INTRODUCING GOOGLE CLOUD

CLOUD COMPUTING OVERVIEW

[00:00](javascript:;) Let’s start at the beginning with an overview of cloud computing. The cloud is a hot topic these days, but what exactly is it? The US National Institute of Standards and Technology created the term

[00:13](javascript:;) cloud computing, although there is nothing US-specific about it. Cloud computing is a way of using information technology (IT) that has these five equally important traits First, customers get computing resources that are on-demand and self-service.

[00:31](javascript:;) Through a web interface, users get the processing power, storage, and network they need with no need for human intervention. Second, customers get access to those resources over the internet, from anywhere they have

[00:44](javascript:;) a connection. Third, the cloud provider has a big pool of those resources and allocates them to users out of that pool. That allows the provider to buy in bulk and pass the savings on to the customers.

[00:57](javascript:;) Customers don't have to know or care about the exact physical location of those resources. Fourth, the resources are elastic–which means they’re flexible, so customers can be. If they need more resources they can get more, and quickly.

[01:15](javascript:;) If they need less, they can scale back. And finally, customers pay only for what they use, or reserve as they go. If they stop using resources, they stop paying.

[01:26](javascript:;) That's it. That's the definition of cloud. But why is the cloud model so compelling nowadays? To understand why, we need to look at some history. The trend towards cloud computing started with

[01:39](javascript:;) a first wave known as colocation Colocation gave users the financial efficiency of renting physical space, instead of investing in data center real estate. Virtualized data centers of today, which is the second wave, share similarities

[01:54](javascript:;) with the private data centers and colocation facilities of decades past. The components of virtualized data centers match the physical building blocks of hosted computing—servers, CPUs, disks, load balancers, and so on—but now they’re virtual devices.

[02:13](javascript:;) With virtualization, enterprises still maintain the infrastructure; but it also remains a user-controlled and user-configured environment. Several years ago, Google realized that its business couldn’t move fast enough within the confines of the virtualization model.

[02:30](javascript:;) So Google switched to a container-based architecture—a fully automated, elastic third-wave cloud that consists of a combination of automated services and scalable data. Services automatically provision and configure the infrastructure used to run applications.

[02:47](javascript:;) Today, Google Cloud makes this third-wave cloud available to Google customers. Google believes that, in the future, every company—regardless of size or industry— will differentiate itself from its competitors through technology.

[03:03](javascript:;) Increasingly, that technology will be in the form of software. Great software is based on high-quality data. This means that every company is, or will eventually become, a data company.

laaS and PaaS

The move to virtualized data centers introduced customers to two new types of offerings: infrastructure as a service, commonly referred to as IaaS, and platform as a service, or PaaS.

[00:13](javascript:;) IaaS offerings provide raw compute, storage, and network capabilities, organized virtually into resources that are similar to physical data centers. PaaS offerings, in contrast, bind code to libraries that provide access to the infrastructure application needs.

[00:31](javascript:;) This allows more resources to be focused on application logic. In the IaaS model, customers pay for the resources they allocate ahead of time; in the PaaS model, customers pay for the resources they actually use.

[00:46](javascript:;) As cloud computing has evolved, the momentum has shifted toward managed infrastructure and managed services. Leveraging managed resources and services allows companies to concentrate more on their business goals and spend less time and money on creating and maintaining their

[01:02](javascript:;) technical infrastructure. It allows companies to deliver products and services to their customers more quickly and reliably. Serverless is yet another step in the evolution of cloud computing. It allows developers to concentrate on their code, rather than on server configuration,

[01:21](javascript:;) by eliminating the need for any infrastructure management. Serverless technologies offered by Google include Cloud Functions, which manages event-driven code as a pay-as-you-go service, and Cloud Run, which allows customers to deploy

[01:36](javascript:;) their containerized microservices based application in a fully-managed environment. While it’s outside the scope of this course, you might have heard about software as a service, SaaS, and wondered what it is and how it fits into the Cloud ecosphere.

[01:52](javascript:;) Software as a Service applications are not installed on your local computer. Instead, they run in the cloud as a service and are consumed directly over the internet by end users

[02:03](javascript:;) Popular Google applications such as Gmail, Docs, and Drive, that are a part of Google Workspace, are all examples of SaaS.

THE GOOGLE CLOUD NETWORK

Google Cloud runs on Google’s own global network. It’s the largest network of its kind, and Google has invested billions of dollars over many years to build it. This network is designed to give customers the highest possible throughput and lowest

[00:15](javascript:;) possible latencies for their applications by leveraging more than 100 content caching nodes worldwide. These are locations where high demand content is cached for quicker access, allowing applications to respond to user requests from the location that will provide the quickest response time.

[00:34](javascript:;) Google Cloud’s infrastructure is based in five major geographic locations: North America, South America, Europe, Asia, and Australia. Having multiple locations is important because choosing where to locate applications affects

[00:52](javascript:;) qualities like availability, durability, and latency, the latter of which measures the time a packet of information takes to travel from its source to its destination. Each of these locations is divided into several different regions and zones.

[01:09](javascript:;) Regions represent independent geographic areas and are composed of zones. For example, London, or europe-west2, is a region that currently comprises three different zones. A zone is an area where Google Cloud resources are deployed.

[01:27](javascript:;) For example, if you launch a virtual machine using Compute Engine it will run in the zone that you specify to ensure resource redundancy. You can also run resources in different regions.

[01:40](javascript:;) This is useful for bringing applications closer to users around the world, and also for protection in case there are issues with an entire region, such as a natural disaster.

[01:50](javascript:;) Some of Google Cloud’s services support placing resources in what we call a multi-region. For example, Cloud Spanner multi-region configurations allow you to replicate the database's data not just in multiple zones, but in multiple zones across multiple regions, as defined

[02:07](javascript:;) by the instance configuration. These additional replicas enable you to read data with low latency from multiple locations close to or within the regions in the configuration, like The Netherlands, and Belgium.

[02:21](javascript:;) Google Cloud currently supports 103 zones in 34 regions, although this number is increasing all the time. You can find the most up-to-date numbers at cloud.google.com/about/locations.

ENVIRONMENTAL IMPACT

The virtual world, which includes Google Cloud’s network, is built on physical infrastructure, and all those racks of humming servers use huge amounts of energy. Altogether, existing data centers use roughly 2% of the world’s electricity.

[00:16](javascript:;) With this in mind, Google works to make their data centers run as efficiently as possible. Just like our customers, Google is trying to do the right things for the planet.

[00:26](javascript:;) We understand that Google Cloud customers have environmental goals of their own, and running their workloads on Google Cloud can be a part of meeting those goals. Therefore, it’s useful to note that

[00:37](javascript:;) Google's data centers were the first to achieve ISO 14001 certification, which is a standard that maps out a framework for an organization to enhance its environmental performance through improving resource efficiency and reducing waste.

[00:53](javascript:;) As an example of how this is being done, here’s Google’s data center in Hamina, Finland. This facility is one of the most advanced and efficient data centers in the Google fleet.

[01:03](javascript:;) Its cooling system, which uses sea water from the Bay of Finland, reduces energy use and is the first of its kind anywhere in the world. In our founding decade, Google became the first major company to be carbon neutral.

[01:18](javascript:;) In our second decade, we were the first company to achieve 100% renewable energy. By 2030, we aim to be the first major company to operate completely carbon free.

SECURITY

Nine of Google’s services have more than one billion users each, and so you can be assured that security is always on the minds of Google's employees. Design for security is prevalent throughout the infrastructure that Google Cloud and Google

[00:14](javascript:;) services run on. Let's talk about a few ways Google works to keep customers' data safe. The security infrastructure can be explained in progressive layers, starting from the physical security of our data centers, continuing on to how the hardware and software that underlie

[00:29](javascript:;) the infrastructure are secured, and finally, describing the technical constraints and processes in place to support operational security. We begin with the Hardware infrastructure layer which comprises three key security features:

[00:46](javascript:;) The first is hardware design and provenance. Both the server boards and the networking equipment in Google data centers are custom-designed by Google. Google also designs custom chips, including a hardware security chip that's currently

[01:01](javascript:;) being deployed on both servers and peripherals. The next feature is a secure boot stack. Google server machines use a variety of technologies to ensure that they are booting the correct

[01:14](javascript:;) software stack, such as cryptographic signatures over the BIOS, bootloader, kernel, and base operating system image. This layer's final feature is premises security. Google designs and builds its own data centers, which incorporate multiple layers of physical

[01:31](javascript:;) security protections. Access to these data centers is limited to only a very small number of Google employees. Google additionally hosts some servers in third-party data centers, where we ensure

[01:44](javascript:;) that there are Google-controlled physical security measures on top of the security layers provided by the data center operator. Next is the Service deployment layer, where the key feature is encryption of inter-service

[01:55](javascript:;) communication. Google’s infrastructure provides cryptographic privacy and integrity for remote procedure call (“RPC”) data on the network. Google’s services communicate with each other using RPC calls. The infrastructure automatically encrypts all infrastructure RPC traffic that goes between

[02:14](javascript:;) data centers. Google has started to deploy hardware cryptographic accelerators that will allow it to extend this default encryption to all infrastructure RPC traffic inside Google data centers. Then we have the User identity layer.

[02:31](javascript:;) Google’s central identity service, which usually manifests to end users as the Google login page, goes beyond asking for a simple username and password. The service also intelligently challenges users for additional information based on

[02:45](javascript:;) risk factors such as whether they have logged in from the same device or a similar location in the past. Users can also employ secondary factors when signing in, including devices based on the

[02:56](javascript:;) Universal 2nd Factor (U2F) open standard. On the Storage services layer we find the encryption at rest security feature. Most applications at Google access physical storage (in other words, “file storage”)

[03:11](javascript:;) indirectly via storage services, and encryption using centrally managed keys is applied at the layer of these storage services. Google also enables hardware encryption support in hard drives and SSDs.

[03:25](javascript:;) The next layer up is the Internet communication layer, and this comprises two key security features. Google services that are being made available on the internet, register themselves with an infrastructure service called the Google Front End, which ensures that all TLS connections

[03:42](javascript:;) are ended using a public-private key pair and an X.509 certificate from a Certified Authority (CA), as well as following best practices such as supporting perfect forward secrecy. The GFE additionally applies protections against Denial of Service attacks.

[04:00](javascript:;) Also provided is Denial of Service (“DoS”) protection. The sheer scale of its infrastructure enables Google to simply absorb many DoS attacks. Google also has multi-tier, multi-layer DoS protections that further reduce the risk of

[04:14](javascript:;) any DoS impact on a service running behind a GFE. The final layer is Google's Operational security layer which provides four key features. First is intrusion detection. Rules and machine intelligence give Google’s operational security teams warnings of possible

[04:34](javascript:;) incidents. Google conducts Red Team exercises to measure and improve the effectiveness of its detection and response mechanisms. Next is reducing insider risk. Google aggressively limits and actively monitors the activities of employees who have been

[04:50](javascript:;) granted administrative access to the infrastructure. Then there’s employee U2F use. To guard against phishing attacks against Google employees, employee accounts require use of U2F-compatible Security Keys. Finally, there are stringent software development practices.

[05:10](javascript:;) Google employs central source control and requires two-party review of new code. Google also provides its developers libraries that prevent them from introducing certain classes of security bugs. Additionally, Google runs a Vulnerability Rewards Program where we pay anyone who is

[05:26](javascript:;) able to discover and inform us of bugs in our infrastructure or applications. You can learn more about Google’s technical-infrastructure security at cloud.google.com/security/security-design.

OPEN SOURCE ECOSYSTEM

Some organizations are afraid to bring their workloads to the cloud because they're afraid they'll get locked into a particular vendor. However, if, for whatever reason, a customer decides that Google

[00:10](javascript:;) is no longer the best provider for their needs, we provide them with the ability to run their applications elsewhere. Google publishes key elements of technology using open source licenses to create ecosystems

[00:22](javascript:;) that provide customers with options other than Google. For example, TensorFlow, an open source software library for machine learning developed inside Google, is at the heart of a strong open source ecosystem.

[00:35](javascript:;) Google provides interoperability at multiple layers of the stack. Kubernetes and Google Kubernetes Engine give customers the ability to mix and match microservices running across different clouds, while Google Cloud’s operations suite lets

[00:50](javascript:;) customers monitor workloads across multiple cloud providers.

PRICING AND BILLING

To round off this section of the course, let’s take a brief look at Google Cloud’s pricing structure. Google was the first major cloud provider to deliver per-second billing for its infrastructure-as-a-service

[00:11](javascript:;) compute offering, Compute Engine. In addition, per-second billing is now also offered for users of Google Kubernetes Engine (our container infrastructure as a service), Dataproc (which is the equivalent of the big data system Hadoop, but operating as a service),

[00:27](javascript:;) and App Engine flexible environment VMs (a platform as a service). We’ll explore these products and services later in the course. Compute Engine offers automatically applied sustained-use discounts, which are automatic

[00:42](javascript:;) discounts that you get for running a virtual machine instance for a significant portion of the billing month. Specifically, when you run an instance for more than 25% of a month, Compute Engine automatically

[00:54](javascript:;) gives you a discount for every incremental minute you use for that instance. Custom virtual machine types allow Compute Engine virtual machines to be fine-tuned with optimal amounts of vCPU and memory for their applications so that

[01:08](javascript:;) you can tailor your pricing for your workloads. Our online pricing calculator can help estimate your costs. Visit cloud.google.com/products/calculator to try it out. Now, you’re probably thinking, “How can I make sure I don’t accidentally run up

[01:27](javascript:;) a big Google Cloud bill?” We provide a few tools to help. You can define budgets at the billing account level or at the project level. A budget can be a fixed limit, or it can be tied to another metric; for example, a percentage

[01:43](javascript:;) of the previous month’s spend. To be notified when costs approach your budget limit, you can create an alert. For example, with a budget limit of $20,000 and an alert set at 90%, you’ll receive

[01:58](javascript:;) a notification alert when your expenses reach $18,000. Alerts are generally set at 50%, 90% and 100%, but can also be customized. Reports is a visual tool in the Google Cloud Console that allows you to monitor expenditure

[02:15](javascript:;) based on a project or services. Finally, Google Cloud also implements quotas, which are designed to prevent the over-consumption of resources because of an error or a malicious attack, protecting both account owners and

[02:29](javascript:;) the Google Cloud community as a whole. There are two types of quotas: rate quotas and allocation quotas. Both are applied at the project level. Rate quotas reset after a specific time.

