_64                          CentOS-7 - Base                      10,070
extras/7/x86_64                        CentOS-7 - Extras                       397
updates/7/x86_64                       CentOS-7 - Updates                      671
repolist: 11,138
```

(a) How to get more details about a particular transaction from the yum history?

```
$ sudo yum history info 10
```

(a) How to get more details about a particular transaction from the yum history?

```
$ sudo yum history info 10
```

```
Output:

Transaction ID : 10
Begin time     : Sun May 24 04:01:01 2020
Begin rpmdb    : 1446:c2ba4ab11c66c9a1b18db9918197745a4e7aeff3
End time       :            04:01:08 2020 (7 seconds)
End rpmdb      : 1447:a0b6a955e660c852e2af4b2140250f4362b9b5a4
User           : LinuxTeck <linuxteck>
Return-Code    : Success
Command Line   : install bind
Transaction performed with:
    Installed     rpm-4.11.3-43.el7.x86_64                     @anaconda
    Installed     yum-3.4.3-167.el7.centos.noarch              @anaconda
    Installed     yum-plugin-fastestmirror-1.1.31-53.el7.noarch @anaconda
Packages Altered:
    Install bind-32:9.11.4-16.P2.el7_8.3.x86_64           @updates
    Updated bind-libs-32:9.11.4-16.P2.el7.x86_64          @anaconda
    Update            32:9.11.4-16.P2.el7_8.3.x86_64      @updates
    Updated bind-libs-lite-32:9.11.4-16.P2.el7.x86_64     @anaconda
    Update                32:9.11.4-16.P2.el7_8.3.x86_64 @updates
    Updated bind-license-32:9.11.4-16.P2.el7.noarch       @anaconda
    Update              32:9.11.4-16.P2.el7_8.3.noarch    @updates
    Updated bind-utils-32:9.11.4-16.P2.el7.x86_64         @anaconda
    Update            32:9.11.4-16.P2.el7_8.3.x86_64      @updates
history info
```

(b) How to undo a transaction?

```
$ sudo yum history undo 10
```

```
Output:

Dependencies Resolved

================================================================================
 Package                Arch             Version                    Repository          Size
================================================================================
Removing:
 bind                   x86_64           32:9.11.4-16.P2.el7_8.3    @updates           5.4 M
Downgrading:
 bind-libs              x86_64           32:9.11.4-16.P2.el7        base               155 k
 bind-libs-lite         x86_64           32:9.11.4-16.P2.el7        base               1.1 M
 bind-license           noarch           32:9.11.4-16.P2.el7        base                89 k
 bind-utils             x86_64           32:9.11.4-16.P2.el7        base               258 k

Transaction Summary
================================================================================
Remove     1 Package
Downgrade  4 Packages
```

Note:

The same transaction ID 10 will also be used in this example. Using the above command we can reverse back the ID 10 to the original state, which means it will remove the "bind" package from the system. Check the above output for better understanding. You can also use the following command to cross-check whether it is removed or not.

```
$ sudo yum list installed | grep bind
```

(c) How to redo a transaction?

```
$ sudo yum history redo 10
```

```
Output:

Dependencies Resolved

================================================================================
 Package                Arch            Version                      Repository          Size
================================================================================
Installing:
 bind                   x86_64          32:9.11.4-16.P2.el7_8.3      updates            2.3 M
Updating:
 bind-libs              x86_64          32:9.11.4-16.P2.el7_8.3      updates            155 k
 bind-libs-lite         x86_64          32:9.11.4-16.P2.el7_8.3      updates            1.1 M
 bind-license           noarch          32:9.11.4-16.P2.el7_8.3      updates             89 k
 bind-utils             x86_64          32:9.11.4-16.P2.el7_8.3      updates            259 k

Transaction Summary
================================================================================
Install  1 Package
Upgrade  4 Packages
```

Note:

The same transaction ID 10 can also be used for this example. Using the above command we can redo/repeat back to the latest state, which means the "bind" package is installed back.

(d) How to rollback transactions?

```
$ sudo yum history rollback 10
```

```
Output:
Dependencies Resolved

================================================================================
 Package                Arch            Version                      Repository          Size
================================================================================
Removing:
 dhcp                   x86_64          12:4.2.5-79.el7.centos       @base              1.4 M
 libecap                x86_64          1.0.0-1.el7                  @base               50 k
 libntlm                x86_64          1.3-6.el7                    @base               85 k
 libsexy                x86_64          0.1.11-23.el7                @base              105 k
 perl-Compress-Raw-Bzip2 x86_64         2.061-3.el7                  @base               57 k
 perl-Compress-Raw-Zlib x86_64          1:2.061-4.el7                @base              137 k
 squid                  x86_64          7:3.5.20-15.el7_8.1          @updates            10 M
 squid-migration-script x86_64          7:3.5.20-15.el7_8.1          @updates            11 k
 xchat                  x86_64          1:2.8.8-25.el7               @base              3.7 M
Reinstalling:
 bind                   x86_64          32:9.11.4-16.P2.el7_8.3      updates            2.3 M
 bind-libs              x86_64          32:9.11.4-16.P2.el7_8.3      updates            155 k
 bind-libs-lite         x86_64          32:9.11.4-16.P2.el7_8.3      updates            1.1 M
 bind-license           noarch          32:9.11.4-16.P2.el7_8.3      updates             89 k
 bind-utils             x86_64          32:9.11.4-16.P2.el7_8.3      updates            259 k

Transaction Summary
================================================================================
Remove      9 Packages
Reinstall   5 Packages
```

Note:

The rollback is also similar to "undo" but the only difference is it will revert everything between that transaction and the latest state of the system. In this example we will rollback the transaction ID 10, which means all the transactions that appeared after the transaction ID 10 will be removed. In the output you can see that I installed all the packages after transaction ID (dhcp,squid,xchat and the dependencies) were removed and bind was reinstalled.

Docker rm command

Thursday, April 21, 2022 5:02 PM

- rm stands for remove here. rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX. To be more precise, rm removes references to objects from the filesystem, where those objects might have had multiple references (for example, a file with two different names). By default, it does not remove directories.

Let us consider 5 files having name **a.txt, b.txt** and so on till **e.txt**.

