[**https://cloud.google.com/docs**](https://cloud.google.com/docs)

**Cloud - Internet**

**Virtualization** helps to run multiple virtual machines on a single underlying hardware to utilize the resources like(CPUs, RAM etc..) in an efficient manner.

**Storage** is a place where our data will be stored.

**network** is a collection of various computing devices(computers, servers, mainframes, network devices, peripherals etc..) connected to allow data sharing.

**Backup** refers to copying of physical or virtual files or databases to a secondary location for preservation in case of equipment failure or catastrophe.

A **database** is an organized collection of structured information or data typically stored electronically in a computer system.

**Types of hardware involved with a data center:** routers, switches, firewalls, cables and modems.

**Before Cloud:**

If u want to host a website, buy a stack of servers, monitoring and maintenance of your servers. But the setup is expensive, troubleshooting problems, since the traffic id idle most of the time, your servers will be idle most of the time.

**Using Cloud:**

No more buying expensive servers. Cloud provider will manage your servers.

Scalability! your server capacity will vary according to traffic.

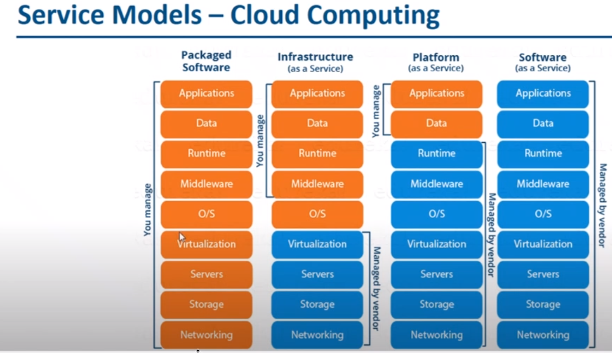
**Cloud Computing:** It is the use of remote servers on the internet to store, manage and process data rather than a local server or your personal computer. (It is a delivery of computing services(servers, storage, Dbs, networking, software etc) over the internet)

**Cloud models**: Service Models(Iaas, PaaS, SaaS), Deployment Models(Public, Private, Hybrid clouds)

Public Cloud (on-demand basis) - AWS, GCP

private Cloud – Openstack, VMWare, Ubuntu

Hybrid Cloud – AzureStack.



**Iaas:** need to manage VM and install SQL DB, SQL DB server (pay for what they allocate)

AWS, Digital Ocean, Google Compute Engine, Microsoft Azure

**Paas:** Database service, u r managing just the data not anything so **Google Cloud** can also come under this. (pay for what they use)

AWS Elastic Beanstalk, Windows Azure, Openshift

**Saas** : no need to manage anything.

Cisco Webex, GoToMeeting, Google Workspace, Dropbox.

Google uses only 2% of electricity

**Serverless Computing:** GCP has two serverless computing options : App engine and cloud functions.

**Regions and Zones:**

* **Region :** Specific geographical location where users can deploy cloud resources. – high availability, low latency, global footprint, adhere to government regulations.
* 34 regions( In india - delhi, Mumbai) and 103 zones all over the world.
* Each region has 3 or more zones.
* Each zone has one or more discrete clusters.
* **Cluster**: distinct physical infrastructure that is housed in a data center.

**GCP Resource Hierarchy:**

* The central abstraction for managing GCP resources is the resource hierarchy.
* It has 3 levels : Organization(root of hierarchy), Folder, Projects, Resources also.
* Policies are there at every level and those are inherited.
* Every project has Unique project Id mutable during creation and immutable after creation.
* Project name need not be unique and is mutable.
* Project number is also unique given by google cloud itself.

<https://www.cloudskillsboost.google/focuses/2794?parent=catalog> 🡪 to practice gcp.( I have signed up with google act 360)

**Billing Accounts:** It is mandatory for creating resources in a project.

By default one billing account is connected to our project.

In Billing -> budgets & alerts -> create Budget.

**Discounts** increases with usage. If u use N1,N2 machine types for more time, u will get discounts.

**IAM( identity and Access Management applies policies) :** Administrators can apply policies that define who can do what on which resources.

Three Roles: Primitive roles, pre-defined roles, Custom roles(user-defined)

Custom role is not defined at folder level, it is at organization or project level.

viewers, editors, owners🡪primitive roles , billing administrator.

**Service Account:** To authenticate to CM to cloud storage. If an application on a VM needs access to cloud storage without using personal credentials to allow access, use Service Account. No need of password. It’s normally used by other services.

What objects in GCP are sometimes resources and sometimes as identities: **Service Accounts.**

**Cloud Shell:** It is a virtual machine that is loaded with development tools. It provides command-line access to your google cloud resources. It offers a persistent 5GB home directory.

[**https://www.cloudskillsboost.google/focuses/1035?parent=catalog**](https://www.cloudskillsboost.google/focuses/1035?parent=catalog) **🡪**handsOn lab on IAM.

21-07-2022

**Compute engine** is a service that provides VM’s that run on GCP. Google Compute Engine**:** Provision & manage virtual machines.

Different Machine families for different Workloads:

* **General Purpose** (E2, N2, N2D,N1): Best price-performance ratio.
* **Memory Optimized(**M2,M1) : ultra-high memory workloads
* **Compute Optimized**(C2): compute intensive workloads

Types of Images: Public images and custom images.(os-image).

**External And Internal IP addresses:**

* External (Public) IP addresses are internet addressable.
* Internal (Private) IP addresses are internal to corporate network.
* You cannot have two resources with same public Ip address.
* All VM instances are assigned atleast one internal IP addresses.
* When you stop an VM instance then external IP is lost.
* so, we have Static IP address.

