GCP Professional Cloud Architect Crash Course

Get Fully Prepared to Crush the Exam

Victor Dantas
Author, Certified Cloud Architect

Pearson

Day 2 Schedule

- **Segment 5: Deployment and operational best practices**
  - Interacting with GCP programmatically
  - Automating infrastructure and application deployments
  - Testing and validating deployments
  - API best practices
  - Troubleshooting best practices
  - Alerting and incident response
  - Demo: deployment and operational best practices
  - Q&A (10min)
- **Segment 6: Designing a solution infrastructure that meets business requirements**
  - Business use cases and strategies
  - Success measurements
  - Buy vs build
  - Movement of data and external integrations
  - Cost optimization
  - Supporting the application design
  - Creating a migration plan
  - Demo: designing a solution infrastructure that meets business requirements
  - Q&A (10min)
  - Break (10min)

Day 2 Schedule

- **Segment 7: Designing a solution infrastructure that meets technical requirements**
  - Designing for high availability and failover
  - Elasticity of cloud resources, quotas and limits
  - Designing for scalability
  - Designing for performance
  - Demo: designing a solution infrastructure that meets technical requirements
  - Q&A (10min)
  - Break (10min)
- **Segment 8: Case studies solutioning – pen-and-paper exercises**
  - Case Study Breakdown: EHR Healthcare
  - Case Study Breakdown: Helicopter Racing League
  - Case Study Breakdown: TerramEarth
  - Case Study Breakdown: Mountkirk Games

Segment 5: Deployment and operational best practices

Objectives

- Interacting with GCP programmatically
- Application development
- Automating infrastructure and application deployments
- Testing and validating deployments
- API best practices
- Troubleshooting best practices
- Alerting and incident response

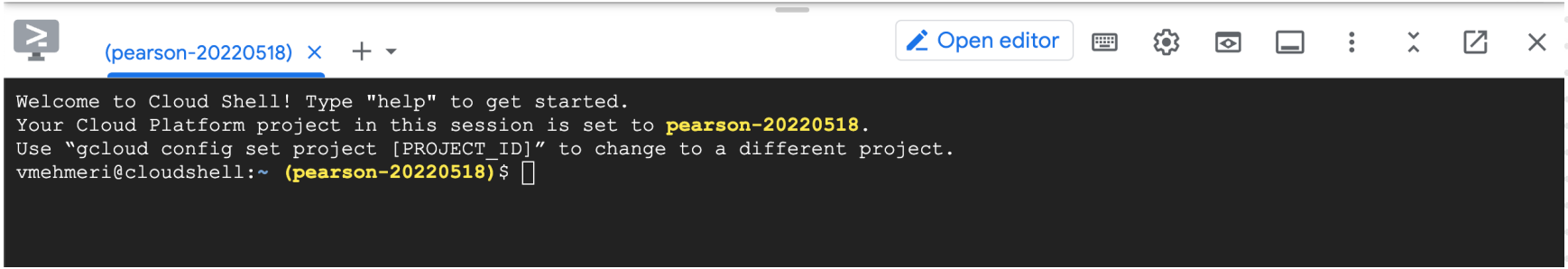
Segment 5: Deployment and operational best practices

Objectives

- Interacting with GCP programmatically
- Application development
- Automating infrastructure and application deployments
- Testing and validating deployments
- API best practices
- Troubleshooting best practices
- Alerting and incident response

Cloud Shell

- Browser-based, interactive shell environment
- Tools pre-installed (*gcloud*, *gsutil*, *bq*, etc.)
- Built-in code editor
- Runs on a shared VM (convenient, but no SLA)



Google Cloud CLI

- Command-line interface (*gcloud*)
- Set of tools to create and manage GCP resources
- Release levels:
  - General Availability (***gcloud***): Commands are considered fully stable
  - Beta (***gcloud beta***): Commands are functionally complete, but could have issues
  - Alpha (***gcloud alpha***): Commands are in early release and may change without notice

Cloud Storage Utility (gsutil)

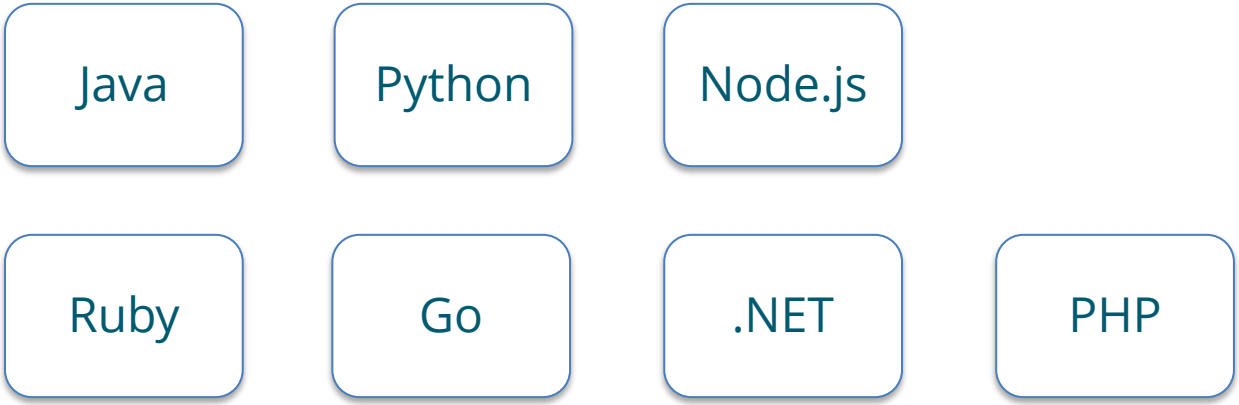
- You get *gsutil* as part of the Google Cloud CLI
- Specific to Cloud Storage for running a wide range of bucket and object management tasks
  - Creating and deleting buckets
  - Uploading, downloading, and deleting objects
  - Listing buckets and objects
  - Editing object and bucket ACLs
  - Etc.

BigQuery Command-Line Tool (bq)

- Can be used to interact with BigQuery service
  - Create datasets and tables
  - Create views
  - List datasets, tables, and views
  - Load data into BigQuery
  - Run queries
  - Etc.

Cloud Software Development Kit (Cloud SDK)

Language-specific client libraries



Segment 5: Deployment and operational best practices

Objectives

- Interacting with GCP programmatically
- Application development
- Automating infrastructure and application deployments
- Testing and validating deployments
- API best practices
- Troubleshooting best practices
- Alerting and incident response

Twelve-factor App Development on GCP

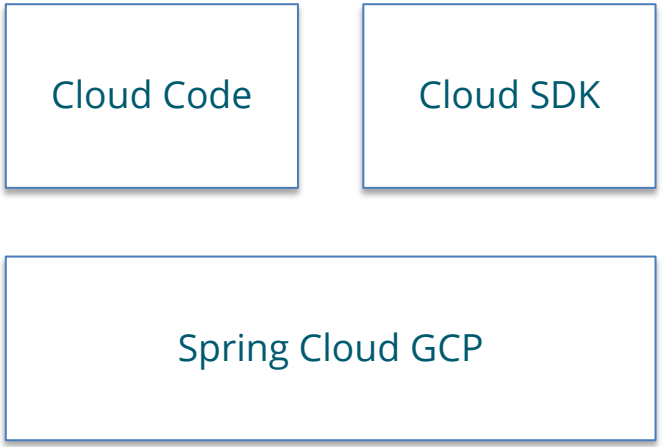
Factor	What to think of
Codebase	**Cloud Source Repositories**. Git repositories
Dependencies	Container-based development. **Container/Artifact Registry**
Configuration	Configuration files. *ConfigMaps (**GKE**)*
Backing services	Keep data external to the app. **Cloud Storage**, **Cloud SQL**, **Memorystore**, etc.
Build, release, run	**Cloud Build**, **Container/Artifact Registry**, **Cloud Deploy**. Separate stages for building, releasing, and deploying
Processes	Avoid "sticky" sessions. Keep processes stateless. **Memorystore** to cache the state if needed

Twelve-factor App Development on GCP

Factor	What to think of
Port binding	Do not hard-code port numbers in code. **Compute Engine**, **GKE**, **App Engine**, **Cloud Run**. Environment variables
Concurrency	Decompose apps. Microservices. **App Engine**, **Managed Instance Groups**, **Cloud Functions**, **GKE**
Disposability	Use native services for infrastructure components. Rolling updates, health checks, **GKE**
Environment parity	Infrastructure-as-Code (IaC). **Cloud Source Repositories**, **Deployment Manager**, **Container/Artifact Registry**
Logs	**Cloud Logging**. **Operations suite**
Admin processes	*CronJobs* (**GKE**). **Cloud Tasks**, **Cloud Scheduler**

Google Cloud Developer Tools

- Code
- Build
- Manage artifacts
- Deploy



Google Cloud Developer Tools

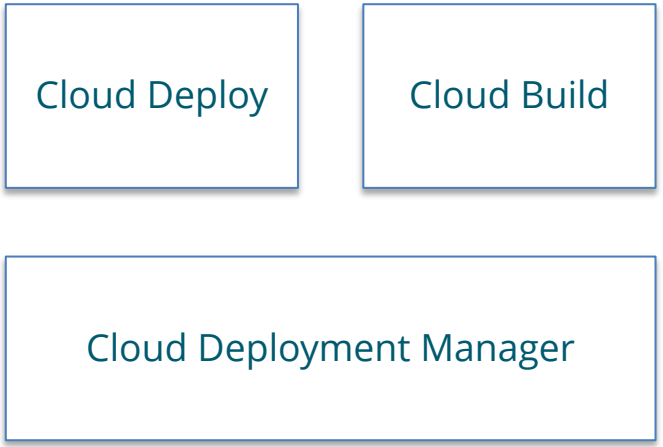
- Code
- Build
- Manage artifacts
- Deploy

