Google APP engine and Kubernetes Engine

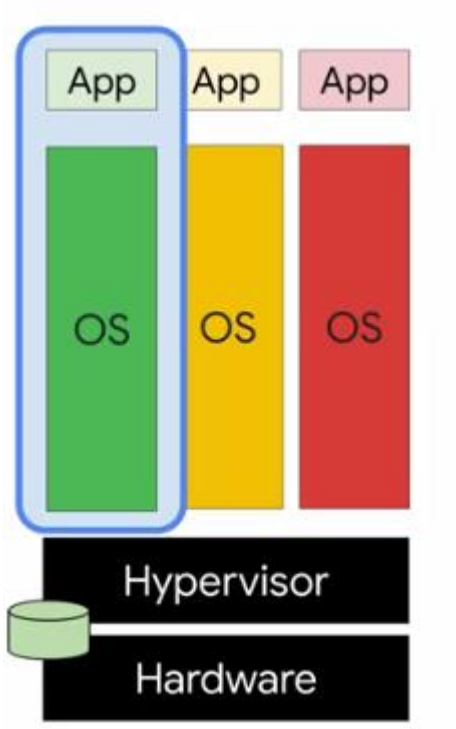
What are Containers and Kubernetes Engine??

Kubernetes Engine:

- As a Infrastructure as a Service offering it saves you from doing infrastructure related routine task.
- As a platform as a service offering, it is built with the needs of developers in mind.

Containers:
Way to package software and are managed in Kubernetes Engine

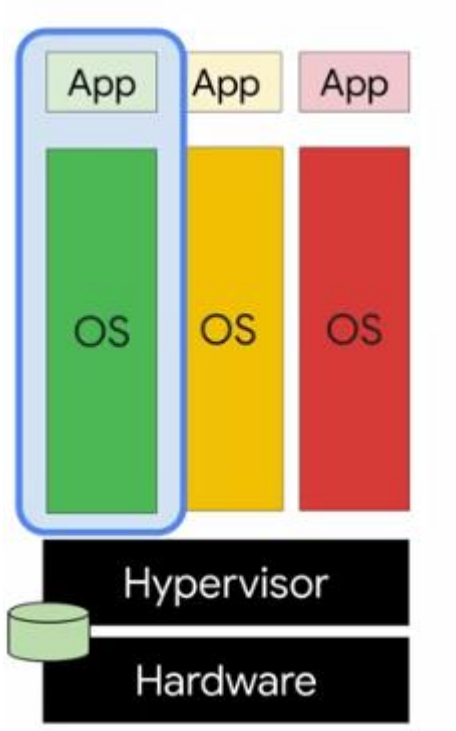
Google Cloud / AWS : IAAS offerings- A Scenario



- IAAS offering allows to share compute resources with others by virtualizing the hardware
- `VM has own instance of an operating system (Guest OS)
- build and run applications on the VM with access to
  - memory,
  - file systems,
  - networking interfaces,
  - other attributes that are offered by physical computers also.
- But flexibility comes with a cost.

Images displays on this slide are not mine

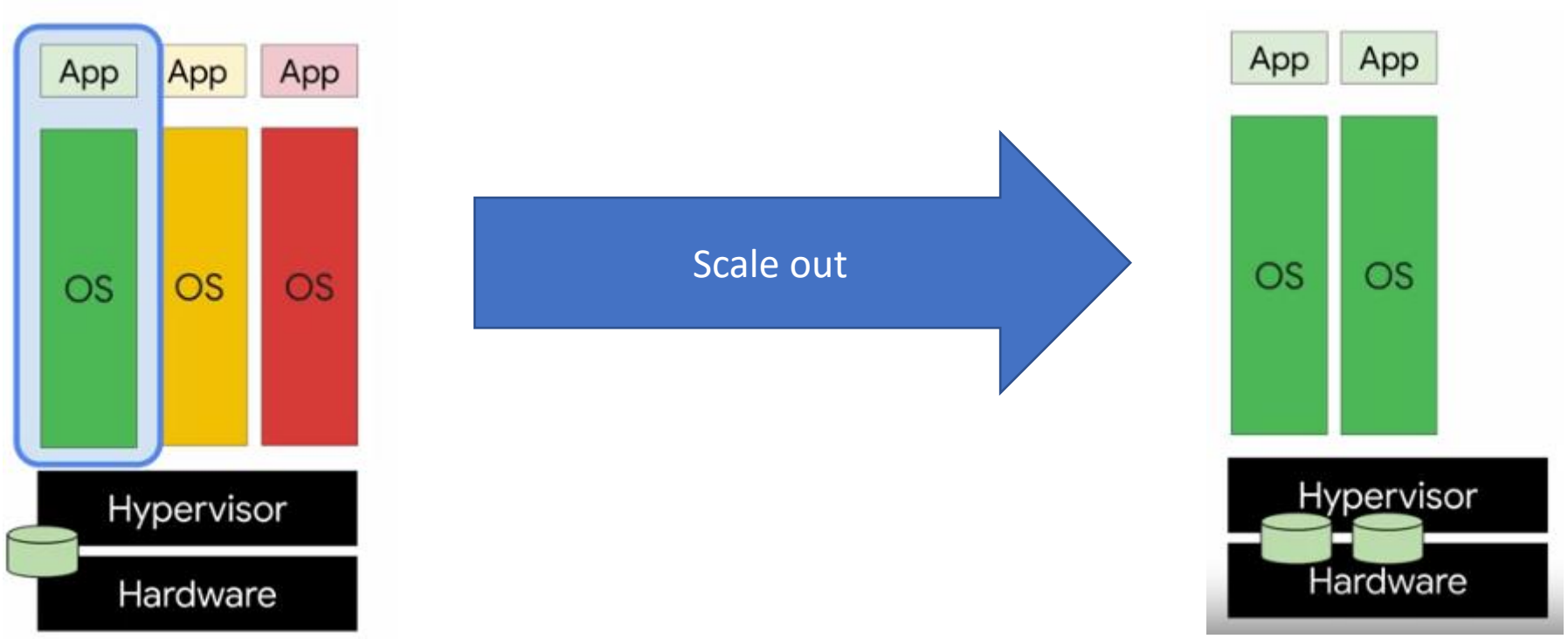
Google Cloud / AWS : IAAS offerings- A Heavyweight VM



