Preparing for the Associate Cloud Engineer (ACE) Exam: Deploying and implementing a cloud solution

1. Section 3.1 - Deploying and implementing Compute Engine resources.
2. Section 3.2 - Deploying and implementing Kubernetes Engine resources.
3. Section 3.3 - Deploying and implementing App Engine and Cloud Functions resources
4. Section 3.4 - Deploying and implementing data solutions

5 Section 3.5 - Deploying and implementing networking resources

6 Section 3.6 - Deploying a Solution using Cloud Launcher

7 Section 3.7 - Deploying an Application using Deployment Manager

1 **Section 3.1 - Deploying and implementing Compute Engine resources.**

2 Section 3.2 - Deploying and implementing Kubernetes Engine resources.

3 Section 3.3 - Deploying and implementing App Engine and Cloud Functions resources

4 Section 3.4 - Deploying and implementing data solutions

Exam Guide: Section 3.1

3.1 Deploying and implementing Compute Engine resources. Tasks include:

- Launching a compute instance using Cloud Console and Cloud SDK (gcloud) (e.g., assign disks, availability policy, SSH keys).
- Creating an autoscaled managed instance group using an instance template.
- Generating/uploading a custom SSH key for instances.
- Configuring a VM for Stackdriver monitoring and logging.
- Assessing compute quotas and requesting increases.
- Installing the Stackdriver Agent for monitoring and logging.

Exam Guide: Section 3.1

3.1 Deploying and implementing Compute Engine resources. Tasks include:

- **Launching a compute instance using Cloud Console and Cloud SDK (gcloud) (e.g., assign disks, availability policy, SSH keys).**
- **Creating an autoscaled managed instance group using an instance template.**
- Generating/uploading a custom SSH key for instances.
- Configuring a VM for Stackdriver monitoring and logging.
- Assessing compute quotas and requesting increases.
- Installing the Stackdriver Agent for monitoring and logging.

Compute Engine offers managed virtual machines

- No upfront investment
- Fast and consistent performance

Compute Engine lets you create and run virtual machines on Google infrastructure. There are no upfront investments, and you can run thousands of virtual CPUs on a system that is designed to be fast and to offer consistent performance.

Compute Engine offers managed virtual machines

- No upfront investment
- Fast and consistent performance
- Create VMs with GCP Console or **gcloud**
- Run images of Linux or Windows Server

You can create a virtual machine instance by using the Google Cloud Platform Console or the gcloud command-line tool. Your VM can run Linux and Windows Server images provided by Google or customized versions of these images, and you can even import images from many of your physical servers.

Compute Engine offers managed virtual machines

- Pick memory and CPU: use predefined types, or make a custom VM
- Pick GPUs if you need them

When you create a VM, pick a machine type, which determines how much memory and how many virtual CPUs it has. These types range from very small to very large indeed! And if you can't find a predefined type that meets your needs perfectly, you can make a custom VM.

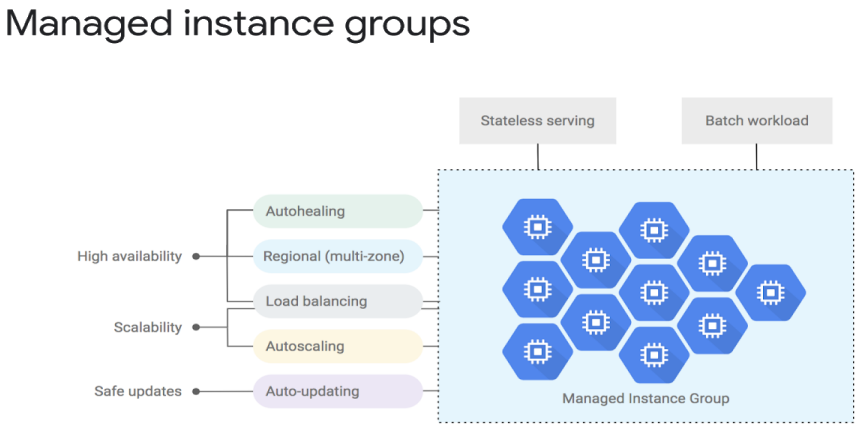
Speaking of processing power, if you have workloads like machine learning and data processing that can take advantage of GPUs, many GCP zones have GPUs available for you.

Compute Engine offers managed virtual machines

- Pick memory and CPU: use predefined types, or make a custom VM
- Pick GPUs if you need them
- Pick persistent disks: standard or SSD
- Pick local SSD for scratch space too if you need it

Just like physical computers need disks, so do VMs. You can choose two kinds of persistent storage: standard or SSD. If your application needs high-performance scratch space, you can

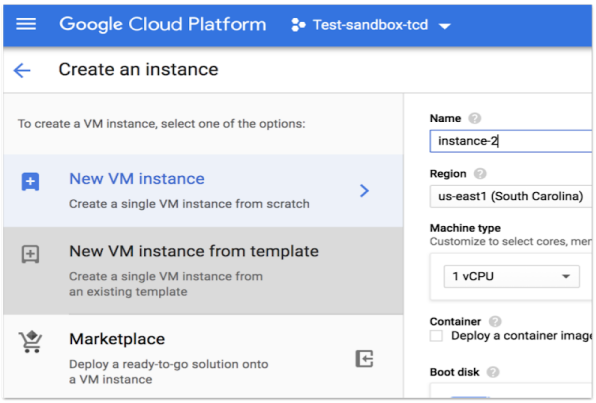
attach a local SSD, but be sure to store data of permanent value somewhere else, because local SSDs' content doesn't last past when the VM terminates. That's why the other kinds are called "persistent disks." Anyway, most people start off with standard persistent disks, and that's the default.



There are two kinds of VM instance groups: un-managed and managed.
Managed instance groups (MIGs) allow you to operate applications on multiple identical VMs. They offer high availability, scalability (using Autoscaling) and automated updates.

Create VMs from instance templates

- Quickly create multiple VMs from pre-existing configurations
- Templates pre-define machine type, boot disk image, labels and other properties
- Create managed instance groups automatically for autoscaling

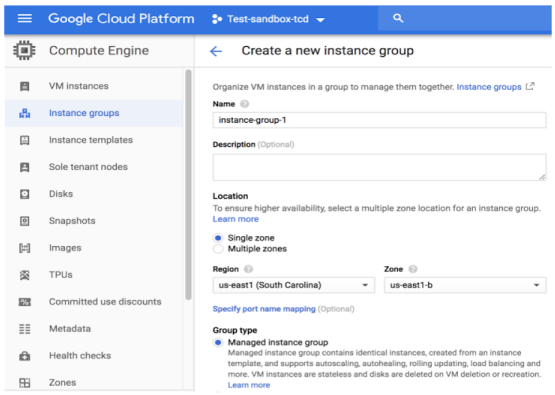


You can use instance templates any time you want to quickly create VM instances based on a preexisting configuration.
Instance templates define the machine type, boot disk image or container image, labels, and other instance properties.
If you want to create a group of identical instances, you must use an instance template to create a managed instance group.