**NIC**(network Interface Card)

In GCP -> Compute Engine API ->Enable it.

You can’t create a vm instance without a vpc network

1. **Create a new VM instance and setup http server**

**VM instances ->** Create Instance -> New VM instance :Instance name should start with small case, Label is a key value pair. E.g: env prod, app http etc. Next select region and zone. Machine Family General Purpose E2 Micro(2Vcpu, 1GBmemory), Boot Disk bydefault it is taking Debian, we can use others like ubuntu x86, size-20GB, select it. Allow full access -> Allow http traffic, Allow Https traffic. ->create

we can also create Instance from template and Machine images.

We don’t have the option to change details of Boot Disk if u creating a instance from machine image.

Open cloud shell

**Setting up a HTTP server:**

Click on SSH after creating vm instance and run the following cmnds.

sudo su apt update apt -y install apache2 sudo service apache2 start sudo update-rc.d apache2 enable echo "Hello World" > /var/www/html/index.html echo "Hello world from $(hostname) $(hostname -I)" > /var/www/html/index.html

sudo su - execute commands as a root user  
apt update - Update package index - pull the latest changes from the APT repositories  
apt -y install apache2 - Install apache 2 web server  
sudo service apache2 start - Start apache 2 web server  
echo "Hello World" > /var/www/html/index.html - Write to index.html

echo "Hello world from $(hostname) $(hostname -I)" > /var/www/html/index.html

use that external Ip Address in browser we get the output Hello world.. .

The same output can be displayed even if u copy the external ip of vm instance which was created using machine image of same instance.

22-07-2022

1. **Create an Ubuntu machine with startUp script**

Create a new instance ->Add few labels: app-http, E2, E2 micro, CPU-Automatic, Boot Disk: Ubuntu, x86, 10, Allow default access, Firewall allow http & https traffic, management: Reservations: Automatically use created reservation, Write cmds in startup script from sudo su to helloworld.

Create it.

Open ssh,

sudo su

service apache2 status

cat /var/www/html/index.html

we will get output because we write the script at the time of instance creation itself in management: Automation, so need to run those cmnds after opening ssh.

Download Gcloud SDK <https://cloud.google.com/sdk/docs/install-sdk>

Open cmd

type gcloud init

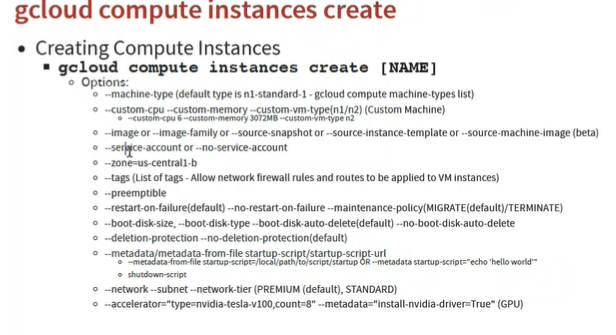
gcloud version

gcloud config list

gcloud compute instances list

gcloud config set compute/region VALUE //can change project names region, zones etc.

gcloud config set project



1. **create windows machine using gcloud. Creating compute instance using gcloud**

Below cmds should run in cmd prompt in the location of script.txt file

#! /bin/bash

apt update

apt -y install apache2

cat <<EOF > /var/www/html/index.html

<html><body><p>Linux startup script added directly.</p></body></html>

cloudinit

gcloud config set compute/zone asia-east1-a

gcloud compute instances create example-gcloud --machine-type=e2-standard-2 --image-project=debian-cloud --image-family=debian-10 --metadata-from-file=startup-script=script.txt

then allow all traffic manually.

[

<https://cloud.google.com/compute/docs/instances/startup-scripts/linux>

<https://cloud.google.com/sdk/gcloud/reference/compute/instances/create>

]

Then copy the external IP, paste in chrome, u get Linux startup script added directly in output.

25-07-2022

**Scaling:**

Total Workload : 4cpu and 8gb memory

**Vertical (scaling up)** Eg: ecommerce: 1CPU and 2GB memory to 4 cpus and 8gb memory

**Horizontal (scaling out)** Eg: Ecommerce 4 VMs of same configuration and 1 cpu and 2 gb memory

Horizontal scaling is better bcoz if one vm goes down, still our application will run in horizontal scaling. Increased availability is there here.s

**Auto scaling:**  we can also decrease or increase the resources.

create vm allow all

In side bar-> snapshots

**Snapshots** preserves the state and data of a cm at a specific point of time(point in time copy)

create a snapshot: source disk: instance name, multi-regional, create.

we can also create scheduled snapshots at a particular time.

can also create a vm instance from snapshot.

1. **Create a simple** VM1 **linux machine with script in mgt. create** snapshot **from vm1 and then** image **from that snapshot, from image create** vm2 **and test the application in vm2. by coping the extl ip of vm2 and pasted in chrome then we will get output: hello world as we written script in vm1.**

#! /bin/bash  
sudo su  
apt update  
apt -y install apache2  
sudo service apache2 start  
sudo update-rc.d apache2 enable  
echo "Hello World" > /var/www/html/index.html  
echo "Hello world from $(hostname) $(hostname -I)" > /var/www/html/index.html

(Creating vm from Image: Click on create image and in the boot disk click on images then create)

**Instance Groups**: Group of VM instances managed as a single entity. Two types: Managed and Unmanaged instance groups. Health checks also there. It can do auto scaling and auto healing also. Rolling Update feature also available.

