

Kubernetes Theory and Complete Project

What is Kubernetes:

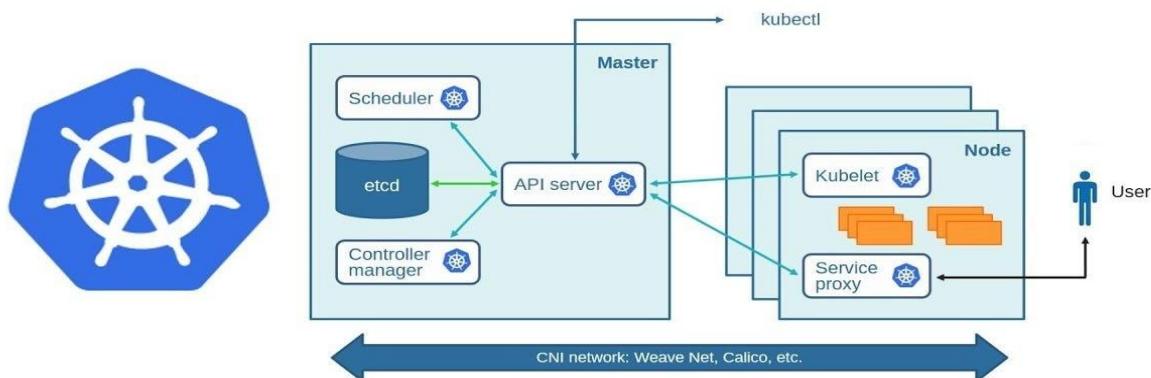
- An open source container orchestration tool used to manage multi container deployment.
- Why kubernetes if we already have Docker Compose, Docker Swarm-
- For production ready deployment of microservices.
- Having less failures and downtimes.
- Good backup and restores strategy.

Microservices vs. Monolithic Architecture:

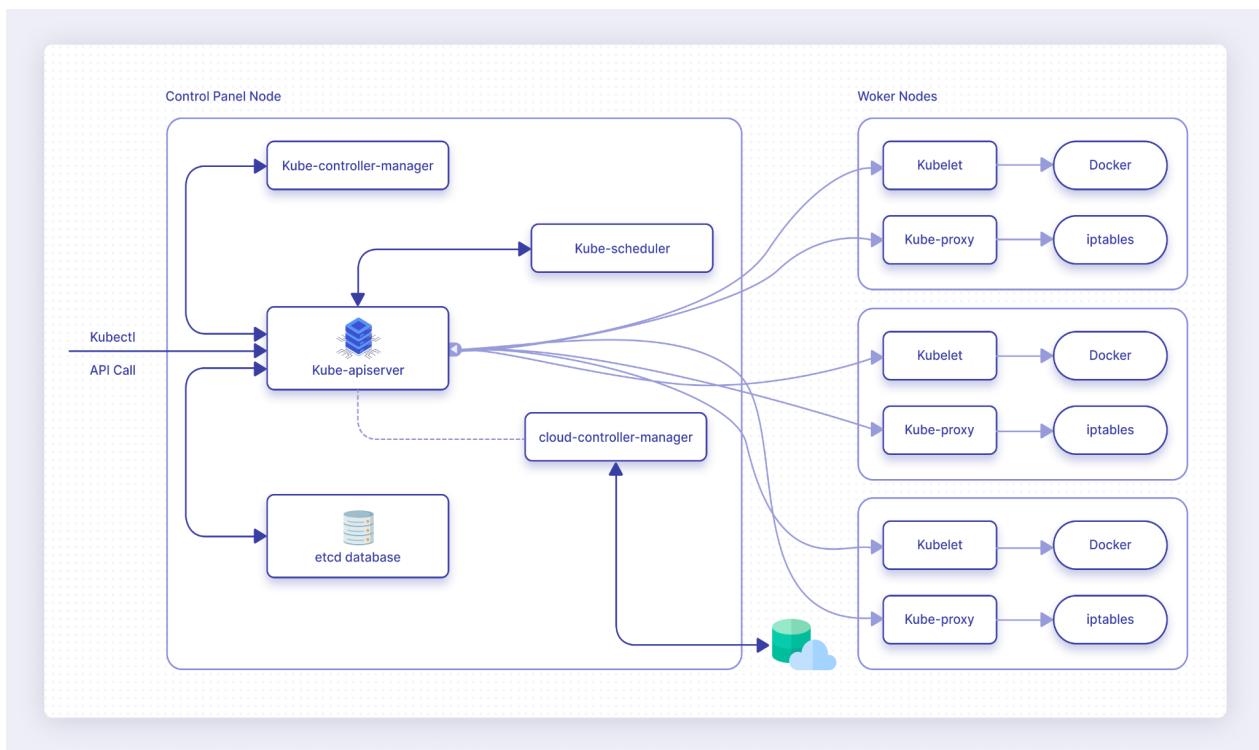
- If all the functionalities of a project exist in a single codebase, then that application is known as a monolithic application. We design our application in various layers like presentation, service, and persistence and then deploy that codebase as a single jar/war file. This is nothing but a monolithic application, where “**mono**” represents the single codebase containing all the required functionalities.
- It is an architectural development style in which the application is made up of smaller services that handle a small portion of the functionality and data by communicating with each other directly using lightweight protocols like HTTP. According to Sam Newman, “Microservices are the small services that work together.”

Architecture Diagram:

Kubernetes



- Based on master-slave architecture where master runs on different server than slaves.
- Actual application code runs on slaves nodes while master consist only configuration files, secret files, etcd, scheduling etc.
- Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others. It manages the slave nodes.
- **Minikube** - k8s cluster with single node. used for test purposes. If you want to quickly spin up a k8s environment on your laptop.
- **kubectl** - command line tool for k8s cluster. you can create, delete, update components and debug stuff in k8s using *kubectl*.
- **kubelet** - k8s process that runs on each node to manage containers: starting, communicating with them etc.
- So in combination, you may have a **minikube** cluster with 1 node that has **kubelet** process running on it, and you can create/update/delete things in the cluster using **kubectl**.