Images displayed on this slide are not mine

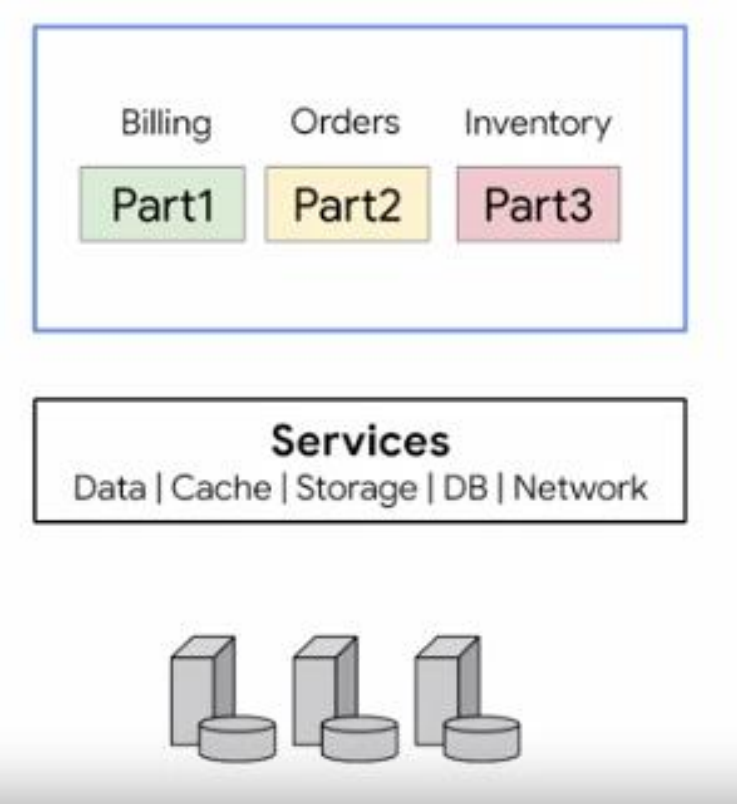
- the smallest unit of compute is a Virtual Machine along with its application.
- Cost associated:
  - operating system is large (even gigabytes in size)
  - OS take minutes to boot up.
  - On scaling more resources are consumed so add to the cost
- Goodies that come with it
  - Virtual Machine are highly configurable,
  - Configurable:
    - underlying system resources: disks and networking
    - install and run your tools of choice.
    - install your own web server database or a middle ware