Google Cloud Developer Tools

- Code
- Build
- Manage artifacts
- Deploy

Google Cloud Developer Tools

- Code
- Build
- Manage artifacts
- Deploy

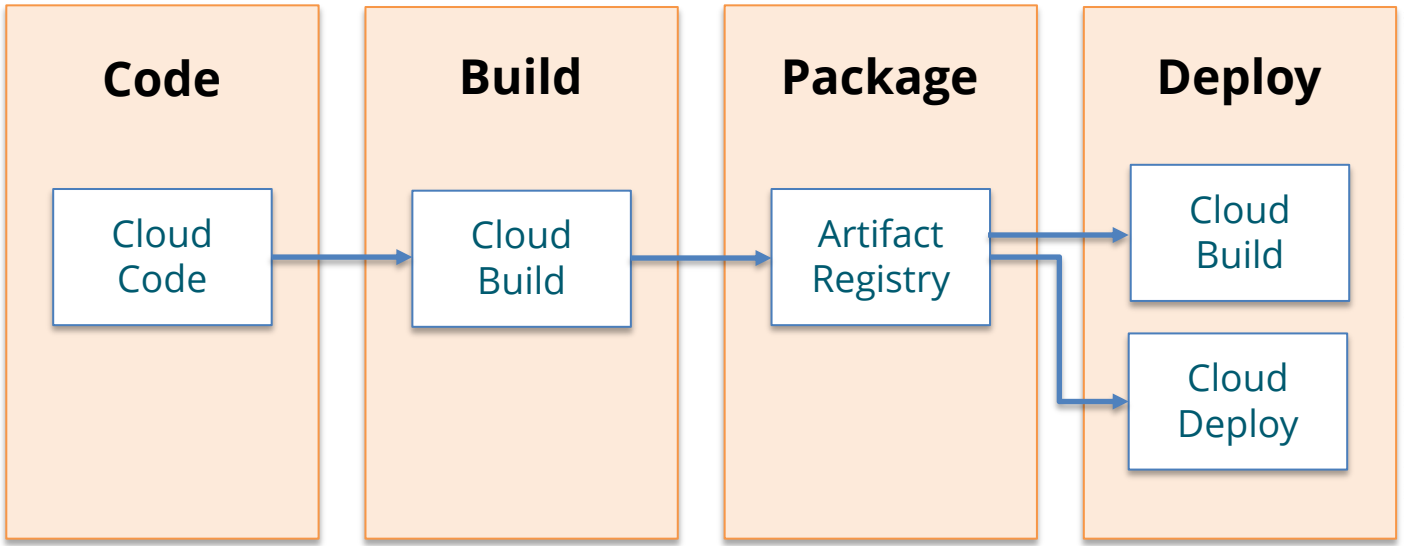


Segment 5: Deployment and operational best practices

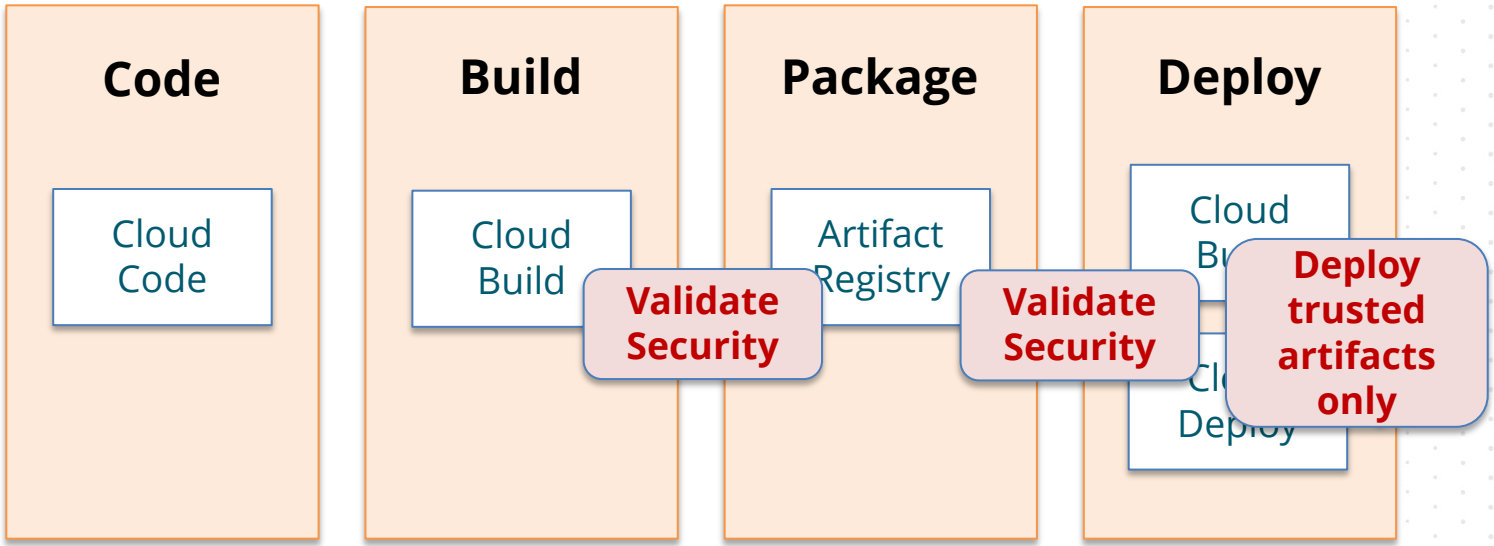
Objectives

- Interacting with GCP programmatically
- Application development
- Automating infrastructure and application deployments
- Testing and validating deployments
- API best practices
- Troubleshooting best practices
- Alerting and incident response

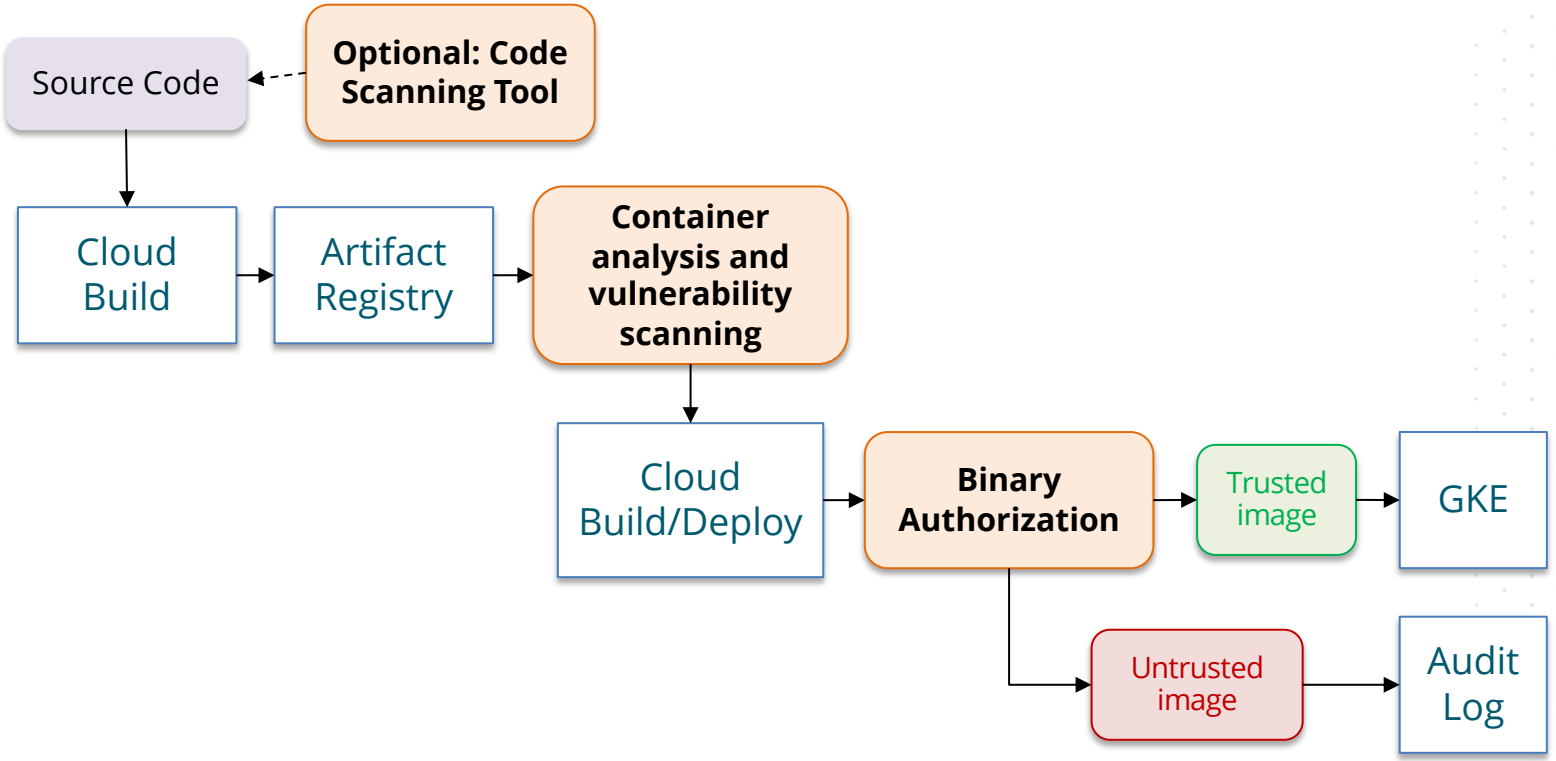
CI/CD on Google Cloud



CI/CD on Google Cloud: DevSecOps



DevSecOps with GKE



Define Infrastructure as Code

Deployment Manager Templates

- Can be written in either **Jinja 2.10.x** or **Python 3.x**

```
- name: vm-template
  type: compute.v1.instance
  properties:
    zone: us-central1-a
    machineType: zones/us-central1-
a/machineTypes/n1-standard-1
    disks:
    - deviceName: boot
      type: PERSISTENT
      boot: true
      autoDelete: true
      initializeParams:
        sourceImage: projects/debian-
cloud/global/images/family/debian-11
    networkInterfaces:
    - network: global/networks/default
```

```
resources.append({
    'name': 'vm-template',
    'type': 'compute.v1.instance',
    'properties': {
        'zone': 'us-central1-a',
        'machineType': 'zones/us-central1-
a/machineTypes/n1-standard-1',
        'disks': [{
            'deviceName': 'boot',
            'type': 'PERSISTENT',
            'boot': True,
            'autoDelete': True,
            'initializeParams': {
                'sourceImage':
                    'projects/debian-
cloud/global/images/family/debian-11'
            }
        }],
        'networkInterfaces': [{
            'network': 'global/networks/default'
        }]
    }
})
```

Setting up a CI/CD Pipeline with Cloud Build

1) Prepare your application code
2) Create a build config file in YAML or JSON containing the build steps

```
steps:
- name: 'gcr.io/cloud-builders/kubectl'
  args: ['set', 'image', 'deployment/mydepl', 'my-image=gcr.io/my-project/myimage']
  env:
  - 'CLOUDSDK_COMPUTE_ZONE=us-east4-b'
  - 'CLOUDSDK_CONTAINER_CLUSTER=my-cluster'
```

```
kubectl set image deployment/mydepl my-image=gcr.io/my-project-myimage
```

Setting up a CI/CD Pipeline with Cloud Build

3) Define a Cloud Build trigger to automatically start a build on code changes
   - Any changes on source repository
   - Push to a branch
   - Push to a tag
   - Pull request (only available with GitHub)
   - Changes only to specific files (file name pattern defined by *glob* strings) – example: **.txt*

Other Ways to Automate Things

Cloud Scheduler

Workflows

Cloud Composer

(Based on Apache Airflow)

Deployment Automation Best Practices

- Use the same deployment process for every environment
- Use the same packages for every environment (**tip: containers**)
- Make it possible to recreate the state of any environment from version control

Segment 5: Deployment and operational best practices

Objectives

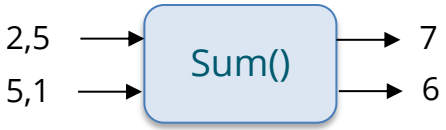
- Interacting with GCP programmatically
- Application development
- Automating infrastructure and application deployments
- Testing and validating deployments
- API best practices
- Troubleshooting best practices
- Alerting and incident response

Types of Tests

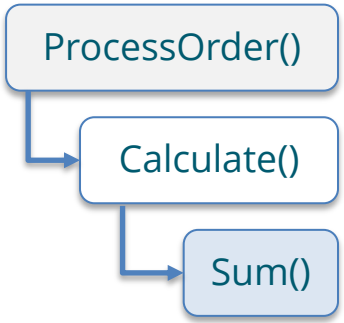
- Unit tests
- Functional acceptance tests
- Nonfunctional acceptance tests (performance, capacity, security, etc.)