Kubernetes Cluster Components: Two major components!

1. Control Plane Components:

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's `replicas` field is unsatisfied).

Control plane components can be run on any machine in the cluster. However, for simplicity, set up scripts typically start all control plane components on the same machine, and do not run user containers on this machine.

1. API Server:

The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

The main implementation of a Kubernetes API server is [kube-apiserver](#). `kube-apiserver` is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of `kube-apiserver` and balance traffic between those instances.

2. Etcd:

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

If your Kubernetes cluster uses etcd as its backing store, make sure you have a [back up](#) plan for those data.

3. kube-scheduler:

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

4. kube-controller-manager:

Control plane component that runs controller processes. Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

Some types of these controllers are:

- Node controller: Responsible for noticing and responding when nodes go down.
- Job controller: Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion.
- EndpointSlice controller: Populates EndpointSlice objects (to provide a link between Services and Pods).
- ServiceAccount controller: Create default ServiceAccounts for new namespaces.

5. cloud-controller-manager:

A Kubernetes control plane component that embeds cloud-specific control logic. The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that only interact with your cluster. The cloud-controller-manager only runs controllers that are specific to your cloud provider.

The following controllers can have cloud provider dependencies:

- Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
- Route controller: For setting up routes in the underlying cloud infrastructure
- Service controller: For creating, updating and deleting cloud provider load balancers

2. Node Components:

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

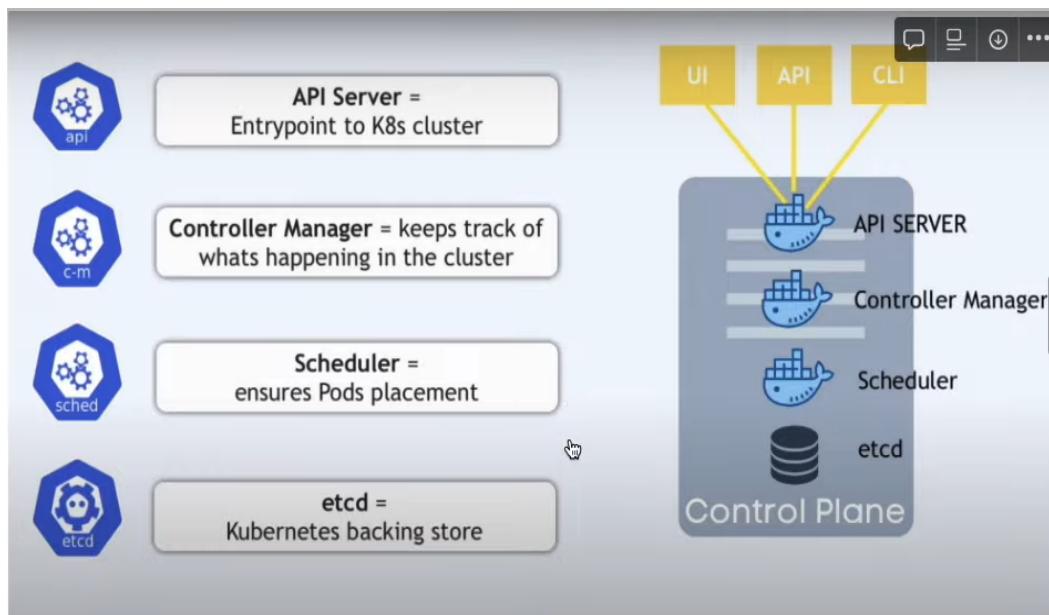
1. kubelet:

An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod. The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

Each node runs two main components - *a kubelet and a container runtime*. The kubelet is in charge of facilitating communication between the control plane and the node.

2. kube-proxy:

kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept. kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.



Node vs Pod vs Kubernetes Container:

- Node is a server or virtual machine. Nodes in Kubernetes can be described as the smallest units of computing power. They are a collection of CPU and memory resources used by the tool to run processes. Nodes are usually available in clusters — intelligent groups of nodes that can distribute workload among their component nodes to increase efficiency.
- Pods are the smallest independent deployable units in Kubernetes. Kubernetes pods are collections of containers that share the same resources and local network. This enables easy communication between containers in a pod. Pods

replicate a logical host for containers that are tightly coupled with each other. Although they can host multiple containers together, limiting the number of containers in one pod to a minimum is advised. This is because pods are meant to be scaled up and down quickly, and each container in the pod is scaled with it irrespective of its requirements.

- In a nutshell, containers are packages of applications or services bundled together with their execution environments. Containers are a powerful CI/CD asset since they can be created and modified programmatically.

How Do Nodes, Pods, and Containers Work With Each Other?

1. Containers are packages of applications and execution environments.
 2. Pods are collections of closely-related or tightly coupled containers.
 3. Nodes are computing resources that house pods to execute workloads.
-

Kubernetes Project

As we know we need to different server for master and slave which might be costly. So there is a tool **Minikube** used to run kubernetes locally.

Step-1. Create an EC2 ubuntu instance with name “kube-instance” and machine type t2.medium.

Important: t2.medium is not free tier, we are using because it is as per the requirement of minikube. So delete it after completion of this project asap.

The screenshot shows the AWS Cloud Console interface for launching a new EC2 instance. The top navigation bar includes links like 'Top Microservices', 'Microsoft Azure', 'Create Service', 'Batch III - A', 'Top Docker', 'Kubernetes', 'Launch instance', 'Key pairs', 'shashiitp19', 'minikube', and 'Other Bookmarks'. The main content area is titled 'Launch instance' and shows a form with a 'Name' field containing 'kube-instance' and an 'Add additional tags' button. Below this is a section titled 'Application and OS Images (Amazon Machine Image)'. It contains a search bar and a 'Quick Start' section with icons for Amazon Linux, macOS, Ubuntu, Windows, and Red Hat. A link to 'Browse more AMIs' is also present. To the right, a 'Summary' panel shows 'Number of instances' set to 1, and a 'Software Image (AMI)' section for Canonical, Ubuntu, 22.04 LTS. The bottom of the screen shows the Windows taskbar with various pinned icons.

