Docker-Exploration

Release 1.0.0

Nishant Baheti

Jan 01, 2023

ADDITIONAL DOCS

CHAPTER

ONE

DOCKER USED FOR DOCUMENTATION : DOCKER CE (COMMUNITY EDITION)

Fig. 1: logo

Fig. 2: concept1

- go to https://get.docker.com/
- take the script
- install it- easy-peasy
- curl -sSL https://get.docker.com/ | sh

CHAPTER

TWO

SOME BASIC DOCKER COMMANDS

Command	Description
docker version	Get the version information of docker.
docker info	Get info.
docker images	Get all available images in local repo.
docker container ps / docker container ps -a	get running containers (-a all stopped & running)
docker container run -p 80:80 -d –name test_container nginx	Run a container with nginx at port 80. bridge host IP 80 and container IP 80.
docker container run –rm -it image_name	run container and automatically remove upon close
docker container logs test_container	get logs for mentioned container
docker container top test_container	Get process/daemons running in the container
docker container rm …	Remove stopped container. Containers to be removed should be stopped.
docker container rm -f	Remove forcefully.
docker container inspect test_container	details of container config
docker container stats	show stats mem usage, cpu usage etc.
docker container run -it –name test_name image_name bash	run container (-i –> interactive,-t –> pseudo tty/ssh) and opens bash(changed default commands)
docker container start -ai container_name	starts existing (-ai start with given starting command) container
docker container stop container_name	stops existing container
docker container exec -it container_name bash	open bash in already running container
docker history image_name:tag	layer information of the image

CHAPTER

THREE

PORT

```
-p 8080:8080

[host_os_port : docker_container_port]
```

CHAPTER

FOUR

WHAT HAPPENS BEHIND DOCKER RUN

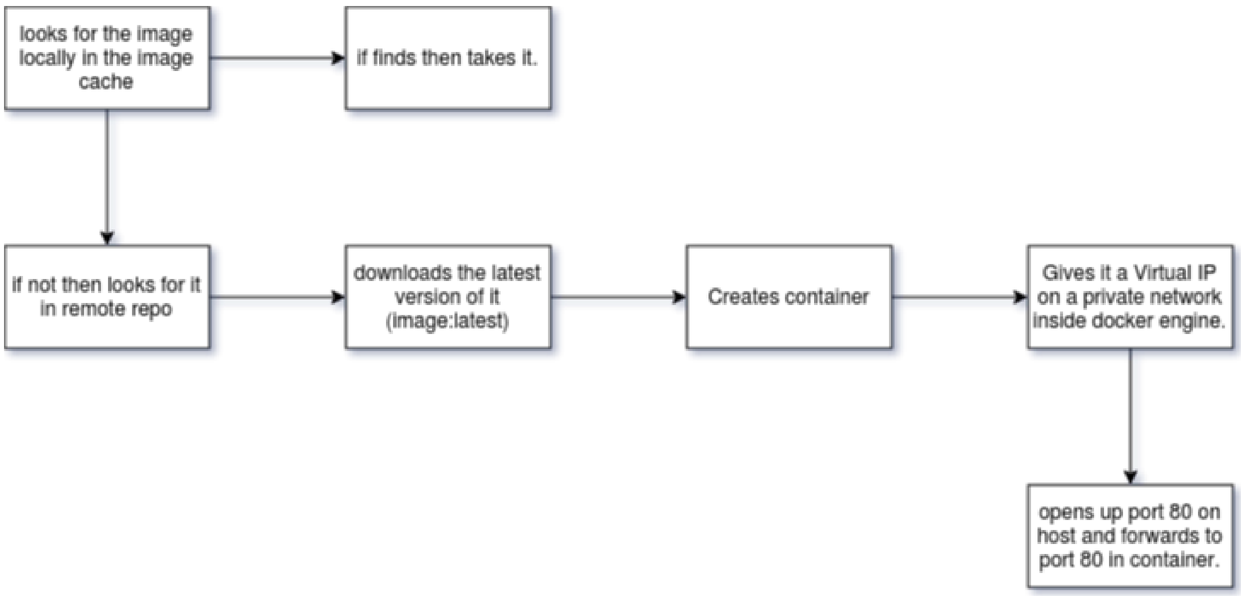

looks for the image locally in the image cache
if finds then takes it.
if not then looks for it in remote repo
downloads the latest version of it (image:latest)
Creates container
Gives it a Virtual IP on a private network inside docker engine.
opens up port 80 on host and forwards to port 80 in container.

Fig. 1: Image

CHAPTER

FIVE

POINTS TO NOTICE

- containers aren't mini VM's, they are just processes(binary files) running on HOST Operating Systems.
- Limited to what resource they can access.
- Exit when process is stopped

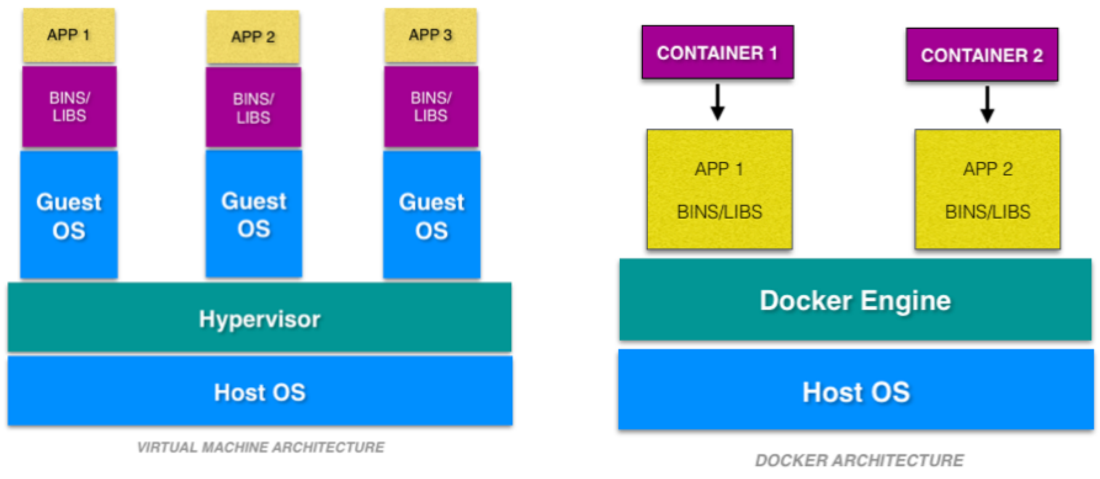


Fig. 1: concept2

CHAPTER

SIX

EXAMPLES

6.1 nginx

- docker pull nginx:latest
- docker run -p 80:80 –name nginx -d nginx:latest
- curl localhost

6.2 mongo

- docker pull mongo:latest
- docker run -p 27017:27017 –name mongo -d mongo:latest
- mongo –host localhost –port 27017

6.3 mysql

- docker pull mysql:latest
- docker run -p 3306:3306 –name mysql -e MYSQL_RANDOM_ROOT_PASSWORD=yes -d mysql:latest
- get first random password from docker container logs mysql (GENERATED ROOT PASSWORD)
- mysql -uroot -p[password from previous step] -h127.0.0.1 -P3306

  or
- docker run -p 3306:3306 –name mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:latest
- mysql -uroot -p my-secret-pw -h127.0.0.1 -P3306

CHAPTER

SEVEN

DOCKER NETWORKS

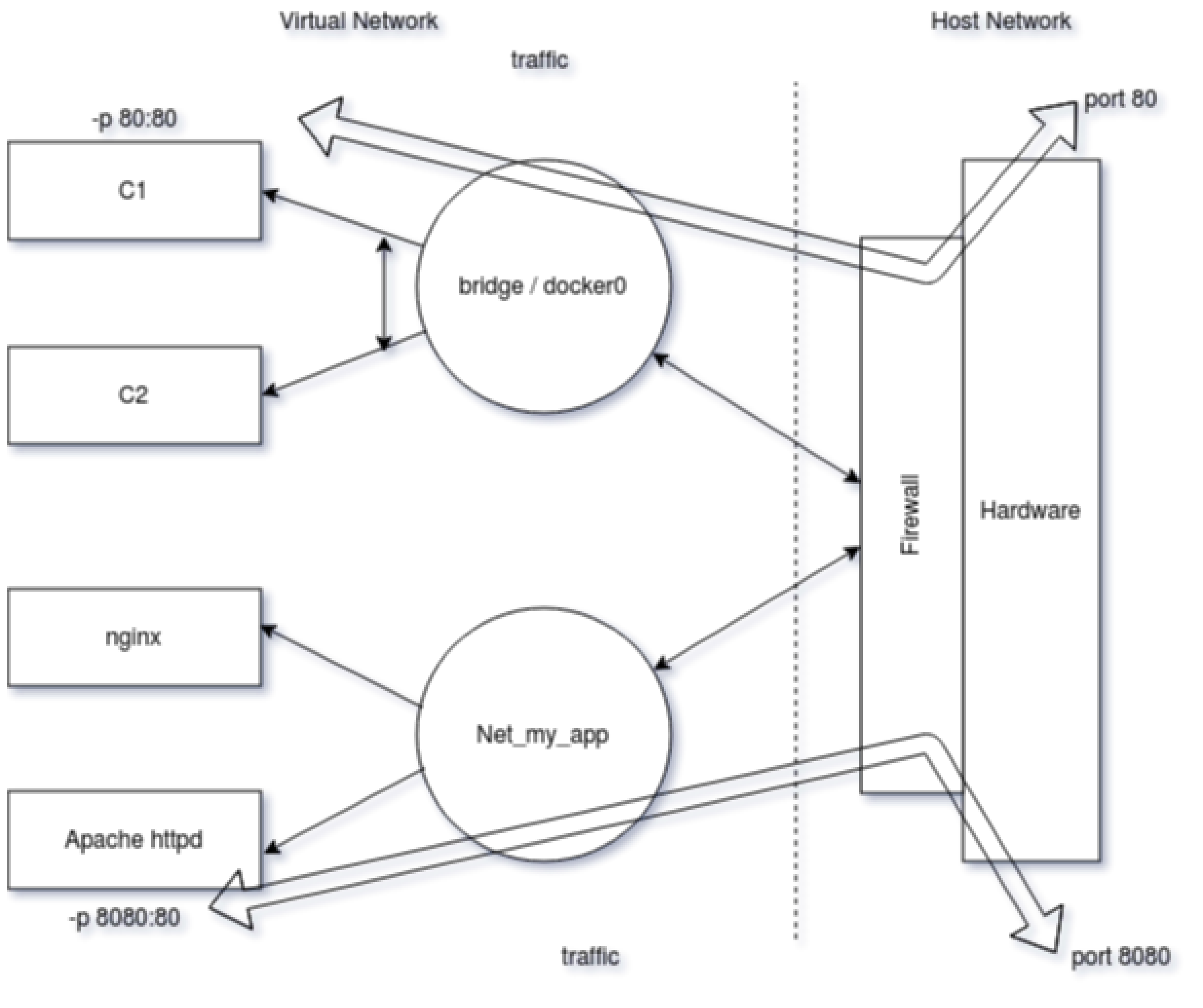


Fig. 1: concept3

Command	Description
docker container port container_name	get port info
docker container inspect –format "{{ .NetworkSettings.IPAddress }}" container_name	get IP
docker network ls	show networks
docker network inspect net_name	inspect a network
docker network create –driver	create a network
docker network connect net_id container_id	attach
docker network disconnect net_id container_id	detach
docker container run –name c_name –network net_name image_name	specifying network name in container while starting
docker container run –name c_name –net net_name –net-alias alias_name image_name	specifying network name and alias in container while starting (same alias containers can be called with same DNS name)

CHAPTER

EIGHT

DNS NAMING (INTER CONTAINER COMMUNICATION)

- containers cant rely on IP's for inter-communication.
- bridge (default) doesnt have this option.
- one container can communicate with another in same network with container name(instead of IP).
- it is easier in docker compose.