Software Testing

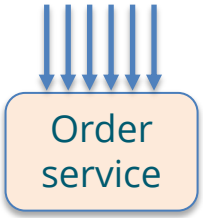
Unit



Integration



Load

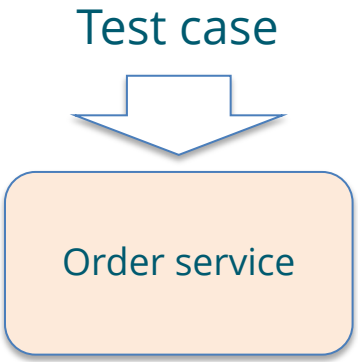


Software Testing: Acceptance Tests

Acceptance criteria

- [x] Payment request is submitted
- [x] If confirmed, inventory is updated
- [x] Shipping request is submitted
- [x] Confirmation email is sent

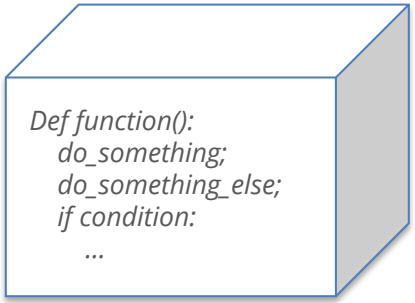
...



Software Testing: Blackbox vs. Whitebox

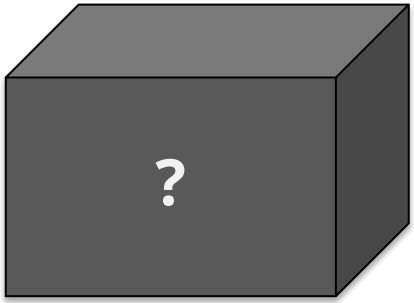
Whitebox

Code-based



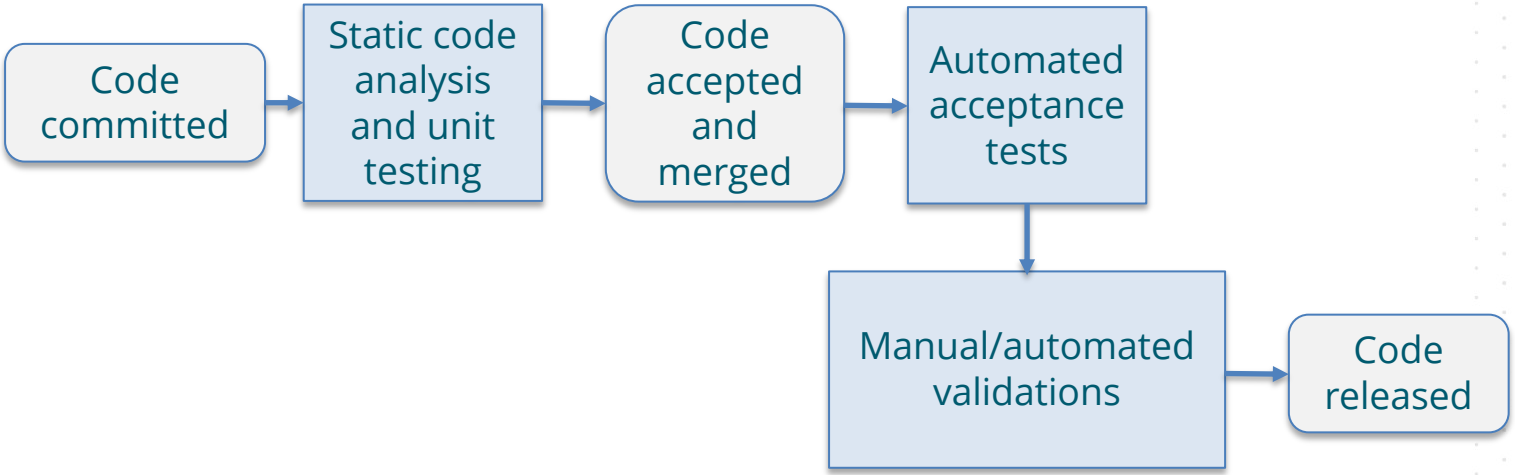
Blackbox

Specification-based



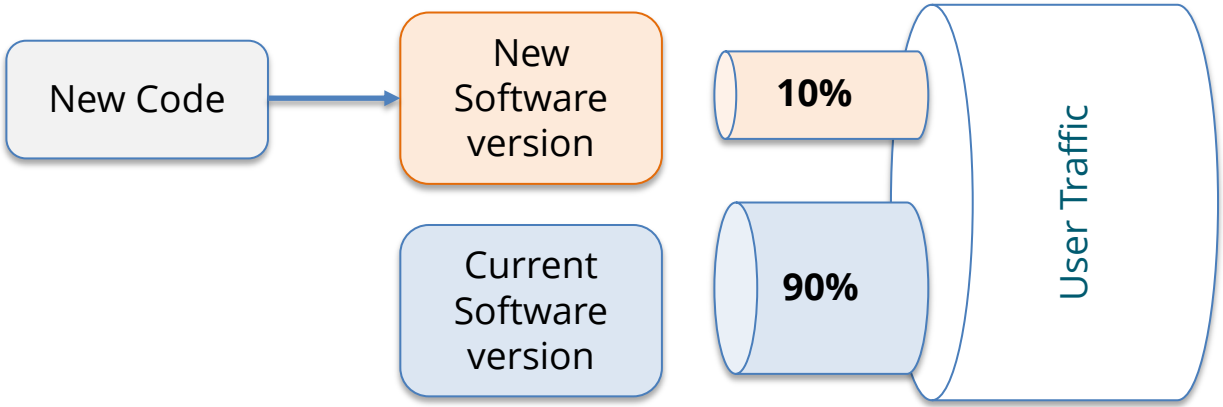
When and How to Test

Test **early** and **automate**



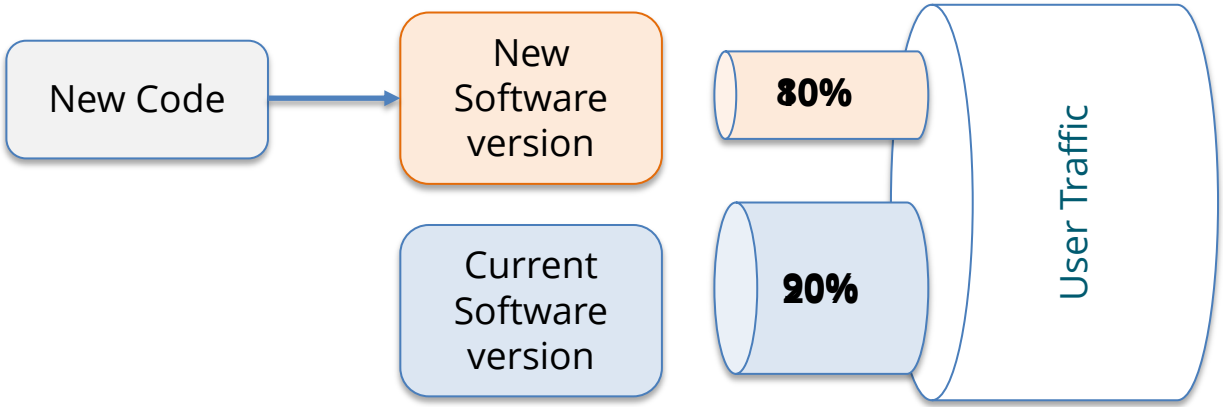
Canary Testing

Roll out new versions incrementally



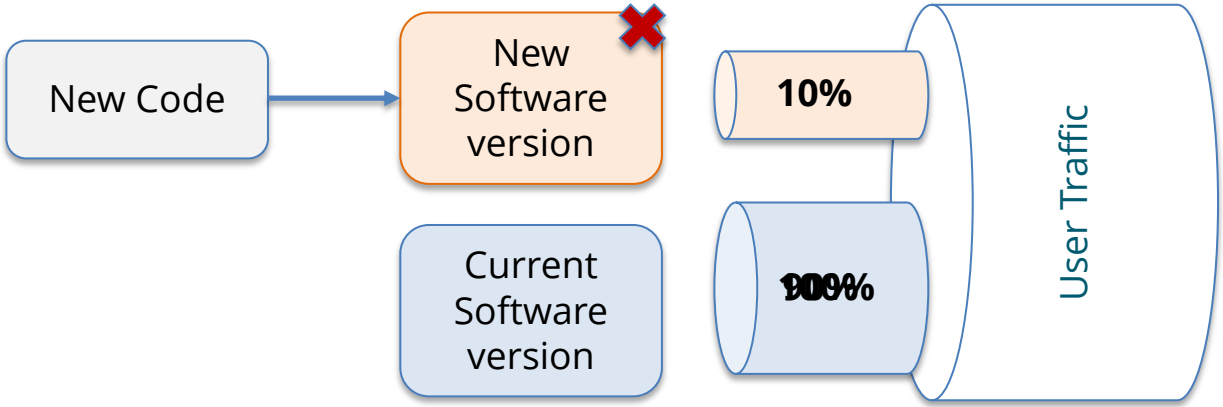
Canary Testing

Roll out new versions incrementally



Canary Testing

Roll back if there's an issue



Only 10% of users affected

Canary Testing on GCP

- Cloud Run

```
gcloud run services update-traffic
SERVICE_NAME \
--to-revisions=${PROD}=90,${CANARY}=10  \
--platform managed
```

