Docker + Kubernete

Wednesday, 16 March 2022

10:50 AM

**Impt Links:**

Deployment YAML: [https://pastebin.com/rZa9Dm1w](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbXlkVUVnN0hCMDVCZjlPUkZjY0JJOWV3YkwwZ3xBQ3Jtc0trZTVXbzE2SDBQa09mNDVST3hBWUl5QjUzMVRabTJ1OWVoMHZUSDRJdkxhczU3TU53bDlRdFJJX2VYdU1lU010MTJ6ZFdDa0Jrd2p3V2VfMGtxYk55X3g1NG1XTTd1OVgxMG9jbFk5Y0JBNzZEZDkxcw&q=https%3A%2F%2Fpastebin.com%2FrZa9Dm1w&v=1xo-0gCVhTU)

Dockerfile: [https://pastebin.com/SZA26rbg](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbmRyY2pTRHlVTllNN3hJam10c3k5ak15cGJKQXxBQ3Jtc0ttNmktb1A2eUEyTm5Wb25ubmFER0llT185UmNqS050NGdOQk40cmg4dFo3RVhieXpWcC10OUhDTWRkaGluVVlIejk4dEFsNDNUZEtBc0ZmTVRVekkwUk5mMG1od3ZmVjFvRG9mM2R2eXBaQjljUklhWQ&q=https%3A%2F%2Fpastebin.com%2FSZA26rbg&v=1xo-0gCVhTU)

How to Containerize a Node App: <https://nodejs.org/en/docs/guides/nodejs-docker-webapp/>

Package-lock Blog Post: <https://medium.com/coinmonks/everything-you-wanted-to-know-about-package-lock-json-b81911aa8ab8>

History :

Earlier , people used to deploy application on some dedicated server that were very powerful and were specially used for deployment, good throughput and scalability. Virtual machine was created for the dedicated software and required OS , libs and tools would be installed. --when had dedicated server , what was the need of virtual machine

But the issue with this type of approach was that all resources of the machine was used/dedicated for only one application.

Hence , a new concept i.e Docker was introduced on container concept. Earlier also, we used to have container concept inside linux but it wasn't highlighted that much and docker was build on this concept only.

Inside docker , we don't need separate OS for each application , instead we can use single OS and even libs and tools installed can be shared between different containers.

Now for working inside docker , we need to create a container that will have an image . **Image** is nothing but the code you want to run . The code can be your application code or your DB code. Your code is packed using docker file and once packed can be run inside any environment. The image is created using Docker File which has some kind of properties. **Dockerfile** isn't used to run your application , instead it is some kind of definition about your prog. Once you create the dockerfile, you can build an image from it and then that image can be run on any comp having docker.

For eg : if UserA and UserB needs to install git on their system , UserA can follow all the steps to install the git on his system and then can share an image of this to UserB . This way UserB will never be required to follow all the steps again.

This image created needs to be put inside Docker **Registry**. This registry can be public or private . Many public repo are present inside docker but due to security concerns, organizations focus on creating their own private repo which can be created by registering yourself on Docker i.e creating an account with them. We can put multiple image inside docker registry. Also, for using public repository , we need to pull them in our account.

Diff between registry and container

Main diff between traditional deployment and docker :

Traditional deployment used to be done on one server using single VM (Hypervisor) but kubernete still use hypervisor

But in docker we use concept of multiple container and for this container runtime is required (ContainerD is provided by docker , we have few others too). We need to install them

But Now I want to scale my application , so we need orchestration i.e properly managing the replicas and clusters , proper load balancing . We have few choices for this :

1. Docker Swam
2. Kubernetes (more efficient , use docker internally)
3. Azure Container Service
4. Amazon Ecs
5. Marathon
6. Google container engine

Que : where image is present , inside container or repository ? Diff between hypervisor and docker runtime

Machine generated alternative text:
Docker 
is an open platform for developers and sysadmins to 
build, ship, and run distributed applications, whether on 
laptops, data center VMS, or the cloud. 

