## Reliable Google Cloud Infrastructure: Design and Process

* build highly reliable and efficient solutions on GC using proven design patterns. It is a continuation of the Architecting with Google Compute Engine or Architecting with GKE courses and assumes hands-on experience with the technologies covered in either of those courses. Through a combination of presentations, design activities, and hands-on labs, participants learn to define and balance business and technical requirements to design GC deployments that are highly reliable, highly available, secure, and cost-effective.
* Introduction
  + Activity Intro: Defining your case study 1 minute - https://youtu.be/ECrXpwd9A78
  + Activity Review: Defining your case study 1 minute - https://youtu.be/xGjcy0UPtzo
* Defining Services
  + In this module, you will learn to describe users of a system in terms of the roles and personas they take. You will learn how to measure success using Key performance indicators (KPIs) and you will examine service level objectives (SLOs), service level indicators (SLIs), and service level agreements (SLAs).
  + Requirements, Analysis, and Design 6 minutes - https://youtu.be/Z5bk6O0qSGc
  + Activity Intro: Analyzing your case study 2 minutes - https://youtu.be/tPoirArLKNY
  + Activity Review: Analyzing your case study 1 minute - https://youtu.be/TxgqQiqAOsQ
  + KPIs and SLIs 7 minutes - https://youtu.be/jz5ubYtyVjs
  + SLOs and SLAs 5 minutes - https://youtu.be/rz8ij6DlTy0
  + Activity Intro: Defining SLIs and SLOs 1 minute - https://youtu.be/rz8ij6DlTy0
  + Activity Review: Defining SLIs and SLOs 1 minute - https://youtu.be/m3TbjR0Fc\_U
  + Quiz - Defining Services - https://www.cloudskillsboost.google/course\_sessions/816692/quizzes/114433
    - Which best describes a user story?
      * It is a short description of a feature written from the user's point of view.
    - Which best describes an SLO?
      * It is a target reliability you want your service to achieve.
      * Correct, an SLO is the agreed-upon target for a measurement or range of values for a service. Reliability could be one of these.
    - Using SMART criteria, which below would be the least effective KPI?
      * User experience design
      * Correct, user experience design is not measurable or time bound and so would not make a relevant KPI
* Microservice Design and Architecture
  + In this module, we introduce application architecture and microservice design.
  + Microservices 6 minutes - https://youtu.be/5ZMu9ejLOSY
  + Microservices Best Practices 5 minutes - https://youtu.be/ypGXNyO7vIc
  + Activity Intro: Designing microservices for your application 1 minute - https://youtu.be/ypGXNyO7vIc
  + Activity Review: Designing microservices for your application 1 minute - https://youtu.be/Ikbj46EtWAQ
  + REST 4 minutes - https://youtu.be/lbD7KMg4upA
  + HTTP 4 minutes - https://youtu.be/9CuoC\_XXKV0
  + APIs 3 minutes - https://youtu.be/8uzBNHDvm6k
  + Activity Intro: Designing REST APIs 1 minute - https://youtu.be/0RnQnHEKN8k
  + Activity Review: Designing REST APIs 1 minute - https://youtu.be/XISsSnGlERE
  + Quiz - Microservice Design and Architecture - https://www.cloudskillsboost.google/course\_sessions/816692/quizzes/114445
    - Which below would violate 12-factor app best practices?
      * Store configuration information in your source repository for easy versioning.
      * Correct. Code and config should be separated, because config varies across deployments but code does not. The true test is whether the repository could be open-sourced without compromising any credentials.
    - You’ve re-architected a monolithic web application so state is not stored in memory on the web servers, but in a database instead. This has caused slow performance when retrieving user sessions though. What might be the best way to fix this?
      * Use a caching service like Redisor Memorystore.
      * Correct. Services should be stateless,and a service like Redisor Memory store provides a fast caching service to store state.They enable services to be stateless and support scale and high availability.
    - You’re writing a service, and you need to handle a client sending you invalid data in the request. What should you return from the service?
      * A 400 error code
      * Correct. 400 is a HTTP status code indicating that a request could not be processed due to an apparent client error.
    - You’re building a RESTful microservice. Which would be a valid data format for returning data to the client?
      * All of the options.
      * Correct.They have a standard Content-Type that can be set on the response header and are text-based. It is usual to use JSON, but both XML and JSON are valid.
* DevOps Automation
  + This module introduces DevOps automation, a key factor in achieving consistency, reliability, and speed of deployment.
  + Continuous Integration Pipelines 7 minutes - https://youtu.be/H3Sx1ULJuGo
  + Infrastructure as Code 5 minutes - https://youtu.be/th38i86N4f0
  + Building a DevOps Pipeline 1 hour - <https://www.cloudskillsboost.google/course_sessions/816692/labs/114451>
    - Overview
      * In this lab, you will build a continuous integration pipeline using Cloud Source Repositories, Cloud Build, build triggers, and Container Registry.
      * Diagram