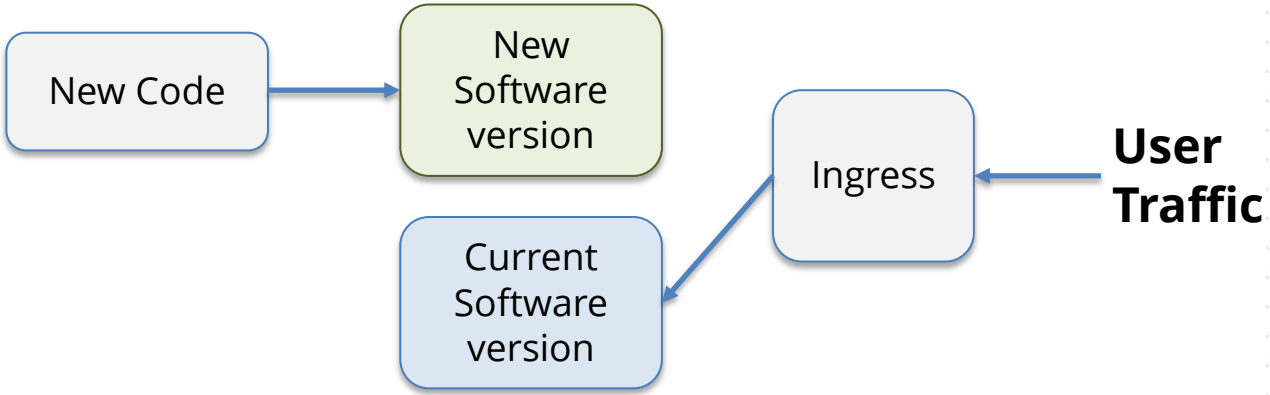
- GKE
  - Expose canary and current deployments under the same service
  - Use the number of replicas to dictate traffic percentage

Canary Testing on GCP

- App Engine
  - Create multiple versions of a service
  - Use “Split traffic” feature

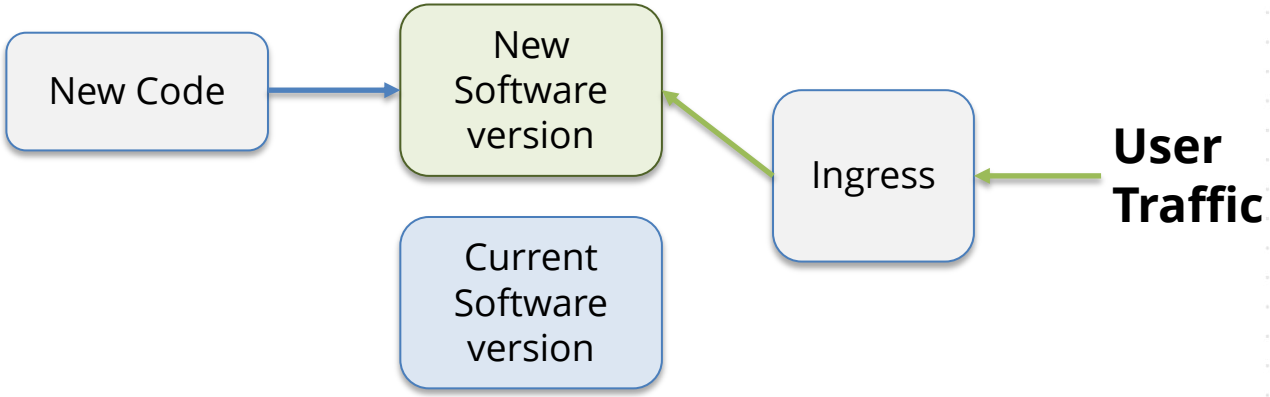
Blue/Green Deployment and Testing

Roll out new version alongside current, then switch



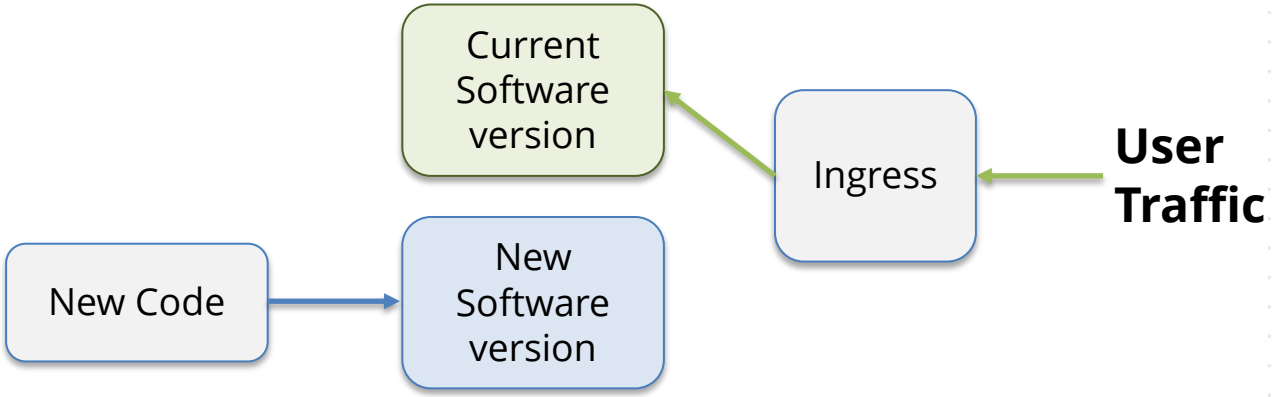
Blue/Green Deployment and Testing

Roll out new version alongside current, then switch



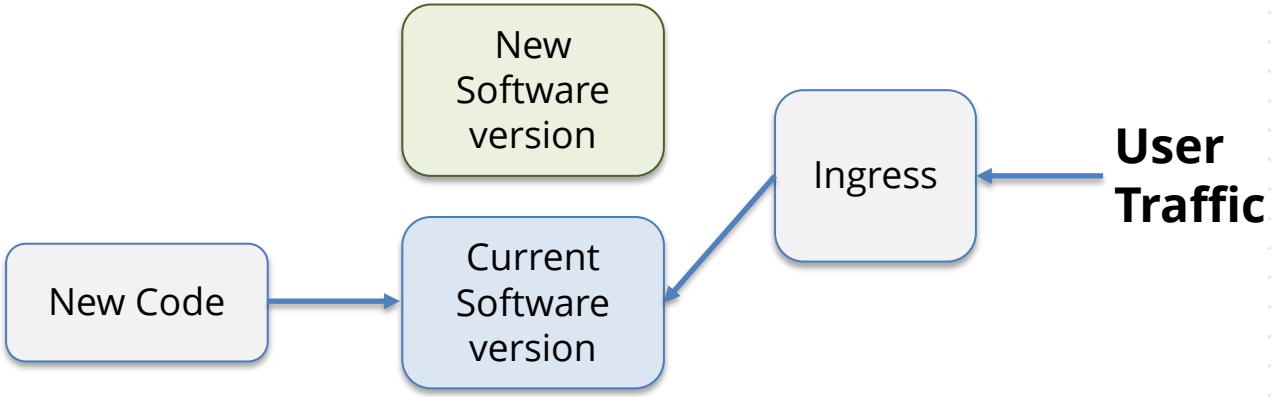
Blue/Green Deployment and Testing

Roll out new version alongside current, then switch



Blue/Green Deployment and Testing

Roll out new version alongside current, then switch



Blue/Green Testing on GCP

- GKE
  - Create separate deployment
  - Update the service selector to point to the new version
  - Update again to roll back if necessary
- Compute Engine / App Engine
  - Deploy separate environments
  - Configure load balancer to switch between “blue” and “green”

GKE Rolling Updates

- Pods are incrementally replaced with new ones
- Rolling updates != canary testing
- Allows for workload updates without downtime
- You can use **kubectl rollout** to inspect, pause, resume, and/or rollback an update

```
kubectl rollout status deployment nginx
```

```
kubectl rollout pause deployment nginx
```

```
kubectl rollout resume deployment nginx
```

```
kubectl rollout undo deployment nginx
```

Test and Validation Best Practices

- Leverage an error budget
- Test in production during periods of low usage (“dark launch”)
- Alternatively, test in pre-production using a synthetic workload that mimics production traffic
- Test the resilience of your applications (fault injection, chaos engineering)
- Test applications and recovery processes periodically

Segment 5: Deployment and operational best practices

Objectives

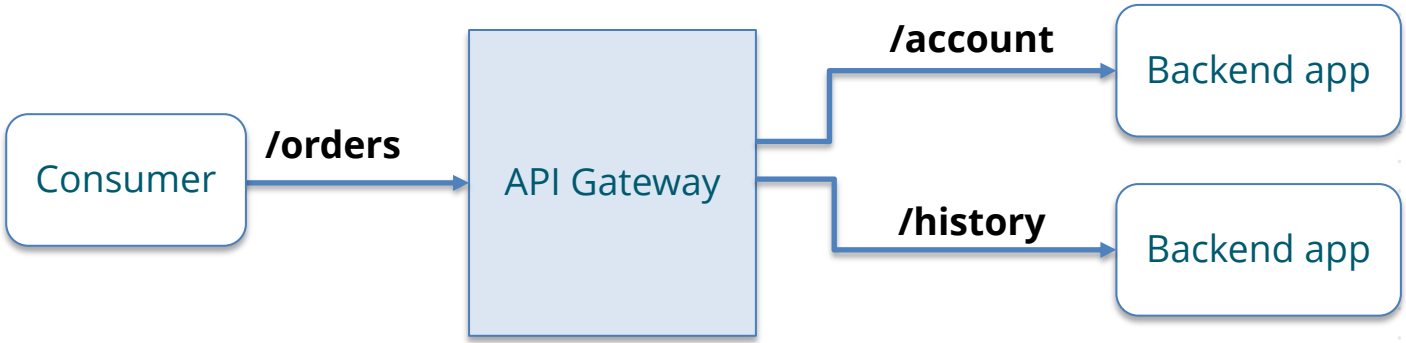
- Interacting with GCP programmatically
- Application development
- Automating infrastructure and application deployments
- Testing and validating deployments
- API best practices
- Troubleshooting best practices
- Alerting and incident response

API Versioning

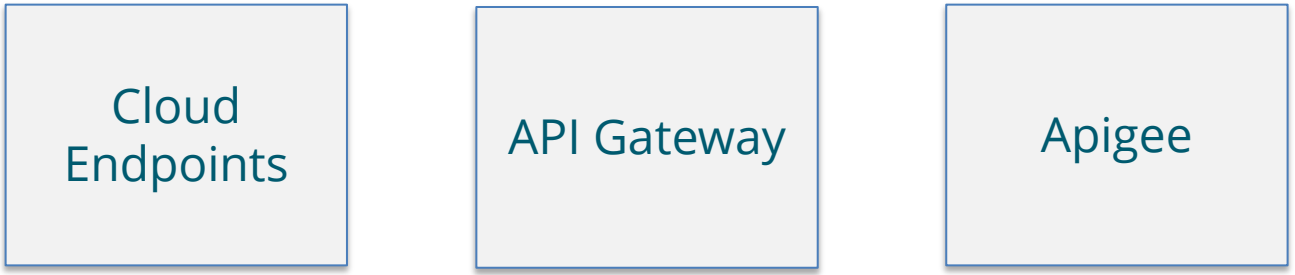
- For backwards-compatible changes, create a new *minor* version (e.g: 1.1 -> 1.2)
- Major version number is incremented when a breaking change is introduced
- Major version number should be part of the URI (*/v1/...*, */v2/...*)
- Different versions of the same API work at the same time for a reasonable transition period

