



Google Cloud

Maintenance and Monitoring

Learning objectives

- Manage new service versions using rolling updates, blue/green deployments, and canary releases.
- Forecast, monitor and optimize service cost using the Google Cloud pricing calculator and billing reports, and by analyzing billing data.
- Observe whether your services are meeting their SLOs using Cloud Monitoring and Dashboards.
- Use Uptime Checks to determine service availability.
- Respond to service outages using Cloud Monitoring Alerts.



Maintenance is primarily concerned with how updates are made to running applications, the different strategies available, and how different deployment platforms support them. For monitoring, this vital area for cloud-native applications is discussed from different perspectives. First from cost, to make sure that resources are being best provisioned against demand. Second, we discuss monitoring and observability to determine and alert on the health of services and applications.

Agenda

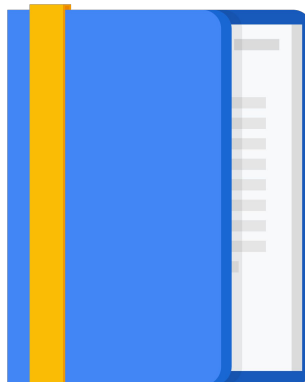
Managing Versions

Cost Planning

Monitoring Dashboards

Design Activity #13

Lab



In a microservice architecture, be careful not to break clients when services are updated

- Include version in URI:
 - If you deploy a breaking change, you need to change the version.
- Need to deploy new versions with zero downtime.
- Need to effectively test versions prior to going live.



A key benefit of a microservice architecture is the ability to independently deploy microservices. This means that the service API has to be protected. Versioning is required, and when new versions are deployed, care must be taken to ensure backward compatibility with the previous version.

The Google API design guidelines are a good reference (<https://cloud.google.com/apis/design/>). In particular, the link references guidelines on compatibility and what should be considered, backward-compatible changes, and incompatible changes.

Given the distributed nature of the cloud, there are a few options for updating application software, including:

- Rolling deployment
- Blue/green deployment
- Canary deployment

The next slides discuss these options.

Rolling updates allow you to deploy new versions with no downtime

- Typically, you have multiple instances of a service behind a load balancer.
 - Update each instance one at a time.
 - Rolling updates work when it is ok to have 2 different versions running simultaneously during the update.
- Rolling updates are a feature of instance groups; just change the instance template.
 - Rolling updates are the default in Kubernetes; just change the Docker image.
 - Completely automated in App Engine.



A rolling update incrementally updates the software on each running instance regardless of the deployment infrastructure. Consider a deployment using managed instance groups: The deployment is defined using an instance template. To perform a rolling update, update requests can be made using the managed group updater feature. This provides developer control over the speed of the deployment, the level of disruption acceptable, and also the scope of the update. This allows the rollout to be automated and also either a full or partial update. To perform the update, a new template is defined that describes the changes, and then a rolling update can be selected on the configuration. Various other parameters, such as the target size for updates, all instances, or a percentage, are configurable.

For more details, see

<https://cloud.google.com/compute/docs/instance-groups/rolling-out-updates-to-managed-instance-groups>.

With Kubernetes, a rolling update can be performed to update images, configuration, labels, annotations, and resource limits/requests. Rolling updates replace the resource pods in a way designed for zero downtime. Rolling updates can be triggered by updating pod templates for:

- DaemonSets
- Deployments
- StatefulSets

For more details, see

<https://cloud.google.com/kubernetes-engine/docs/how-to/updating-apps>.

With App Engine, rolling updates are automatic. The new software versions are deployed, and App Engine seamlessly manages the rolling update with no disruption to clients.

Use a blue/green deployment when you don't want multiple versions of a service running simultaneously

- The blue deployment is the current version.
 - Create an entirely new environment (the green).
 - Once the green deployment is tested, migrate client requests to it.
 - If failures occur, switch it back.
- In Compute Engine, you can use DNS to migrate requests from one load balancer to another.
 - In Kubernetes, configure your service to route to the new pods using labels.
 - Simple configuration change
 - In App Engine, use the Traffic Splitting feature.



Blue/green deployments use two full deployment environments.

One, let's call it *blue*, is running the current deployed production software. The second deployment environment, here named green, is available for deploying updated versions of the software. The new software version is deployed to the green environment and tested. When testing is complete, the workload is shifted from the current (blue) to the new (green) environment. Such a strategy mitigates the risk of a bad deployment by allowing the switch back to a previous deployment if something goes wrong.

For Compute Engine, switching deployments requires configuring a duplicate environment with load balancer and then configuring Cloud DNS to switch from one environment to another. Kubernetes requires the second deployment to be in place and tested, and then when everything is ready, traffic is switched to the green deployment by changing the configuration of the service files and making the selector configuration select the new version.

For App Engine, the traffic splitting feature can be used. The deployment can be deployed and tested, and then the traffic splitting feature switches traffic over to the new deployment when ready.

Canary releases can be used prior to a rolling update to reduce the risk

- The current service version continues to run.
 - Deploy an instance of the new version and give it a portion of requests.
 - Monitor for errors.
- In Compute Engine, you can create a new instance group and add it as an additional backend in your load balancer.
 - In Kubernetes, create a new pod with the same labels as the existing pods; the service will automatically route a portion of requests to it.
 - In App Engine, use the Traffic Splitting feature.