        Description automatically generated
    - Objectives
      * Create a Git repository
      * Create a simple Python application
      * Test Your web application in Cloud Shell
      * Define a Docker build
      * Manage Docker images with Cloud Build and Container Registry
      * Automate builds with triggers
      * Test your build changes
    - Task 1: Create a Git Repository
      * create a Git repository using the Cloud Source Repositories service in Google Cloud. This Git repository will be used to store your source code. Eventually, you will create a build trigger that starts a continuous integration pipeline when code is pushed to it.
      * Navigation > Source Repositories > Click Add repository.
        + Name the repository devops-repo
        + Select your current project ID from the list.
        + Click Create.
      * Enter the following command in Cloud Shell to create a folder called gcp-course:
        + mkdir gcp-course
        + cd gcp-course
      * Now clone the empty repository you just created:
        + gcloud source repos clone devops-repo
      * The previous cmd created an empty folder called devops-repo. Change to that folder:
        + cd devops-repo
    - Task 2: Create a Simple Python Application
      * You need some source code to manage. So, you will create a simple Python Flask web application. The application will be only slightly better than "hello world," but it will be good enough to test the pipeline you will build.
      * Select the gcp-course > devops-repo folder in the explorer tree on the left.

main.py

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

@app.route("/")

def main():

model = {"title": "Hello DevOps Fans."}

return render\_template('index.html', model=model)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=8080, debug=True, threaded=True)

* + - * Right-click on the devops-repo/templates folder.

layout.html.

<!doctype html>

<html lang="en">

<head>

<title>{{model.title}}</title>

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">

</head>

<body>

<div class="container">

{% block content %}{% endblock %}

<footer></footer>

</div>

</body>

</html>

index.html.

{% extends "layout.html" %}

{% block content %}

<div class="jumbotron">

<div class="container">

<h1>{{model.title}}</h1>

</div>

</div>

{% endblock %}

* + - * In Python, application prerequisites are managed using pip. devops-repo/ requirements.txt

itsdangerous==2.0.1

Flask==1.1.2

* + - * You have some files now, so save them to the repository. First, you need to add all the files you created to your local Git repo. In Cloud Shell, enter the following code:
        + cd ~/gcp-course/devops-repo
        + git add --all
      * To commit changes to the repository, you have to identify yourself. Enter the following commands, but with your information:
        + git config --global user.email "you@example.com"
        + git config --global user.name "Your Name"
      * Now, commit the changes locally:
        + git commit -a -m "Initial Commit"
      * You committed the changes locally, but have not updated the Git repository you created in Cloud Source Repositories. Enter the following cmd to push your changes to the cloud:
        + git push origin master
      * Refresh the Source Repositories web page. You should see the files you just created.
    - Task 3: Define a Docker Build
      * devops-repo/Docker. It defines how a Docker container is constructed.
        + FROM python:3.7

This is the base image. You could choose many base images. In this case, you are using one with Python already installed on it.

* + - * + WORKDIR /app
        + COPY . .

These lines copy the source code from the current folder into the /app folder in the container image.

* + - * + RUN pip install gunicorn
        + RUN pip install -r requirements.txt

This uses pip to install the requirements of the Python application into the container. Gunicorn is a Python web server that will be used to run the web app.

* + - * + ENV PORT=80
        + CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 main:app

The env variable sets the port that the application will run on (in this case, 80). The last line runs the web app using the gunicorn web server.

* + - * Verify that the completed file looks as follows and save it:

FROM python:3.7

WORKDIR /app

COPY . .

RUN pip install gunicorn

RUN pip install -r requirements.txt

ENV PORT=80

CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 main:app

* + - Task 4: Manage Docker Images with Cloud Build and Container Registry
      * The Docker image has to be built and then stored somewhere. You will use Cloud Build and Container Registry.
      * Enter the following command to use Cloud Build to build your image:
        + gcloud builds submit --tag gcr.io/$DEVSHELL\_PROJECT\_ID/devops-image:v0.1 .

The image will be stored in Container Registry.

The period at the end of the command represents the path to the Dockerfile: in this case, the current directory.