API Gateway Pattern

Make backend functionality available to consumer applications through an **API gateway**

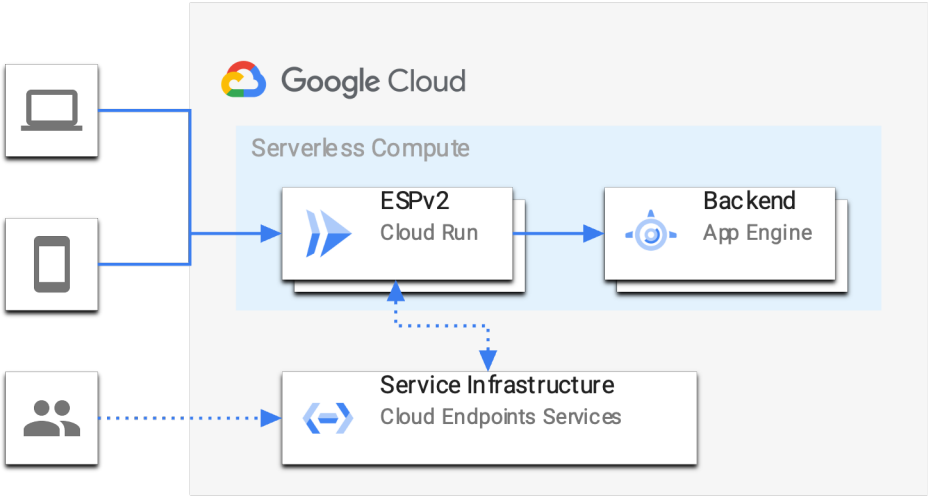


Ways to Implement API Gateways on GCP



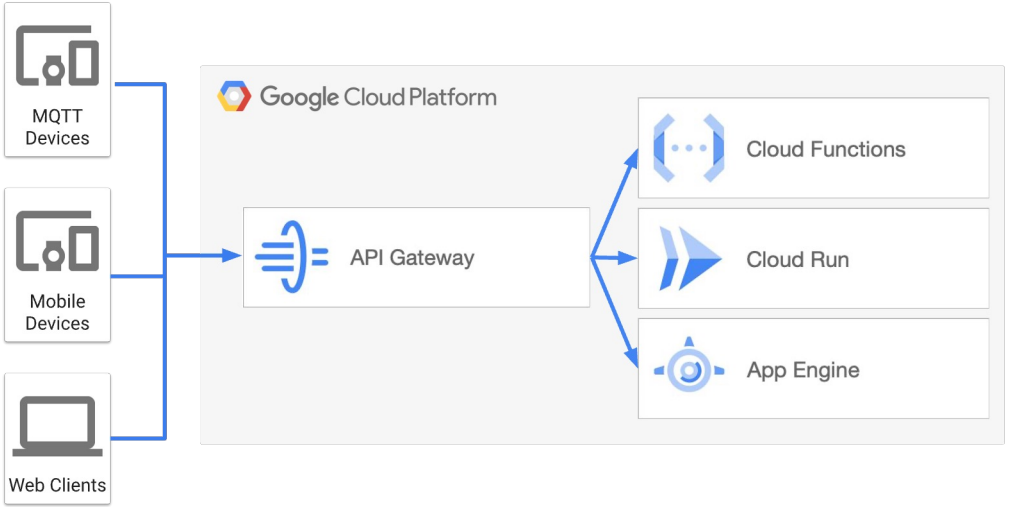
Cloud Endpoints

- You install Google Cloud's Extensible Service Proxy (ESP)
- You have full control over the API gateway



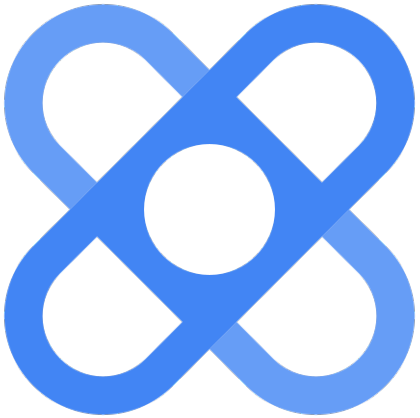
API Gateway

GCP's **API Gateway** is a fully managed gateway for serverless backends



Apigee

- Apigee is a fully featured API management platform
- Can proxy APIs outside of Google Cloud
- ”Enterprise” feature set: security, rate-limiting, quotas, monetization



OpenAPI Specification

- Formerly known as *Swagger RESTful API specification*
- Defines an industry-standard, language-agnostic interface and description format for REST APIs
- **API Gateway, Cloud Endpoints** and **Apigee** support the OpenAPI specification

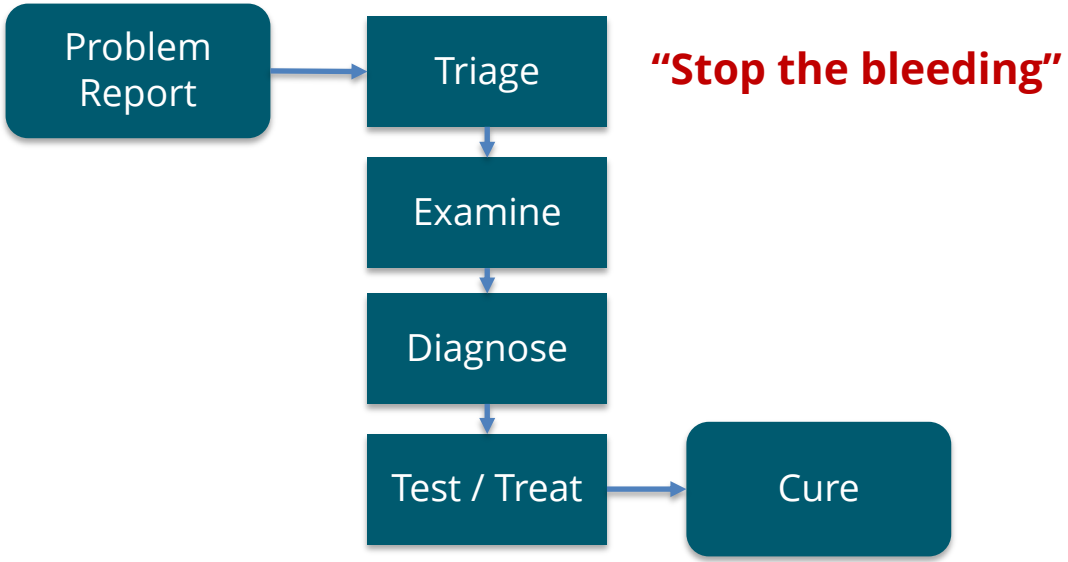
Segment 5: Deployment and operational best practices

Objectives

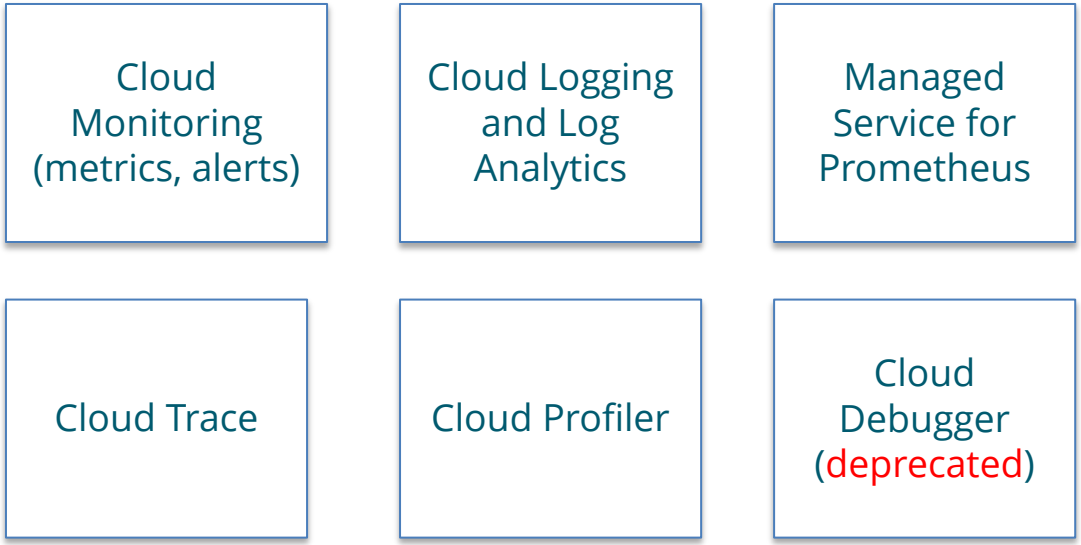
- Interacting with GCP programmatically
- Application development
- Automating infrastructure and application deployments
- Testing and validating deployments
- API best practices
- Troubleshooting best practices
- Alerting and incident response

Troubleshooting Best Practices

- *Blameless post-mortems* culture
- Follow the **Site Reliability Engineering** (**SRE**) troubleshooting process



Troubleshooting on GCP: Operations Suite



DR Patterns

Segment 5: Deployment and operational best practices

Objectives

- Interacting with GCP programmatically
- Application development
- Automating infrastructure and application deployments
- Testing and validating deployments
- API best practices
- Troubleshooting best practices
- Alerting and incident response

Monitoring Alerts

- You define alerting policies that watch for some metric for some time period
  - Metric-absence condition: no data for a specific duration window
  - Metric-threshold condition: values are more/less than a specified threshold for a duration window

Monitoring Alerts: Sample Policy

```
{
    "displayName": "Very high CPU usage",
    "combiner": "OR",
    "conditions": [
        {
            "displayName": "CPU usage is extremely high",
            "conditionThreshold": {
                "aggregations": [
                    {
                        "alignmentPeriod": "60s",
                        "crossSeriesReducer": "REDUCE_MEAN",

                        (…)
                    }
                ],
                "comparison": "COMPARISON_GT",
                "duration": "900s",
                "filter": "metric.type=\"compute.googleapis.com/instance/cpu/utilization\"
                          AND resource.type=\"gce_instance\"",
                "thresholdValue": 0.9,
                "trigger": {
                    "count": 1
                }
            }
        (…)
```

Log-based Alerting Policies

- Alerting policies can also be created for logs
- Well suited for catching **security-related events**
- Also useful for catching **rare** and **important** events in application logs
- You create log-based alerts from Logs Explorer

Incident Response Best Practices

- Assign clear service ownership
- Define a documented and well-exercised incident management plan
- Make sure the teams know “generic mitigations” they can quickly apply to minimize time to mitigation
- Document diagnostic procedures and mitigations for known outage scenarios
- Document query snippets or build a **Data Studio** dashboard with frequent log queries

Demo: Deployment and Operational Best Practices

1. Cloud Monitoring alerts for CPU usage > 80%
2. Custom Cloud Monitoring dashboard

Questions Breakdown

You are designing a new application that has several configuration settings that are environment-specific. Your application also needs to connect to different databases, and you want to store the database credentials securely. What should you do?