In lab, Create a new instance group

in stateless : Not saving data

In stateful: saving and storing data

In lab, Create a new instance group stateless, Create a anew instance template ubuntu machine, provide mgt script also -> create and select it, multiple zone, target distribution even, autoscaling - on, min1 and max 4, metrics: Cpu utilization -60, off, 60, health check: tcp 80 name:http basic check, TCP 80, Logs: On, health criteria. , Initial delay :30, create.

In Vm instances we can the instance group. From the extl Ip we can check, whether our application is running or not.

Connect to serial console, copy that cmd and paste in cloud shell editor

sudo su

apt-get install stress

stress --cpu 8 --timeout 20 & (to utilize the cpu) then new instance will created as we are increasing the load. if the lad decreases then automatically vm’s will deleted.

top (to check utilization of load)

task 1 : instance group : min1 max 4

CPU utilization 60

health check for 80

login into machine and install stress.

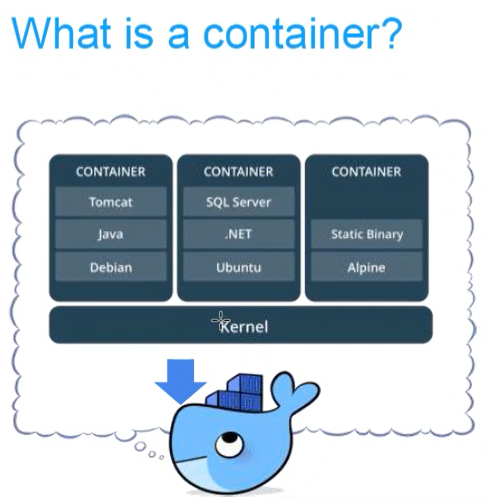
[How to Impose High CPU Load and Stress Test on Linux Using 'Stress-ng' Tool (tecmint.com)](https://www.tecmint.com/linux-cpu-load-stress-test-with-stress-ng-tool/)

do load test and remove that laod then extra vm should be deleted and do health check. stop one vm/service in that vm and check whether new vm is created or not.

26-07-2022

**What is a container:** Standardized packaging for software and dependencies. Its isolate apps from each other. It shares the same OS kernel. It works with all major Linux and Windows server.

**Docker** is a software helps to create containers on VMs.



Containers are an app level construct. VM are at infrastructure level construct to turn one machine to many servers.

Advantages of Docker Container: No OS to boot, less dependencies between process.

Open GCP act -> Create a vm instance -> Ubuntu os -> allow http/https traffic -> create it.

Open ssh -> sudo ssh,

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

Install docker on our machine using above link for cmds.

Between VM and Os kernel we have that **docker**.

1. **Creating a docker container and performing basic operations in online console**

**Online lab link**

<https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-interactive/>

The above link is used to use docker even u don’t have gcp account. In the above link lab, the docker is already installed.

**Image:** The basis of a docker container. The content at rest. Docker images has all needs of microservice(application code, dependencies, application run time)

**Container:** The image when it is running. The standard unit for app service.

**Engine:** The software that executes commands for containers. networking and volumes are part of Engine. Can be clustered together.

**Registry:** Stores, distributes and manages docker images.

*docker pull httpd* (to pull image named httpd from public repository)

*docker images* ( to get list of images)

*docker run httpd* (to start a container)

*docker run -d nginx* (first it will download if not there here and then container will run)-d=detachable mode. to run in background.

*docker ps* (to see list of running containers)

*docker stop <container ID>* (to stop the running container)

*docker rm <Container Id>* (to remove the container)

1. **Running docker container using docker file in online console**

<https://docs.docker.com/get-started/02_our_app/>

**docker file** is a simply a text based script of instructions that is used to create a container image.

1. create Dockerfile
2. Build docker image using dockerfile
3. Create and start container Running the application container

pwd

cd /home

git clone <git repo link>

git clone <https://github.com/docker/getting-started.git>

ll

cd getting-started/

cd app

vi Dockerfile

paste below code (by going to insert mode)

# syntax=docker/dockerfile:1

FROM node:12-alpine

RUN apk add --no-cache python2 g++ make

WORKDIR /app

COPY . .

RUN yarn install --production

CMD ["node", "src/index.js"]

EXPOSE 3000

ESC + :wq! (to save and quit)

cat Dockerfile (to read)

docker build -t getting-started . (I’m using image name as getting-started)

docker images

docker run -dp 3000:3000 getting-started (p=port)(Here use 3000 only as we have exposed in dockerfile)

docker ps

curl localhost:3000 ( to get html page in online lab)

1. **Create a docker container using the** <https://github.com/dockersamples/linux_tweet_app> **in the online interactive lab.**

git clone <https://github.com/dockersamples/linux_tweet_app>

ll (to get list of files/folders in pwd)

cd linux\_tweet\_app/ (as Dockerfile is in linux\_tweet\_app)

docker build -t linux\_tweet\_app . (to build an image) (for image we can give any name)

docker run -dp 80:80 linux\_tweet\_app (to run container)

curl localhost:80 (to check whether our app is running on that port)

27-07-2022

**Kubernetes:** A Open source product that helps manage and scale containerized applications is Kubernetes. So to save time and effort when scaling applications and workloads, Kubernetes can be bootstrapped using Google Kubernetes Engine.

GKE(Google Kubernetes Engine)

It is a software used to manage the containers within the organization.

