

Preparing for Your Associate Cloud Engineer Journey

Module 3: Deploying and Implementing Cloud Solutions



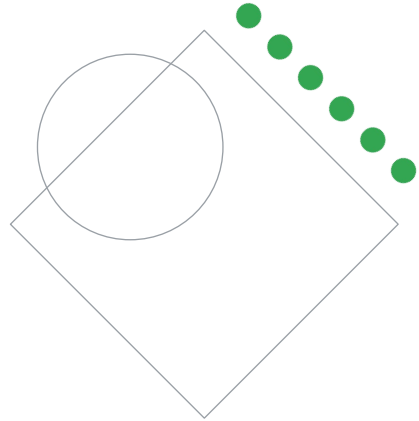
Welcome to Module 3: Deploying and Implementing Cloud Solutions.

Please complete the diagnostic questions now

- Forms are provided for you to answer the diagnostic questions
- The instructor will provide you a link to the forms
- The diagnostic questions are also available in the workbook



Review and study planning

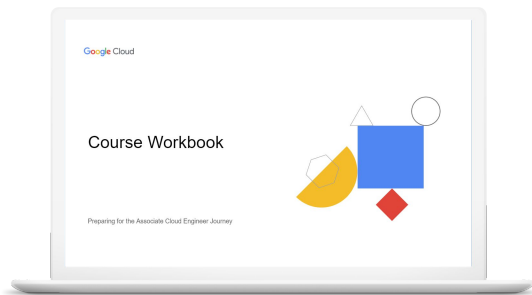


Google Cloud

There's much to cover in this section of the exam guide just as there's much for an Associate Cloud Engineer to do when deploying and implementing cloud solutions. Let's review the diagnostic questions to help you target your study time to focus on the areas you need to develop your skills the most.

Your study plan:

Deploying and implementing a cloud solution



- 3.1 Deploying and implementing Compute Engine resources
- 3.2 Deploying and implementing Google Kubernetes Engine resources
- 3.3 Deploying and implementing Cloud Run and Cloud Functions resources
- 3.4 Deploying and implementing data solutions
- 3.5 Deploying and implementing networking resources
- 3.6 Deploying a solution using Cloud Marketplace
- 3.7 Implementing resources via infrastructure as code

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.

As you can see, there are multiple objectives in this section that have many related tasks so you will probably need to plan for more study time.

3.1 | Deploying and implementing Compute Engine resources

Tasks include:

- Launching a compute instance using Cloud Console and Cloud SDK (GCloud - for example, assign disks, availability policy, SSH keys)
- Creating an autoscaled managed instance group using an instance template
- Generating/uploading a custom SSH key for instances
- Installing and configuring the Cloud Monitoring and Logging Agent
- Assessing compute quotas and requesting increases

Google Cloud

Cymbal Superstore uses Compute Engine for their supply chain application to Google Cloud because they need control over the operating system used by VMs.

Deploying Compute Engine resources can include a range of tasks such as launching compute instances through the console or cloud sdk and creating identical managed groups of instances based on an image template. Access requirements might have you implement SSH keys for you instances. Knowing how to deploy a monitoring agent on your instances is important to know so you can track performance and make changes when needed. Finally, if the number of instances starts bumping up against your project quotas you might need to request increases.

These are the diagnostic questions you answered that relate to this area:

Question 1: Describe how to configure VMs using Compute Engine machine types (settings such as memory and CPU, GPU if necessary, disk type, temp space)

Question 2: Apply concepts of managed instance groups, such as availability, scalability, and automated updates

3.1 Diagnostic Question 01 Discussion



Cymbal Superstore's sales department has a medium-sized MySQL database. This database includes user-defined functions and is used internally by the marketing department at Cymbal Superstore HQ. The sales department asks you to migrate the database to Google Cloud in the most timely and economical way.

What should you do?

- A. Find a MySQL machine image in Cloud Marketplace and configure it to meet your needs.
- B. Implement a database instance using Cloud SQL, back up your local data, and restore it to the new instance.
- C. Configure a Compute Engine VM with an N2 machine type, install MySQL, and restore your data to the new instance.
- D. Use gcloud to implement a Compute Engine instance with an E2-standard-8 machine type, install, and configure MySQL.

Google Cloud

Question:

Cymbal Superstore's sales department has a medium-sized MySQL database. This database includes user-defined functions and is used internally by the marketing department at Cymbal Superstore HQ. The sales department asks you to migrate the database to Google Cloud in the most timely and economical way. What should you do?

A. Find a MySQL machine image in Cloud Marketplace and configure it to meet your needs.

Feedback: Incorrect. This meets the requirements but is not the most timely way to implement a solution because it requires additional manual configuration.

B. Implement a database instance using Cloud SQL, back up your local data, and restore it to the new instance.

Feedback: Incorrect. Cloud SQL does not support user-defined functions, which are used in the database being migrated.

*C. Configure a Compute Engine VM with an N2 machine type, install MySQL, and restore your data to the new instance.

Feedback: Correct! N2 is a balanced machine type, which is recommended for medium-large databases.

D. Use gcloud to implement a Compute Engine instance with an E2-standard-8

machine type, install, and configure MySQL.

Feedback: Incorrect. E2 is a cost-optimized machine type. A recommended machine type for a medium-sized database is a balanced machine type.

Where to look: <https://cloud.google.com/compute/docs/>

Content mapping:

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

Summary:

Explanation/summary on the following slides.

Memory, CPU and GPU

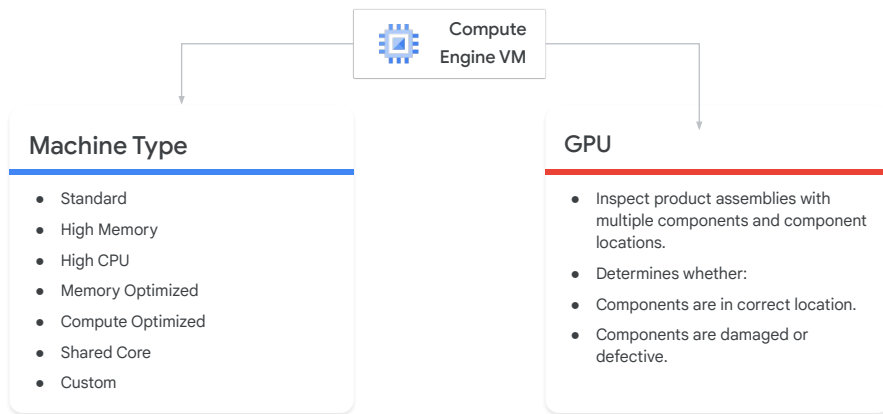
Machine type	CPU/Memory Ratio	GPU for compute workloads:
Standard	1 vCPU/3.75 GB memory	NVIDIA® A100
High Memory	1 vCPU/6.5 GB memory	NVIDIA® T4
High CPU	1 vCPU/.9 GB memory	NVIDIA® V100
Memory optimized	1 vCPU/>14 GB memory	NVIDIA® P100
Compute optimized	3.8 GHz all-core turbo	NVIDIA® P4
Shared Core	.2 vCPU/0.6 GB memory (f1)	NVIDIA® K80
Custom	Even # of CPU's/memory in increments of 256 MB	can be attached to predefined or custom machine types

Google Cloud

Compute Engine allows you to pick the amount of memory and CPU from predefined machine types. Machine types are divided into standard, high memory, high cpu, memory-optimized, compute-optimized or shared-core categories. If none of these meet your needs, you can also create a VM with the specific resources you need.

