Advanced GCP Course Structure for Intermediate Students

Module 1: Advanced Compute Services

- Compute Engine Deep Dive: Advanced VM configurations, custom machine types, and sole-tenant nodes.
- Kubernetes Engine: Managing and scaling applications with GKE, advanced cluster management, and hybrid networking.
- App Engine and Cloud Functions: Building scalable applications with PaaS and serverless architectures, including environment variables, custom runtimes, and traffic splitting.

Module 2: Advanced Networking

- Virtual Private Cloud (VPC) Advanced Configurations: Shared VPC, VPC peering, and firewall rules.
- Hybrid Connectivity: VPN, Cloud Interconnect, and configuring Cloud Router for hybrid networking.
- Cloud Load Balancing: Setup and configurations for HTTP(S), TCP/SSL, and UDP load balancing, including cross-region load balancing.

Module 3: Storage and Databases

- Cloud Storage: Best practices for storage classes, object life cycle management, and security policies.
- Cloud SQL and Cloud Spanner: Advanced configurations, high availability, disaster recovery, and performance optimization.
- NoSQL Databases: Deep dive into Firestore and Bigtable, schema design, and use cases.

Module 4: Big Data and Machine Learning

- BigQuery: Advanced SQL queries, performance optimization, and cost management.
- Dataflow: Building and optimizing data processing pipelines for streaming and batch data.
- AI and Machine Learning Services: Building custom ML models with AutoML and TensorFlow on AI Platform.

Module 5: DevOps and Security

- Cloud Build, Artifact Registry, and Container Registry: Setting up CI/CD pipelines for GCP.
- Deployment Manager and Terraform: Infrastructure as code for resource provisioning and management.
- Security Best Practices: Identity and Access Management (IAM), security keys, and network security.

Module 6: Monitoring, Logging, and Cost Management

- Cloud Monitoring and Logging: Setting up dashboards, alerts, and analyzing logs with BigQuery.
- Cloud Operations Suite: Using Cloud Trace, Cloud Debugger, and Cloud Profiler for application performance management.
- Cost Management: Understanding and optimizing GCP costs with Billing Reports and Cost Management tools.

Hands-on Projects and Labs

Project 1: Hybrid Cloud Networking Setup

Project Overview:

In this project, you'll establish a hybrid cloud environment between an on-premises data center and Google Cloud Platform (GCP). The goal is to simulate a real-world scenario where an organization requires seamless connectivity between their on-premises infrastructure and cloud resources for enhanced flexibility, scalability, and disaster recovery capabilities. You'll use Cloud VPN and Cloud Interconnect to achieve secure and efficient data transfer between the two environments.

Objective:

Design and implement a robust hybrid networking solution using GCP's Cloud VPN and Cloud Interconnect services, ensuring secure, low-latency communication between on-premises and GCP resources. Validate the setup by simulating data transfer and application connectivity across the hybrid network.

Task 1: Design the Hybrid Network Architecture

Objective: Create a detailed network architecture plan that outlines how the on-premises network will connect to GCP.

Activities:

- Assess the existing on-premises network infrastructure to determine the requirements for hybrid connectivity.
- Design a virtual private cloud (VPC) in GCP to host cloud resources that need to communicate with on-premises systems.
- Select appropriate regions and subnets in GCP for optimal performance and compliance.

Task 2: Set Up Cloud VPN for Secure Connectivity

Objective: Implement Cloud VPN to establish a secure IPSec VPN tunnel between the on-premises network and GCP.

Activities:

- Configure a Cloud VPN gateway and tunnel on the GCP side, specifying the IKE version, encryption, and traffic selectors.
- Set up the corresponding VPN gateway on the on-premises side, ensuring compatibility with GCP's VPN configuration.
- Establish the VPN tunnel and verify secure connectivity by testing IP reachability between the two networks.

Task 3: Implement Cloud Interconnect for High-Throughput Needs

Objective: Use Cloud Interconnect to provide a dedicated, high-throughput connection between on-premises and GCP for critical workloads.