```
$ ls
a.txt  b.txt  c.txt  d.txt  e.txt
```

```
Removing one file at a time
$ rm a.txt

$ ls
b.txt  c.txt  d.txt  e.txt

Removing more than one file at a time
$ rm b.txt c.txt

$ ls
d.txt  e.txt
```

Note: No output is produced by **rm**, since it typically only generates messages in the case of an error.

1. -i (Interactive Deletion): Like in cp, the -i option makes the command ask the user for confirmation before removing each file, you have to press **y** for confirm deletion, any other key leaves the file un-deleted.

```
$ rm -i d.txt
rm: remove regular empty file 'd.txt'? y

$ ls
e.txt
```

```
$ ls -l
total 0
-r--r--r--+ 1 User User 0 Jan  2 22:56 e.txt

$ rm e.txt
rm: remove write-protected regular empty file 'e.txt'? n

$ ls
```

```
$ ls -l
total 0
-r--r--r--+ 1 User User 0 Jan  2 22:56 e.txt

$ rm e.txt
rm: remove write-protected regular empty file 'e.txt'? n

$ ls
e.txt

$ rm -f e.txt

$ ls
```

Note: **-f** option of rm command will not work for write-protect directories.

3. -r (Recursive Deletion): With **-r(or -R)** option rm command performs a tree-walk and will delete all the files and sub-directories recursively of the parent directory. At each stage it deletes everything it finds. Normally, **rm** wouldn't delete the directories but when used with this option, it will delete.

Below is the tree of directories and files:

```
$ ls
A

$ cd A

$ ls
B  C

$ ls B
a.txt  b.txt

$ ls C
c.txt  d.txt
```

Now, deletion from **A** directory(as parent directory) will be done as:

```
$ rm *
rm: cannot remove 'B': Is a directory
rm: cannot remove 'C': Is a directory

$ rm -r *

$ ls
```

Every directory and file inside **A** directory is deleted.

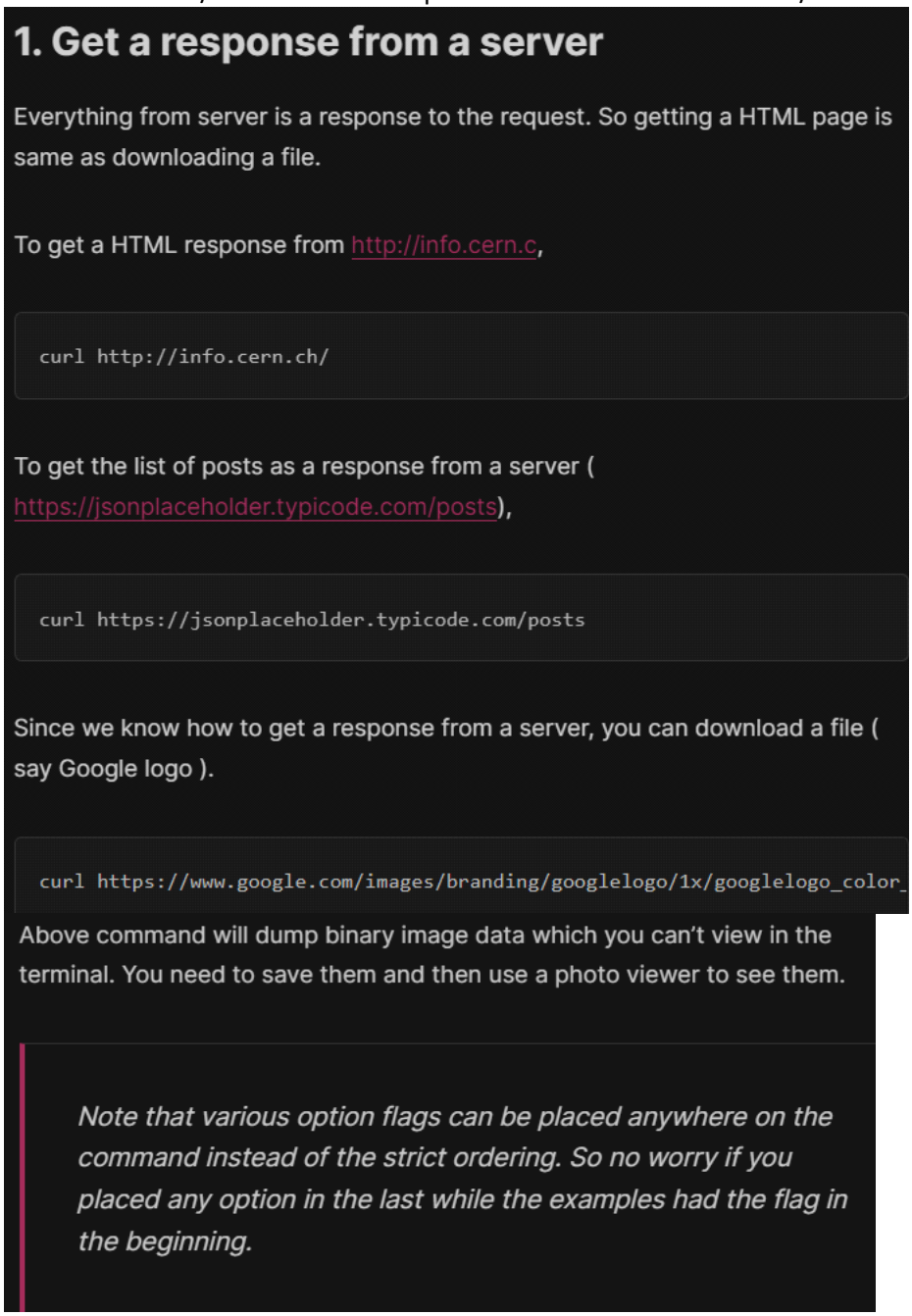
Delete file whose name starting with a hyphen symbol (-): To remove a file whose name begins with a dash ("-"), you can specify a double dash ("--") separately before the file name. This extra dash is necessary so that rm does not misinterpret the file name as an option. Let say their is a file name **-file.txt**, to delete this file write command as:

```
$ ls
-file.txt

$ rm -file.txt
rm: unknown option -- l
Try 'rm ./-file.txt' to remove the file '-file.txt'.
Try 'rm --help' for more information.

$ rm -- -file.txt

$ ls
```

Docker curl command

Thursday, April 21, 2022 5:26 PM

- CURL is a tool for data transfer. It is also available as a library for developers and as a CLI for terminal-based use cases.
- As wrap up, CURL can download HTML pages, fill HTML forms and submit them, download files from a FTP/HTTP server and upload files to the same and read/write cookies.



2. Save the file with a default file name

Every file that is served on the internet has a filename. To use the same filename as the downloaded filename use -O flag.

```
curl -O http://www.google.com/robots.txt
```

3. Save the file with custom name

To save the filename with your own custom name, use -o flag followed (strictly) by a custom name.

```
curl -O http://www.google.com/robots.txt googleRobots.txt
```

4. Download multiple files

To download multiple files, separate them with a white space.

```
curl url1 url2 url3
```

If you want to use -O flag for all the URL's, use

```
curl url1 url2 url3 -O -O -O
```

The same workaround should be done for any flag. This is because the first occurrence of a certain flag is for the first URL, the second flag is for the second URL and so on.

5. Download a range of files

curl has the in-built ability to download a range of files from the server. This can be illustrated from the following example.

```
curl http://www.google.com/logo/logo[1-9].png
```

Above command downloads files from logo1.png, logo2.png, logo3.png and up to logo9.png.

6. Download a file only if latest

To download a file only if the file's modification time is latest than the given time.

```
curl url -z "DD MMM YY MM:HH:SS"
```

7. Resume Downloading

If you have already partially transferred a file, you can resume the transfer by using the -C flag. Offset from which transfer needs to be continued should be passed as a parameter to the -C flag.

```
curl -C 1024 http://seeni.linuxhandbook.org/files/largeFile.mpv -O
```

8. Upload a file

To upload a file to the server, one needs to use -T flag followed by the file path on your local system.

```
curl -T uploadFile.txt http://upload.linuxhandbook.org/files
```

9. Delete a file

To delete a file named deleteFile.txt in a server, one can use -X flag which is intended for any HTTP verb/method(like GET, POST, PUT, DELETE, PATCH). Most of the FTP servers will have configured DELETE method if not all advanced HTTP methods.

```
curl -X DELETE http://upload.linuxhandbook.org/files/deleteFile.txt
```

You can also modify the above command for any HTTP method to do the corresponding task. For Example, if your server allows TRUNCATE method (this is made-up HTTP method, not a standard one) which removes only the content in the file and not the file, one can use the command similar to the below one.

```
curl -X  TRUNCATE http://upload.linuxhandbook.org/files/mysql.dump
```

10. Avoid redirects

When you request http://www.google.com , you will be served only the regional page such as www.google.co.in. This is done with the help of redirects (HTTP packets with status codes in the range 300-399).

You can avoid redirects with the option L.

```
curl -L  http://www.google.com
```

11. Authentication

When the server is configured to serve for only certain individuals with credentials, they will be provided with username and password. One can make login with the help of -u flag.

```
curl -u username:password http://seeni.linuxhandbook.org/files/tasks.txt
```

12. Limit data transfer

If you want to impose a data transfer limit use –limit-rate flag. Following command tries to download the file with rate limit as 10K.

```
curl --limit-rate 10K http://seeni.linuxhandbook.org/files/logoDetails.tgz
```

13. Show/Hide transfer Status

If the response is redirected from the terminal such as downloading, uploading then curl automatically shows the status/progress meter for the transfer.

If you do not want to see the progress meter, just append the command with -s flag. Progress will not be shown for response directed for the terminal.

14. Ignore SSL certificates

Do you remember the situations in which you need to give security certificate exception to visit some websites? If you trust the sources and you want to do a data transfer, you can ignore SSL certificate validation by using -k flag.

```
curl -k https://notSoSecure.org/files/logoDetails.tgz
```

15. Get Header Information also

To display the header information along with transferred data, use the -i flag.

```
curl -i http://www.google.com/robots.txt
```

16. Get Header information Only

If you want only the headers and not the data, use the -I flag

```
curl -I http://www.google.com/robots.txt
```

17. Change User Agent

Some websites and servers don't allow certain kinds of devices to access their systems. But how do they know that we are using a specific kind of device? This is due to the User-Agent HTTP header field. We can change this User Agent with -A flag.

