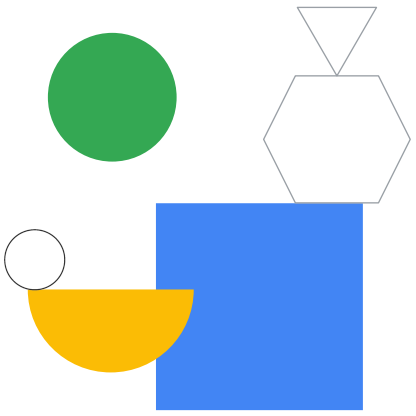
Google Cloud

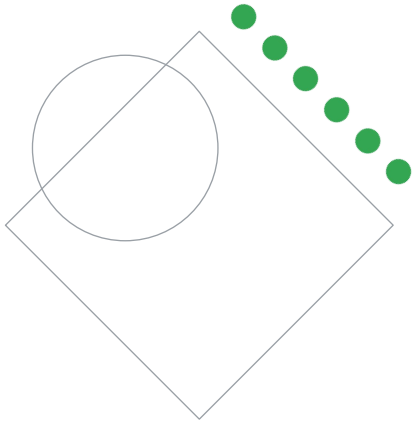
Preparing for Your Associate Cloud Engineer Journey



Module 4: Ensuring Successful Operation of a Cloud Solution

Welcome to Module 4: Ensuring Successful Operation of a Cloud Solution.

Review and study planning

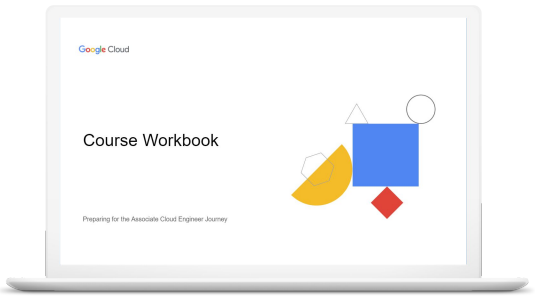


Google Cloud

What areas do you need to develop your skills in order to manage the different aspects of a Google Cloud solution? This is another important area for an Associate Cloud Engineer, and where you'll likely spend much of your time on the job. Let's review the diagnostic questions to help you target your study time to focus on the areas where you need to develop your skills.

Your study plan:

Ensuring successful operation of a cloud solution



4.1	Managing Compute Engine resources
4.2	Managing Google Kubernetes Engine resources
4.3	Managing Cloud Run resources
4.4	Managing storage and database solutions
4.5	Managing networking resources
4.6	Monitoring and logging

Google Cloud

We'll approach this review by looking at the objectives of this exam section and the questions you just answered about each one. We'll introduce an objective, briefly review the answers to the related questions, then talk about where you can find out more in the learning resources and/or in Google Cloud documentation. As we go through each section objective, use the page in your workbook to mark the specific documentation, courses (and modules!), and quests you'll want to emphasize in your study plan.

Just like with the previous section, there are multiple objectives in this section that have many related tasks - so you will probably need to plan for more study time.

4.1 Managing Compute Engine resources

Tasks include:

- Managing a single VM instance (e.g., start, stop, edit configuration, or delete an instance)
- Remotely connecting to the instance
- Attaching a GPU to a new instance and installing necessary dependencies
- Viewing current running VM inventory (instance IDs, details)
- Working with snapshots (e.g., create a snapshot from a VM, view snapshots, delete a snapshot)
- Working with images (e.g., create an image from a VM or a snapshot, view images, delete an image)
- Working with instance groups (e.g., set autoscaling parameters, assign instance template, create an instance template, remove instance group)
- Working with management interfaces (e.g., Cloud Console, Cloud Shell, Cloud SDK)

Google Cloud

As we've discussed before, Cymbal Superstore's supply chain app is built on Compute Engine resources. As an Associate Cloud Engineer you might be put in charge of implementing and updating instance groups, which give you healing and autoscaling capabilities. It also lets you load balance data if needed, as we'll discuss in the networking section. Common actions can be scripted with command line tools or client SDKs.

These are the diagnostic questions you answered that relate to this area:
Question 1: Identify commands required to list and describe Compute Engine disk snapshots
Question 2: Describe the incremental nature of Compute Engine disk snapshots
Question 3: Implement an Instance Group based on an instance template

4.1 Diagnostic Question 01 Discussion

You want to view a description of your available snapshots using the command line interface (CLI). What gcloud command should you use?

A. gcloud compute snapshots list
B. gcloud snapshots list
C. gcloud compute snapshots get
D. gcloud compute list snapshots

Google Cloud

Question:
You want to view a description of your available snapshots using the command line interface (CLI). What gcloud command should you use?

*A. **gcloud compute snapshots list**
Feedback: Correct! gcloud commands are built with groups and subgroups, followed by a command, which is a verb. In this example, Compute is the Group, snapshots is the subgroup, and list is the command.

B. **gcloud snapshots list**
Feedback: Incorrect. Snapshots is not a main group defined in gcloud.

C. **gcloud compute snapshots get**
Feedback: Incorrect. Available commands for snapshots are list, describe, and delete.

D. **gcloud compute list snapshots**
Feedback: Incorrect. Snapshots is a compute command subgroup. It needs to come before the list command.