Activities:

- Decide between Dedicated Interconnect for high-capacity needs or Partner Interconnect for more flexible options.
- Provision the Interconnect connection, including VLAN attachments and configuring BGP sessions for dynamic routing.
- Validate the Interconnect setup by conducting throughput and latency tests for high-priority traffic.

Task 4: Network and Security Configuration

Objective: Ensure the hybrid cloud environment is securely configured, with proper network routing and firewall rules in place.

Activities:

- Configure VPC and on-premises network routes to direct traffic through the VPN tunnel or Cloud Interconnect as appropriate.
- Implement firewall rules in GCP VPC and on-premises systems to protect resources and control traffic flow.
- Set up IAM roles and permissions in GCP to manage access to networking resources securely.

Task 5: Data Transfer Simulation and Application Connectivity Test

Objective: Simulate data transfer scenarios and test application connectivity across the hybrid network to ensure the setup meets organizational requirements.

Activities:

- Simulate data transfer between on-premises and GCP, such as file uploads to Cloud Storage and database replication to Cloud SQL.
- Deploy a test application in GCP that communicates with on-premises databases or applications, validating end-to-end connectivity and performance.
- Document the test results, including any latency, throughput metrics, and potential bottlenecks identified during the simulation.

Deliverables:

- A comprehensive network architecture diagram illustrating the hybrid cloud setup between on-premises and GCP.
- Configuration guides and scripts for setting up Cloud VPN, Cloud Interconnect, network routing, and firewall rules.
- A test plan and report detailing the data transfer simulation and application connectivity tests, including performance metrics and optimization recommendations.

Project 2: Multi-Tier Web Application Deployment

Project Overview :

This project involves deploying a scalable multi-tier web application on Google Cloud Platform (GCP) using Google Kubernetes Engine (GKE) for container orchestration, Cloud SQL for database services, and Cloud CDN (Content Delivery Network) for efficient content delivery. The aim is to simulate a real-world web application deployment that leverages the scalability and reliability of GCP services, ensuring a seamless user experience regardless of traffic volumes.

Objective:

Build and deploy a resilient, scalable web application architecture on GCP. This project will demonstrate your ability to integrate various GCP services to create a comprehensive solution that supports dynamic scaling, data persistence, and global content delivery.

Task 1: Design the Web Application Architecture

Objective: Plan a multi-tier web application architecture that separates the presentation layer, application logic, and data storage.

Activities:

- Identify the components of the web application, including the frontend, backend, and database tiers.
- Design the containerization strategy for the application, deciding on the container images for the frontend and backend services.
- Outline the service interaction flow and data processing logic, ensuring scalability and security are foundational elements of the architecture.

Task 2: Set Up Google Kubernetes Engine (GKE)

Objective: Deploy the application backend and frontend on GKE, configuring the cluster for auto-scaling.

Activities:

Create a GKE cluster, configuring it for high availability across multiple zones if needed. Deploy containerized backend and frontend applications to the GKE cluster, setting up appropriate deployments, services, and ingress controllers. Configure Horizontal Pod Autoscaler (HPA) for backend and frontend deployments to automatically scale the number of pods based on CPU usage or other custom metrics.

Task 3: Integrate Cloud SQL for Data Persistence

Objective: Utilize Cloud SQL as the relational database service for the application, ensuring secure and scalable data management.

Activities:

- Provision a Cloud SQL instance, selecting the appropriate database engine (e.g., MySQL, PostgreSQL) based on application requirements.
- Implement secure connections between the GKE cluster and Cloud SQL instance using Cloud SQL Proxy or Private IP.
- Design the database schema and perform initial data setup. Ensure that the application's backend services are correctly configured to interact with Cloud SQL for data storage and retrieval.

Task 4: Implement Cloud CDN for Efficient Content Delivery

Objective: Leverage Cloud CDN to cache and deliver static and dynamic content efficiently to users worldwide.

Activities:

- Configure Cloud CDN for the application, setting up caching policies for static assets (e.g., images, CSS, JavaScript files) and dynamic content where applicable.
- Integrate Cloud CDN with the application's load balancer or HTTP(S) load balancing services to optimize content delivery.
- Test the CDN setup by measuring load times and hit rates for content delivery across different geographical locations, making adjustments to caching rules as necessary.