[02:44](javascript:;) For example, by default, the GKE service implements a quota of 1,000 calls to its API from each Google Cloud project every 100 seconds. After that 100 seconds, the limit is reset.

[02:59](javascript:;) Allocation quotas govern the number of resources you can have in your projects. For example, by default, each Google Cloud project has a quota allowing it no more than five Virtual Private Cloud networks.

[03:11](javascript:;) Although projects all start with the same quotas, you can change some of them by requesting an increase from Google Cloud Support.

RESOURCES AND ACCESS IN THE CLOUD

GOOGLE CLOUD RESOURCE HIERARCHY

In this section of the course we’ll look at the functional structure of Google Cloud.

[00:05](javascript:;) Google Cloud’s resource hierarchy contains four levels, and starting from the bottom up they are: resources, projects, folders, and an organization node.

[00:17](javascript:;) At the first level are resources.

[00:20](javascript:;) These represent virtual machines, Cloud Storage buckets, tables in BigQuery, or anything else in Google Cloud.

[00:29](javascript:;) Resources are organized into projects, which sit on the second level.

[00:34](javascript:;) Projects can be organized into folders, or even subfolders.

[00:37](javascript:;) These sit at the third level.

[00:40](javascript:;) And then at the top level is an organization node, which encompasses all the projects, folders, and resources in your organization.

[00:47](javascript:;) It’s important to understand this resource hierarchy because it directly relates to how policies are managed and applied when you use Google Cloud.

[00:57](javascript:;) Policies can be defined at the project, folder, and organization node levels.

[01:02](javascript:;) Some Google Cloud services allow policies to be applied to individual resources, too.

[01:09](javascript:;) Policies are also inherited downward.

[01:11](javascript:;) This means that if you apply a policy to a folder, it will also apply to all of the projects within that folder.

[01:17](javascript:;) Let’s take a look at the second level of the resource hierarchy, projects, in a little more detail.

[01:24](javascript:;) Projects are the basis for enabling and using Google Cloud services, like managing APIs, enabling billing, adding and removing collaborators, and enabling other Google services.

[01:36](javascript:;) Each project is a separate entity under the organization node, and each resource belongs to exactly one project.

[01:44](javascript:;) Projects can have different owners and users because they’re billed and managed separately.

[01:49](javascript:;) Each Google Cloud project has three identifying attributes: a project ID, a project name, and a project number.

[01:57](javascript:;) The project ID is a globally unique identifier assigned by Google that can’t be changed after creation.

[02:04](javascript:;) They’re what we refer to as being immutable.

[02:08](javascript:;) Project IDs are used in different contexts to inform Google Cloud of the exact project to work with.

[02:15](javascript:;) Project names, however, are user-created.

[02:17](javascript:;) They don’t have to be unique and they can be changed at any time, so they are not immutable.

[02:23](javascript:;) Google Cloud also assigns each project a unique project number.

[02:27](javascript:;) It’s helpful to know that these Google-generated numbers exist, but we won’t explore them much in this course.

[02:32](javascript:;) They’re mainly used internally by Google Cloud to keep track of resources.

[02:38](javascript:;) Google Cloud’s Resource Manager tool is designed to programmatically help you manage projects.

[02:43](javascript:;) It’s an API that can gather a list of all the projects associated with an account, create new projects, update existing projects, and delete projects.

[02:53](javascript:;) It can even recover projects that were previously deleted,and can be accessed through the RPC API and the REST API.

[03:02](javascript:;) The third level of the Google Cloud resource hierarchy is folders.

[03:06](javascript:;) Folders let you assign policies to resources at a level of granularity you choose.

[03:11](javascript:;) The resources in a folder inherit policies and permissions assigned to that folder.

[03:16](javascript:;) A folder can contain projects, other folders, or a combination of both.

[03:22](javascript:;) You can use folders to group projects under an organization in a hierarchy.

[03:26](javascript:;) For example, your organization might contain multiple departments, each with its own set Google Cloud resources.

[03:33](javascript:;) Folders allow you to group these resources on a per-department basis.

[03:38](javascript:;) Folders also give teams the ability to delegate administrative rights so that they can work independently.

[03:44](javascript:;) As previously mentioned, the resources in a folder inherit policies and permissions from that folder.

[03:51](javascript:;) For example, if you have two different projects that are administered by the same team, you can put policies into a common folder so they have the same permissions.

[04:01](javascript:;) Doing it the other way--putting duplicate copies of those policies on both projects–could be tedious and error-prone.

[04:08](javascript:;) if you needed to change permissions on both resources, you would now have to do that in two places instead of just one.

[04:15](javascript:;) To use folders, you must have an organization node, which is the very topmost resource in the Google Cloud hierarchy.

[04:21](javascript:;) Everything else attached to that account goes under this node, which includes folders, projects, and other resources.

[04:30](javascript:;) There are some special roles associated with this top-level organization node.

[04:34](javascript:;) For example, you can designate an organization policy administrator so that only people with privilege can change policies.

[04:43](javascript:;) You can also assign a project creator role, which is a great way to control who can create projects and, therefore, who can spend money.

[04:50](javascript:;) How a new organization node is created depends on whether your company is also a Google Workspace customer.

[04:56](javascript:;) If you have a Workspace domain, Google Cloud projects will automatically belong to your organization node.

[05:03](javascript:;) Otherwise, you can use Cloud Identity, Google’s identity, access, application, and endpoint management platform, to generate one.

[05:11](javascript:;) Once created, a new organization node will let anyone in the domain create projects and billing accounts, just as they could before.

[05:20](javascript:;) folders underneath it and put projects into it.

[05:23](javascript:;) Both folders and projects are considered to be “children” of the organization node.

IDENTITY AND ACCESS MANAGEMENT (IAM)

When an organization node contains lots of folders, projects, and resources, a workforce might need to restrict who has access to what. To help with this task, administrators can use Identity and Access Management, or IAM.

[00:14](javascript:;) With IAM, administrators can apply policies that define who can do what and on which resources. The “who” part of an IAM policy can be a Google account, a Google group,

[00:26](javascript:;) a service account, or a Cloud Identity domain. A “who” is also called a “principal.” Each principle has its own identifier, usually an email address. The “can do what” part of an IAM policy is defined by a role.

[00:41](javascript:;) An IAM role is a collection of permissions. When you grant a role to a principal, you grant all the permissions that the role contains. For example, to manage virtual machine instances in a project, you must be able to

[00:54](javascript:;) create, delete, start, stop and change virtual machines. So these permissions are grouped into a role to make them easier to understand and easier to manage. When a principal is given a role on a specific element of the resource hierarchy, the resulting policy

[01:13](javascript:;) applies to both the chosen element and all the elements below it in the hierarchy. You can define deny rules that prevent certain principals from using certain permissions, regardless of the roles they're granted.

[01:25](javascript:;) This is because IAM always checks relevant deny policies before checking relevant allow policies. Deny policies, like allow policies, are inherited through the resource hierarchy. There are three kinds of roles in IAM: basic, predefined, and custom.

[01:44](javascript:;) The first role type is basic. Basic roles are quite broad in scope. When applied to a Google Cloud project, they affect all resources in that project. Basic roles include owner, editor, viewer, and billing administrator.

[02:00](javascript:;) Let’s look at these basic roles in a bit more detail. Project viewers can access resources but can’t make changes. Project editors can access and make changes to a resource.

[02:12](javascript:;) And project owners can also access and make changes to a resource. In addition, project owners can manage the associated roles and permissions and set up billing. Often companies want someone to control the billing for a project but not be able to change

[02:28](javascript:;) the resources in the project. This is possible through a billing administrator role. A word of caution: If several people are working together on a project that contains sensitive data, basic roles are probably too broad.

[02:43](javascript:;) Fortunately, IAM provides other ways to assign permissions that are more specifically tailored to meet the needs of typical job roles. This brings us to the second type of role, predefined roles.

[02:56](javascript:;) Specific Google Cloud services offer sets of predefined roles, and they even define where those roles can be applied. Let’s look at Compute Engine, for example, a Google Cloud product that offers virtual

[03:08](javascript:;) machines as a service. With Compute Engine, you can apply specific predefined roles—such as “instanceAdmin”—to Compute Engine resources in a given project, a given folder, or an entire organization. This then allows whoever has these roles

[03:24](javascript:;) to perform a specific set of predefined actions. But what if you need to assign a role that has even more specific permissions? That’s when you’d use a custom role.

[03:35](javascript:;) Many companies use a “least-privilege” model in which each person in your organization is given the minimal amount of privilege needed to do their job. So, for example, maybe you want to define an “instanceOperator” role to allow

[03:48](javascript:;) some users to stop and start Compute Engine virtual machines, but not reconfigure them. Custom roles will allow you to define those exact permissions. Before you start creating custom roles, please note two important details.

[04:02](javascript:;) First, you’ll need to manage the permissions that define the custom role you’ve created. Because of this, some organizations decide they’d rather use the predefined roles. And second, custom roles can only be applied to either the project level or organization

[04:17](javascript:;) level. They can’t be applied to the folder level.

SERVICE ACCOUNTS

What if you want to give permissions to a Compute Engine virtual machine, rather than to a person? Well, that’s what service accounts are for. Let’s say you have an application running in a virtual machine that needs

[00:11](javascript:;) to store data in Cloud Storage, but you don’t want anyone on the internet to have access to that data–just that particular virtual machine. You can create a service account to authenticate that VM to Cloud Storage.

[00:26](javascript:;) Service accounts are named with an email address, but instead of passwords they use cryptographic keys to access resources. So, if a service account has been granted Compute Engine’s Instance Admin role, this

[00:38](javascript:;) would allow an application running in a VM with that service account to create, modify, and delete other VMs. Service accounts do need to be managed. For example, maybe Alice needs to manage which Google accounts can act as service accounts,

[00:54](javascript:;) while Bob just needs to be able to view a list of service accounts. Fortunately, in addition to being an identity, a service account is also a resource, so it

[01:04](javascript:;) can have IAM policies of its own attached to it. This means that Alice can have the editor role on a service account, and Bob can have the viewer role.

[01:13](javascript:;) This is just like granting roles for any other Google Cloud resource.

CLOUD INDENTITY

When new Google Cloud customers start using the platform, it’s common to log in to the Google Cloud Console with a Gmail account and then use Google Groups to collaborate with teammates who are in similar roles.

[00:12](javascript:;) Although this approach is easy to start with, it can present challenges later because the team’s identities are not centrally managed. This can be problematic if, for example, someone leaves the organization.

[00:24](javascript:;) With this setup, there’s no easy way to immediately remove a user’s access to the team’s cloud resources. With a tool called Cloud Identity, organizations can define policies and manage their users

[00:37](javascript:;) and groups using the Google Admin Console. Admins can log in and manage Google Cloud resources using the same usernames and passwords they already use in existing Active Directory or LDAP systems.

[00:51](javascript:;) Using Cloud Identity also means that when someone leaves an organization, an administrator can use the Google Admin Console to disable their account and remove them from groups. Cloud Identity is available in a free edition and also in a premium edition that provides

[01:05](javascript:;) capabilities to manage mobile devices. If you’re a Google Cloud customer who is also a Google Workspace customer, this functionality is already available to you in the Google Admin Console.

INTERACTING WITH GOOGLE CLOUD

There are four ways to access and interact with Google Cloud. The Cloud Console, the Cloud SDK and Cloud Shell, the APIs, and the Cloud Console Mobile App. Let’s explore each of those now.

[00:16](javascript:;) First is the Google Cloud Console, which is Google Cloud’s Graphical User Interface, GUI, that helps you deploy, scale, and diagnose production issues in a simple web-based interface. With the Cloud Console, you can easily find your resources, check their health, have full

[00:35](javascript:;) management control over them, and set budgets to control how much you spend on them. The Cloud Console also provides a search facility to quickly find resources and connect to instances

[00:45](javascript:;) via SSH in the browser. Second is through the Cloud SDK and Cloud Shell. The Cloud SDK is a set of tools that you can use to manage resources and applications hosted

[00:58](javascript:;) on Google Cloud. These include the gcloud tool, which provides the main command-line interface for Google Cloud products and services, gsutil, which lets you access Cloud Storage from the command

[01:10](javascript:;) line, and bq, a command-line tool for BigQuery. When installed, all of the tools within the Cloud SDK are located under the bin directory. Cloud Shell provides command-line access to cloud resources directly from a browser.

[01:27](javascript:;) Cloud Shell is a Debian-based virtual machine with a persistent 5 gigabyte home directory, which makes it easy to manage Google Cloud projects and resources. With Cloud Shell, the Cloud SDK gcloud command and other utilities are always installed,

[01:43](javascript:;) available, up to date, and fully authenticated. The third way to access Google Cloud is through application programming interfaces, or APIs. The services that make up Google Cloud offer APIs so that code you write can control them.

