






Virtual Machines

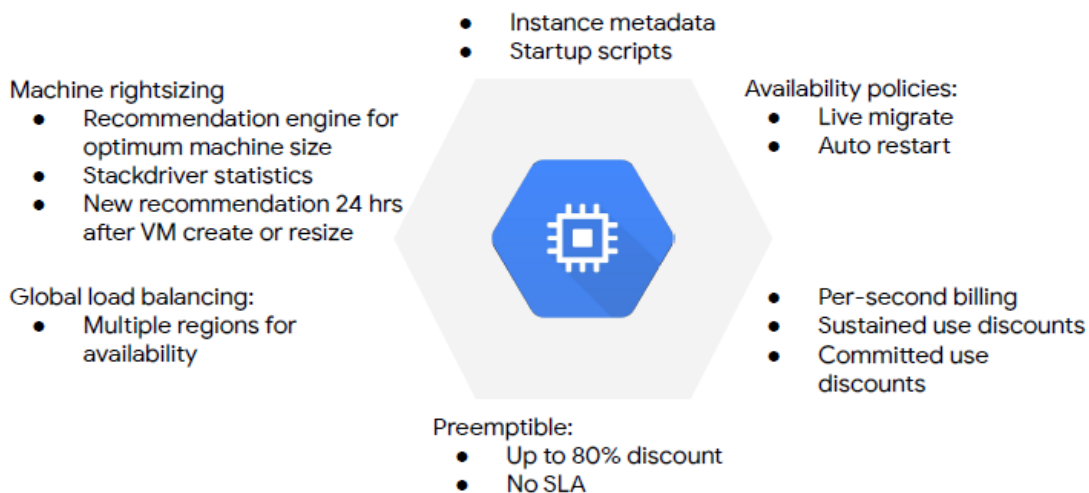
	 Compute Engine	 Kubernetes Engine	 App Engine Standard	 App Engine Flexible	 Cloud Functions
Language support	Any	Any	Python Node.js Go Java PHP	Python Node.js Go Java PHP Ruby .NET Custom Runtimes	Python Node.js Go
Usage model	IaaS	IaaS PaaS	PaaS	PaaS	Microservices Architecture
Scaling	Server Autoscaling	Cluster	Autoscaling managed servers		Serverless
Primary use case	General Workloads	Container Workloads	Scalable web applications Mobile backend applications		Lightweight Event Actions



Compute Engine

- Infrastructure as a Service (IaaS)
- Predefined or custom machine types:
 - vCPUs (cores) and Memory (RAM)
 - Persistent disks: HDD, SSD, and Local SSD
 - Networking
 - Linux or Windows
-

Compute Engine features



Disk Storage :

- **Standard** (standard spinning hard disk drives or HDDs): Local SSDs have even higher throughput and lower latency than SSD persistent disks, because they are attached to the physical hardware.
- **SSD (Solid state drive)** : SSDs are designed to give you a higher number of IOPS per dollar versus standard disks, which will give you a higher amount of capacity for your dollar.
- **Local SSD** : Local SSDs have even higher throughput and lower latency than SSD persistent disks, because they are attached to the physical hardware. Local SSD is used as a swap disk, just like you would do if you want to create a ramdisk, but if you need more capacity, you can store those on a local SSD.

Networking :

- Default, custom networks
- Inbound/outbound firewall rules
 - IP based
 - Instance/group tags
- Regional HTTPS load balancing
- Network load balancing
- Does not require pre-warming
- Global and multi-regional subnetworks

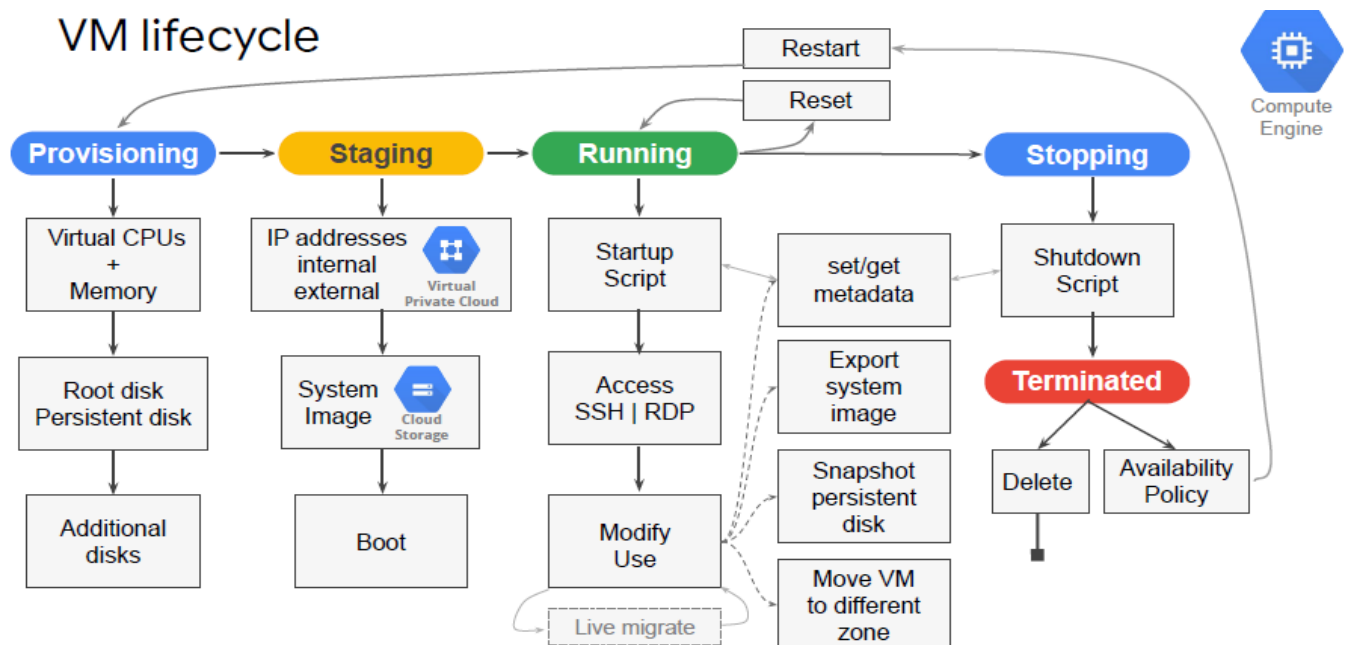
VM Access :

Linux: SSH

- SSH from GCP Console
- CloudShell via Cloud SDK
- SSH from computer or third-party client and generate key pair
- Requires firewall rule to allow tcp:22

Windows: RDP

- RDP clients
- Powershell terminal
- Requires setting the Windows password
- Requires firewall rule to allow tcp:3389



Changing VM state from running

	methods	Shutdown Script time	state
reset	console, gcloud, API, OS	no	remains running
restart	console, gcloud, API, OS	no	terminated → running
reboot	OS: <code>sudo reboot</code>	~90 sec	running → running
stop	console, gcloud, API	~90 sec	running → terminated
shutdown	OS: <code>sudo shutdown</code>	~90 sec	running → terminated
delete	console, gcloud, API	~90 sec	running → N/A
preemption	automatic	~30 sec	N/A

Availability policy: Automatic changes

- Called "scheduling options" in SDK/API

Automatic restart

- Automatic VM restart due to crash or maintenance event
- Not preemption or a user-initiated terminate

On host maintenance

- Determines whether host is live-migrated or terminated due to a maintenance event. Live migration is the default.