* + - * Navigation > Container Registry > Your image should be on the list.
      * Navigation > Cloud Build service > your build should be listed in the history.
      * You will now try running this image from a Compute Engine virtual machine.
        + Compute Engine service > Create Instance.
        + specify the following, and leave the remaining settings as their defaults:

Property Value

Container Deploy a container image to this VM instance

Container image gcr.io/<your-project-id-here>/devops-image:v0.1 (change the project ID where indicated)

Firewall Allow HTTP traffic

* + - * + Click Create.
        + Once the VM starts, access external IP address
      * You will now save your changes to your Git repository.
        + git add --all
      * Commit your changes locally:
        + git commit -am "Added Docker Support"
      * Push your changes to Cloud Source Repositories:
        + git push origin master
      * Return to Cloud Source Repositories and verify that your changes were added to source control.
    - Task 5: Automate Builds with Triggers
      * Navigation > Container Registry > a folder devops-image with at least one container in it.
      * Navigation > Cloud Build > one or more builds should be in your history.
      * Click the Triggers link on the left > Click Create trigger.
        + Name the trigger devops-trigger
        + Select your devops-repo Git repository.
        + Select .\*(any branch) for the branch.
        + Choose Dockerfile for Build Configuration and select the dafault image.
        + Accept the rest of the defaults, and click Create.

Graphical user interface, text, application, email

Description automatically generated

* + - * + To test the trigger, click Run and then Run trigger.
        + Click the History link and you should see a build running.
      * Return to the Container Registry service > see devops-repo folder, with a new image in it.
      * Update Main.py to "Hello Build Trigger."
      * Commit the change with the following command:
        + git commit -a -m "Testing Build Trigger"
        + git push origin master
      * Return to the Cloud Build service. You should see another build running.
    - Task 6: Test Your Build Changes
      * When the build completes, click on it to see its details. Under Execution Details, copy the image link, format should be gcr.io/qwiklabs-gcp-00-f23112/devops-repoxx34345xx.
      * create a new VM like before. paste the image you just copied.
      * Test in your browser. Your new message should be displayed.
  + Lab Review: Building a DevOps Pipeline 23 minutes - https://youtu.be/hYV7Dlh30ys
  + Quiz - DevOps Automation
    - What Google Cloud feature would be easiest to use to automate a build in response to code being checked into your source code repository?
      * Build triggers
      * This answer is correct. Cloud Build triggers have been designed specifically to trigger a build automatically when changes are made to source code.
    - Which Google Cloud tools can be used to build a continuous integration pipeline?
      * Cloud Build
      * Container Registry
      * Cloud Source Repositories
      * All of the options are correct. Source Repositories provides a private Git repository, Cloud Build builds containers, and Container Registry is a Docker images repository that performs vulnerability analysis. All three components are typically used in a continuous integration pipeline where on a commit, code is built and tested and an image is built and published to a registry.
* Choosing Storage Solutions
  + In this module, we discuss Google Cloud storage and data solutions and how to select the most suitable one to meet your business and technical requirements.
  + Key Storage Characteristics 5 minutes - https://youtu.be/Nmuz6nr9T6Q
  + Activity Intro: Defining storage characteristics 1 minute - https://youtu.be/7MYh3uloFA0
  + Activity Review: Defining storage characteristics 1 minute - https://youtu.be/Dpt-cf6OQow
  + Choosing Google Cloud Storage and Data Solutions 8 minutes - https://youtu.be/RPj4Og7l-Ns
  + Activity Intro: Choosing Google Cloud storage and data services - https://youtu.be/Ng91m--jM2o
  + Activity Review: Choosing Google Cloud storage and data services - https://youtu.be/Q07TUS6Gmvg
  + Quiz - Choosing Storage Solutions
    - You are a global financial services company with users all over the world. You need a database service that can provide low latency worldwide with strong consistency. Which service might you
      * Spanner