Where to look:
https://cloud.google.com/compute/docs/disks/create-snapshots#listing-snapshots
https://cloud.google.com/compute/docs/disks/create-snapshots#viewing-snapshot

Content mapping:

- Instructor-led Training/OnDemand
  - Google Cloud Fundamentals: Core Infrastructure
    - M3 Compute Engine and Networking
  - Architecting with Google Compute Engine
    - M3 Virtual Machines

Summary:
Explanation/summary on the following slide.

Getting information about snapshots

To list Compute Engine disk snapshots:

```
gcloud compute snapshots list --project PROJECT_ID
```

To describe snapshots:

```
gcloud compute snapshots describe SNAPSHOT_NAME
```

Google Cloud

To list snapshots run the following command:

```
gcloud compute snapshots list --project PROJECT_ID
Flags available
      --limit maximum number of results you want back
      --regexp a filter for results you want returned
      --sort-by field to sort by
      --uri only print URI's of the resources returned
```

To describe snapshots, which returns creation time, size, and source disk, run the following command:

```
gcloud compute snapshots describe SNAPSHOT_NAME
Flags available
      --format what kind of format you want printed out
      --project project to be used for command
      --quiet disables interactive prompts
```

4.1 Diagnostic Question 02 Discussion

You have a scheduled snapshot you are trying to delete, but the operation returns an error.

What should you do to resolve this problem?

A. Delete the downstream incremental snapshots before deleting the main reference.
B. Delete the object the snapshot was created from.
C. Detach the snapshot schedule before deleting it.
D. Restore the snapshot to a persistent disk before deleting it.

Google Cloud

Question:
You have a scheduled snapshot you are trying to delete, but the operation returns an error. What should you do to resolve this problem?

A. Delete the downstream incremental snapshots before deleting the main reference.
Feedback: Incorrect. This is not required to delete a scheduled snapshot and would be a lot of manual work.

B. Delete the object the snapshot was created from.
Feedback: Incorrect. This is not required to delete a scheduled snapshot and is destructive.

*C. Detach the snapshot schedule before deleting it.
Feedback: Correct! You can't delete a snapshot schedule that is still attached to a persistent disk.

D. Restore the snapshot to a persistent disk before deleting it.
Feedback: Incorrect. This does not allow you to delete a scheduled snapshot.

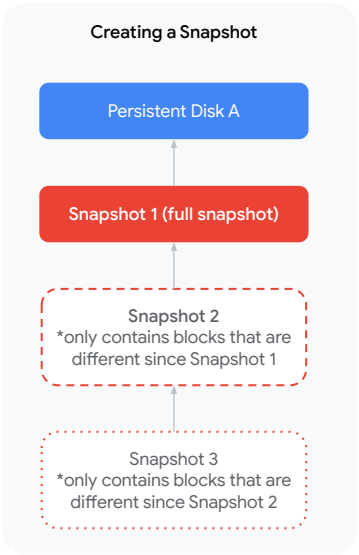
Where to look:
https://cloud.google.com/compute/docs/disks/snapshots#incremental-snapshots

Content mapping:

- Instructor-led Training/OnDemand
  - Google Cloud Fundamentals: Core Infrastructure
    - M3 Compute Engine and Networking
  - Architecting with Google Compute Engine
    - M3 Virtual Machines

Summary:
Explanation/summary on the following slide.

Snapshots are incremental



Google Cloud

Reference from snapshot concepts page:
https://cloud.google.com/compute/docs/disks/snapshots

Snapshots are incremental in nature and less expensive than creating full images of a disk. You can only create them for persistent disks. They are stored in a Cloud Storage bucket managed by the snapshot service and are automatically compressed. You can choose regional or multi-regional storage, which will affect cost. Snapshots are stored across multiple locations with automatic checksums. You schedule them using the gcloud command line and cron. You can also set up a snapshot schedule in the Google cloud console or command line. A Snapshot schedule and its source persistent disk have to be in the same region. You can restore them to a new persistent disk. The new disk can be in a different zone or region, so you can use snapshots to move VMs. You can create them while they are running. Snapshots of a disk have to be at least 10 minutes apart.

Here's how the incremental nature of snapshots works:

- The first snapshot is full and contains all data on the persistent disk it was run on.
- Each subsequent snapshot only contains new or modified data since the first snapshot.

- However, sometimes to clean up resources and cost, a new snapshot might be a full backup.

Deleting a snapshot copies its data to downstream incremental snapshots that are dependent on it, increasing the downstream snapshot's size. You can't delete a snapshot that has a schedule associated with it.

4.1 Diagnostic Question 03 Discussion

Which of the following tasks are part of the process when configuring a managed instance group? (Pick two.)

A. Defining Health checks.
B. Providing Number of instances.
C. Specifying Persistent disks.
D. Choosing instance Machine type.
E. Configuring the operating system.

Google Cloud

Question:
Which of the following tasks are part of the process when configuring a managed instance group? (Pick two.)

*A. Defining Health checks
Feedback: Correct! Health checks are part of your managed instance group configuration.

*B. Providing Number of instances
Feedback: Correct! Number of instances is part of your managed instance group configuration.