Live migration

- During maintenance event, VM is migrated to different hardware without interruption.
- Metadata indicates occurrence of live migration.

Stopped (Terminated) VM

- No charge for stopped VM
 - Charged for attached disks and IPs
- Actions
 - Change the machine type.
 - Add or removed attached disks; change auto-delete settings.
 - Modify instance tags.
 - Modify custom VM or project-wide metadata.
 - Remove or set a new static IP.
 - Modify VM availability policy.
 - Can't change the image of a stopped VM.

Creating a VM

1

console.google.com

2

command line
including Cloudshell

3

REST API

Many VM options

- Project
- Region
- Zone
- Subnetwork
- Machine type
- Disk options
- Image
- IP options

VM Machine Types

- Predefined Machines:**

Standard Machine : 3.75 GB Memory per vCPU

Machine name	vCPUs	Memory (GB)	Max # PD	Max total PD size
n1-standard-1	1	3.75	128	64 TB
n1-standard-2	2	7.50		
n1-standard-4	4	15		
n1-standard-8	8	30		
n1-standard-16	16	60		
n1-standard-32	32	120		
n1-standard-64	64	240		
n1-standard-96	96	360		

Ideal for tasks that have a balance of CPU and memory needs.

High Memory Machine : 6.5 GB Memory per vCPU

Machine name	vCPUs	Memory (GB)	Max # PD	Max total PD size
n1-highmem-2	2	13	128	64 TB
n1-highmem-4	4	26		
n1-highmem-8	8	52		
n1-highmem-16	16	104		
n1-highmem-32	32	208		
n1-highmem-64	64	416		
n1-highmem-96	96	624		

ideal for task that require more memory relative to vCPUs

High-CPU Machine : 0.9 GB Memory per vCPU

Machine name	vCPUs	Memory (GB)	Max # PD	Max total PD size
n1-highcpu-2	2	1.80	128	64 TB
n1-highcpu-4	4	3.60		
n1-highcpu-8	8	7.20		
n1-highcpu-16	16	14.4		
n1-highcpu-32	32	28.8		
n1-highcpu-64	64	57.6		
n1-highcpu-96	96	86.4		

ideal for tasks that require more vCPUs than memory

Memory-Optimized Machine : > 14 GB Memory per vCPU

Machine name	vCPUs	Memory (GB)	Max # PD	Max total PD size
n1-ultramem-40	40	961	128	64 TB
n1-ultramem-80	80	1922		
n1-megamem-96	96	1433.6		
6				
n1-ultramem-160	160	3844		

Compute-Optimized Machine : Highest performance per vCPU (3.8Ghz sustained all-core turbo)

Machine name	vCPUs	Memory (GB)	Max # PD	Max total PD size
c2-standard-4	4	16	128	64 TB
c2-standard-8	8	32		
c2-standard-16	16	64		
c2-standard-30	30	120		
c2-standard-60	60	240		

ideal for compute-intensive workloads

Shared Core Machine : 0.9 GB Memory per vCPU

Machine name	vCPUs	Memory (GB)	Max # PD	Max total PD size
f1-micro	0.2	0.60	16	3 TB
g1-small	0.5	1.70		

- **Custom Machines:**
 - You specify the amount of memory and number of vCPUs.

When to select custom:

- Requirements fit between the predefined types
- Need more memory or more CPU

Customize the amount of memory and vCPU for machine:

- Either 1 vCPU or even number of vCPU
- 0.9 GB per vCPU, up to 6.5 GB per vCPU
- 0.9 (default)
 - Total memory must be multiple of 256 MB

<p>Pricing</p> <ul style="list-style-type: none"> ● Per-second billing, with minimum of 1 minute ○ vCPUs, GPUs, and GB of memory ● Resource-based pricing: <ul style="list-style-type: none"> ○ Each vCPU and each GB of memory is billed separately ● Discounts: <ul style="list-style-type: none"> ○ Sustained use ○ Committed use ○ Preemptible VM instances ● Recommendation Engine <ul style="list-style-type: none"> ○ Notifies you of underutilized instances ● Free usage limits 	<p>Sustained Usage Discount</p> <table border="1" data-bbox="841 520 1377 856"> <thead> <tr> <th>Usage Level (% of month)</th><th>% at which incremental is charged</th></tr> </thead> <tbody> <tr> <td>0% - 25%</td><td>100% of base rate</td></tr> <tr> <td>25% - 50%</td><td>80% of base rate</td></tr> <tr> <td>50% - 75%</td><td>60% of base rate</td></tr> <tr> <td>75% - 100%</td><td>40% of base rate</td></tr> </tbody> </table> <p>Up to 30% discount for instances that run the entire month</p>	Usage Level (% of month)	% at which incremental is charged	0% - 25%	100% of base rate	25% - 50%	80% of base rate	50% - 75%	60% of base rate	75% - 100%	40% of base rate
Usage Level (% of month)	% at which incremental is charged										
0% - 25%	100% of base rate										
25% - 50%	80% of base rate										
50% - 75%	60% of base rate										
75% - 100%	40% of base rate										
<p>Committed Usage Discount</p> <ul style="list-style-type: none"> • If your workload is stable and predictable, you can purchase a specific number of vCPUs and memory for a discount off of normal prices in return for committing to a usage term of 1 year or 3 years. • The discount is up to 57% for most machine types or custom machine types. • The discount is up to 70% for memory-optimized machine types. 	<p>Preemptible VM</p> <ul style="list-style-type: none"> • Lower price for interruptible service (up to 80%) • VM might be terminated at any time <ul style="list-style-type: none"> ○ No charge if terminated in the first 10 minutes ○ 24 hours max ○ 30-second terminate warning, but not guaranteed <ul style="list-style-type: none"> ■ Time for a shutdown script • No live migrate; no auto restart • You can request that CPU quota for a region be split between regular and preemption <ul style="list-style-type: none"> ○ Default: preemptible VMs count against region CPU quota 										

Shielded VMs

- Shielded VM's offer verifiable integrity of your VM instances, so you can be confident that your instances haven't been compromised by boot- or kernel-level malware or rootkits.
- Shielded VM's verifiable integrity is achieved through the use of Secure Boot, virtual trusted platform module or **vTPM-enabled Measured Boot**, and integrity monitoring.