[01:59](javascript:;) The Cloud Console includes a tool called the Google APIs Explorer that shows which APIs are available, and in which versions. You can try these APIs interactively, even those that require user authentication.

[02:15](javascript:;) Suppose you’ve explored an API, and you’re ready to build an application that uses it. Do you have to start coding from scratch? No. Google provides Cloud Client libraries and Google API Client libraries in many popular

[02:28](javascript:;) languages to take a lot of the drudgery out of the task of calling Google Cloud from your code. Languages currently represented in these libraries are Java, Python, PHP, C#, Go, Node.js, Ruby,

[02:45](javascript:;) and C++. And finally, the fourth way to access and interact with Google Cloud is with the Cloud Console Mobile App, which can be used to start, stop, and use SSH to connect to Compute Engine instances and see logs from

[03:01](javascript:;) each instance. It also lets you stop and start Cloud SQL instances. Additionally, you can administer applications deployed on App Engine by viewing errors, rolling back deployments, and changing traffic splitting.

[03:17](javascript:;) The Cloud Console Mobile App provides up-to-date billing information for your projects and billing alerts for projects that are going over budget. You can set up customizable graphs showing key metrics such as CPU usage, network usage,

[03:33](javascript:;) requests per second, and server errors. The mobile app also offers alerts and incident management. You can download the Cloud Console Mobile App at cloud.google.com/console-app.

VIRTUAL MACHINES AND NETWORKS IN THE CLOUD

VIRTUAL PRIVATE CLOUD NETWORKING

n this section of the course, we’re going to explore how Google Compute Engine works with a focus on virtual networking. Many users start with Google Cloud by defining their own virtual private cloud inside their

[00:12](javascript:;) first Google Cloud project or by starting with the default virtual private cloud. So, what is a virtual private cloud? A virtual private cloud, or VPC, is a secure, individual, private cloud-computing model

[00:27](javascript:;) hosted within a public cloud – like Google Cloud! On a VPC, customers can run code, store data, host websites, and do anything else they could do in an ordinary private cloud,

[00:40](javascript:;) but this private cloud is hosted remotely by a public cloud provider. This means that VPCs combine the scalability and convenience of public cloud computing with the data isolation of private cloud computing.

[00:58](javascript:;) VPC networks connect Google Cloud resources to each other and to the internet. This includes segmenting networks, using firewall rules to restrict access to instances, and creating static routes to forward traffic to specific destinations.

[01:13](javascript:;) Here's something that tends to surprise a lot of new Google Cloud users: Google VPC networks are global. They can also have subnets, which is a segmented piece of the larger network, in any Google

[01:26](javascript:;) Cloud region worldwide. Subnets can span the zones that make up a region. This architecture makes it easy to define network layouts with global scope. Resources can even be in different zones on the same subnet.

[01:42](javascript:;) The size of a subnet can be increased by expanding the range of IP addresses allocated to it, and doing so won’t affect virtual machines that are already configured. For example, let’s take a VPC with one network that currently has one subnet defined in Google

[01:59](javascript:;) Cloud’s us-east1 region. If the VPC has two Compute Engine VMs attached to it, it means they’re neighbors on the same subnet even though they’re in different zones!

COMPUTE ENGINE

Earlier in the course, we explored infrastructure as a service, or IaaS. Now let’s explore Google Cloud’s IaaS solution: Compute Engine. With Compute Engine, users can create and run virtual machines on Google infrastructure.

[00:15](javascript:;) There are no upfront investments, and thousands of virtual CPUs can run on a system that’s designed to be fast and to offer consistent performance. Each virtual machine contains the power and functionality of a full-fledged operating

[00:30](javascript:;) system. This means a virtual machine can be configured much like a physical server: by specifying the amount of CPU power and memory needed, the amount and type of storage needed, and

[00:41](javascript:;) the operating system. A virtual machine instance can be created via the Google Cloud console, which is a web-based tool to manage Google Cloud projects and resources, the Google Cloud CLI, or the Compute Engine

[00:53](javascript:;) API. The instance can run Linux and Windows Server images provided by Google or any customized versions of these images. You can also build and run images of other operating systems and flexibly reconfigure

[01:08](javascript:;) virtual machines. A quick way to get started with Google Cloud is through the Cloud Marketplace, which offers solutions from both Google and third-party vendors. With these solutions, there’s no need to manually configure the software, virtual machine

[01:23](javascript:;) instances, storage, or network settings, although many of them can be modified before launch if that’s required. Most software packages in Cloud Marketplace are available at no additional charge beyond

[01:35](javascript:;) the normal usage fees for Google Cloud resources. Some Cloud Marketplace images charge usage fees, particularly those published by third parties, with commercially licensed software, but they all show estimates of their monthly

[01:48](javascript:;) charges before they’re launched. At this point, you might be wondering about Compute Engine’s pricing and billing structure. For the use of virtual machines, Compute Engine bills by the second with a one-minute minimum,

[02:02](javascript:;) and sustained-use discounts start to apply automatically to virtual machines the longer they run. So, for each VM that runs for more than 25% of a month, Compute Engine automatically applies

[02:13](javascript:;) a discount for every additional minute. Compute Engine also offers committed-use discounts. This means that for stable and predictable workloads, a specific amount of vCPUs and memory can be purchased for up to a 57% discount off of normal prices in return for committing

[02:31](javascript:;) to a usage term of one year or three years. And then there are Preemptible and Spot VMs. Let’s say you have a workload that doesn’t require a human to sit and wait for it to

[02:42](javascript:;) finish–such as a batch job analyzing a large dataset. You can save money, in some cases up to 90%, by choosing Preemptible or Spot VMs to run the job. A Preemptible or Spot VM is different from an ordinary Compute Engine VM in only one

[02:59](javascript:;) respect: Compute Engine has permission to terminate a job if its resources are needed elsewhere. Although savings are possible with preemptible or spot VMs, you'll need to ensure that your

[03:09](javascript:;) job can be stopped and restarted. Spot VMs differ from Preemptible VMs by offering more features. For example, preemptible VMs can only run for up to 24 hours at a time, but Spot VMs

[03:22](javascript:;) do not have a maximum runtime. However, the pricing is, currently the same for both. In terms of storage, Compute Engine doesn’t require a particular option or machine type to get high throughput between processing and persistent disks.

[03:36](javascript:;) That’s the default, and it comes to you at no extra cost. And finally, you’ll only pay for what you need with custom machine types. Compute Engine lets you choose the machine properties of your instances, like the number

[03:50](javascript:;) of virtual CPUs and the amount of memory, by using a set of predefined machine types or by creating your own custom machine types.

SCALING VIRTUAL MACHINES

As we’ve just seen, with Compute Engine, you can choose the most appropriate machine properties for your instances, like the number of virtual CPUs and the amount of memory, by using a set of predefined machine types, or

[00:12](javascript:;) by creating custom machine types. To do this, Compute Engine has a feature called Autoscaling, where VMs can be added to or subtracted from an application based on load metrics.

[00:26](javascript:;) The other part of making that work is balancing the incoming traffic among the VMs. Google’s Virtual Private Cloud (VPC) supports several different kinds of load balancing, which we’ll explore shortly.

[00:39](javascript:;) With Compute Engine, you can in fact configure very large VMs, which are great for workloads such as in-memory databases and CPU-intensive analytics, but most Google Cloud customers start off with scaling out, not up.

[00:53](javascript:;) The maximum number of CPUs per VM is tied to its “machine family” and is also constrained by the quota available to the user, which is zone-dependent. Specifications for currently available VM machine types can be found at cloud.google.com/compute/docs/machine-types

IMPORTANT VPC COMPATIBILITIES

Now let’s explore some of the most important Virtual Private Cloud compatibility features. Much like physical networks, VPCs have routing tables. VPC routing tables are built-in so you don’t have to provision or manage a router.

[00:15](javascript:;) They’re used to forward traffic from one instance to another within the same network, across subnetworks, or even between Google Cloud zones, without requiring an external IP address. Another thing you don’t have to provision or manage for Google Cloud is a firewall.

[00:32](javascript:;) VPCs provide a global distributed firewall, which can be controlled to restrict access to instances through both incoming and outgoing traffic. Firewall rules can be defined through network tags on Compute Engine instances, which is

[00:45](javascript:;) really convenient. For example, you can tag all your web servers with, say, “WEB,” and write a firewall rule saying that traffic on ports 80 or 443 is allowed into all VMs with the “WEB”

[00:58](javascript:;) tag, no matter what their IP address happens to be. You’ll remember that VPCs belong to Google Cloud projects, but what if your company has several Google Cloud projects, and the VPCs need to talk to each other?

[01:12](javascript:;) With VPC Peering, a relationship between two VPCs can be established to exchange traffic. Alternatively, to use the full power of Identity Access Management (IAM) to control who and what in one project can interact with a VPC in another, you can configure a Shared VPC.

CLOUD LOAD BALANCING

Previously, we explored how virtual machines can autoscale to respond to changing loads.

[00:06](javascript:;) But how do your customers get to your application when it might be provided by four VMs one moment, and by 40 VMs at another?

[00:14](javascript:;) That’s done through Cloud Load Balancing.

[00:17](javascript:;) The job of a load balancer is to distribute user traffic across multiple instances of an application.

[00:23](javascript:;) By spreading the load, load balancing reduces the risk that applications experience performance issues.

[00:29](javascript:;) Cloud Load Balancing is a fully distributed, software-defined, managed service for all your traffic.

[00:35](javascript:;) And because the load balancers don’t run in VMs that you have to manage, you don’t have to worry about scaling or managing them.

[00:42](javascript:;) You can put Cloud Load Balancing in front of all of your traffic: HTTP or HTTPS, other TCP and SSL traffic, and UDP traffic too.

[00:52](javascript:;) Cloud Load Balancing provides cross-region load balancing, including automatic multi-region failover, which gently moves traffic in fractions if backends become unhealthy.

[01:02](javascript:;) Cloud Load Balancing reacts quickly to changes in users, traffic, network, backend health, and other related conditions.

[01:10](javascript:;) And what if you anticipate a huge spike in demand?

[01:13](javascript:;) Say, your online game is already a hit; do you need to file a support ticket to warn Google of the incoming load?

[01:20](javascript:;) No.

[01:21](javascript:;) No so-called “pre-warming” is required.

[01:24](javascript:;) VPC offers a suite of load-balancing options: If you need cross-regional load balancing for a web application, use Global HTTP(S) load balancing.

[01:33](javascript:;) For Secure Sockets Layer traffic that is not HTTP, use the Global SSL Proxy load balancer.

[01:40](javascript:;) If it’s other TCP traffic that doesn’t use SSL, use the Global TCP Proxy load balancer.

[01:46](javascript:;) Those last two proxy services only work for specific port numbers, and they only work for TCP.

[01:52](javascript:;) If you want to load balance UDP traffic, or traffic on any port number, you can still load balance across a Google Cloud region with the Regional load balancer.

[02:00](javascript:;) Finally, what all those services have in common is that they’re intended for traffic coming into the Google network from the internet.

[02:07](javascript:;) But what if you want to load balance traffic inside your project, say, between the presentation layer and the business layer of your application?

[02:15](javascript:;) For that, use the Regional internal load balancer.

[02:19](javascript:;) It accepts traffic on a Google Cloud internal IP address and load balances it across Compute Engine VMs.

CLOUD DNS AND CLOUD CDN

One of the most famous free Google services is 8.8.8.8, which provides a public Domain Name Service to the world.

[00:07](javascript:;) DNS is what translates internet hostnames to addresses, and as you might imagine, Google has a highly developed DNS infrastructure.

[00:16](javascript:;) It makes 8.8.8.8 available so that everyone can take advantage of it.

[00:21](javascript:;) But what about the internet hostnames and addresses of applications built in Google Cloud?

[00:26](javascript:;) Google Cloud offers Cloud DNS to help the world find them.

[00:30](javascript:;) It’s a managed DNS service that runs on the same infrastructure as Google.

[00:35](javascript:;) It has low latency and high availability, and it’s a cost-effective way to make your applications and services available to your users.

[00:44](javascript:;) The DNS information you publish is served from redundant locations around the world.

[00:49](javascript:;) Cloud DNS is also programmable.

[00:51](javascript:;) You can publish and manage millions of DNS zones and records using the Cloud Console, the command-line interface, or the API.

[01:00](javascript:;) Google also has a global system of edge caches.

[01:03](javascript:;) Edge caching refers to the use of caching servers to store content closer to end users.

[01:09](javascript:;) You can use this system to accelerate content delivery in your application by using Cloud CDN - Content Delivery Network.

[01:17](javascript:;) This means your customers will experience lower network latency, the origins of your content will experience reduced load, and you can even save money.

[01:27](javascript:;) After HTTP(S) Load Balancing is set up, Cloud CDN can be enabled with a single checkbox.

[01:34](javascript:;) There are many other CDNs available out there, of course.

[01:37](javascript:;) If you are already using one, chances are, it’s a part of Google Cloud’s CDN Interconnect partner program, and you can continue to use it.

CONNECTING NETWORKS TO GOOGLE VPC

Many Google Cloud customers want to connect their Google Virtual Private Clouds to other networks in their system, such as on-premises networks or networks in other clouds. There are several effective ways to accomplish this.

[00:14](javascript:;) One option is to start with a Virtual Private Network connection over the internet and use the IPsec VPN protocol to create a “tunnel” connection. To make the connection dynamic, a Google Cloud feature called Cloud Router can be used.