Kubernetes used to Automatically create and manage large number of containers. Kubernetes is a set of APIs to deploy containers on a set of nodes is called **Cluster.**

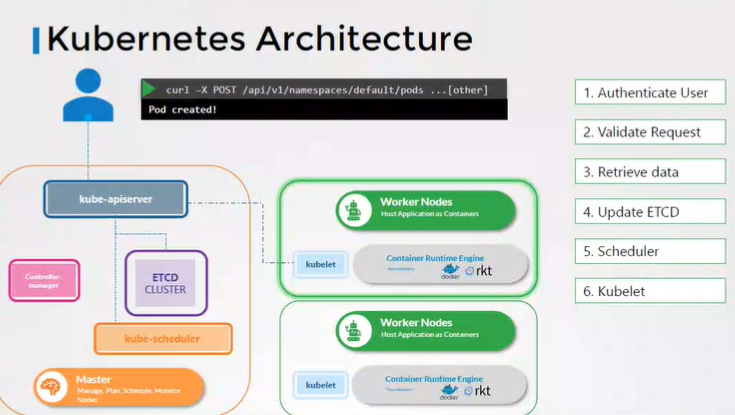
**Container Orchestration features**: Auto Scaling, Service Discovery, Load balancer, Self-Healing, Zero Downtime Deployments

**Master:** Manage, plan, schedule, monitor nodes

* ETCD is a database where all the configurations for cluster is there. That cluster has Nodes, PODs, secrets, roles, Accounts etc.
* kube - apiserver : used to authenticate user, validate request and retrieve data.
* kube Controller Manager: to watch status, remediate situation. Imp: Replication Controller.
* kube Scheduler: to schedule the container on which node it would be created.

**Worker Nodes**: host application as Containers

* kubelet: It is the primary node agent that runs on each node.
* kube-proxy

**pods :** Logical name of container, somewhat similar to compute engine managed instance group

1. **Creating a Kubernetes cluster and create pods for nginx and httpd on Gcp**

In gcp -> Open -> Kubernetes engine API enable it. Under Kubernetes Engine Service -> create a cluster -> GKE Autopilot configure -> public cluster -> create.

After that click on connect, copy that cmd shown and Open cloud shell and paste that cmd -> Authorize. Instead of cloud shell in gcp we use our cmd prompt as well but vi is not there so we need to save that pod.yaml file in the some location and then apply it from same location of saved file.

we can do the same in interactive lab as well. first run cmd- *minikube start*, then write below cmnds

*kubectl get nodes* (2 worker nodes were shown here by default)

*vi pod.yaml* (for nginx pod)extension should be .yaml <https://codingbee.net/tutorials/kubernetes/hello-world-pods> -for reference

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

*kubectl apply -f pod.yaml*

*kubectl describe pods nginx*

*kubectl get pods -o wide* (to get IP and more details like in which node our pod is running also)

*vi httpd.yaml* (for httpd pod) <https://raw.githubusercontent.com/kubernetes/website/main/content/en/examples/pods/simple-pod.yaml>

apiVersion: v1 # see https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.13/

kind: Pod # type of object that's defined in this file

metadata:

name: pod-httpd # the name displayed in the first column of 'kubectl get pods'

labels:

app: apache\_webserver # this tag is added to help this object to link to the service object.

spec:

containers:

- name: cntr-httpd # name of the container that will reside in the pod

image: httpd:latest # using the official apache image from docker hub, along with a tag

ports: # this bit is purely for informational purposes only and can be omitted.

- containerPort: 80 # what port the container will be listening on

**---**

**apiVersion: v1**

**kind: Pod**

**metadata:**

**name: pod-httpd**

**labels:**

**app: apache\_webserver**

**spec:**

**containers:**

**- name: cntr-httpd**

**image: httpd:latest**

**ports:**

**- containerPort: 80**

*kubectl apply -f httpd.yaml*

*kubectl get pods*

*kubectl get nodes*

**28-07-2022**

1. **Create a cluster and then create pod and write service object from which loadbalancer will automatically create and can access the nginx application**

pool - maybe a collection of worker nodes.

In gcp/quicklabs -> Open -> Kubernetes engine API enable it. Under Kubernetes Engine Service -> create a cluster ->GKE Standard -> No of nodes: 3, Enable Cluster autoscaler min:1, max:3 in Node pool. Node: e2-micro, boot size-10GB,keep other things default -> create it.

Click on connect and copy that cmnd paste in cloud shell

*kubectl get nodes*

write nginx-pod.yaml and same in some location

vi nginx-pod.yaml

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

go to that location from same cmd promt then, //if it is cmd

*kubectl apply -f nginx-pod.yaml*

*kubectl get pods* (nginx is running now)

*kubectl describe pod nginx* (to get all details we can see whether label is added and others)s

To access write service object, I’m naming as service.yaml

---  
apiVersion: v1  
kind: Service  
metadata:  
name: app-service  
spec:  
type: LoadBalancer  
selector:  
app: nginx-prod  
ports:  
- name: port  
targetPort: 80  
port: 80

*kubectl apply -f service.yaml*

*kubectl get svc*

*kubectl describe svc app-service*

copy Ip of LoadBalancer Ingress :80 in chrome 🡪 we get welcome to nginx (or)

Go to services and Ingress, click on that Endpoints we get welcome to nginx

Load balancer is automatically created by the help of service object.

if u are using replicas in deployment.yaml,

suppose replicas = 3, then 3 pods running on which nginx is there. if u delete anyone then it will create one new pod automatically.

