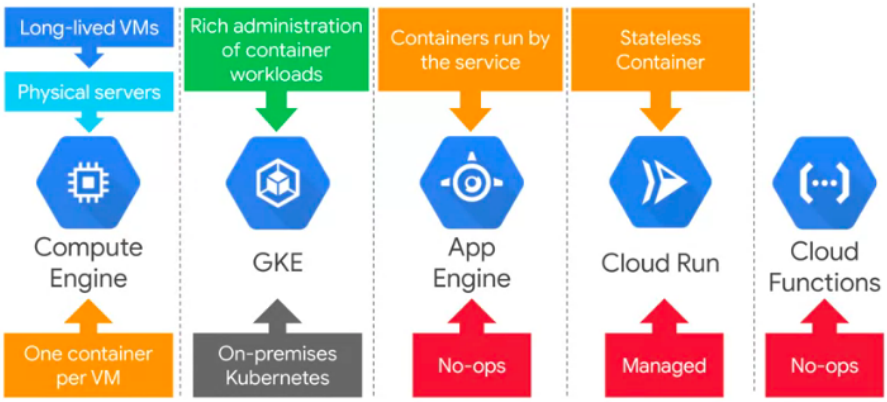
Getting Started with GKE 2023 Ivan Vlad S.

- 1 **Containers & Kubernetes**
  - Containers
    - Containers = isolated user spaces for running app code
      - delivery vehicles for app code that is lightweight, stand-alone, resource efficient, portable execution packages
    - > lightweight, does not carry full OS, can be scheduled or packaged tightly onto the underlying system
      - allows you to execute final code on VMs without worrying about software dependencies like app run times, system tools, libraries, etc.
    - > created and shutdown quickly since you're just starting and spotting the app processes, no need to boot up VMs
    - make it easier to build apps utilizing microservices design pattern.
      - microservices = loosely coupled, fine-grained components
      - this modular design pattern allows OS to scale and upgrade components without affecting the app as a whole
  - Container Images
    - Containers use a varied set of Linux technologies
      - processes, linux namespace, cgroups, union file systems
    - containers are structured in layers
      - ephemeral read/write layer
        - thin R/W layers, container layer
        - an app running in a container can only modify this layer
      - read-only layers
        - dockerfile: FROM, COPY, RUN, CMD
        - Base image layers
    - containers promote smaller shared images
    - can build containers through docker, Cloud Build, or any container registry (e.g., gcr.io)

- retrieve source code for builds from a variety of storage locations
  - Process:
    ✪ Developers >>> GitHub, git repo, Cloud Source Repos, Artifact Registry >>> Cloud build >>> Cloud Functions, GKE, App Engine

- K8s
  - container-centric mgmt environment
    - > manage containers at scale
      > manage container infra
      - automates deployment, scaling, load balancing, logging, monitoring, and other mgmt features of containerized apps
  - open source, container mgmt, automation, declarative config, imperative config
    - declarative config = (infra) you describe the desired state you want to achieve instead of issuing a series of commands to achieve that state.
      - K8s job = make the deployed system conform to your desired state and then keep it there in spite of failures
      - saves you work
      - ✪ big strength of K8s = auto keep a system in a state that you declare
    - imperative config = issue commands to change the system state
      - experience admins use imperative config only for a quick temporary fix and as a tool in building a declarative config
  - ✪ K8s Features
    - Supports both stateful and stateless apps
    - autoscaling
      - can scale in and out based on resource utilization
    - resource limits
      - specify resource request levels and resource limits for your workloads and K8s will obey them
    - extensibility
      - can extend K8s through rich ecosystem of plugins and add-ons
    - portability
      - open source, can be deployed anywhere, move K8s workloads without vendor lock-in

- GKE
  - GCPs managed service offering for K8s
    - help you deploy, manage, and scale K8s environments for your containerized apps on GCP
    - GKE is a component of GCP compute offerings, makes it easy to bring your K8s workloads to the cloud
  - uses container optimized OS, managed by google
    - optimized to scale quickly and with a minimal resource footprint
  - start by directing the service to instantiate a K8s system for you
    - this is called a cluster
      - the VMs that host containers inside GKE clusters are called nodes
  - features
    - fully managed
    - auto repair
      - GKE repairs unhealthy nodes, periodic health checks on each node
    - container optimized OS
    - auto upgrade
    - cluster scaling
    - seamless integration
      - with GCP Cloud Build and Container Registry
    - IAM
      - allows control for access through use of accounts and role perms
    - integrated logging and monitoring
      - GCP Cloud Operations suite
    - integrated networking
      - with GCP VPCs
    - cloud console
      - more powerful and more secure than normal K8s dash, and saves setup time
- Compute Options Detail, how to decide?

- 

2 Kubernetes Architecture

- K8's concepts
  - Objects: each thing K8s manages is represented by an object, you can view and change these objects, attributes, and state
    >> persistent entities representing the sate of the cluster
    - Object Spec
      - desired state described by us
    - Object Status
      - current state described by K8s
  - principle of declarative mgmt
    - K8s expects you to tell it what you want the state of the objects under its mgmt to be, it will work to bring that state into being and keep it there
      - the K8s control plane will continuously monitor the state of the cluster, endlessly comparing reality to what has been declared and remedying state as needed
  - Containers in a Pod share resources
    - a Pod embodies the environment where containers live and that environment can accomodate one or more containers
      - containers within a pod are tightly coupled with one another and can communicate using localhost IP address
- K8s control plane
  - kube APIserver
    - interact using kubectl
      - only component where clients interact directly