With a canary release, a new deployment is made with the current deployment still running. A small percentage of traffic is sent to the new deployment and it is monitored. Once confidence is built in the new deployment and it handles live traffic well, more traffic is routed to the new deployment until 100% is routed this way. When comparing the new deployment, it is important to compare it against a baseline and not the current production deployment.

With Compute Engine, a simple way to achieve a canary release is to add an additional backend to the load balancer. For Kubernetes, it is possible to create a new pod with the same labels as existing pods so that traffic will be routed to the new pod. This is a weak strategy because the traffic is routed based on replica ratio, so to receive a small amount of traffic for the new deployment, the number of replicas would typically need to be high (100 replicas for 1% traffic to the new version). A service mesh, such as Istio, solves this. Traffic routing and replica deployment are independent, meaning pods can scale up and down based on traffic load but independent of traffic routing. Routing rules allow the percentage of traffic sent to each route to be defined, and thus provide string support for canary releases.

As with blue/green deployments, for App Engine the traffic splitting feature can be used for canary releases.

Agenda

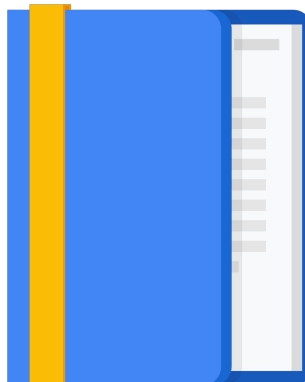
Managing Versions

Cost Planning

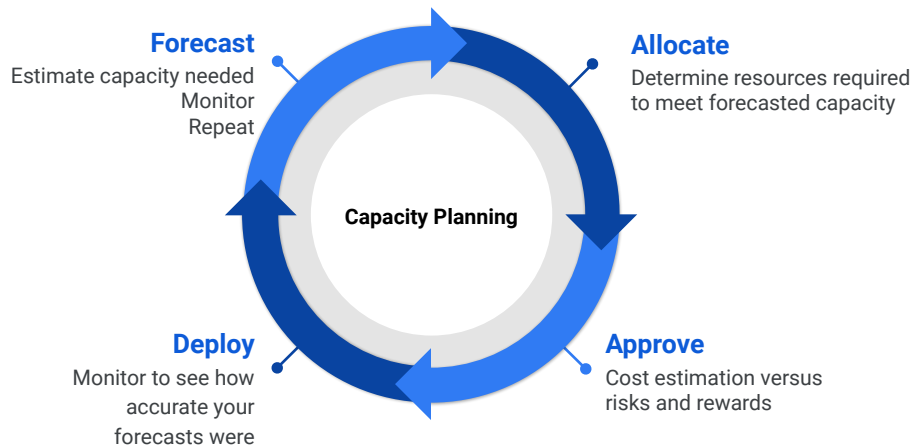
Monitoring Dashboards

Design Activity #13

Lab



Capacity planning is a continuous, iterative cycle



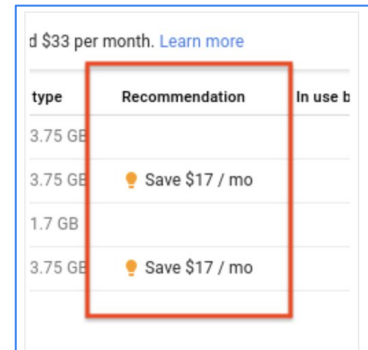
Capacity planning is part of an overall cost management strategy. This typically includes the following areas:

- Resource planning: this is a balance of estimate capacity and the resources that are required to deliver that capacity.
- Cost estimating: based on the resources, what are the estimated costs, and are these viable?
- Cost budgeting: how can we best make use of budget?
- Cost control: keeping track of budgets and making sure they are in line with predictions and estimates.

The next few slides talk about how costs can be estimated and how some technical decisions can help reduce costs.

Optimizing cost of compute

- Start with small VMs, and test to see whether they work.
- Consider more small machines with auto scaling turned on.
- Consider committed use discounts.
- Consider at least some preemptible instances:
 - 80% discount
 - Use auto healing to recreate VMs when they are preempted.
- Google Cloud rightsizing recommendations will alert you when VMs are underutilized.



d \$33 per month. Learn more		
type	Recommendation	In use b
3.75 GB		
3.75 GB	💡 Save \$17 / mo	
1.7 GB		
3.75 GB	💡 Save \$17 / mo	



A good starting point for anybody working with cost optimization is to become familiar with the VM instance pricing. One key point to be aware of is the billing model, which means all vCPUs, GPUs, and GB of memory are charged for a minimum of 1 minute. After that, instances are charged in increments of 1 second. Then the pricing is resource based—each vCPU and GB is billed separately. This is where it is important to be aware of the discounts available:

- **Sustained use**, which are automatic discounts.
- **Committed use** discounts, which are most suitable for workloads with predictable resource needs and require a commitment of between 1 and 3 years. The discount is up to 57% for most resources, but can be up to 70% for memory-optimized machines.
- **Preemptible use** where Compute Engine may terminate these instances at a tradeoff of much lower price. Preemptible instances are Compute Engine's excess capacity so have variable availability.

For more details on the pricing model, see <https://cloud.google.com/compute/vm-instance-pricing>.