```
curl -A "Mozilla FireFox(42.0)" http://notAllowedForCLI.sites.org/randomFil
```

18. Sending data to the Server

If the server needs some data such as token or API key, use -d flag to send the data. Data that needs to be sent should follow the flag in the command. One can use "&" to combine multiple data. This is usually done by GET and POST requests in browsers. This is one of the ways by which you can send your form information.

```
curl -d "token=34343abvfgh&name='seeni'" http://api.restful.org/getcontent
```

19. Write Cookies to a File

Cookies are some small information that allows maintaining a session with a stateless HTTP protocol. If you want to know more about Cookies, refer to this great resource.

To write cookies to a file, -c flag followed by the cookie filename should be used.

```
curl -c googleCookie.txt http://www.google.com/files
```

20. Reading Cookies from a File

To read a cookie from the file, -b flag followed by cookie filename can be used.

```
curl -b googleCookie.txt http://www.google.com/files
```

Note that -b flag only reads the cookie from the file. So if the server resends another cookie, you might need to use -c option to write them.

21. Start a new Session

If you want to initiate a new session by discarding the cookies, use -j flag. It starts a new session even if you have provided the cookie file to read with -b flag.

```
curl -b googleCookie.txt http://www.google.com/files -j
```

Docker Cache and wget command

Thursday, April 21, 2022 7:11 PM

- The Docker build process may take some time to finish. It may download base images, copy files, and download and install packages, just to mention a few common tasks. This is the reason why docker build uses a cache.

Docker Cache: https://www.baeldung.com/linux/docker-build-cache#:~:text=The%20simplest%20solution%20to%20avoid%20these%20issues%20is,that%20is%20not%20affected%20by%20the%20no-cache%20argument.

wget Command Examples in Linux: https://www.thegeekdiary.com/wget-command-examples-in-linux/#:~:text=The%20wget%20command%20is%20one%20of%20the%20most,Some%20of%20its%20popular%20features%20are%20as%20follows%3A

wget is basically a downloader and curl can be used for both download and upload.

Grep is the most useful command on Linux/Unix for system administrators, developers and DevOps engineers. The main purpose of this tool is to check for string or patterns in a specified file, or filtering other commands output.

Grep command: https://hands-on.cloud/how-to-use-grep-in-linux/#:~:text=Grep%20is%20the%20most%20useful%20command%20on%20Linux%2FUnix,mentioned%20above%2C%20is%20known%20as%20a%20regular%20expression.

Docker Commands Cheat Sheet

Thursday, April 21, 2022 7:59 PM

https://buddy.works/tutorials/docker-commands-cheat-sheet#volume-mount-using-bind-mount

Useful man pages links

Saturday, April 23, 2022 6:28 PM

Curl: https://www.man7.org/linux/man-pages/man1/curl.1.html
Tar : https://linux.die.net/man/1/tar
Group add: https://linux.die.net/man/8/groupadd
User: https://linux.die.net/man/8/useradd

Doubts

Saturday, April 23, 2022 10:25 PM

In dockerfile we are creating files in the container, how could we see it, and how could we name it , is there any pattern for that like app folder and usr folder. I am getting how commands work ,but what i cant understand is , ***how and what files we need to move from local to the container. And where to move it.***

**RUN apk add --no-cache wget **
**&& wget https://aka.ms/downloadazcopy-v10-linux -O /tmp/azcopy.tgz **
**&& export BIN_LOCATION=$(tar -tzf /tmp/azcopy.tgz | grep "/azcopy") **
&& tar -xzf /tmp/azcopy.tgz $BIN_LOCATION --strip-components=1 -C /usr/bin
Here where are the destination locations are being present, are they cleared while going to the new build or they are cleared as a layer and stored in azcopy build copy

curl -sSf -H "${ART_API}" -O "${ART_URL}/cip-generic-prod-apollo-local/ ${APOLLO_VERSION}/apollo.tar"
Here how apollo.tar is saved in current directory, I am getting an tar file , how to get a un tar file using the Dockerfile.

**curl -sSf -H "X-JFrog-Art-Api:AKCp5dKiDXqVDeRRBVaof6dBoA72njDrLq4aiBm2V9vtp3pgc5r8LuPCTDDAU7o88yDM67ur9" -O "https://artifactory.adlm.nielseniq.com:443/artifactory/cip-ansible/infra/llvm-toolset-9.0-1.el7.x86_64.rpm" && **
**curl -sSf -H "X-JFrog-Art-Api:AKCp5dKiDXqVDeRRBVaof6dBoA72njDrLq4aiBm2V9vtp3pgc5r8LuPCTDDAU7o88yDM67ur9" -O "https://artifactory.adlm.nielseniq.com:443/artifactory/cip-ansible/infra/llvm-toolset-9.0-runtime-9.0-1.el7.x86_64.rpm" && **
**curl -sSf -H "X-JFrog-Art-Api:AKCp5dKiDXqVDeRRBVaof6dBoA72njDrLq4aiBm2V9vtp3pgc5r8LuPCTDDAU7o88yDM67ur9" -O "https://artifactory.adlm.nielseniq.com:443/artifactory/cip-ansible/infra/llvm-toolset-9.0-build-9.0-1.el7.x86_64.rpm" && **
**curl -sSf -H "X-JFrog-Art-Api:AKCp5dKiDXqVDeRRBVaof6dBoA72njDrLq4aiBm2V9vtp3pgc5r8LuPCTDDAU7o88yDM67ur9" -O "https://artifactory.adlm.nielseniq.com:443/artifactory/cip-ansible/infra/llvm-toolset-9.0-conventional-9.0-1.el7.x86_64.rpm" && **
**yum -y localinstall llvm-toolset-9.0* && **
**ln -s /usr/lib64/libtinfo.so.5 /usr/lib64/libtinfo.so && **
**ln -s /usr/lib/oracle/19.11/client64/lib/libocci.so.19.1 /usr/lib/oracle/19.11/client64/lib/libocci.so && **
**mkdir /app/src && mkdir /app/bin && mkdir /app/common_libs && chmod 777 -R /app/* && **
**rm -rf /app/llvm-toolset-9.0* && **
**yum clean all && **
rm -rf /var/cache/yum
where all these are installing on the local system or the server. And we didn't create the directories like lib64 but how they are accessable

COPY ./apollo.conf /app/conf/
where the app folder is located, if it is in the container can we read it.

source scl_source enable llvm-toolset-9.0
What is this line

yum -y localinstall llvm-toolset-9.0* && \

What does this line do? What local install do?

**cp /opt/rh/llvm-toolset-9.0/root/usr/lib64/libLLVM-9.so /app/lib/ && **
We didn't create any opt directory upt

Ask doubts on folder creations

What is the pattern to declare requirements.txt
What is the convention of folders to be created in containers like app or bin or usr

export TCU_BATCH_CONNECT=cpssys/CPSSYS@DPCNAZ1
export TNS_ADMIN=/CPS/data/vignesh/tns
What are these lines

export APOLLO_DIR="/CPS/data/vignesh/CPS/apollo/"
How we untarred this apollo.tar

Docker Volumes

Monday, August 1, 2022 6:55 PM