[00:29](javascript:;) Cloud Router lets other networks and Google VPC, exchange route information over the VPN using the Border Gateway Protocol. Using this method, if you add a new subnet to your Google VPC, your on-premises network

[00:42](javascript:;) will automatically get routes to it. But using the internet to connect networks isn't always the best option for everyone, either because of security concerns or because of bandwidth reliability.

[00:54](javascript:;) So, a second option is to consider “peering” with Google using Direct Peering. Peering means putting a router in the same public data center as a Google point of presence

[01:06](javascript:;) and using it to exchange traffic between networks. Google has more than 100 points of presence around the world. Customers who aren’t already in a point of presence can work with a partner in the

[01:18](javascript:;) Carrier Peering program to get connected. Carrier peering gives you direct access from your on-premises network through a service provider's network to Google Workspace and to Google Cloud products that can be exposed

[01:31](javascript:;) through one or more public IP addresses. One downside of peering, though, is that it isn’t covered by a Google Service Level Agreement. If getting the highest uptimes for interconnection is important, using Dedicated Interconnect

[01:45](javascript:;) would be a good solution. This option allows for one or more direct, private connections to Google. If these connections have topologies that meet Google’s specifications, they can be covered by an SLA of up to 99.99%.

[01:58](javascript:;) Also, these connections can be backed up by a VPN for even greater reliability. And the final option we’ll explore is Partner Interconnect, which provides connectivity between an on-premises network and a VPC network through a supported service provider.

[02:17](javascript:;) A Partner Interconnect connection is useful if a data center is in a physical location that can't reach a Dedicated Interconnect colocation facility, or if the data needs don’t warrant an entire 10 GigaBytes per second connection.

[02:31](javascript:;) Depending on availability needs, Partner Interconnect can be configured to support mission-critical services or applications that can tolerate some downtime. As with Dedicated Interconnect, if these connections have topologies that meet Google’s specifications,

[02:46](javascript:;) they can be covered by an SLA of up to 99.99%, but note that Google isn’t responsible for any aspects of Partner Interconnect provided by the third-party service provider, nor any

[02:56](javascript:;) issues outside of Google's network.

STORAGE IN THE CLOUD

GOOGLE CLOUD STORAGE OPTIONS

Every application needs to store data, like media to be streamed or perhaps even sensor data from devices, and different applications and workloads require different storage database solutions.

[00:12](javascript:;) Google Cloud has storage options for structured, unstructured, transactional, and relational data.

[00:20](javascript:;) In this section of the course, we’ll explore Google Cloud’s five core storage products: Cloud Storage, Cloud SQL, Cloud Spanner, Firestore, and Cloud Bigtable.

[00:34](javascript:;) Depending on your application, you might use one or several of these services to do the job.

CLOUD STORAGE

Let’s begin with Cloud Storage, which is a service that offers developers and IT organizations durable and highly available object storage. But what is object storage? Object storage is a computer data storage architecture that manages data as “objects”

[00:17](javascript:;) and not as a file and folder hierarchy (file storage), or as chunks of a disk (block storage). These objects are stored in a packaged format which contains the binary form of the actual data itself, as well as relevant associated meta-data (such

[00:33](javascript:;) as date created, author, resource type, and permissions), and a globally unique identifier. These unique keys are in the form of URLs, which means object storage interacts well with web technologies.

[00:47](javascript:;) Data commonly stored as objects include video, pictures, and audio recordings. Cloud Storage is Google’s object storage product. It allows customers to store any amount of data, and to retrieve it as often as needed.

[01:00](javascript:;) It’s a fully managed scalable service that has a wide variety of uses. A few examples include serving website content, storing data for archival and disaster recovery, and distributing large data objects to end users via Direct Download.

[01:17](javascript:;) Cloud Storage’s primary use is whenever binary large-object storage (also known as a “BLOB”) is needed for online content such as videos and photos, for backup and archived data and for storage of intermediate results in

[01:30](javascript:;) processing workflows. Cloud Storage files are organized into buckets. A bucket needs a globally unique name and a specific geographic location for where it should be stored, and an ideal location for a bucket is where latency is minimized.

[01:46](javascript:;) For example, if most of your users are in Europe, you probably want to pick a European location, so either a specific Google Cloud region in Europe, or else the EU multi-region.

[01:57](javascript:;) The storage objects offered by Cloud Storage are immutable, which means that you do not edit them, but instead a new version is created with every change made. Administrators have the option to either allow each new version to completely overwrite the

[02:11](javascript:;) older one, or to keep track of each change made to a particular object by enabling “versioning” within a bucket. If you choose to use versioning, Cloud Storage will keep a detailed history of modifications

[02:23](javascript:;) -- that is, overwrites or deletes -- of all objects contained in that bucket. If you don’t turn on object versioning, by default new versions will always overwrite older versions.

[02:36](javascript:;) With object versioning enabled, you can list the archived versions of an object, restore an object to an older state, or permanently delete a version of an object, as needed.

[02:47](javascript:;) In many cases, personally identifiable information may be contained in data objects, so controlling access to stored data is essential to ensuring security and privacy are maintained. Using IAM roles and, where needed, access control lists (ACLs), organizations can conform

[03:04](javascript:;) to security best practices, which require each user to have access and permissions to only the resources they need to do their jobs, and no more than that. There are a couple of options to control user access to objects and buckets.

[03:20](javascript:;) For most purposes, IAM is sufficient. Roles are inherited from project to bucket to object. If you need finer control, you can create access control lists. Each access control list consists of two pieces of information.

[03:36](javascript:;) The first is a scope, which defines who can access and perform an action. This can be a specific user or group of users. The second is a permission, which defines what actions can be performed, like read or

[03:50](javascript:;) write. Because storing and retrieving large amounts of object data can quickly become expensive, Cloud Storage also offers lifecycle management policies. For example, you could tell Cloud Storage to delete objects older than 365 days;

[04:06](javascript:;) or to delete objects created before January 1, 2013; or to keep only the 3 most recent versions of each object in a bucket that has versioning enabled. Having this control ensures that you’re not paying for more than you actually need.

CLOUD STORAGE: STORAGE CLASSES AND DATA TRANSFER

There are four primary storage classes in Cloud Storage. The first is Standard Storage. Standard Storage is considered best for frequently accessed, or “hot,” data. It’s also great for data that’s stored for only brief periods of time.

[00:16](javascript:;) The second storage class is Nearline Storage. This is best for storing infrequently accessed data, like reading or modifying data on average once a month or less. Examples might include data backups, long-tail multimedia content, or data archiving.

[00:34](javascript:;) The third storage class is Coldline Storage. This is also a low-cost option for storing infrequently accessed data. However, as compared to Nearline Storage, Coldline Storage is meant for reading or modifying

[00:46](javascript:;) data, at most, once every 90 days. The fourth storage class is Archive Storage. This is the lowest-cost option, used ideally for data archiving, online backup, and disaster recovery. It’s the best choice for data that you plan to access less than once a year, because it

[01:06](javascript:;) has higher costs for data access and operations and a 365-day minimum storage duration. Although each of these four classes has differences, it’s worth noting there are several characteristics that apply across all of these storage classes.

[01:22](javascript:;) These include: Unlimited storage with no minimum object size requirement, Worldwide accessibility and locations, Low latency and high durability, A uniform experience, which extends to security, tools, and APIs, and,

[01:41](javascript:;) Geo-redundancy if data is stored in a multi-region or dual-region. This means placing physical servers in geographically diverse data centers to protect against catastrophic events and natural disasters, and load-balancing traffic for optimal performance.

[01:57](javascript:;) Cloud Storage has no minimum fee because you pay only for what you use, and prior provisioning of capacity isn’t necessary. And from a security perspective, Cloud Storage always encrypts data on the server side, before

[02:10](javascript:;) it’s written to disk, at no additional charge. Data traveling between a customer’s device and Google is encrypted by default using HTTPS/TLS (Transport Layer Security). Regardless of which storage class you choose, there are several ways to bring data into

[02:28](javascript:;) Cloud Storage. Many customers simply carry out their own online transfer using gsutil, which is the Cloud Storage command from the Cloud SDK. Data can also be moved in by using a drag and drop option in the Cloud Console, if accessed

[02:44](javascript:;) through the Google Chrome web browser. But what if you have to upload terabytes or even petabytes of data? Storage Transfer Service enables you to import large amounts of online data into Cloud Storage

[02:56](javascript:;) quickly and cost-effectively. The Storage Transfer Service lets you schedule and manage batch transfers to Cloud Storage from another cloud provider, from a different Cloud Storage region, or from an HTTP(S) endpoint.

[03:09](javascript:;) And then there is the Transfer Appliance, which is a rackable, high-capacity storage server that you lease from Google Cloud. You connect it to your network, load it with data, and then ship it to an upload facility

[03:24](javascript:;) where the data is uploaded to Cloud Storage. You can transfer up to a petabyte of data on a single appliance. Cloud Storage’s tight integration with other Google Cloud products and services means that

[03:36](javascript:;) there are many additional ways to move data into the service. For example, you can import and export tables to and from both BigQuery and Cloud SQL. You can also store App Engine logs, Firestore backups, and objects

[03:51](javascript:;) used by App Engine applications, like images. Cloud Storage can also store instance startup scripts, Compute Engine images, and objects used by Compute Engine applications.

CLOUD SQL

Google Cloud’s second core storage option is Cloud SQL. Cloud SQL offers fully managed relational databases, including MySQL, PostgreSQL, and SQL Server as a service. It’s designed to hand off mundane, but necessary and often time-consuming, tasks to Google—like

[00:19](javascript:;) applying patches and updates managing backups, and configuring replications—so your focus can be on building great applications. Cloud SQL: Doesn't require any software installation or maintenance. Can scale up to 64 processor cores, 400+ GB of RAM, and 30 TB of storage.

[00:39](javascript:;) Supports automatic replication scenarios, such as from a Cloud SQL primary instance, an external primary instance, and external MySQL instances. Supports managed backups, so backed-up data is securely stored and accessible if a restore

[00:56](javascript:;) is required. The cost of an instance covers seven backups. Encrypts customer data when on Google’s internal networks and when stored in database tables, temporary files, and backups. Includes a network firewall, which controls network access to each database instance.

[01:15](javascript:;) A benefit of Cloud SQL instances is that they are accessible by other Google Cloud services, and even external services. Cloud SQL can be used with App Engine using standard drivers like Connector/J for Java

[01:29](javascript:;) or MySQLdb for Python. Compute Engine instances can be authorized to access Cloud SQL instances and configure the Cloud SQL instance to be in the same zone as your virtual machine.

[01:43](javascript:;) Cloud SQL also supports other applications and tools that you might use, like SQL Workbench, Toad, and other external applications using standard MySQL drivers.

CLOUD SPANNER

The third core storage option offered by Google Cloud is Cloud Spanner. Cloud Spanner is a fully managed relational database service that scales horizontally, is strongly consistent, and speaks SQL.

[00:12](javascript:;) Battle tested by Google’s own mission-critical applications and services, Spanner is the service that powers Google’s $80 billion business. Cloud Spanner is especially suited for applications that require: A SQL relational database management system with joins and secondary indexes Built-in

[00:30](javascript:;) high availability Strong global consistency And high numbers of input and output operations per second. We’re talking tens of thousands of reads and writes per second or more.

FIRESTORE

Google Cloud’s fourth core storage option is Firestore. Firestore is a flexible, horizontally scalable, NoSQL cloud database for mobile, web, and server development. With Firestore, data is stored in documents and then organized into collections.

[00:18](javascript:;) Documents can contain complex nested objects in addition to subcollections. Firestore’s NoSQL queries can then be used to retrieve individual, specific documents or to retrieve all the documents in a collection that match your query parameters.

[00:34](javascript:;) Queries can include multiple, chained filters and combine filtering and sorting options. They're also indexed by default, so query performance is proportional to the size of the result set, not the dataset.

[00:47](javascript:;) Firestore uses data synchronization to update data on any connected device. However, it's also designed to make simple, one-time fetch queries efficiently. It caches data that an app is actively using, so the app can write, read, listen to, and

[01:04](javascript:;) query data even if the device is offline. When the device comes back online, Firestore synchronizes any local changes back to Firestore. Firestore leverages Google Cloud’s powerful infrastructure: automatic multi-region data

[01:13](javascript:;) replication, Firestore leverages Google Cloud’s powerful infrastructure: automatic multi-region data replication, strong consistency guarantees atomic batch operations and real transaction support. From a pricing perspective, you’re charged for each document read, write, and delete

[01:32](javascript:;) that you perform with Firestore. Queries are also charged at the rate of one “document read” per query, whether the query returns data or not. You’re also charged for the amount of storage your data consumes and for certain kinds of

[01:44](javascript:;) network bandwidth used to access your data. Ingress is currently free, and in many cases so is egress. Consult the pricing page for Firestore for details, or you can use Google’s Billing

[01:56](javascript:;) Calculator to estimate prices for your particular use case. In addition to the 10GiB of free network egress per month between US regions, Firestore has a free quota per day of:

[02:09](javascript:;) 50,000 document reads 20,000 document writes 20,000 document deletes, and, 1 GB of stored data Charges only begin once the free daily quota has been exceeded. This allows you to get started developing with Firestore for very little, or even for

[02:27](javascript:;) free.

CLOUD BIGTABLE

The last of Google Cloud’s core storage options we’re going to explore is Cloud Bigtable. Cloud Bigtable is Google's NoSQL big data database service. It's the same database that powers many core Google services, including Search, Analytics,

[00:16](javascript:;) Maps, and Gmail. Bigtable is designed to handle massive workloads at consistent low latency and high throughput, so it's a great choice for both operational and analytical applications, including Internet

[00:29](javascript:;) of Things, user analytics, and financial data analysis. When deciding which storage option is best, customers often choose Bigtable if: They’re working with more than 1TB of semi-structured or structured data.