Instance templates are designed to create instances with identical configurations, so it is not possible to update an existing instance template or change an instance template after it has been created.

If an instance template goes out of date, or you need to make changes to the configuration, create a new instance template.

Creating managed instance groups with templates



A managed instance group contains identical instances that you can manage as a single entity in a single zone.
Managed instance groups maintain high availability of your apps by proactively keeping your instances available, which means in RUNNING state.
Managed instance groups support autoscaling, load balancing, rolling updates, autohealing, and more.
You can also create regional managed instance groups, which contain instances across multiple zones within the same region.
Go to the Instance Groups page in the GCP Console, and click Create an instance group.
Enter a name for the managed instance group, and select the zone where you want to locate the group.
Under Group type, select Managed instance group.
Under Instance template, select an instance template. If no templates are available, create one.
Specify the number of instances that you want to create in the group. Optionally, you can enable Autoscaling so that the group will automatically add or remove based on instance CPU usage or autohealing to perform health checking on instances within the instance group.
Click Create to create the new group.

1 Section 3.1 - Deploying and implementing Compute Engine resources.

2 **Section 3.2 - Deploying and implementing Kubernetes Engine resources.**

3 Section 3.3 - Deploying and implementing App Engine and Cloud Functions resources

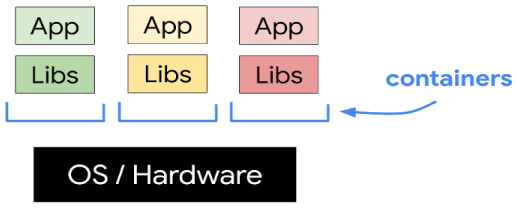
4 Section 3.4 - Deploying and implementing data solutions

Exam Guide: Section 3.2

3.2 Deploying and implementing Kubernetes Engine resources. Tasks include:

- Deploying a Kubernetes Engine cluster.
- Deploying a container application to Kubernetes Engine using pods.
- Configuring Kubernetes Engine application monitoring and logging.

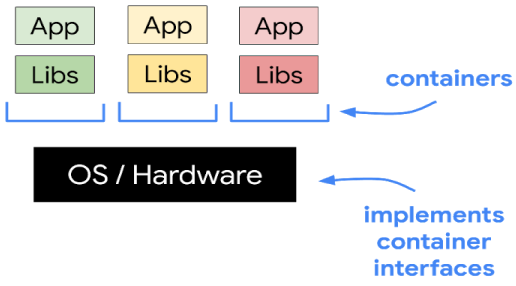
What are Containers?



Containers give you the independent scalability of workloads in Platform as a Service (PaaS) and an abstraction layer of the OS and hardware in Infrastructure as a Service (IaaS).

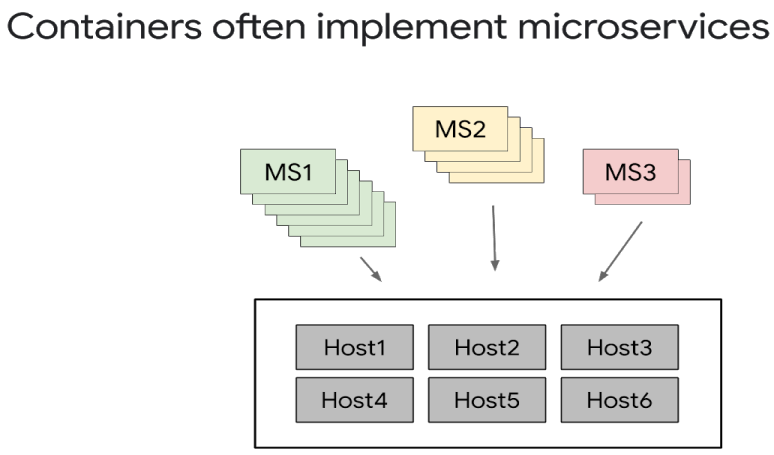
Containers give you an invisible “box” around your code and its dependencies, with limited access to its own partition of the file system and hardware.
It only requires a few system calls to create and it starts as quickly as a process.

What are Containers?

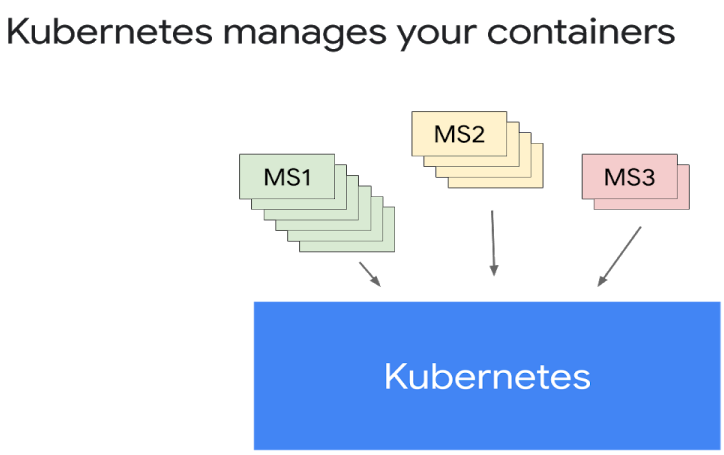


All you need on each host is an OS kernel that supports containers and a container runtime.

In essence, you are virtualizing the OS. It scales like PaaS, but gives you nearly the same flexibility as IaaS.
With this abstraction, your code is ultra portable and you can treat the OS and hardware as a black box.
So you can go from development, to staging, to production, or from your laptop to the cloud, without changing or rebuilding anything.



If you want to scale, for example, a web server, you can do so in seconds and deploy dozens or hundreds of them (depending on the size or your workload) on a single host.
Now that's a simple example of scaling one container running the whole application on a single host.
You'll more likely want to build your applications using lots of containers, each performing their own function like microservices.
If you build applications this way, and connect them with network connections, you can make them modular, deploy easily, and scale independently across a group of hosts.
The hosts can then scale up and down and start and stop containers as demand for your app changes or as hosts fail.