Task 5: Performance Tuning and Security Best Practices

Objective: Optimize the web application deployment for performance and secure it according to GCP best practices.

Activities:

- Implement SSL/TLS for secure communication between the clients and the web application.
- Apply GCP security best practices, including network security policies, IAM roles, and permissions for application components.
- Conduct load testing to identify bottlenecks and optimize the application and infrastructure settings for improved performance.

Deliverables:

- A detailed architecture diagram showing the multi-tier web application setup on GCP.
- Deployment scripts and configuration files for GKE, Cloud SQL, and Cloud CDN integration.
- A performance optimization report, including load testing results, CDN caching efficiency metrics, and implemented security measures.

Project 3: Data Lake to Insights

Project Overview :

This project focuses on building a comprehensive data analytics pipeline on Google Cloud Platform (GCP) that spans from data ingestion to insight visualization. The objective is to create a data lake using Cloud Storage, leverage Dataflow for data ingestion and preprocessing, utilize BigQuery for data analysis, and employ Data Studio for visualization. This end-to-end project simulates a real-world scenario where raw data is transformed into actionable insights.

Objective:

Develop a scalable data analytics pipeline that enables efficient data ingestion, storage, analysis, and visualization. This project will demonstrate the ability to integrate various GCP services to process large volumes of data and uncover insights that can drive business decisions.

Task 1: Design and Set Up a Data Lake Using Cloud Storage

Objective: Establish a scalable and secure data lake architecture using Cloud Storage to store raw data.

Activities:

- Design a bucket architecture considering data types, access patterns, and security requirements.
- Create Cloud Storage buckets with appropriate storage classes and lifecycle management policies.
- Implement fine-grained access control and encryption settings to ensure data security and compliance.

Task 2: Data Ingestion and Preprocessing with Dataflow

Objective: Use Dataflow to ingest data into the data lake and preprocess it for analysis.

Activities:

- Design and implement Dataflow pipelines for data ingestion from various sources (e.g., streaming data, batch files).
- Apply transformations to clean, aggregate, and enrich the raw data as per the analysis requirements.
- Store the processed data back into Cloud Storage or directly into BigQuery for further analysis.

Task 3: Data Analysis with BigQuery

Objective: Perform scalable data analysis using BigQuery, GCP's serverless, highly scalable, and cost-effective multi-cloud data warehouse.

Activities:

- Design the schema and load the preprocessed data into BigQuery.
- Write SQL queries to analyze the data, looking for patterns, trends, and anomalies.
- Optimize query performance by partitioning, clustering tables, and using materialized views if necessary.

Task 4: Visualization and Insights with Data Studio

Objective: Create interactive dashboards and reports in Data Studio to visualize and communicate insights from the data.

Activities:

- Connect Data Studio to BigQuery datasets.
- Design and build interactive dashboards that highlight key metrics, trends, and insights from the analysis.
- Ensure dashboards are user-friendly, providing drill-down capabilities for deeper analysis where needed.

Task 5: Automation and Orchestration

Objective: Automate the data ingestion, processing, and analysis workflow to ensure the pipeline operates efficiently and reliably.

Activities:

- Use Cloud Composer (a managed Apache Airflow service) to orchestrate and schedule the Dataflow pipelines and BigQuery analysis jobs.
- Implement monitoring and alerting mechanisms using Stackdriver to track pipeline performance and catch any issues early.
- Document the automation process and provide guidelines for pipeline maintenance and troubleshooting.

Deliverables:

- Architecture diagram illustrating the complete data analytics pipeline from ingestion to visualization.
- Source code for Dataflow pipelines and SQL queries used for data analysis.
- Interactive dashboards and reports in Data Studio showcasing the insights derived from the data.
- A comprehensive guide detailing the pipeline setup, automation process, and maintenance instructions.

==========

Project 4: Serverless Microservices Architecture

Project Overview :

In this project, students will develop a serverless microservices architecture for a hypothetical e-commerce platform. The platform will utilize Google Cloud Functions for business logic implementation, API Gateway for managing and routing API requests, and Firestore for scalable, real-time data storage. This project aims to simulate the development and deployment of a serverless

application that can scale automatically with demand, demonstrating the power and flexibility of Google Cloud's serverless offerings.