C. Specifying Persistent disks
Feedback: Incorrect. This is part of your instance template definition.

D. Choosing instance Machine type
Feedback: Incorrect. This is part of your instance template definition.

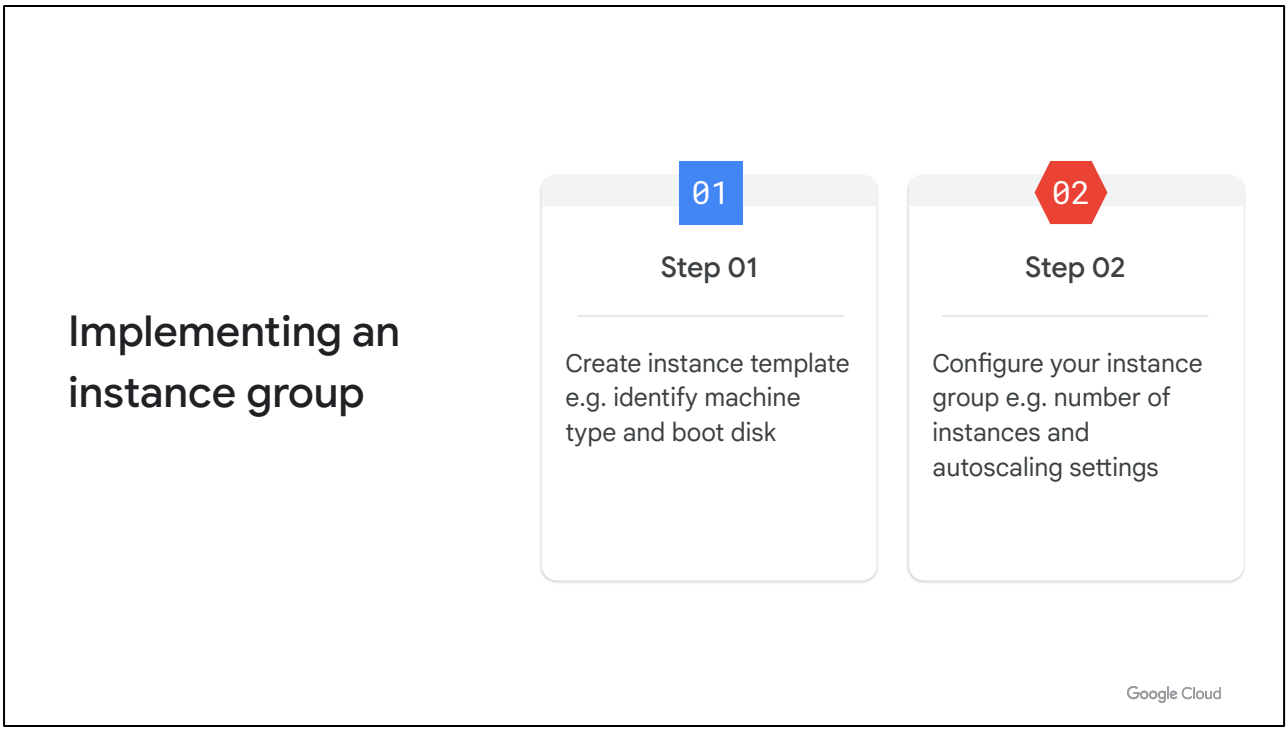
E. Configuring the operating system
Feedback: Incorrect. This is part of your instance template definition.

Where to look: https://cloud.google.com/compute/docs/instance-templates
https://cloud.google.com/compute/docs/instance-groups

Content mapping:

- Instructor-led Training/OnDemand
  - Architecting with Google Compute Engine
    - M9 Load Balancing and Autoscaling

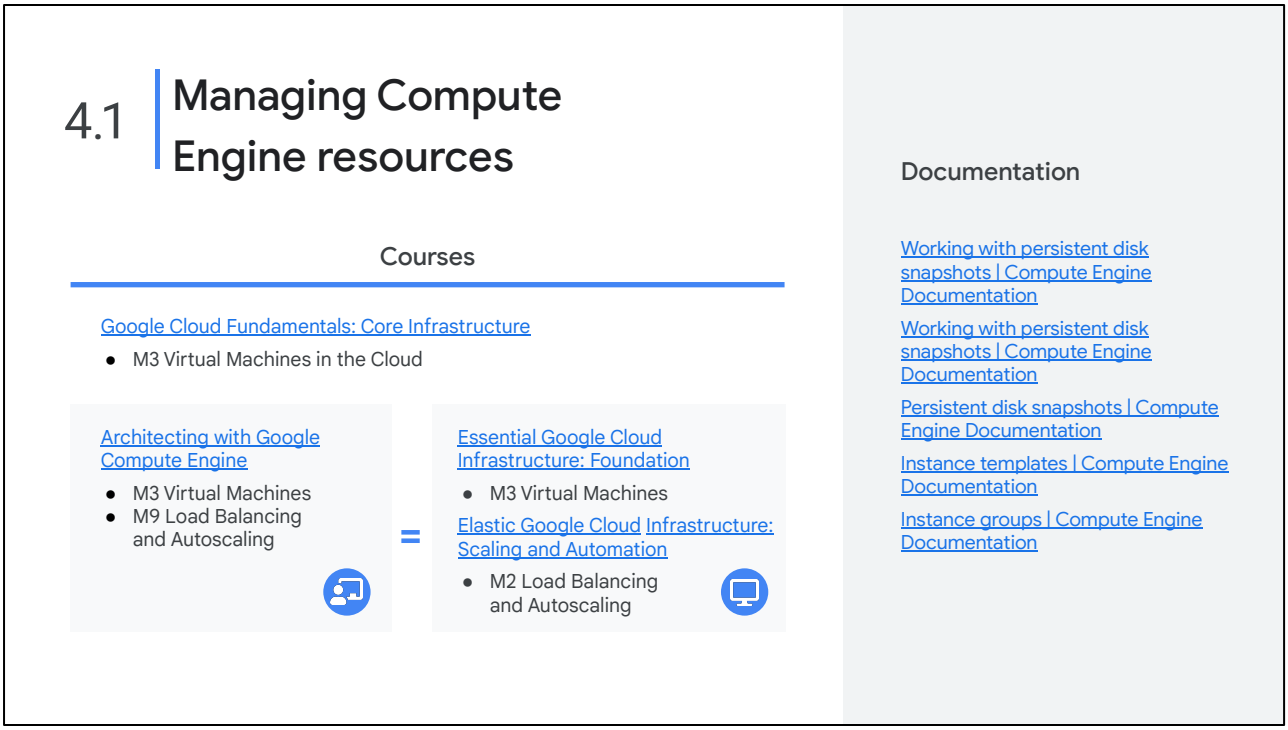
Summary:
Explanation/summary on the following slide.