Google Cloud has a feature known as *rightsizing* that is very useful. This feature provides recommendations on the correct size of your VM instances to more efficiently use instance resources. It uses Cloud Monitoring to gather system metrics and makes recommendations based on the previous 8 days' usage. More details on rightsizing are at

<https://cloud.google.com/compute/docs/instances/apply-sizing-recommendations-for-instances>.

Optimizing disk cost

- Don't over-allocate disk space.
- Determine what performance characteristics your applications require:
 - I/O Pattern: small reads and writes or large reads and writes
 - Configure your instances to optimize storage performance.
- Depending on I/O requirements, consider Standard over SSD disks.

Monthly capacity	Standard PD	SSD PD
10 GB	\$0.40	\$1.70
1 TB	\$40	\$170
16 TB	\$655.36	\$5,570.56



There are several different types of block storage instances you can use. Each type has different price, performance, and durability characteristics. The types include:

- **Standard persistent disks:** suited for large data processing workloads as a cost-effective solution
- **SSD persistent disks:** suited for enterprise applications and high-performance database needs
- **Local SSDs:** only exists for lifetime of an instance but provides performance and lower latency

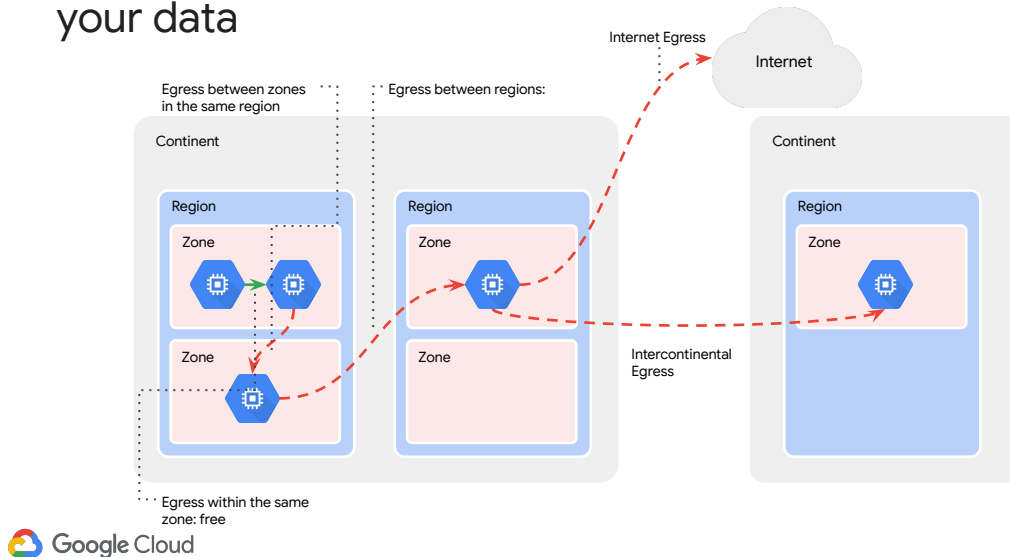
There are zonal and regional versions of the standard and SSD disks. Regional disks have higher costs. For details on the performance of the different types, see https://cloud.google.com/compute/docs/disks/performance#type_comparison.

It is important to note that when selecting a volume type, persistent disk has no per-I/O cost. So monthly I/O does not affect cost but will potentially affect performance, because each disk type has a limit of IOPS per GB. So for IOPS-oriented workloads, it is good practice to compare prices per IOPS/GB. In such cases, SSD may be more cost effective than standard-based storage because standard disk storage is cheaper per GB but the IOPS are much lower.

Also consider the factors that may affect performance. For example, with standard persistent disks, the throughput performance increases linearly with the size of disk up to a per instance limit. For details on these metrics, see

[https://cloud.google.com/compute/docs/disks/performance#price_performance.](https://cloud.google.com/compute/docs/disks/performance#price_performance)

To optimize network costs, keep machines close to your data



At a high level, it is important to start by considering the network tiers available:

Premium: provides exceptionally high performance using Google's global network
Standard: reduced performance but with cost benefits

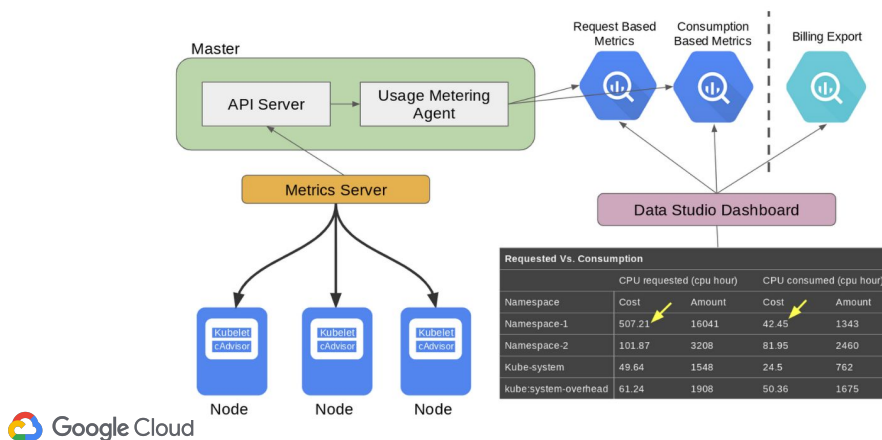
The high-level considerations are that premium is for when high performance, reliability, a global footprint, and the best possible user experience are priorities. Standard is where cost is a major consideration, and as an architect you are trading this off for performance. There are some technical differences to consider also, such as Cloud Load Balancing only being regional for standard. For full details on the differences, see <https://cloud.google.com/network-tiers/docs/>.