      * This answer is correct. A key feature of Spanner is scale for relational data with strong consistency, and it is globally distributed to provide low latency. The high availability and automatic replication are also strong features for financial services.
    - Currently, you are using Firestore to store information about products, reviews, and user sessions.You'd like to speed up data access in a simple, cost-effective way. What would you recommend?
      * Cache the data using Memorystore.
      * This answer is correct. Memorystore provides the best fit when considering data model, performance, scale, cost, and availability.
    - You want to analyze sales trends. To help achieve this, you want to combine data from your on-premises Oracle database with Google Analytics data and your web server logs. Where might you store the data so it is both easy to query and cost-effective?
      * BigQuery
      * This answer is correct. BigQuery is a data warehouse used for data analytics, and so is built for this type of use case. It provides the infrastructure to ingest data from many different sources, which is a requirement too. The cost model of paying for storage and then only for queries run is attractive too.
    - You need to store user preferences, product information, and reviews for a website you are building. There won't be a huge amount of data. What would be a simple, cost-effective, managed solution?
      * Firestore.
      * This answer is correct. Firestore provides automatic scale ACID transactions and live synchronization and is integrated with Google Cloud and Firebase. It also has a free tier.
* Google Cloud and Hybrid Network Architecture
  + In this module, we discuss Google Cloud network architectures, including hybrid architectures.
  + Designing Google Cloud Networks 4 minutes - https://youtu.be/2VyY90QkOCM
  + Designing Google Cloud Load Balancers 4 minutes - https://youtu.be/c0NuyDkFWeQ
  + Activity Intro: Defining network characteristics 1 minute - https://youtu.be/tP0\_iEFmaJw
  + Activity Review: Defining network characteristics 1 minute - https://youtu.be/0tj9xDHjxSg
  + Connecting Networks 12 minutes - https://youtu.be/P1P0w3SDWCU
  + Activity Intro: Diagramming your network 1 minute - https://youtu.be/cfjrK3imfCs
  + Activity Review: Diagramming your network 1 minute - https://youtu.be/sWPyrsj\_7no
  + Quiz - Google Cloud and Hybrid Network Architecture
    - You are a large bank deploying an online banking service to Google Cloud. The service needs high volume access to mainframe data on-premises. Which connectivity option would likely be best?
      * Cloud Interconnect
      * This answer is correct, because Cloud Interconnect provides high bandwidth and low latency. It does need encryption at the application level.
    - You want a secure, private connection between your network and a Google Cloud network. There is not a lot of volume, but the connection needs to be extremely reliable. Which configuration below would you choose?
      * VPN with high availability and Cloud Router.
      * This is the correct choice, because this offers a secure extremely reliable connection and is more cost-effective than Cloud Interconnect.
    - You have a contract with a service provider to manage your Google VPC networks. You want to connect a network they own to your VPC. Both networks are in Google Cloud. Which Connection option should you choose?
      * VPC peering
      * This answer is correct, because VPC peering allows connectivity across two VPC networks regardless of whether they belong to the same project or same organization.
    - You are deploying a large-scale web application with users all over the world and a lot of static content. Which load balancer configuration would likely be the best?
      * HTTP load balancer with SSL configured and the CDN enabled.
      * This answer is correct, because the traffic is HTTP(S), the load balancer should be external and global, and CDN enabled will help performance and cost.
* Deploying Applications to Google Cloud
  + In this module, we discuss the different options of deploying applications to Google Cloud. Google Cloud offers many possible deployment platforms, and the choice is not always immediately obvious.
  + Google Cloud Infrastructure as a Service 1 minute - https://youtu.be/xpK003zs6L8
  + Google Cloud Deployment Platforms 4 minutes - https://youtu.be/xpK003zs6L8
  + Deploying Apps to Google Cloud 45 minutes - <https://www.cloudskillsboost.google/course_sessions/816692/labs/114478>
    - Overview
      * deploy applications to the Google Cloud services App Engine, K8s Engine, and Cloud Run.
      * Diagram