Deployment.yaml will make sure that atleast one pod is running.

1. **Create Autopilot cluster and create pod and then deployment object and selector object of httpd**
2. create a cluster
3. apply deployment object
4. apply service object
5. try to access the application from service load balancer

apiVersion: apps/v1

kind: Deployment

metadata:

name: httpd-deployment

labels:

app: httpd

spec:

replicas: 1

selector:

matchLabels:

app: httpd

template:

metadata:

labels:

app: httpd

spec:

containers:

- name: httpd

image: httpd:2.4

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: deploy-service

spec:

type: LoadBalancer

selector:

app: httpd

ports:

- name: port

targetPort: 80

port: 80

29-07-2022

Create a GKE standard cluster and enable cluster autoscaler. exercise

we can edit few things like upgrade, enable vertical pod scaling, maintenance window if u do some maintenance in particular week days, enabling node auto provisioning etc. and most fields are not accessible.

1. **Create a cluster and deploy redis guestbook application**

<https://kubernetes.io/docs/tutorials/stateless-application/guestbook/>

Uncomment type loadbalancer in frontend service.

*kubectl apply -f redis-master.yaml* //deployment

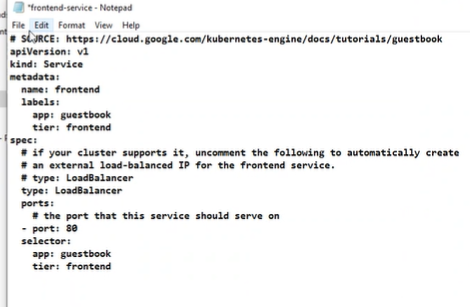
*kubectl apply -f redis-master-service.yaml*

*kubect apply -f redis-follower.yaml*

*kubectl apply -f redis-follower-service.yaml*

*kubectl apply -f frontend.yaml*

*kubectl apply -f frontend-service.yaml*



*kubectl get pods* //should show 6 : frontend-3, follower:2, leader:1

*kubectl get svc*  //follower, frontend, master, Kubernetes(Default)

From the service&ingress, click on master endpoint url we can access the application.

**Serverless** -

* Doesn’t mean ‘no servers’
* Don’t worry about infrastructure we just need to write the code (flexible scaling and automated high availability)
* Pay for use (zero request = zero cost) Pay for request not for servers

Example: Leve1 1(need to pay for no.of instances): **Google App Engine**, AWS Fargate , Level 2(pay for requests only): Google Functions. paas

App Engine (one project per application)

1. **Deploying simple project in App Engine**

App Engine -> Dashboard

Language: Python, standard/flexible

*git clone <githubLink>*

*cd <where main.py actually locates>*

*gcloud app deploy app.yaml*

In service it will create default service, click on it, we will get output.

1. **Deploy Angular project directly in default project, and deploy flask project in another project, for that we have to write --project <newly\_created\_project\_id> at the time of deployment.**

Create a new project click on dropdown above which is showing, there create a new project.

git clone <https://github.com/GoogleCloudPlatform/python-docs-samples>

cd <to angular/flask location>

gcloud app deploy app.yaml

1. link for angular project

<https://github.com/GoogleCloudPlatform/python-docs-samples/tree/main/appengine/standard/angular>

1. link for flask project

<https://github.com/GoogleCloudPlatform/python-docs-samples/tree/main/appengine/standard/flask/hello_world>

01-08-2022

**Google Cloud VPC**(virtual private cloud): used to create our own private network in cloud.

* Network traffic within a VPC is isolated(not visible) from all other Google cloud VPCs.
* we can control all the traffic coming in and going outside a vpc.
* VPC secures resources from unauthorized access. And it enables secure communication between your cloud resources.
* VPC is a global resource contains one or more subnets.
* Vpc create different subnets for public and private resources. Resources in public subnet can be accessed
* Ingress Rules: Incoming traffic from outside to GCP targets
* Egress Rules: Outgoing traffic to destination from GCP targets

1. **Create a private and public subnet in VPC networks and adding/deleting firewall rules**

In gcp -> vpc networks -> Name: prod-vpc, mode: custom, new subnet, name: public, asia-east1, IPV4(single-stack), range: 10.10.0.0/20 Done, New subnet: private, asia-east2, Ipv4, 10.11.0.0/20 Done. Firewall rules: allow all, -> regional ->MTU:1460 -> create. We can see under default.

Now create a VM with in the region asia-east1, allow default access, allow http, https traffic. In networks->Edit network interface select prod-vpc -> create it. Open it SSH in gcloud command/cmd prompt, it will successfully work.

In the vpc networks, if I delete ssh rule in prod-vpc, then we cannot access from cloud shell/cmd prompt. Then again add that firewall rule, Allow SSH custom,ip4 0.0.0.0., tcp-22 and check now. It will allow now.

1. **VPC network peering**

When we want the connection between two vpc’s then use VPC network peering.

1. Create 2 VPC networks

Create one vpc network prod-vpc asia-east1 public with Ipv4->10.10.0.0./20

Another vpc network with name non-prod public asia-east1, public-subnet->10.12.0.0/20

1. Create 2 VM instances under prod-vpc and nonprod in networking in advanced settings.

create a vm in prod-vpc(10.10.0.0/20) and vm in non-prod. copy internal Ip address of instance 1. Connect with instance-2 and use cmd ping <copied\_ip>

1. Create 2 vpc network peering connections

vpc-peering, non-prod,In project, prod-vpc, check import&export in exchange subnet routes.

vpc-peering-return, prod-vpc, non-prod, Check import&export

now ping<ip> will be done successfully.