Google Cloud / AWS : IAAS offerings- A Scaling Scenario



Images displayed on this slide are not mine

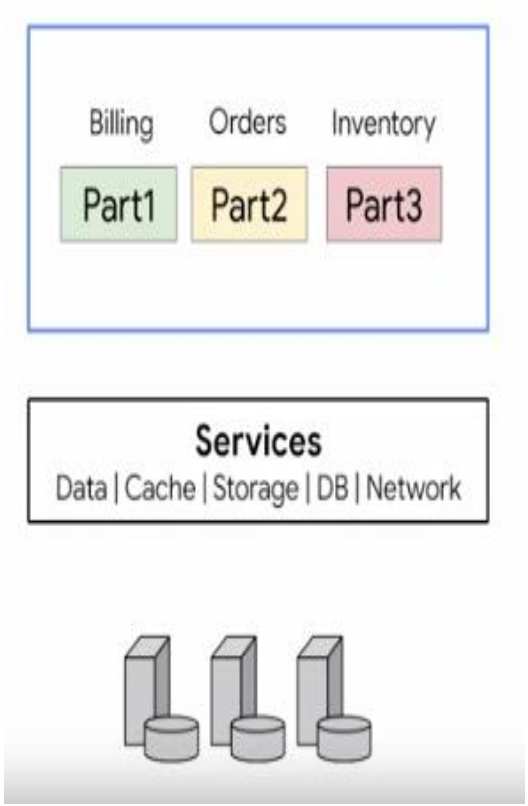
Google Cloud PAAS offerings: APP Engine Scenario

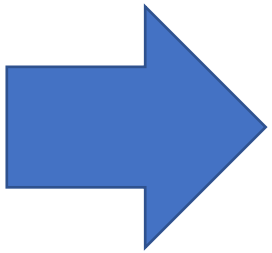


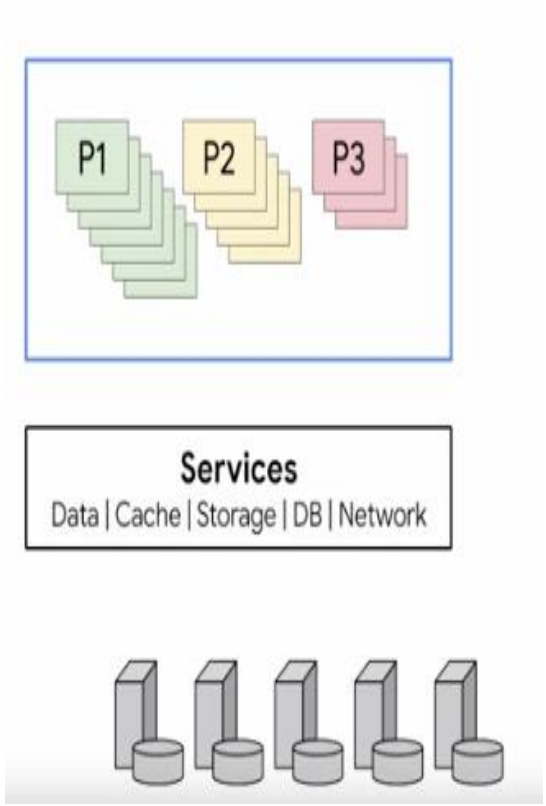
- No Blank Virtual Machine
- Access to a set of services that applications require
- write your code and self-contained workloads and include any dependent libraries.

Images displayed on this slide are not mine

Google Cloud PAAS offerings: APP Engine Scaling







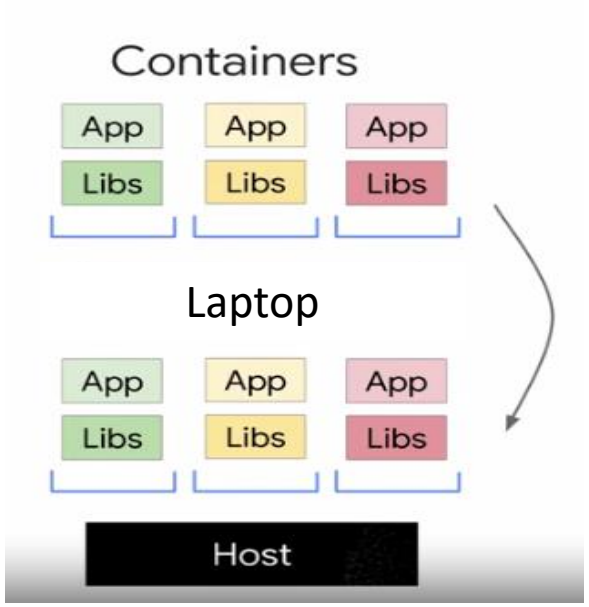
Platform scales your applications seamlessly and independently by workload and infrastructure.