If you need GPU support for a compute-heavy workload, you can choose to attach GPUs to certain machine types. You can only use GPUs with general-purpose N1 VMs or accelerator-optimized A2 VMs. Availability of these machine types varies by zone, so make sure you pick a zone that has GPU capability.

Compute Engine options



Google Cloud

Compute Engine allows you to pick the amount of memory and CPU from predefined machine types. Machine types are divided into standard, high memory, high cpu, memory-optimized, compute-optimized or shared-core categories. If none of these meet your needs, you can also create a VM with the specific resources you need.

If you need GPU support for a compute-heavy workload, you can choose to attach GPUs to certain machine types. You can only use GPUs with general-purpose N1 VMs or accelerator-optimized A2 VMs. Availability of these machine types varies by zone, so make sure you pick a zone that has GPU capability.

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

Google Cloud

Storage options for your instances include zonal persistent disks, regional persistent disks, and local SSD. Persistent disks are built on network storage devices that are separate from the physical hardware your instance is running on. Each persistent disk references data distributed across several physical disks. Regional persistent disks share replicas of the physical disks across two zones, so you are protected from a single zone outage. Disk types you can attach to your virtual machine include standard (HDD), SSD, or local SSD. When you create a virtual machine instance in the console it uses balanced SSD, while when you create one via a `gcloud` command, it uses standard HDD. Balanced SSD gives you higher I/O than standard HDD, but less cost and I/O than fully capable SSD disks.

Local SSDs can be added to your instances based on machine type. They provide very high I/O since they are physically connected to the server your VM is running on. They are ephemeral, and thus go away when your VM is stopped or terminated. Data on your local SSD will survive a reboot. You are responsible for formatting and striping the local SSD per your requirements.

3.1 | Diagnostic Question 02 Discussion



The backend of Cymbal Superstore's e-commerce system consists of managed instance groups. You need to update the operating system of the instances in an automated way using minimal resources.

What should you do?

- A. Create a new instance template. Click **Update VMs**. Set the update type to Opportunistic. Click **Start**.
- B. Create a new instance template, then click **Update VMs**. Set the update type to PROACTIVE. Click **Start**.
- C. Create a new instance template. Click **Update VMs**. Set max surge to 5. Click **Start**.
- D. Abandon each of the instances in the managed instance group. Delete the instance template, replace it with a new one, and recreate the instances in the managed group.

Google Cloud

Questions:

The backend of Cymbal Superstore's e-commerce system consists of managed instance groups. You need to update the operating system of the instances in an automated way using minimal resources. What do you do?

A. Create a new instance template. Click **Update VMs**. Set the update type to Opportunistic. Click **Start**.

Feedback: Incorrect. Opportunistic updates are not interactive.

*B. Create a new instance template, then click **Update VMs**. Set the update type to PROACTIVE. Click **Start**.

Feedback: Correct! This institutes a rolling update where the surge is set to 1 automatically, which minimizes resources as requested.

C. Create a new instance template. Click **Update VMs**. Set max surge to 5. Click **Start**.

Feedback: Incorrect. Max surge creates 5 new machines at a time. It does not use minimal resources.

D. Abandon each of the instances in the managed instance group. Delete the instance template, replace it with a new one, and recreate the instances in the managed group.

Feedback: Incorrect. This is not an automated approach. The abandoned instances

are not deleted or replaced. It does not minimize resource use.

Where to look:

<https://cloud.google.com/compute/docs/instance-groups/creating-groups-of-managed-instances>

Content mapping:

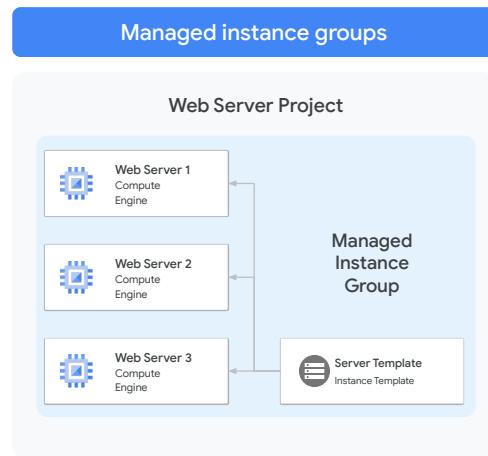
- Instructor-led Training/OnDemand
 - Architecting with Google Compute Engine
 - M9 Load Balancing and Autoscaling
 - M10 Infrastructure Automation

Summary:

Explanation/summary on the following slide.

Managed instance groups

- Deploy identical instances based on instance template
- Instance group can be resized
- Manager ensures all instances are RUNNING
- Typically used with autoscaler
- Can be single zone or regional



Google Cloud

Summary: Concepts

- **Availability** - a managed instance group ensures availability by keeping VM instances running. If a VM fails or stops, the MIG recreates it based on the instance template. You can make your MIG health checks application-based, which looks for an expected response from your application. The MIG will automatically recreate VMs that are not responding correctly. Another availability feature is spreading load across multiple zones using a regional MIG. Finally, you can use a load balancer to evenly distribute traffic across all instances in the group.
- **Scalability** - you can define autoscaling policies to grow instances in the group to meet demand. They can also scale back down when load drops which reduces cost.
- **Automated updates** - when it comes time to update software, automated updates lets you define how you will upgrade the instances in a group. You can specify how many resources to use and how many instances can be unavailable at the same time. Available update scenarios include rolling updates and canary updates. Rolling updates define how you want all instances eventually upgraded to the new template. Canary updates let you specify a certain number of instances to upgrade for testing purposes.