Once a decision has been made, it is important to understand the network pricing model. Here the use of internal or external IP addresses is a major consideration. Using external IP addresses, the egress rates are higher. The location of the resources also has a potential impact on cost. The unit of charge for networking is the gigabyte.

The diagram above provides a high-level overview of egress combinations, but for a full picture, see the documentation on combinations and pricing here: <https://cloud.google.com/compute/network-pricing>

GKE usage metering can prevent over-provisioning Kubernetes clusters

Compares requested resources with consumed resources.



GKE usage metering provides fine-grained visibility to Kubernetes clusters. With it, you can see your GKE clusters' resource usage broken down by namespaces and labels, and attribute it to meaningful entities (for example, department, customer, application, or environment). Functionality includes the ability to help reduce waste from over-provisioning, with the addition of consumption-based metrics that allow you to compare and contrast the resource requests with actual utilization.

There are three kinds of resource metrics in Kubernetes: resource requests, resource limits, and resource consumption. A resource request is a lower bound on the amount of resources a container will receive, a resource limit is an upper bound, and resource consumption is the actual amount used by a container at runtime.

A resource request is the primary driver of cost, because the Kubernetes scheduler uses the resource request of a new pod, compared to the capacity of a node and the resource requests of the pods running on the node, to know whether a new pod will fit on a node. And if the pod doesn't fit on any of the available nodes, the cluster autoscaler (if enabled) adds nodes to the cluster. If a pod uses fewer resources than it requests, the system is needlessly reserving resources.

With GKE usage metering, an agent collects consumption metrics in addition to the resource requests, by polling pod metrics objects from the metrics server. The resource request records and resource consumption records are exported to two separate tables in a BigQuery dataset that you specify. Comparing requested with consumed resources makes it easy to spot waste and take corrective measures.

Compare the costs of different storage alternatives before deciding which one to use

Choose a storage service that meets your capacity requirements at a reasonable cost:

- Storing 1GB in Firestore is free.
- Storing 1GB in Cloud Bigtable would be around \$1400/month.



Comparing costs of the different storage options is often the most difficult decision. The different storage options discussed earlier in the course highlighted that each storage option is a solution for a particular type of storage requirement. So while cost is clearly important, it is often the technical requirement that is a major influencer on the type of storage to be used. After that, price also clearly plays a big role, and selecting the wrong storage type can have significant cost implications, as illustrated in the slide above.

There is the free tier (<https://cloud.google.com/free/>) that can have an impact on the choice of storage too. Estimating the cost afterwards can be the raw storage and also the network usage, the operations usage, and if, for example Cloud Storage, the retrieval and early deletion fees. The pricing calculator and the individual storage pricing pages can help:

Cloud Storage: <https://cloud.google.com/storage/pricing>

Cloud SQL: <https://cloud.google.com/sql/pricing>

Firestore: <https://firebase.google.com/docs/firestore/pricing>

Cloud Bigtable: <https://cloud.google.com/bigtable/pricing>

Spanner: <https://cloud.google.com/spanner/pricing>

Consider alternative services to save cost rather than allocating more resources

- CDN
- Caching
- Messaging
- Queueing
- Etc.



Architectural design can help with significant savings on cost. For example, using Cloud CDN for static content or Memorystore as a cache can lead to significant savings. Messaging/queueing with Pub/Sub can be used to decouple communicating services and also reduce storage needs, as a datastore is sometimes used as the communication medium in poorly architected applications.

Use the Google Cloud Pricing Calculator to estimate costs

- Base your cost estimates on your forecasting and capacity planning.
- Compare the costs of different compute and storage services.