A tool that helps you do this well is Kubernetes.

Kubernetes makes it easy to orchestrate many containers on many hosts, scale them as microservices, and deploy rollouts and rollbacks.

First, I'll show you how you build and run containers.

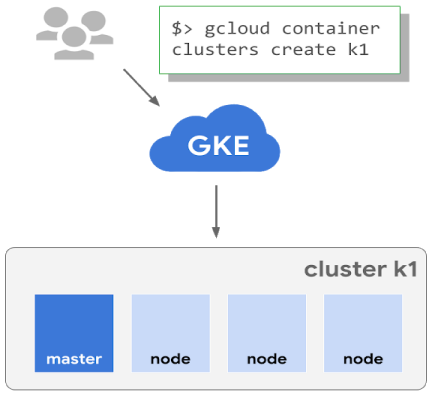
Many developers use an open-source tool called Docker that defines a format for bundling your application, its dependencies, and machine-specific settings into a container; you could also use a different tool like Google Container Builder. It's up to you.

Exam Guide: Section 3.2

3.2 Deploying and implementing Kubernetes Engine resources. Tasks include:

- **Deploying a Kubernetes Engine cluster.**
- **Deploying a container application to Kubernetes Engine using pods.**
- Configuring Kubernetes Engine application monitoring and logging.

Kubernetes manages your containers



At the highest level, Kubernetes is a set of APIs that you can use to deploy containers on a set of nodes called a cluster.
The system is divided into a set of master components that run as the control plane and a set of nodes that run containers. In Kubernetes, a node represents a computing instance, like a machine. In Google Cloud, nodes are virtual machines running in Compute Engine.

You can describe a set of applications and how they should interact with each other and Kubernetes figures how to make that happen

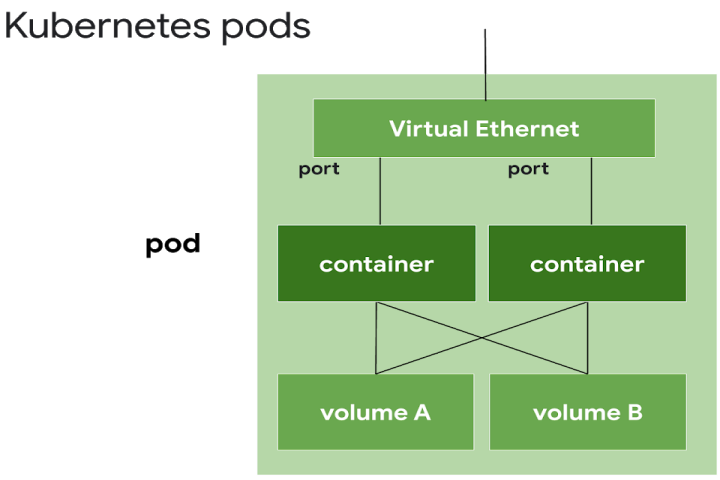
Now that you've built a container, you'll want to deploy one into a cluster.
Kubernetes can be configured with many options and add-ons, but can be time consuming to bootstrap from the ground up. Instead, you can bootstrap Kubernetes using Kubernetes Engine or (GKE).

GKE is a hosted Kubernetes by Google. GKE clusters can be customized and they support different machine types, number of nodes, and network settings.

To start up Kubernetes on a cluster in GKE, all you do is run this command:

At this point, you should have a cluster called 'k1' configured and ready to go.

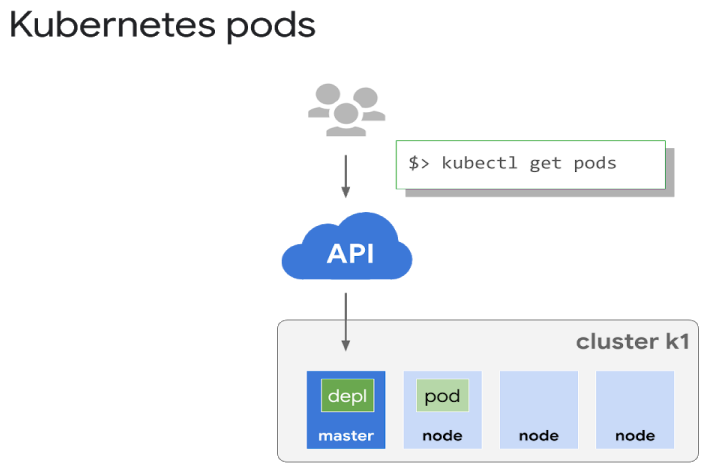
You can check its status in admin console.



Then you deploy containers on nodes using a wrapper around one or more containers called a Pod.
A Pod is the smallest unit in Kubernetes that you create or deploy. A Pod represents a running process on your cluster as either a component of your application or an entire app.

Generally, you only have one container per pod, but if you have multiple containers with a hard dependency, you can package them into a single pod and share networking and storage. The Pod provides a unique network IP and set of ports for your containers, and options that govern how containers should run.

Containers inside a Pod can communicate with one another using localhost and ports that remain fixed as they're started and stopped on different nodes.



A Deployment represents a group of replicas of the same Pod and keeps your Pods running even when nodes they run on fail. It could represent a component of an application or an entire app. In this case, it's the nginx web server.

To see the running nginx Pods, run the command:

$ kubectl get pods

1. Section 3.1 - Deploying and implementing Compute Engine resources.
2. Section 3.2 - Deploying and implementing Kubernetes Engine resources.
3. **Section 3.3 - Deploying and implementing App Engine and Cloud Functions resources**
4. Section 3.4 - Deploying and implementing data solutions

Exam Guide: Section 3.3

3.3 Deploying and implementing App Engine and Cloud Functions resources.

Tasks include:

- Deploying an application to App Engine (e.g., scaling configuration, versions, and traffic splitting).
- Deploying a Cloud Function that receives Google Cloud events (e.g., Cloud Pub/Sub events, Cloud Storage object change notification events).

Exam Guide: Section 3.3

3.3 Deploying and implementing App Engine and Cloud Functions resources.

Tasks include:

- **Deploying an application to App Engine (e.g., scaling configuration, versions, and traffic splitting).**
- Deploying a Cloud Function that receives Google Cloud events (e.g., Cloud Pub/Sub events, Cloud Storage object change notification events).