- each node runs a kubelet > reports back to kube APIserver
  - kubelet = K8s agent on each node
- etcd = cluster's db
  - store the state of the cluster
- kube scheduler
  - responsible for scheduling pods into nodes
- kube controller manager
  - continuously monitors state of the cluster through kubeAPIserver
    - use to manage workloads

- K8s Engine Concepts
  - GKE manages entire K8s control plane
    - GKE provisions control planes as part of abstract parts of the GKE service that are not exposed to GCP customers
  - K8s doesn't create nodes, cluster admins create nodes and add them to K8s
    - GKE manages this by deploying and registering Compute Engine instances as nodes
      - use node pools to manage different kinds of nodes
  - zonal vs regional clusters
    - regional cluster ensure that availability of app is maintained across multiple zones in single region - 3 zones is default
      - a regional or zonal GKE cluster can also be set up as a private cluster

- K8s object mgmt
  - objects are defined in a YAML file, USE version control on YAML files
    - YAML file
      - apiVersion
      - kind
      - metadata
      - spec
    - all objects are assigned unique ID
      - cannot have 2 or the same object types with same names
    - labels = key value pairs, tag objects with

- labels can be matched by label selectors
- pods have a life cycle, they are ephemeral
  - controller object: manage state of the pods
    - ReplicaSets
    - Deployments
    - Replication Controllers
    - StatefulSets
    - DaemonSets
    - Jobs
  - deployment object: ensure that a defined set of pods is running at any given time
- namespaces
  - provide scope for naming resources such as pods, deployments, and contollers
    - default
    - kube-system
    - kube-public
  - K8s allows to abstract a single physical cluster into multiple clusters known as namespaces
    - implement resource quotas across cluster
    - allow to use object names that would otherwise be duplicates of one another
- services
  - services provide load-balanced access to specific pods
    - ClusterIP
      - the default, exposes service on an IP address that is only accessible from within this cluster
    - NodePort
      - exposes service on the IP address of each node in the cluster, at a specific port number
    - LoadBalancer
      - exposes the service externally

- Migrate for Anthos
  - Migrate for Anthos moves VMs to containers
    - move and convert workloads into containers
    - workloads can start as physical servers or VMs
    - Moves workload compute to container immediately (<10 min)
    - data can be migrated all at once or "streamed" to the cloud until the app is live in the cloud
  - a migration requires an architecture to be built
    - allow Migrate for Compute Engine to create a pipeline for streaming or migrating the data from on prem or another CSP into GCP
      - Migrate for Anthos is then installed on a GKE processing cluster and is composed of many K8s resources >> container goes into Cloud Storage >> images stores in Container Registry
  - Migration path
    - configure processing cluster >> add migration source >> generate and review plan >> generate artifacts
  - migrate for Anthos installation
    - requires processing cluster >> install uses migctl

3️⃣ Kubernetes Workloads

- kubectl command
  - transforms CLI entries into API calls
    - use kubectl to see a list of pods in a cluster
      - many command uses: create K8s objects, view objects, delete objects, view and export configs
  - kubectl must be configured first
    - relies on config file $HOME/.kube/config
  - syntax has several parts
    - [command]
      - what do you want to do
    - [TYPE]
      - on what type of object

- [NAME]
  - what is that objects name
- [flags]
  - any special requests?

- deployments
  - declare the state of pods
    - roll out updates to the pods, roll back pods to previous revision, scale or autoscale pods, well-suited for stateless apps
  - deployment = 2 step process
    - .yaml file >>> deployment object >>> deployment controller >>> node
  - 3 different lifecycles states
    - progressing , complete, failed
  - use kubectl to inspect your deployment
    - can output deployment config in a YAML format
  - 3 ways to create deployment
    - 1, create deployment declaratively using a manifest file such as the YAML file you've just seen and kubectl apply command
    - 2. (imperatively) use kubectl run command that specifies the parameters inline
    - 3. use GKE workloads menu in GCP console
- Services & Scaling
  - you can scale the deployment manually or autoscale
- Updating deployments
  - using kubectl apply, set image deployment, edit commands
    - can edit deployment manifest in GCP console
- blue-green deployments
  - Service is a load-balancing front end for Pods
  - blue/green deployment strategy = useful strategy when you want to deploy a new version of an app and also ensure that app services remain available while the Deployment is updated.
    - creates new Deployment with newer version of the app (V1, V2)

- Canary deployments
  - update strategy based on blue/green method
  - a/b testing is used to make business decisions based on the results derived from data
  - shadow testing allows you to run a new, hidden version
  - can roll back using kubectl
    - console shows you revision list with summaries and creation dates
- Managing deployments
  - can pause rollouts with kubectl pause command , same with resume and status
- Pod networking
  - Pod = group of containers with shared storage and networking
    - your workload doesn't run in a single pod
      - comms through pod-to-pod communication on the same node
  - nodes get Pod IP addresses from address ranges assigned to your VPC
- Volumes
  - are a directory which is accessible to all of the containers in a pod
    - some volumes are ephemeral
    - some volumes are persistent
      - > manage durable storage in a cluster
        > independent of the pods lifecycle
        >provisioned dynamically through PersistentVolumeClaims or explicitly created by a cluster admin
  - volume types
    - ephemeral types explained
      - emptyDir
        > ephemeral: shares pod's lifecycle
      - ConfigMap
        > object can be referenced in a volume
      - Secret
        > stores sensitive info, such as passwords
      - donwardAPI
        > makes data about pods data available to containers

- PersistentVolumes
  - PersistentVolume (PV)
    > independent of a pods lifecycle
    > managed by K8s
    > manually or dynamically provisioned
    > persistent disks are used by GKE as PVs
    *PVs must be claimed
  - PersistentVolumeClaim (PVC)