Objective:

Design, develop, and deploy a serverless microservices-based application that leverages Cloud Functions, API Gateway, and Firestore to create a scalable, maintainable, and cost-effective e-commerce solution.

Task 1: Architecting the Serverless Microservices

Objective: Design the serverless architecture for the e-commerce platform, identifying key microservices and their responsibilities.

Activities:

- Define microservices such as user management, product catalog, order processing, and payment processing.
- Design RESTful APIs for each microservice, specifying endpoints, request/response structures, and HTTP methods.
- Plan the integration strategy between microservices, Cloud Functions, API Gateway, and Firestore.

Task 2: Setting Up Cloud Functions

Objective: Implement the business logic for each microservice as individual Cloud Functions.

Activities:

- Develop Cloud Functions for each identified microservice, writing code to handle specific tasks like user registration, product listing, order placement, and payment processing.
- Secure Cloud Functions with appropriate IAM roles and permissions to ensure they can access only the necessary resources.
- Test Cloud Functions locally and deploy them to Google Cloud, ensuring they are properly triggered by HTTP requests or other event sources.

Task 3: Configuring API Gateway

Objective: Use API Gateway to manage and route API requests to the appropriate Cloud Functions.

Activities:

- Create an API Gateway configuration that routes incoming API requests to the corresponding Cloud Functions based on the endpoint and HTTP method.
- Implement rate limiting and authentication mechanisms in API Gateway to protect the microservices from abuse and unauthorized access.
- Deploy the API configuration and test the endpoints to ensure requests are correctly routed and handled by Cloud Functions.

Task 4: Integrating Firestore for Data Storage

Objective: Utilize Firestore as the real-time NoSQL database for storing and retrieving application data.

Activities:

- Design the Firestore data model for the application, considering collections for users, products, orders, and payments.
- Implement data access logic within Cloud Functions to interact with Firestore, including CRUD operations and queries.
- Secure access to Firestore data using Firestore security rules that match the application's access control requirements.

Task 5: Application Testing and Deployment

Objective: Thoroughly test the serverless application to ensure reliability and performance, then deploy the solution.

Activities:

- Conduct integration testing to verify that the microservices interact correctly with each other and with Firestore.
- Perform load testing to evaluate the application's scalability and responsiveness under high traffic.
- Deploy the serverless application, configuring domain names in API Gateway for a professional URL structure.

Deliverables:

- An architectural diagram of the serverless microservices setup, detailing the interaction between Cloud Functions, API Gateway, and Firestore.
- Source code for all Cloud Functions and the API Gateway configuration.
- Firestore data model documentation and security rules.
- Testing reports that include integration and load testing results.
- Deployment guide and post-deployment monitoring and maintenance recommendations.

==========

Project 5: ML Model from Data to Deployment

Project Overview :

This project involves the end-to-end process of developing a machine learning (ML) model, from data preparation to deployment and consumption within an application. Students will utilize either Google Cloud AutoML or TensorFlow on Google Cloud AI Platform for model training, depending on the complexity of the task and the level of customization required. The project aims to simulate a real-world scenario where an ML model is used to generate insights or predictions to enhance application functionalities.

Objective:

Develop a scalable and efficient ML model to address a specific problem statement, train the model using Google Cloud's ML services, deploy the trained model on AI Platform, and integrate the model with a web or mobile application for real-time predictions.

Task 1: Problem Definition and Data Preparation

Objective: Identify a problem that can be solved with ML and prepare the dataset for model training.

Activities:

- Define a clear ML problem statement, choosing from classification, regression, or clustering based on the project goals.
- Collect and prepare a dataset suitable for the problem, performing data cleaning, feature engineering, and dataset splitting (train, validate, test sets).

Task 2: Model Training with AutoML or TensorFlow

Objective: Train an ML model using either Google Cloud AutoML for a no-code approach or TensorFlow on AI Platform for a custom model.