02-08-2022

**Cloud VPN : (**to provide more security to connect our GCE(Google cloud Engine) to our own private network)

* It connects on-premise network to GCP network.
* Implemented using IPSec VPN tunnel(Internet Protocol Secure Tunnel)
* Traffic through internet(public). Traffic encrypted using Internet Key Exchange protocol.
* High availability VPN- has 2 tunnels, supports dynamic routing only, supports IPV4 and IPV6 traffic., **Classic VPN** has 1 tunnel, supports dynamic routing and static routing also, supports ipv4 traffic only.

In GCP Enable Compute Engine API, then go to-> Hybrid Connectivity -> Create a VPN -> High Availability

Hybrid connectivity: Vpn setup wizard -> HA -> gateway name: onprem, network: default, asia-east1, IPv4. Add vpn , peer VPN gateway name: onpremise, 2 interfaces, 172.31.254.1, 172.31.254.2. create 2 tunnels., Create a pair of VPN, Create a router onpremrouter, Google ASN: (should be between 64512 to 65534)create 2 tunnels, tunnel-1, click on generate and copy. create another tunnel-2, create and continue. Bgp session name:bgp-1, give peer ASN, same for bgp-2, create it. Ok.

**Load Balancer:** fully distributed, software defined, managed service for all your traffic. It used to distribute the load across number of servers.

Distributes user traffic across instances of an application in single region or multiple regions. It enables High availability, Auto scaling, resiliency.

In GCP-> Load balancing : we have http(S), tcp, udp load balancing.

1. **How to create a HTTP(S) load balancer and check how its balancing the load**

[Setting up a global external HTTP(S) load balancer with a Compute Engine backend | Load Balancing | Google Cloud](https://cloud.google.com/load-balancing/docs/https/setup-global-ext-https-compute) -> use this steps as reference

<https://www.cloudskillsboost.google/focuses/12007?parent=catalog>

use above lab link.

1. Create a instance template -> Name: lb-backend-template, add below script under management.

#! /bin/bash

sudo apt-get update

sudo apt-get install apache2 -y

sudo a2ensite default-ssl

sudo a2enmod ssl

sudo vm\_hostname="$(curl -H "Metadata-Flavor:Google" \

http://169.254.169.254/computeMetadata/v1/instance/name)"

sudo echo "Page served from: $vm\_hostname" | \

tee /var/www/html/index.html

sudo systemctl restart apache2

Network-tags: allow-health-check

1. Create an instance group -> Name: lb-backend-example, use the template, single zone: asia-east1, Select the created instance template, Min: 2 and Max:4. ->create.

Edit lb-backend-example: In port mapping: http, 80

1. create firewall rule -> Name: fw-allow-health-check, target tags: allow-health-check, Ipv4 ranges: 130.211.0.0/22 and 35.191.0.0/16 TCP-80.
2. Go to Ip address -> Reserve a static Ip address: name: lb-ipv4-1 -> Submit.
3. Create a load balancer: http -> web-map-http, Frontend: Name: web-map-http, in Ip address:lb-ipv4-1, Backend: create a backend service: Name: web-backend-service, Instance group: lb-backend-example, Health Check: create-> http-basic-check , protocol: http ->save ->Enable logging ->save ->Create.
4. Copy the ip of load balancer frontend, and put in browser to check accessibility. Here load is balancing by using 2 vms. for the same request sometimes it may use vm1 or vm2.
5. Here the script is giving only page served from, so open the respective instances from SSH, and change the script like page served from VM1 by giving cmd *sudo vi /var/www/html/index.html*

**DNS (Domain Name System)**

The Domain Name System (DNS) turns domain names into IP addresses, which browsers use to load internet pages. Every device connected to the internet has its own IP address, which is used by other devices to locate the device. Ex: google.com to 172.250.192.78

In GCP -> Enable Cloud DNS API.

Create a VM instance

In Network Services -> Cloud DNS -> Create zone: public, exam-zone,examzone.com -> create.

Create record set -> DNS name: http:app, Record Type: A.

**Storage**

1. The type of Storage of our hard disk is **Block Storage**
2. U have created a file share to share a set of files with your colleagues in a enterprise then the type of storage u use is **File Storage**

**Share an SSD persistent disk in multi-writer mode between VM instances**

1. Disk Creation for Multi-writer mode: gcloud beta compute disks create ssddisk --size 200 --type pd-ssd --multi-writer --zone=us-central1-a instance size h2 only allowing this. But we can’t do this in quicklabs.
2. Create 2 vm instances in same Zone.
3. Attach disk to VM: gcloud compute instances attach-disk instance-1 --disk ssddisk --zone=us-central1-a
4. **Adding a disk to VM and mounting filesystem**
5. Create a VM instance: In Disks: Add new disk ->save ->Create. (leave everything default)
6. In that instance click on SSH
7. sudo su
8. Install fdisk: apt-get -y install fdisk
9. list the disks attached to VM: lsblk -> we can see no mountpoints under sdb.
10. fdisk /dev/sdb n:to create new partition, w: for save and quit
11. To create a file system(ext2/3/4)on Linux system: mkfs -t ext4 /dev/sdb
12. Create a directory named test: mkdir /test
13. Mounting /dev/sdb file system on test directory: mount -t ext4 /dev/sdb /test
14. To view detailed output: df -kh

04-08-2022

**Storage - (Objects and Bucket)**

In GCP -> Cloud Storage -> Browser -> bucket -> Name: ex, standard -> create

upload files: index.html and click on uploaded file , from the authenticated url we can access it from internet.

we can upload images, any type of files etc.