3.1 | Deploying and implementing Compute Engine resources

Courses

[Google Cloud Fundamentals: Core Infrastructure](#)

- M3 Virtual Machines in the Cloud

[Architecting with Google Compute Engine](#)

- M3 Virtual Machines
- M9 Load Balancing and Autoscaling
- M10 Infrastructure Automation



=

[Essential Google Cloud Infrastructure: Foundation](#)

- M3 Virtual Machines
- [Elastic Google Cloud Infrastructure: Scaling and Automation](#)
- M2 Load Balancing and Autoscaling
- M3 Infrastructure Automation



Documentation

[Compute Engine documentation | Compute Engine Documentation](#)

[Creating managed instance groups | Compute Engine 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 questions we just reviewed are covered in these modules 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/compute/docs/>

<https://cloud.google.com/compute/docs/instance-groups/creating-groups-of-managed-instances>

3.2 | Deploying and Implementing Google Kubernetes Engine resources

Tasks include:

- Installing and configuring the command line interface (CLI) for Kubernetes (kubectl)
- Deploying a Kubernetes Engine cluster with different configurations including AutoPilot, regional clusters, private clusters, etc.
- Deploying a containerized application to Kubernetes Engine
- Configuring Google Kubernetes Engine monitoring and logging

Google Cloud

Cymbal Superstore opted to migrate their on-premises, container-based e-commerce application to GKE. As an Associate Cloud Engineer, you should be comfortable with the Kubernetes CLI, kubectl, and the steps to deploy clusters and applications to GKE. You'll also need to configure monitoring and logging in GKE.

This diagnostic question addressed GKE deployments:

Question 3 Create a container development and management environment using Google Kubernetes Engine.

3.2 Diagnostic Question 03 Discussion



The development team for the supply chain project is ready to start building their new cloud app using a small Kubernetes cluster for the pilot. The cluster should only be available to team members and does not need to be highly available. The developers also need the ability to change the cluster architecture as they deploy new capabilities.

How would you implement this?

- A. Implement an autopilot cluster in us-central1-a with a default pool and an Ubuntu image.
- B. Implement a private standard zonal cluster in us-central1-a with a default pool and an Ubuntu image.
- C. Implement a private standard regional cluster in us-central1 with a default pool and container-optimized image type.
- D. Implement an autopilot cluster in us-central1 with an Ubuntu image type.

Google Cloud

Question:

The development team for the supply chain project is ready to start building their new cloud app using a small Kubernetes cluster for the pilot. The cluster should only be available to team members and does not need to be highly available. The developers also need the ability to change the cluster architecture as they deploy new capabilities. How would you implement this?

A. Implement an autopilot cluster in us-central1-a with a default pool and an Ubuntu image.

Feedback: Incorrect. Autopilot clusters are regional and us-central1-a specifies a zone. Also, autopilot clusters are managed at the pod level.

*B. Implement a private standard zonal cluster in us-central1-a with a default pool and an Ubuntu image.

Feedback: Correct! Standard clusters can be zonal. The default pool provides nodes used by the cluster.

C. Implement a private standard regional cluster in us-central1 with a default pool and container-optimized image type.

Feedback: Incorrect. The container-optimized image that supports autopilot type does not support custom packages.

D. Implement an autopilot cluster in us-central1 with an Ubuntu image type.

Feedback: Incorrect. Autopilot doesn't support Ubuntu image types.

Where to look:

<https://cloud.google.com/kubernetes-engine/docs/concepts/types-of-clusters>

Content mapping:

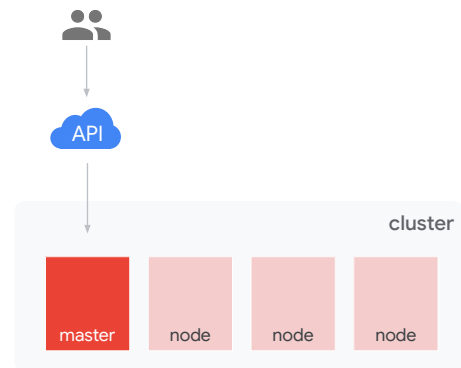
- Instructor-led Training/OnDemand
 - Google Cloud Fundamentals: Core Infrastructure
 - M5 Containers in the Cloud
 - Getting Started with GKE
 - M2 Introduction to Containers and Kubernetes
 - M3 Kubernetes Architecture
- Quests
 - Set Up and Configure a Cloud Environment in Google Cloud (<https://www.qwiklabs.com/quests/119>)

Summary:

Explanation/summary on the following slide.

You use Kubernetes APIs to deploy containers on a set of nodes called a cluster

- Masters run the control plane.
- Nodes run containers.
- Nodes are VMs (in GKE they're Compute Engine instances).
- You describe the apps, Kubernetes figures out how to make that happen.



- Mode - GKE has two modes to choose from: autopilot mode and standard mode. Autopilot is fully-provisioned and managed. You are charged according to the resources pods use as you deploy and replicate them based on the pod spec. Standard mode provides you flexibility to define and manage the cluster structure yourself.
- Availability - in a GKE cluster, availability deals with both the control plane and the distribution of your nodes. A zonal cluster has a single control plane in a single zone. You can distribute the nodes of a zonal cluster across multiple zones, providing node availability in case of a node outage. A regional cluster, on the other hand, has multiple replicas of the control plane in multiple zones with a given region. Nodes in a regional cluster are replicated across three zones, though you can change this behavior as you add new node pools.

Version: At setup you can choose to load a specific version of GKE or enroll in a release channel. If you don't specify either one of those, the current default version is chosen. It is a best practice to enable auto-upgrade for cluster nodes and the cluster itself.

Network routing: Routing between pods in GKE can be accomplished using alias IPs or Google Cloud Routes. The first option is also known as a VPC-native cluster, and the second one is called a routes-based cluster.

Network Isolation: Public GKE networks let you set up routing from public networks to your cluster. Private networks use internal addresses for pods and nodes and are isolated from public networks.

Features: Cluster features for Kubernetes will be either Alpha, Beta, or Stable, depending on their development status.

3.2 | Deploying and Implementing Google Kubernetes Engine resources

Courses

[Google Cloud Fundamentals: Core Infrastructure](#)

- M5 Containers in the Cloud

[Getting Started with Google Kubernetes Engine](#)

- M2 Introduction to Containers and Kubernetes
- M3 Kubernetes Architecture

Skill Badges



[Set Up and Configure a Cloud Environment in Google Cloud Quest](#)

Documentation

[Types of clusters | Kubernetes Engine 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 questions we just reviewed are covered in these modules 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/types-of-clusters>

3.3 | Deploying and implementing Cloud Run and Cloud Functions resources

Tasks include, where applicable:

- Deploying an application and updating scaling configuration, versions, and traffic splitting
- Deploying an application that receives Google Cloud events (for example, Pub/Sub events, Cloud Storage object change notification events)

Google Cloud

Cymbal Superstore's transportation management application uses Cloud Functions. An Associate Cloud Engineer should be able to deploy and implement serverless solutions, such as this one which receives Google Cloud events.