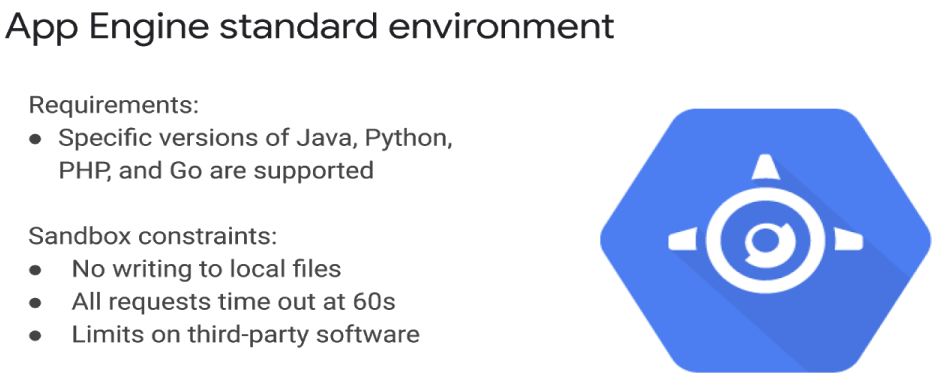
App Engine standard environment

- Easily deploy your applications
- Autoscale workloads
- Free daily quota
- Usage-based pricing
- SDKs for development, testing and deployment

Of the two App Engine environments, Standard and Flexible, Standard is the simpler. It offers a simpler deployment experience than the Flexible environment, and finer-grained autoscaling. Like the Standard environment, it also offers a free daily usage quota for the use of some services. What's distinctive about the Standard environment, though is that low-utilization applications might be able to run at no charge.

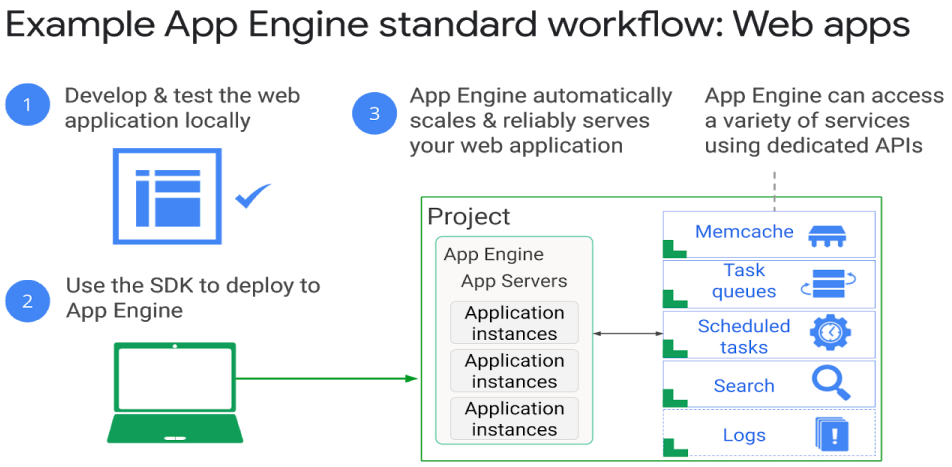
Google provides App Engine Software Development Kits in several languages, so that you can test your application locally before you upload it to the real App Engine service. The SDKs also

provide simple commands for doing the deployment.



App Engine Standard Environment provides runtimes for specific versions of Java, Python, PHP, and Go. The runtimes also include libraries that support App Engine APIs, and for many applications, the standard environment runtimes and libraries may be all you need. But if you want to code in another language, Standard Environment is not right for you, and you'll want to consider the Flexible environment

The Standard environment also enforces restrictions on your code by making it run in a so-called sandbox. That's a software construct that's independent of the hardware, operating system, or physical location of the server it runs on. The sandbox is part of why App Engine Standard environment can scale and manage your application in a very fine-grained way. Like all sandboxes, it imposes some constraints. For example, your application can't write to the local filesystem. It'll have to write to a database service instead if it needs to make data persistent. Also, all the requests your application gets, have a 60-second timeout, and you can't install arbitrary third-party software. If these constraints don't work for you, that would also be a reason to choose the Flexible environment.



In this diagram we see App Engine Standard Environment in practice. You'll develop your application and run a test version of it locally using the App Engine SDK. Then, when you're ready, you'll use the SDK to deploy it.

Each App Engine application runs in a GCP project. App Engine automatically provisions server instances and scales and load-balances them. Meanwhile, your application can make calls to a variety of services using dedicated APIs. For example, a NoSQL datastore to make data persistent; caching of that data using memcache; searching; logging; user login; and the ability to launch actions not triggered by direct user requests, like task queues and a task scheduler.

Exam Guide: Section 3.3

3.3 Deploying and implementing App Engine and Cloud Functions resources.

Tasks include:

- Deploying an application to App Engine (e.g., scaling configuration, versions, and traffic splitting).
- **Deploying a Cloud Function that receives Google Cloud events (e.g., Cloud Pub/Sub events, Cloud Storage object change notification events).**

Cloud Functions

- Create single-purpose functions that respond to events without a server or runtime
  - Event examples: New instance created, file added to Cloud Storage
- Written in Javascript, Python or Go; execute in managed Node.js environment on Google Cloud Platform

Cloud Functions is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events without the need to manage a server or a runtime environment. You can use these functions to construct applications from bite-sized business logic. You can also use Cloud Functions to connect and extend cloud services.

You are billed, to the nearest 100 milliseconds, only while your code is running.

Cloud Functions are written in Javascript, Python or Go and execute in a managed Node.js environment on Google Cloud Platform. Events from Cloud Storage and Cloud Pub/Sub can trigger Cloud Functions asynchronously, or you can use HTTP invocation for synchronous execution.

Cloud Events are things that happen in your cloud environment. These might be things like changes to data in a database, files added to a storage system, or a new virtual machine instance being created.

Events occur whether or not you choose to respond to them. Creating a response to an event is done with a trigger. A trigger is a declaration that you are interested in a certain event or set of events. You create triggers to capture events and act on them.