Images (Boot Disk Image):

<p>When creating a virtual machine, you can choose the boot disk image. This image includes :</p> <ul style="list-style-type: none">• Boot loader• Operating system• File system structure• Pre-configured software• Other customizations	<ul style="list-style-type: none">• Public base images<ul style="list-style-type: none">◦ Google, third-party vendors, and community; Premium images (p)◦ Linux (CentOS, CoreOS, Debian, RHEL(p), SUSE(p), Ubuntu, openSUSE, and FreeBSD)◦ Windows (Windows Server 2019(p), 2016(p), 2012-r2(p), SQL Server pre-installed on Windows(p))• Custom images<ul style="list-style-type: none">◦ Create new image from VM: pre-configured and installed SW◦ Import from on-prem, workstation, or another cloud <p>Management features: image sharing, image family, deprecation</p>
<p>Boot Disk</p> <ul style="list-style-type: none">• VM comes with a single root persistent disk.<ul style="list-style-type: none">◦ Image is loaded onto root disk during first boot:◦ Bootable: you can attach to a VM and boot from it.◦ Durable: can survive VM terminate.• Some OS images are customized for Compute Engine.• Can survive VM deletion if “Delete boot disk when instance is deleted” is disabled.	<p>Persistent Disk</p> <ul style="list-style-type: none">• Network storage appearing as a block device• Attached to a VM through the network interface• Durable storage: can survive VM terminate• Bootable: you can attach to a VM and boot from it• Snapshots: incremental backups• Performance: Scales with size• Features• HDD (magnetic) or SSD (faster, solid-state) options• Disk resizing: even running and attached!• Can be attached in read-only mode to multiple VMs• Encryption keys:<ul style="list-style-type: none">◦ Google-managed◦ Customer-managed◦ Customer-supplied• You can also attach a disk in read-only mode to multiple VMs.
<p>Local SSD</p> <ul style="list-style-type: none">• Local SSD disks are physically attached to a VM• More IOPS, lower latency, and higher throughput than persistent disk.• you can attach up to 8 local SSD disks with 375 GB each, resulting in a total of 3 TB.• Data survives a reset, but not a VM stop or terminate• VM-specific: cannot be reattached to a different VM	<p>RAM Disk</p> <ul style="list-style-type: none">• Tmpfs : use tmpfs if you want to store data in memory• Faster than local disk, slower than memory• ○ Use when your application expects a file system structure and cannot directly store its data in memory• ○ Fast scratch disk, or fast cache• Very volatile; erase on stop or restart• May need a larger machine type if RAM was sized for the application• Consider using a persistent disk to back up RAM disk data

Summary of Disk Options :

	Persistent disk HDD	Persistent disk SSD	Local SSD disk	RAM disk
Data redundancy	Yes	Yes	No	No
Encryption at rest	Yes	Yes	Yes	N/A
Snapshotting	Yes	Yes	No	No
Bootable	Yes	Yes	No	Not
Use case	General, bulk file storage	Very random IOPS	High IOPS and low latency	low latency and risk of data loss

- Persistent disks can be rebooted and snapshotted
- local SSDs and RAM disks are ephemeral
- Choose a persistent HDD disk when you don't need performance but just need capacity.
- If you have high performance needs, start looking at the SSD options.
- The persistent disks offer data redundancy because the data on each persistent disk is distributed across several physical disks.
- Local SSDs provide even higher performance, but without the data redundancy.
- RAM disks are very volatile, but they provide the highest performance.
- ***For the Standard, High Memory, High-CPU, Memory-optimized, and Compute-optimized machine types, you can attach up to 128 disks.***
- ***For the Shared-core machine type, you can attach up to 16 disks.***

Compute Engine Actions :

- **Storing and retrieving instance metadata** is a very common Compute Engine action.
 - Every VM instance stores its metadata on a metadata server.
 - The metadata server is particularly useful in combination with startup and shutdown scripts.
- **Resize persistent disk :**
 - you can grow disks in size, you can never shrink them
- **Snapshot:**
 - Back up critical data
 - Migrate data between zones
 - Transfer to SSD to improve performance
 - **Persistent disk snapshots**
 - **Snapshot is not available for local SSD, available only to persistent disks**
 - Snapshots are incremental and automatically compressed
 - Snapshots can be restored to a new persistent disk, allowing for a move to a new zone
 - Snapshot doesn't back up VM metadata, tags, etc
 - Creates an incremental backup to Cloud Storage
 - Not visible in your buckets; managed by the snapshot service
 - Consider cron jobs for periodic incremental backup
- **Move an instance to a new zone :** gcloud compute instances move
 - If you move your instance within the same region, you can automate the move by using the gcloud compute instances move command.
 - Automated process (moving within region):
 - gcloud compute instances move
 - Update references to VM; not automatic
 - Manual process (moving between regions):
 - Snapshot all persistent disks on the source VM.

- Create new persistent disks in destination zone restored from snapshots.
- Create new VM in the destination zone and attach new persistent disks.
- Assign static IP to new VM.
- Update references to VM.
- Delete the snapshots, original disks, and original VM.

GCP API Management Services :

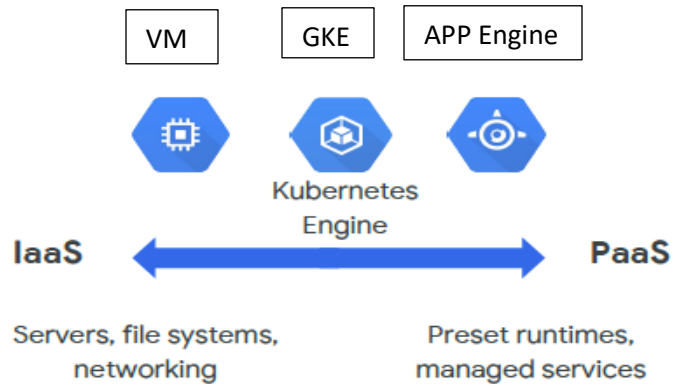
1. Google Cloud Endpoints

- Cloud Endpoints is a distributed API management system.
- It provides an API console, hosting, logging, monitoring, and other features to help you create, share, maintain, and secure your APIs.
- You can use Cloud Endpoints with any APIs that support the OpenAPI Specification, formerly known as the Swagger.
- Cloud Endpoints uses the distributed Extensible Service Proxy to provide low latency and high performance for serving even the most demanding APIs.
- Extensible Service Proxy is a service proxy based on NGINX.
- It runs in its own Docker container for better isolation and scalability.
- The proxy is containerized and distributed in the Container Registry and Docker registry, and can be used with App Engine, Kubernetes Engine, Compute Engine or Kubernetes.
- Expose your API using a RESTful interface
- Control access and validate calls with JSON Web Tokens and Google API keys
- Identify web, mobile users with Auth0 and Firebase Authentication

2. Apigee Edge

- Apigee Edge helps you secure and monetize APIs
- A platform for making APIs available to your customers and partners
- Contains analytics, monetization, and a developer portal