Managed instance groups help you create and manage groups of identical VM instances. They are based on an instance template that defines how new VMs added to the instance group should be configured. You can specify the size of an instance group and change it at any time. The managed instance group will make sure the number of instances matches what you request, and monitors instances via health checks. If an instance goes down, the managed instance group will start another instance to replace it. Managed instance groups can be zonal or regional. Regional instance groups create instances across multiple zones in a region so your application can still run in case of a zonal outage.

The first step to creating a managed instance group is to create an instance template. An instance template contains information about how to create instances in the group by specifying machine type, boot disk, connectivity, disks, and other details pertinent to your needs. This information is similar to what you would provide if you were configuring an individual instance.

After you create an instance template you need to configure your managed instance group. Here is where you specify location settings, describe port mappings, and reference the instance template. You also specify the number of instances in your group, configure autoscaling, and create health checks for your instances to determine which instances should receive traffic.



Let's take a moment to consider resources that can help you build your knowledge and skills in this area.

The concepts in the diagnostic questions we just reviewed are covered in these modules and in this documentation. You'll find this list in your workbook so you can take a note of what you want to include later when you build your study plan. Based on your experience with the diagnostic questions, you may want to include some or all of these.

https://cloud.google.com/compute/docs/disks/create-snapshots#listing-snapshots
https://cloud.google.com/compute/docs/disks/create-snapshots#viewing-snapshot
https://cloud.google.com/compute/docs/disks/snapshots#incremental-snapshots
https://cloud.google.com/compute/docs/instance-templates
https://cloud.google.com/compute/docs/instance-groups

4.2 Managing Google Kubernetes Engine resources

Tasks include:

- Viewing current running cluster inventory (nodes, pods, services)
- Browsing Docker images and viewing their details in the Artifact Registry
- Working with node pools (e.g., add, edit, or remove a node pool)
- Working with pods (e.g., add, edit, or remove pods)
- Working with services (e.g., add, edit, or remove a service)
- Working with stateful applications (e.g. persistent volumes, stateful sets)
- Managing Horizontal and Vertical autoscaling configurations.
- Working with management interfaces (e.g., Cloud Console, Cloud Shell, Cloud SDK, kubectl)

Google Cloud

GKE is a managed service that provides production-level orchestration for your container-based applications. Cymbal Superstore's e-commerce app is implemented through services being exposed by GKE. As an Associate Cloud Engineer on the ecommerce app team, certain tasks might require you to interact with a GKE cluster and its nodes. You will also have to know about GKE's workload objects, such as pods, deployments, and services. Containers in GKE are based on images which are shared via the Google Container registry. You need to be familiar with how to create images and deploy them to the registry.

These diagnostic questions addressed managing GKE resources:
Question 4: Contrast the differences between an internal and external load balancer in Google Kubernetes Engine
Question 5: Describe the relationship between Kubernetes pods, services, and deployments
Question 6: Apply kubectl commands to manage pods, deployments, and services

4.2 Diagnostic Question 04 Discussion

Cymbal Superstore's GKE cluster requires an internal http(s) load balancer. You are creating the configuration files required for this resource.

What is the proper setting for this scenario?

A. Annotate your ingress object with an ingress.class of "gce."
B. Configure your service object with a type: LoadBalancer.
C. Annotate your service object with a neg reference.
D. Implement custom static routes in your VPC.

Google Cloud

Question:
Cymbal Superstore's GKE cluster requires an internal http(s) load balancer. You are creating the configuration files required for this resource. What is the proper setting for this scenario?

A. Annotate your ingress object with an ingress.class of "gce."
Feedback: Incorrect. To implement an internal load balancer, the ingress class needs to be "gce-internal."

B. Configure your service object with a type: LoadBalancer.
Feedback: incorrect. Using Load Balancer at the service level implements a Layer 4 network load balancer, not an http(s) load balancer.

*C. Annotate your service object with a neg reference.
Feedback: Correct! This is correct because an internal http(s) load balancer can only use NEGs.

D. Implement custom static routes in your VPC.
Feedback: Incorrect. This describes a routes-based cluster. In order to support internal load balancing, your cluster needs to use VPC-native mode, where your cluster provides IP addresses to your pods from an alias IP range.