Objects are stored in buckets, bucket names are globally unique. Max object size is

**Cloud Storage :**

Different kinds of data can be stored(media files, archives, packages, logs, backups). It is designed for durability.

|  |  |  |
| --- | --- | --- |
| Storage Class | Min Storage Duration | Use Case/ Best for what data |
| Standard Storage Class-more cost | None | Frequently used data/short-term storage |
| Nearline Storage Class | 30 days | Best for data read/modify once a month |
| Coldline Storage Class | 90 days | best for data read/modify once a quarter |
| Archive Storage Class-less cost | 365days | Less than once a year/Long term preservation |

Object Versioning - used to keep the older versions of our file and we can restore it back. we can keep limit as well. Give Object versioning at the time of bucket creation itself.

Create Bucket -> give unique bucket name -> Choose how to protect object data: select object versioning -> create. Then load one file, if u update/modify/delete, it will store all the versions. try with loading a notepad file with some data, and update that data and upload the same file again(overwrite) to see how it works actually.

Open the created bucked -> Under life cycle -> create a rule ->life cycle rule - we can keep rules like delete the files after these many days etc. setting storage to archive and based on age 180 days.

06-08-2022

**Database:**

* Databases provide organized and persistent storage for your data.
* Database Snapshots, Add transactions, Stand By. - Availability and Durability increases.
* **RPO(Recovery Point Objective)**: Maximum acceptable period of data loss. RPO=backup time.
* **RTO(Recovery Time Objective):** Maximum acceptable downtime.
* **Read Replicas** reduces the load on the master databases. Connect reporting and analytical applications to read replicas.
* **Consistency:** To ensure data in multiple databases are updated simultaneously.
* Strong Consistency - Synchronous replication to all replicas
* Eventual Consistency - Asynchronous replication.(used when scalability is more important than data integrity).
* **Relational Databases:** predefined schema with tables and relationships. Very strong transactional capabilities. used for OLTP(online transaction processing, OLAP(online analytics processing)
* **OLTP:** Applications where large number of users make large number of small transactions. (Cloud SQL, Cloud Spanner). use

**Cloud SQL:** Fully Managed relational database service, configure your needs and don’t worry about managing the database. Supports MySQL, PostgreSQL and SQL server. Provides high availability.

First Enable Compute Engine API and Cloud SQL Admin API

In GCP -> SQL -> Create an Instance -> Select MySQL ->Instance Id: mysql, No Password, DB version:8.0, Start with Development, Choose Single Zone, Custom:1 memory:4, Disk 20GB Enable automatic storage, connections: public IP -> Create a database.

Click on created instance -> Open cloud shell

*gcloud config set project <Project\_ID> //*if u open cloud shell outside the instance.s

*gcloud sql connect mysql --user=root --quiet*

mysql promt will open now.

*CREATE DATABASE <db\_nameEX:ace\_exam\_book>*

*USE <db\_name>*

*CREATE TABLE <tb\_nameEX:books> (title VARCHAR(255), num\_chapters INT, entity\_id INT NOT NULLAUTO\_INCREMENT, PRIMARY KEY(entity\_id));*

*INSERT INTO books(title, num\_chapters,entity\_id)VALUES (‘Architecture Exam Study Gide’,18,1)*

SELECT \* from books; //to get books table

08-08-2022

**Cloud Spanner**

If u need high volume of data, if u need high scalability and high availability(99.9%) then use Cloud Spanner instead of cloud SQL.(Availability is 99.5%).

Reference Link to do the lab:

<https://cloud.google.com/spanner/docs/quickstart-console>

**NO SQL Databases**

>Cloud Firestore - Managed *serverless* NOSQL document database Designed for transactional mobile and web applications, recommended for small to medium databases.

<https://www.cloudskillsboost.google/focuses/941?parent=catalog> ->for reference to do lab

>Cloud BigTable - Managed Scalable NOSQL wide column database, *Not serverless*(need to create instances), Recommend for large databases and large analytical workloads.

<https://cloud.google.com/bigtable/docs/create-instance-write-data-cbt-cli> ->reference to do lab

**OLAP**

BigQuery - Datawarehouse

Exabyte scale modern dataware housing solution from GCP. Query external data sources without storing data in BigQuery. (cloud storage, Cloud SQL, BigTable)

<https://www.cloudskillsboost.google/focuses/1145?parent=catalog> -> for reference.

09-08-2022

**GCP Marketplace:** To create any standard application and deploy directly.

<https://rpsconsulting.qwiklabs.com/classrooms/497/labs/4739> ->reference link

**Deployment Manager:** Same application in 5 environments, the manual process becomes tough.

It automates deployment and modification of google cloud resources in a controlled and predictable way.

All configuration is defined in a simple text file - YAML.

<https://cloud.google.com/deployment-manager/docs/manage-cloud-resources-deployment> ->ref link.

**Terraform**

Terraform is an open-source infrastructure as code (IaC) software tool that allows DevOps engineers to programmatically provision the physical resources an application requires to run. Infrastructure as code is an IT practice that manages an application's underlying IT infrastructure through programming.

file extension: .tf

we can simply modify/create resources using terraform.