APP Engine & Containers & Kubernetes

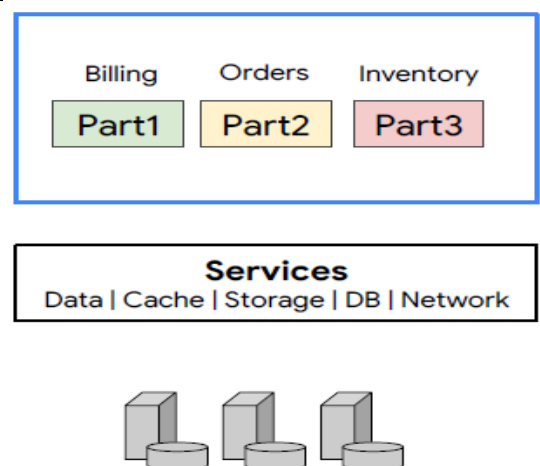


App Engine

- App Engine Standard environment
- App Engine Flexible environment

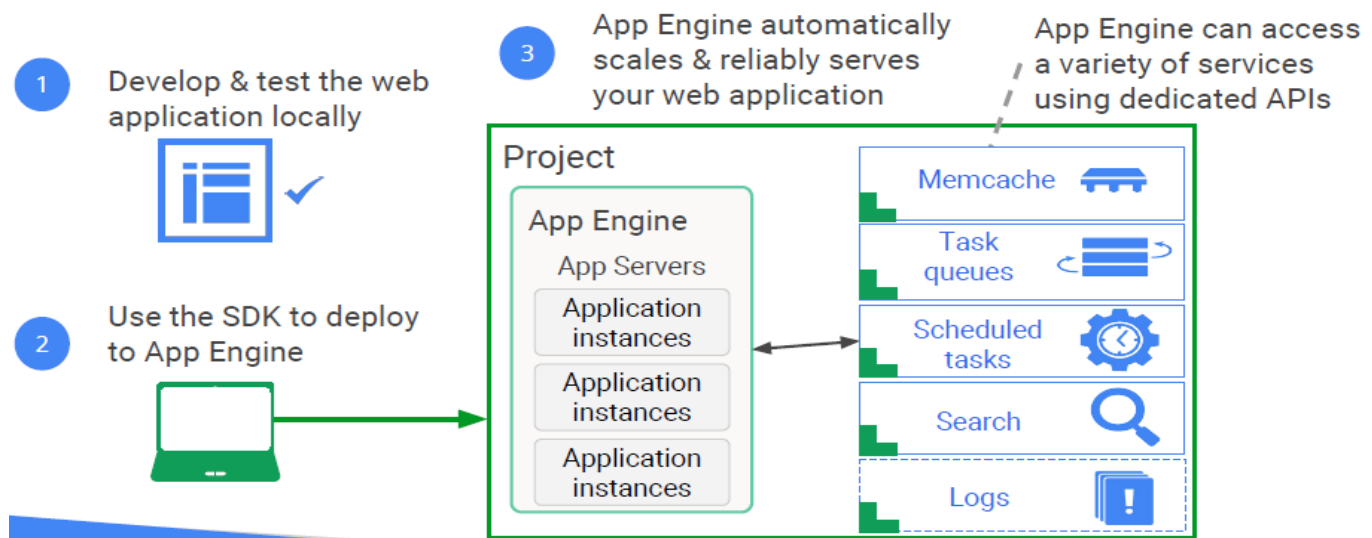
APP Engine :

- PaaS offering.
- App Engine you get access to programming services.
- All you do is write your code in self-contained workloads that use these services and include any dependent libraries.
- As demand for your app increases, the platform scales your app seamlessly and independently by workload and infrastructure.
- This scales rapidly but you won't be able to fine-tune the underlying architecture to save cost.
- App Engine makes deployment, maintenance, and scalability easy
- Suited for building scalable web applications and mobile backends
- App Engine provides you with built-in services and APIs such as NoSQL datastores, Memcached, load balancing, health checks, application logging, and a user authentication API, common to most applications.
- Upload your code and Google will manage your app's availability.
- There are no servers for you to provision or maintain.
- Security Scanner automatically scans and detects common web application vulnerabilities. It enables early threat identification and delivers very low false-positive rates.



App Engine Standard environment

- Easily deploy your applications
- Autoscale workloads to meet demand
- Economical : Free daily quota & Usage based pricing
- SDKs for development, testing and deployment
- The App Engine standard environment is based on container instances running on Google's infrastructure.
- Containers are preconfigured with one of several available runtimes (Java 7, Python 2.7, Go and PHP).
- Each runtime also includes libraries that support App Engine standard APIs.
- It includes the following features:
 - Persistent storage with queries, sorting, and transactions
 - Automatic scaling and load balancing
 - Asynchronous task queues for performing work outside the scope of a request
 - Scheduled tasks for triggering events at specified times or regular intervals
 - Integration with other Google cloud services and APIs
- **Requirements**
 - Specific versions of Java, Python, PHP, and Go are supported
 - **Your application must conform to sandbox constraints:**
 - No writing to local file system
 - All requests time out at 60 seconds
 - Third-party software installations are limited
- *Applications run in a secure, sandboxed environment, allowing the App Engine standard environment to distribute requests across multiple servers, and scaling servers to meet traffic demands.*
- *Your application runs within its own secure, reliable environment that is independent of the hardware, operating system, or physical location of the server.*
- **App Engine standard workflow: Web Applications**



- Each App Engine application runs in a GCP project.
- App Engine automatically provisions server instances and scales and load-balances them.
- Your application can make calls to a variety of services using dedicated APIs.

App Engine Flexible environment

- Build and deploy containerized apps with a click
- **No sandbox constraints**
- Can access App Engine resources
- Standard runtimes: Python, Java, Go, Node.js
- Custom runtime support: Any language that supports HTTP requests
- **Package your runtime as a Dockerfile**
- Your **application runs inside Docker containers on Google Compute Engine virtual machines (VMs)**.
- App Engine manages these Compute Engine machines for you.
- They're health-checked, healed as necessary, and you get to choose what geographical region they run in.
- Critical, backward-compatible updates to their operating systems are automatically applied.
- Microservices, authorization, SQL and NoSQL databases, traffic splitting, logging, search, versioning, security scanning, Memcached, and content delivery networks are all supported natively.
- you can use SSH to connect to every single VM and Docker container for debugging purposes and further customization.
- VM instances are restarted on a weekly basis. During restarts, Google's management services will apply any necessary operating system and security updates.
-

Standard Environment Vs Flexible Environment

	Standard Environment	Flexible Environment
<i>Instance startup</i>	Milliseconds	Minutes
<i>SSH access</i>	No	Yes (although not by default)
<i>Write to local disk</i>	No	Yes (but writes are ephemeral)
<i>Support for 3rd-party binaries</i>	No	Yes
<i>Network access</i>	Via App Engine services	Yes
<i>Pricing model</i>	After free daily use, pay per instance class, with automatic shutdown	Pay for resource allocation per hour; no automatic shutdown

Deploying Apps: Kubernetes Engine vs App Engine

	Kubernetes Engine	App Engine Flexible	App Engine Standard
<i>Language support</i>	Any	Any	Java, Python, Go, PHP
<i>Service model</i>	Hybrid	PaaS	PaaS
<i>Primary use case</i>	Container-based workloads	Web and mobile applications, container-based workloads	Web and mobile applications



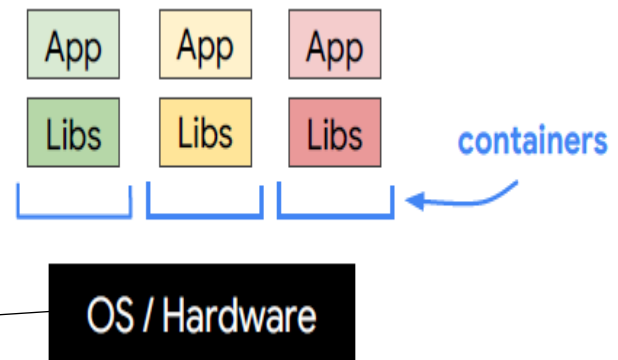
Toward managed infrastructure

Toward dynamic infrastructure

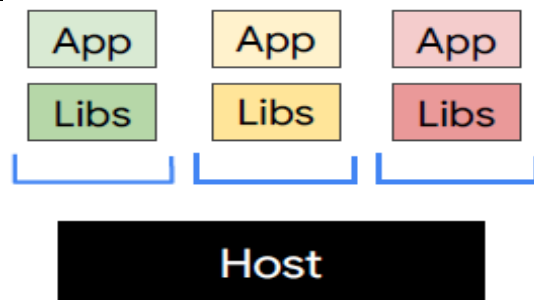
Containers :

- Idea of a **container** is to give you the independent scalability of workloads in PaaS and an abstraction layer of the OS and hardware in IaaS.
- What you get is an invisible box around your code and its dependencies, with limited access to its own partition of the file system and hardware.
- It only requires a few system calls to create and it starts as quickly as a process.