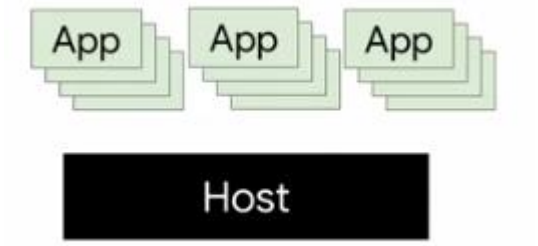
BUT

Lose the underlying control of the server architecture

Images displayed on this slide are not mine

Containers-A Introduction



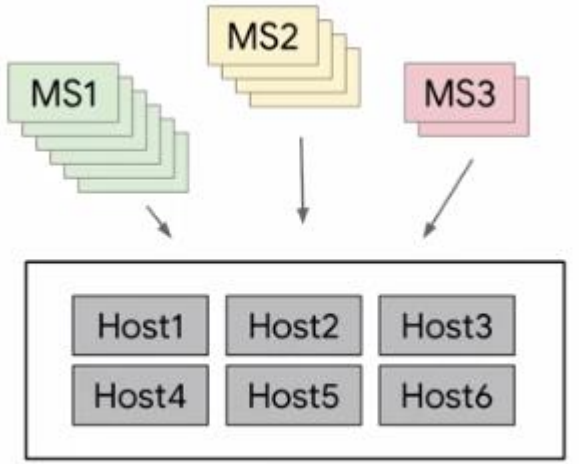


- **Containers provide**
  - **independent scalability of workloads like PaaS environment**
  - **abstraction layer of the operating system and hardware like Infrastructure as a Service environment.**
- Container starts as quickly as a new process
- Container can run on any host having an operating system that supports Containers and a Container run-time.
- It virtualizing the operating system rather than the underlying hardware
- Scales like PaaS with flexibility like IaaS
- container abstraction makes your code very portable.
- Ex: Web server can scale in seconds and deploy as many as needed.

Images displayed on this slide are not mine

More about Containers

- Build our own applications using many Containers and by applying micro-services pattern.
  - Each container performs its own function
  - Code contained in this Containers can communicate with each other over a network fabric.
- make applications modular.
- deploy it easily and scale independently across a group of hosts
- Trigged by demand, host can
  - start and stop containers
  - scale up and down
  - hosts fail and are replaced



Kubernetes

Images displayed on this slide are not mine

Kubernetes

- Kubernetes makes it easy to build and run many Containers on many hosts. Scale them on a need basis, roll out new versions of containers and roll back to the old version when needed to do so

Tools to Build Containers

DOCKERS
(Open Source)

Cloud Build
(Google)

Let’s Build a Container

- Python Web Application using a Flask Framework.

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
  return "Hello World!\n"

@app.route("/version")
def version():
  return "Helloworld 1.0\n"

if __name__ == "__main__":
  app.run(host='0.0.0.0')
```

Code displayed on this slide is not mine

How To Deploy the application??

- Create a Docker file to specify how our code gets packaged into a Container

```
FROM ubuntu:18.10
RUN apt-get update -y && \
    apt-get install -y python3-pip python3-dev
COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN pip3 install -r requirements.txt
COPY . /app
ENTRYPOINT ["python3", "app.py"]
```

Requirements Text