Where to look:https://cloud.google.com/kubernetes-engine/docs/concepts/ingress-ilb

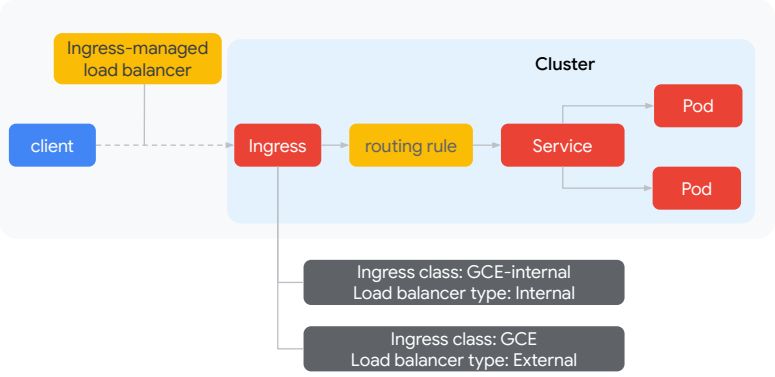
https://cloud.google.com/kubernetes-engine/docs/concepts/ingress-xlb
https://cloud.google.com/kubernetes-engine/docs/how-to/load-balance-ingress
https://cloud.google.com/kubernetes-engine/docs/how-to/internal-load-balance-ingress

Content mapping:

- Instructor-led Training/OnDemand
  - Google Cloud Fundamentals: Core Infrastructure
    - M5 GKE
  - Getting Started with GKE
    - M4 Introduction to Kubernetes Workloads

Summary:
Explanation/summary on the following slide.

Internal vs External load balancing in Kubernetes



Google Cloud

To implement network load balancing you create a service object with these settings:

- `type: LoadBalancer.`
- Set External Traffic Policy to cluster or local

Cluster - traffic will be load balanced to any healthy GKE node and then kube-proxy will send it to a node with the pod.

Local - nodes without the pod will be reported as unhealthy. Traffic will only be sent to nodes with the pod. Traffic will be sent directly to pod with source ip header info included.

To implement external http(s) load balancing create an ingress object with the following settings:

- Routing depends on URL path, session affinity, and the balancing mode of backend Network endpoint groups (NEGS)
- The object type is ingress.
- Using `ingress.class: “gce”` annotation in the metadata deploys an external load balancer.
- External load balancer is deployed at Google Points of presence.
- Static IP for ingress lasts as long as the object.

To implement an internal http(s) load balancer create an ingress object with the following settings:

- Routing depends on URL path, session affinity, and balancing mode of the backend NEGS.
- The object kind is ingress.
- Metadata requires an `Ingress.class: “gce-internal”` to spawn an internal load balancer.
- Proxies are deployed in a proxy only subnet in a specific region in your VPC.
- Only NEGs are supported. Use the following annotation in your service metadata:
  - `cloud.google.com/neg: '{"ingress": true}'`
- Forwarding rule is assigned from the GKE node address range.

4.2 Diagnostic Question 05 Discussion

What Kubernetes object provides access to logic running in your cluster via endpoints that you define?

A. Pod templates
B. Pods
C. Services
D. Deployments

Google Cloud

Question:
What Kubernetes object provides access to logic running in your cluster via endpoints that you define?

A. Pod templates
Feedback: Incorrect. Pod templates define how pods will be configured as part of a deployment.

B. Pods
Feedback: Incorrect. Pods provide the executable resources your containers run in.

*C. Services
Feedback: Correct! Service endpoints are defined by pods with labels that match those specified in the service configuration file. Services then specify how those pods are exposed.

D. Deployments
Feedback: Incorrect. Deployments help you with availability and the health of a set of pod replicas. They do not help you configure external access.

Where to look:

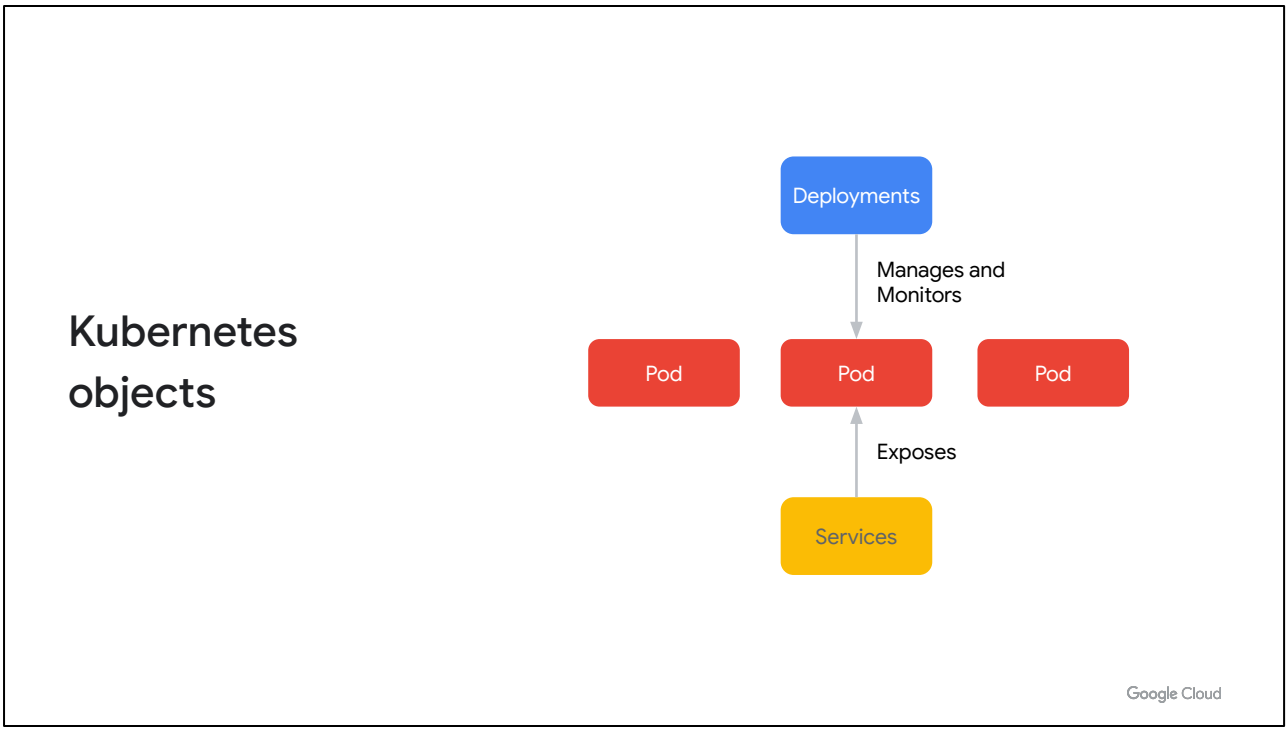
https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview
https://cloud.google.com/kubernetes-engine/docs/concepts/pod
https://cloud.google.com/kubernetes-engine/docs/concepts/deployment
https://cloud.google.com/kubernetes-engine/docs/concepts/service