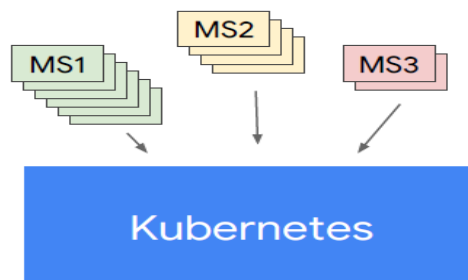
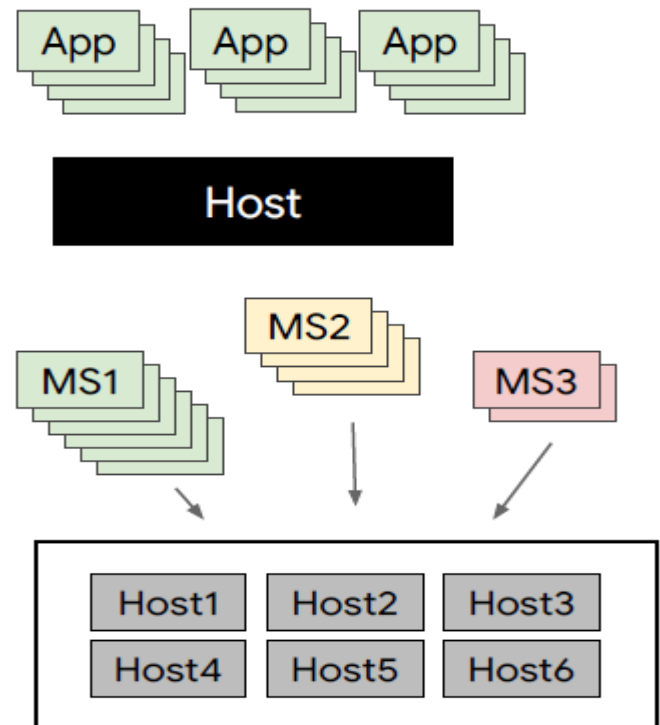
implements
container interfaces



- All you need on each host is an OS kernel that supports containers and a container runtime.
- In essence, you are virtualizing the OS. It scales like PaaS but gives you nearly the same flexibility as IaaS.
- With this abstraction, your code is ultra-portable, and you can treat the OS and hardware as a black box (**HOST**).
- you can go from development, to staging, to production, or from your laptop to the cloud, without changing or rebuilding anything.

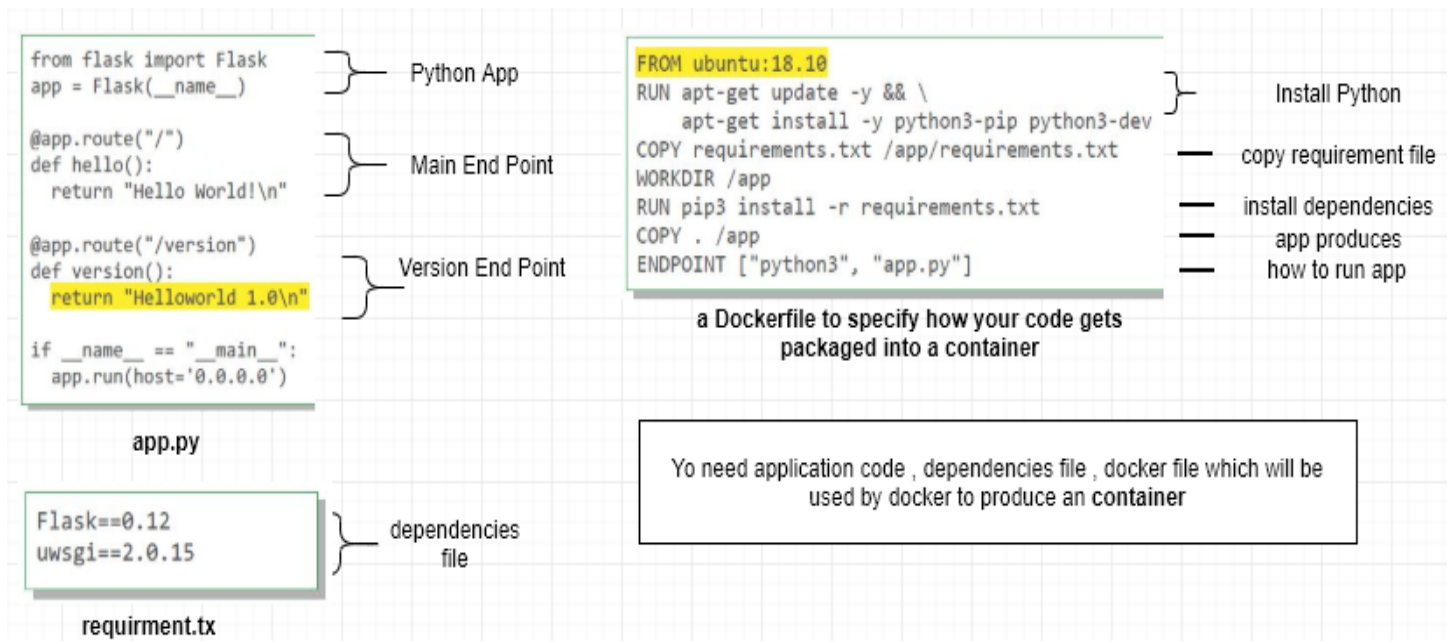


- If you want to scale, for example, a web server, you can do so in seconds and deploy dozens or hundreds of them (depending on the size or your workload) on a single host.
- That's a simple example of scaling one container running the whole application on a single host.
- You'll likely want to build your applications using lots of containers, each performing their own function like microservices.
- If you build them this way, and connect them with network connections, you can make them modular, deploy easily, and scale independently across a group of hosts.
- And the hosts can scale up and down and start and stop containers as demand for your app changes or as hosts fail.
- -----
- A tool that helps you do this well is **Kubernetes**.
- **Kubernetes** makes it easy to orchestrate many containers on many hosts, scale them as microservices, and deploy rollouts and rollbacks.



How to Build Container:

- **Kubernetes** makes it easy to orchestrate many containers on many hosts, scale them as microservices, and deploy rollouts and rollbacks.
- Use an open-source tool called **Docker** or **Google Container Builder** that defines a format for bundling your application, its dependencies, and machine-specific settings into a container.



Build & Run :

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

- Use the "**docker build**" command to build the container. This builds the container and stores it locally as a runnable image (**docker image**).
- You can save and upload the image to a container registry service and share or download it from there.
- Use the "**docker run**" command to run the image. This will package the application.
- Packaging the application involves :
 - application configuration
 - service discovery
 - managing updates
 - monitoring
 - These are the components of a reliable, scalable, distributed system.