```
Flask==0.12
uwsgi==2.0.15
```

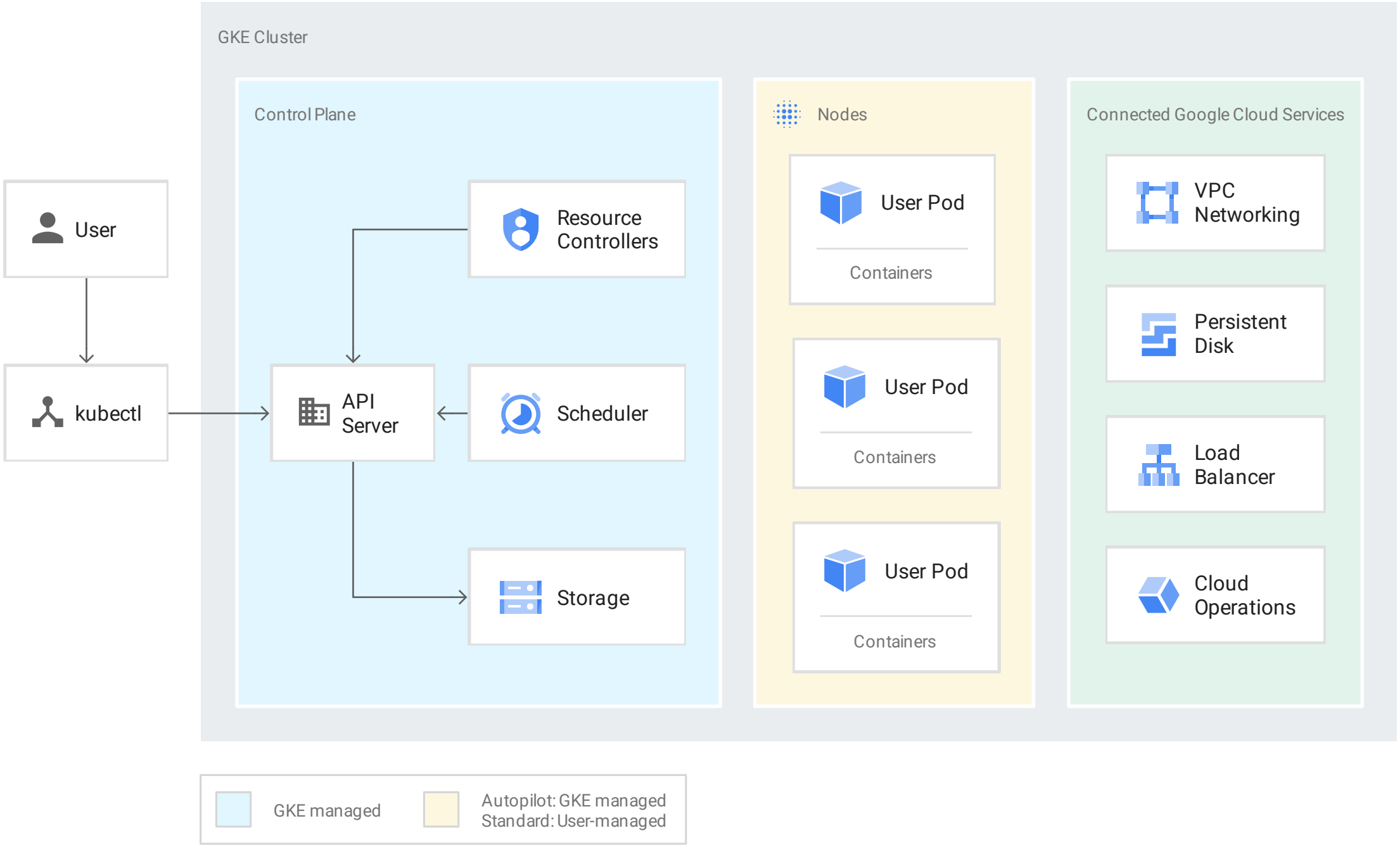
Code displayed on this slide is not mine

Build and Run the Container

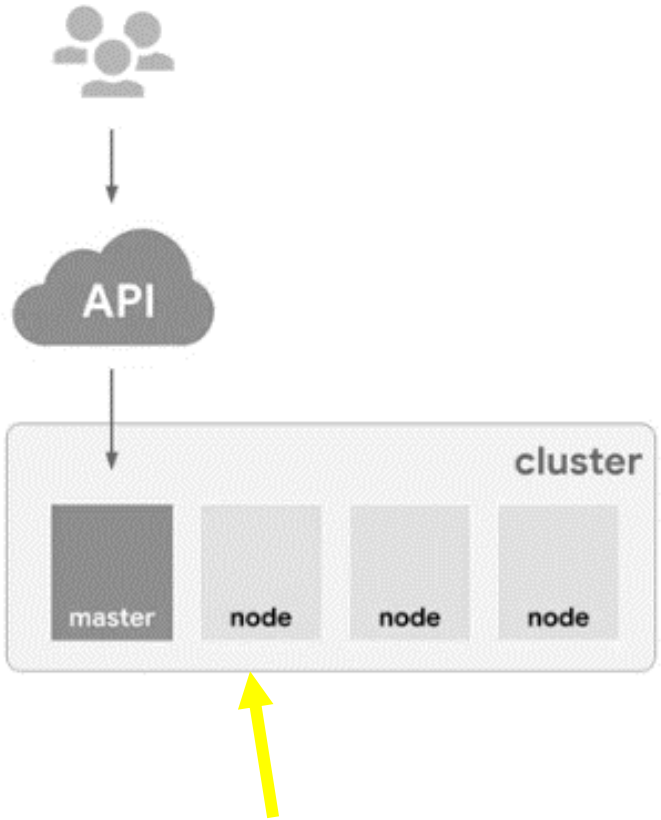
```
docker build -t py-server
docker run -d py-server
```

- Build command: builds the Container and store it on the local system as a runnable image.
- run command: to run the image

Kubernetes and Google GKE/AWS EKS


GKE Cluster
Control Plane
Nodes
Connected Google Cloud Services
User
kubectl
Resource Controllers
API Server
Scheduler
Storage
User Pod
Containers
User Pod
Containers
User Pod
Containers
VPC Networking
Persistent Disk
Load Balancer
Cloud Operations
GKE managed
Autopilot: GKE managed
Standard: User-managed

Kubernetes

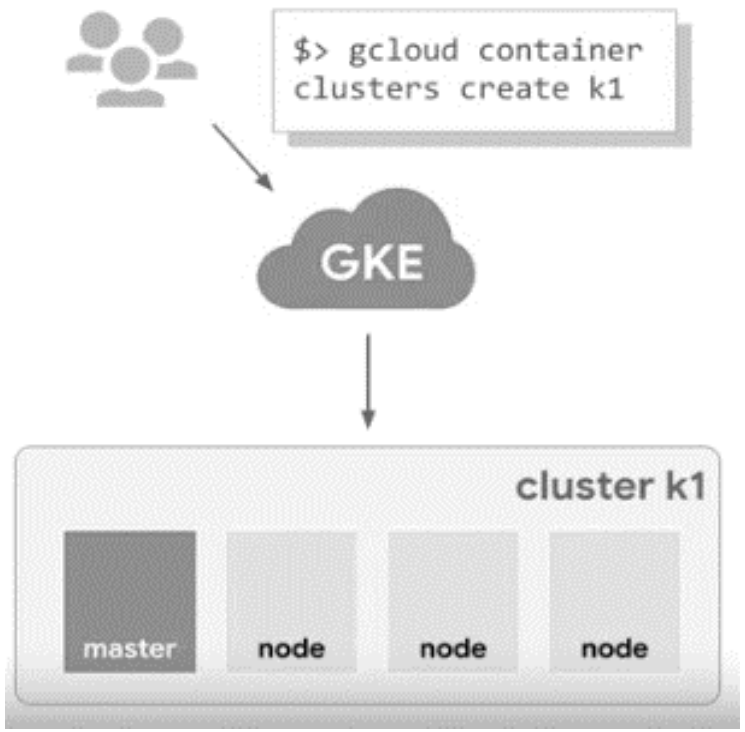


computing instance

- open-source orchestrator for containers to easily manage and scale your applications.
- Kubernetes offers an API that allows authorized persons to control its operation through several utilities. Ex: Kubectl command.
- Allows to deploy containers on a set of nodes called a cluster.
- Cluster is a
  - set of master components that control the system and
  - a set of nodes that run containers.

Images displayed on this slide is not mine