These types of tasks were covered in the following questions:

Question 4.) Differentiate among serverless options including App Engine standard and flexible environment, and Cloud Run.

Question 5.) Describe event function as a service capabilities of Cloud Functions.

3.3 Diagnostic Question 04 Discussion



You need to quickly deploy a containerized web application on Google Cloud. You know the services you want to be exposed. You do not want to manage infrastructure. You only want to pay when requests are being handled and need support for custom packages.

- A. App Engine Flexible
- B. App Engine Standard
- C. Cloud Run
- D. Cloud Functions

What technology meets these needs?

Google Cloud

Question:

You need to quickly deploy a containerized web application on Google Cloud. You know the services you want to be exposed. You do not want to manage infrastructure. You only want to pay when requests are being handled and need support for custom packages. What technology meets these needs?

A. App Engine Flexible

Feedback: Incorrect. App Engine Flexible does not scale to zero.

B. App Engine Standard

Feedback: Incorrect. App Engine Standard does not allow custom packages.

*C. Cloud Run

Feedback: Correct! Cloud Run is serverless, exposes your services as an endpoint, and abstracts all infrastructure.

D. Cloud Functions

Feedback: Incorrect. You do not deploy your logic using containers when developing for Cloud Functions. Cloud Functions executes small snippets of code in a serverless way.

Where to look:

<https://cloud.google.com/appengine/docs/the-appengine-environments>

<https://cloud.google.com/hosting-options>

<https://cloud.google.com/blog/topics/developers-practitioners/cloud-run-story-serverless-containers>

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 capabilities

- Serverless Container management
- Based on a service resource
- A service exposes an endpoint
 - Regional
 - Replicated across zones
- Scales based on incoming requests



Google Cloud

- Cloud Run provides a service to manage containers in a serverless way, which means you don't need to manage infrastructure when you deploy an application. The main resource container for Cloud Run is a service. A service is a regional resource that is replicated in multiple zones and exposed as an endpoint. Underlying infrastructure scales automatically based on incoming requests. If there are no requests coming in it can scale to zero to save you money.
- Changes to containers or environment settings create new revisions. Revisions can be rolled out in a way that supports canary testing by splitting traffic according to your specifications.
- Cloud Run is built using an open source initiative called knative. It is billed to the nearest 100 MS as you deploy containers to it.
- Cloud Run can use system libraries and tools made available to the container environment. It has a timeout of 60 minutes for longer running requests. Cloud Run can send multiple concurrent requests to each container instance, improving latency, and saving costs for large volumes of incoming traffic.

Another traditional serverless application manager available in Google Cloud is App Engine. App Engine has two management environments: standard and flexible.

In the standard environment, apps run in a sandbox using a specific language runtime. Standard environment is good for rapid scaling. It is limited to specific languages. It can scale to 0 when there is no incoming traffic. It starts up in seconds. In the standard environment you are not allowed to make changes to the runtime.

In contrast to the standard environment, App Engine flexible runs in Docker containers in Compute Engine VMs. Flexible supports more programming languages. It can use native code and you can access and manage the underlying Compute Engine resource base.

App Engine flexible does not scale to 0. Startup is in minutes. Deployment time is in minutes (longer than standard). It does allow you to modify the runtime environment.

3.3 Diagnostic Question 05 Discussion



You need to analyze and act on files being added to a Cloud Storage bucket. Your programming team is proficient in Python. The analysis you need to do takes at most 5 minutes. You implement a Cloud Function to accomplish your processing and specify a trigger resource pointing to your bucket.

- A. `--trigger-event google.storage.object.finalize`
- B. `--trigger-event google.storage.object.create`
- C. `--trigger-event google.storage.object.change`
- D. `--trigger-event google.storage.object.add`

How should you configure the `--trigger-event` parameter using `gcloud`?

Google Cloud

Question:

You need to analyze and act on files being added to a Cloud Storage bucket. Your programming team is proficient in Python. The analysis you need to do takes at most 5 minutes. You implement a Cloud Function to accomplish your processing and specify a trigger resource pointing to your bucket. How should you configure the `--trigger-event` parameter using `gcloud`?

*A. `--trigger-event google.storage.object.finalize`

Feedback: Correct! Finalize event trigger when a write to Cloud Storage is complete.

B. `--trigger-event google.storage.object.create`

Feedback: Incorrect. This is not a cloud storage notification event.

C. `--trigger-event google.storage.object.change`

Feedback: Incorrect. This is not a cloud storage notification event.

D. `--trigger-event google.storage.object.add`

Feedback: Incorrect. This is not a cloud storage notification event.

Where to look:

<https://cloud.google.com/blog/topics/developers-practitioners/learn-cloud-functions-snippets>

<https://cloud.google.com/functions>

Content mapping:

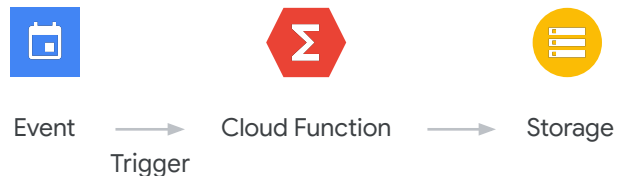
- Instructor-led Training/OnDemand
 - Google Cloud Fundamentals: Core Infrastructure
 - M7 Developing, Deploying, and Monitoring in the Cloud

Summary:

Explanation/summary on the following slide.

Cloud Functions capabilities

- Serverless function execution
- Event based
- Functions trigger when an event occurs
- Scales by number of events received
- Functions are stateless - need to persist data if you need to share it outside the function



Google Cloud

- Cloud Functions is Google Cloud's answer to serverless functions. It is a fully managed service based on events that happen across your cloud environment, including services and infrastructure. The functions you develop run in response to those events. There are no servers to manage or scaling to configure. The service provides the underlying resources required to execute your function.
- A trigger sends an https request to an endpoint the service is listening on. This endpoint then responds by deploying and executing the function and returning the results specified in your code. Pricing is based on the number of events, compute time, and memory required in network ingress/egress. If no requests are coming in, your function doesn't cost anything.
- Cloud Functions use cases include IoT processing and lightweight ETL.
- Depending on the programming language you choose, the Cloud Functions service provides a base image and runtime that will be updated and patched automatically. This helps keep execution of your deployed functions secure.
- By design, functions you write for use by the Cloud Functions service are stateless. If you need to share and persist data across function runs, you should consider using Datastore or Cloud Storage. Each Cloud Function instance handles only one concurrent request at a time. If, while handling a request, another one comes in, Cloud Functions will ask for more instances to be created. This is another reason functions need to be stateless, because they can run on different instances. You can implement minimum instance limits to avoid latency associated with cold starts.