8.1 try this

- docker pull nginx:latest
- docker network create custom_network
- docker network ls
- docker run -it -d -p 8081:80 –network custom_network –name nginx2 nginx:latest
- docker run -it -d -p 8080:80 –network custom_network –name nginx1 nginx:latest
- docker container ls
- docker container exec -it nginx1 curl http://nginx2

CHAPTER NINE

IMAGE

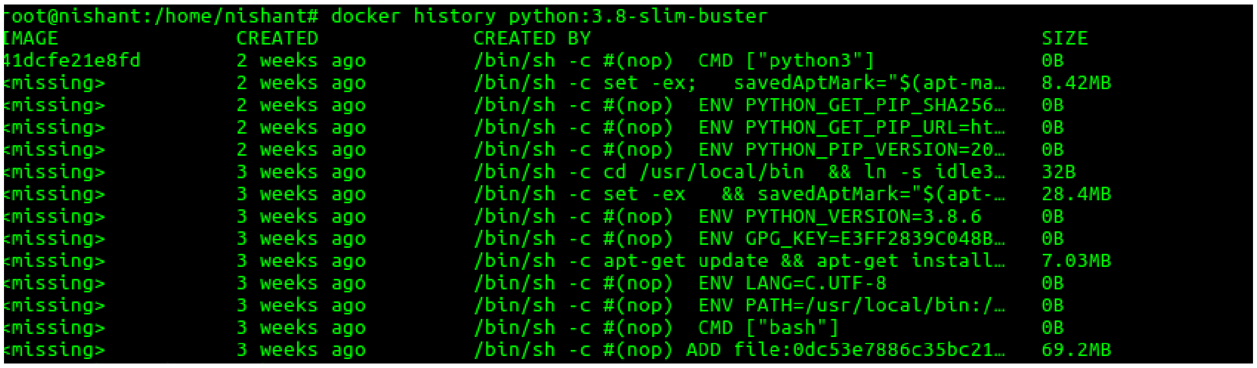
- app binaries and dependencies
- metadata about image data or how to run the image
- An image is an ordered collection of root filesystem changes and corresponding execution parameters for use within a container runtime.
- Not a complete OS. No kerel ,kernel modules etc.

9.1 Image Layers

image
env
apt
ubuntu

image1	image2	
port	other operation	only diff is added in runtime container
copy	copy	common till here
apt	apt	
Debian jessie	Debain jessie	

example of layers:



```
root@nishant:/home/nishant# docker history python:3.8-slim-buster
IMAGE               CREATED             CREATED BY                                      SIZE
41dcfe21e8fd        2 weeks ago         /bin/sh -c #(nop)  CMD ["python3"]              0B
<missing>           2 weeks ago         /bin/sh -c set -ex;   savedAptMark="$(apt-ma…   8.42MB
<missing>           2 weeks ago         /bin/sh -c #(nop)  ENV PYTHON_GET_PIP_SHA256…   0B
<missing>           2 weeks ago         /bin/sh -c #(nop)  ENV PYTHON_GET_PIP_URL=ht…   0B
<missing>           2 weeks ago         /bin/sh -c #(nop)  ENV PYTHON_PIP_VERSION=20…   0B
<missing>           3 weeks ago         /bin/sh -c cd /usr/local/bin  && ln -s idle3…   32B
<missing>           3 weeks ago         /bin/sh -c set -ex   && savedAptMark="$(apt-…   28.4MB
<missing>           3 weeks ago         /bin/sh -c #(nop)  ENV PYTHON_VERSION=3.8.6     0B
<missing>           3 weeks ago         /bin/sh -c #(nop)  ENV GPG_KEY=E3FF2839C048B…   0B
<missing>           3 weeks ago         /bin/sh -c apt-get update && apt-get install…   7.03MB
<missing>           3 weeks ago         /bin/sh -c #(nop)  ENV LANG=C.UTF-8             0B
<missing>           3 weeks ago         /bin/sh -c #(nop)  ENV PATH=/usr/local/bin:/…   0B
<missing>           3 weeks ago         /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>           3 weeks ago         /bin/sh -c #(nop) ADD file:0dc53e7886c35bc21…   69.2MB
```

Fig. 1: imagelayers

9.2 Image representation

```
<user>/<repo>:<tag>
```

CHAPTER TEN

DOCKERFILE

Dockerfile is a recipe for creating image.

Command	Description
docker image build -f some-dockerfile	build image from a dockerfile
docker image build -t custom_nginx .	build docker image with tag custom_nginx from current working directory

Keyword	Description
FROM	All dockerfile must have to minimal distribution. want to go completely from scratch use “FROM scratch”
ENV	Setting up environment variables. inject main key/values for image.
RUN	Run shell commads
EXPOSE	Expose ports on docker virtual network still need to use -p / -P on host os
CMD	Final command to be run every time container is launched/started
COPY	Copy from local(host) os to docker(guest/virtual) os
ENTRYPOINT	Entrypoint for a container at runtime
WORKDIR	is prefered to using “RUN cd /some/path”
VOLUME	Create a new volume location and assign it to the directory in the container will outlive the container when container is updated. (requires manual deletion)
ADD	

```
It is adviced to keep least changing things in the
docker images to keep on top(initial steps) and more
variable things in later steps so that whenver any step changes or updates till that␣
↪step cache will help to
speed up the process of building the image.
```

CHAPTER

ELEVEN

PRUNE

Command	Description
docker image prune	remove all dangling images
docker system prune	remove everything

CHAPTER

TWELVE

CONTAINER LIFETIME AND PERSISTENT DATA

1. immutable (unchanging) and ephemeral (temporary/ disposable).
2. “immutable infrastructure” : only re-deploy containers, never change.
3. But if there is some data that has to be present (like database or unique data).
4. data can be preserved when container is getting updated with latest version. docker gives us feature to ensure “separation of concerns”.
5. This is called as “Presistent data”.
6. 2 solutions for this - Volumns and Bind Mounts.
7. VOLUMES : make special location outside of container UFS(union file

   system).
8. BIND MOUNT : link container path to host path.

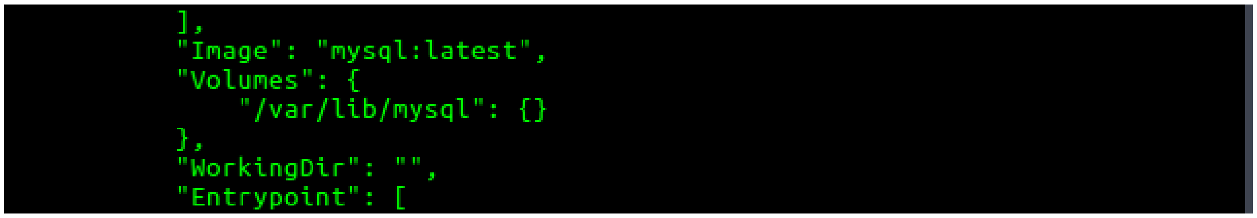
CHAPTER

THIRTEEN

PERSISTENT DATA

- **DATA VOLUMES**

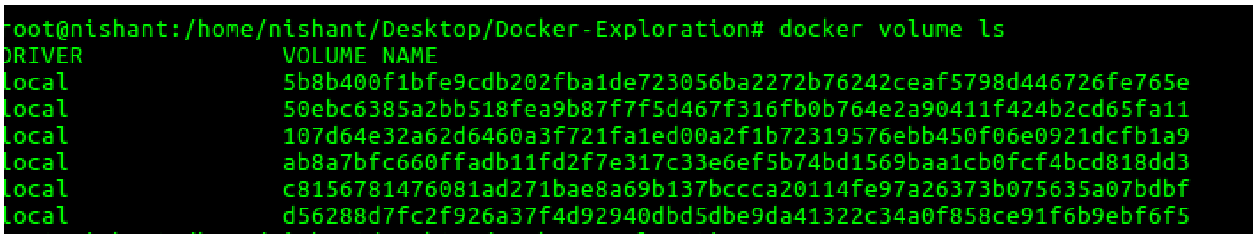
1. Create a new volume location and assign it to the directory in the container
2. will outlive the container when container is updated.
3. requires manual deletion



```
           ],
           "Image": "mysql:latest",
           "Volumes": {
               "/var/lib/mysql": {}
           },
           "WorkingDir": "",
           "Entrypoint": [
```

Fig. 1: volumeInfo

Command	Description
docker volume ls	list of volumes
docker volume inspect volume_name	information about volume
docker volume create volumne_name	create volume

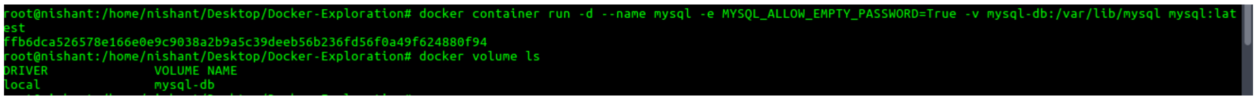


```
oot@nishant:/home/nishant/Desktop/Docker-Exploration# docker volume ls
RIVER                 VOLUME NAME
ocal                  5b8b400f1bfe9cdb202fba1de723056ba2272b76242ceaf5798d446726fe765e
ocal                  50ebc6385a2bb518fea9b87f7f5d467f316fb0b764e2a90411f424b2cd65fa11
ocal                  107d64e32a62d6460a3f721fa1ed00a2f1b72319576ebb450f06e0921dcfb1a9
ocal                  ab8a7bfc660ffadb11fd2f7e317c33e6ef5b74bd1569baa1cb0fcf4bcd818dd3
ocal                  c8156781476081ad271bae8a69b137bccca20114fe97a26373b075635a07bdbf
ocal                  d56288d7fc2f926a37f4d92940dbd5dbe9da41322c34a0f858ce91f6b9ebf6f5
```

Fig. 2: volumes1

```
docker container run -d --name mysql -e MYSQL_ALLOW_EMPTY_PASSWORD=True -v mysql-db:/var/
↪lib/mysql mysql:latest
```

- if name is provided then it will register by name otherwise by default a random name would be generated. (Named volumes)
- -v [name]:[path/to/volume]



```
root@nishant:/home/nishant/Desktop/Docker-Exploration# docker container run -d --name mysql -e MYSQL_ALLOW_EMPTY_PASSWORD=True -v mysql-db:/var/lib/mysql mysql:lat
est
ffb6dca526578e166e0e9c9038a2b9a5c39deeb56b236fd56f0a49f624880f94
root@nishant:/home/nishant/Desktop/Docker-Exploration# docker volume ls
DRIVER              VOLUME NAME
local               mysql-db
```

Fig. 3: volumes2

- **BIND MOUNTING**

1. Maps a host file or dir to container file or directory.
2. basically two locations pointing to same file.
3. Skips UFS, host files overwrite any in container.
4. Cant use Dockerfile, has to be mentioned in docker container run command.
5. -v [/host/fs/path]:[/container/fs/path]
6. Try

```
docker container run -it -d -p 3000:80 --name nginx -v /home/nishant/Desktop/Docker-
↪Exploration/htmlexample:/usr/share/nginx/html nginx:latest
```

CHAPTER

FOURTEEN

DOCKER COMPOSE

- Configure relationships between containers.
- Save docker container run settings in easy-to-read file
- One liner developer env setup.
- 1. YAML file - containers, networks, volumes, env.(default docker-compose.yml/yaml)
  2. CLI tool - docker-compose

14.1 docker-compose CLI

- CLI tool is not a production grade tool but ideal for development and test.

Command	Description
docker-compose up	setup volumes,networks and start all containers
docker-compose up -f file_name	setup volumes,networks and start all containers with a custom file_name
docker-compose down	stop all containers and remove containers/vols/nets
docker-compose up -d	setup volumes,networks and start all containers and detach
docker-compose ps	get services running
docker-compose run	
docker-compose stop	

14.2 docker-compose versioning

There are three legacy versions of the Compose file format:

- Version 1. This is specified by omitting a version key at the root of the YAML.
- Version 2.x. This is specified with a version: '2' or version: '2.1', etc., entry at the root of the YAML.
- Version 3.x, designed to be cross-compatible between Compose and the Docker Engine's swarm mode. This is specified with a version: '3' or version: '3.1', etc., entry at the root of the YAML.