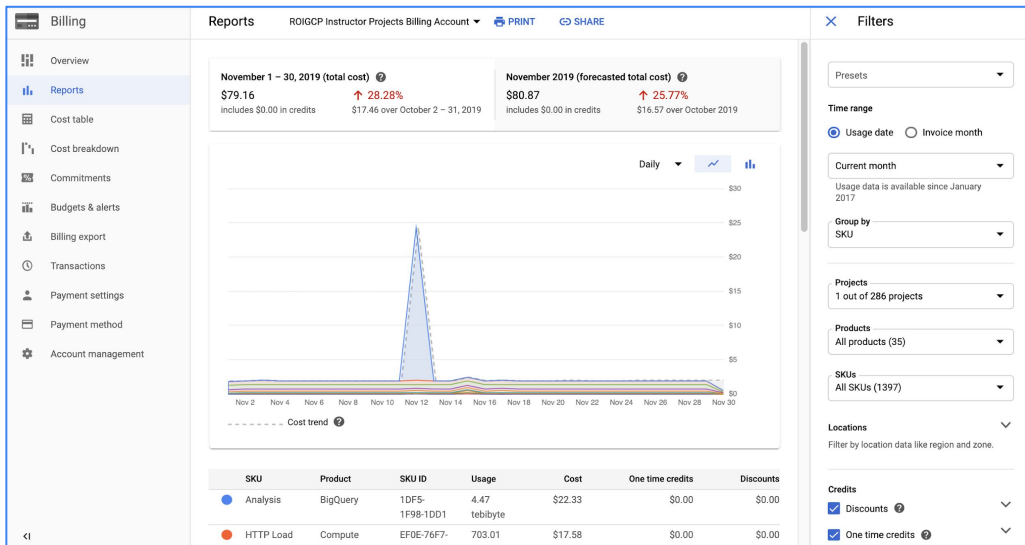
The screenshot displays the Google Cloud Platform Pricing Calculator interface. At the top, a navigation bar includes icons for various services: Cloud Firestore, Cloud DataProc, Cloud Dataflow, Cloud SQL (selected), Cloud IoT Core, Cloud Pub/Sub, and GKE. Below this is a search bar with the text "Search for a product you are interested in." The main content area is divided into two sections. The left section, titled "Google Cloud SQL", has tabs for "FIRST GENERATION", "SECOND GENERATION", and "POSTGRES SQL" (which is active). Under the "POSTGRES SQL" tab, there are input fields for "Number of instances" (set to 1), "SQL Instance Type" (set to db-pg-11-micro), a checkbox for "Enable High Availability Configuration", "Location" (set to Iowa (us-central1)), and "Storage (Provisioned Amount)" (set to SSD). The right section, titled "Estimate", shows the configuration details for "Cloud SQL for Postgres": "CP-DB-PG-CUSTOM-4-16", "# of instances: 1", "Location: Iowa", "730.0 total hours per month", "SSD Storage: 1,024.0 GB", "Backup: 1,024.0 GB", and a "Sustained Use Discount" of 30%. The "Total Estimated Cost" is displayed as "USD 834.79 per 1 month". At the bottom of the right section, there are buttons for "EMAIL ESTIMATE" and "SAVE ESTIMATE".

<https://cloud.google.com/products/calculator>



The pricing calculator is the go-to resource for gaining cost estimates. Remember that the costs are just an estimate, and actual cost may be higher or lower. The estimates by default use the timeframe of one month. If any inputs vary from this, they will state this. For example, Firestore document operations read, write, and delete are asked for on a per day basis.

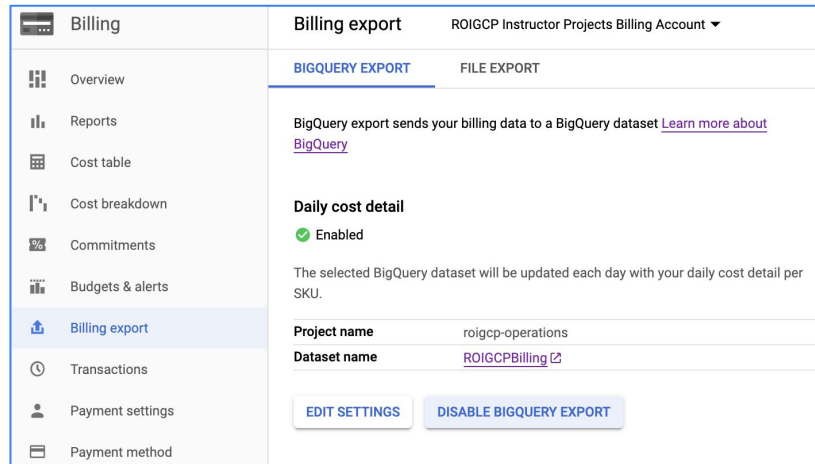
Billing reports provide detailed cost breakdowns



The Cloud Billing Reports page provides an at-a-glance summary of usage costs and trends. Multiple filters enable searching of the data particular to projects, products, and region. Sizing recommendations for compute based on usage trends will be presented here. If you are using GKE, GKE usage metering is a useful utility to help prevent a common Kubernetes symptom of overprovisioning. The data can be joined with Google Cloud billing data to estimate cost breakdowns by namespace and labels. More details can be found here:

<https://cloud.google.com/blog/products/containers-kubernetes/use-gke-usage-metering-to-combat-over-provisioning>.

For advanced cost analysis, export billing data to BigQuery



It is possible to export Cloud Billing data to BigQuery. This then allows detailed analysis of billing data. The data can be exported automatically. The data can then be visualized in Google Data Studio. For more details, see <https://cloud.google.com/billing/docs/how-to/visualize-data>.

Visualize spend with Google Data Studio



Data Studio



Billing Dashboard



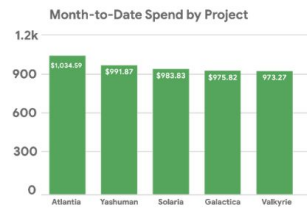
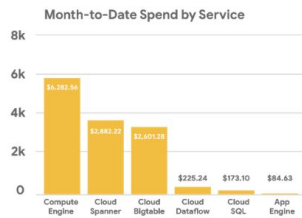
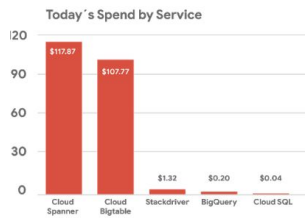
Daily View



Monthly View



Overall



Google Data Studio is a fully managed visual analytics service that can help anyone in your organization unlock insights from data through easy-to-create and interactive dashboards that inspire smarter business decision-making. When Data Studio is combined with BigQuery BI Engine, an in-memory analysis service, data exploration and visual interactivity reach sub-second speeds over massive datasets.