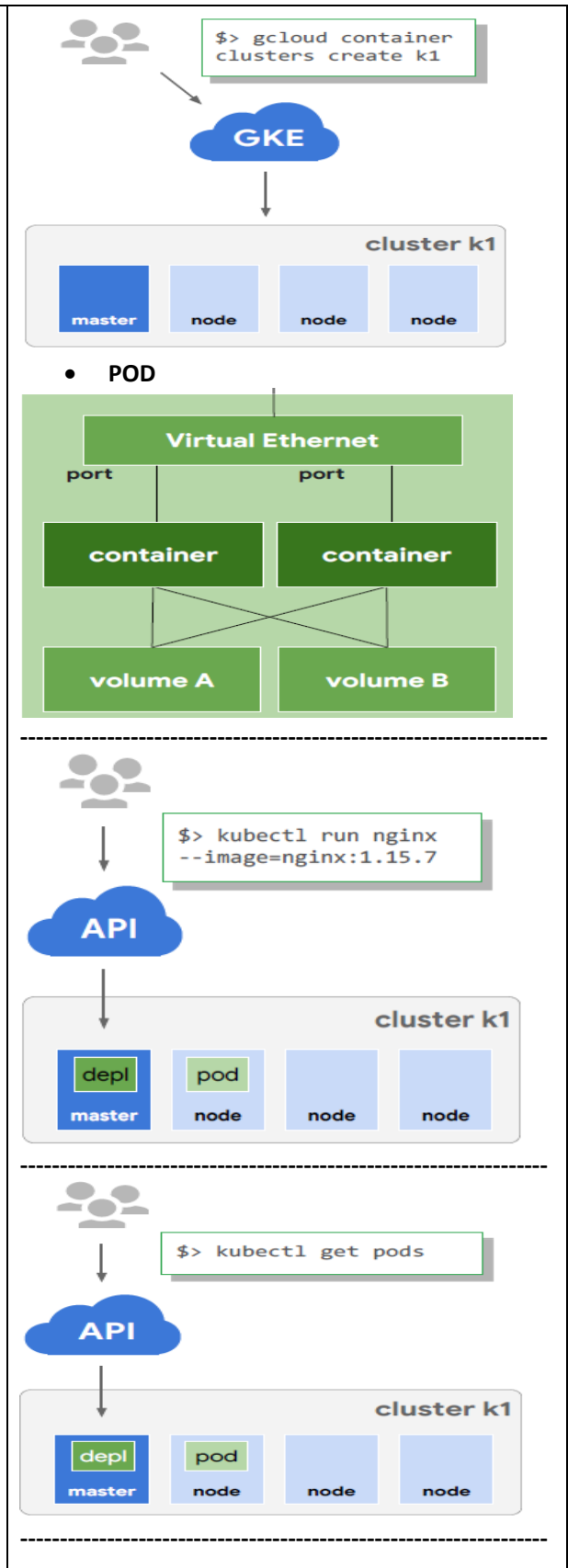
Kubernetes :

- Kubernetes is an **open-source orchestrator** that abstracts containers at a higher level so you can better manage and scale your applications.
- At the highest level, Kubernetes is a set of APIs that you can use to deploy containers on a set of **nodes** called a **cluster**.

- you've built a container; you'll want to deploy one into a cluster.
- System is divided into a set of **master** components that run as the control plane and a set of **nodes** that run containers.
- In Kubernetes, a node represents a computing instance, like a machine.
- In Google Cloud, nodes are virtual machines running in Compute Engine.
- Kubernetes can be configured, or you can bootstrap Kubernetes using **Kubernetes Engine** or (**GKE**).
- **GKE** is a hosted Kubernetes by Google.
- **GKE** clusters can be customized and they support different machine types, number of nodes, and network settings.
- To start Kubernetes on pre-created cluster K1
gcloud container clusters create k1
- **Deploy containers** on nodes using a wrapper around one or more containers called a **Pod**.
- **Pod** is the smallest unit in Kubernetes that you create or deploy.
- **Pod** represents a running process on your cluster as either a component of your application or an entire app.
- Generally, you only have one container per pod, but if you have multiple containers with a hard dependency, you can package them into a single pod and share networking and storage.
- **Pod** provides a unique network IP and set of ports for your containers, and options that govern how containers should run.
- **Containers inside a Pod** can communicate with one another using localhost and ports that remain fixed as they're started and stopped on different nodes.
- To **run a container in a Pod** in Kubernetes is to use the **kubectl run** command.
- This **starts a Deployment** with a container running in a Pod and the container inside the Pod is an image of the nginx server.
- **Deployment** represents a group of replicas of the same Pod and keeps your Pods running even when nodes they run on fail.
- To see the running Pods, run the command:
\$ kubectl get pods
- Pods in a Deployment are only accessible inside your GKE cluster. To make them publicly available, you can connect a load balancer to your Deployment by running the ***\$ kubectl expose*** command.