This screenshot continues the 'Launch instance' wizard. The left sidebar shows the 'Description' of the selected AMI: 'Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-03-25'. The 'Architecture' is listed as '64-bit (x86)'. The 'AMI ID' is 'ami-05f998315cca9bfe3' and it is marked as a 'Verified provider'. The 'Instance type' section shows 't2.medium' selected, with detailed pricing information for On-Demand Linux, RHEL, SUSE, and Windows instances. The right side of the screen displays the 'Summary' panel with the same configuration. The bottom of the screen shows the Windows taskbar.

The screenshot shows the AWS EC2 Instances 'Launch an instance' page. At the top, there is a success message: 'Successfully initiated launch of instance (i-04ca57757d3e2f4f5)'. Below this, there is a link to 'Launch log'. The main area is titled 'Next Steps' with a search bar and a navigation menu. Under 'Next Steps', there are four options: 'Create billing and free tier usage alerts', 'Connect to your instance', 'Connect an RDS database', and 'Create EBS snapshot policy'. The 'Connect to your instance' option is highlighted. At the bottom of the page, there is a footer with various links and system status information.

Connect to instance-

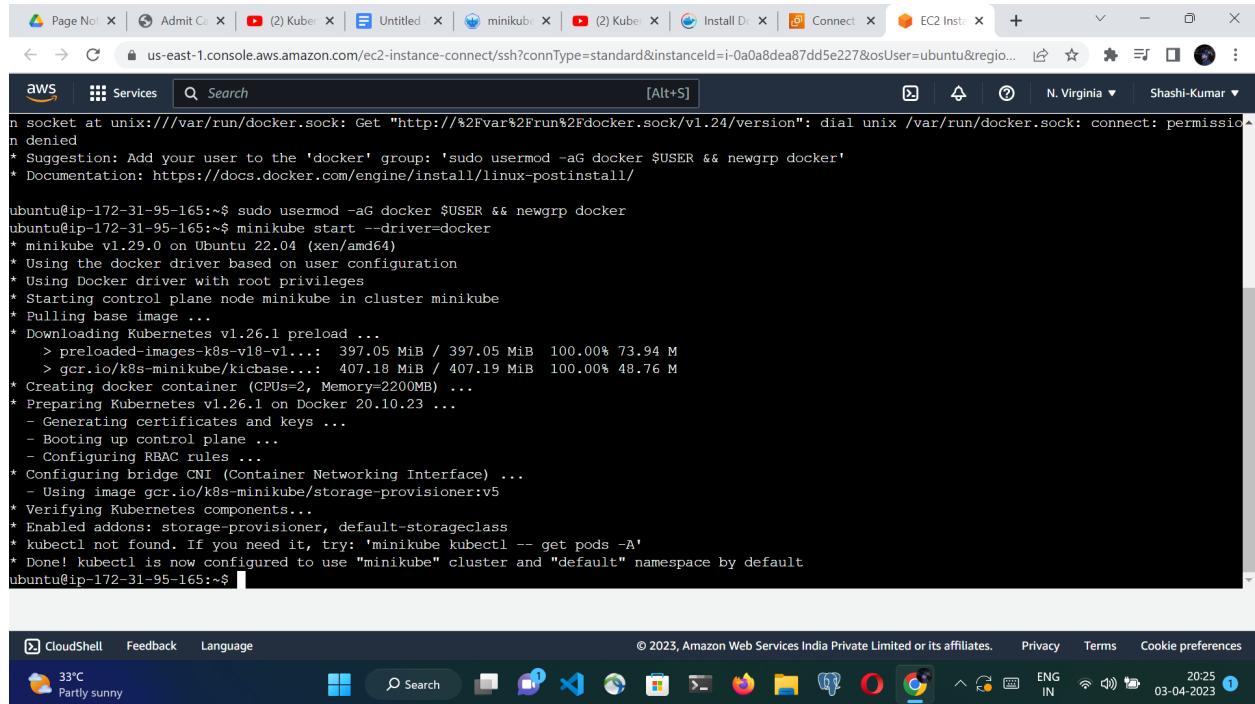
The screenshot shows the AWS EC2 Connect to instance page for instance i-0a0a8dea87dd5e227. The page title is 'Connect to instance Info'. It provides options to connect via 'EC2 Instance Connect', 'Session Manager', 'SSH client', or 'EC2 serial console'. The 'EC2 Instance Connect' tab is selected. The page displays the instance ID (i-0a0a8dea87dd5e227), public IP address (54.174.138.112), and user name (ubuntu). A note at the bottom states: 'Note: In most cases, the default user name, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.' At the bottom right, there are 'Cancel' and 'Connect' buttons. The footer includes standard AWS navigation links and system status information.

Step-2.

Install docker: <https://docs.docker.com/engine/install/ubuntu/>

Install Minikube: <https://minikube.sigs.k8s.io/docs/start/>

Minikube installed successfully.



The screenshot shows a browser window with multiple tabs open, including 'Page Not Found', 'Admit Go', 'YouTube (2)', 'Untitled', 'minikube', 'YouTube (2)', 'Install Docker', 'Connect', and 'EC2 Inst...'. The main content area displays the terminal output of the minikube installation process. The output includes commands like 'sudo usermod -aG docker \$USER && newgrp docker', 'minikube start --driver=docker', and 'minikube v1.29.0 on Ubuntu 22.04 (xen/amd64)'. It also shows the download and configuration of Kubernetes components, including certificates, RBAC rules, and storage provisioner images. The terminal ends with 'ubuntu@ip-172-31-95-165:~\$'.

```
n socket at unix:///var/run/docker.sock: Get "http://$2Fvar$2Frun$2Fdocker.sock/v1.24/version": dial unix /var/run/docker.sock: connect: permission denied
* Suggestion: Add your user to the 'docker' group: 'sudo usermod -aG docker $USER && newgrp docker'
* Documentation: https://docs.docker.com/engine/install/linux-postinstall/
ubuntu@ip-172-31-95-165:~$ sudo usermod -aG docker $USER && newgrp docker
ubuntu@ip-172-31-95-165:~$ minikube start --driver=docker
* minikube v1.29.0 on Ubuntu 22.04 (xen/amd64)
* Using the docker driver based on user configuration
* Using Docker driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.26.1 preload ...
  > preloaded-images-k8s-v18-v1...: 397.05 MiB / 397.05 MiB 100.00% 73.94 M
  > gcr.io/k8s-minikube/kicbase...: 407.18 MiB / 407.19 MiB 100.00% 48.76 M
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.26.1 on Docker 20.10.23 ...
- Generating certificates and keys ...
- Booting up control plane ...
- Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
- Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Verifying Kubernetes components...
* Enabled addons: storage-provisioner, default-storageclass
* kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
ubuntu@ip-172-31-95-165:~$
```