A. Store the environment-specific configurations in the source code, using different branches. Store the credentials in environment variables.

B. Store the environment-specific configurations in environment variables. Store the credentials in the source code.

C. Store the environment-specific configurations in environment variables. Store the credentials in Secret Manager.

D. Store both the environment-specific configurations and the database credentials in environment variables.

Questions Breakdown

You are designing a new application that has several **configuration settings that are environment-specific**. Your application also needs to connect to different databases, and you want to **store the database credentials securely**. What should you do?

A. Store the environment-specific configurations in the source code, using different branches. Store the credentials in environment variables.

B. Store the environment-specific configurations in environment variables. Store the credentials in the source code.

C. Store the environment-specific configurations in environment variables. Store the credentials in Secret Manager.

D. Store both the environment-specific configurations and the database credentials in environment variables.

Questions Breakdown

Your team is developing a new application to be deployed on a Google Kubernetes Engine (GKE) cluster. You want to have any changes pushed to the main branch on your team's GitHub repository to automatically initiate a build that will package the application and test it. What should you do to enforce this process?

A. Have each developer configure and run a test suite on their workstation that builds and tests the code before committing to a remote branch.

B. Spin up a test environment on GCP and create a custom script that checks for changes in the main branch of the GitHub repository, runs build and test steps, and publishes validated artifacts to Artifact Registry.

C. Configure a Cloud Build build config that includes the required build and test steps. Configure a Cloud Build trigger for changes to the main branch.

D. Configure a Cloud Build config that includes the required build steps. Ensure the built-in Container Analysis functionality is enabled to run code tests.

Questions Breakdown

Your team is developing a new application to be deployed on a Google Kubernetes Engine (GKE) cluster. You want to have any **changes pushed to the main branch** on your team's GitHub repository to **automatically initiate a build** that will **package** the application **and test it**. What should you do to **enforce this process**?

A. Have each developer configure and run a test suite on their workstation that builds and tests the code before committing to a remote branch.

B. Spin up a test environment on GCP and create a custom script that checks for changes in the main branch of the GitHub repository, runs build and test steps, and publishes validated artifacts to Artifact Registry.

C. Configure a Cloud Build build config that includes the required build and test steps. Configure a Cloud Build trigger for changes to the main branch.

D. Configure a Cloud Build config that includes the required build steps. Ensure the built-in Container Analysis functionality is enabled to run code tests.

Questions Breakdown

You're responsible for the security aspect of the software development life cycle (SDLC) of your team's application that runs on Google Kubernetes Engine (GKE). You want to ensure that only trusted container images are deployed to GKE clusters. What should you do?

A. Use Cloud Build to define a build step that checks if the code comes from your team's source repository and marks the container as trusted.

B. Create a Cloud Function that is triggered each time there are code changes on the repository. Have the Cloud Function deploy the container only if all functional tests are successful.

C. Configure Cloud IAM so that only trusted principals have permissions to deploy containers on GKE.

D. Enable Binary Authorization on the GKE clusters and configure a Binary Authorization policy.

Questions Breakdown

You're responsible for the security aspect of the software development life cycle (SDLC) of your team's application that runs on Google Kubernetes Engine (**GKE**). You want to ensure that **only trusted container images are deployed** to GKE clusters. What should you do?

A. Use Cloud Build to define a build step that checks if the code comes from your team's source repository and marks the container as trusted.

B. Create a Cloud Function that is triggered each time there are code changes on the repository. Have the Cloud Function deploy the container only if all functional tests are successful.

C. Configure Cloud IAM so that only trusted principals have permissions to deploy containers on GKE.

D. Enable Binary Authorization on the GKE clusters and configure a Binary Authorization policy.

Questions Breakdown

You need to improve the reliability and quality of software released in production by reducing the number of rollbacks and the rollback time for faulty software. What should you do? (Choose two.)

A. Reduce the number of database writes.

B. Re-architect the application into microservices.

C. Adopt a blue/green deployment model.

D. Automate software tests in a CI/CD pipeline.

E. Migrate from zonal to regional resources on GCP.

Questions Breakdown

You need to improve the reliability and quality of software released in production by **reducing the number of rollbacks** and the **rollback time** for faulty software. What should you do? (Choose two.)

A. Reduce the number of database writes.

B. Re-architect the application into microservices.

C. Adopt a blue/green deployment model.

D. Automate software tests in a CI/CD pipeline.

E. Migrate from zonal to regional resources on GCP.

Questions Breakdown

You have recently updated an application running on App Engine with a new version. You're getting reports that the application is taking a longer time than usual to load for some users. What should you do?

A. Enable VPC Flow logs to investigate the network.

B. Debug the source code until you find and fix the issue, then push a new version to App Engine.

C. Roll back to the previous, working version. Then, use Cloud Trace and Cloud Logging to diagnose the problem in a test/staging environment.

D. Redirect production traffic to the staging environment while you investigate the issue with Cloud Profiler. Redirect it back once you've fixed the issue.

Questions Breakdown

You have **recently updated** an application running on App Engine with a new version. You're getting reports that the **application is taking a longer time than usual to load** for some users. What should you do?

A. Enable VPC Flow logs to investigate the network.

B. Debug the source code until you find and fix the issue, then push a new version to App Engine.

C. Roll back to the previous, working version. Then, use Cloud Trace and Cloud Logging to diagnose the problem in a test/staging environment.

D. Redirect production traffic to the staging environment while you investigate the issue with Cloud Profiler. Redirect it back once you've fixed the issue.

Segment 6: Designing a solution infrastructure that meets business requirements

Objectives

- Business use cases and strategies
- Success measurements
- Buy vs build
- Movement of data and external integrations
- Cost optimization
- Supporting the application design
- Creating a migration plan

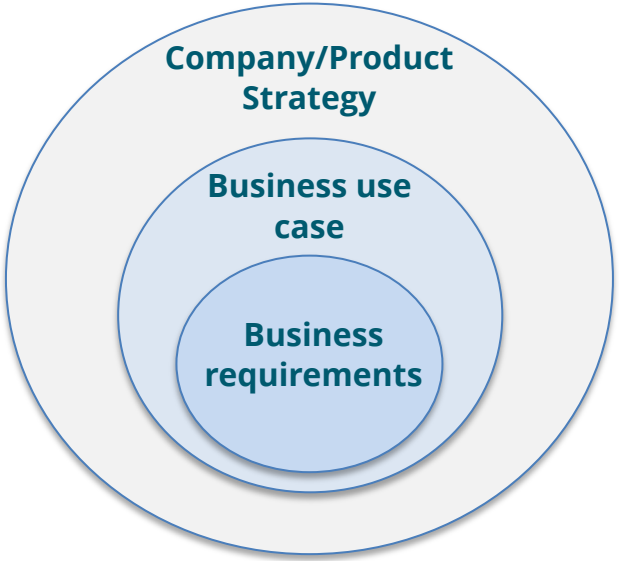
Segment 6: Designing a solution infrastructure that meets business requirements

Objectives

- Business use cases and strategies
- Success measurements
- Buy vs build
- Movement of data and external integrations
- Cost optimization
- Supporting the application design
- Creating a migration plan

Business Use Cases and Strategies

Start from the big picture



Example

- Company Strategy: cloud-first, global reach
- Business use case: SaaS app
- Business requirements: low latency to users, high availability

Business Requirements

Examples

Requirement	What to think of
Reduce infrastructure administration costs	Managed services, serverless, automation, PaaS and SaaS

Business Requirements

Examples

Requirement	What to think of
Maintain high availability	Multi-AZ or Multi-region deployments. Managed services, serverless. Load balancers.

Business Requirements

Examples

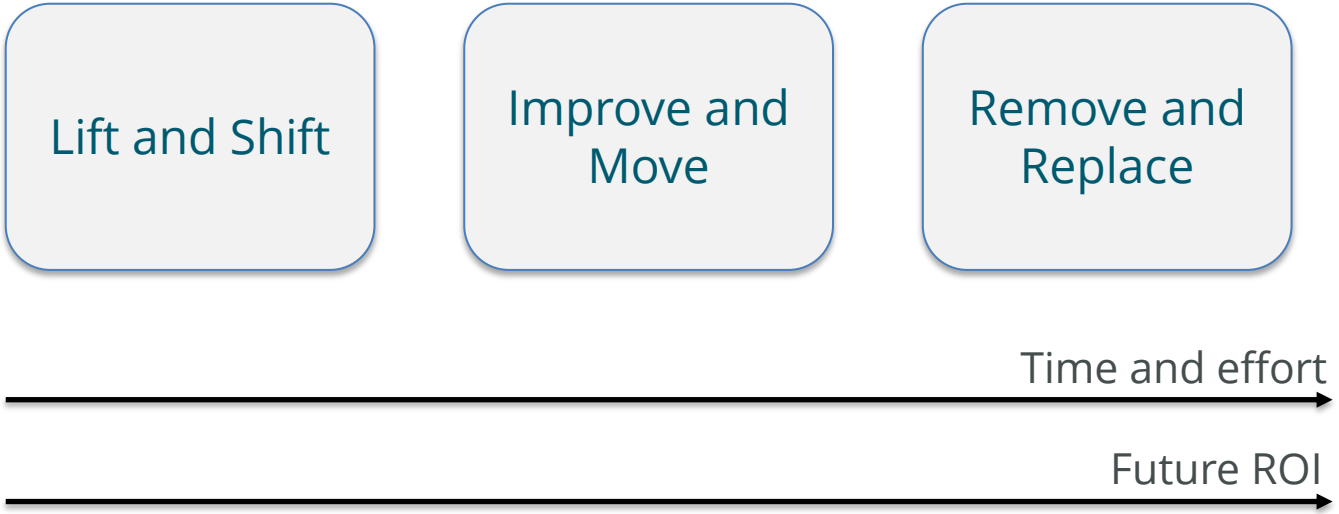
Requirement	What to think of
Increase development agility	CI/CD, dev/prod environment parity, containers, infrastructure as code, blue/green and canary deployments.

Business Requirements

Examples

Requirement	What to think of
Increase ability to generate predictions and insights	Modern data warehousing, AI/ML, data pipelines, BI and data visualization tools.