kubectl expose deployments nginx --port=80 --

type=LoadBalancer



- Kubernetes creates a **Service** with a fixed IP for your Pods
- GKE **attach an external load balancer** with a public IP address to that **Service** so others outside the cluster can access it".
- In GKE, the load balancer is created as a **Network Load Balancer**.
- Any client that hits that IP address will be routed to a Pod behind the Service, in this case there is only one--your simple nginx Pod.
- **Service** is an abstraction which defines a logical set of Pods and a policy by which to access them.
- As Deployments create and destroy Pods, **Pods get their own IP address**.
- But those addresses don't remain stable over time.
- *A Service groups a set of Pods and provides a stable endpoint (or fixed IP) for them.*
- For example, if you create two sets of Pods called frontend and backend, and put them behind their own Services, backend Pods may change, but frontend Pods are not aware of this. They simply refer to the backend Service.
- Run **\$ kubectl get services** command to get the public IP to hit the container remotely.

```
$> kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nginx	LoadBalancer	10.0.65.118	104.198.149.140	80/TCP	5m

- Scale a Deployment, run the **\$ kubectl scale** command . In this case, three Pods are created in your Deployment and they're placed behind the Service and share one fixed IP.
- Ex of how to autoscale the Deployment to between 10 and 15 Pods when CPU utilization reaches 80 percent.

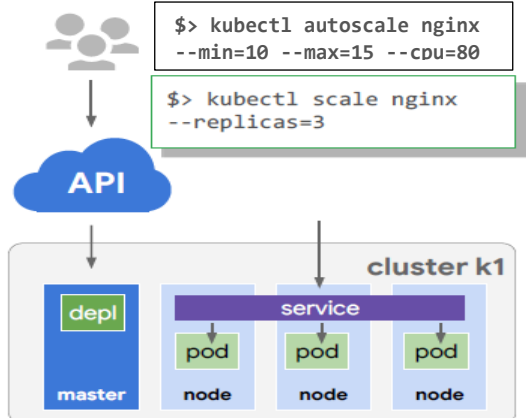
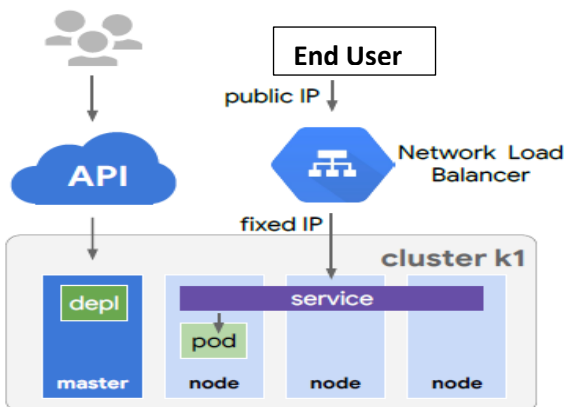
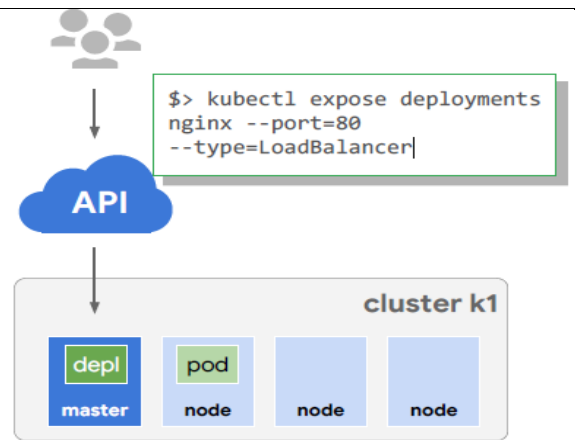
```
$> kubectl autoscale nginx --min=10 --max=15 --cpu=80
```

Instead of using all above imperative command to build & scale , we can do following

- Scale your Deployment using an existing **Deployment config file**
- To get the file, you can run a **\$ kubectl get pods** command
- It defines a **selector** field, so your Deployment knows how to group specific Pods as replicas, and you add a **label** to the Pod template, so they get selected.
- To run five replicas instead of three, all you do is update the Deployment config file. **replicas : 5**
- Run the **\$ kubectl apply command** to use the config file
- **\$> kubectl apply -f nginx-deployment.yaml**
- look at your replicas to see their updated state.

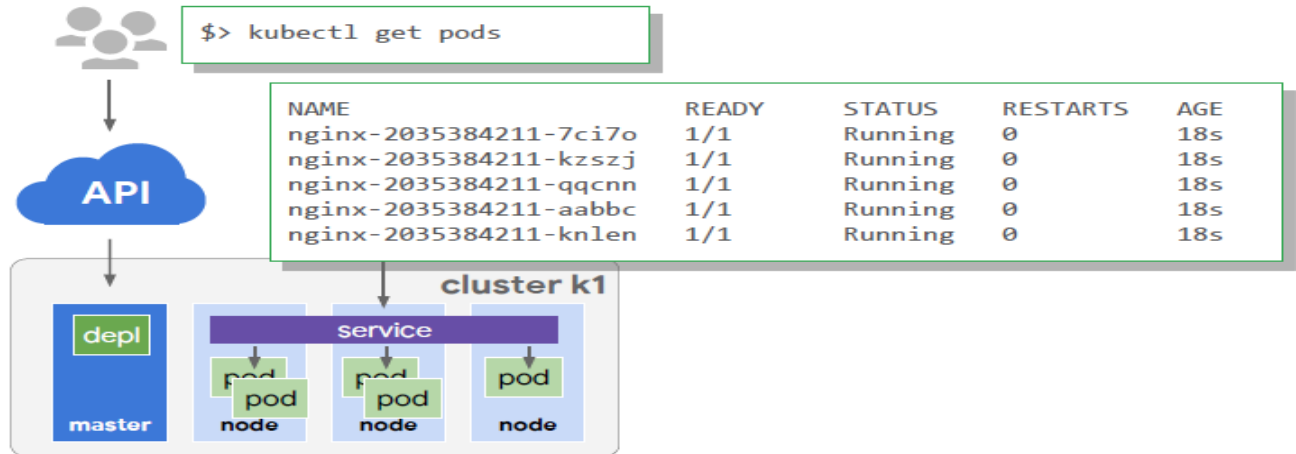
```
$> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
nginx-2035384211	5	5	5	18s

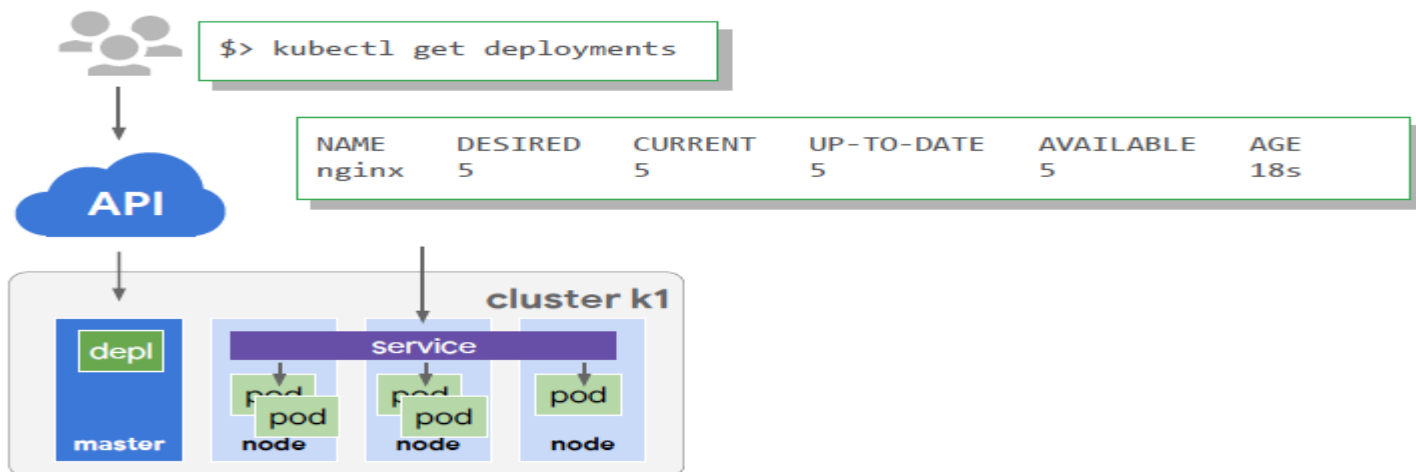


```
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.15.7
          ports:
            - containerPort: 80
```

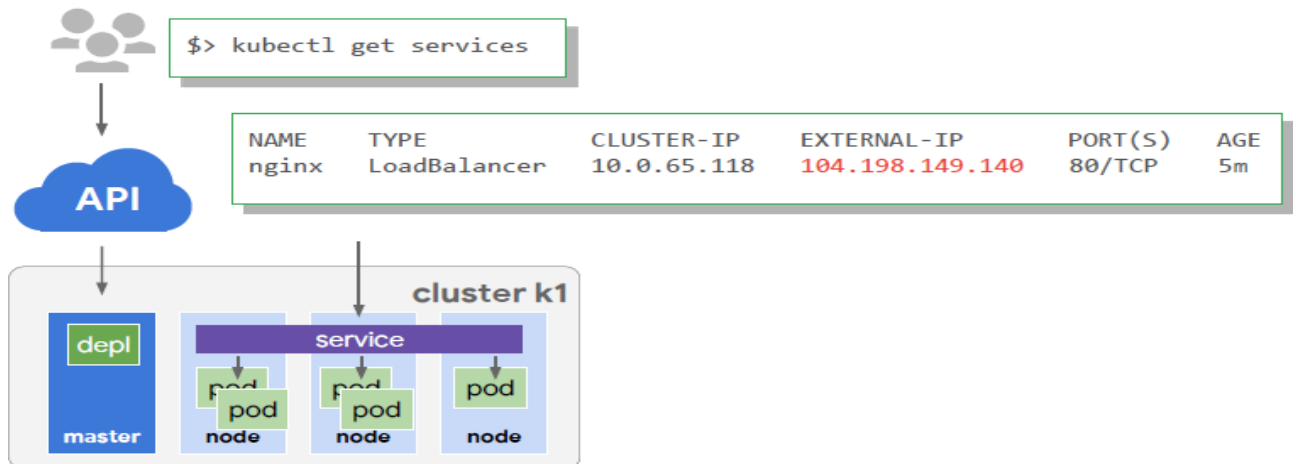

- Use **\$ kubectl get pods** command to watch the pods come online. In this case, all five are READY and RUNNING.