1. Section 3.1 - Deploying and implementing Compute Engine resources.
2. Section 3.2 - Deploying and implementing Kubernetes Engine resources.
3. Section 3.3 - Deploying and implementing App Engine and Cloud Functions resources
4. **Section 3.4 - Deploying and implementing data solutions**

Exam Guide: Section 3.4

3.4 Deploying and implementing data solutions. Tasks include:

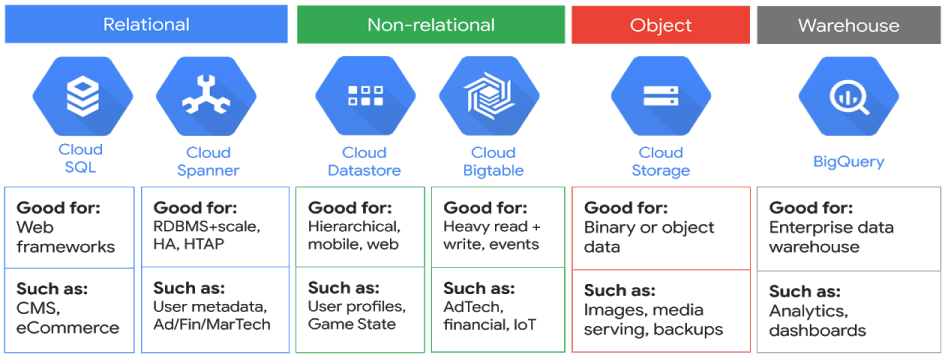
- Initializing data systems with products (e.g., Cloud SQL, Cloud Datastore, BigQuery, Cloud Spanner, Cloud Pub/Sub, Cloud Bigtable, Cloud Dataproc, Cloud Storage).
- Loading data (e.g., Command line upload, API transfer, Import / export, load data from Cloud Storage, streaming data to Cloud Pub/Sub).

Exam Guide: Section 3.4

3.4 Deploying and implementing data solutions. Tasks include:

- **Initializing data systems with products (e.g., Cloud SQL, Cloud Datastore, BigQuery, Cloud Spanner, Cloud Pub/Sub, Cloud Bigtable, Cloud Dataproc, Cloud Storage).**
- Loading data (e.g., Command line upload, API transfer, Import / export, load data from Cloud Storage, streaming data to Cloud Pub/Sub).

Comparing data storage and database options



Let's quickly review your options for data storage and databases.

Comparing storage options: technical details

	Cloud Datastore	Bigtable	Cloud Storage	Cloud SQL	Cloud Spanner	BigQuery
Type	NoSQL document	NoSQL wide column	Blobstore	Relational SQL for OLTP	Relational SQL for OLTP	Relational SQL for OLAP
Transactions	Yes	Single-row	No	Yes	Yes	No
Complex queries	No	No	No	Yes	Yes	Yes
Capacity	Terabytes+	Petabytes+	Petabytes+	Terabytes	Petabytes	Petabytes+
Unit size	1 MB/entity	~10 MB/cell ~100 MB/row	5 TB/object	Determined by DB engine	10,240 MiB/ row	10 MB/row

This table focuses on the technical differentiators of the storage services. Each row is a technical specification and each column is a service. Let me cover each service from left to right.

Consider using Cloud Datastore, if you need to store structured objects, or if you require support for transactions and SQL-like queries. This storage services provides terabytes of capacity with a maximum unit size of 1 MB per entity.

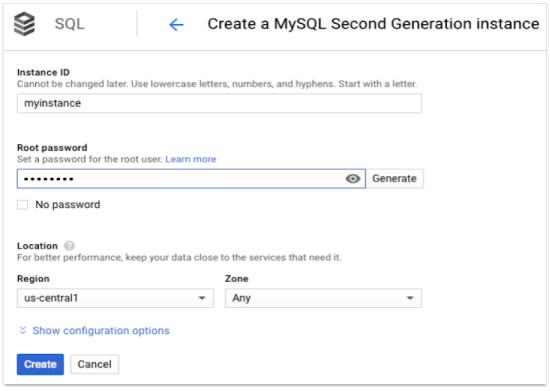
Consider using Cloud Bigtable, if you need to store a large amount of structured objects. Cloud Bigtable does not support SQL queries, nor does it support multi-row transactions. This storage service provides petabytes of capacity with a maximum unit size of 10 MB per cell and 100 MB per row.

Consider using Cloud Storage, if you need to store immutable blobs larger than 10 MB, such as large images or movies. This storage service provides petabytes of capacity with a maximum unit size of 5 TB per object.

Consider using Cloud SQL or Cloud Spanner if you need full SQL support for an online transaction processing system. Cloud SQL provides terabytes of capacity, while Cloud Spanner provides petabytes. If Cloud SQL does not fit your requirements because you need horizontal scalability, not just through read replicas, consider using Cloud Spanner.

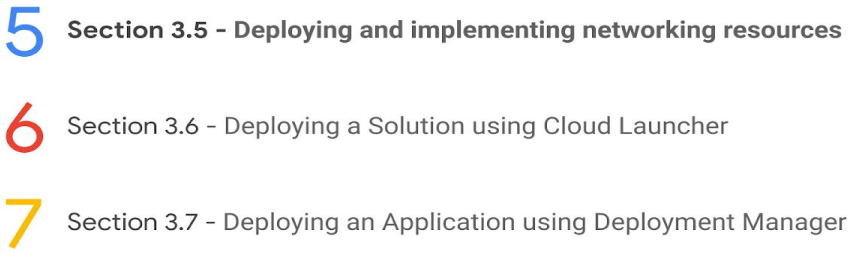
The usual reason to store data in BigQuery is to use its big data analysis and interactive querying capabilities. You would not want to use BigQuery, for example, as the backing store for an online application.

Creating a CloudSQL instance, step by step



Select or create a Google Cloud Platform project.
Make sure that billing is enabled for your Google Cloud Platform project.
Go to the Cloud SQL Instances page in the Google Cloud Platform Console.
Select your project and click Continue.
Click Create Instance.
Click MySQL.

Click Choose Second Generation.
Enter myinstance for Instance ID.
Enter a password for the root user. Use the default values for the other fields.
Click Create. You are returned to the instances list; your new instance is greyed out while it initializes and starts.

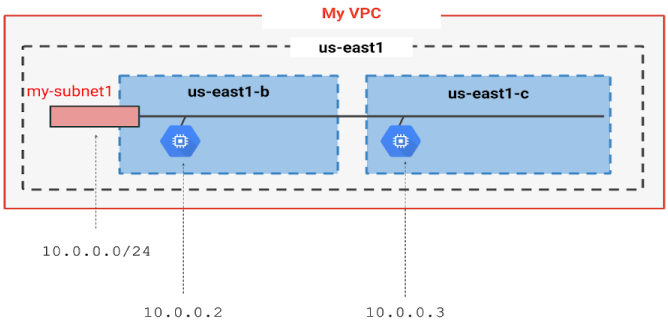


Exam Guide: Section 3.5

3.5 Deploying and implementing networking resources. Tasks include:

- Creating a VPC with subnets. (e.g., Custom-mode VPC, Shared VPC).
- Launching a Compute Engine instance with custom network configuration (e.g., Internal-only IP address, Google private access, Static external and private IP address, network tags).
- Creating ingress and egress firewall rules for a VPC (e.g., IP subnets, Tags, Service accounts).
- Creating a VPN between a Google VPC & an external network using Cloud VPN.
- Creating a load balancer to distribute application network traffic to an application (e.g., Global HTTP(S) load balancer, Global SSL Proxy load balancer, Global TCP Proxy load balancer, Regional Network load balancer, Regional Internal load balancer).