        Description automatically generated
    - Objectives
      * Download a sample app from GitHub
      * Deploy to App Engine
      * Deploy to Kubernetes Engine
      * Deploy to Cloud Run
    - Task 1: Download a sample app from GitHub
      * Download/Clone a sample application from GitHub & test:
        + mkdir gcp-course
        + cd gcp-course
        + git clone https://GitHub.com/GoogleCloudPlatform/training-data-analyst.git
        + cd training-data-analyst/courses/design-process/deploying-apps-to-gcp
        + docker build -t test-python .
        + docker run --rm -p 8080:8080 test-python
      * Click Web Preview in the Google Cloud Shell. Then, select Preview on port 8080.
    - Task 2: Deploy to App Engine
      * App Engine is a completely automated deployment platform. It supports many languages, including Python, Java, JavaScript, and Go. To use it, you create a configuration file and deploy your applications with a couple of simple commands. In this task, you create a file named app.yaml and deploy it to App Engine.
      * Select the gcp-course/training-data-analyst/courses/design-process/deploying-apps-to-gcp/app.yaml folder
        + runtime: python37

Note: There are other settings you can add to the app.yaml file, but in this case only the language runtime is required.

* + - * In a project, an App Engine application has to be created:
        + gcloud app create --region=us-central
      * Now deploy your app with the following command:
        + gcloud app deploy --version=one --quiet
      * Navigation > App Engine > Dashboard > upper-right corner is a link to app > click to test
        + Note:By default, the URL in the form of https://project-id.appspot.com.
      * Make a change to the program to see how easy the App Engine makes managing versions.
      * main.py update it ”Hello App Engine”
      * Now, deploy version two with the following command:
        + gcloud app deploy --version=two --no-promote --quiet

Note: The --no-promote parameter tells App Engine to continue serving requests with the old version. This allows you to test the new version before putting it into production.

* + - * When the command completes, return to the App Engine dashboard. Click the link again, and version one will still be returned. It should return Hello GCP. This is because of the --no-promote parameter in the previous command.
      * On the left, click the Versions tab. Notice that two versions are listed.
        + Click on the version two link to test it. It should return Hello App Engine.
        + To migrate production traffic to version two, click Split Traffic at the top. Change the version to two, click Save.
        + Refresh the browser tab that earlier returned Hello GCP. It should now return the new version.
    - Task 3: Deploy to Kubernetes Engine
      * Kubernetes Engine allows you to create a cluster of machines and deploy any number of applications to it. Kubernetes abstracts the details of managing machines and allows you to automate the deployment of your applications with simple CLI commands.
      * To deploy an application to Kubernetes, you first need to create the cluster. Then you need to add a configuration file for each application you will deploy to the cluster.
      * Navigation > K8s Engine > Create > select GKE Standard option > click Configure & Create.
        + Click the three dots to the right of the cluster and then click Connect.
        + In the Connect to the cluster screen, click Run in Cloud Shell. This opens Cloud Shell with the connect command entered automatically.
        + To test your connection, enter the following command:

kubectl get nodes

* + - * + main.py update “Hello Kubernetes Engine”
        + Add a file training-data-analyst/courses/design-process/deploying-apps-to-gcp/ kubernetes-config.yaml

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: devops-deployment

labels:

app: devops

tier: frontend

spec:

replicas: 3

selector:

matchLabels:

app: devops

tier: frontend

template:

metadata:

labels:

app: devops

tier: frontend

spec:

containers:

- name: devops-demo

image: <YOUR IMAGE PATH HERE>

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: devops-deployment-lb

labels:

app: devops

tier: frontend-lb

spec:

type: LoadBalancer

ports:

- port: 80

targetPort: 80

selector:

app: devops

tier: frontend

Note: In the first section of the YAML file above, you are configuring a deployment. In this case, you are deploying 3 instances of your Python web app. Notice the image attribute. You will update this value with your image in a minute after you build it. In the second section, you are configuring a service of the type "load balancer". The load balancer will have a public IP address. Users will access your application through the load balancer.

* + - * + To use K8s Engine, you need to build a Docker image. Run the following cmds to use Cloud Build to create the image and store it in Container Registry:

cd ~/gcp-course/training-data-analyst/courses/design-process/deploying-apps-to-gcp

gcloud builds submit --tag gcr.io/$DEVSHELL\_PROJECT\_ID/devops-image:v0.2 .

* + - * + When the previous command completes, the image name will be listed in the output. The image name is in the form gcr.io/project-id/devops-image:v0.2.
        + Paste that image value in the kubernetes-config.yaml file, overwriting the string <YOUR IMAGE PATH HERE>.
        + Enter the following Kubernetes command to deploy your application:

kubectl apply -f kubernetes-config.yaml

* + - * + In the configuration file, 3 replicas of the application were specified. See :

kubectl get pods

* + - * + A load balancer was also added in the configuration file, Try IP in browser