Content mapping:

- Instructor-led Training/OnDemand
  - Google Cloud Fundamentals: Core Infrastructure
    - M5 GKE
  - Getting Started with GKE
    - M3 Kubernetes Architecture
- Quests
  - Set Up and Configure a Cloud Environment in Google Cloud (https://www.qwiklabs.com/quests/119)

Summary:

Explanation/summary on the following slide.



What is a pod? A pod Is the smallest deployable object in Kubernetes. It is a single instance of a running process that contains one or more docker containers. Pods provide networking and storage to containers, and contain dependencies the container needs to run and communicate.

What is a deployment? A deployment manages a set of multiple identical pods. It uses a replica set to define the number of pods. A deployment monitors pods in a replica set and replaces unhealthy instances to ensure your application remains available. A deployment uses a pod template, which provides a spec of what each deployed pod should look like. When you update the pod template in a deployment, it starts a rolling upgrade of the pods in the deployment.

What is a service? A service is a group of pod endpoints that you can configure access for. You use selectors to define which pods are included in a service.
A service gives you a stable IP that belongs to the service. Pods have internal IP addresses but they can change as pods get restarted and replaced.
A service can be configured to implement load balancing.

4.2 Diagnostic Question 06 Discussion

What is the declarative way to initialize and update Kubernetes objects?

A. kubectl apply
B. kubectl create
C. kubectl replace
D. kubectl run

Google Cloud

Question:
What is the declarative way to initialize and update Kubernetes objects?

*A. **kubectl apply**
Feedback: Correct! **kubectl apply** creates and updates Kubernetes objects in a declarative way from manifest files.

B. **kubectl create**
Feedback: Incorrect. **kubectl create** creates objects in an imperative way. You can build an object from a manifest but you can't change it after the fact. You will get an error.

C. **kubectl replace**
Feedback: Incorrect. **kubectl replace** downloads the current copy of the spec and lets you change it. The command replaces the object with a new one based on the spec you provide.

D. **kubectl run**
Feedback: Incorrect. **kubectl run** creates a Kubernetes object in an imperative way using arguments you specify on the command line.

Where to look:
https://cloud.google.com/kubernetes-engine/docs/how-to/deploying-workloads-over

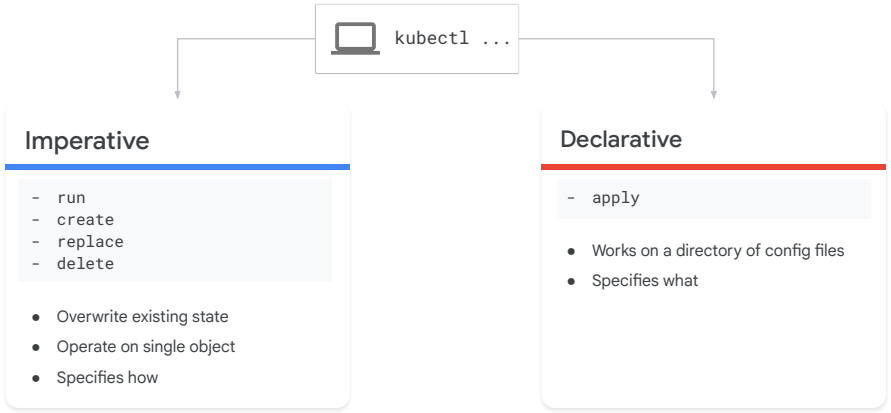
view#imperative_commands
https://kubernetes.io/docs/concepts/overview/working-with-objects/object-management/

Content mapping:

- Instructor-led Training/OnDemand
  - Getting Started with GKE
    - M4 Introduction to Kubernetes Workloads
- Quests
  - Set Up and Configure a Cloud Environment in Google Cloud (https://www.qwiklabs.com/quests/119)

Summary:
Explanation/summary on the following slide.

Types of kubectl commands



Google Cloud

You execute kubectl commands to manage objects such as pods, deployments, and services.

Imperative commands such as run, create, replace, delete act on a live object or single config file and overwrite any state changes that have occurred on an existing object.