3.3 | Deploying and implementing Cloud Run and Cloud Functions resources

Courses

[Google Cloud Fundamentals: Core Infrastructure](#)

- M6 Applications in the Cloud
- M7 Developing, Deploying, and Monitoring in the Cloud

Documentation

[Choose an App Engine environment | App Engine Documentation](#)

[Application Hosting Options](#)

[Cloud Run: What no one tells you about Serverless \(and how it's done\)](#)

[Learn Cloud Functions in a snap! Cloud Functions](#)

Now that we've reviewed the diagnostic questions related to Section 3.3 Deploying and implementing Cloud Run and Cloud Functions resources, 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 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/appengine/docs/the-appengine-environments>

<https://cloud.google.com/hosting-options>

<https://cloud.google.com/blog/topics/developers-practitioners/cloud-run-story-serverless-containers>

<https://cloud.google.com/blog/topics/developers-practitioners/learn-cloud-functions-snap>

<https://cloud.google.com/functions>

3.4 | Deploying and implementing data solutions

Tasks include:

- Initializing data systems with products (for example, Cloud SQL, Firestore, BigQuery, Cloud Spanner, Pub/Sub, Cloud Bigtable, Cloud Dataproc, Cloud Dataflow, Cloud Storage)
- Loading data (for example, command line upload, API transfer, import/export, load data from Cloud Storage, streaming data to Pub/Sub)

Google Cloud

Cymbal Superstore has several different data requirements based on the storage needs of their different applications. Their e-commerce system is slated to use Cloud Spanner. They need to do analytics on historical data using BigQuery. They need to store their IOT truck data in BigTable. Their supply chain management system needs a Cloud SQL store. As an Associate Cloud Engineer, you'll need to be able to deploy and implement a wide range of data solutions.

Question tested your knowledge of the steps for setting up a Cloud Storage bucket, question 7 the steps for setting up an instance using Cloud SQL, and question 8 the steps to load data to a BigQuery table.

3.4 | Diagnostic Question 06 Discussion



You require a Cloud Storage bucket serving users in New York City. There is a need for geo-redundancy. You do not plan on using ACLs.

What CLI command do you use?

- A. Run a **gcloud mb** command specifying the name of the bucket and accepting defaults for the other mb settings.
- B. Run a **gsutil mb** command specifying a multi-regional location and an option to turn ACL evaluation off.
- C. Run a **gsutil mb** command specifying a dual-region bucket and an option to turn ACL evaluation off.
- D. Run a **gsutil mb** command specifying a dual-region bucket and accepting defaults for the other mb settings.

Google Cloud

Question:

You require a Cloud Storage bucket serving users in New York City. There is a need for geo-redundancy. You do not plan on using ACLs. What CLI command do you use?

A. Run a `gcloud mb` command specifying the name of the bucket and accepting defaults for the other mb settings.

Feedback: Incorrect. `gcloud` is not used to create buckets.

B. Run a `gsutil mb` command specifying a multi-regional location and an option to turn ACL evaluation off. Feedback: Incorrect. Most users are in NY. Multi-regional location availability of "US" is not required.

*C. Run a `gsutil mb` command specifying a dual-region bucket and an option to turn ACL evaluation off. Feedback: Correct! NAM4 implements a dual-region bucket with `us-east1` and `us-central1` as the configured regions.

D. Run a `gsutil mb` command specifying a dual-region bucket and accepting defaults for the other mb settings Feedback: Incorrect. This command is missing the `-b` option that disables ACLs as required in the example.

Where to look:

<https://cloud.google.com/storage/docs/creating-buckets>

<https://cloud.google.com/storage/docs/introduction>

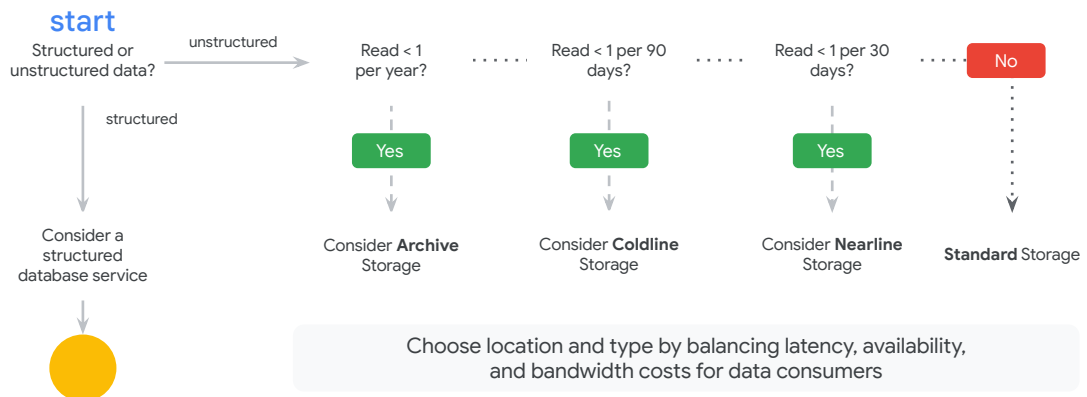
Content mapping:

- Instructor-led Training/OnDemand
 - Google Cloud Fundamentals: Core Infrastructure
 - M4 Storage in the Cloud
- Quests
 - Perform Foundational Infrastructure Tasks in Google Cloud Quest (<https://www.qwiklabs.com/quests/118>)

Summary:

Explanation/summary on the following slide.

Choosing a storage class



Google Cloud

Cloud Storage helps you store binary objects in Google Cloud. It can house data in any format as an immutable object. In Cloud Storage, objects are stored in containers called buckets.

Buckets can be used to upload and download objects, and permissions can be assigned to specify who has access to them.