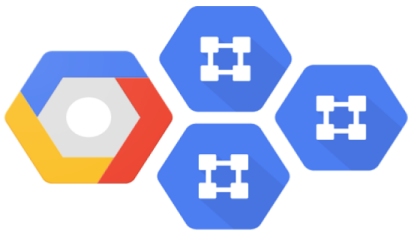
GCP VPC networks are global; subnets are regional



Here's something that surprises a lot of people who are new to GCP. The Virtual Private Cloud networks that you define have global scope. They can have subnets in any GCP region worldwide. Subnets can span the zones that make up a region. This architecture makes it easy for you to define your own network layout with global scope. You can also have resources in different zones on the same subnet.

Virtual Private Cloud (VPC) Networking

- Each VPC network is contained in a GCP project.
- You can provision Cloud Platform resources, connect them to each other, and isolate them from one another.



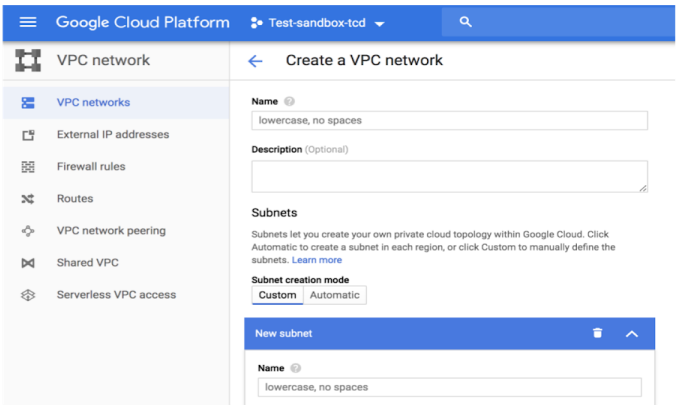
The way a lot of people get started with GCP is to define their own Virtual Private Cloud inside their first GCP project. Or they can simply choose the default VPC and get started with that. Regardless, your VPC networks connect your Google Cloud Platform resources to each other and to the internet. You can segment your networks, use firewall rules to restrict access to instances, and create static routes to forward traffic to specific destinations.

Exam Guide: Section 3.5

3.5 Deploying and implementing networking resources. Tasks include:

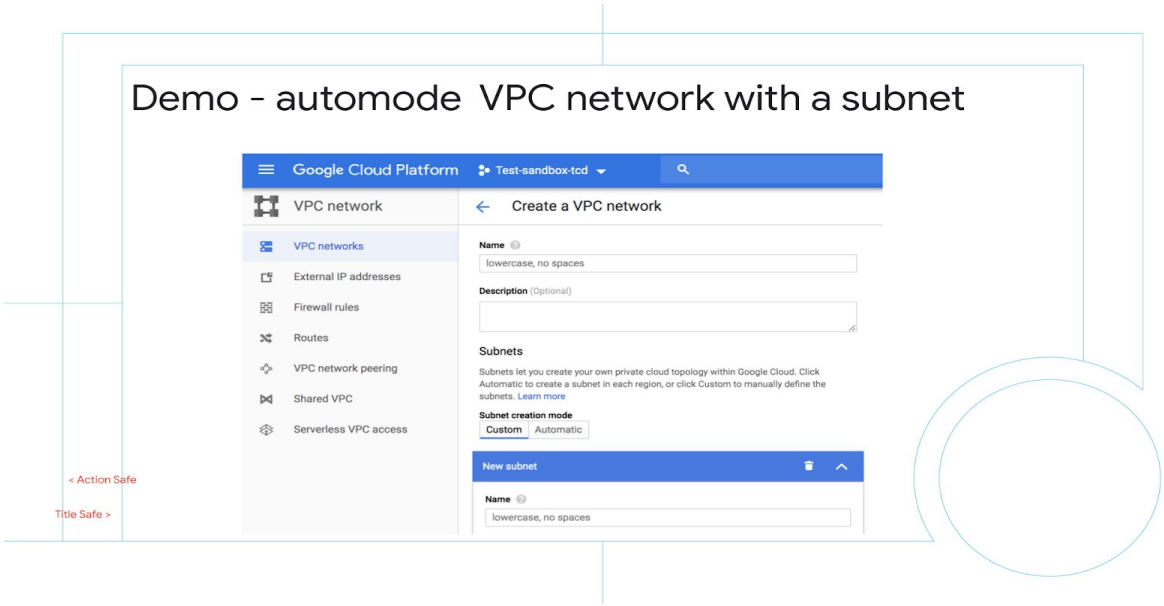
- **Creating a VPC with subnets. (e.g., Custom-mode VPC, Shared VPC).**
- Launching a Compute Engine instance with custom network configuration (e.g., Internal-only IP address, Google private access, Static external and private IP address, network tags).
- Creating ingress and egress firewall rules for a VPC (e.g., IP subnets, Tags, Service accounts).
- Creating a VPN between a Google VPC & an external network using Cloud VPN.
- Creating a load balancer to distribute application network traffic to an application (e.g., Global HTTP(S) load balancer, Global SSL Proxy load balancer, Global TCP Proxy load balancer, Regional Network load balancer, Regional Internal load balancer).

Creating an automode VPC network with a subnet



You can choose to create an auto mode or custom mode VPC network. Each new network that you create must have a unique name within the same project.

Auto mode networks create one subnet in each GCP region automatically when you create the network. As new regions become available, new subnets in those regions are automatically added to the auto mode network. IP ranges for the automatically created subnets come from a predetermined set of ranges. All auto mode networks use the same set of IP ranges.