CHAPTER

FIFTEEN

CONTAINERS EVERYWHERE

15.1 Some major tasks

- automate container lifecycle
- easily scale up/down/out/in
- container recreation upon failing
- replace container without downtime (blue/green deploy)
- control/track container started
- create cross-node virtual network
- only trusted servers run containers
- store secrets, keys, passwords and access them in right containers

CHAPTER

SIXTEEN

DOCKER SWARM - CONTAINER ORCHESTRATION

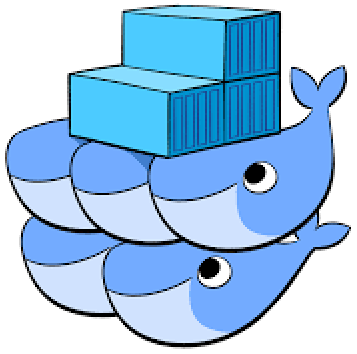
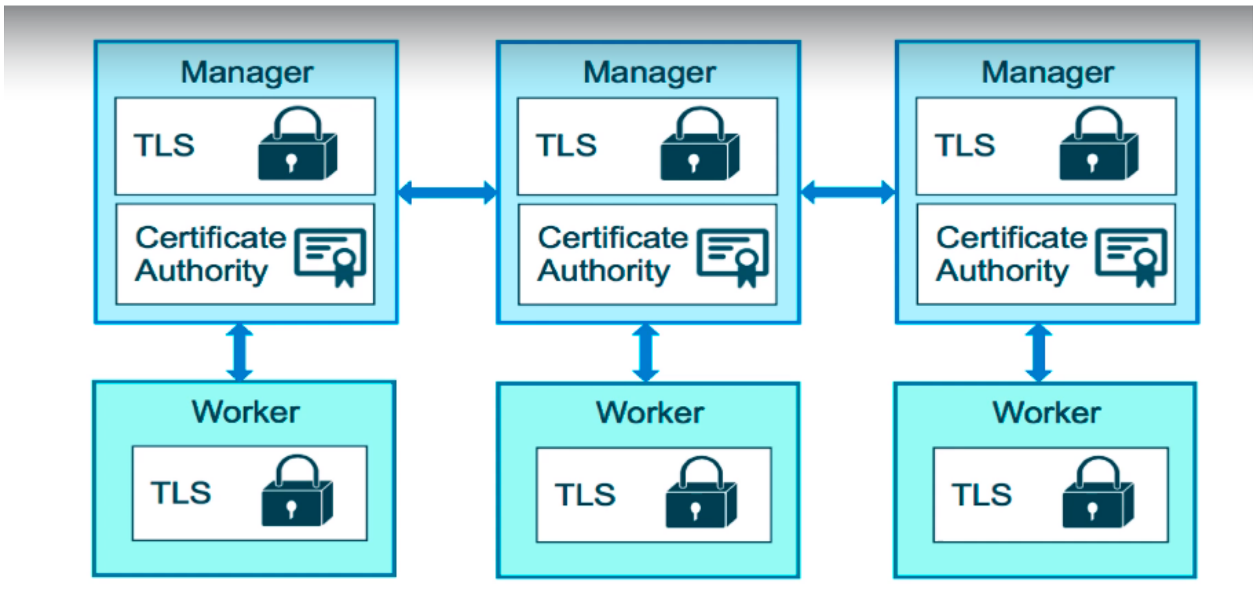
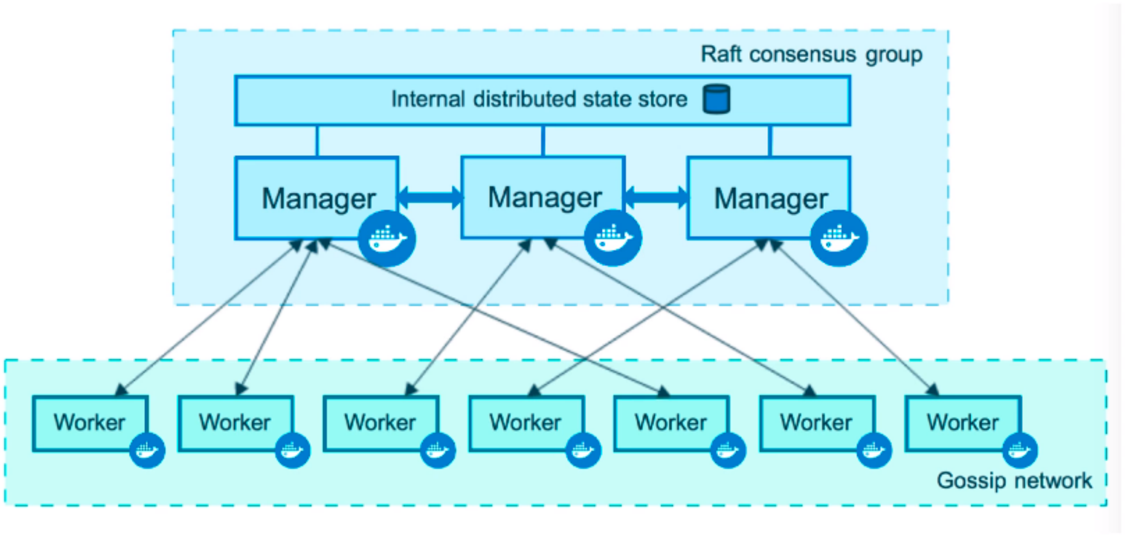
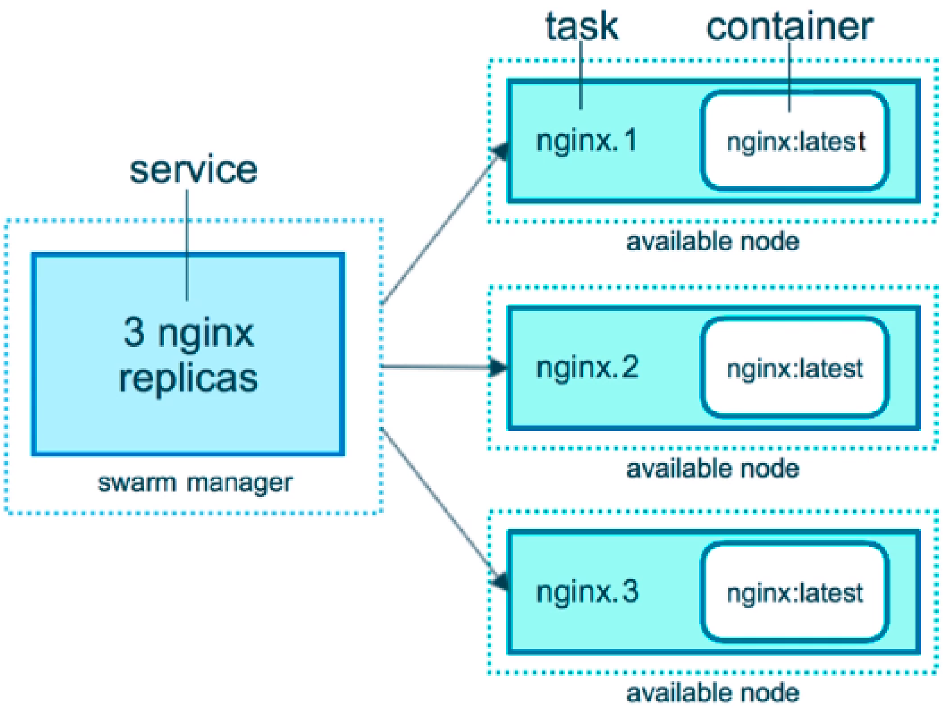


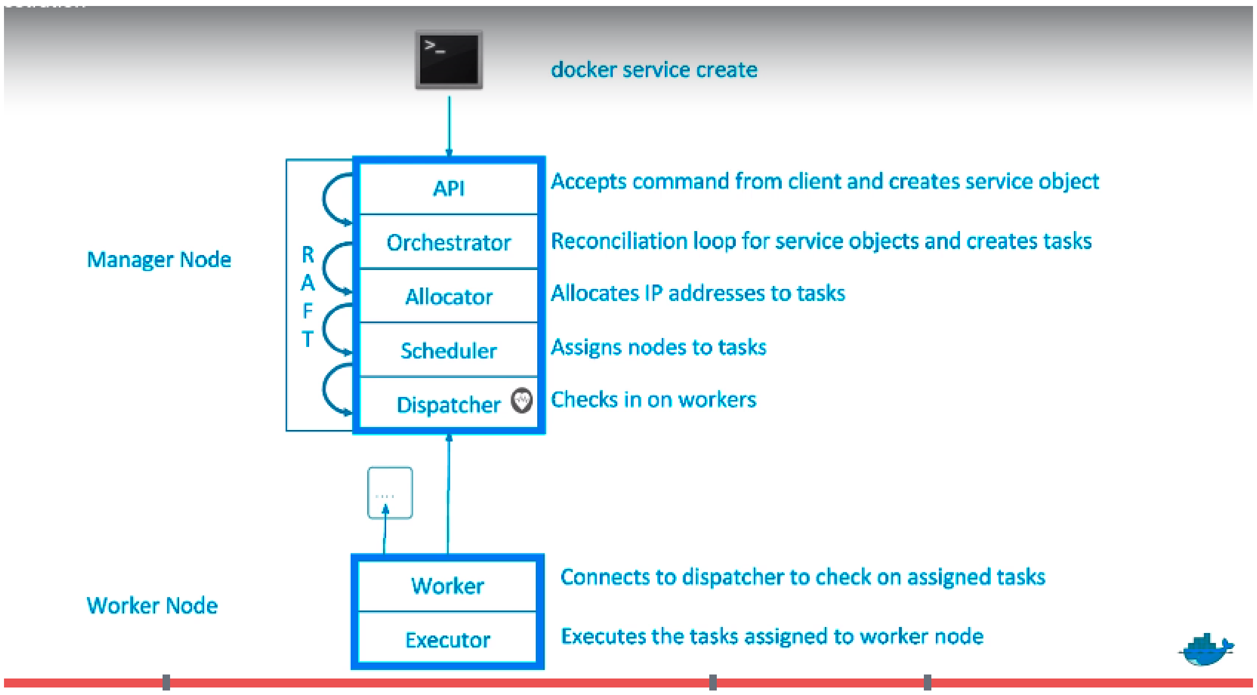
Fig. 1: swarm5

- Swarm mode is a clustering solution built inside Docker
- docker swarm, docker node, docker service, docker stack, docker secret




Raft consensus group
Internal distributed state store
Manager
Manager
Manager
Worker
Worker
Worker
Worker
Worker
Worker
Worker
Gossip network


task
container
service
nginx.1
nginx:latest
available node
3 nginx replicas
swarm manager
nginx.2
nginx:latest
available node
nginx.3
nginx:latest
available node

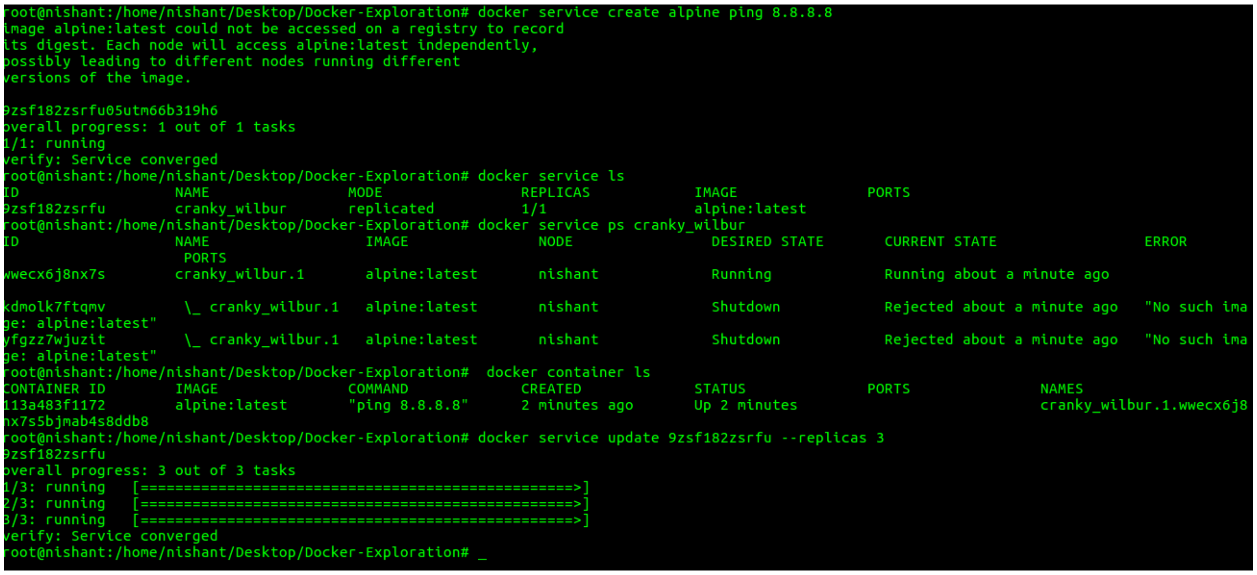

docker service create
Manager Node
R A F T
API
Accepts command from client and creates service object
Orchestrator
Reconciliation loop for service objects and creates tasks
Allocator
Allocates IP addresses to tasks
Scheduler
Assigns nodes to tasks
Dispatcher
Checks in on workers
Worker Node
Worker
Connects to dispatcher to check on assigned tasks
Executor
Executes the tasks assigned to worker node

16.1 docker swarm init

- PKI and security automation
  1. Root signing certificate created for swarm
  2. certificate is issued for first manager node
  3. join tokens are created
- RAFT database created to store root CA, configs and secrets
  1. no additional key value storage system
  2. replicates logs amongs managers.