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 33°C Partly sunny 20:25 ENG IN 03-04-2023

After minikube installation:

- sudo snap install kubectl --classic (this will install kubectl CLI)
- kubectl get po -A

```

* Enabled addons: storage-provisioner, default-storageclass
* kubelet not found. If you need it, try: 'minikube kubelet -- get pods -A'
* Done! kubelet is now configured to use "minikube" cluster and "default" namespace by default
ubuntu@ip-172-31-95-165:~$ kubectl get po -A
kubectl: command not found
ubuntu@ip-172-31-95-165:~$ kubectl get po -A
kubectl: command not found
ubuntu@ip-172-31-95-165:~$ sudo snap install kubectl
error: This revision of snap "kubectl" was published using classic confinement and thus may perform
arbitrary system changes outside of the security sandbox that snaps are usually confined to,
which may put your system at risk.

If you understand and want to proceed repeat the command including --classic.
ubuntu@ip-172-31-95-165:~$ kubectl get po -A
kubectl: command not found
ubuntu@ip-172-31-95-165:~$ sudo snap install kubectl --classic
kubectl 1.26.3 from Canonical installed
ubuntu@ip-172-31-95-165:~$ kubectl get po -A
NAMESPACE     NAME           READY   STATUS    RESTARTS   AGE
kube-system   coredns-787d4945fb-2z5bd   1/1     Running   0          7m29s
kube-system   etcd-minikube      1/1     Running   0          7m35s
kube-system   kube-apiserver-minikube  1/1     Running   0          7m43s
kube-system   kube-controller-manager-minikube  1/1     Running   0          7m35s
kube-system   kube-proxy-7lr8x      1/1     Running   0          7m30s
kube-system   kube-scheduler-minikube  1/1     Running   0          7m35s
kube-system   storage-provisioner  1/1     Running   0          7m40s
ubuntu@ip-172-31-95-165:~$ 

```

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 33°C Partly sunny 20:35 ENG IN 03-04-2023

“Learn How to Manage Docker Containers”

Step-3. Now first create a docker image of our app and run it to verify if app is running. After this just delete that image.

- git clone <https://github.com/LondheShubham153/django-todo-cicd.git> kube-project
- cd kube-project
- ls

Verify if you can see all the files.

aws Services Search [Alt+S] N. Virginia Shashi-Kumar

```
ubuntu@ip-172-31-95-165:~$ ls
django-app minikube-linux-amd64 my-app snap
ubuntu@ip-172-31-95-165:~$ git clone https://github.com/LondheShubham153/django-todo-cicd.git kube-project
Cloning into 'kube-project'...
remote: Enumerating objects: 311, done.
remote: Counting objects: 100% (49/49), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 311 (delta 38), reused 32 (delta 32), pack-reused 262
Receiving objects: 100% (311/311), 124.25 KiB | 10.35 MiB/s, done.
Resolving deltas: 100% (160/160), done.
ubuntu@ip-172-31-95-165:~$ ls
django-app kube-project minikube-linux-amd64 my-app snap
ubuntu@ip-172-31-95-165:~$ cd kube-project/
ubuntu@ip-172-31-95-165:~/kube-project$ ls
Dockerfile LICENSE README.md db.sqlite3 docker-compose.yml k8s manage.py staticfiles todoApp todos
ubuntu@ip-172-31-95-165:~/kube-project$
```

- vim Dockerfile and verify if it is correct and save.

aws | Services | Search [Alt+S] | N. Virginia | Shashi-Kumar

```
FROM python:3
RUN pip install django==3.2

COPY . .

RUN python manage.py migrate
EXPOSE 8000
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

"Dockerfile" 10L, 150B
```

Step-4. Build and run docker image and push it to Docker Hub repository for centralized management.