Activities:

- For AutoML: Use Google Cloud AutoML to train a model, selecting the appropriate AutoML product (e.g., AutoML Tables, AutoML Vision) based on the data type.
- For TensorFlow: Develop a custom model architecture using TensorFlow, and set up a training job on AI Platform, tuning hyperparameters as needed.
- Evaluate the model's performance using the validation set, iterating on the model design as necessary to achieve desired accuracy or performance metrics.

Task 3: Model Deployment on AI Platform

Objective: Deploy the trained ML model on Google Cloud AI Platform for serving predictions.

Activities:

- Deploy the trained model to AI Platform, creating a model resource and version, and configuring the deployment settings such as machine type and scaling policies.
- Test the deployed model by sending prediction requests to the AI Platform prediction endpoint with sample input data.

Task 4: Application Integration for Consuming the Model

Objective: Integrate the deployed ML model into a web or mobile application, enabling real-time predictions within the application.

Activities:

- Develop a simple web or mobile application frontend that allows users to input data for predictions.
- Implement backend logic to interact with the AI Platform prediction endpoint, sending data to the model and displaying predictions to the user.

- Ensure the application handles prediction requests and responses efficiently, providing a seamless user experience.

Task 5: Monitoring, Logging, and Model Updates

Objective: Set up monitoring and logging for the deployed model and establish a process for updating the model with new data or improvements.

Activities:

- Use Google Cloud Monitoring and Logging to track the usage, performance, and errors of the prediction endpoint.
- Plan for regular model evaluations with new data to detect model drift or performance degradation over time.
- Develop a strategy for retraining and redeploying the model with AI Platform, ensuring minimal disruption to the application.

Deliverables:

- A detailed report on the ML problem statement, data preparation process, and rationale for choosing AutoML or a custom TensorFlow model.
- Source code for the ML model training and evaluation, including any scripts used for data preprocessing and hyperparameter tuning.
- Documentation on the model deployment process on AI Platform and integration steps with the application for real-time predictions.
- A working prototype of the web or mobile application integrated with the ML model, demonstrating real-time predictions.
- A monitoring and model update strategy, including guidelines for logging, performance tracking, and model retraining and redeployment processes.DevOps Project: Microservices Deployment on Amazon EKS with CI/CD Pipeline

==========

Project 6: DevOps Automation with CI/CD Pipeline

Project Overview :

This project focuses on the implementation of a Continuous Integration and Continuous Deployment (CI/CD) pipeline using Google Cloud Build, targeting automated deployments to Google Kubernetes Engine (GKE). The project simulates a real-world DevOps scenario where an application's deployment process is automated, including code testing, building, and deployment phases, with an emphasis on reliability and efficiency. Canary deployments are incorporated to ensure that new versions are rolled out safely.

Objective:

To automate the application deployment process using a CI/CD pipeline that integrates automated testing, builds the application, and deploys updates to a GKE cluster with minimal human intervention.

Implement canary deployments to gradually roll out new features, ensuring system stability and user satisfaction.

Task 1: Setting Up the Project Environment

Objective: Prepare the GKE cluster and Google Cloud Build setup.

Activities:

- Create a GKE cluster to host the application, configuring the cluster with necessary permissions and network settings.
- Set up a Google Cloud Build project, connecting it to the source code repository (e.g., GitHub or Cloud Source Repositories).

Task 2: Developing the CI/CD Pipeline

Objective: Implement a CI/CD pipeline using Cloud Build that automates the testing, building, and deploying of the application.

Activities:

- Define a cloudbuild.yaml file that specifies the steps for code testing, building Docker images, and deploying to GKE.
- Integrate automated testing scripts that run unit and integration tests on every commit to the repository, ensuring code quality.
- Configure Docker image building and pushing to Google Container Registry (GCR) as part of the pipeline.

Task 3: Implementing Canary Deployments

Objective: Set up canary deployments to gradually roll out new versions of the application to a subset of users.

Activities:

- Modify the deployment configuration to support canary releases, using Kubernetes deployments and services to manage version traffic.
- Use Google Cloud's traffic management features or a service mesh like Istio to control the percentage of traffic directed to the canary version.
- Establish metrics and criteria for evaluating the success of the canary deployment, including rollback strategies in case of issues.