[00:43](javascript:;) Data is fast with high throughput, or it’s rapidly changing. They’re working with NoSQL data. This usually means transactions where strong relational semantics are not required. Data is a time-series or has natural semantic ordering.

[00:58](javascript:;) They’re working with big data, running asynchronous batch or synchronous real-time processing on the data. Or they’re running machine learning algorithms on the data. Cloud Bigtable can interact with other Google Cloud services and third-party clients.

[01:15](javascript:;) Using APIs, data can be read from and written to Cloud Bigtable through a data service layer like Managed VMs, the HBase REST Server, or a Java Server using the HBase client.

[01:28](javascript:;) Typically this is used to serve data to applications, dashboards, and data services. Data can also be streamed in through a variety of popular stream processing frameworks like Dataflow Streaming, Spark Streaming, and Storm.

[01:44](javascript:;) And if streaming is not an option, data can also be read from and written to Cloud Bigtable through batch processes like Hadoop MapReduce, Dataflow, or Spark. Often, summarized or newly calculated data is written back to Cloud Bigtable or to a

[02:00](javascript:;) downstream database.

COMPARING STORAGE OPTIONS

Now that we’ve covered Google Cloud’s core storage options, let’s do a comparison to help highlight the most suitable service for a specific application or workflow.

[00:10](javascript:;) Consider using Cloud Storage if you need to store immutable blobs larger than 10 megabytes, such as large images or movies.

[00:18](javascript:;) This storage service provides petabytes of capacity with a maximum unit size of 5 terabytes per object.

[00:26](javascript:;) Consider using Cloud SQL or Cloud Spanner if you need full SQL support for an online transaction processing system.

[00:33](javascript:;) Cloud SQL provides up to 64 terabytes, depending on machine type, and Cloud Spanner provides petabytes.

[00:40](javascript:;) Cloud SQL is best for web frameworks and existing applications, like storing user credentials and customer orders.

[00:48](javascript:;) If Cloud SQL doesn’t fit your requirements because you need horizontal scalability, not just through read replicas, consider using Cloud Spanner.

[00:57](javascript:;) Consider Firestore if you need massive scaling and predictability together with real time query results and offline query support.

[01:05](javascript:;) This storage service provides terabytes of capacity with a maximum unit size of 1 megabyte per entity.

[01:12](javascript:;) Firestore is best for storing, syncing, and querying data for mobile and web apps.

[01:18](javascript:;) Finally, consider using Cloud Bigtable if you need to store a large number of structured objects.

[01:25](javascript:;) Cloud Bigtable doesn’t support SQL queries, nor does it support multi-row transactions.

[01:31](javascript:;) This storage service provides petabytes of capacity with a maximum unit size of 10 megabytes per cell and 100 megabytes per row.

[01:39](javascript:;) Bigtable is best for analytical data with heavy read and write events, like AdTech, financial, or IoT data.

[01:48](javascript:;) Depending on your application, it’s possible that you might use one, or several, of these services to do the job.

[01:54](javascript:;) You may have noticed that BigQuery hasn’t been mentioned in this section of the course.

[01:58](javascript:;) This is because it sits on the edge between data storage and data processing, and is covered in more depth in other courses.

[02:06](javascript:;) The usual reason to store data in BigQuery is so you can use its big data analysis and interactive querying capabilities, but it’s not purely a data storage product.

CONTAINERS IN THE CLOUD

INTRODUCTION TO CONTAINERS

In this section of the course we’ll explore containers and help you understand how they are used. Infrastructure as a service, or IaaS, allows you to share compute resources with other

[00:11](javascript:;) developers by using virtual machines to virtualize the hardware. This lets each developer deploy their own operating system (OS), access the hardware, and build their applications in a self-contained environment with access to RAM, file systems,

[00:28](javascript:;) networking interfaces, etc. This is where containers come in. The idea of a container is to give the independent scalability of workloads in PaaS and an abstraction layer of the OS and hardware in IaaS.

[00:43](javascript:;) A configurable system lets you install your favorite runtime, web server, database, or middleware, configure the underlying system resources, such as disk space, disk I/O, or networking, and build as you like.

[00:57](javascript:;) But flexibility comes with a cost. The smallest unit of compute is an app with its VM. The guest OS might be large, even gigabytes in size, and take minutes to boot.

[01:08](javascript:;) As demand for your application increases, you have to copy an entire VM and boot the guest OS for each instance of your app, which can be slow and costly.

[01:18](javascript:;) Now, with App Engine, you get access to programming services, so you only need to write your code in self-contained workloads that use these services and include any dependent libraries.

[01:31](javascript:;) This means that as demand for your app increases, the platform scales your app seamlessly and independently by workload and infrastructure. This scales rapidly, but there’s no option to fine-tune the underlying architecture to

[01:44](javascript:;) save cost. A container is an invisible box around your code and its dependencies with limited access to its own partition of the file system and hardware. It only requires a few system calls to create and it starts as quickly as a process.

[02:00](javascript:;) All that’s needed on each host is an OS kernel that supports containers and a container runtime. In essence, the OS is being virtualized. It scales like PaaS but gives you nearly the same flexibility as IaaS.

[02:15](javascript:;) This makes code ultra portable, and the OS and hardware can be treated as a black box. So you can go from development, to staging, to production, or from your laptop to the

[02:26](javascript:;) cloud, without changing or rebuilding anything. As an example, let’s say you want to scale a web server. With a container, you can do this in seconds and deploy dozens or hundreds of them, depending

[02:38](javascript:;) on the size or your workload, on a single host. That's just a simple example of scaling one container running the whole application on a single host. However, you'll probably want to build your applications using lots of containers, each

[02:52](javascript:;) performing their own function like microservices. If you build them this way and connect them with network connections, you can make them modular, deploy easily, and scale independently across a group of hosts.

[03:05](javascript:;) The hosts can scale up and down and start and stop containers as demand for your app changes or as hosts fail.

KUBERNETES

A product that helps manage and scale containerized applications is Kubernetes. So to save time and effort when scaling applications and workloads, Kubernetes can be bootstrapped using Google Kubernetes Engine or GKE.

[00:14](javascript:;) So, what is Kubernetes? Kubernetes is an open-source platform for managing containerized workloads and services. It makes it easy to orchestrate many containers on many hosts, scale them as microservices,

[00:29](javascript:;) and easily deploy rollouts and rollbacks. 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.

[00:40](javascript:;) The system is divided into a set of primary 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.

[00:52](javascript:;) Note that this is different to a node on Google Cloud which is a virtual machine running in Compute Engine. You can describe a set of applications and how they should interact with each other,

[01:03](javascript:;) and Kubernetes determines how to make that happen. Deploying containers on nodes by using a wrapper around one or more containers is what defines a Pod. A Pod is the smallest unit in Kubernetes that you create or deploy.

[01:18](javascript:;) It 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

[01:30](javascript:;) a hard dependency, you can package them into a single Pod and share networking and storage resources between them. The Pod provides a unique network IP and set of ports for your containers and configurable

[01:42](javascript:;) options that govern how your containers should run. One way to run a container in a Pod in Kubernetes is to use the kubectl run command, which starts a Deployment with a container running inside a Pod.

[01:55](javascript:;) A Deployment represents a group of replicas of the same Pod and keeps your Pods running even when the nodes they run on fail. A Deployment could represent a component of an application or even an entire app.

[02:09](javascript:;) To see a list of the running Pods in your project, run the command: $ kubectl get pods Kubernetes creates a Service with a fixed IP address for your Pods, and a controller

[02:19](javascript:;) says "I need to attach an external load balancer with a public IP address to that Service so others outside the cluster can access it." In GKE, the load balancer is created as a network load balancer.

[02:32](javascript:;) Any client that reaches that IP address will be routed to a Pod behind the Service. A Service is an abstraction which defines a logical set of Pods and a policy by which

[02:43](javascript:;) to access them. As Deployments create and destroy Pods, Pods will be assigned their own IP addresses, but those addresses don't remain stable over time. A Service group is a set of Pods and provides a stable endpoint (or fixed IP address) for

[03:00](javascript:;) them. For example, if you create two sets of Pods called frontend and backend and put them behind their own Services, the backend Pods might change, but frontend Pods are not aware of

[03:12](javascript:;) this. They simply refer to the backend Service. To scale a Deployment, run the kubectl scale command. In this example, three Pods are created in your Deployment, and they're placed behind

[03:25](javascript:;) the Service and share one fixed IP address. You could also use autoscaling with other kinds of parameters; for example, you can specify that the number of Pods should increase when CPU utilization reaches a certain limit.

[03:41](javascript:;) So far, we’ve seen how to run imperative commands like expose and scale. This works well to learn and test Kubernetes step-by-step. But the real strength of Kubernetes comes when you work in a declarative way.

[03:54](javascript:;) Instead of issuing commands, you provide a configuration file that tells Kubernetes what you want your desired state to look like, and Kubernetes determines how to do it. You accomplish this by using a Deployment config file.

[04:08](javascript:;) To get this file, you can run a kubectl get deployments command, and you'll get a Deployment configuration file that looks like this. You can check your Deployment to make sure the proper number of replicas is running by

[04:20](javascript:;) using either kubectl get deployments or kubectl describe deployments. To run five replicas instead of three, all you do is update the Deployment config file and run the kubectl apply command to use the updated config file.

[04:36](javascript:;) You can still reach your endpoint as before by using kubectl get services to get the external IP of the Service and reach the public IP address from a client.

[04:46](javascript:;) The last question is, what happens when you want to update a new version of your app? Well, you want to update your container to get new code in front of users, but rolling

[04:55](javascript:;) out all those changes at one time would be risky. So in this case, you would use kubectl rollout or change your deployment configuration file and then apply the change using kubectl apply.

[05:07](javascript:;) New Pods will then be created according to your new update strategy. Here’s an example configuration that will create new version Pods individually and wait for a new Pod to be available before destroying one of the old Pods.

GOOGLE KUBERNETES ENGINE

So now that we have a basic understanding of containers and Kubernetes, let’s talk about Google Kubernetes Engine, or GKE. GKE is a Google-hosted managed Kubernetes service in the cloud.

[00:12](javascript:;) The GKE environment consists of multiple machines, specifically Compute Engine instances, grouped together to form a cluster. You can create a Kubernetes cluster with Kubernetes Engine by using the Google Cloud Console or the g-cloud command that's provided by the Cloud software

[00:28](javascript:;) development kit. GKE clusters can be customized, and they support different machine types, number of nodes, and network settings. Kubernetes provides the mechanisms through which you interact with your cluster.

[00:43](javascript:;) Kubernetes commands and resources are used to deploy and manage applications, perform administration tasks, set policies, and monitor the health of deployed workloads. Running a GKE cluster comes with the benefit of advanced cluster management features that

[00:58](javascript:;) Google Cloud provides. These include: Google Cloud's load-balancing for Compute Engine instances Node pools to designate subsets of nodes within a cluster for additional flexibility Automatic scaling of your cluster's node instance count

[01:14](javascript:;) Automatic upgrades for your cluster's node software Node auto-repair to maintain node health and availability Logging and monitoring with Google Cloud's operations suite for visibility into your cluster Running your application in GKE clusters is also a good foundation to have if you’ll

[01:31](javascript:;) need to bridge your on-prem and cloud resources. To start up Kubernetes on a cluster in GKE, all you do is run this command: $> gcloud container clusters create k1

HYBRID AND MULTI – CLOUD

Now that you’ve seen how containers work, we’re going to take that information a step further and explore how they can be used in a modern hybrid cloud and multi-cloud

[00:08](javascript:;) architecture. Let’s begin by looking at a typical on-premises distributed systems architecture, which is how businesses traditionally met their enterprise computing needs before cloud computing. Most enterprise-scale applications are designed as distributed systems,

[00:26](javascript:;) spreading the computing workload required to provide services over two or more networked servers. In recent years, containers have become a popular way to break these workloads down into “microservices” so they can be more easily maintained and

[00:39](javascript:;) expanded. Traditionally, these enterprise systems–and their workloads, containerized or not–have been located on-premises, which means they’re housed on a set of high-capacity servers running somewhere within the company’s network or within a company-owned data center.

[00:55](javascript:;) When an application’s computing needs begin to outstrip its available computing resources, a company using on-premises systems would need to requisition more (or more powerful) servers, install them on the company network (after any necessary network changes or expansion),

[01:11](javascript:;) configure the new servers, and finally load the application and its dependencies onto the new servers, before resource bottlenecks could be resolved. The time required to complete an on-premises upgrade of this kind could be anywhere from

[01:24](javascript:;) several months to one or more years. It might also be quite costly, especially when you consider the useful lifespan of the average server is only 3-5 years. But what if you need more computing power now, not months from now?

[01:39](javascript:;) What if your company wants to relocate some workloads away from on-premises to the cloud to take advantage of lower costs and higher availability, but is unwilling (or unable) to move the entire enterprise application from the on-premises network?

[01:53](javascript:;) And what if you want to use specialized products and services that are only available in the cloud? This is where a modern hybrid or multi-cloud architecture can help. It allows you to keep parts of your systems infrastructure on-premises while moving other

[02:08](javascript:;) parts to the cloud, thus creating an environment that is uniquely suited to your company’s needs. Move only specific workloads to the cloud at your own pace, because a full-scale migration

[02:18](javascript:;) is not required for it to work. Take advantage of the flexibility, scalability, and lower computing costs offered by cloud services for running the workloads you decide to migrate. And add specialized services, such as machine learning, content caching, data analysis,

[02:33](javascript:;) long-term storage, and IoT, to your computing resources toolkit. In the next video you’ll learn about Google Cloud’s answer to modern hybrid and multi-cloud distributed systems and services management.