```
PS C:\Users\thoma> docker run --rm -v ${PWD}:/myvol ubuntu /bin/bash -c "ls -lha > /myvol/myfile.txt"
PS C:\Users\thoma> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              P
PS C:\Users\thoma> .\myfile.txt
```

```
PS C:\Users\thoma> docker run --rm -v ${PWD}:/files klutchell/rar a /files/myrar.rar /files/myfile.txt

RAR 5.40   Copyright (c) 1993-2016 Alexander Roshal   15 Aug 2016
Trial version             Type RAR -? for help

Evaluation copy. Please register.

Creating archive /files/myrar.rar

Adding    /files/myfile.txt                                     OK
Done
```

```
PS C:\Users\thoma> docker run --rm -v ${PWD}:/files -w /files klutchell/rar a /files/myrar.rar /files/myfile.txt
```

Imagine "cd /files" after starting the container

```
λ ~ → mkdir tmp
λ ~ → cd tmp
λ tmp → ls
λ tmp → touch some.txt
λ tmp → docker run -it debian bash
root@bfbb79e08497:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@bfbb79e08497:/# pwd
/
root@bfbb79e08497:/# whoami
root
root@bfbb79e08497:/# exit
exit
λ tmp → cat some.txt
λ tmp → cat ./some.txt
```

```
λ tmp → docker run -it -v "$(pwd)":/src debian bash
root@52e230e54852:/# ls
bin   dev  home  lib64  mnt  proc  run   src  sys  usr
boot  etc  lib   media  opt  root  sbin  srv  tmp  var
root@52e230e54852:/# ls /src
some.txt
root@52e230e54852:/#
```

```
[root@osboxes ~]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
fc7181108d40: Pull complete
c4277fc40ec2: Pull complete
780053e98559: Pull complete
Digest: sha256:bdbf36b7f1f77ffe7bd2a32e59235dff6ecf131e3b6b5b96061c652f30685f3a
Status: Downloaded newer image for nginx:latest
[root@osboxes ~]# mkdir -p /tmp/nginx/html
[root@osboxes ~]# docker run -t -d -P -v /tmp/nginx/html:/usr/share/nginx/html --name nginxcontainer nginx:latest
b6694ba913115fdbb1812ce50e9fc6752cc88bbde769248fa91082af72b94abf
[root@osboxes ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
      NAMES
b6694ba91311        nginx:latest        "nginx -g 'daemon of…"   4 seconds ago       Up 4 seconds        0.0.0.0:32769->80
/tcp   nginxcontainer
[root@osboxes ~]#
```

```
[root@osboxes ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
      NAMES
b6694ba91311        nginx:latest        "nginx -g 'daemon of…"   28 seconds ago      Up 28 seconds       0.0.0.0:32769->80
/tcp   nginxcontainer
[root@osboxes ~]# 192.168.56.101:32769^C
[root@osboxes ~]# ls
anaconda-ks.cfg  cloudera-devops  dockerbuild   docker_compose01   docker_hadoop  index.html           test
apache2docker    devops           docker_build  docker_compose_02  docker_test    initial-setup-ks.cfg  test.html
[root@osboxes ~]# vim index.html
[root@osboxes ~]# docker run -t -d -P -v /tmp/nginx/html:/usr/share/nginx/html --name nginxcontainer nginx:latest^C
[root@osboxes ~]# ls /tmp/nginx/html/
[root@osboxes ~]# cp index.html /tmp/nginx/html/
[root@osboxes ~]# ls /tmp/nginx/html/
index.html
```

Details about the container present in:

docker inspect container_ID