You can manage and interact with Cloud Storage via the console, via the command line and the gsutil command set, via client libraries, or through APIs. Steps for creating a cloud bucket include:

1. Naming your bucket - has to be globally unique, and not contain any sensitive information
2. Choose location type and option:
 - Regional is a specific geographical area where a datacenter campus resides. It minimizes latency and network bandwidth for consumers grouped in a specific region
 - Dual-region is a specific pair of regions. It provides geo-redundancy
 - Multi-region is a disbursed geographic area, such as the US or Europe. You can use it to serve content to consumers outside of Google and distributed over large areas.
3. Pick a default storage class for bucket. You can override this per object.
 - Standard is for immediate access and has no minimum storage

- duration
- Nearline has a 30 day minimum duration and data retrieval charges
- Coldline has a 90 day min duration and data retrieval charges
- Archive has a 365 day min duration and data retrieval charges

4. Click **Create** or submit the command.

3.4 Diagnostic Question 07 Discussion



Cymbal Superstore asks you to implement Cloud SQL as a database backend to their supply chain application. You want to configure automatic failover in case of a zone outage. You decide to use the **`gcloud sql instances create`** command set to accomplish this.

- A. `--availability-type`
- B. `--replica-type`
- C. `--secondary-zone`
- D. `--master-instance-name`

Which `gcloud` command line argument is required to configure the stated failover capability as you create the required instances?

Google Cloud

Question:

Cymbal Superstore asks you to implement Cloud SQL as a database backend to their supply chain application. You want to configure automatic failover in case of a zone outage. You decide to use the **`gcloud sql instances create`** command set to accomplish this. Which `gcloud` command line argument is required to configure the stated failover capability as you create the required instances?

*A. `--availability-type`

Feedback: Correct! This option allows you to specify zonal or regional availability, with regional providing automatic failover to a standby node in another region.

B. `--replica-type`

Feedback: Incorrect. If you have `--master-instance-name`, this option allows you to define the replica type: a default of read, or a legacy MySQL replica type of failover, which has been deprecated.

C. `--secondary-zone`

Feedback: Incorrect. This is an optional argument that is valid only when you have a specified availability type: regional.

D. `--master-instance-name`

Feedback: Incorrect. This option creates a read replica based on the master instance. It replicates data but does not automate failover.

Where to look:

<https://cloud.google.com/sql/docs/mysql/features>

<https://cloud.google.com/sql/docs/mysql/create-instance>

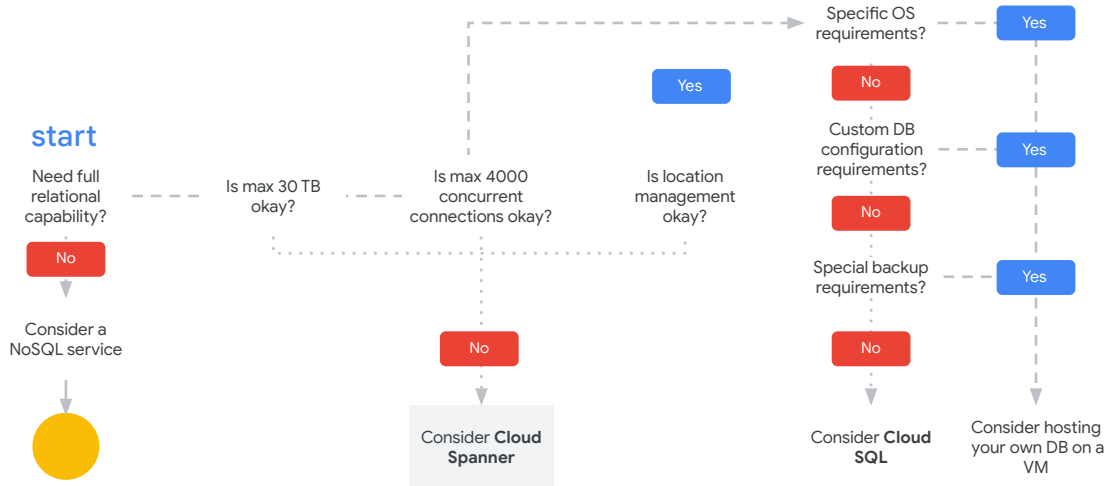
Content mapping:

- Instructor-led Training/OnDemand
 - Google Cloud Fundamentals: Core Infrastructure
 - M4 Storage in the Cloud
 - Architecting with Google Compute Engine
 - M5 Storage and Database Services
- Quests
 - Set Up and Configure a Cloud Environment in Google Cloud (<https://www.qwiklabs.com/quests/119>)

Summary:

Explanation/summary on the following slide.

Choosing Cloud SQL



Google Cloud

Cloud SQL is a Google Cloud service that manages a database instance for you. You are responsible for how you structure your data within it. Cloud SQL can handle common database tasks for you, such as automating backups, implementing high availability, handling data encryption, updating infrastructure and software, and providing logging and monitoring services. You can use it to deploy MySQL, PostgreSQL, or SQL Server databases to Google Cloud. It uses persistent disks attached to underlying Compute Engine instances to house your database, and implements a static IP address for you to connect to it.

Steps for setting up a Cloud SQL instance:

1. Create Instance.
2. Select database type.
3. Enter name: do not use sensitive or personally identifiable information. Your instance name can be publicly available.
4. Enter password for root user.
5. Select proper version: choose carefully, it cannot be edited.
6. Regional and zonal availability settings. Can't be modified. Pick a region where most people will access. You can also choose multi-region.
7. Select both primary and secondary zone. Both default to any with secondary being different than the primary.
8. Config settings include machine type, private or public ip, storage type, storage capacity, threshold for automated storage increase, as well as an increase setting to specify a limit on how big your database grows.

3.4 Diagnostic Question 08 Discussion



Cymbal Superstore's marketing department needs to load some slowly changing data into BigQuery. The data arrives hourly in a Cloud Storage bucket. You want to minimize cost and implement this in the fewest steps.

What should you do?

- A. Implement a **bq load** command in a command line script and schedule it with cron.
- B. Read the data from your bucket by using the BigQuery streaming API in a program.
- C. Create a Cloud Function to push data to BigQuery through a Dataflow pipeline.
- D. Use the BigQuery data transfer service to schedule a transfer between your bucket and BigQuery.

Google Cloud

Question:

Cymbal Superstore's marketing department needs to load some slowly changing data into BigQuery. The data arrives hourly in a Cloud Storage bucket. You want to minimize cost and implement this in the fewest steps. What should you do?

A. Implement a **bq load** command in a command line script and schedule it with cron.
Feedback: Incorrect. This solution doesn't cost anything but is more complex than setting up a data transfer.

B. Read the data from your bucket by using the BigQuery streaming API in a program.
Feedback: Incorrect. The streaming API has pricing associated with it based on how much data you stream in.

C. Create a Cloud Function to push data to BigQuery through a Dataflow pipeline.
Feedback: Incorrect. A Dataflow pipeline will incur charges for the resources performing the sink into BigQuery.