- docker build -t shashi1995/kube:v1 .
- docker images
- docker run -it -p 8000:8000 your-image-id

```
ubuntu@ip-172-31-95-165:~$ cd kube-project/
ubuntu@ip-172-31-95-165:~/kube-project$ docker build -t shashi1995/kube:v1 .
[+] Building 3.7s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 189B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3
=> [auth] library/python:pull token for registry-1.docker.io
=> [1/4] FROM docker.io/library/python:3sha256:5329e75033c4446dc92d702cf8ebbeb63e549d9b83076a776f6753e10817fc3c
=> [internal] load build context
=> => transferring context: 496.76kB
=> CACHED [2/4] RUN pip install django==3.2
=> [3/4] COPY .
=> [4/4] RUN python manage.py migrate
=> exporting to image
=> exporting layers
=> => writing image sha256:ed0d3de580ca497357661c545f29bc8695206632db83d1c0150balb336618be6
=> => naming to docker.io/shashi1995/kube:v1
ubuntu@ip-172-31-95-165:~/kube-project$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
i-0a0a8dea87dd5e227 (kube-instance)

PublicIPs: 54.174.138.112 PrivateIPs: 172.31.95.165
```

```
ubuntu@ip-172-31-95-165:~$ cd kube-project/
ubuntu@ip-172-31-95-165:~/kube-project$ docker build -t shashi1995/kube:v1 .
[internal] load build context
=> => transferring context: 496.76kB
=> CACHED [2/4] RUN pip install django==3.2
=> [3/4] COPY .
=> [4/4] RUN python manage.py migrate
=> exporting to image
=> exporting layers
=> => writing image sha256:ed0d3de580ca497357661c545f29bc8695206632db83d1c0150balb336618be6
=> => naming to docker.io/shashi1995/kube:v1
ubuntu@ip-172-31-95-165:~/kube-project$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
shashi1995/kube     v1      ed0d3de580ca  20 seconds ago  973MB
my-repo              v1      a0dc1da84d86  11 hours ago   973MB
shashi1995/my-repo  v1      a0dc1da84d86  11 hours ago   973MB
testapp              v1      0f857f70500e  11 hours ago   973MB
gcr.io/k8s-minikube/kicbase  v0.0.37  01c0ce65fff7  2 months ago  1.15GB
hello-world          latest   febd59fea6a5  18 months ago  13.3kB
ubuntu@ip-172-31-95-165:~/kube-project$ docker run -it -p 8000:8000 ed0d3de580ca
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
    HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
```

```

hello-world latest feb5d9fea6a5 18 months ago 13.3kB
ubuntu@ip-172-31-95-165:~/kube-project$ docker run -it -p 8000:8000 ed0d3de580ca
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
    HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

System check identified 1 issue (0 silenced).
April 04, 2023 - 12:54:14
Django version 3.2, using settings 'todoApp.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
[04/Apr/2023 12:54:22] "GET /todos/ HTTP/1.1" 200 4127
[04/Apr/2023 12:54:22] "GET /static/css/style.css HTTP/1.1" 200 225
Not Found: /favicon.ico
[04/Apr/2023 12:54:23] "GET /favicon.ico HTTP/1.1" 404 2460
[04/Apr/2023 12:54:27] "GET /todos/ HTTP/1.1" 200 4127
[04/Apr/2023 12:54:30] "GET /todos/ HTTP/1.1" 200 4127
^Cubuntu@ip-172-31-95-165:~/kube-project$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See

```

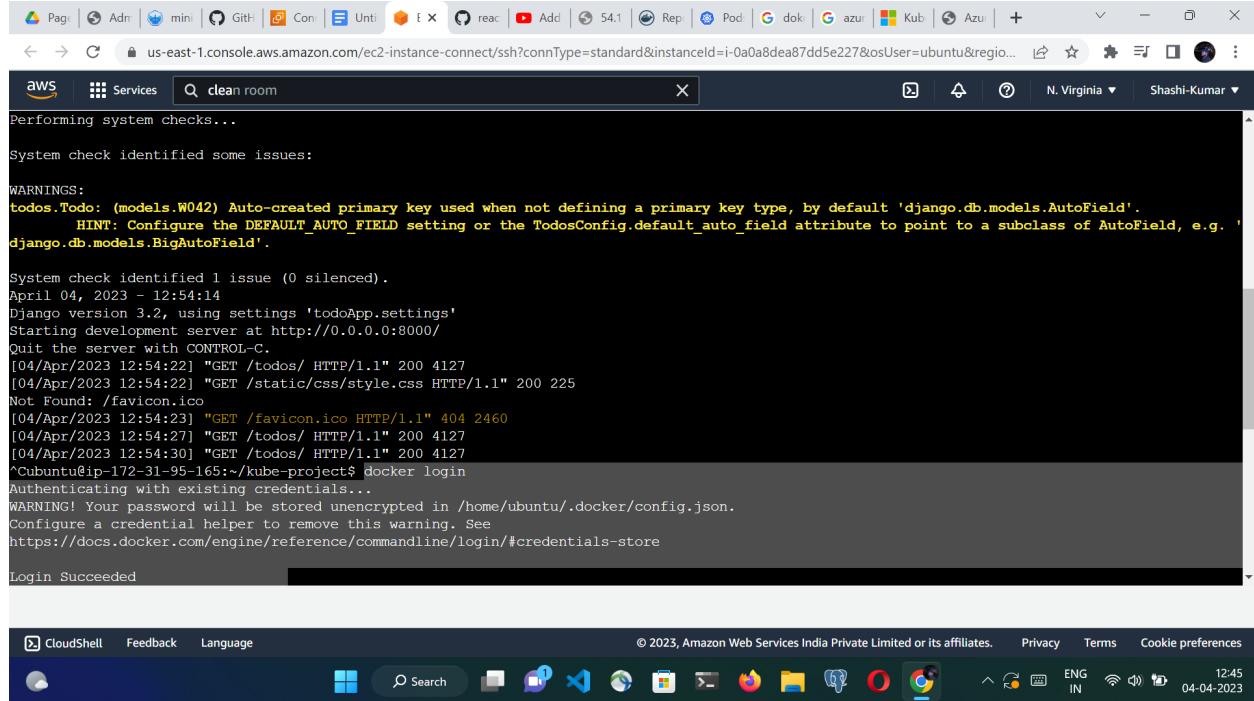
After running the container, our app is running on 8000-

Todo List - Modified once again

	Task	Delete
<input type="checkbox"/>	Docker file banao	
<input type="checkbox"/>	Send Resume Google now !	
<input type="checkbox"/>	Hacktoberfest Updates	

Step-5. Login to DockerHub and push you image there. So that whenever required you can pull it.

- docker login



```
aws Pag: Adm mini Git: Con Unt: f x reac Add 54.1 Rep: Pod dok azu Kub: Azu + 
← → C https://us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0a0a8dea87dd5e227&osUser=ubuntu&region=N. Virginia Shashi-Kumar 
AWS Services Search clean room X 
Performing system checks...
System check identified some issues:
WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

System check identified 1 issue (0 silenced).
April 04, 2023 - 12:54:14
Django version 3.2, using settings 'todoApp.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
[04/Apr/2023 12:54:22] "GET /todos/ HTTP/1.1" 200 4127
[04/Apr/2023 12:54:22] "GET /static/css/style.css HTTP/1.1" 200 225
Not Found: /favicon.ico
[04/Apr/2023 12:54:23] "GET /favicon.ico HTTP/1.1" 404 2460
[04/Apr/2023 12:54:27] "GET /todos/ HTTP/1.1" 200 4127
[04/Apr/2023 12:54:30] "GET /todos/ HTTP/1.1" 200 4127
$Cubuntu@ip-172-31-95-165:/kube-project$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Push the images to your docker hub:

- docker push shashi1995/kube:v1