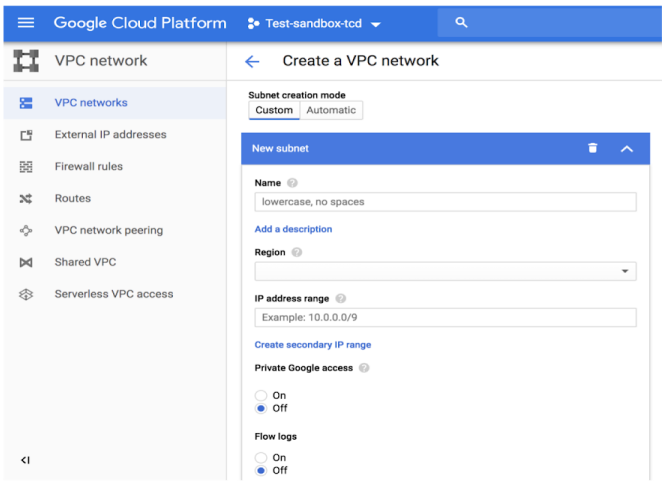


https://cloud.google.com/vpc/docs/using-vpc

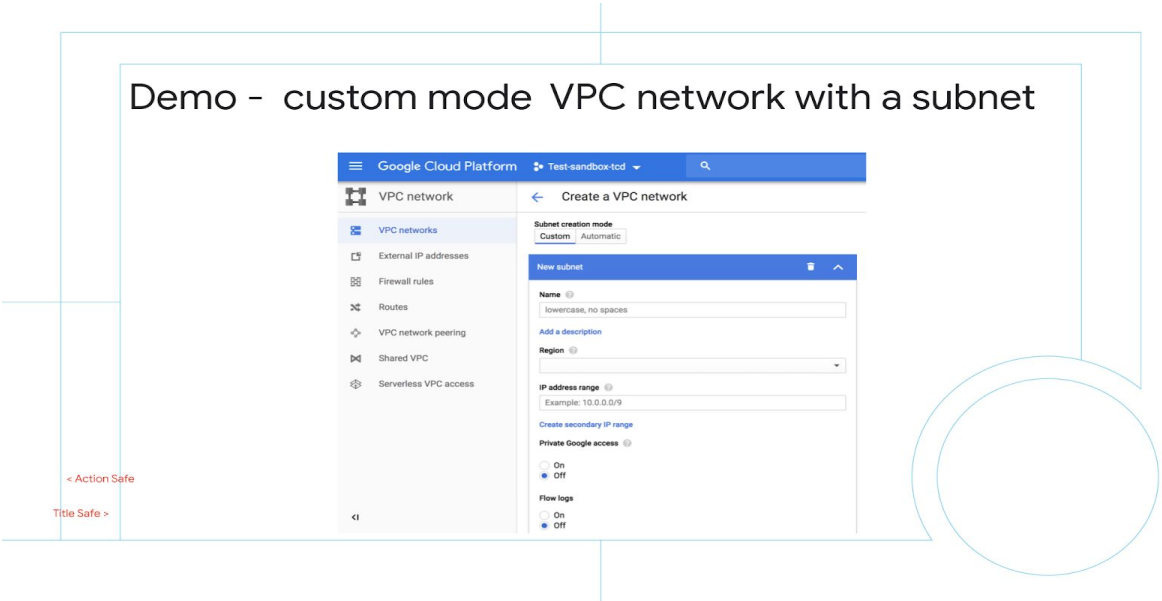
To set up an automode VPC network:

Go to the VPC networks page in the Google Cloud Platform Console. GO TO THE VPC NETWORKS PAGE
Click Create VPC network.
Enter a Name for the network.
Choose Automatic for the Subnet creation mode.
In the Firewall rules section, select one or more predefined firewall rules that address common use cases for connectivity to VMs. Choose the Dynamic routing mode for the VPC network. For more information, see dynamic routing mode. You can change the dynamic routing mode later.
Click Create.

Creating a custom mode VPC network with a subnet



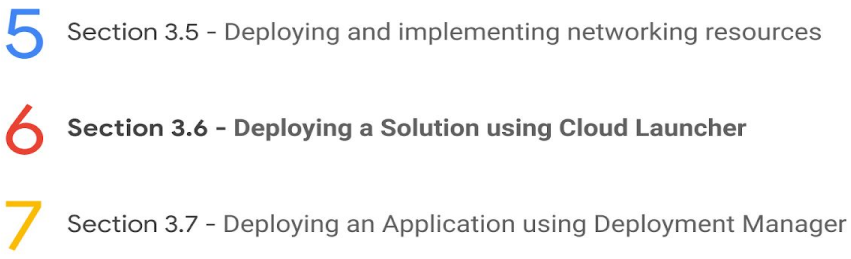
You can control the subnets created within a custom mode VPC network.
With a custom mode vpc, you create subnets when you create the network, or you can add subnets later.



You can also control the subnets created within a custom mode VPC network.
With a custom mode vpc, you create subnets when you create the network, or you can add subnets later.

https://cloud.google.com/vpc/docs/using-vpc

To create a custom mode VPC network with subnet:
Go to the VPC networks page in the Google Cloud Platform Console. GO TO THE VPC NETWORKS PAGE
Click Create VPC network.
Enter a Name for the network.
Choose Custom for the Subnet creation mode.
In the New subnet section, specify the following configuration parameters for a subnet:
Provide a Name for the subnet.
Select a Region.
Enter an IP address range. This is the primary IP range for the subnet.
To define a secondary range for the subnet, click Create secondary IP range.
Private Google access: Choose whether to enable Private Google Access for the subnet when you create it or later by editing it.
Flow logs: Choose whether to enable VPC flow logs for the subnet when you create it or later by editing it.
Click Done.
To add more subnets, click Add subnet and repeat the previous steps. You can also add more subnets to the network after you have created the network.
Choose the Dynamic routing mode for the VPC network. For more information, see dynamic routing mode. You can change the dynamic routing mode later.
Click Create.



Exam Guide: Section 3.6

3.6 Deploying a Solution using Cloud Launcher. Tasks include:

- Browsing Cloud Launcher catalog and viewing solution details.
- Deploying a Cloud Launcher marketplace solution.

Exam Guide: Section 3.6

3.6 Deploying a Solution using Cloud Launcher. Tasks include:

- **Browsing Cloud Launcher catalog and viewing solution details.**
- **Deploying a Cloud Launcher marketplace solution.**

Cloud Marketplace gives quick access to solutions

- A solution marketplace containing pre-packaged, ready-to-deploy solutions
  - Some offered by Google
  - Others by third-party vendors
  - You pay for the underlying GCP resource usage.
    - Some solutions also assess third-party license fees.