Set budgets and alerts to keep your team aware of how much they are spending

	Percent of budget	Amount	Trigger on
Name * Budget Name	50 %	\$ 250	Actual
Projects All projects (2)	90 %	\$ 450	Actual
	100 %	\$ 500	Actual

2 Amount

Budget type
Specified amount

A fixed amount that your spend will be compared against.

Target amount
\$ 500

Specified amount

Last month's spend

Programmatic Budgets: Cloud Pub/Sub → Cloud Functions



It is possible to create budgets to monitor Google Cloud charges. After a budget amount has been set, budget alert rules that are used to trigger notifications can be defined, so users are informed about how spend is tracking against their budgets.

You can apply budget alerts to a Cloud Billing Account, to one or more projects, and/or one or more products. You can set the budget to an amount you specify or match it to the previous month's spend. When costs (actual costs or forecasted costs) exceed a percentage of your budget (based on the rules you set), alert emails are sent to Billing Account Administrators and Billing Account Users on the target Cloud Billing Account.

Budgets do not cap service usage. They are just used to define an alert whose aim is to prompt the user to take action. The budget and alert will not prevent the service use or billing of services over the budget.

Agenda

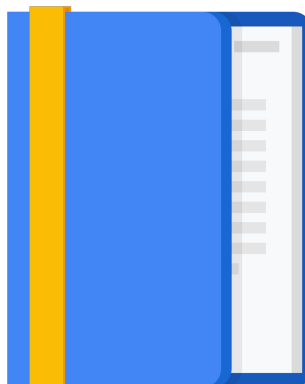
Managing Versions

Cost Planning

Monitoring Dashboards

Design Activity #13

Lab



Google Cloud unifies the tools you need to monitor your service SLOs and SLAs



It is vital when architecting systems to consider reliability. It is very difficult to provide reliable services without insights into how the software and the infrastructure is functioning. Google has published, in the SRE book, the four golden signals (<https://landing.google.com/sre/books/>), and similar metrics to monitor USE and RED for services and infrastructure have been suggested elsewhere.

The aim of monitoring is to be able to reduce the MTTR, which means being able to observe and identify when things are not working as expected.

Google Cloud offers a set of services for collecting data on services and infrastructure. At a high level, these services can be considered as:

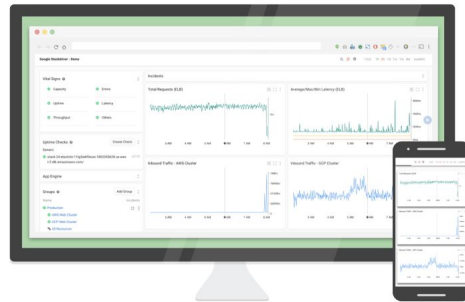
- **Monitoring:** Collecting core metrics at regular intervals.
- **Logging:** Messages written when an event occurs. Linked to logging is an error reporting service that provides a client library for reporting errors that can then be analyzed in the monitoring dashboard.
- **Alerting:** Notifications are sent when certain conditions are met from log data or metrics; e.g., CPU utilization has exceeded a certain threshold.

In addition to these services, Google Cloud also provides a set of services for application performance management. Debugger allows code to be debugged in production without slowing that code down. Trace measures latency for requests made in the application's distributed calls. This is extremely useful for determining where problems or bottlenecks are occurring in the application. The Profiler can be used to determine the CPU and memory usage of applications, which can provide

insights into resources required or identify areas that require performance optimization. Profiling is also useful for predicting when future problems may arise.

Monitoring dashboards monitor your services

- Monitor the things you pay for:
 - CPU use
 - Storage capacity
 - Reads and writes
 - Network egress
 - Etc.
- Monitor your SLIs to determine whether you are meeting your SLOs.



Cloud Monitoring makes it easy to monitor SLIs as well as the items that are chargeable. The SRE four golden signals often cover the above, with saturation linked to the CPU utilization, memory, read writes, etc., and SLIs for services often linked to latency, traffic, and errors. Alerts can be set to bring immediate attention to problems.

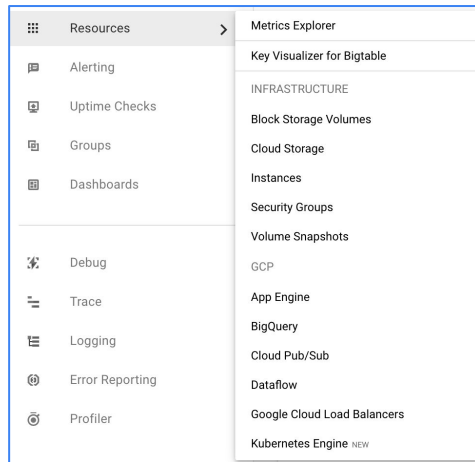
Example charts in a Monitoring dashboard



Here is an example of some charts in a Monitoring dashboard. On the left you can see the CPU usage for different Compute Engine instances, and on the right is the ingress traffic for those instances.