Task 4: Monitoring and Logging

Objective: Implement monitoring and logging to track the pipeline's performance and the application's health post-deployment.

Activities:

- Set up Google Cloud Monitoring and Logging for the CI/CD pipeline and the GKE cluster, ensuring visibility into the build and deployment processes.
- Configure alerts based on specific metrics (e.g., failed deployment steps, high error rates in the canary deployment) to enable quick response to issues.

Task 5: Documentation and Best Practices

Objective: Document the CI/CD pipeline setup, canary deployment process, and monitoring strategies.

Activities:

- Create comprehensive documentation covering the pipeline configuration, deployment strategies, and rollback plans for canary releases.
- Include a section on best practices for using CI/CD with GKE, highlighting how to maintain high availability and minimize downtime during deployments.

Deliverables:

- A fully functional CI/CD pipeline configured in Google Cloud Build, with source code and cloudbuild.yaml file.
- Canary deployment configuration files and a strategy document outlining the process, evaluation criteria, and rollback mechanisms.
- Monitoring and logging setup with documented alerts and metrics to watch during and after deployments.
- Comprehensive documentation detailing the project setup, pipeline flow, canary deployment strategy, and monitoring/alerting systems.
- A summary of lessons learned and best practices for implementing CI/CD pipelines and canary deployments in GKE.

==========

Project 7: Comprehensive GCP Security Audit

Project Overview :

This project is designed to simulate a real-world scenario where students will perform a comprehensive security audit of a Google Cloud Platform (GCP) environment. The audit will involve examining the current security posture using tools such as IAM (Identity and Access Management) and the Security Command Center. Based on the findings, students will then implement necessary security improvements to ensure the environment adheres to best practices for security.

Objective:

To conduct a detailed security assessment of a GCP environment, identify potential security issues, and implement enhancements to mitigate risks, ensuring the infrastructure is secure against common threats and vulnerabilities.

Task 1: Initial Setup and Environment Review

Objective: Familiarize with the GCP environment and plan the audit.

Activities:

- Gain access to the GCP environment to be audited.
- Review the current architecture, focusing on key components that will be audited, including Compute Engine instances, Cloud Storage buckets, and networking setup.

Task 2: IAM Policies Audit

Objective: Review and assess IAM policies to ensure the principle of least privilege is followed.

Activities:

- Use the IAM & Admin console to review current IAM policies.
- Identify overly permissive roles or accounts with unnecessary access to resources.
- Recommend adjustments to IAM policies to tighten security, ensuring users and service accounts have only the permissions necessary for their roles.

Task 3: Security Command Center Analysis

Objective: Leverage the Security Command Center to identify security threats and vulnerabilities..

Activities:

- Set up and configure the Security Command Center for the project.
- Run a comprehensive scan to detect security threats, misconfigurations, and vulnerabilities.
- Analyze the findings, prioritizing issues based on severity.

Task 4: Implementing Security Improvements

Objective: Address the identified security issues and improve the security posture of the GCP environment.

Activities:

- Implement changes to IAM policies based on the audit findings, applying the principle of least privilege.
- Resolve vulnerabilities identified by the Security Command Center, such as patching outdated software, securing Cloud Storage buckets, and fixing network misconfigurations.
- Configure security alerts for ongoing monitoring of critical security events.

Task 5: Best Practices and Documentation

Objective: Document the audit process, findings, and implemented improvements. Share best practices for maintaining a secure GCP environment.

Activities:

- Document each step of the audit process, including the rationale behind each security improvement implemented.
- Create a best practices guide for GCP security, covering IAM policies, resource configurations, and the use of the Security Command Center.
- Offer recommendations for regular security reviews and continuous monitoring strategies.

Deliverables:

- A detailed audit report outlining the initial security posture, findings from the IAM and Security Command Center analysis, and a list of implemented security improvements.
- Updated IAM policies and configurations that adhere to the principle of least privilege.
- Documentation of resolved vulnerabilities and security improvements made across the GCP environment.
- A best practices guide for GCP security, including recommendations for ongoing security management and monitoring.
- A presentation summarizing the audit process, key findings, and the impact of the security improvements on the organization's security posture.