kubectl get services

* + - Task 4: Deploy to Cloud Run
      * Cloud Run simplifies and automates deployments to Kubernetes. When you use Cloud Run, you don't need a configuration file. You simply choose a cluster for your application. With Cloud Run, you can use a cluster managed by Google, or you can use your own Kubernetes cluster.
      * To use Cloud Run, your application needs to be deployed using a Docker image and it must be stateless.
      * Main.py update it to “Hello Cloud Run”
      * To use Cloud Run, you need to build a Docker image. In Cloud Shell, enter the following commands to use Cloud Build to create the image and store it in Container Registry:
        + cd ~/gcp-course/training-data-analyst/courses/design-process/deploying-apps-to-gcp
        + gcloud builds submit --tag gcr.io/$DEVSHELL\_PROJECT\_ID/cloud-run-image:v0.1 .
      * Navigation > Cloud Run > Create service > This enables the Cloud Run API.
        + Click the Select link in the Container image URL text box. In the resulting dialog, expand cloud-run-image and select the image listed. Then click Select.
        + In Service name, type hello-cloud-run.
        + In Autoscaling set the Maximum number of instances to 6. Leave the rest as defaults.
        + For Authentication, select Allow unauthenticated invocations.
        + In Container, Variables & Secrets, Connections, Security for Container, select default in the Execution environment section.
        + Finally, click Create. Click on the URL that is automatically generated for the application. It should return Hello Cloud Run.
  + Lab Review Deploying Apps to Google Cloud 17 minutes - https://youtu.be/rwPNf\_pz7c8
  + Quiz - Deploying Applications to Google Cloud
    - You need to deploy an existing application that was written in.NETversion 4. The application requires Windows servers, and you don't want to change it. Which should you use?
      * ComputeEngine
      * This is the correct answer, because the approach is a lift and shift which is best supported by ComputeEngine, because Compute Engine offers full control over virtual machines including operating systems. No repackaging would be required.
    - You've been asked to write a program that uses Vision API to check for inappropriate content in photos that are uploaded to a CloudStorage bucket. Any photos that are inappropriate should be deleted. What might be the simplest, cheapest way to deploy in this program?
      * Cloud Functions
      * This is the correct answer, because the requirements for simplest and cheapest are met with CloudFunctions. CloudFunctions are for single purpose functions like image analysis. CloudFunctions also can be triggered by Cloud Storage events, so they provide seamless integration. The payment model based on number of requests, processing time of request (measured in 100ms units), and then other resources consumed is the most suitable of all options offered above. There is a free tier too. Cloud Functions also provides automatic scaling, high availability, and fault tolerance.
    - You have containerized multiple applications using Docker and have deployed them using Compute Engine VMs. You want to save on costs and simplify container management. What might you do?
      * Migrate the containers to GKE.
      * This is the correct answer. The applications are containerized, and GKE will help with the resource efficiency and hence cost, automate many aspects of the container management, and provide the best solution for the scenario.
* Designing Reliable Systems
  + Key Performance Metrics 1 minute - https://youtu.be/lEaOe0KO9cc
  + Designing for Reliability 7 minutes - https://youtu.be/HgvG14Y0N1s
  + Activity Intro: Designing Reliable, Scalable Applications 1 minute - https://youtu.be/8T5wqMZF8zU
  + Activity Review: Designing Reliable, Scalable Applications 1 minute - https://youtu.be/AYQTPQ3E0dU
  + Disaster Planning 6 minutes - https://youtu.be/n-bnABrJ\_rw
  + Activity Intro: Disaster planning 1 minute - https://youtu.be/UFHnu8BWFq0
  + Activity Review: Disaster planning 1 minute - https://youtu.be/kFSAZn4MdME
  + Quiz - Designing Reliable Systems
    - You need a relational database for a system that requires extremely high availability (99.999%). The system must run uninterrupted even in the event of a regional outage. Which database would you choose?
      * Spanner