Sometimes when u run an application on diff machine , u may get missing lib or component error. But if you have docker and the application is deployed on docker, then there is no need to worry abt missing lib and other stuff bcz u have an image with u and u know this image is fully tested. All u need is docker and everything will run at ur end too. All u need is a container for deployment on docker.

**Container** is not a virtual machine. Share same principle but there is a huge diff.

With docker , containers can share bins and libs. Eg if u need node and express for two projects deployed on docker inside diff containers then they can share same express and node instead of installing node and express separately on each container. No need of guest operating sys as well, therefore really cuts out size.

Machine generated alternative text:
Docker 
"Containers are a way to package software in a format that can run isolated on a 
shared operating system. Unlike VMS, containers do not bundle a full operating 
system - only libraries and settings required to make the software work are needed. 
This makes for efficient, lightweight, self-contained systems and guarantees that 
software will always run the same, regardless of where it's deployed." 

Machine generated alternative text:
Docker vs VM 
App A 
Bins/Libs 
Guest OS 
VM 
App B 
Bins/Libs 
Guest OS 
Hypervisor 
App C 
Bins/Libs 
Guest OS 
App A 
Bins/Libs 
CONTAINER 
App B 
Bins/Libs 
Docker 
Host OS 
App C 
Bins/Libs 

**Docker Image** :

A Docker image is **a read-only template that contains a set of instructions for creating a container that can run on the Docker platform**. It provides a convenient way to package up applications and preconfigured server environments, which you can use for your own private use or share publicly with other Docker users.

Read from here :<https://jfrog.com/knowledge-base/a-beginners-guide-to-understanding-and-building-docker-images/#:~:text=A%20Docker%20image%20is%20a,publicly%20with%20other%20Docker%20users>.

Que : what is context and docker build context ? :

 my-nginx:0.1 . Where is this present ???

Machine generated alternative text:
Kubernetes 
is an open-source system for automating 
deployment, scaling, and management of 
containerized applications. 

Helps to set the application across diff machines. i.e earlier we had to set application on diff servers manually and if some error occurred, have to manually login into that machine to eliminate it (has master slave concept) . However, Kubernetes allows to handle multiple machine from one machine i.e used for distributed application.

[Introduction to Microservices, Docker, and Kubernetes](https://www.youtube.com/watch?v=1xo-0gCVhTU)

**DOCKER COMMANDS + KUBERNETES :**

1. To chec if docker is installed :

$docker

1. To check all running pods :

$sudo docker ps (can remove sudo )

1. To kill docker pod :

$sudo docker kill p\_o\_d\_Id

1. To run any application in docker :

$sudo docker run YourApplicationName

1. How to bind a port from your machine to a port withing a docker application

Eg: port 3000 on ur machine should be bind to port 80 of docker, so that if you run <http://localhost:3000/> ,

It should redirect to port 80 of docker container.

When u open the browser on port 3000, it gives container id as well i.e My hostname is wdfbb7643 :

$sudo docker run -p 3000:80 tutum/hello-world

where tutum is container and hello-world is the application

1. If you want to run in the background, i.e no need keep running your terminal then put -d flag (detach mode) :

$sudo docker run -d -p 3000:80 tutum/hello-world

You can do $sudo docker ps to check the status of newly created container

1. How to run same application on diff ports i.e copies :

$sudo docker run -d -p 3000:80 tutum/hello-world

$sudo docker run -d -p 3001:80 tutum/hello-world

$sudo docker run -d -p 3002:80 tutum/hello-world

Do we create container as well or just instances

1. Deploy the above created container on Kubernatives

(Need to have MiniKube installed , which is a mini localhost version of Kubernetes used for testing purpose on local)

+ Kubectl (comman line tool)

Now, bootup miniKube and create one cluster for you:

a) $minikube start

Minikube runs on a VM Docker and not inside a VM

b) Meanwhile, we can write a deployment file

Machine generated alternative text:
apiVersion: extensions/vlbetal 
kind: Deployment 
metadata : 
name: hello-world 
spec : 
replicas: 5 
template: 
metadata: 
labels: 
app: hello-world 
spec : 
containers: 
- name: hello-world 
image: tutum/hello-world 
ports: 
- contaErport: 80 