```

Login Succeeded
ubuntu@ip-172-31-95-165:~/kube-project$ docker push shashi1995/kube:latest
The push refers to repository [docker.io/shashi1995/kube]
tag does not exist: shashi1995/kube:latest
ubuntu@ip-172-31-95-165:~/kube-project$ docker push shashi1995/kube:v1
The push refers to repository [docker.io/shashi1995/kube]
5cb0f48fd74a: Pushed
f2acabef3b2e: Pushed
d77cccd805585: Mounted from shashi1995/my-repo
7db0b7dc960a: Mounted from shashi1995/my-repo
76ff57e73979: Mounted from shashi1995/my-repo
13cf5de1dd97: Mounted from shashi1995/my-repo
5c563cc0b216: Mounted from shashi1995/my-repo
d4514fb2aac: Mounted from shashi1995/my-repo
5ab567b9150b: Mounted from shashi1995/my-repo
a50e3914fb92: Mounted from shashi1995/my-repo
053a1f71007e: Mounted from shashi1995/my-repo
ec09eb3ea03: Mounted from shashi1995/my-repo
v1: digest: sha256:e8e511c3a1589f4c134349e80bb6b18a643545b028fbfce66050d787634b5061 size: 2851
ubuntu@ip-172-31-95-165:~/kube-project$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
shashi1995/kube    v1      ed0d3de580ca  19 minutes ago  973MB
my-repo             v1      a0dc1da84d86  11 hours ago   973MB
shashi1995/my-repo v1      a0dc1da84d86  11 hours ago   973MB
testapp             v1      0f857f70500e  11 hours ago   973MB
gcr.io/k8s-minikube/kicbase v0.0.37  01c0ce65fff7  2 months ago  1.15GB

```

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 12:50 04-04-2023

Verify from dockerhub-

Repository	Last Pushed	Stars	Downloads	Status
shashi1995/kube	18 minutes ago	0	1	Inactive
shashi1995/my-repo	11 hours ago	0	1	Inactive

https://hub.docker.com/repository/docker/shashi1995/kube

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 12:57 04-04-2023

“Learn How to Manage Deployment, Replication, Auto-Healing and Auto-Scaling.”

Q- Now why we are using kubernetes when the app can be run through docker only?

- Docker is not production ready approach, which means there are several issues that may occur like-
 1. Containers are short term, means it may be crashed any time.
 2. In containers we are running processes and processes are prune to stop. E.g - some how our ec2 instance is down, due to heavy load ec2 is not able to provide required resources.
 3. Some by mistake press Ctrl + C key, which will terminate the running container.
 4. Someone may kill our containers
 5. No autohealing and autoscaling

Start with Kubernetes:

1. Create a pod and run it with kubectl. It will pull the image from dockerhub.

Create a new directory for all kubernetes files.

- mkdir kube-files
- cd kube-files
- vim pod.yaml (Create a pod.yaml file and save it)

The screenshot shows a web browser window with multiple tabs open at the top, including 'Page', 'Admin', 'minik', 'GitHub', 'Conn', 'EC2', 'Add', 'Until', 'Todo', 'Pods', 'Repo', and 'x'. The main content area displays the 'Using Pods' page from the official Kubernetes documentation. The page title is 'Using Pods'. A sidebar on the left contains a navigation menu with sections like Home, Getting started, Concepts, Workloads, and Pods. The 'Pods' section is currently selected. The main content area shows a code snippet for a YAML file named 'simple-pod.yaml'.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

On the right side, there is a sidebar with links for editing the page, creating child pages, creating issues, and printing the section. Below the sidebar, there is a list of related topics such as 'What is a Pod?', 'Using Pods', 'Workload resources for managing pods', etc.

2. Edit it with your image and name-

The screenshot shows the AWS CloudShell interface. The URL in the address bar is 'us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0a0a8dea87dd5e227&osUser=ubuntu®ion=us-east-1'. The main content area displays a YAML configuration for a pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
  - name: my-app
    image: shashil995/kube:v1
    ports:
    - containerPort: 8000
```

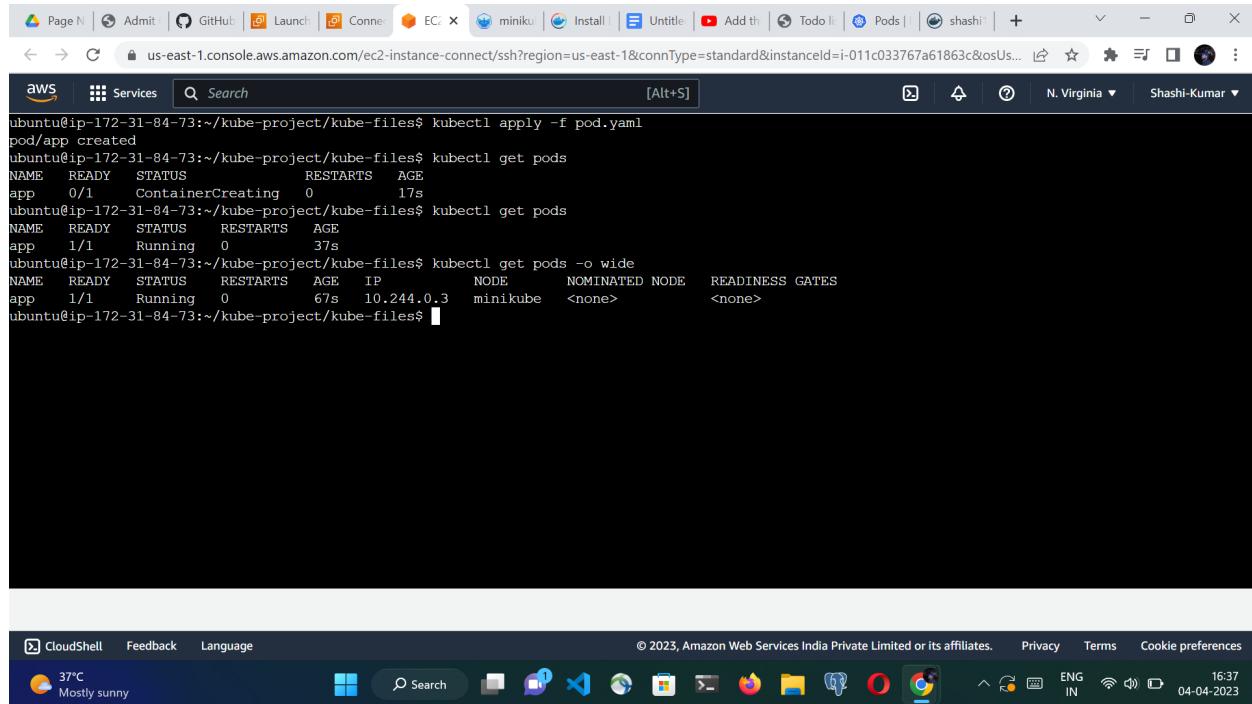
At the bottom of the screen, there is a toolbar with various icons for navigation and system functions. The status bar at the bottom right shows the date and time as '04-04-2023 14:44'.