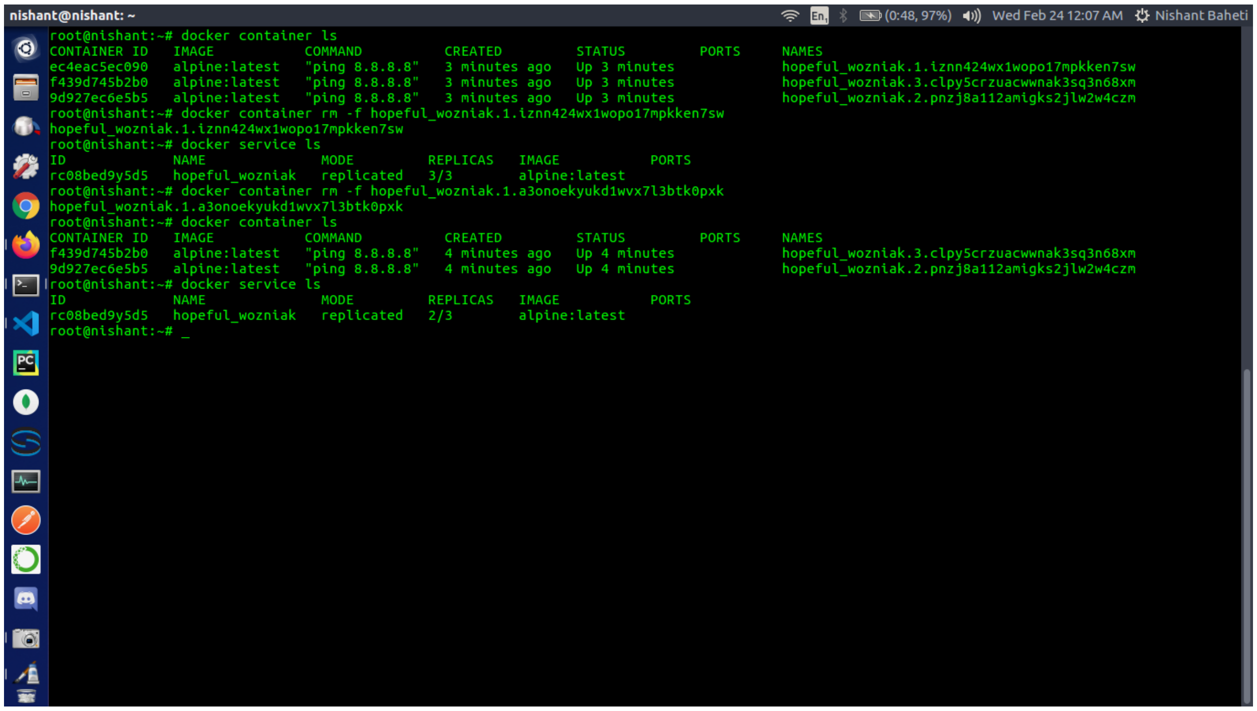
Command	Description
docker swarm init	initialize
docker node ls	list down nodes
docker service create	creating a container service
docker service ls	list down services
docker service ps service_name	process information
docker service update service_id –replicas number	update replicas
docker service rm service_name	remove service and delete all containers one by one

- if a service is running and we stop one of its replicas by running “docker container rm -f some_id/name” then it will show in the results of “docker service ls” (one less replica) but within seconds it will again start it and it will show in the result if “docker service ps service_name” that one service was stopped.



```
root@nishant:/home/nishant/Desktop/Docker-Exploration# docker service create alpine ping 8.8.8.8
image alpine:latest could not be accessed on a registry to record
its digest. Each node will access alpine:latest independently,
possibly leading to different nodes running different
versions of the image.

9zsf182zsrfu05utm66b319h6
overall progress: 1 out of 1 tasks
1/1: running
verify: Service converged
root@nishant:/home/nishant/Desktop/Docker-Exploration# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
9zsf182zsrfu        cranky_wilbur       replicated          1/1                 alpine:latest
root@nishant:/home/nishant/Desktop/Docker-Exploration# docker service ps cranky_wilbur
ID                  NAME                  IMAGE               NODE                DESIRED STATE       CURRENT STATE                ERROR
                     PORTS
wwecx6j8nx7s        cranky_wilbur.1       alpine:latest       nishant             Running             Running about a minute ago

kdmolk7ftqmv         \_ cranky_wilbur.1   alpine:latest       nishant             Shutdown            Rejected about a minute ago   "No such ima
ge: alpine:latest"
yfgzz7wjuzit         \_ cranky_wilbur.1   alpine:latest       nishant             Shutdown            Rejected about a minute ago   "No such ima
ge: alpine:latest"
root@nishant:/home/nishant/Desktop/Docker-Exploration#  docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
113a483f1172        alpine:latest       "ping 8.8.8.8"      2 minutes ago       Up 2 minutes                            cranky_wilbur.1.wwecx6j8
nx7s5bjmab4s8ddb8
root@nishant:/home/nishant/Desktop/Docker-Exploration# docker service update 9zsf182zsrfu --replicas 3
9zsf182zsrfu
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
root@nishant:/home/nishant/Desktop/Docker-Exploration# _
```

Fig. 2: docker-service1



nishant@nishant: ~ (0:48, 97%) Wed Feb 24 12:07 AM Nishant Baheti

```
root@nishant:~# docker container ls
CONTAINER ID   IMAGE           COMMAND          CREATED         STATUS         PORTS     NAMES
ec4eac5ec090   alpine:latest   "ping 8.8.8.8"   3 minutes ago   Up 3 minutes             hopeful_wozniak.1.iznn424wx1wopo17mpkken7sw
f439d745b2b0   alpine:latest   "ping 8.8.8.8"   3 minutes ago   Up 3 minutes             hopeful_wozniak.3.clpy5crzuacwwnak3sq3n68xm
9d927ec6e5b5   alpine:latest   "ping 8.8.8.8"   3 minutes ago   Up 3 minutes             hopeful_wozniak.2.pnzj8a112amigks2jlw2w4czm
root@nishant:~# docker container rm -f hopeful_wozniak.1.iznn424wx1wopo17mpkken7sw
hopeful_wozniak.1.iznn424wx1wopo17mpkken7sw
root@nishant:~# docker service ls
ID             NAME              MODE         REPLICAS   IMAGE           PORTS
rc08bed9y5d5   hopeful_wozniak   replicated   3/3        alpine:latest
root@nishant:~# docker container rm -f hopeful_wozniak.1.a3onoekyukd1wvx7l3btk0pxk
hopeful_wozniak.1.a3onoekyukd1wvx7l3btk0pxk
root@nishant:~# docker container ls
CONTAINER ID   IMAGE           COMMAND          CREATED         STATUS         PORTS     NAMES
f439d745b2b0   alpine:latest   "ping 8.8.8.8"   4 minutes ago   Up 4 minutes             hopeful_wozniak.3.clpy5crzuacwwnak3sq3n68xm
9d927ec6e5b5   alpine:latest   "ping 8.8.8.8"   4 minutes ago   Up 4 minutes             hopeful_wozniak.2.pnzj8a112amigks2jlw2w4czm
root@nishant:~# docker service ls
ID             NAME              MODE         REPLICAS   IMAGE           PORTS
rc08bed9y5d5   hopeful_wozniak   replicated   2/3        alpine:latest
root@nishant:~# _
```

Fig. 3: docker-service2

16.2 PLAYGROUND

- https://labs.play-with-docker.com
- use above link to create instances and play with them

16.3 Steps

- get 3 instances
- in one instance run

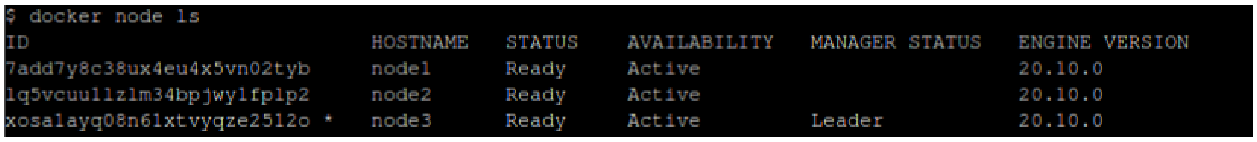
```
docker swarm  init --advertise-addr <public_ip>
```

- this will give a url like

```
docker swarm join --token <some token>
```

- run this command in other two instances to join them in this cluster
- now docker swarm commands cant be run in these worker nodes
- Run in the leader instance

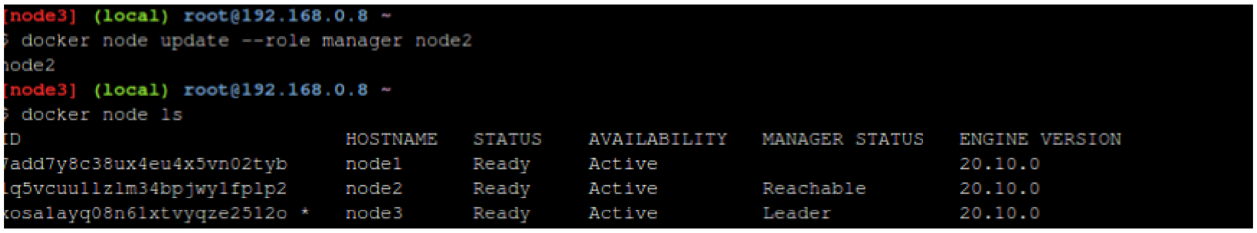
```
docker node ls
```



```
$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
7add7y8c38ux4eu4x5vn02tyb     node1      Ready     Active                          20.10.0
1q5vcuu1lzlm34bpjwy1fplp2     node2      Ready     Active                          20.10.0
xosalayq08n61xtvyqze2512o *   node3      Ready     Active         Leader           20.10.0
```

Fig. 4: dokcer-swarm1

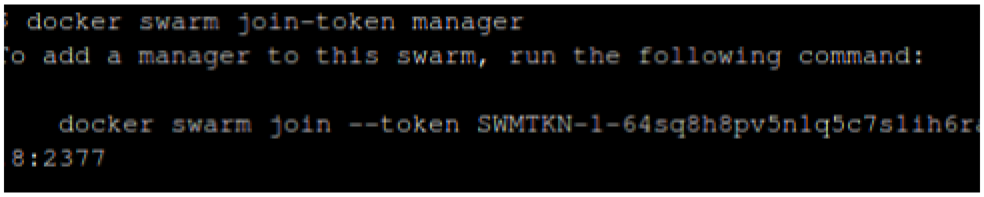
- change the role of a node



```
[node3] (local) root@192.168.0.8 ~
$ docker node update --role manager node2
node2
[node3] (local) root@192.168.0.8 ~
$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
7add7y8c38ux4eu4x5vn02tyb     node1      Ready     Active                          20.10.0
1q5vcuu1lzlm34bpjwy1fplp2     node2      Ready     Active         Reachable        20.10.0
xosalayq08n61xtvyqze2512o *   node3      Ready     Active         Leader           20.10.0
```

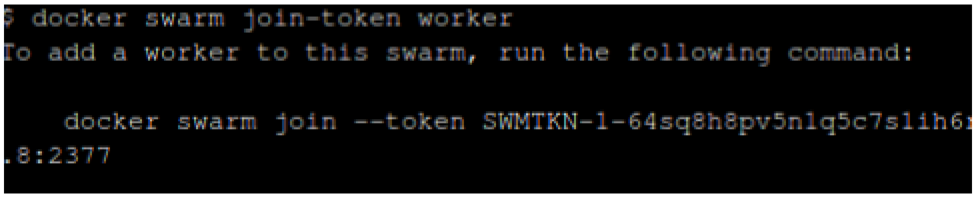
Fig. 5: docker-swarm2

- get the manager token to join anytime and add instance with predefined manager role
- get the worker token to join anytime
- now create a service with 3 replicas



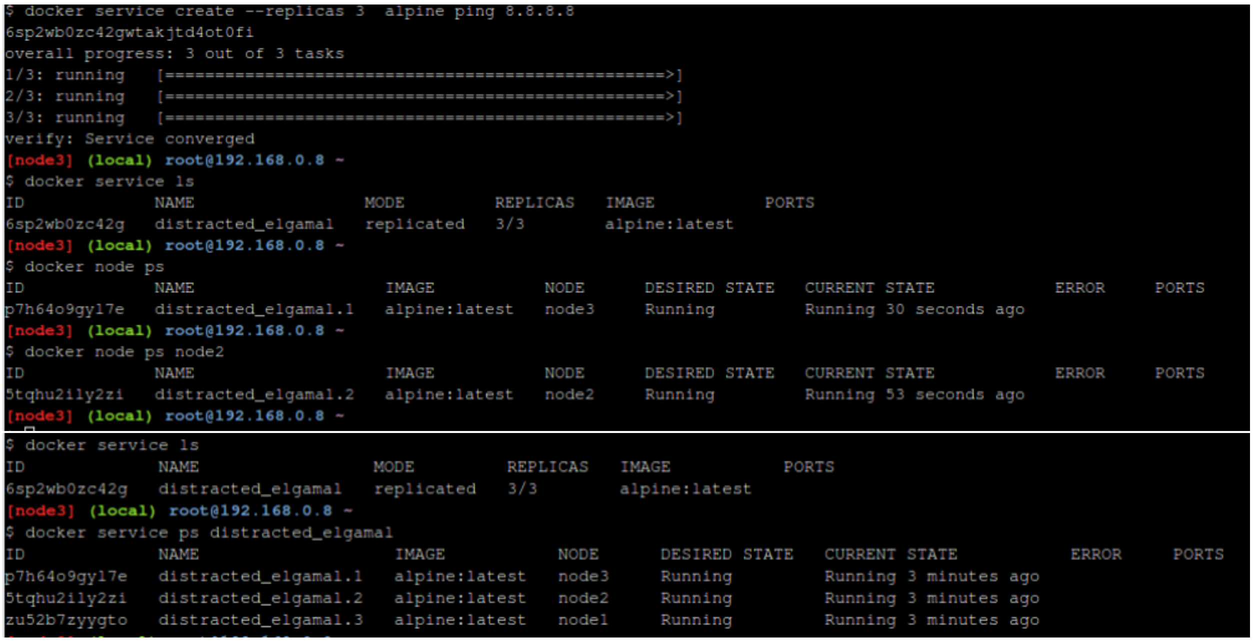
```
docker swarm join-token manager
o add a manager to this swarm, run the following command:

   docker swarm join --token SWMTKN-1-64sq8h8pv5n1q5c7s1ih6r
8:2377
```