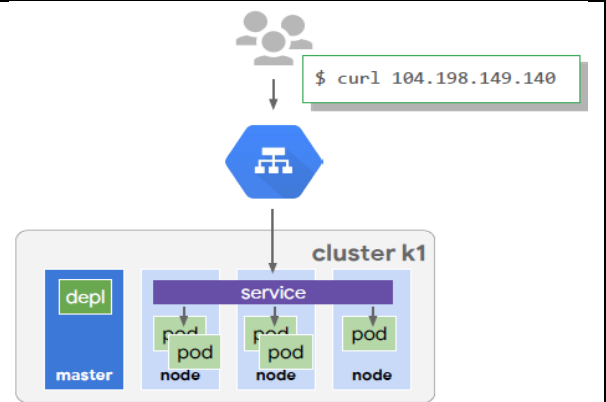
- check the Deployment to make sure the proper number of replicas are running using either **\$ kubectl get deployments** or **\$ kubectl describe deployments**. In this case, all five Pod replicas are AVAILABLE.



- You can still hit your endpoint like before using **\$ kubectl get services** to get the external IP of the Service and hit the public IP from a client.



- you have five copies of your nginx Pod running in GKE, and you have a single Service that's proxying the traffic to all five Pods. This allows you to share the load and scale your Service in Kubernetes



- The last question is what happens when you want to update a new version of your app?
- You want to update your container to get new code out in front of users, but it would be risky to roll out all those changes at once.
- You use **`$ kubectl rollout`** to change your deployment configuration file and apply the change using **`$ kubectl apply`**.
- New Pods will be created according to your update strategy.
- Here is an example configuration that will create new version Pods one by one and wait for a new Pod to be available before destroying one of the old Pods.

```
spec:
  # ...
  replicas: 5
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  # ...
```

Google Cloud Big Data & Machine Learning

- Google Cloud's big data services are fully managed and scalable



Cloud
Dataproc

Managed
Hadoop
MapReduce,
Spark, Pig, and
Hive service



Cloud
Dataflow

Stream and
batch
processing;
unified and
simplified
pipelines



BigQuery

Analytics
database;
stream data at
100,000
rows per second



Cloud
Pub/Sub

Scalable and
flexible
enterprise
messaging



Cloud
Datalab

Interactive data
exploration

-
- It is an integrated, serverless platform. "Serverless" means you don't have to provision compute instances to run your jobs.
- The services are fully managed, and you pay only for the resources you consume.
- The platform is "integrated" so GCP data services work together to help you create custom solutions.

Cloud Pub/Sub :

- Cloud Pub/Sub is a fully managed real-time messaging service that allows you to send and receive messages between independent applications.
- Cloud Pub/Sub is designed to provide "at least once" delivery at low latency with on-demand scalability to 1 million messages per second.
- Supports many-to-many asynchronous messaging.
- Application components make push/pull subscriptions to topics
- **Cloud Pub/Sub features:**
 - Highly Scalable : Any customer can send up to 10,000 messages per second, by default—and millions per second and beyond, upon request.
 - Push and Pull Delivery
 - Encryption
 - Replicated Storage :Designed to provide "at least once" message delivery by storing every message on multiple servers in multiple zones.
 - Message Queue :Build a highly scalable queue of messages using a single topic and subscription to support a one-to-one communication pattern.
 - End-to-End Acknowledgement
 - Fan-out : Publish messages to a topic once, and multiple subscribers receive copies to support one-to-many or many-to-many communication patterns.
 - REST API : Simple, stateless interface using JSON messages with API libraries in many programming languages.

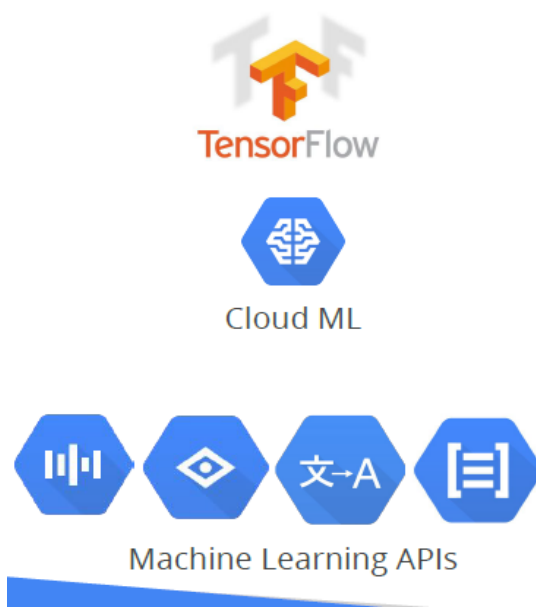
Why use Cloud Pub/Sub?

- Building block for data ingestion in Dataflow, Internet of Things (IoT), Marketing Analytics
- Foundation for Dataflow streaming
- Push notifications for cloud-based applications
- Connect applications across Google Cloud Platform (push/pull between Compute Engine and App Engine)

Cloud DataLab :

- Cloud Datalab lets you use **Jupyter notebooks** to explore, analyze, and visualize data on the Google Cloud Platform.
- Cloud Datalab offers interactive data exploration
- Interactive tool for large-scale data exploration, transformation, analysis, and visualization
- Integrated, open source : Built on Jupyter (formerly IPython)
- **Features:**
 - **Integrated** : Cloud Datalab handles authentication and cloud computation out of the box and is integrated with BigQuery, Compute Engine, and Cloud Storage.
 - **Multi-Language Support** : Python, SQL, and JavaScript (for BigQuery user-defined functions).
 - **Notebook Format** : Cloud Datalab combines code, documentation, results, and visualizations together in an intuitive notebook format.
 - Pay-per-use Pricing
 - Interactive Data Visualization
 - Use Google Charts or matplotlib for easy visualizations
 - **Collaborative** : Git-based source control of notebooks with the option to sync with non-Google source code repositories like GitHub and Bitbucket.
 - **Open Source** : Developers who want to extend Cloud Datalab can fork and/or submit pull requests on the GitHub hosted project.
 - Custom Deployment
 - Specify your minimum VM requirements, the network host, and more.
 - IPython Support : Cloud Datalab is based on Jupyter (formerly IPython) so you can use a large number of existing packages for statistics, machine learning

Cloud Machine Learning Platform



Open source tool to build and run neural network models

- Wide platform support: CPU or GPU; mobile, server, or cloud

Fully managed machine learning service

- Familiar notebook-based developer experience
- Optimized for Google infrastructure; integrates with BigQuery and Cloud Storage

Pre-trained machine learning models built by Google

- Speech: Stream results in real time, detects 80 languages
- Vision: Identify objects, landmarks, text, and content
- Translate: Language translation including detection
- Natural language: Structure, meaning of text

Quiz

28

When would you use Cloud Dataproc?

You can use it to migrate on-premises Hadoop jobs to the cloud. You can also use it for data mining and analysis of cloud-based data.

Name two use cases for Cloud Dataflow ?

ETL & Orchestration

Name three use cases for the Google machine learning platform ?

Fraud detection,
Sentiment analysis,
Content personalization