how to create a Kubernetes cluster?

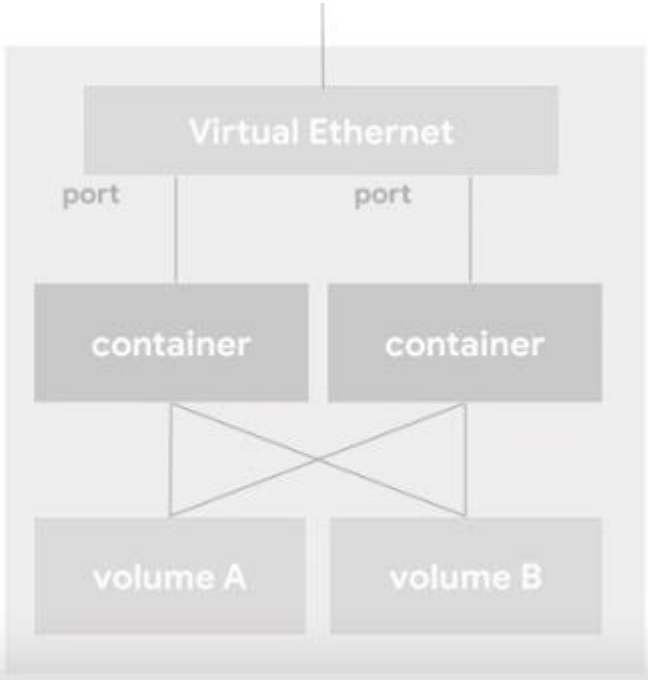


- Google Cloud provides Kubernetes Engine as a managed service in the cloud.
- Create a Kubernetes cluster with Kubernetes Engine using the GCP console or the g-cloud command that's provided by the Cloud SDK.
- GKE clusters support different machine types, numbers of nodes and network settings.

Images displayed on this slide is not mine

PODS

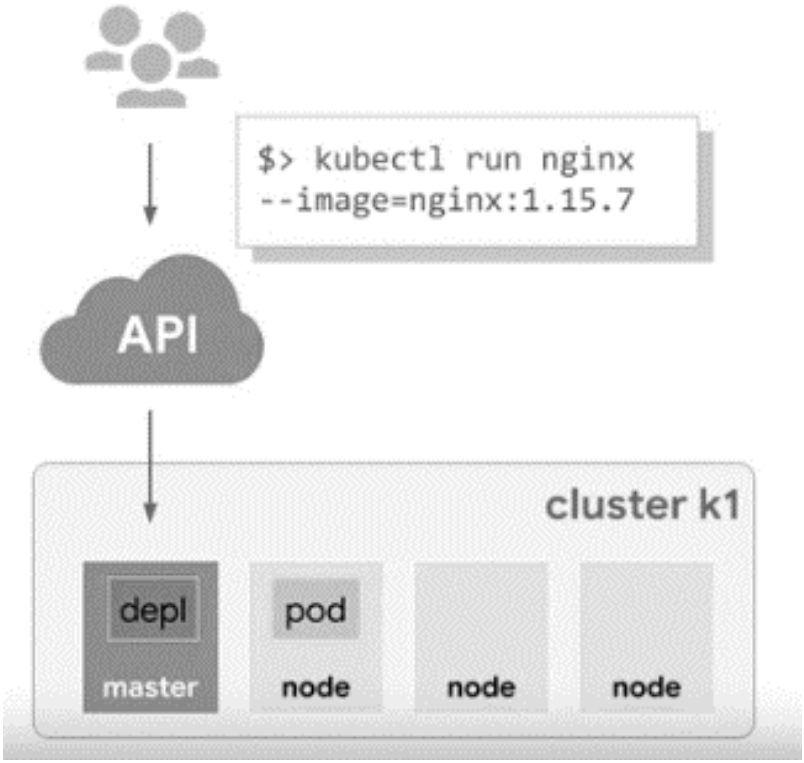
Kubernetes deploys a container or a set of related containers inside an abstraction called a pod.



- Pod :
- smallest deployable unit in Kubernetes.
- Like a running process on your cluster.
- It can be one component of your application or even an entire application.
- one container per pod.
- multiple containers with a hard dependency then package them into a single pod.
- automatically share networking and they can have disk storage volumes in common
- gets a unique IP address and set of ports for their containers.
- pod can communicate with each other using the localhost network inte

Images displayed on this slide is not mine

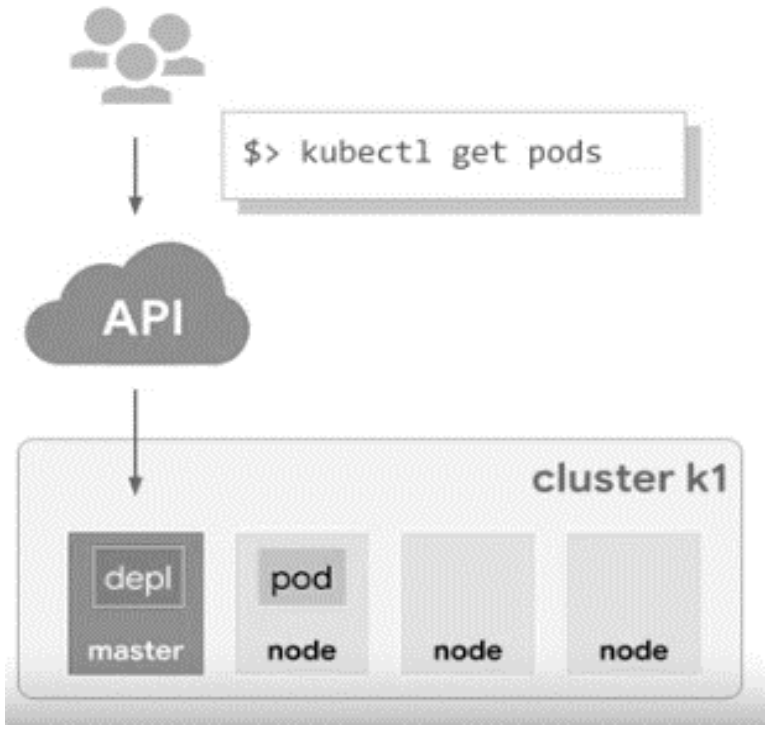
run a container in a pod



- kubectl run command