      * This answer is correct, because Cloud Spanner meets all the requirements. It is a global relational database with high availability. Multi-regional instances have a monthly uptime of >=99.999%.
    - You're creating a service and you want to protect it from being overloaded by too many client retries in the event of a partial outage. Which design pattern would you implement?
      * Circuit breaker
      * This answer is correct, because the circuit breaker will attempt to prevent an operation that is likely to fail and therefore will protect the resource that is in partial outage and hopefully prevent cascading failure.
* Security
  + Security Concepts 4 minutes - https://youtu.be/X-ZzZHQVd-g
  + Securing People 3 minutes - https://youtu.be/3qqzhINqiPM
  + Securing Machine Access 2 minutes - https://youtu.be/9NVjQ8eg7RM
  + Network Security 5 minutes - https://youtu.be/o4XvGc\_KucA
  + Encryption 2 minutes - https://youtu.be/zR8fo7xUsIk
  + Activity Intro: Modeling Secure Google Cloud Services 1 minute - https://youtu.be/cgZ6KcR5Y98
  + Activity Review: Modeling Secure Google Cloud Services 1 minute - https://youtu.be/w15XOqvuL\_c
  + Quiz – Security
    - You don't want programmers to have access to production resources. What's the easiest way to do this in Google Cloud?
      * Create development and production projects, and don't give developers access to production.
      * he correct answer, because the simplest way is to have separate projects and not give developers access to the production project.
    - Which Google Cloud features could help prevent DDoS attacks?
      * CloudCDN
      * Google Cloud Armor
      * HTTP global load balancer
      * This answer is correct, because HTTP Load Balancing mitigates and absorbs many layer 4 below attacks such as SYN flood, IP fragment floods, and port exhaustion. CDN caches cacheable content at points of presence close to users. In the event of a DDoS attack for cacheable content, the requests are sent to points of presence, not to your servers/infrastructures, thus increasing the likelihood of the attack being absorbed. Google Cloud Armor is built for DDos mitigation, working with Cloud Load Balancing to detect DDoS attacks.
    - What Google Cloud service can you use to enforce the principle of least privilege when using Google Cloud?
      * IAMmembersandroles
      * This is the correct answer,because the principle of least privilege requires user permissions that are just enough to do what they need, and no more. IAM provides this level of control.
    - What do you have to do to enable encryption when using Cloud Storage?
      * Nothing as encryption is enabled by default.
      * This answer is correct, because Cloud Storage always encrypts data on the server side before it is written to disk. For server side encryption there are options of customer supplied or customer managed encryption keys, but these are only usually used for compliance reasons and are not necessary.
* Maintenance and Monitoring
  + Managing Versions 3 minutes - https://youtu.be/gs8hc9ZrUbM
  + Cost Planning 7 minutes - https://youtu.be/PNTKX1CDDiA
  + Monitoring Dashboards 2 minutes - https://youtu.be/I39ouDIWm6w
  + Activity Intro: Cost estimating and planning 1 minute - https://youtu.be/qjJtm9WvgWA
  + Activity Review: Cost estimating and planning 1 minute - https://youtu.be/OQpILOFuHYs
  + Monitoring Applications in Google Cloud 45 minutes - <https://www.cloudskillsboost.google/course_sessions/816692/labs/114509>
    - Overview
      * deploy an application to GC and then use the tools provided by GC to monitor it. You will use Cloud Logging, Trace, Profiler, and dashboards and create uptime checks and alerting policies.
    - Objectives
      * Download a sample app from Github
      * Deploy an application to App Engine
      * Examine the Cloud logs
      * View Profiler information
      * Explore Cloud Trace
      * Monitor resources using dashboards
      * Create uptime checks and alerts
    - Task 1: Download a sample app from Github
      * mkdir gcp-logging
      * cd gcp-logging
      * git clone https://GitHub.com/GoogleCloudPlatform/training-data-analyst.git
      * cd training-data-analyst/courses/design-process/deploying-apps-to-gcp
      * Expand the gcp-logging/training-data-analyst/courses/design-process/deploying-apps-to-gcp/main.py
      * Add the following import statement at the top of the file (line 2):
        + import googlecloudprofiler