Declarative commands use a config stored in a directory to deploy and apply changes to your app objects.

- uses kubectl -apply on a directory
- You don't specify create, replace or delete commands.

Example commands:

kubectl -run #generates a new object in a cluster, by default of deployment
Kubectl -create #generates a new object from a config file
Kubectl -get #display requested resources
kubectl -expose #creates a new service that distributes traffic to labelled pods

Pods are not created by themselves but are based on template made available

in deployments.

You can use the name of an existing object or defined config file. The object you create service for can be one of the following: deployment, service, replica set, replication controller or pod.

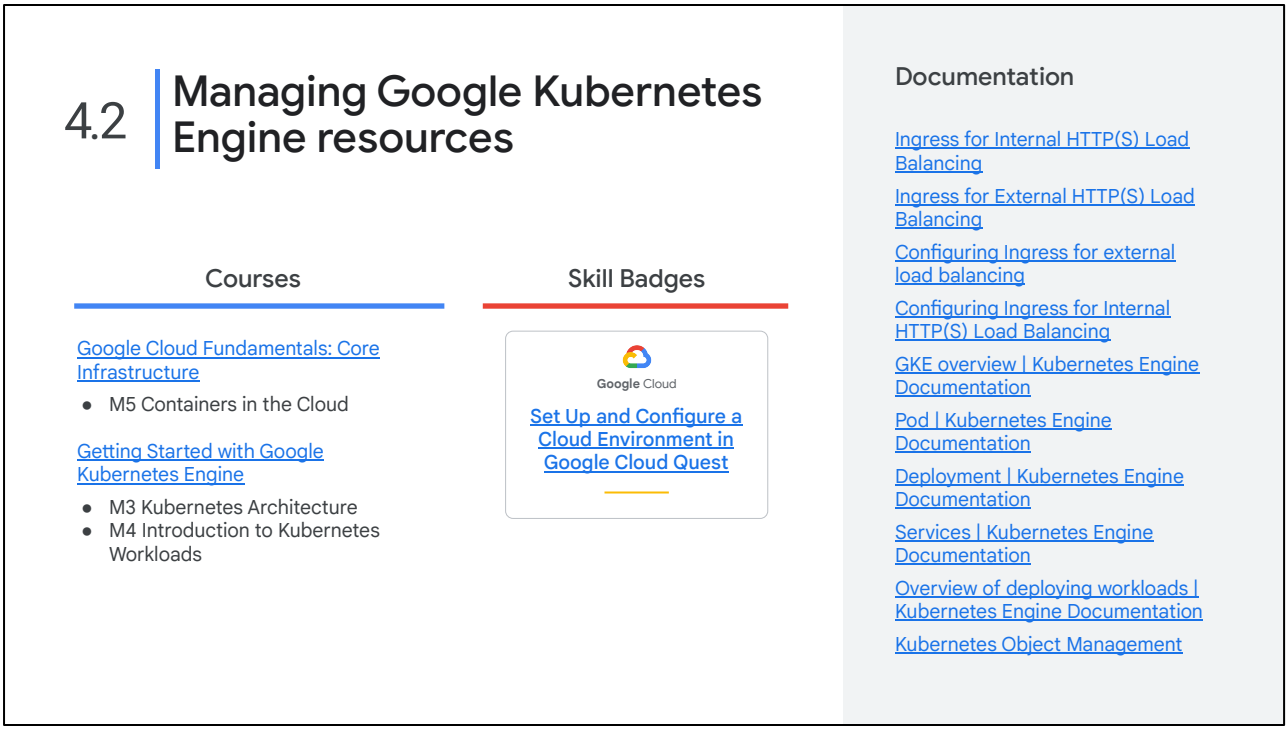
Important specs in a deployment manifest:

- kind:Deployment
- Replicas:3
- Spec:template specifies labels, container name, and image it is based on.

If you need to change a deployment you change the config file and do a kubectl -apply

Important specs in a service manifest:

Kind: service
Spec:selector - labels of pods included in service - have to match them all
Port: incoming port
targetPort: port the pod is listening on



Let's take a moment to consider resources that can help you build your knowledge and skills in this area.

The concepts in the diagnostic questions we just reviewed are covered in these modules, skill badges, and documentation. You'll find this list in your workbook so you can take a note of what you want to include later when you build your study plan. Based on your experience with the diagnostic questions, you may want to include some or all of these.

https://cloud.google.com/kubernetes-engine/docs/concepts/ingress-ilb
https://cloud.google.com/kubernetes-engine/docs/concepts/ingress-xlb
https://cloud.google.com/kubernetes-engine/docs/how-to/load-balance-ingress
https://cloud.google.com/kubernetes-engine/docs/how-to/internal-load-balance-ingress
https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview
https://cloud.google.com/kubernetes-engine/docs/concepts/pod
https://cloud.google.com/kubernetes-engine/docs/concepts/deployment
https://cloud.google.com/kubernetes-engine/docs/concepts/service
https://cloud.google.com/kubernetes-engine/docs/how-to/deploying-workloads-overview#imperative_commands

https://kubernetes.io/docs/concepts/overview/working-with-objects/object-management/

4.3 Managing Cloud Run resources

Tasks include:

- Adjusting application traffic splitting parameters
- Setting scaling parameters for autoscaling instances
- Determining whether to run Cloud Run (fully managed) or Cloud Run for Anthos

Google Cloud

Cloud Run and Cloud Functions are Google's serverless approach to handling containers and functional code. In both of these technologies, you pay for resources based on how many requests are coming in. One big difference between the two of them is that Cloud Run is optimized for multiple concurrent connections to each instance, while Cloud Functions only lets you have only one connection per function instance.