*D. Use the BigQuery data transfer service to schedule a transfer between your bucket and BigQuery.
Feedback: Correct! BigQuery transfer service is the simplest process to set up transfers between Cloud Storage and BigQuery. It is encompassed by one command. It is also free.

Where to look:

<https://cloud.google.com/blog/topics/developers-practitioners/bigquery-explained-data-ingestion>

<https://cloud.google.com/bigquery/docs/loading-data>

Content mapping:

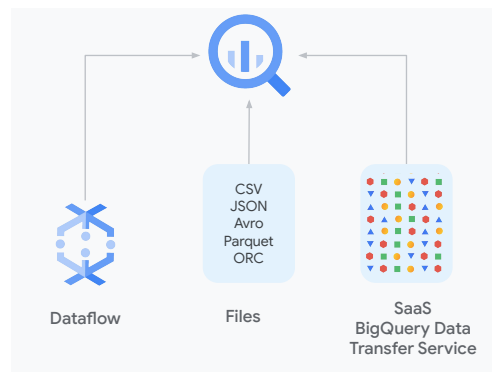
- Instructor-led Training/OnDemand
 - Google Cloud Fundamentals: Core Infrastructure
 - M8 Big Data and Machine Learning in the Cloud
- Quests
 - Set Up and Configure a Cloud Environment in Google Cloud (<https://www.qwiklabs.com/quests/119>)

Summary:

Explanation/summary on the following slide.

Ways to batch load data into BigQuery

Loading data into BigQuery tables (batch, periodic) offers the best performance.



Google Cloud

BigQuery offers several ways to load data. The first, and oldest, is loading data in batch. Batch loading is done in a single operation. It can be used to load data from csv files, databases, and log files. Batch loading is often used when you need to ingest files from your local computer. It is the recommended way to load slowly changing data. There is no charge for batch loads into BigQuery.

The ways you can implement a batch load in BigQuery include:

1. Create a load job.
2. Use BigQuery Data Transfer Service from Software as a Service products. This is the simplest approach.
3. Use Cloud Composer, a Google Cloud managed version of Apache Airflow.
4. Use the bq command line tool and the cron scheduler on the command line interface.
5. Use BigQuery connectors for big data products such as Spark or Hadoop.

For real-time use cases you can stream data into BigQuery using the Streaming API. Data streamed into BigQuery can be used for querying immediately. The streaming API can be used to track and query application events or log stream information.

The third way to ingest data into BigQuery is to use Dataflow and Apache Beam. These two technologies define a processing pipeline where your source or sink can be BigQuery. A possible use-case for this is to trigger a Cloud Function when an event happens. The Cloud Function could contain the logic to start an Apache Beam pipeline using a Dataflow runner that would execute transformations required and

then save your data into BigQuery when it is done.

Another way to load data is to execute queries in BigQuery native storage or federated queries on external data and save the results to a table. CTAS (create table as select) is a way to do this using DML as well.

Finally, many third party applications have connectors you can use to get data in to Big Query. You would need to look at the documentation of the product you want to ingest data from.

Reference slide from Data Engineering Course Module 3

3.4 | Deploying and implementing data solutions

Courses

[Google Cloud Fundamentals: Core Infrastructure](#)

- M4 Storage in the Cloud
- M8 Big Data and Machine Learning in the Cloud

[Architecting with Google Compute Engine](#)



- M5 Storage and Database Services



[Essential Google Cloud Infrastructure: Core Services](#)

- M2 Storage and Database Services



Skill Badges



[Perform Foundational Infrastructure Tasks in Google Cloud Quest](#)



[Set Up and Configure a Cloud Environment in Google Cloud Quest](#)

Documentation

[Creating storage buckets | Cloud Storage](#)

[What is Cloud Storage?](#)

[Cloud SQL for MySQL features](#)

[Creating instances | Cloud SQL for MySQL](#)

[How to load, import, or ingest data into BigQuery for analysis](#)

[Introduction to loading data | BigQuery](#)

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/storage/docs/creating-buckets>

<https://cloud.google.com/storage/docs/introduction>

<https://cloud.google.com/sql/docs/mysql/features>

<https://cloud.google.com/sql/docs/mysql/create-instance>

<https://cloud.google.com/blog/topics/developers-practitioners/bigquery-explained-data-ingestion>

<https://cloud.google.com/bigquery/docs/loading-data>

3.5 | Deploying and implementing networking resources

Tasks include:

- Creating a VPC with subnets (for example, custom-mode VPC, shared VPC)
- Launching a Compute Engine instance with custom network configuration (for example, internal-only IP address, Google private access, static external and private IP address, network tags)
- Creating ingress and egress firewall rules for a VPC (for example, IP subnets, network tags, service accounts)
- Creating a VPN between a Google VPC and an external network using Cloud VPN
- Creating a load balancer to distribute application network traffic to an application (for example, Global HTTP(S) load balancer, Global SSL Proxy load balancer, Global TCP Proxy load balancer, regional network load balancer, regional internal load balancer)

Google Cloud

Knowing how to connect your cloud resources, services, and users is an important aspect of being an Associate Cloud Engineer. Cymbal Superstore needs to implement global external http(s) access to their e-commerce system and regional external http(s) access to their supply chain.

Question 9 tested your knowledge of creating a VPC with subnets.

3.5 | Diagnostic Question 09 Discussion



Which Virtual Private Cloud (VPC) network type allows you to fully control IP ranges and the definition of regional subnets?

- A. Default Project network
- B. Auto mode network
- C. Custom mode network
- D. An auto mode network converted to a custom network

Google Cloud

Question:

Which Virtual Private Cloud (VPC) network type allows you to fully control IP ranges and the definition of regional subnets?

A. Default Project network

Feedback: Incorrect. A project's default network is an auto mode network that creates one subnet in each Google Cloud region automatically with a predetermined set of IP ranges.

B. Auto mode network

Feedback: Incorrect. An auto mode network creates one subnet in each Google Cloud region automatically with a predetermined set of IP ranges.

*C. Custom mode network

Feedback: Correct! A custom mode network gives you control over regions that you place your subnets in and lets you specify IP ranges for them as well.

D. An auto mode network converted to a custom network

Feedback: Incorrect. An auto mode network converted to a custom network retains the currently assigned IP addresses and requires additional steps to change subnet characteristics.