Fig. 6: docker-swarm3



```
docker swarm join-token worker
To add a worker to this swarm, run the following command:

   docker swarm join --token SWMTKN-1-64sq8h8pv5n1q5c7s1ih6
.8:2377
```

Fig. 7: docker-swarm4



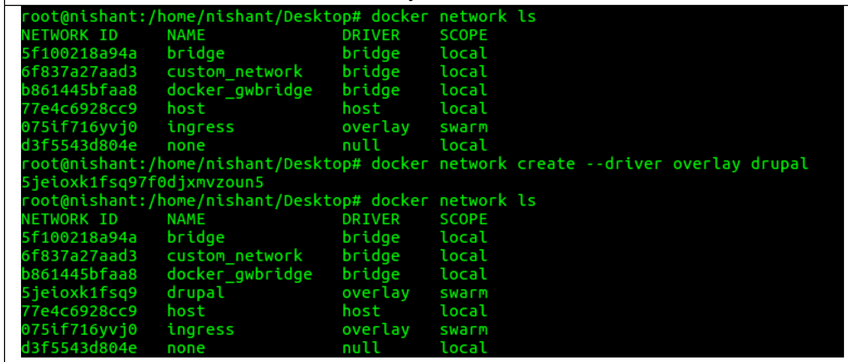
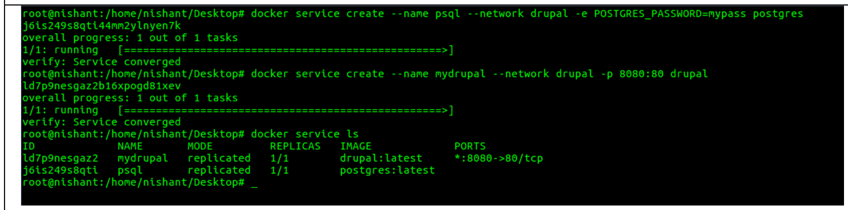
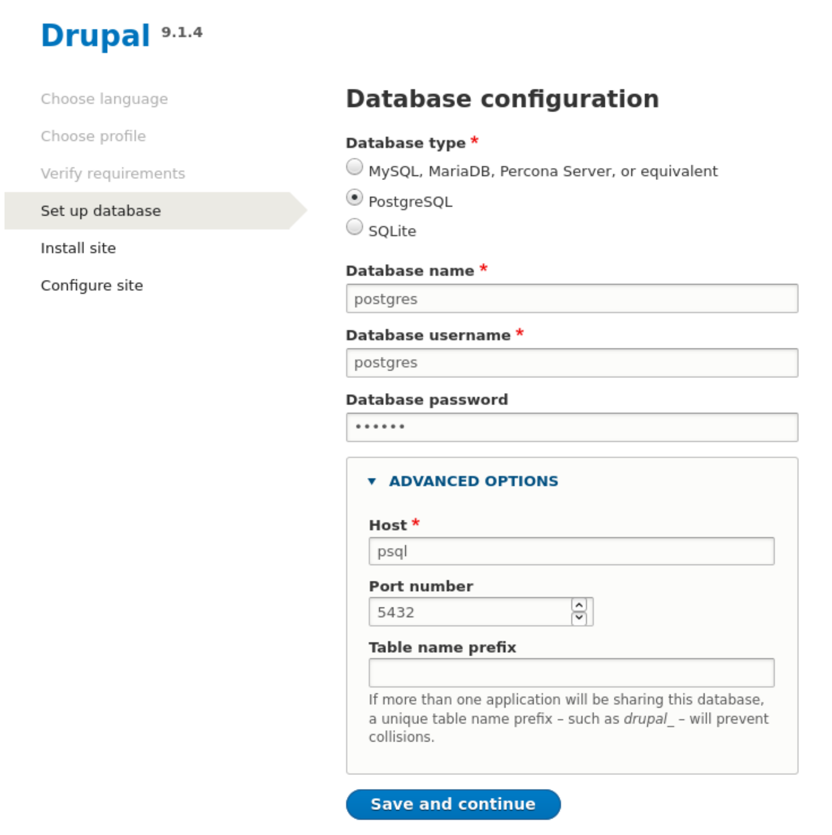
```
$ docker service create --replicas 3  alpine ping 8.8.8.8
6sp2wb0zc42gwtakjtd4ot0fi
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
[node3] (local) root@192.168.0.8 ~
$ docker service ls
ID             NAME                MODE         REPLICAS   IMAGE           PORTS
6sp2wb0zc42g   distracted_elgamal  replicated   3/3        alpine:latest
[node3] (local) root@192.168.0.8 ~
$ docker node ps
ID             NAME                   IMAGE           NODE      DESIRED STATE   CURRENT STATE            ERROR     PORTS
p7h64o9gyl7e   distracted_elgamal.1   alpine:latest   node3     Running         Running 30 seconds ago
[node3] (local) root@192.168.0.8 ~
$ docker node ps node2
ID             NAME                   IMAGE           NODE      DESIRED STATE   CURRENT STATE            ERROR     PORTS
5tqhu2ily2zi   distracted_elgamal.2   alpine:latest   node2     Running         Running 53 seconds ago
[node3] (local) root@192.168.0.8 ~
$ docker service ls
ID             NAME                 MODE         REPLICAS    IMAGE           PORTS
6sp2wb0zc42g   distracted_elgamal   replicated   3/3         alpine:latest
[node3] (local) root@192.168.0.8 ~
$ docker service ps distracted_elgamal
ID             NAME                   IMAGE           NODE      DESIRED STATE   CURRENT STATE           ERROR     PORTS
p7h64o9gyl7e   distracted_elgamal.1   alpine:latest   node3     Running         Running 3 minutes ago
5tqhu2ily2zi   distracted_elgamal.2   alpine:latest   node2     Running         Running 3 minutes ago
zu52b7zyygto   distracted_elgamal.3   alpine:latest   node1     Running         Running 3 minutes ago
```

CHAPTER

SEVENTEEN

OVERLAY MULTI HOST NETWORKING

- choose –driver overlay when creating network
- for container to container traffic inside a Single Swarm
- Optional IPSec (AES) encryption on network creation
- Each service can connect to multiple networks

Command	Description
docker network create –driver overlay network_name	create a overlay network
	creating a network
	creating two services on one network
	accessing them by their service name (look at host)

```
root@nishant:/home/nishant/Desktop# docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
5f100218a94a   bridge            bridge    local
6f837a27aad3   custom_network    bridge    local
b861445bfaa8   docker_gwbridge   bridge    local
77e4c6928cc9   host              host      local
075if716yvj0   ingress           overlay   swarm
d3f5543d804e   none              null      local
root@nishant:/home/nishant/Desktop# docker network create --driver overlay drupal
5jeioxk1fsq97f0djxmvzoun5
root@nishant:/home/nishant/Desktop# docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
5f100218a94a   bridge            bridge    local
6f837a27aad3   custom_network    bridge    local
b861445bfaa8   docker_gwbridge   bridge    local
5jeioxk1fsq9   drupal            overlay   swarm
77e4c6928cc9   host              host      local
075if716yvj0   ingress           overlay   swarm
d3f5543d804e   none              null      local
```

```
root@nishant:/home/nishant/Desktop# docker service create --name psql --network drupal -e POSTGRES_PASSWORD=mypass postgres
j6is249s8qti44mm2ylnyen7k
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
root@nishant:/home/nishant/Desktop# docker service create --name mydrupal --network drupal -p 8080:80 drupal
ld7p9nesgaz2b16xpogd81xev
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
root@nishant:/home/nishant/Desktop# docker service ls
ID             NAME       MODE         REPLICAS   IMAGE             PORTS
ld7p9nesgaz2   mydrupal   replicated   1/1        drupal:latest     *:8080->80/tcp
j6is249s8qti   psql       replicated   1/1        postgres:latest
root@nishant:/home/nishant/Desktop# _
```

Database configuration

Database type *
- MySQL, MariaDB, Percona Server, or equivalent
- [✓ PostgreSQL]
- SQLite

Database name *: postgres

Database username *: postgres

Database password: ••••••

Host *: psql

Port number: 5432

Table name prefix:

If more than one application will be sharing this database, a unique table name prefix – such as *drupal_* – will prevent collisions.

17.1 Routing Mesh (Internal Load Balancer)

- Routes/distributes ingress (incoming) packets for a service to a proper task
- spans all the nodes
- Uses IPVS from linux kernel (kernel primitives)
- Load balances swarm services across their tasks
- ways to work
  - container to container overlay network (talking to virtual IP/VIP)
  - external traffic incoming to publishing ports (all nodes listen)
- stateless load balancing

CHAPTER

EIGHTEEN

DOCKER STACK

18.1 Production Grade Compose

- New layer of abstraction to swarms called stacks
- accepts compose files
- `docker stack deploy`

```
         services  task and container
             ^          ^
        || service1 -| node 1  |
        ||          -| node 2  |  || Volumes ||
        ||-------------------- |
Stack ->|| service2 -| node 1  |
        ||          -| node 2  |
        ||-------------------- | || Overlay Networks ||
        || service3 -| node 1  |
        ||          -| node 2  |
```

Command	Description
docker stack deploy -c compose_file app_name	queue deploy services from a compose file
docker stack ls	list all the apps in the stack
docker stack ps app_name	list down services in the app
docker stack services app_name	gives important info about services like replicas,mode etc.

CHAPTER

NINETEEN

DOCKER SECRETS

- key value store in docker run time
- attach it to services only those can use it

Command	Description
docker secret create secret_name secret_file.txt	put value in secret by a file
echo "some_value" | docker secret create secret_name -	put value in secret by echoing
docker secret ls	list down secrets
———	———
with service	
docker service create –name service_name –secret secret_name	create a service with a secret mentioned that can be used by container
docker service update –secret-rm secret_name	remove secret

CHAPTER TWENTY

SWARM APP LIFECYCLE

Three important things in this trilogy is swarm, stack and secrets

```
$ docker-compose up #for development env
$ docker-compose up #for CI env
$ docker stack deploy #for production env
```

CHAPTER

TWENTYONE

KUBERNETES

- container orchestration
- runs on top of docker (usually)
- provides api/cli to manage containers across servers

21.1 sandbox

- https://labs.play-with-k8s.com/
- katacoda

21.2 Other flavours

- minikube
- MicroK8s

21.3 Cloud providers

- Azure Kubernetes Services (AKS)
- AWS (EKS)
- Google Cloud

21.4 Terminologies

- kubectl - cube control (cli)
- node - single server inside the cluster
- kubelet - Kubernetes agent running on nodes

```
In swarm in build docker swarm agent is available for workers to talk back to the␣
↪master nodes kubernetes needs one explicitly
```