3. Create our pod:

`kubectl apply -f pod.yaml (pod created)`

If getting server not connecting error-Delete your minikube and start: minikube delete && minikube start

```
kubectl get pods  
kubectl get pods -o wide
```



The screenshot shows a terminal window within an AWS CloudShell interface. The terminal is running on an Ubuntu instance (ip-172-31-84-73). The user has run several commands to manage a pod named 'app':

```
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl apply -f pod.yaml  
pod/app created  
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl get pods  
NAME READY STATUS RESTARTS AGE  
app 0/1 ContainerCreating 0 17s  
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl get pods  
NAME READY STATUS RESTARTS AGE  
app 1/1 Running 0 37s  
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl get pods -o wide  
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES  
app 1/1 Running 0 67s 10.244.0.3 minikube <none> <none>  
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$
```

The terminal also displays the AWS logo and navigation links at the top, and various system icons and status bars at the bottom.

4. Now check if our app is running or not internally because as of now we have not exposed it to the outer world.

```
curl -L http://podip:8000
```

```
Send Resume Google now !
<a href="/todos/3/delete" title="Delete">
  <i class="far fa-trash-alt"></i>
</a>
</div>

<div class="list-group-item ">
<form style="display: inline;" method="post" action="/todos/2/update">
  <input type="hidden" name="csrftoken" value="5ksCvjt6cV9vcp3XOVdCBnGn0MuJRqOVgHYir9OkbbwdxQWGUJQKs4G5hVzDie">
  <input type="checkbox" name="isCompleted" onchange="this.form.submit()" class="todo-status-checkbox" title=" mark as done ">
</form>
Hacktoberfest Updates
<a href="/todos/2/delete" title="Delete">
  <i class="far fa-trash-alt"></i>
</a>
</div>
</div>

</body>
</html>docker@minikube:~$
```

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 37°C Mostly sunny ENG IN 16:42 04-04-2023

5. But again this pod can be deleted or can be crashed. For example: lets delete our pod

- kubectl delete pod app-name

```
<input type="checkbox" name="isCompleted" onchange="this.form.submit()" class="todo-status-checkbox" title=" mark as done ">
</form>
Hacktoberfest Updates
<a href="/todos/2/delete" title="Delete">
  <i class="far fa-trash-alt"></i>
</a>
</div>
</div>
</div>

</body>
</html>docker@minikube:~$ exit
logout
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
app      1/1     Running   0          11m
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl delete pod app
pod "app" deleted
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl get pods
No resources found in default namespace.
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$
```

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 37°C Mostly sunny ENG IN 16:48 04-04-2023

**6. So still we are suffering from this production ready deployment like auto healing and auto scaling.
But this is problem of pods only.**

**In reality in production there is no deployment of pods.
Actually there is a proper kubernetes deployment.**

The screenshot shows a browser window displaying the Kubernetes documentation at <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>. The page title is "Creating a Deployment". The left sidebar has a "Concepts" section expanded, showing "Overview", "Cluster Architecture", "Containers", "Windows in Kubernetes", "Workloads", "Pods", and "Workload Resource". The main content area shows a code snippet for "controllers/nginx-deployment.yaml":

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
```

On the right side, there are "Edit this page" and "Create child page" buttons, along with links to "Create an issue" and "Print entire section". A sidebar titled "Use Case" lists: "Creating a Deployment", "Pod-template-hash label", "Updating a Deployment", and "Rollover (aka multiple updates in-flight)". The bottom of the screen shows a Windows taskbar with various icons and a system tray indicating "37°C Mostly sunny", the date "04-04-2023", and the time "16:52".

7. Create 'deployment.yaml' file

For label and selector have the same name, because using this information controller checks the status of our deployment and accordingly takes actions.

- vim deployment.yaml

Edit the details like name and label-selector with same name, port -8000 and give the image path from dockerhub repo.

aws | Services | Search [Alt+S] | N. Virginia | Shashi-Kumar

```
name: app-deployment

labels:

  app: my-app

spec:

replicas: 3

selector:

  matchLabels:

    app: my-app

template:

  metadata:

    labels:

      app: my-app

  spec:
```

- `kubectl get pods`

```
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
app-deployment-5d4f46b94-j8kjw   1/1     Running   0          15m
app-deployment-5d4f46b94-k4xkv   1/1     Running   0          15m
app-deployment-5d4f46b94-mtkm8   1/1     Running   0          15m
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$
```

3 pods are running and Now if you delete anyone, controller automatically creates a new container just in seconds.

Try this by deleting:

- `kubectl delete pod pod-name`

The screenshot shows a terminal window within an AWS CloudShell interface. The user has run the command `kubectl get pods`, which lists three pods: `app-deployment-5d4f46b94-j8kjw`, `app-deployment-5d4f46b94-k4xkv`, and `app-deployment-5d4f46b94-mtkm8`. The user then runs `kubectl delete pod app-deployment-5d4f46b94-j8kjw`, which successfully deletes the first pod. Finally, the user runs `kubectl get pods` again, and the list shows only the two remaining pods: `app-deployment-5d4f46b94-k4xkv` and `app-deployment-5d4f46b94-mtkm8`.