- Ex: Container running inside the pod is an image of the popular nginx open source web server.
- The kubectl command fetch an image of nginx from a container registry.

Images displayed on this slide is not mine

What is Deployment (DePL) and what does it do??

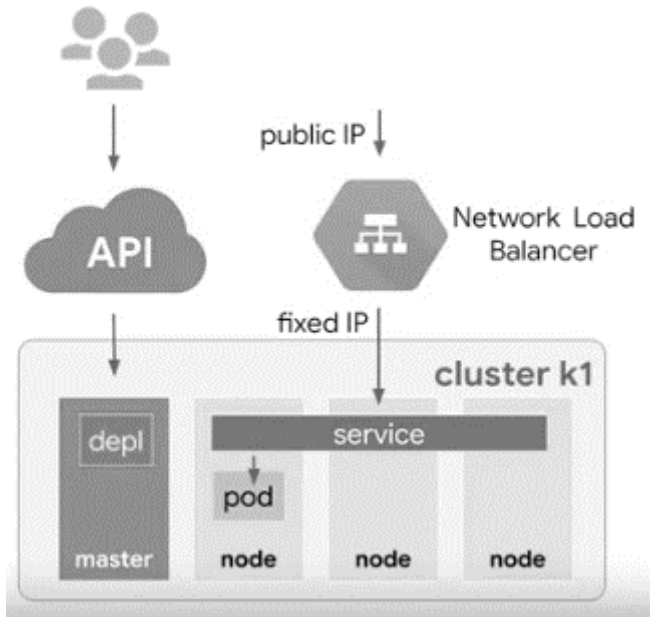


- represents a group of replicas of the same pod.
- keeps pods running even if a node on which some of the pods run on; fails
- contain a component of your application or entire application.
- (Ex nginx web server)
- By default, pods in a deployment are only accessible inside your cluster

Images displayed on this slide is not mine

How people on the Internet can access the content available on nginx web server?

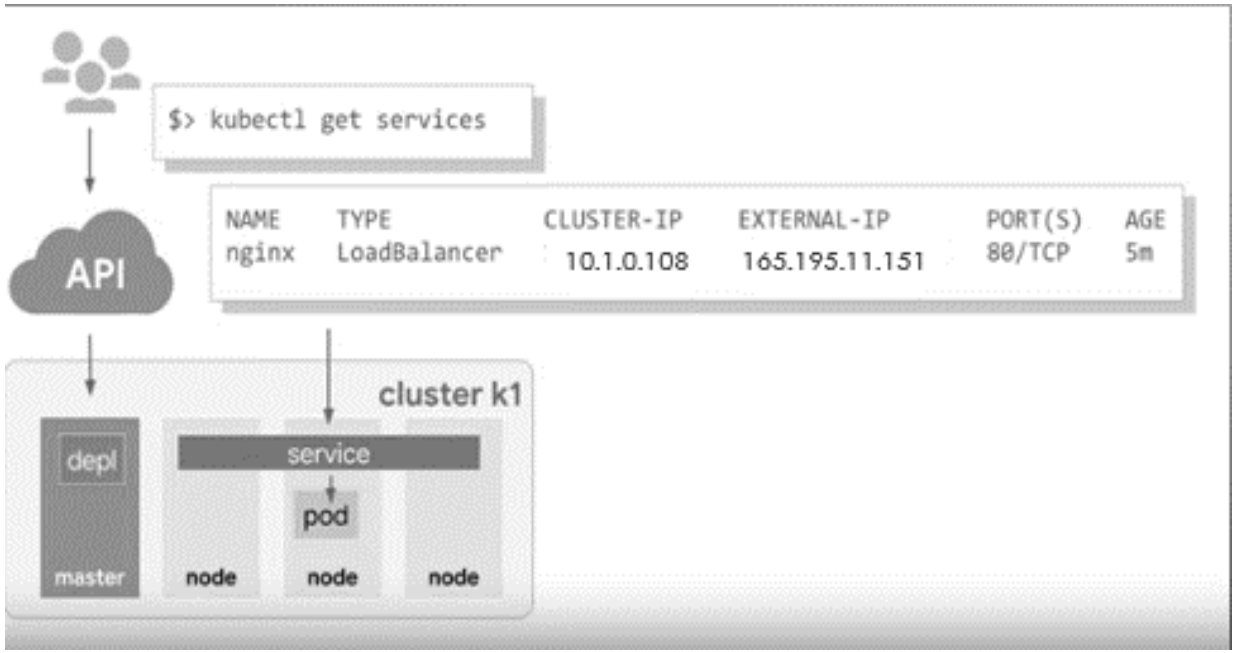
```
$> kubectl expose deployments
nginx --port=80
--type=LoadBalancer
```



- connect a load balancer to it by running the kubectl expose command.
- GKE, this kind of load balancer is created as a network load balancer.
- creates a service with a fixed IP address for the pods
- It is managed load balancing services that Compute Engine makes available to virtual machines
- Any client that hits that IP address will be routed to a pod behind the service.

Images displayed on this slide is not mine

What is this Service??



- groups a set of pods together and provides a stable endpoint for them
- kubectl get services command shows your service's public IP address

Images displayed on this slide is not mine

Scaling

```
$> kubectl scale nginx
--replicas=3
```

```
$> kubectl autoscale
nginx --min=10 --max=15
--cpu=80
```

- To scale a deployment, run the kubectl scale command.
- Autoscale command for auto scaling

Learning at the end of Sessions

- Student at the end of the sessions 16 and 17 have learned about the following
  - what are IAAS and PAAS offerings??
  - what are containers and kubernetes engine??
  - Tools to create containers
  - How to build a Docker container and deploy application on it??
  - How to run the container?
  - Learned about GKE
  - What are GKE clusters??
  - How to create a GKE cluster with many nodes and master deployment component??
  - What are pods and how to run the container in the pod??
  - Container scaling and auto scaling
  - How to make container content accessible over the internet??

References

- https://cloud.google.com/kubernetes-engine
- https://aws.amazon.com/eks/getting-started/
- Images are not mine
- Coursera Course offered by Google Cloud
- Qwiklabs.com

Thank You