Where to look: <https://cloud.google.com/vpc/docs/vpc>

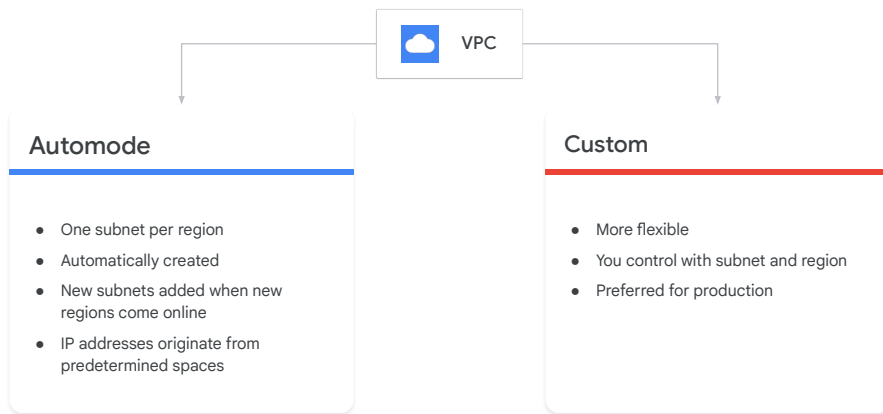
Content mapping:

- Instructor-led Training/OnDemand
 - Architecting with Google Compute Engine
 - M2 Virtual Networks
- Quests
 - Set Up and Configure a Cloud Environment in Google Cloud (<https://www.qwiklabs.com/quests/119>)

Summary:

Explanation/summary on the following slide.

Compute Engine options



Google Cloud

Virtual private clouds are part of Google's software defined network environment. VPCs provide connectivity to Compute Engine instances. They offer internal TCP/UDP load balancing systems. They allow you to implement Cloud VPN tunnels to communicate with your on-premises network, and they distribute external traffic to backend servers.

There are two types of network configurations that are available when you decide to create a new VPC.

Auto mode networks create one subnet in each region automatically. New subnets are added automatically when new regions come on line. IP addresses are created from a predetermined set of address spaces. The default VPC created when you create a project is an auto mode VPC.

The benefits of using auto mode include being easy to set up and use and subnets are created in each region. You need to ensure IP ranges do not overlap with on-premise resources.

The other kind of network is a custom network. When you create a custom network, you create and configure the subnets you want and only in the regions you want. If you try to create instances in a region that doesn't have a subnet defined, you will get an error message. Custom networks are recommended for production environments. Custom networks are a good choice when you don't need subnets in every region.

You can convert an auto mode to a custom network, but not the reverse.

3.5 | Deploying and implementing networking resources

Courses

[Architecting with Google Compute Engine](#)

- M2 Virtual Networks



=

[Essential Google Cloud Infrastructure: Foundation](#)

- M2 Virtual Networks



Skill Badges



[Set Up and Configure a Cloud Environment in Google Cloud Quest](#)

Documentation

[VPC network overview](#)

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 this module, Skill Badge, 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/vpc/docs/vpc>

3.6 | Deploying a solution using Cloud Marketplace

Tasks include:

- Browsing Cloud Marketplace catalog and viewing solution details
- Deploying a Cloud Marketplace solution

Google Cloud

Cloud Marketplace keeps you from reinventing the wheel if you have a software stack that you need supported on a Compute Engine VM instance. Cymbal Superstore could use Cloud Marketplace to find a LAMP stack to support the migration of their supply chain management system.

We didn't have any questions specifically about Cloud Marketplace, but you should be familiar with deploying a Cloud Marketplace solution.

3.7 | Implementing resources via infrastructure as code

- Building infrastructure via Cloud Foundation Toolkit templates and implementing best practices
- Installing and configuring Config Connector in GKE to create, update, delete, and secure resources

Google Cloud

Interacting directly with Google cloud console is great, but for repetitive tasks it might be better to be able to implement resources in an automated way. Modern DevOps processes have made declarative resource allocation a reality. Tools like Terraform take the resource requirements you provide via a configuration file. These infrastructure as code systems worry about how these resources are built. You just have to specify what you want built. For Cymbal Superstore, building dev and test environment resources in a declarative fashion speeds things up as you are moving your application through different unit and functional tests. You can also reset those resources to specific configurations you need depending on what you want to test.

You explored these types of tasks in question 10.

3.5 | Diagnostic Question 10 Discussion



What action does the **terraform** apply command perform?

- A. Downloads the latest version of the terraform provider.
- B. Verifies syntax of terraform config file.
- C. Shows a preview of resources that will be created.
- D. Sets up resources requested in the terraform config file.

Google Cloud

Question:

What action does the **terraform apply** command perform?

A. Downloads the latest version of the terraform provider.

Feedback: Incorrect. Terraform init downloads the latest version.

B. Verifies syntax of terraform config file.

Feedback: Incorrect. Terraform plan verifies the syntax.

C. Shows a preview of resources that will be created.

Feedback: Incorrect. Terraform plan outputs a preview of resources.

*D. Sets up resources requested in the terraform config file.

Feedback: Correct! Terraform Apply sets up resources specified in the terraform config file.

Where to look:

<https://www.terraform.io/intro/index.html>

<https://cloud.google.com/docs/terraform>

Content mapping:

- Instructor-led Training/OnDemand
 - Architecting with Google Compute Engine

■ M11 Infrastructure Automation

Summary:

Explanation/summary on the following slide.

Terraform lifecycle



Google Cloud

Terraform is an open source tool for implementing resources in a declarative way. You specify a Terraform config file that describes the resources you want to deploy. Terraform is known as an infrastructure as code service.

A benefit of implementing resources in this way is that your configuration files can be source controlled, thus following devops best practices.

In Google Cloud, you store your Terraform config files in a Cloud Storage bucket with object versioning enabled. Cloud Build submits Terraform commands via a YAML file. It needs access to the Cloud Storage bucket where your Terraform config files are stored.

The different files required for implementing Terraform in Google Cloud include:

- Cloudbuild.yaml - build configuration file that contains instructions for Cloud Build
- Backend.tf - stores remote Terraform state information
- Terraform.tfstate - local file that stores Terraform state
- Main.tf - contains the terraform config

Commands ran by Cloud Build include:

- Terraform init - downloads latest version of Terraform provider
- Terraform plan - verifies syntax, ensures supporting files exist, shows a preview of resources that will be created
- Terraform apply - sets up requested resources output by the Terraform plan
- Terraform destroy - destroy all resources in the specified config file

3.7 | Implementing resources via infrastructure as code

Courses

[Architecting with Google Compute Engine](#)

- M10 Infrastructure Automation



=

[Elastic Google Cloud Infrastructure: Scaling and Automation](#)

- M3 Infrastructure Automation



Documentation

[Introduction](#)

[Using Terraform with Google Cloud](#)

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 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://www.terraform.io/intro/index.html>

<https://cloud.google.com/docs/terraform>