```
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
app-deployment-5d4f46b94-j8kjw  1/1     Running   0          15m
app-deployment-5d4f46b94-k4xkv  1/1     Running   0          15m
app-deployment-5d4f46b94-mtkm8  1/1     Running   0          15m
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl delete pod app-deployment-5d4f46b94-j8kjw
pod "app-deployment-5d4f46b94-j8kjw" deleted
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
app-deployment-5d4f46b94-k4xkv  1/1     Running   0          18m
app-deployment-5d4f46b94-mtkm8  1/1     Running   0          18m
app-deployment-5d4f46b94-smg49  1/1     Running   0          5s
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$
```

CloudShell | Feedback | Language | © 2023, Amazon Web Services India Private Limited or its affiliates. | Privacy | Terms | Cookie preferences | 37°C Mostly sunny | Search | 🌐 | 📱 | 🎨 | 🚧 | 🌐 | ENG IN | 17:23 | 04-04-2023

Important - As you can see after deleting still there are 3 pods running because kubernetes now managing the **Auto-Healing**.

For **auto-scaling** you can change replicas under “deployment.yaml” to any number.

The screenshot shows the AWS CloudShell interface. The terminal window displays a YAML configuration file for a Kubernetes Deployment named 'app-deployment'. The file specifies 5 replicas and a selector matching the 'app: my-app' label. The browser window in the background shows the AWS Services menu.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deployment
  labels:
    app: my-app
spec:
  replicas: 5
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
```

“Learn How to Manage Network And Services”

Step-8. Now curl and see if our app is running;

curl -L <http://pod-ip:8000>

It will not work because app is running inside cluster.

Now we will create service node port file for binding ports.

The screenshot shows a browser window displaying the Kubernetes documentation at <https://kubernetes.io/docs/concepts/services-networking/service/>. The page title is "Service". The main content area shows a YAML configuration for a service named "my-service" with type "NodePort". The "ports" section includes a comment about the target port being set to the same value as the port field. Below the code, there's a note about custom IP address configuration for NodePort services. To the right, there's a sidebar with links to other Kubernetes concepts like Services in Kubernetes, Cloud-native service discovery, and Port definitions.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    # By default and for convenience, the `targetPort` is set to the same value as the `port` field.
    - port: 80
      targetPort: 80
      # Optional field
      # By default and for convenience, the Kubernetes control plane will allocate a port from a
      nodePort: 30007
```

Why Service.yaml -

All the containers runs inside the cluster. So they are not exposed for outside world. There can be n number of containers so to manage all of them together we will have service.yaml. In which we creates a binding between inside(8000) to outside(80) port.

Change target port as 8000.

- vim service.yaml
- kubectl apply -f service.yaml
- kubectl get svc

Have service.yaml like this only

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-todo
spec:
  type: NodePort
  selector:
    app: my-app
  ports:
    # By default and for convenience, the `targetPort` is set to the same value as the `port` field.
    - port: 80
```

```
targetPort: 8000
# Optional field
# By default and for convenience, the Kubernetes control plane will allocate a port from a range (default:
30000-32767)
nodePort: 30009
```

Now curl the url:

```
minikube service my-app-todo --url
```

curl -L url-from above command.

E.g curl -L <http://192.168.49.2:30009>

See ip app is running;

```
</form>
Send Resume Google now !
<a href="/todos/3/delete" title="Delete">
    <i class="far fa-trash-alt"></i>
</a>
</div>

<div class="list-group-item ">
<form style="display: inline;" method="post" action="/todos/2/update">
    <input type="hidden" name="csrfmiddlewaretoken" value="NwCcZcFdtEQdWP3X4qzkGwVkdYSgpPwm6a0INaW0JDK39wzjOxBzwlRj7wOXBx">
    <input type="checkbox" name="isCompleted" onchange="this.form.submit()" class="todo-status-checkbox" title=" mark as done ">
</form>
Hacktoberfest Updates
<a href="/todos/2/delete" title="Delete">
    <i class="far fa-trash-alt"></i>
</a>
</div>
</div>

</body>
```

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

27°C Partly cloudy Search Icons ENG IN 00:43 05-04-2023

“Learn How to do Host Ip allocation through proxy”

- sudo vim /etc/hosts

Add your host ip with any name of your choice.

E.g. “192.168.19.5 my-app.com”

```

127.0.0.1 localhost
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
192.168.49.2 my-django-app.com

```

-- INSERT --

12,1 All

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences GBPIN +0.57% 00:52 05-04-2023 ENG IN

- curl -L http://my-django-app.com:30009

```

ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ sudo vim /etc/hosts
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ curl -L http://my-django-app.com:30009
curl: (3) URL using bad/illegal format or missing URL
ubuntu@ip-172-31-84-73:~/kube-project/kube-files$ curl -L http://my-django-app.com:30009

<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">
    <!-- Our CSS file -->
    <link rel="stylesheet" href="/static/css/style.css" />
```

```

<title>Todo list</title>

  </head>
  <body>
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849bE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwjlyYfoRSjoZ+n" crossorigin="anonymous"></script><script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RH/RH1h�J4p3wD5Wfdh8S/gHdzTF4DhJ7jeR0ZSMENqKwFtgbNnw6" crossorigin="anonymous"></script><script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2DluUdjOO3uMBJnjuUD4Ih7YwaYdliqfktj0Uod8GCEx130g8ifwB6" crossorigin="anonymous"></script>
```

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 27°C Haze 00:51 05-04-2023 ENG IN

Finally all done.

SUMMARY

- “Learn How to Manage Docker Containers.”
- “Learn How to Manage Deployment, Replication, Auto-Healing and Auto-Scaling.”
- “Learn How to Manage Network And Services”
- “Learn How to do Host Ip allocation through proxy”

Thank You