ANTHOS

You might have heard a lot recently concerning the adoption of “hybrid” architecture for powering distributed systems and services. You might have even heard about Google’s answer to modern hybrid and multi-cloud distributed

[00:11](javascript:;) systems and services management, called Anthos. But what exactly is Anthos? Anthos is a hybrid and multi-cloud solution powered by the latest innovations in distributed systems and service management software from Google.

[00:27](javascript:;) The Anthos framework rests on Kubernetes and GKE On-Prem. This provides the foundation for an architecture that’s fully integrated and has centralized management through a central control plane that supports policy-based application lifecycle

[00:40](javascript:;) delivery across hybrid and multiple cloud environments. Anthos also provides a rich set of tools for monitoring and maintaining the consistency of your applications across all of your network, whether on-premises, in the cloud, or in multiple

[00:55](javascript:;) clouds. Let’s take a deeper look at this framework as we build a modern hybrid infrastructure stack, step by step, with Anthos. First, let’s look at Google Kubernetes Engine on the cloud side of our hybrid network.

[01:11](javascript:;) Google Kubernetes Engine: Is a managed, production-ready environment for deploying containerized applications. Operates seamlessly with high availability and an SLA. Runs Certified Kubernetes, thus ensuring portability across clouds and on-premises.

[01:29](javascript:;) Includes auto node repair, auto upgrade, and autoscaling. And uses regional clusters for high availability with multiple control planes and node storage replication across multiple zones. Its counterpart on the on-premises side of our hybrid network is GKE On-Prem.

[01:48](javascript:;) GKE On-Prem: Is a turnkey, production-grade, conformant version of Kubernetes with a best-practice configuration pre-loaded. Provides an easy upgrade path to the latest Kubernetes releases that have been validated and tested by Google.

[02:05](javascript:;) Provides access to container services on Google Cloud such as Cloud Build, Container Registry, and Cloud Audit Logs. Integrates with Istio, Knative, and Cloud Marketplace solutions. And ensures a consistent Kubernetes version and experience across cloud and on-premises

[02:21](javascript:;) environments. Both Google Kubernetes Engine and GKE On-Prem integrate with Marketplace so that all of the clusters in your network, whether on-premises or in the cloud, have access to the same repository

[02:34](javascript:;) of containerized applications. This allows you to use the same configurations on both sides of the network, which reduces time spent maintaining conformity between your clusters. You also spend less time developing applications because of a write-once/replicate-anywhere

[02:49](javascript:;) approach. Enterprise applications might use hundreds of microservices to handle computing workloads. Keeping track of all these services and monitoring their health can quickly become a challenge. Anthos Service Mesh and Istio Open Source Service Mesh take all of the guesswork out

[03:09](javascript:;) of managing and securing your microservices. These service mesh layers communicate across the hybrid network using Cloud Interconnect to sync and pass their data. Cloud Logging and Cloud Monitoring are the built-in logging and monitoring solutions

[03:25](javascript:;) for Google Cloud. Google Cloud’s operations suite offers a fully managed logging, metrics collection, monitoring, dashboarding, and alerting solution that watches all sides of your hybrid or multi-cloud network. It’s the ideal solution for customers wanting a single, easy to configure, powerful cloud-based

[03:44](javascript:;) observability solution that also gives you a ‘single pane of glass’ dashboard to monitor all of your environments. Finally, Anthos Configuration Management provides a single authoritative source of truth for

[03:57](javascript:;) your clusters’ configuration. This is kept in the Policy Repository, which is actually a git repository. The repository can be located on-premises or hosted in the cloud. The Anthos Configuration Management agents use the Policy Repository to enforce configurations

[04:15](javascript:;) locally in each environment, thus managing the complexity of owning clusters across environments. Anthos Configuration Management also gives administrators and developers the ability to deploy code changes with a single repository commit and the option to implement configuration

[04:32](javascript:;) inheritance by using Namespaces, which is a way to prevent naming and permissions collisions within your application. You can learn more about Anthos by heading to cloud.google.com/anthos.

APPLICATION IN THE CLOUD

APP ENGINE

So far in this course, we’ve provided an introduction to Google Cloud and explored the options and benefits related to using virtual machines networks, storage, and containers in the Cloud.

[00:12](javascript:;) In this section of the course, we’ll turn our attention to developing applications in the Cloud. We’ll begin with App Engine, which is a fully managed, serverless platform for developing

[00:22](javascript:;) and hosting web applications at scale. So, how does it work? With App Engine, you can choose from popular coding languages, libraries, and frameworks to develop apps with tools you’re familiar with, and then automatically provision servers

[00:37](javascript:;) and scale app instances based on demand. This means you can upload your code and Google will manage your app's availability. Coding options include Eclipse, IntelliJ, Maven, Git, Jenkins, and PyCharm.

[00:55](javascript:;) With App Engine, there are no servers to provision or maintain. App Engine provides built-in services and APIs, like NoSQL datastores Memcache load balancing, health checks, application logging, and a user authentication API that’s common

[01:12](javascript:;) to most applications. App Engine also offers software development kits, SDKs, to help you develop, deploy, and manage your apps on your local machine. Each SDK includes: All of the APIs and libraries available to

[01:28](javascript:;) App Engine, A simulated, secure sandbox environment that emulates all of the App Engine services on your local computer, And deployment tools to upload your application to the cloud and manage different versions.

[01:41](javascript:;) The SDK manages your application locally, and the Google Cloud Console manages your application in production. You can use the Cloud Console’s web-based interface to create new applications, configure domain names, change which version of your application is

[01:57](javascript:;) live, examine access and error logs, and much more. From a security perspective, the Security Command Center–Google Cloud’s security and risk management platform–keeps web applications safe. Through the Cloud Console, you can use the Security Command Center to automatically scan

[02:16](javascript:;) and detect common web application vulnerabilities.

APP ENGINE ENVIRONMENTS

There are two types of App Engine environments: standard and flexible. The App Engine standard environment is based on container instances running on Google's infrastructure. Containers are preconfigured with a runtime from a standardized list of supported languages

[00:15](javascript:;) and versions, which includes libraries that support App Engine standard APIs. For many applications, the standard environment runtimes and libraries may be all you need. Standard environment features include: Persistent storage with queries, sorting, and transactions

[00:34](javascript:;) Automatic scaling and load balancing Asynchronous task queues for performing work outside the scope of a request Scheduled tasks for triggering events at specified times or regular intervals And integration with other Google Cloud services and APIs

[00:51](javascript:;) There are a couple of requirements for using the standard environment: You must use specified versions of Java, Python, PHP, Go, Node.js, and Ruby, and Your application must conform to sandbox constraints that are dependent on runtime.

[01:10](javascript:;) Applications run in a secure, sandboxed environment. This allows the App Engine standard environment to distribute requests across multiple servers and scale servers to meet traffic demands. This means that your application runs within its own secure, reliable environment

[01:26](javascript:;) that is independent of the hardware operating system, or physical location of the server. A standard environment workflow typically follows these three steps: First, a web application is developed and tested locally.

[01:41](javascript:;) Second, the SDK is used to deploy the application to App Engine. And third, App Engine scales and services the application. App Engine also offers a flexible environment. If the standard environment’s sandbox model is too restrictive for you, the flexible environment

[02:00](javascript:;) can let you specify the type of container your web application will run in. This option lets an application run inside Docker containers on Google Cloud’s Compute Engine virtual machines.

[02:10](javascript:;) In this case, App Engine manages Compute Engine machines for you. This means that: Instances are health-checked, healed as necessary, and co-located with other module instances within the project. Critical, backward-compatible updates are automatically applied to the underlying operating

[02:28](javascript:;) system. VM instances are automatically located by geographical region according to the settings in your project. Google's management services ensure that all of a project's VM instances are co-located for optimal performance.

[02:44](javascript:;) And VM instances are restarted on a weekly basis. During restarts, Google's management services will apply any necessary operating system and security updates. The flexible environment supports microservices, authorization, SQL and NoSQL databases, traffic splitting,

[03:02](javascript:;) logging, search, versioning, security scanning, Memcache, and content delivery networks. App Engine Flexible allows users to also benefit from custom configurations and libraries while still keeping their main focus on what they do best – writing code.

[03:21](javascript:;) In addition, the App Engine flexible environment allows you to customize the runtime and the operating system of your virtual machine by using Dockerfiles. As in App Engine Standard, supported runtimes include Python, Java, Go, Node.js, PHP, and

[03:39](javascript:;) Ruby. However, in App Engine Flexible, developers can also use different versions of these runtimes or provide their own custom runtime by supplying a custom Docker image or using a Dockerfile

[03:51](javascript:;) from the open source community. So, how do these two environments compare to each other? Let’s start with the standard environment, which is fast. It starts up instances of your application in seconds, but you have less access to the

[04:05](javascript:;) infrastructure in which your application runs. With the standard environment, you can’t use SSH to connect to the virtual machines on which your application runs, and you can’t write to a local disk.

[04:17](javascript:;) The standard environment does support third-party binaries for certain languages, and you can use App Engine to make calls to the network. Finally, in terms of pricing, after a free tier usage, you pay per instance class with

[04:29](javascript:;) automatic shutdown. The flexible environment takes minutes to start up, instead of seconds. But it lets you use SSH to connect to the virtual machines on which your application runs, it lets you use local disk for scratch space,

[04:43](javascript:;) it lets you install third-party software, and it lets your application make calls to the network without going through App Engine. In terms of pricing, with the flexible environment, you pay for resource allocation per hour with

[04:55](javascript:;) no automatic shutdown. Because App Engine uses Docker containers, you may be wondering how App Engine compares to Google Kubernetes Engine. App Engine's standard environment is for people who want the service to take maximum control

[05:05](javascript:;) of their web and mobile application’s deployment and scaling. App Engine's standard environment is for people who want the service to take maximum control of their web and mobile application’s deployment and scaling.

[05:14](javascript:;) Google Kubernetes Engine, however, gives the application owner the full flexibility of Kubernetes. App Engine's flexible environment is somewhere between the two.

GOOGLE CLOUD API MANAGEMENT TOOLS

Now that you’ve had a thorough overview of App Engine, let’s transition to Cloud Endpoints and Apigee API Management. These are management tools for Google Cloud’s application programming interface, or API.

[00:12](javascript:;) So what exactly is an API? A software service’s implementation can be complex and changeable. If other software services had to be explicitly coded in detail in order to use that service,

[00:24](javascript:;) the result would be brittle and error-prone. So instead, application developers structure the software they write so that it presents a clean, well-defined interface that hides unnecessary detail, and then they document

[00:37](javascript:;) that interface. That’s an application programming interface. The underlying implementation can change, as long as the interface doesn’t, and other pieces of software that use the API don’t have to know or care.

[00:51](javascript:;) Sometimes you do have to change an API, perhaps to add or deprecate a feature. To cleanly make this kind of change to the API, developers create versions. For example, version 2 of an API might contain calls that version 1 does not.

[01:06](javascript:;) This means that programs that consume the API can specify the API version they want to use in their calls. Supporting an API is a very important task, and Google Cloud provides three API management

[01:18](javascript:;) tools: Cloud Endpoints, API Gateway, and Apigee API Management. Cloud Endpoints is a distributed API management system that uses a distributed Extensible Service Proxy, which is a service proxy that runs in its own Docker container.

[01:35](javascript:;) The goal is to help you create and maintain even the most demanding APIs with low latency and high performance. Cloud Endpoints provides an API console, hosting, logging, monitoring, and other features to

[01:49](javascript:;) help you create, share, maintain, and secure your APIs. You can use Cloud Endpoints with any APIs that support the OpenAPI Specification. Cloud Endpoints supports applications running in App Engine, Google Kubernetes Engine, and

[02:06](javascript:;) Compute Engine. Clients include Android, iOS, and Javascript. API Gateway is another API management tool. Web-based services today provide a huge variety of functionality, meaning everything from map, weather, and image services, to games, auctions, and many other service types.

[02:26](javascript:;) Service providers have many options for how to implement, deploy, and manage their services. For example, one service might be developed in Java or .NET, while another uses Node.js. Backend implementations can also vary for a single service provider.

[02:45](javascript:;) A service provider might have legacy services implemented using one architecture, and new services implemented using a completely different architecture. API Gateway enables you to provide secure access to your backend services through a

[02:58](javascript:;) well-defined REST API that is consistent across all of your services, regardless of the service implementation. Clients consume your REST APIS to implement standalone apps for a mobile device or tablet,

[03:11](javascript:;) through apps running in a browser, or through any other type of app that can make a request to an HTTP endpoint. Another Google Cloud platform available for developing and managing API proxies is Apigee

[03:24](javascript:;) API Management. Unlike Cloud Endpoints, Apigee API Management has a specific focus on business problems, like rate limiting, quotas, and analytics. In fact, many Apigee API Management users provide a software service to other companies.

[03:41](javascript:;) Backend services for Apigee API Management don't have to be in Google Cloud, and as a result, engineers also often use it to take apart legacy applications. So, instead of replacing a large, important application in one move, they can use Apigee

[03:54](javascript:;) API Management to peel off its services individually instead. This allows them to stand up microservices to implement each in turn until the legacy application can finally be retired.

CLOUD RUN

The final application platform we’ll explore in this section of the course is Cloud Run, a managed compute platform that lets you run stateless containers via web requests or Pub/Sub

[00:10](javascript:;) events. Cloud Run is serverless. That means it removes all infrastructure management tasks so you can focus on developing applications. It’s built on Knative, an open API and runtime environment built on Kubernetes that gives

[00:25](javascript:;) you freedom to move your workloads across different environments and platforms. It can be fully managed on Google Cloud, on Google Kubernetes Engine, or anywhere Knative runs. Cloud Run is fast.