- control plane - set of containers that manages the clusters

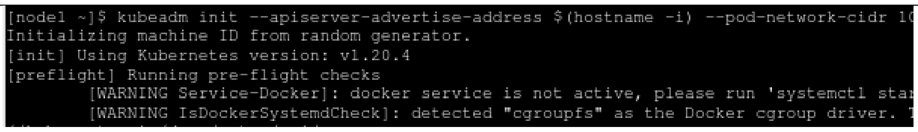
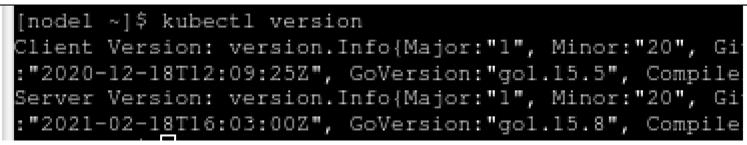
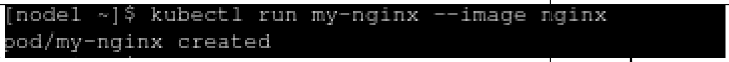
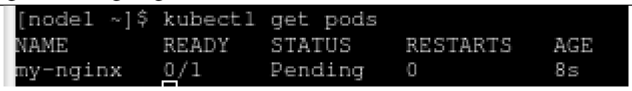
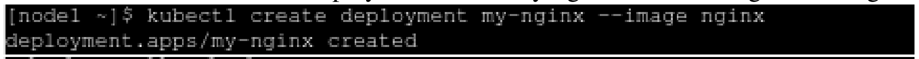
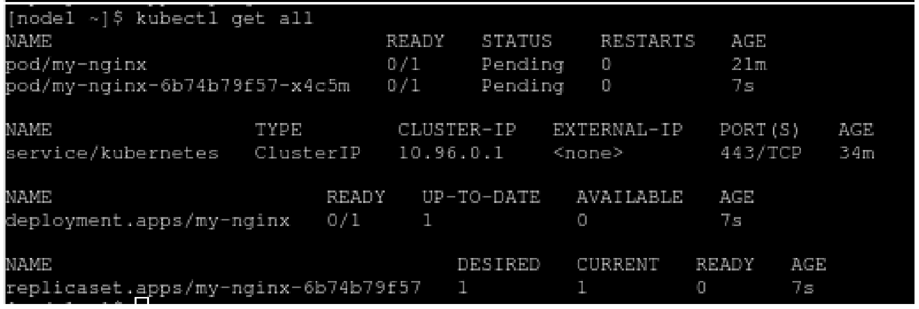
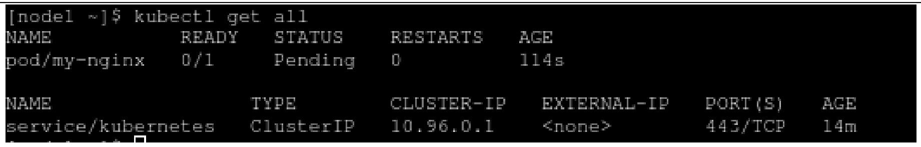
    - includes api server , scheduler, control manager, etcd and more
    - sometimes called the master

```
       MASTER
|=======================|
| etcd                  |
| api                   |
| scheduler             |
| controller manager    |
| core dns              |
| .                     |
| .                     |
| based on need         |
|                       |
| Docker                |
|=======================|

       NODE
|=======================|
| kubelet               |
| kube-proxy            |
| .                     |
| .                     |
| based on need         |
|                       |
|                       |
|                       |
| Docker                |
|=======================|
```

- pod - one or more containers running together on one Node
    - basic unit of deployment, containers are always in pods
- controller - for creating /updating pods and other objects
    - Deployment
    - ReplicaSet
    - StatefulSet
    - DaemonSet
    - Job
    - CronJob
- service - network endpoint to connect to a pod
- namespace - filter group
- secrets, ConfigMaps …

21.5 in play with k8s

- I created 3 instances
- I am going to make node1 as master/ manager node
- Rest of the nodes will be worker nodes
- Main goal is to create deplotyments

Snaps	Description
kubectl get nodes	get nodes connected to the cluster
	starting master node (command already provided with k8s playground)
	getting version (one client and one server)
kubectl run my_nginx –image nginx 	
kubectl get pods 	get pods
kubectl create deployment my-nginx –image nginx  	create deployment
	get all contents
kubectl delete deployment my-nginx	delete the deployment

```
[node1 ~]$ kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10
Initializing machine ID from random generator.
[init] Using Kubernetes version: v1.20.4
[preflight] Running pre-flight checks
        [WARNING Service-Docker]: docker service is not active, please run 'systemctl star
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. T
```

```
[node1 ~]$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", Gi
:"2020-12-18T12:09:25Z", GoVersion:"go1.15.5", Compile
Server Version: version.Info{Major:"1", Minor:"20", Gi
:"2021-02-18T16:03:00Z", GoVersion:"go1.15.8", Compile
```

```
[node1 ~]$ kubectl run my-nginx --image nginx
pod/my-nginx created
```

```
[node1 ~]$ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
my-nginx   0/1     Pending   0          8s
```

```
[node1 ~]$ kubectl create deployment my-nginx --image nginx
deployment.apps/my-nginx created
```

```
[node1 ~]$ kubectl get all
NAME                             READY   STATUS    RESTARTS   AGE
pod/my-nginx                     0/1     Pending   0          21m
pod/my-nginx-6b74b79f57-x4c5m    0/1     Pending   0          7s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   34m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-nginx   0/1     1            0           7s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/my-nginx-6b74b79f57   1         1         0       7s
```

```
[node1 ~]$ kubectl get all
NAME           READY   STATUS    RESTARTS   AGE
pod/my-nginx   0/1     Pending   0          114s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   14m
```

```
Pods --> ReplicaSet --> Deployment
```

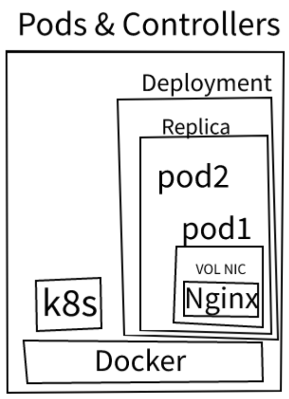
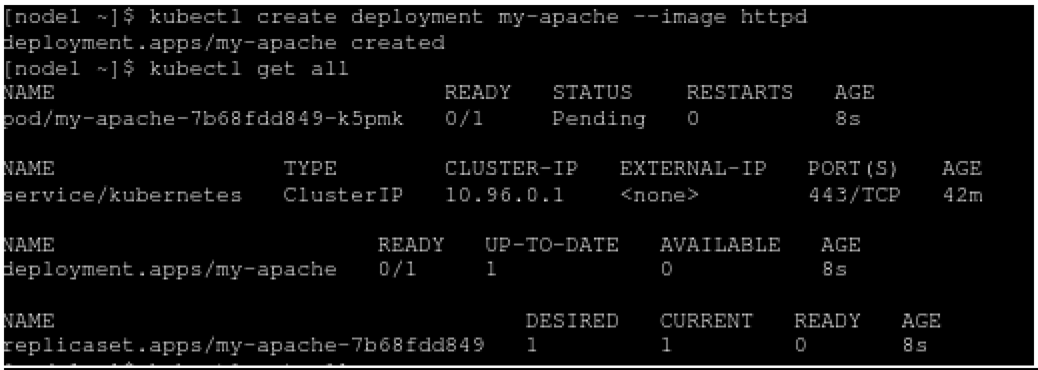
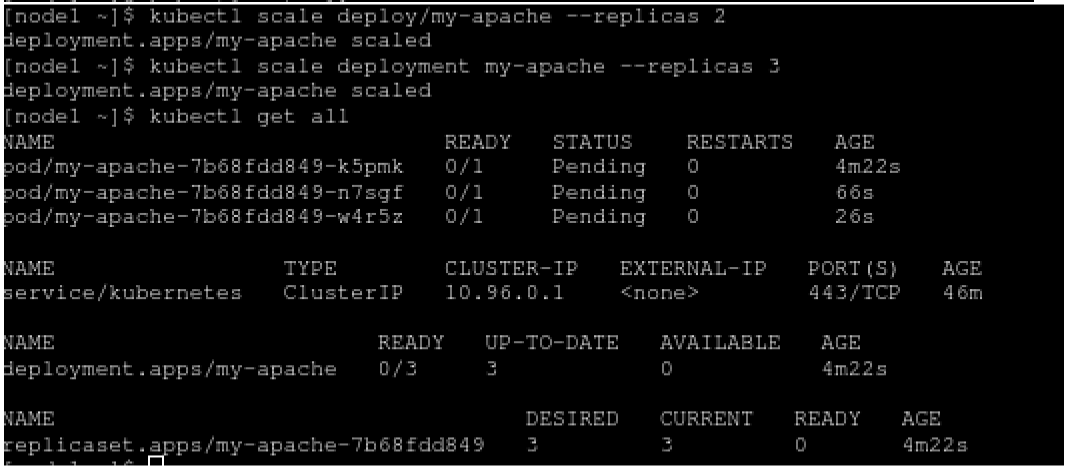


Fig. 1: kube6

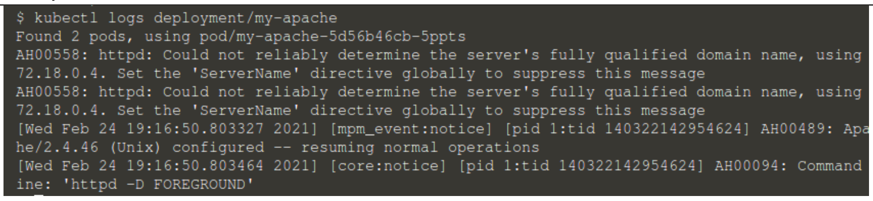
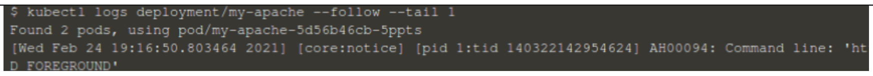
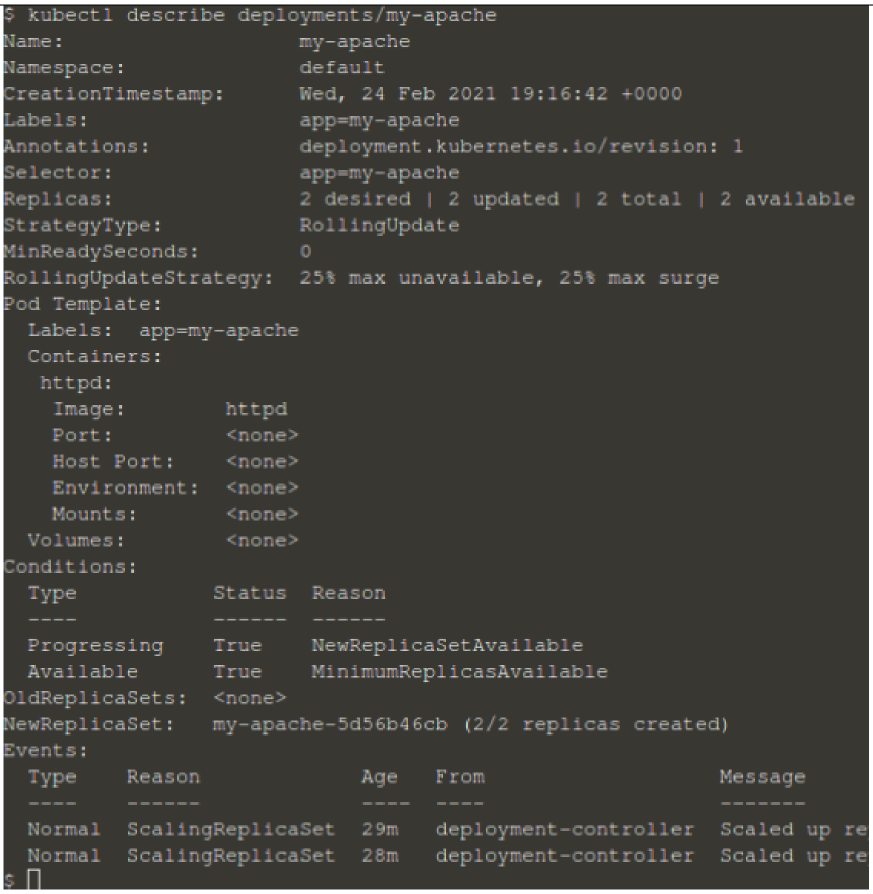
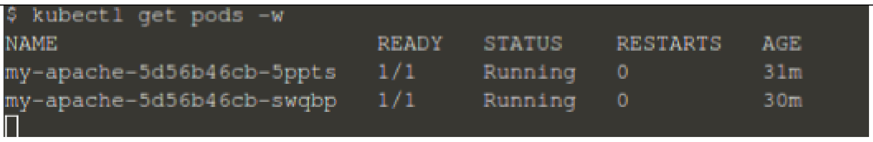
21.6 Scaling ReplicaSets



```
[node1 ~]$ kubectl create deployment my-apache --image httpd
deployment.apps/my-apache created
[node1 ~]$ kubectl get all
NAME                             READY   STATUS    RESTARTS   AGE
pod/my-apache-7b68fdd849-k5pmk   0/1     Pending   0          8s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   42m

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-apache   0/1     1            0           8s

NAME                                   DESIRED   CURRENT   READY   AGE
replicaset.apps/my-apache-7b68fdd849   1         1         0       8s
```



```
[node1 ~]$ kubectl scale deploy/my-apache --replicas 2
deployment.apps/my-apache scaled
[node1 ~]$ kubectl scale deployment my-apache --replicas 3
deployment.apps/my-apache scaled
[node1 ~]$ kubectl get all
NAME                             READY   STATUS    RESTARTS   AGE
pod/my-apache-7b68fdd849-k5pmk   0/1     Pending   0          4m22s
pod/my-apache-7b68fdd849-n7sgf   0/1     Pending   0          66s
pod/my-apache-7b68fdd849-w4r5z   0/1     Pending   0          26s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   46m

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-apache   0/3     3            0           4m22s

NAME                                   DESIRED   CURRENT   READY   AGE
replicaset.apps/my-apache-7b68fdd849   3         3         0       4m22s
```