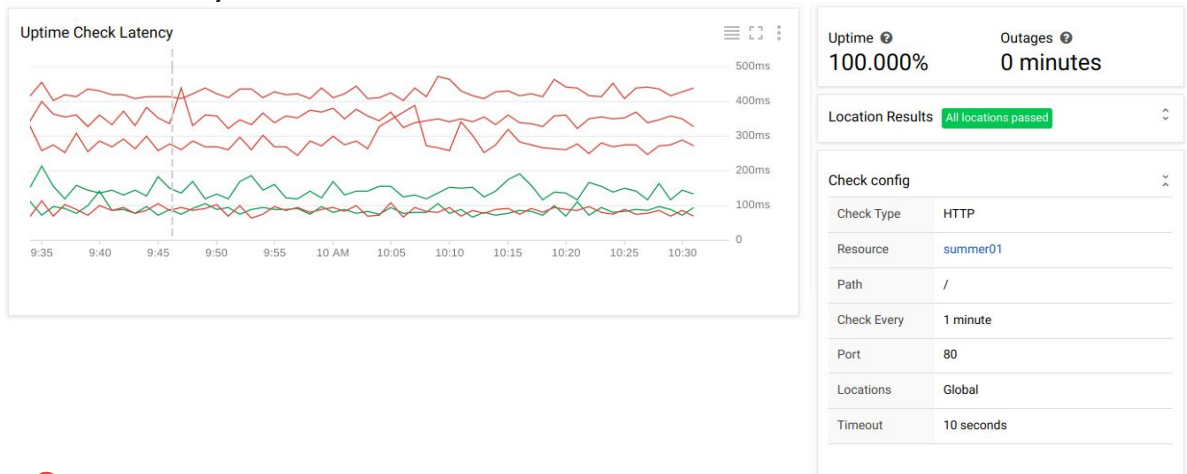
Charts like these provide valuable insights into usage patterns.

To help you get started, Cloud Monitoring creates default dashboards for your project resources



Cloud Monitoring provides predefined dashboards that require no setup. Custom dashboards can also be defined.

Create uptime checks to monitor availability and latency



These again link to SLIs and SLOs. It is best practice to monitor these and determine trends.

Create alerts when your service fails to meet your SLOs

Create new alerting policy

1 Conditions

Basic Conditions

HTTP check on instance summer01

Violates when: Uptime Check Health on Instance (GCE) summer01 fails

[Edit](#) [Delete](#)

+ Add Another Condition

2 Notifications (optional)

When alerting policy violations occur, you will be notified via these channels. [Learn more](#)

Email

demo@example.com

×

+ Add Another Notification

3 Documentation (optional)

When email notifications are sent, they'll include any text entered here. This can convey useful information about the problem and ways to approach fixing it.

[Edit](#) [Preview](#) [Markdown Formatting Help](#)

 Main Server health check failed
+ Server named summer01 failed a Stackdriver uptime check
+ IP Address of the server is: 104.197.58.79

4 Name this policy

A policy's name is used in identifying which policies were triggered, as well as managing configurations of different policies.

Uptime Check Policy

[Save Policy](#) [Cancel](#)



Your SLO will be more strict than your SLA, so it is important to be alerted when you are not meeting an SLO because its an early warning that the SLA is under threat.

Here is an example of what creating an alerting policy looks like. On the left, you can see an HTTP check condition on the summer01 instance. This will send an email that is customized with the content of the documentation section on the right.

Activity 13: Cost estimating and planning

Refer to your Design and Process Workbook.

- Use the price calculator to create an initial estimate for deploying your case study application.



Lab

Monitoring Applications in Google Cloud



Monitoring



Logging



Trace



Profiler

Objectives

- Examine the Cloud Logs.
- View Profiler Information.
- Explore Cloud Trace.
- Monitor Resources using Dashboards.
- Create Uptime Checks and Alerts.

Quiz

You've made a minor fix to one of your services. You want to deploy the new version with no downtime. Which would you choose?

- A. Rolling update
- B. Blue/green deployment
- C. Canary deployment
- D. A/B test



You've made a minor fix to one of your services. You want to deploy the new version with no downtime. Which would you choose?

- A. Rolling update
- B. Blue/green deployment
- C. Canary deployment
- D. A/B test

Quiz

You've made a minor fix to one of your services. You want to deploy the new version with no downtime. Which would you choose?

- A. Rolling update
- B. Blue/green deployment
- C. Canary deployment
- D. A/B test



- A. This answer is correct. A rolling update will update instances incrementally until all have been updated. This strategy can be controlled, for example, by using a managed instance group with Compute Engine or with GKE.
- B. This is not correct. Blue/green deployments require two full deployments. The requirement in the question is to update a particular service, not the entire application.
- C. This is not correct. This is a possible solution for a service that aims to eliminate/reduce risks by applying the update to a small subset of users to test out a new feature. Because the requirement is a minor fix, a rolling update is preferred.
- D. This is not correct. A/B testing is usually used to compare the behavior/usage patterns of two versions of an application or web page against each other.

Quiz

You made a minor update to a service and would like to test it in production by sending a small portion of requests to the new version. Which would you choose?

- A. Rolling update
- B. Blue/green deployment
- C. Canary deployment
- D. A/B test



You made a minor update to a service and would like to test it in production by sending a small portion of requests to the new version. Which would you choose?

- A. Rolling update
- B. Blue/green deployment
- C. Canary deployment
- D. A/B test

Quiz

You made a minor update to a service and would like to test it in production by sending a small portion of requests to the new version. Which would you choose?