Cloud Migration Strategies



Segment 6: Designing a solution infrastructure that meets business requirements

Objectives

- Business use cases and strategies
- Success measurements
- Buy vs build
- Movement of data and external integrations
- Cost optimization
- Supporting the application design
- Creating a migration plan

Success Measurements

- **Business measurements of success**
  - Total Cost of Ownership (TCO)
  - Return on Investment (ROI)
  - Development agility (time from code to production)
  - Key Performance Indicators (KPI)
- **Technical measurements of success (SLIs)**
  - Service availability
  - Service response times
  - Error rate
  - Mean time to recovery (MTTR)

Success Measurements

Relevant to users

- Service availability
- Service response time
- Service error rate

Irrelevant to users

- Server availability
- Server CPU utilization
- Database errors

Design Decision Trade-offs

Some decisions are based on a priority.

- Example: Your organization may place very high priority on security. Your organization also wants to drive agility.

> The solution's design may sacrifice some agility in favor of security by e.g., introducing guardrails and policies.

Design Decision Trade-offs

Some decisions are just implicitly better.

- Example: Your organization wants to set up a development environment that is expected to be used only during business days.

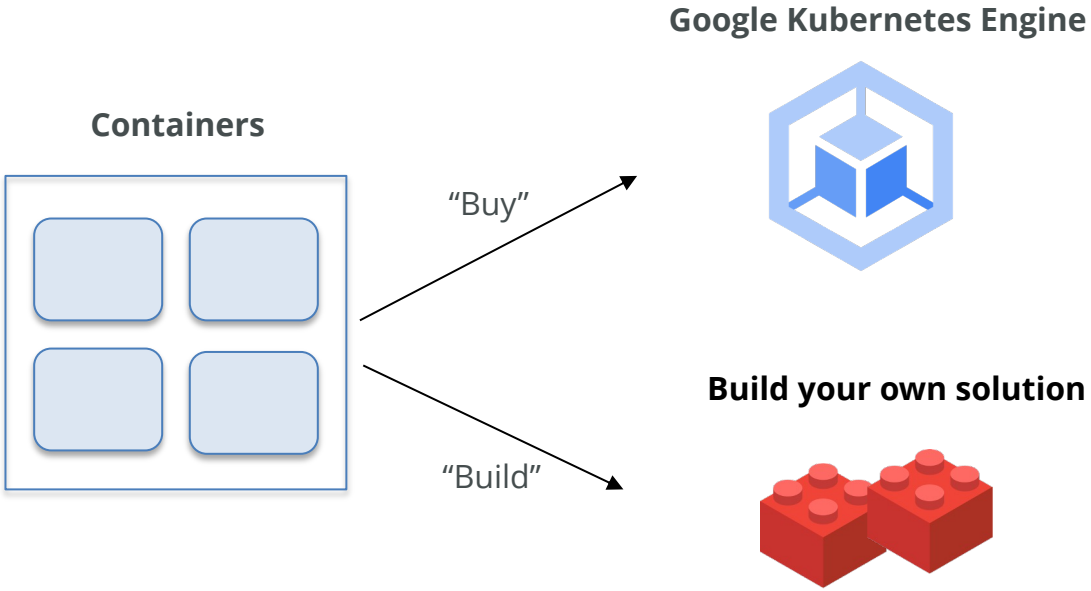
> A solution may include always-on VMs. Another solution may include VMs with a shutdown schedule. Both solutions meet the requirements, but one is cheaper.

Segment 6: Designing a solution infrastructure that meets business requirements

Objectives

- Business use cases and strategies
- Success measurements
- Buy vs build
- Movement of data and external integrations
- Cost optimization
- Supporting the application design
- Creating a migration plan

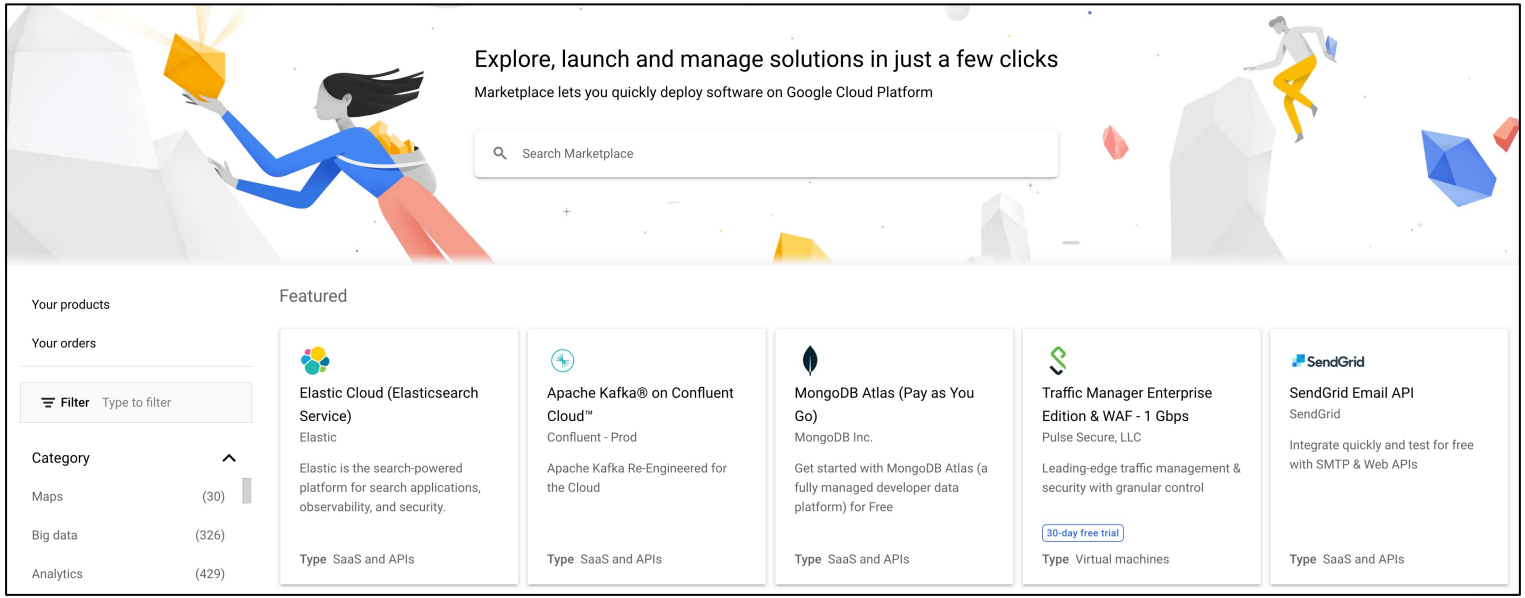
Buy vs. Build


Containers
"Buy"
Google Kubernetes Engine
"Build"
Build your own solution

Where to “Buy” from Partners

Google Cloud Marketplace

The marketplace lets you quickly deploy functional software packages that run on Google Cloud, including Software-as-a-Service (SaaS)

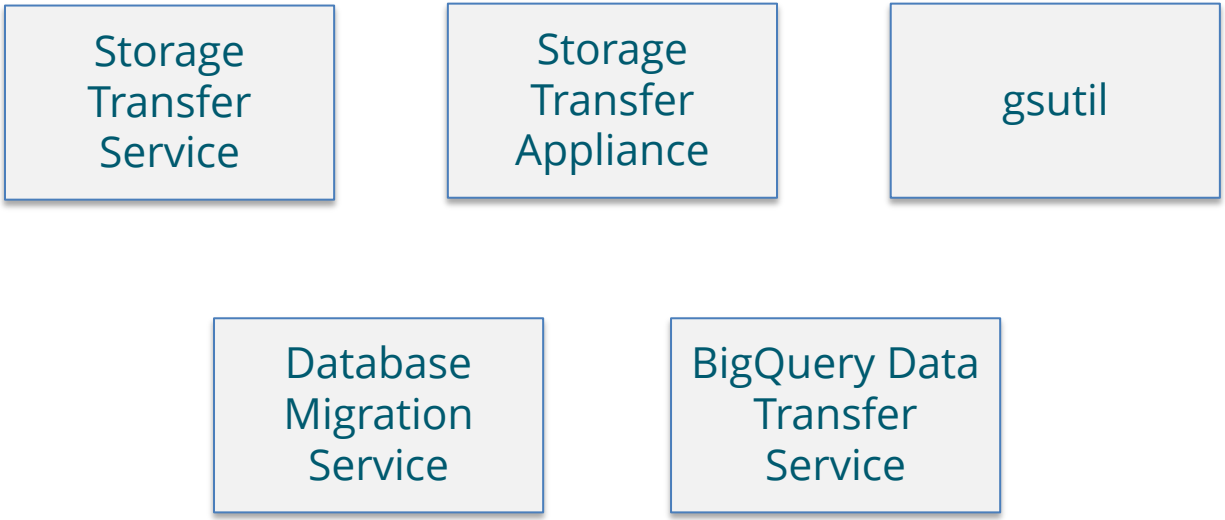


Segment 6: Designing a solution infrastructure that meets business requirements

Objectives

- Business use cases and strategies
- Success measurements
- Buy vs build
- Movement of data and external integrations
- Cost optimization
- Supporting the application design
- Creating a migration plan

Movement of Data: Options



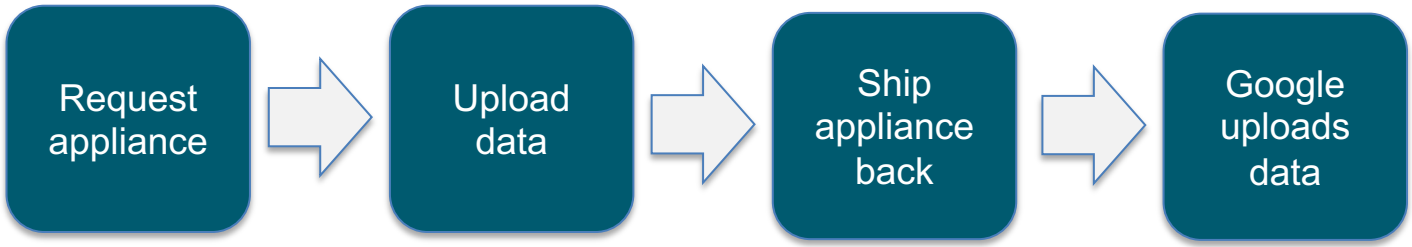
Movement of Data: Storage Transfer Service

Transfer data securely between **object** and **file** storage across Google Cloud, AWS, Azure, and on-premises.

- Encrypts data in transit
- Uses checksums to perform data integrity checks
- Does incremental transfer (move files/objects that are new, updated, or deleted since last transfer)
- Preserves object and file metadata during transfer
- Can set up a repeating schedule for transferring data

Movement of Data: Storage Transfer Appliance

- High-capacity, ruggedized, tamper-resistant storage device
- Ship your data to a Google upload facility
- Data is encrypted with AES 256 encryption
- Data is uploaded to Cloud Storage
- Available in two sizes: 100 TB and 480 TB



Movement of Data: Transferring Large Datasets

- Transfer Service for on-premises can be used to transfer large amounts of data: billions of files and hundreds of TBs of data in a single transfer. Network connections in the tens of Gbps.
- **Data Catalog** can be used to organize data into logical groupings that are moved and used together.
- If not enough bandwidth to meet project deadline: use **Transfer Appliance** for offline transfer.