```
[runtime@vm-sapg01 docker]$ ls
abc.txt
[runtime@vm-sapg01 docker]$ pwd
/home/runtime/pratik/docker
[runtime@vm-sapg01 docker]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
demo-img            latest              f7a04aabf198        3 weeks ago         148MB
[runtime@vm-sapg01 docker]$ docker run -d --volume /home/runtime/pratik/docker:/tmp f7a04aabf198 sleep in
finity
57f3e4f2957c2cb89c98998fc512eb29e0ebc29a9f6fcb99e9a58078926189db
[runtime@vm-sapg01 docker]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
              NAMES
57f3e4f2957c        f7a04aabf198        "sleep infinity"    4 seconds ago       Up 2 seconds
              pensive_rosalind
[runtime@vm-sapg01 docker]$ docker exec -it 57f3e4f2957c sh
sh-4.4$ cd /tmp/
sh-4.4$ ls
abc.txt
sh-4.4$ cat abc.txt
My name is pratik
sh-4.4$ touch xyz.txt
sh-4.4$ ls
abc.txt  xyz.txt
sh-4.4$ exit
```

```
[runtime@vm-sapg01 docker]$ ls
abc.txt  xyz.txt
[runtime@vm-sapg01 docker]$
```

Add a volume to an existing docker container & Mount Host directory into a running docker container

```
pavi@Linuxamination:~$ ls /var/www/html
index.html  info.php  nedcalc  phpldapadmin-1.2.2  phpldapadmin-1.2.2.zip
pavi@Linuxamination:~$ sudo docker ps -a
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS
            PORTS                                                                                 NAME
S
f6db791530ee        alekseychudov/centos8-systemd  "/usr/lib/systemd/sy…"   6 days ago          Exited (130) 36
 hours ago                                                                                        linu
xaminationC8
3138ee90c6ff        2eb2d388e1a2                   "/bin/bash"              2 months ago        Up 3 hours
                                                                                                 sill
y_keldysh
8d6825efc274        ubuntu:16.04                   "/bin/bash"              15 months ago       Up 11 minutes
            0.0.0.0:89->80/tcp, 0.0.0.0:3506->3306/tcp, 0.0.0.0:5442->5432/tcp, 0.0.0.0:26017->27017/tcp   inte
lligent_franklin
pavi@Linuxamination:~$ sudo docker stop 8d6825efc274
8d6825efc274
pavi@Linuxamination:~$ sudo ls /var/lib/docker/containers
3138ee90c6ff3a6f33b4738eabe6ce3de9861aef47f46b368aaebd9d16824164
8d6825efc274cb62ea45d70219d15805c8c61395b85aca4475b77b3cd59767db
b973e43f6d381044b1eeb3b3b1dbcc70cc72fd54ec76e259b82b4c97bb9dab0a
f6db791530ee8b06dc714fd0d4c1248f0284ce806bbba395f8c960fa864cd8d3
pavi@Linuxamination:~$
```


We need to modify the mapping of container with host machine in hostconfig.json [Binds] and config.v2.json [mountPoints]

```
...b/docker/containers/8d6825efc274cb62ea45d70219d15805c8c61395b85aca4475b77b3cd59767db/hostconfig.json Modified
{"Binds":["/var/www/html:/var/www/html"],"ContainerIDFile":"","LogConfig":{"Type":"json-file","Config":{}},"Net>
```

```
1 hostconfig.json
2 {"Binds":["/var/www/html:/var/www/html"],
3
4 configv2.json
5 "MountPoints":{"/var/www/html":{"Source":"/var/www/html","Destination":"/var/www/html","RW":true,"Name":"","Driver":"","-
  Type":"bind","Propagation":"rprivate","Spec":{"Type":"bind","Source":"/var/www/html","Target":"/var/www/html"},"SkipMoun-
  tpointCreation":false}},
```

```
pavi@Linuxamination:~$ sudo nano /var/lib/docker/containers/8d6825efc274cb62ea45d70219d15805c8c61395b85aca4475b7
7b3cd59767db/hostconfig.json
pavi@Linuxamination:~$ sudo nano /var/lib/docker/containers/8d6825efc274cb62ea45d70219d15805c8c61395b85aca4475b7
7b3cd59767db/config.v2.json
```



```
pavi@Linuxamination:~$ sudo service docker restart
pavi@Linuxamination:~$ sudo docker ps -a
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS
            PORTS               NAMES
f6db791530ee        alekseychudov/centos8-systemd  "/usr/lib/systemd/sy…"   6 days ago          Exited (130) 36
 hours ago                      linuxaminationC8
3138ee90c6ff        2eb2d388e1a2                   "/bin/bash"              2 months ago        Up 5 seconds
                                silly_keldysh
8d6825efc274        ubuntu:16.04                   "/bin/bash"              15 months ago       Exited (0) 3 mi
nutes ago                       intelligent_franklin
pavi@Linuxamination:~$ sudo docker start 8d6825efc274
8d6825efc274
pavi@Linuxamination:~$ sudo docker exec -it 8d6825efc274 bash
root@8d6825efc274:/# ls /var/www/html
index.html  info.php  nedcalc  phpldapadmin-1.2.2  phpldapadmin-1.2.2.zip
root@8d6825efc274:/#
```


Mapping multiple directories and creating volume from host to the container

Reason to mounting a docker volume:

- When a container completes it's execution , we lose all the data in the container.
- To make the data persistent , we use volume mount in a docker container.

AZ copy notes

Tuesday, September 13, 2022 11:48 AM

- AzCopy is a command-line tool that moves data into and out of Azure Storage.
- AzCopy is a command-line utility that you can use to copy blobs or files to or from a storage account.
- If you want to copy data to and from your Azure Table storage service, then install AzCopy version 7.3
- We must declare the path of azcopy in the path , so that we can use azcopy executable in any location without declaring path for the executable as that is already declared.
- As an owner of your Azure Storage account, you aren't automatically assigned permissions to access data. Before you can do anything meaningful with AzCopy, you need to decide how you'll provide authorization credentials to the storage service.
- You can provide authorization credentials by using Azure Active Directory (AD), or by using a Shared Access Signature (SAS) token.
- Option 1: Azure Active Directory
- This option is available for blob Storage only. By using Azure Active Directory, you can provide credentials once instead of having to append a SAS token to each command.
- Option 2: Use a SAS token
- This example command recursively copies data from a local directory to a blob container. A fictitious SAS token is appended to the end of the container URL.