If you want a quick way to get started with GCP, with minimal effort this is what Cloud Launcher (now Cloud Marketplace) provides. It's a tool for quickly deploying functional software packages on Google Cloud Platform. There's no need to manually configure the software, virtual machine instances, storage, or network settings, although you can modify many of them before you launch if you like. Most software packages in Cloud Launcher are at no additional charge beyond the normal usage fees for GCP resources. Some Cloud Launcher images charge usage fees, particularly those published by third parties, with commercially licensed software. But they all show you estimates of their monthly charges before you launch them. Do be aware that these estimates are just that--estimates--and in particular they don't attempt to estimate networking costs, since those will vary based on how you use the applications. A second note of caution: GCP updates the base images for these software packages to fix critical issues and vulnerabilities, but it doesn't update software after it's been deployed. Fortunately, you'll have access to the deployed systems so you can maintain them.

Cloud Launcher (GCP Marketplace) catalog

GCP Marketplace offers ready-to-go development stacks, solutions, and services to accelerate development. So you spend less time installing and more time developing.

- Deploy production-grade solutions in a few clicks
- Single bill for all your GCP and 3rd party services
- Manage solutions using Deployment Manager
- Notifications when a security update is available
- Direct access to partner support√

GCP Marketplace offers ready-to-go development stacks, solutions, and services to accelerate development. So you spend less time installing and more time developing.
Deploy production-grade solutions in a few clicks
Single bill for all your GCP and 3rd party services
Manage solutions using Deployment Manager
Notifications when a security update is available
Direct access to partner support

5 Section 3.5 - Deploying and implementing networking resources

6 Section 3.6 - Deploying a Solution using Cloud Launcher

7 **Section 3.7 - Deploying an Application using Deployment Manager**

Exam Guide: Section 3.7

3.7 Deploying an Application using Deployment Manager. Tasks include:

- Developing Deployment Manager templates to automate deployment of an application.
- Launching a Deployment Manager template to provision GCP resources and configure an application automatically.

Exam Guide: Section 3.7

3.7 Deploying an Application using Deployment Manager. Tasks include:

- Developing Deployment Manager templates to automate deployment of an application.
- **Launching a Deployment Manager template to provision GCP resources and configure an application automatically.**

Deployment Manager automates the creation and management of your GCP resources

- Infrastructure management service
- Create a .yaml template describing your environment and use Deployment Manager to create resources
- Provides repeatable deployments

Deployment Manager is an infrastructure management service that automates the creation and management of your Google Cloud Platform resources for you.

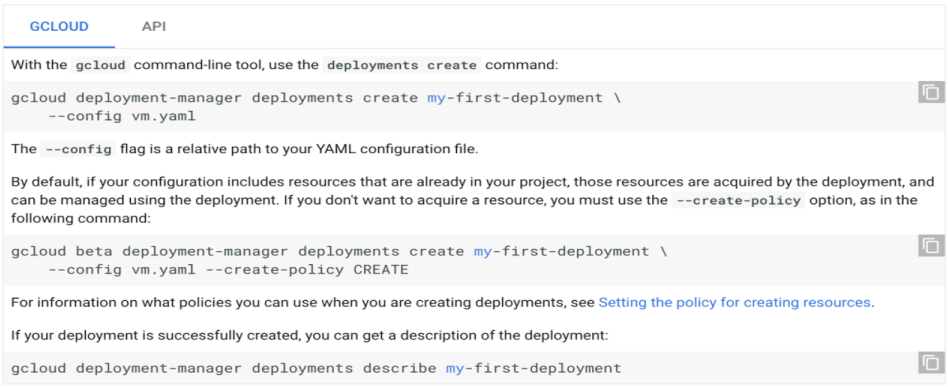
Setting up your environment in GCP can entail many steps: setting up compute, network, and storage resources and keeping track of their configurations. You can do it all by hand if you want to, taking an imperative approach. But it is more efficient to use a template. That means a specification of what the environment should look like, declarative rather than imperative.

GCP provides Deployment Manager to let do just that. It's an infrastructure management service that automates the creation and management of your Google Cloud Platform resources for you.

To use Deployment Manager, you create a template file, using either the YAML markup language or Python, that describes what you want the components of your environment to look like. Then you give the template to Deployment Manager, which figures out and does the actions needed to create the environment your template describes. If you need to change your environment, edit your template, and then tell Deployment Manager to update the environment to match the change.

Here's a tip: you can store and version-control your Deployment Manager templates in Cloud Source Repositories.

Deploying an application with Deployment Manager



A deployment is an instantiation of a set of resources that are defined in a configuration. You provide a valid configuration in the request to create the deployment. A deployment can contain a number of resources, across a variety of Google Cloud Platform services. When you create a deployment, Deployment Manager creates all of the described resources in the respective Google Cloud Platform APIs.

When you create a deployment, you are creating a Deployment resource that contains a collection of resources. Each resource is explicitly defined in a configuration that you provide in your request.

When you create a new deployment, if a resource that you want to create already exists in your project, it is acquired by the deployment. In such cases, Deployment Manager does not create a new resource.

Deployment Manager is an infrastructure deployment service that automates the creation and management of your Google Cloud Platform resources for you. You can create flexible templates that deploy a variety of Cloud Platform services, such as Google Cloud Storage, Google Compute Engine, and Google Cloud SQL.

In this lab, you will launch a service using an infrastructure orchestration tool called Deployment Manager and monitor the service using Stackdriver. In Stackdriver, you will set up basic black box monitoring with a Stackdriver dashboard and establish an Uptime Check (alert notification) to trigger incident response.

This lab is part of the Qwiklabs Cloud Architecture Quest.

Suggested study resources for this section

Compute Engine: https://cloud.google.com/compute/docs/

Cloud Source Repositories https://cloud.google.com/source-repositories/docs/

Deployment Manager https://cloud.google.com/deployment-manager/docs/

Instance Groups: https://cloud.google.com/compute/docs/instance-groups/

Autoscaling: https://cloud.google.com/compute/docs/autoscaler/

Instance Templates: https://cloud.google.com/compute/docs/instance-templates/

Create VMs from instance template:
https://cloud.google.com/compute/docs/instances/create-vm-from-instance-template

Creating groups of managed instances with templates:
https://cloud.google.com/compute/docs/instance-groups/creating-groups-of-managed-instances

Using VPC networks: https://cloud.google.com/vpc/docs/using-vpc

Deployment manager fundamentals: https://cloud.google.com/deployment-manager/docs/fundamentals