For Cymbal Superstore, Cloud Run could be used to quickly test updates to containers. As an Associate Cloud Engineer, you could be tasked to implement traffic splitting to test changes or rollback updates that didn't work well. There are also settings you need to know for autoscaling, such as min and max instances, that will let you make tradeoffs of relative latency versus cost. You also have the choice of using a fully managed version in Google Cloud or a hybrid version available as part of Anthos. The hybrid version runs on abstracted GKE resources allocated by your Anthos cluster.

These types of tasks were covered in this question:
Question 7: Express the differences between manual, basic, and automatic scaling in serverless: App Engine or Cloud Run

4.3 Diagnostic Question 07 Discussion

You have a Cloud Run service with a database backend. You want to limit the number of connections to your database.

What should you do?

A. Set Min instances.
B. Set Max instances.
C. Set CPU Utilization.
D. Set Concurrency settings.

Google Cloud

Question:
You have a Cloud Run service with a database backend. You want to limit the number of connections to your database. What should you do?

A. Set Min instances.
Feedback: Incorrect. Min instances reduce latency when you start getting requests after a period of no activity. It keeps you from scaling down to zero.

*B. Set Max instances.
Feedback: Correct! Max instances control costs, keeping you from starting too many instances by limiting your number of connections to a backing service.

C. Set CPU Utilization.
Feedback: Incorrect. Default CPU utilization is 60%. It doesn't affect the number of connections to your backing service.

D. Set Concurrency settings.
Feedback: Incorrect. Concurrency is how many users can connect to a particular instance. It does not directly affect connections to backend services.

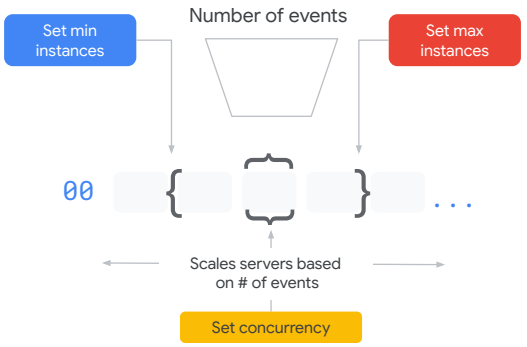
Where to look: https://cloud.google.com/run/docs/about-instance-autoscaling

Content mapping:

- Instructor-led Training/OnDemand
    - Google Cloud Fundamentals: Core Infrastructure
        - M6 Applications in the Cloud

Summary:
Explanation/summary on the following slide.

Cloud Run autoscaling



Google Cloud

How does autoscaling work in Cloud Run?

Cloud Run automatically scales the number of container instances required for each deployed revision. When no traffic is received, the deployment automatically scales to zero.

Other ways you can affect Cloud Run autoscaling:

- CPU utilization, with a default 60% utilization.
- Concurrency settings, with a default of 80 concurrent requests. You can increase it to 1000. You can also lower it if you need to.
- Max number of instances limits total number of instances. It can help you control costs and limit connections to a backing service. Defaults to 1000. Quota increase required if you want more.
- Min number of instances keeps a certain number of instances up. You will incur cost even when no instances are handling requests.

Instances that are started might remain idle for up to 15 minutes to reduce latency associated with cold starts. You don't get charged for these idle instances. You set a min and max on the container tab in the advanced settings dialog.

4.3 Managing Cloud Run resources

Courses	Documentation
Google Cloud Fundamentals: Core Infrastructure • M6 Applications in the Cloud	About container instance autoscaling | Cloud Run Documentation

Let’s take a moment to consider resources that can help you build your knowledge and skills in this area.

The concepts in the diagnostic question we just reviewed are covered in this modules and in this documentation. You’ll find this list in your workbook so you can take a note of what you want to include later when you build your study plan. Based on your experience with the diagnostic questions, you may want to include some or all of these.

https://cloud.google.com/run/docs/about-instance-autoscaling

4.4 Managing storage and database solutions

Tasks include:

- Managing and securing objects in and between Cloud Storage buckets
- Setting object life cycle management policies for Cloud Storage buckets
- Executing queries to retrieve data from data instances (e.g., Cloud SQL, BigQuery, Cloud Spanner, Cloud Datastore, Cloud Bigtable)
- Estimating costs of data storage resources
- Backing up and restoring database instances (e.g., Cloud SQL, Cloud Datastore)
- Reviewing job status in Cloud Dataproc, Cloud Dataflow, or BigQuery

Google Cloud

We discussed earlier about how to define an external table definition in BigQuery. Cymbal Superstore's transportation management app required you to do this so that you could query the location of Cymbal Superstore's delivery vehicles from Big Table.That is just one example of the type of storage management tasks you will have to do as an Associate Cloud Engineer

Consider this example. You store static images of products for Cymbal Superstore's ecommerce app in Cloud Storage. As an Associate Cloud Engineer, you would be expected to know how to secure access to these images from the application through IAM roles assigned to a service account. When you upgrade product images you would like to keep the previous images, but move them to a different storage type based on object versioning. You could do this using the object lifecycle management feature of Cloud Storage.

You explored these types of tasks in this question:
Question 8: Implement different types of Google Cloud Storage Lifecycle Actions (delete, set storage class) using lifecycle conditions