[00:38](javascript:;) It can automatically scale up and down from zero almost instantaneously, and it charges you only for the resources you use, calculated down to the nearest 100 milliseconds, so you‘ll

[00:49](javascript:;) never pay for your over-provisioned resources. The Cloud Run developer workflow is a straightforward three-step process. First, you write your application using your favorite programming language. This application should start a server that listens for web requests.

[01:07](javascript:;) Second, you build and package your application into a container image. Third, the container image is pushed to Artifact Registry, where Cloud Run will deploy it. Note that Cloud Run can only deploy images that are stored in Artifact Registry.

[01:25](javascript:;) You can build, push and deploy your own code from your local source if you have the required permissions. You can also deploy an image that already exists in Artifact Registry.

[01:37](javascript:;) Once you’ve deployed your container image, you’ll get a unique HTTPS URL back. Cloud Run then starts your container on demand to handle requests, and ensures that all incoming requests are handled by dynamically adding and removing containers.

[01:53](javascript:;) Because Cloud Run is serverless, it means that you, as a developer, can focus on building your application and not on building and maintaining the infrastructure that powers it. For some use cases, a container-based workflow is great,

[02:06](javascript:;) because it gives you a great amount of transparency and flexibility. If you build the container image you have the power to decide exactly what file ends up in your container image, and how it gets there.

[02:18](javascript:;) However, building an application is hard enough already, let alone having to think about containerization and the responsibilities that come with that. Sometimes, you’re just looking for a way to turn source code into an HTTPS endpoint,

[02:31](javascript:;) and you want your vendor to make sure your container image is secure, well-configured built in a consistent way. With Cloud Run, you can do both. You can use a container-based workflow, as well as a source-based workflow.

[02:47](javascript:;) If you use the source-based approach, you’ll deploy your source code, instead of a container image. Cloud Run then builds your source and packages the application into a container image for

[02:57](javascript:;) you. Cloud Run does this using Buildpacks - an open source project. Cloud Run handles HTTPS serving for you. That means you only have to worry about handling web requests,

[03:11](javascript:;) and you can let Cloud Run take care of adding the encryption. By default, your application is exposed on a unique subdomain of the global run.app domain. You can also use your own, custom domain.

[03:25](javascript:;) Cloud Run manages everything else: Generating a valid SSL certificate Configuring SSL termination correctly with secure settings And handling incoming requests, decrypting them, and forwarding them to your application The pricing model on Cloud Run is unique; as you only pay for the system resources you

[03:44](javascript:;) use while a container is handling web requests, with a granularity of 100ms, and when it’s starting or shutting down. You don’t pay for anything if your container doesn’t handle requests.

[03:57](javascript:;) Additionally, there is a small fee for every one million requests you serve. The price of container time increases With CPU and memory. A container with more vCPU and memory is more expensive.

[04:11](javascript:;) Today, Cloud run can allocate up to 4 vCPUs and 8 gigabytes of memory. Most of the other compute products (such as Compute Engine), charge for servers as long as they’re running, even if you’re not using them.

[04:23](javascript:;) That means you’re often paying for idle server capacity. You can use Cloud Run to run any binary, as long as it’s compiled for Linux sixty-four bit. Now, this means you can use Cloud Run to run web applications written using popular languages,

[04:40](javascript:;) such as: Java Python Node.js PHP Go C++ And you can also run code written in less popular languages: Cobol Haskell Perl As long as your app handles web requests, you’re good to go.

DEVELOPING AND DEPLOYING IN THE CLOUD

DEVELOPMENT IN THE CLOUD

Many users develop impressive applications using Google Cloud’s products and services. And when an app is ready, Google Cloud can also be used to deploy it. In this section of the course, we’ll explore Google Cloud methods for development in the

[00:14](javascript:;) cloud, which includes Cloud Source Repositories, Cloud Functions, and Terraform. After that, we’ll look at deployment with infrastructure as code. Let’s begin by looking at development in the cloud. Many Google Cloud customers use Git repositories to store, version, and manage their source

[00:32](javascript:;) code trees. That means they either run their own Git instances, which is a great option if total control is required, or they use a hosted-Git provider, which means less work if total control isn’t

[00:43](javascript:;) required. But what if there were a third option, where you could keep code private to a Google Cloud project and use IAM permissions to protect it, but not have to maintain the Git instance

[00:54](javascript:;) yourself? Well, that third option is available with Cloud Source Repositories. Cloud Source Repositories provides full-featured Git repositories hosted on Google Cloud that support the collaborative development of any application or service, including those that

[01:10](javascript:;) run on App Engine and Compute Engine. With Cloud Source Repositories, you can have any number of private Git repositories. This allows code associated with a cloud project to be organized the way you choose.

[01:24](javascript:;) It also allows Google Cloud diagnostics tools, like Debugger and Error Reporting, to use the code from Git repositories to track down issues to specific errors in deployed code without slowing down your users.

[01:36](javascript:;) If your code is already in GitHub or BitBucket repositories, it can be migrated into your cloud project and used just like any other repository, including browsing and diagnostics. Many applications contain event-driven parts.

[01:49](javascript:;) For example, maybe you have an application that lets users upload images. When that event takes place, the image might need to be processed in a few different ways, like converting a thumbnail into different sizes, and storing each new file in a repository.

[02:06](javascript:;) You could integrate this function into your application, but then you’d have to provide compute resources for it–whether it happens once a millisecond or once a day. With Cloud Functions, you could write a single-purpose function that completes the necessary image

[02:20](javascript:;) manipulations and then arrange for it to automatically run whenever a new image is uploaded. 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

[02:35](javascript:;) need to manage a server or a runtime environment. You can use these functions to construct application workflows from individual business logic tasks. You can also use Cloud Functions to connect and extend cloud services.

[02:49](javascript:;) You’re billed to the nearest 100 milliseconds, but only while your code is running. Cloud Functions supports writing source code in a number of programming languages. These include Node.js, Python, Go, Java, .Net Core, Ruby, and PHP.

[03:08](javascript:;) For more information about the supported specific versions, refer to the runtimes documentation. Events from Cloud Storage and Pub/Sub can trigger Cloud Functions asynchronously, or you can use HTTP invocation for synchronous execution.

DEPLOYMENT: INFRASTRUCTURE AS CODE

Creating an environment in Google Cloud can mean lots of work–like setting up a compute network and storage resources and then keeping track of their configurations. This process can be done manually by writing the commands you need to set up your environment

[00:13](javascript:;) the way you want. However, this is labor-intensive and requires updating commands if you want to change the environment or manually writing new commands if you want to clone an environment.

[00:24](javascript:;) It’s more efficient to use a template. Using a template allows you to write the specifications for your application environment in the same way you’d write a configuration file, but your template can then be deployed in a scaled

[00:36](javascript:;) environment to quickly create as many identical application environments as needed. This can be done with Terraform. To use Terraform, you create a template file, using HashiCorp Configuration Language (HCL),

[00:49](javascript:;) that describes what you want the components of your environment to look like. Terraform then uses that template to determine the actions needed to create the environment your template describes.

[01:01](javascript:;) If you need to change the environment, you can edit your template and then use Terraform to update the environment to match the change. You can store and version-control your Terraform templates in Cloud Source Repositories.

LOGGING AND MONITORING IN THE CLOUD

THE IMPORTANCE OF MONITORING

In this section of the course we’ll transition our focus from developing and deploying in the cloud, to logging and monitoring. Let’s begin with monitoring. Monitoring is the foundation of product reliability.

[00:13](javascript:;) It reveals what needs urgent attention and shows trends in application usage patterns, which can yield better capacity planning and generally help improve an application client's experience and lessen their pain.

[00:26](javascript:;) In Google's Site Reliability Engineering book, which is available to read at landing.google.com/sre/books, monitoring is defined as: "Collecting, processing, aggregating, and displaying real-time quantitative data about a system, such as query counts and types, error counts and types, processing times,

[00:47](javascript:;) and server lifetimes." With monitoring, you can ensure continued system operations, uncover trend analyses over time, build dashboards, alert personnel when systems violate predefined service level objectives (SLOs), compare systems and systems changed, and provide data for improved incident

[01:07](javascript:;) response–just to name a few tasks. An application client normally only sees the public side of a product, and as a result, developers and business stakeholders both tend to think

[01:19](javascript:;) that the most crucial way to make the client happy is by spending the most time and effort on developing that part of the product. However, to be truly reliable, even the very best products still must be deployed

[01:31](javascript:;) into environments with enough capacity to handle the anticipated client load. Great products also need thorough testing, preferably automated testing, and a refined continuous integration/continuous development (CI/CD) release pipeline. Postmortems and root cause analyses are the DevOps team's way of letting the client know

[01:50](javascript:;) why an incident happened and why it’s unlikely to happen again. In this context we’re discussing a system or software failure, but the term “incident” can also be used to describe a breach of security.

[02:01](javascript:;) Here, transparency is key to building trust. why an incident happened and why it’s unlikely to happen again. In this context we’re discussing a system or software failure, but the term “incident”

[02:04](javascript:;) can also be used to describe a breach of security. Here, transparency is key to building trust. We need our products to improve continually, and we need data we can receive from monitoring

[02:11](javascript:;) to make sure that happens. We need dashboards to provide business intelligence so our DevOps personnel have the data they need to do their jobs. We need automated alerts because humans tend to look at things only when there's something

[02:24](javascript:;) important to look at. An even better option is to construct automated systems to handle as many alerts as possible so humans only have to look at the most critical issues.

[02:35](javascript:;) Finally, we need monitoring tools that help provide data crucial to debugging application functional and performance issues. We’ll look more closely at Google’s integrated monitoring tools a bit later in this module.

MEASURING PERFORMANCE AND RELIABILITY

There are “Four Golden Signals” that measure a system’s performance and reliability. They are latency, traffic, saturation, and errors. Latency measures how long it takes a particular part of a system to return a result.

[00:16](javascript:;) Latency is important because: It directly affects the user experience. Changes in latency could indicate emerging issues. Its values may be tied to capacity demands. And it can be used to measure system improvements.

[00:32](javascript:;) But how exactly is it measured? Sample latency metrics include: Page load latency Number of requests waiting for a thread Query duration Service response time Transaction duration Time to first response Time to complete data return

[00:54](javascript:;) The next signal is traffic, which measures how many requests are reaching your system. Traffic is important because: It’s an indicator of current system demand. Its historical trends are used for capacity planning.

[01:09](javascript:;) And it’s a core measure when calculating infrastructure spend. Sample traffic metrics include: number of HTTP requests per second number of requests for static vs. dynamic content Network I/O number of concurrent sessions number of transactions per second

[01:33](javascript:;) number of retrievals per second number of active requests number of write ops number of read ops And number of active connections The third signal is saturation, which measures how close to capacity a system is.

[01:52](javascript:;) It’s important to note, though, that capacity is often a subjective measure that depends on the underlying service or application. Saturation is important because: It's an indicator of how full the service is.

[02:05](javascript:;) It focuses on the most constrained resources. And it’s frequently tied to degrading performance as capacity is reached. Sample capacity metrics include: % memory utilization % thread pool utilization % cache utilization % disk utilization

[02:25](javascript:;) % CPU utilization Disk quota Memory quota number of of available connections And number of of users on the system The fourth signal is errors, which are events that measure system failures or other issues.

[02:44](javascript:;) Errors are often raised when a flaw, failure, or fault in a computer program or system causes it to produce incorrect or unexpected results, or behave in unintended ways. Errors are important because:

[02:57](javascript:;) They may indicate that something is failing. They may indicate configuration or capacity issues. They can indicate service level objective violations. And an error might mean it's time to send out an alert.

[03:11](javascript:;) Sample error metrics include: Wrong answers or incorrect content number of 400/500 HTTP codes number of failed requests number of exceptions number of stack traces Servers that fail liveness checks And number of dropped connections

UNDERSTANDING SLIs, SLOS and SLAs

Now let’s shift our focus to SLIs, SLOs and SLAs, which are all types of targets set for a system’s Four Golden Signal metrics. Service level indicators, or SLIs, are carefully selected monitoring metrics that measure one

[00:15](javascript:;) aspect of a service's reliability. Ideally, SLIs should have a close linear relationship with your users' experience of that reliability, and we recommend expressing them as the ratio of two numbers: the number of good events

[00:31](javascript:;) divided by the count of all valid events. A Service level objective, or SLO, combines a service level indicator with a target reliability. If you express your SLIs as is commonly recommended, your SLOs will generally be somewhere just

[00:47](javascript:;) short of 100%, for example, 99.9%, or "three nines." You can't measure everything, so when possible, you should choose SLOs that are S.M.A.R.T. SLOs should be specific. A question such as “Is the site fast enough for you?” is not specific; it's subjective.

[01:07](javascript:;) A statement such as “The 95th percentile of results are returned in under 100 milliseconds” is specific. SLOs need to be based on indicators that are measurable. A lot of monitoring is numbers, grouped over time, with math applied.

[01:22](javascript:;) An SLI must be a number or a delta; something we can measure and place in a mathematical equation. SLO goals should be achievable. "100% Availability" might sound good, but it's not possible to

[01:37](javascript:;) obtain, let alone maintain, over an extended window of time. SLOs should be relevant. Does it matter to the user? Will it help achieve application-related goals? If not, then it’s a poor metric.