```
azcopy copy "C:\local\path" "https://account.blob.core.windows.net/mycontainer1/?sv=2018-03-28&ss=bjqt&srt=sco&sp=rwddgcup&se=2019-05-01T05:01:17Z&st=2019-04-30T21:01:17Z&spr=https&sig=MGCXiyEzbtttkr3ewJIh2AR8KrghSy1DGM9ovN734bQF4%3D" --recursive=true
```

- A shared access signature (SAS) provides secure delegated access to resources in your storage account. With a SAS, you have granular control over how a client can access your data. For example:
    - What resources the client may access.
    - What permissions they have to those resources.
    - How long the SAS is valid.
- Azure Storage supports three types of shared access signatures:
    - User delegation SAS
        - A user delegation SAS is secured with Azure Active Directory (Azure AD) credentials and also by the permissions specified for the SAS. A user delegation SAS applies to Blob storage only.
    - Service SAS
        - A service SAS is secured with the storage account key. A service SAS delegates access to a resource in only one of the Azure Storage services: Blob storage, Queue storage, Table storage, or Azure Files.
    - Account SAS
        - An account SAS is secured with the storage account key. An account SAS delegates access to resources in one or more of the storage services. All of the operations available via a service or user delegation SAS are also available via an account SAS.
        - We can also perform , Service-level operations (For example, the Get/Set Service Properties and Get Service Stats operations). Read, write, and delete operations that aren't permitted with a service SAS.
- After you've authorized your identity or obtained a SAS token, you can begin transferring data.
- It's not possible to audit the generation of SAS tokens. Any user that has privileges to generate a SAS token, either by using the account key, or via an Azure role assignment, can do so without the knowledge of the owner of the storage account. Be careful to restrict permissions that allow users to generate SAS tokens. To prevent users from generating a SAS that is signed with the account key for blob and queue workloads, you can disallow Shared Key access to the storage account.

Both a service SAS and an account SAS are signed with the storage account key. To create a SAS that is signed with the account key, an application must have access to the account key.

- When a request includes a SAS token, that request is authorized based on how that SAS token is signed. The access key or credentials that you use to create a SAS token are also used by Azure Storage to grant access to a client that possesses the SAS.

Type of SAS	Type of authorization
User delegation SAS (Blob storage only)	Azure AD
Service SAS	Shared Key
Account SAS	Shared Key

- The SAS token is a string that you generate on the client side, for example by using one of the Azure Storage client libraries. The SAS token is not tracked by Azure Storage in any way. You can create an unlimited number of SAS tokens on the client side. After you create a SAS, you can distribute it to client applications that require access to resources in your storage account.

Code development and Deployment

Monday, September 19, 2022 12:06 PM

DP Repository - Where application code resides
Common Repository - Where we have the dependencies for the applications
Apollo Repositories -> Source code of apollo resides here

Hotfix: Generally Release Management team will create Hotfix branch

In branch name XX.XX.0 => Europe and LATAM and APAC
If there is any issue in the XX.XX.0 then the new branch will be based on that will be XX.XX.1 , XX.XX.3 (Odd numbers)
In branch name XX.XX.2 => US
If there is any issue in the XX.XX.2 then the new branch will be based on that will be XX.XX.4 , XX.XX.6 (Even Numbers)

We perform pull request to merge the development branch to the release branch. And QA team will be assessing the changes that we have done.

Automatic deployment of an application will be done using Jenkins , after we generate the pull request and Reviewers and QA team validation. And if there are any error's in automatic deployment then we get an automated mail related to the error.

https://nielsenenterprise-my.sharepoint.com/:b:/r/personal/kanaparthijeevan_sai_nielseniq_com/Documents/Attachments/Installation-Setup-Git-Bitbucket.docx.pdf?csf=1&web=1&e=vNbEFT

Docker SDK for Python

Monday, October 3, 2022 10:30 AM

Slides: https://ep2019.europython.eu/media/conference/slides/HVM9f5G-docker-meets-python-a-look-on-the-docker-sdk-for-python.pdf

Docker swarm / Docker Secrets

Wednesday, October 26, 2022 3:47 PM

- To use docker secrets , we must have the "Swarm" active , we can check it's status using "docker info" command in command line terminal.
- To know how many and what nodes we have running in docker swarm , use "docker node ls" command , so we can see the nodes that are running or down which worked for docker swarm. Here even we can see which node is the master node and which is worker node in manager status.
- We can't use this docker secrets for normal docker run command , we must use this docker secrets in docker swarm environment.

```
root@node1:~
[root@node1 ~]# docker run -d --name postgresdb -p 5432:5432 -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=password postg
res:9.4
67e9c5823947de534baaeacc7de338f1792a5eccdd6916f86201654a50745c36
[root@node1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
       NAMES
67e9c5823947        postgres:9.4        "docker-entrypoint.s…"   3 seconds ago       Up 2 seconds        0.0.0.0:5432->543
2/tcp   postgresdb
[root@node1 ~]# docker rm -f postgresdb
postgresdb
[root@node1 ~]#
```

To work on the docker secret , the first thing we need is docker swarm and these are the possible "docker secret" options to work on , and to know current list of secrets use "docker secret ls" command.