Metadata: name is name used for the 'kind' declared here

Next , you need to declare service under same file

Service used to access container/pods outside of cluster

Machine generated alternative text:
deployment.yml X 
kind: Service 
apiversion: VI 
netadata: 
name: helloworldservice 
spec : 
selector: 
app: hello-K)rld 
ports: 
protocol: "rcp• 
Port accessible inside cluster 
port: 8080 
Port to forward to inside the pod 
targetPort: 80 
z port accessible outside cluster 
nodeport: 
type: Load3a1ancer 

'type' is loadbalancer , i.e this service will be acting as a loadbalancer between 5 instances.

1. Once minikube has started , you need to use kubectl to interact
2. To get running pods details:

$sudo kubectl get pods

To get running deployments:

$sudo kubectl get deployments

To get minicube dashboard/GUI application to interact with your clusters

You can craete deployemnts n pods from here or from cmd

$kubectl dashboard

ctrl+ C to come out

To change path :

cd Desktop/kubernetes/

ls >>> gives 2 files : Deployment.yml DockerFile : you created

To create a new resource from yml file (-f for file):

$sudo kubectl create -f deployment.yml

If you already have a resource , change 'create' to 'apply'

Machine generated alternative text:
cd Desktop/kubernetes/ 
[ j ames] [N/ Desktop/kubernetes] 
Is 
deployment. yml Dockerfile 
r [ james] [4 Desktop/kubernetes] 
sudo kubectl create -f deployment.yml 
service "helloworldservice" created 
deployment "hello-world" created 
L—• sudo kubectl get pods 
NAME 
hello-world 
hello-world 
hello-world 
hello-world 
h 
c! 
11 
READY 
1/1 
1/1 
1/1 
1/1 
STATUS 
Running 
Running 
Running 
Running 
RESTARTS 
-84fc897894-b66sg 
-84fc897894-pt6rb 
-84fc897894-sm1r6 
-84fc897894-w84fx 
AGE 
6s 

Inside our deployment.yml , we had services first and then deployment , so service is created first

Now you can check you pod using :

$sudo kubectl get pods >>> you will get 5 pods instances

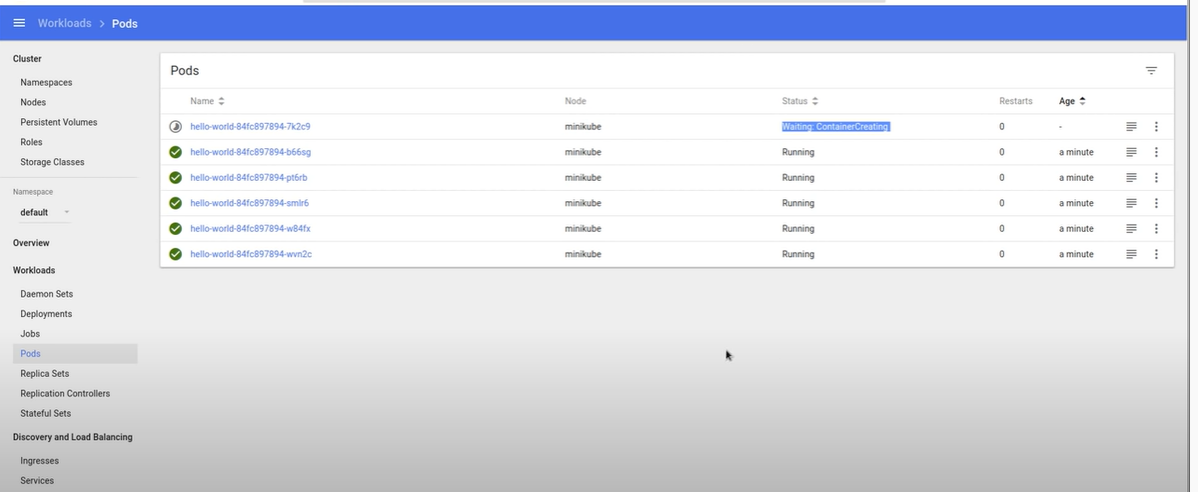
To check deployment:

$sudo kubectl get deployments >> it will give u info of deployment

Machine generated alternative text:
[ james] [ — [Desktop/kubernetes] 
sudo kubectl get pods 
10 - world - 84fc897894-b66sg 
- 84fc897894-pt6rb 
*let 10 -world -84fc897894-smIr6 
- 84fc897894-wvn2c 
READY 
1/1 
1/1 
1/1 
1/1 
1/1 
STATUS 
Running 
Running 
Running 
Running 
Running 
RESTARTS 
AGE 
6s 
6s 
6s 
6s 
6s 
[ j ames] [4 Desktop/ kubernetes] 
sudo kubectl get deployments 
gAME 
DESIRED CURRENT UP-TO-DATE 
5 
5 
5 
AVAILABLE 
5 
AGE 
20s 
[ j ames] [N/Desktop/kubernetes] 
minikube 

Now open dashboard , you will see 5 pods instances running , try to crash/delete one of them.

You will see, other 4 instances are running + kubernetes will try to restart the crashed pod



[1]+ Done 
minikube 
sudo kubectl get pods 
NAME 
hello-world-84fc897894- 7k2c9 
hel to-world -84 fc897894- b66sg 
hello-world-84fc897894-pt6rb 
to-world -84fc897894-smIr6 
hel 10 -world -84fc897894-w84fx 
-84 fc897894-wvn2c 
READY 
1/1 
1/1 
1/1 
1/1 
1/1 
dashboard 
STATUS 
Runnin 
erminatine 
Running 
Running 
Running 
Running 
RESTARTS 
AGE 
ISS 
1m 
1m 
1m 
1m 
1m 

Gives 6 instances now, one of them is terminated by us , and one is newly added by kubernete to recover the deleted one.

**Load balancing** :

1. To get ip address on which minikube is running your cluster :

$ minikube ip

>> 192.168.99.100

So if you browse : <http://192.168.99.100:30001/> (you will get your running cluster , and this port was defined in deployment.yml)

n tutum 
Hello world! 
My hostname is 
n a 0.0. 

whenever you will refresh the page , you will see every time a new pod is loaded out of 5 available pods . This is coz of load balancing.

Whichever is available in pool comes through.

**Scaling** :

>> To handle multiple users. Say your site is heavily loaded from 9 a.m to 10 a.m as most of the users login at that time. So, you need more copies at that particular time for better load balancing . So, open the dashboard >> Deployment >> click on three vertical dots present on your deployment row >> Scale a Deployment >> change copies from 5 to 10

But in video it was done manually .

When u don't need those copies, just remove them

**Containerize a node app** :

1. Create a node project
2. Create it's docker file

(there is one link from where you can get pre defined docker files for specified version of node) : check video at- 40.31 [Introduction to Microservices, Docker, and Kubernetes](https://www.youtube.com/watch?v=1xo-0gCVhTU)

Impt Links:

Deployment YAML: [https://pastebin.com/rZa9Dm1w](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbXlkVUVnN0hCMDVCZjlPUkZjY0JJOWV3YkwwZ3xBQ3Jtc0trZTVXbzE2SDBQa09mNDVST3hBWUl5QjUzMVRabTJ1OWVoMHZUSDRJdkxhczU3TU53bDlRdFJJX2VYdU1lU010MTJ6ZFdDa0Jrd2p3V2VfMGtxYk55X3g1NG1XTTd1OVgxMG9jbFk5Y0JBNzZEZDkxcw&q=https%3A%2F%2Fpastebin.com%2FrZa9Dm1w&v=1xo-0gCVhTU)

Dockerfile: [https://pastebin.com/SZA26rbg](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbmRyY2pTRHlVTllNN3hJam10c3k5ak15cGJKQXxBQ3Jtc0ttNmktb1A2eUEyTm5Wb25ubmFER0llT185UmNqS050NGdOQk40cmg4dFo3RVhieXpWcC10OUhDTWRkaGluVVlIejk4dEFsNDNUZEtBc0ZmTVRVekkwUk5mMG1od3ZmVjFvRG9mM2R2eXBaQjljUklhWQ&q=https%3A%2F%2Fpastebin.com%2FSZA26rbg&v=1xo-0gCVhTU)