Terraform Workflow:- write -> init -> validate -> plan -> apply -> destroy

<https://registry.terraform.io/providers/hashicorp/google/4.31.0>

<https://www.cloudskillsboost.google/focuses/15842?parent=catalog>

<https://rpsconsulting.qwiklabs.com/classrooms/497/labs/4744> ->for reference

Terraform Module: By using this, we can create combination of multiple resources.

1. **Create a VPC network and create vm(e2-small) with subnet (Ip range:10.1.0.0/20) using terraform.**
2. Open editor from Cloud shell and create below files

network-main.tf

# create VPC

resource "google\_compute\_network" "vpc" {

  name                    = "kopicloud-vpc"

  auto\_create\_subnetworks = "false"

  routing\_mode            = "GLOBAL"

}

# create public subnet

resource "google\_compute\_subnetwork" "network\_subnet" {

  name          = "kopicloud-subnet"

  ip\_cidr\_range = "10.1.0.0/20"

  network       = google\_compute\_network.vpc.name

}

main.tf

terraform {

  required\_providers {

    google = {

      source = "hashicorp/google"

    }

  }

}

provider "google" {

  version = "3.5.0"

  project = "<project\_Id>"

  region  = "us-central1"

  zone    = "us-central1-c"

}

resource "google\_compute\_instance" "vm\_instance\_public" {

  name         = "kopicloud-vm"

  machine\_type = "e2-small"

  hostname     = "kopicloud-vm.kopicloud.com"

  tags         = ["ssh","http"]

  boot\_disk {

    initialize\_params {

      image = "ubuntu-os-cloud/ubuntu-1804-lts"

    }

  }

  network\_interface {

    network       = google\_compute\_network.vpc.name

    subnetwork    = google\_compute\_subnetwork.network\_subnet.name

    access\_config { }

  }

}

1. In cloud shell run below cmnds:

terraform init

terraform apply (yes)

we can see the instance and subnet in console…

12-08-2022

**Cloud Logging and Monitoring:**

**Metrics**

1. **Latency** measures how long it takes a particular part of a system to return a result. Latency should be as low as possible.
2. **Traffic** measures how many requests are reaching your system.
3. **Saturation** measures how close to capacity a system is.
4. **Errors** are events that measure system failures or other issues.

**Cloud Logging**

Cloud Logging is part of the Operations suite of products in Google Cloud. It includes storage for logs, a user interface called the Logs Viewer, and an API to manage logs programmatically. Use Cloud Logging to read and write log entries, search and filter your logs, export your logs, and create logs-based metrics. Cloud Logging is a fully managed service that performs at scale and can ingest application and platform log data, as well as custom log data from GKE environments, VMs, and other services inside and outside of Google Cloud. Cloud Logging provides two kinds user-defined logs-based metrics - **Counter** and**Distribution.**

**Counter metrics:** Counter metrics count the number of log entries matching an advanced logs filter.

**Distribution metrics**: Distribution metrics accumulate numeric data from log entries matching a filter, and perform mathematical calculations against them.

<https://www.cloudskillsboost.google/focuses/10911?parent=catalog> ->lab ref link

**Cloud Monitoring:** Cloud Monitoring collects metrics, events, and metadata from Google Cloud, Amazon Web Services (AWS), hosted uptime probes, and application instrumentation.

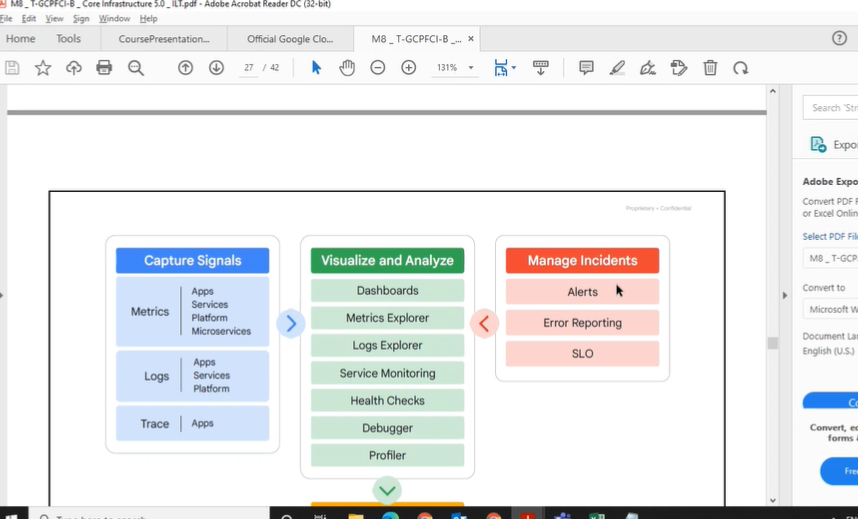
The **Ops Agent** collects logs and metrics on Compute Engine instances, sending your logs to Cloud Logging and your metrics to Cloud Monitoring.

**SLC (Service level indicators)** carefully selected monitoring metrics that measure one aspect pf service’s reliability.

**SLO(Service Level Objective)** combines a service level indicator with a target reliability (nearly 99.9%).

**SLA(Service Level Agreement)** are commitments made to your customers that your systems and applications will have only a certain amount of “Down Time”.

<https://www.cloudskillsboost.google/focuses/10599?parent=catalog> ->lab ref link



<https://drive.google.com/drive/folders/1VRJFQMDm3eVTEtSg5OVH30CtWDuclwuo?usp=sharing>

<https://docs.google.com/spreadsheets/d/1CVaaPuUbvBkNjOvaKrQ98Gvn2xYxmS1P/edit#gid=750328052>