```
[root@node1 ~]# docker secret --help

Usage:  docker secret COMMAND

Manage Docker secrets

Commands:
  create      Create a secret from a file or STDIN as content
  inspect     Display detailed information on one or more secrets
  ls          List secrets
  rm          Remove one or more secrets

Run 'docker secret COMMAND --help' for more information on a command.
[root@node1 ~]# docker secret ls
ID                  NAME                DRIVER              CREATED             UPDATED
[root@node1 ~]#
```

```
[root@node1 ~]# ls
anaconda-ks.cfg  Dockerimage  dockersecrets  nginx  sonar  splunk  test  web
[root@node1 ~]# cd dockersecrets/
[root@node1 dockersecrets]# ls
[root@node1 dockersecrets]# vi dbpassword.txt
[root@node1 dockersecrets]# ls
dbpassword.txt
[root@node1 dockersecrets]# docker secret ls
ID                  NAME                DRIVER              CREATED             UPDATED
[root@node1 dockersecrets]# docker secret create db_pass dbpassword.txt
a0rukhake4dc6quoqya4w56pv
[root@node1 dockersecrets]# docker secret ls
ID                          NAME                DRIVER              CREATED             UPDATED
a0rukhake4dc6quoqya4w56pv   db_pass                                 6 seconds ago       6 seconds ago
[root@node1 dockersecrets]# docker secret inspect db_pass
[
    {
        "ID": "a0rukhake4dc6quoqya4w56pv",
        "Version": {
            "Index": 999
        },
        "CreatedAt": "2020-10-23T17:34:45.212777815Z",
        "UpdatedAt": "2020-10-23T17:34:45.212777815Z",
        "Spec": {
            "Name": "db_pass",
            "Labels": {}
        }
    }
]
[root@node1 dockersecrets]#
```

```
[root@node1 dockersecrets]# docker secret ls
ID                          NAME                DRIVER              CREATED             UPDATED
a0rukhake4dc6quoqya4w56pv   db_pass                                 55 seconds ago      55 seconds ago
[root@node1 dockersecrets]# docker service create --name postgresdb -p 5432:5432 -e POSTGRES_USER=postgres --secret db_pas
s -e POSTGRES_PASSWORD_FILE=/run/secrets/db_pass postgres:9.4
iriwldsubj35yydbhnz34zkma
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[root@node1 dockersecrets]# docker service ls
```

```
[root@node1 dockersecrets]# docker secret ls
ID                          NAME                DRIVER              CREATED             UPDATED
a0rukhake4dc6quoqya4w56pv   db_pass                                 55 seconds ago      55 seconds ago
[root@node1 dockersecrets]# docker service create --name postgresdb -p 5432:5432 -e POSTGRES_USER=postgres --secret db_pas
s -e POSTGRES_PASSWORD_FILE=/run/secrets/db_pass postgres:9.4
iriwldsubj35yydbhnz34zkma
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[root@node1 dockersecrets]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
iriwldsubj35        postgresdb          replicated          1/1                 postgres:9.4        *:5432->5432/tcp
[root@node1 dockersecrets]# docker service ps postgresdb
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE
   ERROR               PORTS
wnypppavcrn7        postgresdb.1        postgres:9.4        node1               Running             Running 35 seconds ago

[root@node1 dockersecrets]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
   NAMES
650e9b89ab3f        postgres:9.4        "docker-entrypoint.s…"   51 seconds ago      Up 50 seconds       5432/tcp
   postgresdb.1.wnypppavcrn74ml6up0pmei2c
[root@node1 dockersecrets]# docker exec -it postgresdb.1.wnypppavcrn74ml6up0pmei2c bin/bash
root@650e9b89ab3f:/# ls
bin   dev                          docker-entrypoint.sh  home  lib64  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint-initdb.d   etc                   lib   media  opt  root  sbin  sys  usr
root@650e9b89ab3f:/# cd /run/secrets
root@650e9b89ab3f:/run/secrets# ls
db_pass
```

```
root@650e9b89ab3f:/run/secrets# cat db_pass
password123
root@650e9b89ab3f:/run/secrets# exit
exit
[root@node1 dockersecrets]# ls
dbpassword.txt
[root@node1 dockersecrets]# cat dbpassword.txt
password123
[root@node1 dockersecrets]# ls
dbpassword.txt
[root@node1 dockersecrets]# rm -rf dbpassword.txt
[root@node1 dockersecrets]# ls
[root@node1 dockersecrets]# docker secret ls
ID                          NAME                DRIVER              CREATED             UPDATED
a0rukhake4dc6quoqya4w56pv   db_pass                                 5 minutes ago       5 minutes ago
[root@node1 dockersecrets]#
```

Device Mapper Storage Driver

Friday, December 9, 2022 12:17 PM

- Device Mapper is a kernel-based framework that underpins many advanced volume management technologies on Linux
- Docker's devicemapper storage driver leverages the thin provisioning and snapshotting capabilities of this framework for image and container management
- specific configuration is required to use it with Docker
- The devicemapper driver uses block devices dedicated to Docker and operates at the block level, rather than the file level.
- These devices can be extended by adding physical storage to your Docker host, and they perform better than using a filesystem at the operating system (OS) level.
- Prerequisites:
    - devicemapper is supported on Docker Engine - Community running on CentOS, Fedora, SLES 15, Ubuntu, Debian, or RHEL
    - devicemapper requires the lvm2 and device-mapper-persistent-data packages to be installed.