How to Containerize a Node App: <https://nodejs.org/en/docs/guides/nodejs-docker-webapp/>

Package-lock Blog Post: <https://medium.com/coinmonks/everything-you-wanted-to-know-about-package-lock-json-b81911aa8ab8>

EXP"'" 
OPEN 
@ package.json 
index.js 
• Dockerfile 
4 deDlOvment.vml 
containers 
Dockerf"e . 
ROY node ; 
Create ago directory 
WORKDIR /usr/src/app I 
copy package. json . 
COPY package - j son 
instau 
EXPOSE 
CYO 
start' 

1. Note : Don't include each and everything in your image as it will only make your image bigger . For eg: don't include 'node\_module' . For this, you need to write .**dockerignore**

• • Dockertile 
.doaerignore 
DO' k erfile 

1. Now, once you have created everything , you need to **build** it (-t is target)

docker build . -t <your username>/node-web-app

For eg : for our project :

$sudo docker build -t jamesquinseley/exampleapp:v1.0.0 . (the last dot is the path to the directy , since we are present in current directory , so only . Will suffice)

1. Run the image :

Running your image with -d runs the container in detached mode, leaving the container 
running in the background. The -p flag redirects a public port to a private port inside the 
container. Run the image you previously built: 
docker run -p 49160: 808Ø -d <your username>/node-web-app 

$sudo docker run -d -p 8080:8080 jamesquinsley/exampleapp:v1.0.0

After this your application will be up and running on port 8080

**Now let's run it on Kubernetes**:

All we need to do is to change our deployment.yml a little bit i.e since we are using previously created yml , we need to change service /deployment name

1. Service >> change name from 'helloworldservice' to 'exampleservice'
2. Service >> app to 'myapp'
3. Service >> port : 808o
4. Service >> ports >> nodePort : 30002 since 30001 is already taken
5. Same goes for deployment too, check video at 49:00 for it

Machine generated alternative text:
pa ckage.json 
indexjs 
Dockerfile 
deployment.yml • 
kind: Service 
apiversion: VI 
metadata: 
name: exampleservice 
spec : 
selector: 
app: myapp 
ports: 
- protocol: "TCP" 
Port accessible inside cluster 
port: 8981 
Port to forward to inside the pod 
targetport: 8883 
$$ Port accessible outside cluster 
nodeport: 300011 1 
type: Load3a1ancer 

Machine generated alternative text:
package.json 
index-js 
to 
Dockerfile 
inside the pod 
deployment.yml x 
port: 8381 
Port to forward 
targetport: 8380 
port accessible 
nodeport: 38882 
type: Load3a1ancer 
outside cluster 
24 
apiversion: extensions/vlbetal 
kind: Deployment 
metadata: 
name: myappdeployment 
spec : 
replicas: 5 
template: 
metadata: 
labels: 
app: myapp 
spec : 
containers: 
name: myapp 
image: jamesquigley/exampleapp 
por t S : 
- containerport: 88801 

Specify exact image name here

Machine generated alternative text:
metadata: 
labels: 
app: myapp 
spec : 
containers: 
name: myapp 
image: jamesquigley/exampleapp: 
por t S : 
- containerport: 

1. Now open terminal /cmd and navigate to project folder and

$sudo kubectl create -f deployment.yml

And open dashboard too

You won't see anything coming up , it's because we created and build the image locally but never pushed it

$sudo docker push jamesquinsley/exampleapp:v1.0.0

Now check dahsboard

If you want to redploy , use '**apply**' instead of '**create**'

So now to check our apps running on minikube :

We have 2 apps :

Tutum on port 30001

Node Appln on port 30002

And we need ip so,

$ minikube ip

>> 192.168.99.100

<http://192.168.99.100:30001/> >> tulum appln

<http://192.168.99.100:30002/> >> Node appln