Note: Profiler allows you to monitor the resources your applications use. For more information, see https://cloud.google.com/profiler/docs/.

* + - * After the main() function, add the following code snippet to start Profiler (after line 11):

try:

googlecloudprofiler.start(verbose=3)

except (ValueError, NotImplementedError) as exc:

print(exc)

* + - * Profiler will continuously report application metrics.
      * Note: This code simply turns Profiler on. Once on, Profiler starts reporting application metrics to Google Cloud.
      * You also have to add the Profiler library to your requirements.txt file. Open that file in the code editor and add the following:
        + google-cloud-profiler
      * Profiler has to be enabled in the project. In Cloud Shell, enter the following command:
        + gcloud services enable cloudprofiler.googleapis.com
      * To test the program
        + docker build -t test-python .
        + docker run --rm -p 8080:8080 test-python
    - Task 2: Deploy an application to App Engine
      * Now you will deploy the program to App Engine and use Google Cloud tools to monitor it.
      * gcp-logging/training-data-analyst/courses/design-process/deploying-apps-to-gcp/app.yaml folder.
        + runtime: python37
      * In a project, an App Engine application has to be created.
        + gcloud app create --region=us-central
        + gcloud app deploy --version=one --quiet
      * Navigation > App Engine > Dashboard > click the link to test
    - Task 3: Examine the Cloud logs
      * App Engine > Versions link > Click Tools > click Logs.
      * The logs should indicate that Profiler has started and profiles are being generated. If you get to this point too quickly, wait a minute and click Refresh.
    - Task 4: View Profiler information
      * Navigation > Profiler
      * Note: The gray bar at the top represents the total amount of CPU time used by the program. The bars below that represent the amount of CPU time used by the program's functions relative to the total. At this point, there is no traffic, so the chart is not very interesting. Throw some load at the application.
      * Navigation > Compute Engine > Create Instance > click Create > click SSH.
      * You will generate some traffic to your App Engine app using the web testing tool called Apache Bench. Enter the following commands to install it:
        + sudo apt update
        + sudo apt install apache2-utils -y
      * Update <your-project-id> with your PROJECT\_ID from connection details panel and enter the following command to generate some traffic to your App Engine application:
        + ab -n 1000 -c 10 https://<your-project-id>.appspot.com/
      * The command will make a thousand requests, 10 at a time, to your application.
      * When the requests are finished, on the Navigation menu, click Profiler.
      * Now there is a more interesting chart. Each bar represents a function. The width of the bars represents how much CPU time each function consumed.
      * The Profiler is a way developers can track down parts of a program that are consuming too many resources.
    - Task 5: Explore Cloud Trace
      * Every request to your application is added to the Trace list. Navigation > Trace.
      * The overview screen shows recent requests and allows you to create reports to analyze traffic. Because your program is new and has only one page, it's not very interesting, but in a real app there would be lots of useful information.
      * Click Trace list.
      * This shows a history of requests and their latency. Again, it's not very exciting because the application hasn't been running for very long. The chart in the upper-left plots requests and how long they took. The table to the right shows a list of requests. If you select a request, more detail will be displayed at the bottom of the screen.
      * Return to the SSH window where you entered the Apache Bench command previously.
        + ab -n 1000 -c 10 https://<your-project-id>.appspot.com/
      * You can also experiment with different values for the -n and -c parameters.
      * Repeat this a couple of times, and then return to the Trace list page.
    - Task 6: Monitor resources using Dashboards
      * Navigation > Monitoring.
      * click Dashboards. Cloud Monitoring analyzes the resources used in your projects and generates some default dashboards for you. In this exercise you have used App Engine and Compute Engine virtual machines, so a table similar to the one shown below should be displayed:
      * Click on the App Engine dashboard, and then select your project name. A dashboard of pertinent information for your App Engine application will appear.
      * In the left pane, click Dashboards.
      * Click on the VM Instances dashboard, and then select your instance. A dashboard for your VM will appear.
      * Optionally, return to the Dashboards page and click the +Create Dashboard. Try to create a custom dashboard.
      * For New Dashboard Name, type the custom dashboard name you have chosen. You can continue with your custom dashboard by adding the charts.
    - Task 7: Create uptime checks and alerts
      * Uptime checks > click the + Create Uptime Check:
        + Property Value
        + Title App Engine Uptime Check
        + Protocol HTTPS
        + Resource Type URL
        + Hostname <your-project-id>.appspot.com
        + Path /
        + Check Frequency 1 minute
        + Click on Next and then click Test to verify that your uptime check can connect to the resource. When you see a green check mark everything can connect.
        + When you are asked whether you want to create an Alerting Policy, do so.
        + In Alert & Notification, Name the policy Uptime Check Alert.
        + Click on drop down arrow next to Notification Channels, then click on Manage Notification Channels. A Notification channels page will open in new tab.
        + Scroll down the page and click on ADD NEW for Email.
        + In Create Email Channel dialog box, enter your personal email address in the Email Address field and a Display name.
        + Click on Save.
      * Go back to the previous tab. Click on Notification Channels again, then click on the Refresh icon to get the display name you mentioned in the previous step.
      * Now, select your Display name and click OK.
      * Click Create. The uptime check you configured takes a while for it to become active.
      * Disable the application to see whether your uptime check and alerting policy work. In the Cloud Console, on the Navigation menu, click App Engine.
        + Click Settings.
        + Click Disable application. Follow the instructions to disable the application.
      * Return to the App Engine Dashboard and test the URL. It shouldn't work anymore.
      * Monitoring > Uptime checks. Your uptime check should be failing. Wait & click refresh.
      * Click Alerting. An incident should have been fired.
      * Check your email. You should get a message from Cloud Monitoring.
      * App Engine Settings and re-enable your application. Then return to the Uptime checks page. The uptime check should be working again. If not, wait and then click refresh.
      * Return to the Alerting page. Your incident should be resolved. As before, you might have to wait a minute and then click refresh.
      * Check your email again. You should get a second email indicating that the alert recovered.
      * To make sure you don't get any emails after the project is deleted, delete your alerting policy and then delete your notification channel. At the top of the Alerting page, click Edit Notification Channels.
      * Find your email address and click the trash can icon to delete it.
      * Now click Uptime checks and delete your App Engine Uptime check.
  + Lab Review: Monitoring Applications in Google Cloud 19 minutes - https://youtu.be/y-so5lPBVnU
  + Quiz - Maintenance and Monitoring
    - 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?
      * Canary deployment
      * 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.
    - Your service has an availability SLO of 99%. What could you use to monitor whether you are meeting it?
      * Uptime check
      * 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.
    - 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?
      * Preemptible machines
      * 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.
    - 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?
      * Rolling update
      * 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.