Snaps	Description
	logs
	logs follow changes and tail last 1 line logs
	describe pod/deployments etc
	watch

```
$ kubectl logs deployment/my-apache
Found 2 pods, using pod/my-apache-5d56b46cb-5ppts
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using
72.18.0.4. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using
72.18.0.4. Set the 'ServerName' directive globally to suppress this message
[Wed Feb 24 19:16:50.803327 2021] [mpm_event:notice] [pid 1:tid 140322142954624] AH00489: Apa
he/2.4.46 (Unix) configured -- resuming normal operations
[Wed Feb 24 19:16:50.803464 2021] [core:notice] [pid 1:tid 140322142954624] AH00094: Command
ine: 'httpd -D FOREGROUND'
```

```
$ kubectl logs deployment/my-apache --follow --tail 1
Found 2 pods, using pod/my-apache-5d56b46cb-5ppts
[Wed Feb 24 19:16:50.803464 2021] [core:notice] [pid 1:tid 140322142954624] AH00094: Command line: 'ht
D FOREGROUND'
```

```
$ kubectl describe deployments/my-apache
Name:                   my-apache
Namespace:              default
CreationTimestamp:      Wed, 24 Feb 2021 19:16:42 +0000
Labels:                 app=my-apache
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=my-apache
Replicas:               2 desired | 2 updated | 2 total | 2 available
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=my-apache
  Containers:
   httpd:
    Image:        httpd
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Progressing    True    NewReplicaSetAvailable
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   my-apache-5d56b46cb (2/2 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  29m   deployment-controller  Scaled up re
  Normal  ScalingReplicaSet  28m   deployment-controller  Scaled up re
$
```

```
$ kubectl get pods -w
NAME                        READY   STATUS    RESTARTS   AGE
my-apache-5d56b46cb-5ppts   1/1     Running   0          31m
my-apache-5d56b46cb-swqbp   1/1     Running   0          30m
```

21.7 Service Types

- kubectl expose creates a service for exisiting pods
- Service is a stable address for pod
- it we want to connect to pod, we need a service
- CoreDNS allows us to resolve `services` by name
- Types of services :
  1. ClusterIP
  2. NodePort
  3. LoadBalancer
  4. ExternalName

21.8 ClusterIP (default)

- Single, Internal Virtual IP allocation
- Reachable within the cluster
- pods can reach service on port number

21.9 NodePort

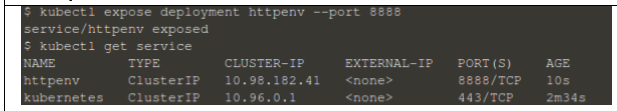
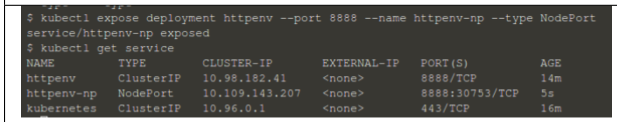
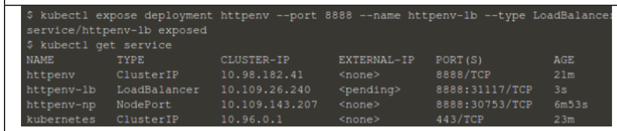
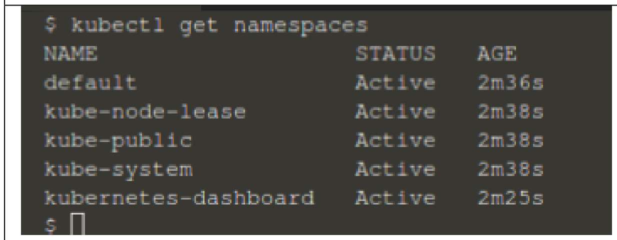
- High port on each node
- Outside the cluster
- port is open for every node's IP
- Anyone can reach node can connect

21.10 LoadBalancer

- Controls a Load Balancer external to the cluster
- Only available when infrastructure providers gives it (AWS ELB etc)
- Create NodePort+ClusterIP, connect LB to NodePort to send

21.11 ExternalName

- Add CNAME DNS record to CoreDNS only
- Not used for pods , but for giving pods a DNS name that can be used outside Kubernetes cluster.

Snaps	Description
	create service expose port with cluster IP
	create service NodePort. different than docker as left port if internal port and right one is node port for outside cluster
	create service with LoadBalancer
	namespaces

```
$ kubectl expose deployment httpenv --port 8888
service/httpenv exposed
$ kubectl get service
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
httpenv      ClusterIP   10.98.182.41   <none>        8888/TCP   10s
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    2m34s
```

```
$ kubectl expose deployment httpenv --port 8888 --name httpenv-np --type NodePort
service/httpenv-np exposed
$ kubectl get service
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
httpenv      ClusterIP   10.98.182.41    <none>        8888/TCP         14m
httpenv-np   NodePort    10.109.143.207  <none>        8888:30753/TCP   5s
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          16m
```

```
$ kubectl expose deployment httpenv --port 8888 --name httpenv-lb --type LoadBalance
service/httpenv-lb exposed
$ kubectl get service
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
httpenv      ClusterIP      10.98.182.41     <none>        8888/TCP         21m
httpenv-lb   LoadBalancer   10.109.26.240    <pending>     8888:31117/TCP   3s
httpenv-np   NodePort       10.109.143.207   <none>        8888:30753/TCP   6m53s
kubernetes   ClusterIP      10.96.0.1        <none>        443/TCP          23m
```

```
$ kubectl get namespaces
NAME                   STATUS   AGE
default                Active   2m36s
kube-node-lease        Active   2m38s
kube-public            Active   2m38s
kube-system            Active   2m38s
kubernetes-dashboard   Active   2m25s
$ 
```

CHAPTER TWENTYTWO

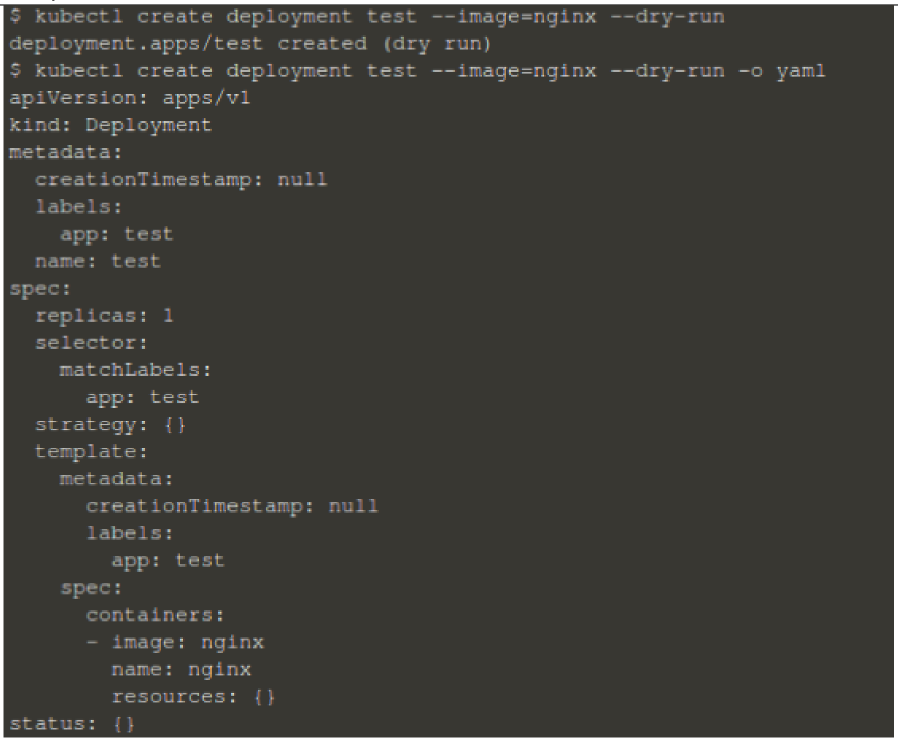
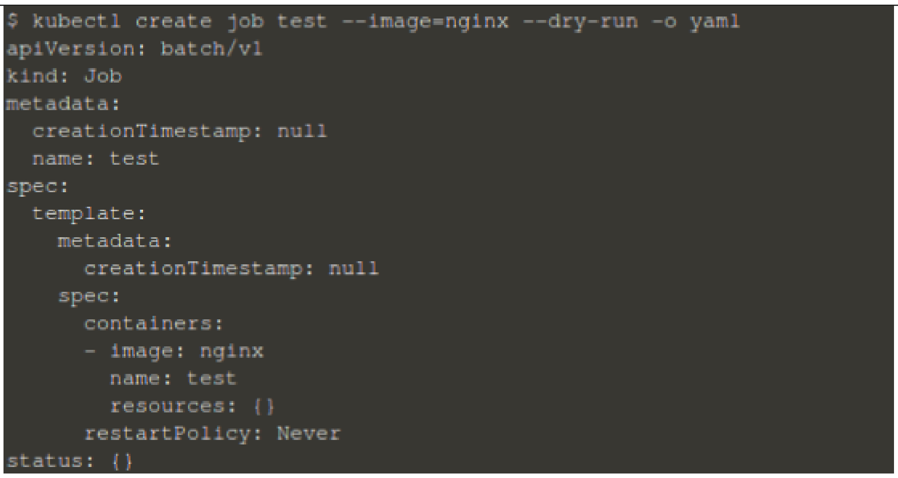
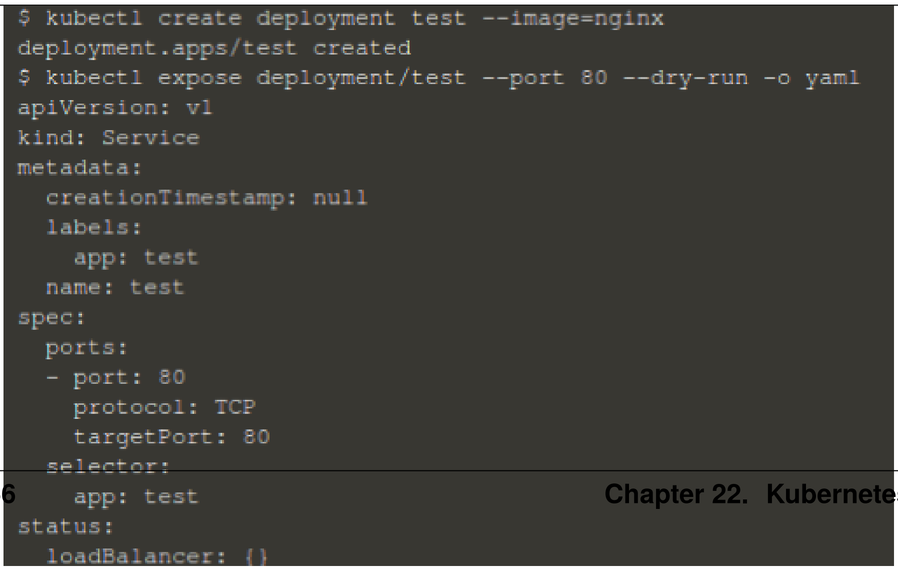
KUBERNETES MANAGEMENT TECHNIQUES

22.1 Generators (Automation behind commands)

- Helper templates
- Every resource in kubernetes has a ‘spec’ or specification

```
> kubectl create deployment smaple --iamge nginx --dry-run -o yaml
```

- output those templates `--dry-run -o yaml`
- these yaml defaults can be a starting points to create new ones

Snaps	Description
	Get Generator info for deployemnt
	Get Generator info for job
	Get Generator info for expose

```
$ kubectl create deployment test --image=nginx --dry-run
deployment.apps/test created (dry run)
$ kubectl create deployment test --image=nginx --dry-run -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: test
  name: test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: test
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

```
$ kubectl create job test --image=nginx --dry-run -o yaml
apiVersion: batch/v1
kind: Job
metadata:
  creationTimestamp: null
  name: test
spec:
  template:
    metadata:
      creationTimestamp: null
    spec:
      containers:
      - image: nginx
        name: test
        resources: {}
      restartPolicy: Never
status: {}
```