[01:52](javascript:;) And SLOs should be time-bound. You want a service to be 99% available? That’s fine. Is that per year? Per month? Per day? Does the calculation look at specific windows of set time, from Sunday to Sunday for example,

[02:06](javascript:;) or is it a rolling period of the last seven days? If we don't know the answers to those types of questions, it can’t be measured accurately. And then there are Service Level Agreements, or SLAs, which are commitments made to your

[02:20](javascript:;) customers that your systems and applications will have only a certain amount of “down time.” An SLA describes the minimum levels of service that you promise to provide to your customers

[02:30](javascript:;) and what happens when you break that promise. If your service has paying customers, an SLA may include some way of compensating them with refunds or credits when that service has an outage that is longer

[02:42](javascript:;) than this agreement allows. To give you the opportunity to detect problems and take remedial action before your reputation is damaged, your alerting thresholds are often substantially higher than the minimum levels of service documented in your SLA.

[02:59](javascript:;) For SLOs, SLIs, and SLAs to help improve service reliability, all parts of the business must agree that they are an accurate measure of user experience and must also agree to use

[03:09](javascript:;) them as a primary driver for decision making. Being out of SLO must have concrete, well-documented consequences, just as there are consequences for breaching SLAs. For example, slowing down the rate of change and directing more engineering effort toward

[03:26](javascript:;) eliminating risks and improving reliability are actions that could be taken to get your product back to meeting its SLOs faster. Operations teams need strong executive support to enforce these consequences and effect change

[03:40](javascript:;) in your development practice.

INTEGRATED OBSERVABILITY TOOLS

Let’s wrap up this section of the course by taking a look at Google Cloud’s integrated monitoring, logging, error reporting, and debugging tools. If you've ever worked with on-premises environments, you know that you can physically touch the

[00:14](javascript:;) servers. If an application becomes unresponsive, someone can physically determine why that happened. In the cloud though, the servers aren't yours—they're Google’s—and you can’t physically inspect them. So the question becomes, how do you know what's happening with your server, or database, or

[00:32](javascript:;) application? The answer is by using Google’s integrated observability tools. Observability starts with signals, which are metric, logging, and trace data captured and integrated into Google products from the hardware layer up.

[00:49](javascript:;) From those products: The signal data flows into the Google Cloud operation's tools where it can be visualized in dashboards and through the Metrics Explorer. Automated and custom logs can be dissected and analyzed in the Logs Explorer.

[01:04](javascript:;) Services can be monitored for compliance with service level objectives (SLOs), and error budgets can be tracked. Health checks can be used to check uptime and latency for external-facing sites and

[01:15](javascript:;) services. And running applications can be debugged and profiled. When incidents occur: Signal data can generate automated alerts to code or, through various information channels, to key personnel. Error Reporting can help operations and developer teams spot, count, and analyze crashes in

[01:35](javascript:;) cloud-based services. Service Level Objectives should be adhered to. And the visualization and analysis tools can then help troubleshoot what's happening in Google Cloud. Ultimately, you won't miss that easy server access, because Google provides more precise

[01:52](javascript:;) insights into your Cloud install than you ever had on-premises. Over the next few videos we’ll explore the products and tools offered by Google Cloud that are most applicable for those in operations roles that work with monitoring, logging,

[02:07](javascript:;) error reporting, and debugging.

MONITORING TOOLS

When DevOps personnel want to track exactly what's happening inside Google Cloud projects, they often first think of monitoring. As we stated previously, monitoring starts with signal data. Metrics take measurements and use math to align those measurements over time.

[00:18](javascript:;) For example, it might be taking raw CPU usage measurement values and averaging them to produce a single value per minute. Google Cloud, by default, collects more than a thousand different streams of metric data,

[00:32](javascript:;) which can be incorporated into dashboards, alerts, and several other key tools. When data scientists run massive, scalable queries in BigQuery, it’s important for them to know how many queries are currently in flight, how many bytes have been scanned

[00:47](javascript:;) and added to the bill, and data slot usage patterns. It could also be critical to DevOps teams running containerized applications in Cloud Run to know CPU and memory utilization and app bill time.

[01:02](javascript:;) If those same DevOps teams want to augment the signal metrics from their custom application wherever it's running, they could use the open-source OpenTelementry and create their own metrics. And workloads on Compute Engine will benefit from CPU and memory utilization data, along

[01:19](javascript:;) with uptime, disk throughput, and many other metrics. Cloud Monitoring provides visibility into the performance, uptime, and overall health of cloud-powered applications. It collects metrics, events, and metadata from projects, logs, services, systems, agents,

[01:41](javascript:;) custom code, and various common application components, including Cassandra, Nginx, Apache Web Server, and Elasticsearch. Cloud Monitoring ingests that data and generates insights via dashboards, Metrics Explorer charts, and automated alerts.

[01:58](javascript:;) In this video we’ll take a look at Google Cloud’s integrated logging tools.

LOGGING TOOLS

In this video we’ll take a look at Google Cloud’s integrated logging tools. Cloud Logging allows users to collect, store, search, analyze, monitor, and alert on log entries and events.

[00:14](javascript:;) Automated logging is integrated into Google Cloud products like App Engine, Cloud Run, Compute Engine VMs running the logging agent, and GKE. Most log analysis starts with Google Cloud’s integrated Logs Explorer.

[00:30](javascript:;) Logging entries can also be exported to several destinations for alternative or further analysis. Pub/Sub messages can be analyzed in near-real time using custom code or stream processing technologies like Dataflow.

[00:45](javascript:;) BigQuery allows analysts to examine logging data through SQL queries. And archived log files in Cloud Storage can be analyzed with several tools and techniques. Log data can be exported as files to Cloud Storage,

[00:59](javascript:;) as messages through Pub/Sub, or into BigQuery tables. Log-based metrics can be created and integrated into Cloud Monitoring dashboards, alerts, and service SLOs. Default log retention in Cloud Logging depends on the log type.

[01:16](javascript:;) Data access logs are retained by default for 30 days, but this is configurable up to a maximum of 3,650 days. Admin logs are stored by default for 400 days.

[01:29](javascript:;) Logs can be exported to Cloud Storage or BigQuery to extend retention. The logs visible to you in Cloud Logging vary, depending on which Google Cloud resources you're using in your project or organization.

[01:42](javascript:;) Four key log categories are audit logs, agent logs, network logs, and service logs. Cloud Audit Logs helps answer the question, "Who did what, where, and when?" Admin activity tracks configuration changes.

[02:01](javascript:;) Data access tracks calls that read the configuration or metadata of resources and user-driven calls that create, modify, or read user-provided resource data. System events are non-human Google Cloud administrative actions that change the configuration of resources.

[02:21](javascript:;) And Access Transparency provides you with logs that capture the actions Google personnel take when accessing your content. Agent logs use a Google-customized and packaged Fluentd agent that can be installed on any

[02:34](javascript:;) AWS or Google Cloud VM to ingest log data from Google Cloud instances–for example, Compute Engine, Managed VMs, or Containers–and AWS EC2 instances. Network logs provide both network and security operations with in-depth network service telemetry.

[02:55](javascript:;) VPC Flow Logs records samples of VPC network flow and can be used for network monitoring, forensics, real-time security analysis, and expense optimization. Firewall Rules Logging allows you to audit, verify, and analyze the effects of your firewall

[03:14](javascript:;) rules. NAT Gateway logs capture information on NAT network connections and errors. Service logs provide access to logs created by developers deploying code to Google Cloud. For example, if they build a container using Node.js and deploy it to Cloud Run, any logging

[03:35](javascript:;) to Standard Out or Standard Error will automatically be sent to Cloud Logging for easy, centralized viewing. Let’s round off this section of the course by taking a look at the tools offered by Google

[03:42](javascript:;) Cloud for error reporting and debugging.

ERROR REPORTING AND DEBUGGING TOOLS

Let’s round off this section of the course by taking a look at the tools offered by Google Cloud for error reporting and debugging. Error Reporting counts, analyzes, and aggregates the crashes in your running cloud services.

[00:15](javascript:;) Crashes in most modern languages are “Exceptions,” which aren’t caught and handled by the code itself. Its management interface displays the results with sorting and filtering capabilities. A dedicated view shows the error details: time chart, occurrences, affected user count,

[00:34](javascript:;) first- and last-seen dates, and a cleaned exception stack trace. You can also create alerts to receive notifications on new errors. Cloud Trace, based on the tools Google uses on its production services, is a tracing system

[00:44](javascript:;) that collects latency data from your distributed applications and displays it in the Google Cloud console. Trace can capture traces from applications deployed on App Engine, Compute Engine VMs, and Google Kubernetes Engine containers.

[01:00](javascript:;) Performance insights are provided in near-real time, and Trace automatically analyzes all of your application's traces to generate in-depth latency reports to surface performance degradations. Trace continuously gathers and analyzes trace data to automatically identify recent changes

[01:18](javascript:;) to your application's performance. Poorly performing code increases the latency and cost of applications and web services every day, without anyone knowing or doing anything about it. Cloud Profiler changes this by using statistical techniques and extremely low-impact instrumentation

[01:37](javascript:;) that runs across all production application instances to provide a complete CPU and heap picture of an application without slowing it down. With broad platform support that includes Compute Engine VMs, App Engine, and Kubernetes,

[01:53](javascript:;) it allows developers to analyze applications running anywhere, including Google Cloud, other cloud platforms, or on-premises, with support for Java, Go, Python, and Node.js. Cloud Profiler presents the call hierarchy and resource consumption of the relevant function

[02:12](javascript:;) in an interactive flame graph that helps developers understand which paths consume the most resources and the different ways in which their code is actually called.

COURSE SUMMARY

Congratulations on completing the Google Cloud Core Infrastructure course! Before you go, let’s take a few minutes to review what we’ve covered. In module 1, you were introduced to Google Cloud and cloud computing.

[00:14](javascript:;) Specifically, you explored: The concept of managed infrastructure and managed services, through IaaS, or infrastructure as a service, and PaaS, or platform as a service. The Google Cloud network. Google Cloud’s focus on security throughout our infrastructure.

[00:30](javascript:;) How Google publishes key elements of technology using open source licenses. And Google Cloud’s pricing structure and billing tools. In module 2, you learned about the Google Cloud Resource Hierarchy, which is made up

[00:43](javascript:;) of four levels: resources, projects, folders, and an organization node. You also learned about: Defining policies and their downward inheritance. When to use Identity and Access Management, or IAM, And the four ways to access and interact with Google Cloud: through the Google Cloud console,

[01:03](javascript:;) the Cloud SDK and Cloud Shell, APIs, and the Google Cloud console Mobile App. In module 3, you explored how Compute Engine works, with a focus on virtual machines and

[01:14](javascript:;) virtual networking. You were introduced to: The VPC, or virtual private cloud. Compute Engine’s Autoscaling feature. And important Google Virtual Private Cloud compatibility features, like routing tables, firewalls, VPC peering and shared VPC, all of which result in the need for less network

[01:35](javascript:;) management. You also explored Cloud Load Balancing, a fully distributed, software-defined, managed service for all your traffic. Finally, you compared how on-premises or other-cloud networks can be interconnected with a Google

[01:49](javascript:;) VPC. In module 4, you explored Google Cloud's five core storage options: Cloud Storage, Cloud Bigtable, Cloud SQL, Cloud Spanner, and Firestore. You were also examined the four storage classes that make up Cloud Storage:

[02:07](javascript:;) \* Standard Storage, which is used for frequently accessed hot data, \* Nearline Storage and Coldline Storage, which are used for less-frequently accessed cool data, \* and Archive Storage. In module 5, you learned about containers, which are invisible boxes around your code

[02:23](javascript:;) and its dependencies. You were introduced to three container-based products: Kubernetes, an open-source platform for managing containerized workloads and services. Google Kubernetes Engine (GKE), a Google-hosted managed Kubernetes service in the cloud.

[02:40](javascript:;) And Anthos, Google’s answer to modern hybrid and multi-cloud distributed systems and services management. In module 6, the focus was on developing applications in the cloud. You explored: App Engine, a fully managed, serverless platform

[02:55](javascript:;) for developing and hosting web applications at scale, and the two of App Engine environments: standard and flexible. Three API management tools provided by Google Cloud: Cloud Endpoints, API Gateway, and Apigee

[03:07](javascript:;) API Management. And Cloud Run, a managed compute platform that lets you run stateless containers via web requests or Pub/Sub events. The focus for module 7 was developing and deploying in the cloud.

[03:21](javascript:;) You learned about: Cloud Source Repositories, which are full-featured Git repositories hosted on Google Cloud. Cloud Functions, a lightweight, event-based, asynchronous compute solution to create single-purpose functions. And Terraform, which lets you use a template to write the specifications for your application

[03:37](javascript:;) environment in the same way you’d write a configuration file. And in the final module, you focused on logging and monitoring on Google Cloud. The “Four Golden Signals” that measure a system’s performance and reliability:

[03:50](javascript:;) latency, traffic, saturation, and errors. Service level indicators (SLIs), service level objectives (SLOs), and service level agreements (SLAs), which are all types of targets set for a system’s Four Golden Signal metrics.

[04:06](javascript:;) And finally, Google’s integrated observability tools, which include Cloud Monitoring, Cloud Logging, Error Reporting, Cloud Trace, and Cloud Profiler. We hope that this course is just the beginning of your Google Cloud journey.

[04:21](javascript:;) For more training and hands-on practice, explore the different learning paths available at cloud.google.com/training. And if you’re interested in validating your expertise and showcasing your ability to transform businesses with Google Cloud technology, you might consider working toward a Google Cloud

[04:37](javascript:;) certification. You can learn more about Google Cloud’s certification offerings at cloud.google.com/certification. Thanks for completing this course. We’ll see you next time!