- A. Rolling update
- B. Blue/green deployment
- C. Canary deployment
- D. A/B test



- A. This is not correct. A rolling update will update instances incrementally until all have been updated, which is not what the requirement asks for (small portion of requests).
- B. This is not correct. Blue/green deployments require two full deployments. The requirement in the question is to send a small portion of requests to an updated service.
- C. This answer is correct. Canary deployment aims to eliminate/reduce risks by applying the update to a small subset of users to test out a new feature and best fits the requirements of the question.
- D. This is not correct. A/B testing is usually used to compare the behavior/usage patterns of two versions of an application or web page against each other.

Quiz

You're deploying test environments using Compute Engine VMs. Some downtime is acceptable, and it is very important to deploy them as inexpensively as possible. What single thing below could save you the most money?

- A. Committed use discount
- B. Sustained use discount
- C. Sole tenant nodes
- D. Preemptible machines



You're deploying test environments using Compute Engine VMs. Some downtime is acceptable, and it is very important to deploy them as inexpensively as possible. What single thing below could save you the most money?

- A. Committed use discount
- B. Sustained use discount
- C. Sole tenant nodes
- D. Preemptible machines

Quiz

You're deploying test environments using Compute Engine VMs. Some downtime is acceptable, and it is very important to deploy them as inexpensively as possible. What single thing below could save you the most money?

- A. Committed use discount
- B. Sustained use discount
- C. Sole tenant nodes
- D. Preemptible machines



- A. This is not correct. Committed use requires a commitment for 1 or 3 years. Because this requirement is for test environments and some downtime is acceptable, this is not the most cost-effective solution.
- B. This is not correct. Sustained use discounts are automatically applied over a certain threshold. Because this requirement is for test environments, it is unlikely that the thresholds will be met per month for discounts, and therefore it is not the most cost-effective solution.
- C. This is not correct. Sole tenant nodes are not the most cost-effective solution for this use case.
- D. This answer is correct. These are the most cost-effective solution as required in the question (up to 80% lower than equivalent non-preemptible machines) and the fact that some downtime is acceptable means that the requirement can be met with preemptible machines.

Quiz

Your service has an availability SLO of 99%. What could you use to monitor whether you are meeting it?

- A. Health check
- B. Uptime check
- C. Liveness probe
- D. Readiness probe



Your service has an availability SLO of 99%. What could you use to monitor whether you are meeting it?

- A. Health check
- B. Uptime check
- C. Liveness probe
- D. Readiness probe

Quiz

Your service has an availability SLO of 99%. What could you use to monitor whether you are meeting it?

A. Health check

B. Uptime check

C. Liveness probe

D. Readiness probe



B. This answer is correct. Availability is the percentage of time a system is running and able to process requests. Monitoring this metric will enable the derivation of the SLO metric.

The health check is not correct because this determines whether the application is available at a particular moment in time, not how long it has been running. Liveness and Readiness are probes that enable the detection of a service being alive or not (liveness) and ready to serve traffic (readiness). Both are types of health checks and will not help monitoring the availability SLO.

Review

Maintenance and Monitoring



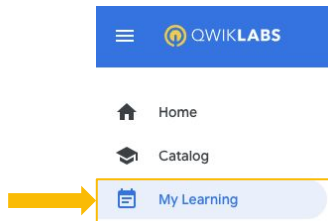
In this module you learned about managing new versions of your microservices using rolling updates, canary deployments, and blue/green deployments. It's important when deploying microservices that you deploy new versions with no downtime, but also that the new versions don't break the clients that use your services.

You also learned about cost planning and optimization, and you estimated the cost of running your case study application.

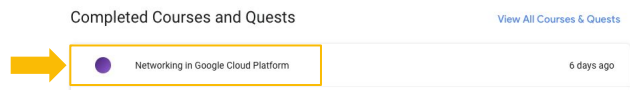
You finished the module by learning how to leverage the monitoring tools provided by Google Cloud. These tools can be invaluable for managing your services and monitoring your SLIs and SLOs.

End of class - Materials

- 1 Click on **My Learning** in the left-hand navigation bar



- 2 Select the class from the **Completed Courses** list



Materials are available for 2 years



You can view the course materials within Qwiklabs as follows:

1. Click on **My Learning** in the left-hand navigation bar.
2. Select the class from the **Completed Courses** list.

Materials are available for 2 years following the completion of a course.

Professional Cloud Architect Certification



Speaking of certification, we recommend to validate your hands-on Google Cloud skills and advance your career with the **Professional Cloud Architect** certification. Certification can help you gain credibility and give you an advantage in today's highly competitive market. A Professional Cloud Architect enables organizations to leverage Google Cloud technologies. With a thorough understanding of cloud architecture and Google Cloud Platform, this individual can design, develop, and manage robust, secure, scalable, highly available, and dynamic solutions to drive business objectives.

The Google Cloud Certified - Professional Cloud Architect exam assesses your ability to::

- Design and plan a cloud solution architecture
- Manage and provision the cloud solution infrastructure
- Design for security and compliance
- Analyze and optimize technical and business processes
- Manage implementations of cloud architecture
- Ensure solution and operations reliability

You can also prepare using the [Official Google Cloud Certified Professional Cloud Architect Study Guide](#), published by Wiley.

More resources

Monitoring

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

Cost Management

<https://cloud.google.com/cost-management/>



The links provide access to some useful resources on cost management and Monitoring.