```
$ kubectl create deployment test --image=nginx
deployment.apps/test created
$ kubectl expose deployment/test --port 80 --dry-run -o yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: test
  name: test
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: test
status:
  loadBalancer: {}
```

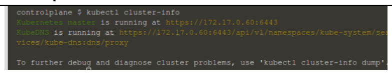
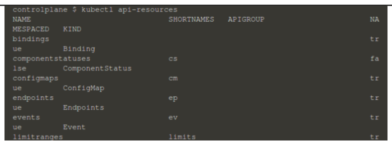
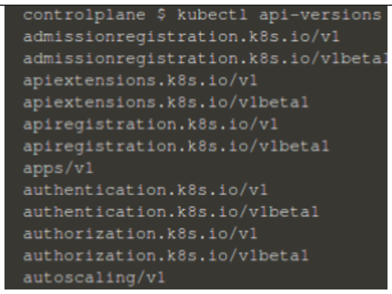
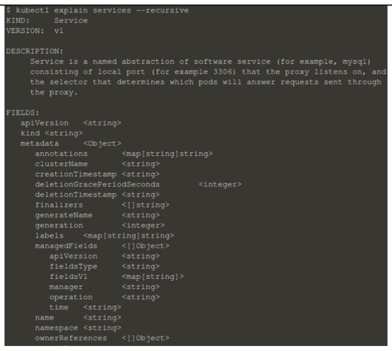
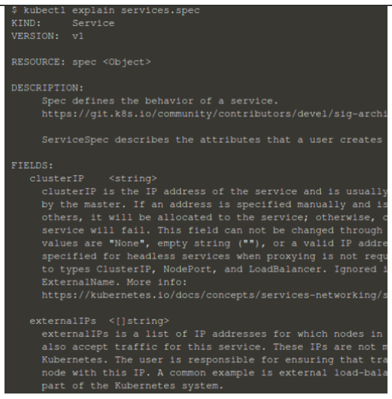
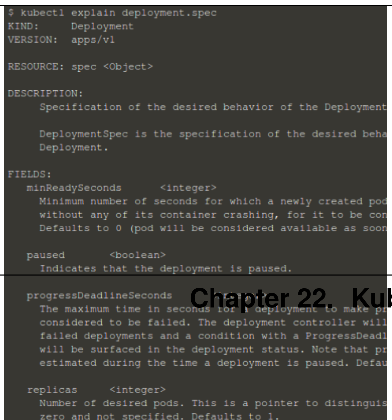
Imperative	Decalarative
how program operates	what a program should accomplish
ex.- making your own coffee	ex.- give instructions to a barista
not easy to automate	automation is good
know every step	dont know current state, only final result is known
•	requires to know all yaml keys

22.2 Management approaches

- Imperative commands
  - create, expose, edit, scale etc
- Imperative objects
  - create -f file.yml , replace -f file.yml
- Declarative objects
  - apply -f file.yml

22.3 Kubernetes Configuration YAML

- Each file contains one or more configuration files
- Each manifest describes an API object (deployment, job, secret)
- Each mainfest needs these four parts-
  - apiVersion:
  - kind:
  - metadata:
  - spec:
- `kubectl apply -f <directory>/`
- selectors is used for patternmatching for different services

info	Snaps	Description
cluster		cluster info
`kind`		api resources (kind will give info for yaml file)
`apiVersion`		api versions
`metadata`	•	only `name` of the service is required
`spec`	•	all the action
explain services recursively		explain services get keywords
explain services description		explain services get keywords
explain deployments description		explain services get keywords

```
controlplane $ kubectl cluster-info
Kubernetes master is running at https://172.17.0.60:6443
KubeDNS is running at https://172.17.0.60:6443/api/v1/namespaces/kube-system/ser
vices/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

```
controlplane $ kubectl api-resources
NAME                              SHORTNAMES   APIGROUP                   NA
MESPACED   KIND
bindings                                                                  tr
ue         Binding
componentstatuses                 cs                                      fa
lse        ComponentStatus
configmaps                        cm                                      tr
ue         ConfigMap
endpoints                         ep                                      tr
ue         Endpoints
events                            ev                                      tr
ue         Event
limitranges                       limits                                  tr
```

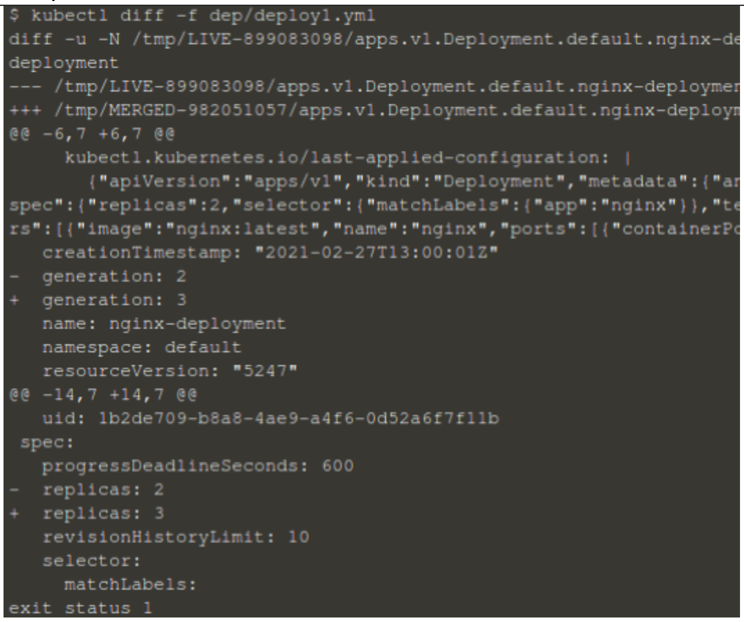
```
controlplane $ kubectl api-versions
admissionregistration.k8s.io/v1
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
apps/v1
authentication.k8s.io/v1
authentication.k8s.io/v1beta1
authorization.k8s.io/v1
authorization.k8s.io/v1beta1
autoscaling/v1
```

```
$ kubectl explain services --recursive
KIND:     Service
VERSION:  v1

DESCRIPTION:
     Service is a named abstraction of software service (for example, mysql)
     consisting of local port (for example 3306) that the proxy listens on, and
     the selector that determines which pods will answer requests sent through
     the proxy.

FIELDS:
   apiVersion   <string>
   kind <string>
   metadata     <Object>
      annotations       <map[string]string>
      clusterName       <string>
      creationTimestamp <string>
      deletionGracePeriodSeconds        <integer>
      deletionTimestamp <string>
      finalizers        <[]string>
      generateName      <string>
      generation        <integer>
      labels    <map[string]string>
      managedFields     <[]Object>
         apiVersion     <string>
         fieldsType     <string>
         fieldsV1       <map[string]>
         manager        <string>
         operation      <string>
         time   <string>
      name      <string>
      namespace <string>
      ownerReferences   <[]Object>
```

```
$ kubectl explain services.spec
KIND:     Service
VERSION:  v1

RESOURCE: spec <Object>

DESCRIPTION:
     Spec defines the behavior of a service.
     https://git.k8s.io/community/contributors/devel/sig-archi

     ServiceSpec describes the attributes that a user creates

FIELDS:
   clusterIP    <string>
     clusterIP is the IP address of the service and is usually
     by the master. If an address is specified manually and is
     others, it will be allocated to the service; otherwise, c
     service will fail. This field can not be changed through
     values are "None", empty string (""), or a valid IP addre
     specified for headless services when proxying is not requ
     to types ClusterIP, NodePort, and LoadBalancer. Ignored i
     ExternalName. More info:
     https://kubernetes.io/docs/concepts/services-networking/s

   externalIPs  <[]string>
     externalIPs is a list of IP addresses for which nodes in
     also accept traffic for this service. These IPs are not m
     Kubernetes. The user is responsible for ensuring that tra
     node with this IP. A common example is external load-bala
     part of the Kubernetes system.
```

```
$ kubectl explain deployment.spec
KIND:     Deployment
VERSION:  apps/v1

RESOURCE: spec <Object>

DESCRIPTION:
     Specification of the desired behavior of the Deployment

     DeploymentSpec is the specification of the desired beha
     Deployment.

FIELDS:
   minReadySeconds      <integer>
     Minimum number of seconds for which a newly created pod
     without any of its container crashing, for it to be con
     Defaults to 0 (pod will be considered available as soon

   paused       <boolean>
     Indicates that the deployment is paused.

   progressDeadlineSeconds      <integer>
     The maximum time in seconds for a deployment to make pr
     considered to be failed. The deployment controller will
     failed deployments and a condition with a ProgressDeadl
     will be surfaced in the deployment status. Note that pr
     estimated during the time a deployment is paused. Defau

   replicas     <integer>
     Number of desired pods. This is a pointer to distinguis
     zero and not specified. Defaults to 1.
```

- https://kubernetes.io/docs/reference/#api-reference

Snaps	Description
	find the difference between running service and updated yml

```
$ kubectl diff -f dep/deploy1.yml
diff -u -N /tmp/LIVE-899083098/apps.v1.Deployment.default.nginx-de
deployment
--- /tmp/LIVE-899083098/apps.v1.Deployment.default.nginx-deploymen
+++ /tmp/MERGED-982051057/apps.v1.Deployment.default.nginx-deploym
@@ -6,7 +6,7 @@
     kubectl.kubernetes.io/last-applied-configuration: |
       {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"ar
spec":{"replicas":2,"selector":{"matchLabels":{"app":"nginx"}},"te
rs":[{"image":"nginx:latest","name":"nginx","ports":[{"containerPo
   creationTimestamp: "2021-02-27T13:00:01Z"
-  generation: 2
+  generation: 3
   name: nginx-deployment
   namespace: default
   resourceVersion: "5247"
@@ -14,7 +14,7 @@
   uid: 1b2de709-b8a8-4ae9-a4f6-0d52a6f7f11b
 spec:
   progressDeadlineSeconds: 600
-  replicas: 2
+  replicas: 3
   revisionHistoryLimit: 10
   selector:
     matchLabels:
exit status 1
```

22.4 Labels and Annotations

- labels under metadata
- for grouping, filtering etc.
- examples - tier: frontend, app: api, env: prod etc.(There are no specific standards to do so, it depends on the team you are working in)
- no meant to hold complex or large information, instead of `label` use `annotaions.`
- filter on label used in a get
  - `kubectl get pods -l app=nginx`
- apply commands only for matching labels
  - `kubectl apply -f some_file.yaml -l app=nginx`

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  minReadySeconds: 5
  template:
    metadata:
```

(continues on next page)

(continued from previous page)

```
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

22.4.1 Label Selectors

- Indicators to services and deployments, which pods are theirs to pick up.

in above example the resources are going to match labels from selectors to classify nodes and apply things.

22.5 Storage in K8s

```
Initial idea behind containers to be immutable, distributed and replaceable (in hindsight
statefulness came later on as feature to have something stored to be used if container
instance changes like database)
```

- we can create VOLUME similar to docker swarm
- **2 types**
  - Volumes
    - Tied to lifecycle of a pod
    - All containers in a pod can share them
  - Persistent Volumes
    - Created at cluster level, outlives a Pod
    - Sep storage config from pod
    - multiple pods can share them
- **CSI (Container Storage Interface)** plugins from different vendors to connect to storage to have uniformity.

22.6 Ingress Controller

- Lets talk about http
- How do we route outside connections based on hostname or url?
- `ingress controller is the way to do it`.
- Ingress controller is the way to differenciate different routes(considering all of them are using 80 or 443) hosted in a cluster.
- It is not inherently installed in k8s.
- Nginx is a populer one, but other examples are Taefik, HAProxy, etc.

- Implemention is specific to controller chosen.

22.7 Custom resources

Reference

Simply just additional API extensions that are not default in k8s but they can be part of k8s functionality once added.

22.8 Higher Deployment Abstractions

- We have yaml files/ configurations, but how to use them for deployment.
- `Helm` is the most populer one to do so. Helm is to k8s, what k8s is to containers. yaml templates.
- `Compose on k8s` comes with docker desktop. Instead of going to docker stack it will ask for k8s deployment (need to try this out).
- most distros support Helm.

`New things CNAB and docker app`

22.9 Namespaces

```
user@user~/$ kubectl get namespaces
user@user~/$ kubectl get all --all-namespaces
user@user~/$ kubectl config get-contexts
```

22.10 Docker Security

Reference

https://docs.docker.com/engine/security/

https://sysdig.com/blog/20-docker-security-tools/

22.11 Docker Bench Sceurity

https://github.com/docker/docker-bench-security

> in a bunch of docker official images available online, there are users created `groupadd & useradd`. Our job while using those images is use the user mentioned and not run the image with root previleges.

```
WORKDIR /app
USER <user_name>
```

CHAPTER TWENTYTHREE

INDICES AND TABLES

- genindex
- modindex
- search