Movement of Data: Transferring Small Datasets

- For small transfers (<1TB and enough bandwidth), can use **gsutil** tool
  - For multi-threaded transfers, use **gsutil –m**
  - For a single large file, use **Composite transfers**

Deciding What to Use

Data Source	Scenario	Product
AWS or Azure	Any	Storage Transfer Service
Cloud Storage (to Cloud Storage, different bucket)	Any	Storage Transfer Service
On-premises	Enough bandwidth to meet deadline, less than 1 TB of data	gsutil
On-premises	Enough bandwidth to meet deadline, but more than 1 TB of data	Storage Transfer Service for on-premises data
On-premises	Not enough bandwidth to meet deadline	Transfer Appliance

Movement of Data: Database Migration Service

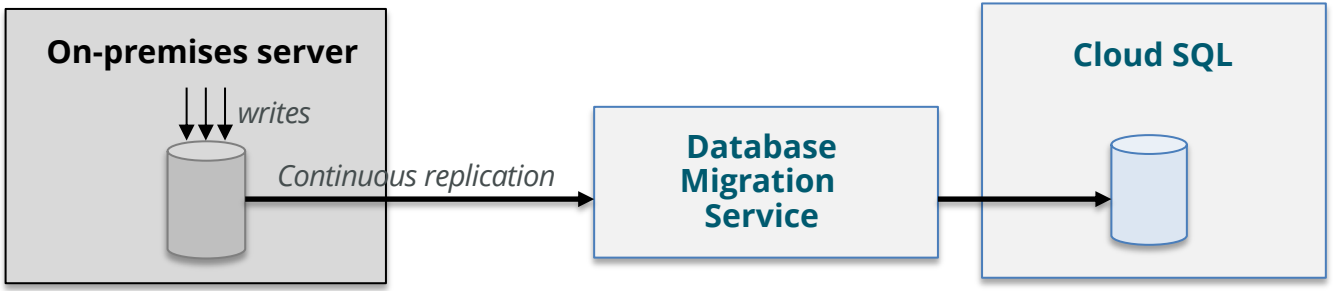
Migrate databases to Cloud SQL

- Can migrate from on-premises, GCE, and other clouds
- Available for MySQL and PostgreSQL*

*SQL Server and Oracle in preview

Movement of Data: Database Migration Service

- Serverless and guided experience
- Replicate data continuously for minimal downtime migrations



Movement of Data: BigQuery Data Transfer Service

Automates data movement into BigQuery on a schedule

- Currently can only be used to transfer data into BigQuery
- Supported sources:
  - Cloud Storage
  - Amazon S3
  - Teradata
  - Amazon Redshift
  - Google SaaS apps (Google Ads, Google Play, etc.)
  - Several third-party transfers available in Google Cloud Marketplace

Segment 6: Designing a solution infrastructure that meets business requirements

Objectives

- Business use cases and strategies
- Success measurements
- Buy vs build
- Movement of data and external integrations
- Cost optimization
- Supporting the application design
- Creating a migration plan

Cost Optimization Best Practices

Leverage billing and cost management tools

- Organize and structure costs
- Analyze billing reports
- Use labels to attribute costs back to departments/teams
- Use quotas, budgets, and alerts to closely monitor cost trends and forecast costs over time

Cost Optimization Best Practices

- Don’t pay for resources you don’t use
  - Identify idle VMs (tip: use **Recommender** service and the Idle Resource Recommender)
  - Schedule VMs to auto start and stop
- Rightsize VMs
  - Leverage custom machine types
  - Apply machine type recommendations
- Leverage **Preemptible**/**Spot** VMs with fault-tolerant workloads

Cost Optimization Best Practices

Optimize Cloud Storage costs

- Leverage storage classes
- Leverage lifecycle policies

Storage class	Minimum duration	Typical monthly availability
Standard Storage	None	>99.99% in multi-regions and dual-regions 99.99% in regions
Nearline Storage	30 days	99.95% in multi-regions and dual-regions 99.9% in regions
Coldline Storage	90 days	99.95% in multi-regions and dual-regions 99.9% in regions
Archive Storage	365 days	99.95% in multi-regions and dual-regions 99.9% in regions

Cost Optimization Best Practices

Tune BigQuery

- Enforce controls to limit query costs
- Use partitioning and clustering
- Checking for unnecessary streaming inserts (use batch loading instead, it’s free)
- Use Flex Slots

Cost Optimization Best Practices

- Optimize networking costs
  - Identify “top talkers” and optimize regional and intercontinental network egress
  - Filter out logs you don’t need in Cloud Logging and enable sampling, if possible, for VPC Flow Logs and Cloud Load Balancing
- Leverage committed use discounts
  - Ideal for workloads with predictable resource needs, available as 1- or 3-year term(s).

Segment 6: Designing a solution infrastructure that meets business requirements

Objectives

- Business use cases and strategies
- Success measurements
- Buy vs build
- Movement of data and external integrations
- Cost optimization
- Supporting the application design
- Creating a migration plan

Application Design Considerations

- Fault-tolerance
- Performance
- Security & Compliance
- Portability

System Design Considerations

- Zones and Regions
  - Deploy over multiple regions
  - Select regions based on geographic proximity
- Security
  - Use Organization Policy Service to enforce guardrails
  - Cloud IAM and least privilege
  - Data encryption at rest and in-transit
- Scalability
  - MIGs to support VM management
  - Pod autoscalers
  - Managed, serverless services

System Design Considerations

- Compute Resources
  - Choose compute platform based on technical requirements of the workload, lifecycle automation processes, regionalization, and security
- Network
  - Choose VPC topology to support application communication and security requirements
  - Choose hybrid or multicloud connectivity to support external integrations
- Storage Resources
  - Choose appropriate storage type based on data type and requirements

Segment 6: Designing a solution infrastructure that meets business requirements

Objectives

- Business use cases and strategies
- Success measurements
- Buy vs build
- Movement of data and external integrations
- Cost optimization
- Supporting the application design
- Creating a migration plan

Migration Phases

Assess

Thorough discovery of existing environment

Identifying app dependencies and requirements

TCO Calculations

Performance benchmarks

Plan

Foundational cloud infrastructure

Identity management

Organization and project structure

Networking

Deploy

Implement and execute a deployment process

Move workloads

Refine cloud infrastructure

Optimize

Adopt cloud-native Technologies

Scalability, DR

Cost optimization

Training

AI/ML and insights

Pearson

Good Candidates to Migrate First

- Not business critical
- Requires minimal app changes
- Doesn’t need large volume of data
- No strict compliance requirements
- Doesn't require third-party proprietary licenses

Migrating Applications: Anthos

Use **Migrate to Containers** to convert VM-based workloads into containers in GKE, Anthos, or Cloud Run

Source can be:

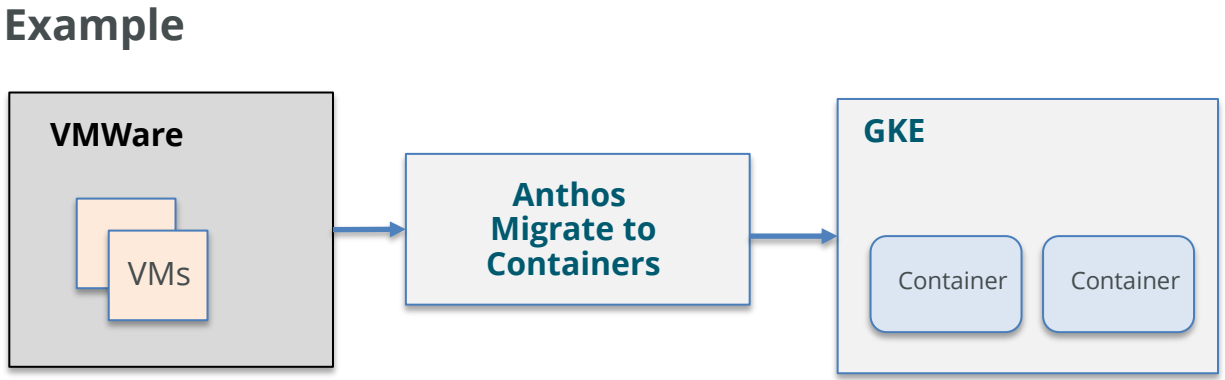
- VMware on-premises
- AWS
- Azure
- Compute Engine

Destination can be:

- GKE and Autopilot clusters
- Anthos
- Anthos on bare metal
- Cloud Run

Migrating Applications: Anthos

Use **Migrate to Containers** to convert VM-based workloads into containers in GKE, Anthos, or Cloud Run



Migrating Applications: Migrate for Compute Engine

- Enables ”lift and shift” migrations, with minor automatic modifications
- Uses data replication technology for disk data

Authentication for On-Premises Workloads

Service Account Keys

- RSA key pairs
- Lets you authenticate as the service account by having access to the private key
- User-managed key pairs are a security risk

Workload Identity Federation

- Identity federation with AWS, Azure, or any identity provider that supports OpenID Connect / SAML 2.0.
- Use IAM to grant external identities IAM roles, including ability to impersonate service accounts
- No need to maintain service account keys

Managing Dependencies

- During the “Assess” phase of the migration:
  - Build a comprehensive inventory of applications, including supporting services
  - Catalog applications according to properties and dependencies
  - Map and document all dependencies, including technical requirements
- During migration:
  - Ensure firewall and routing rules allow traffic to all dependencies
  - Ensure connection strings and URLs are updated
  - Migrate apps with fewest number of dependencies first

Demo: Designing a Solution Infrastructure that Meets Business Requirements

Business Requirements

i. Provisioning of cloud infrastructure resources must be auditable
ii. Changes to cloud infrastructure must go through a review process that minimizes impact to development velocity
iii. Data cannot traverse the public internet
iv. Data must be encrypted in transit and at rest
v. A failure in one part of the system should not bring down the entire system

Demo: Designing a Solution Infrastructure that Meets Business Requirements

Business Requirements

i. Provisioning of cloud infrastructure resources must be auditable
ii. Changes to cloud infrastructure must go through a review process that minimizes impact to development velocity
iii. Data cannot traverse the public internet
iv. Data must be encrypted in transit and at rest
v. A failure in one part of the system should not bring down the entire system

Technical Requirements

i. Deployment Manager, Cloud Source Repositories
ii. Code review and approvals must be in place for infrastructure provisioning. The review process should be decentralized and owned by each team
iii. All API access should be private and use of external IP addresses restricted
iv. All API communications over HTTPS, enforced SHA-256 encryption on all data stores
v. Microservices architecture

Questions Breakdown

You are responsible for planning a migration of your company's workloads to Google Cloud. What actions should you do first?

A. Run a thorough discovery and assessment of the current environment, identify app dependencies, and calculate total cost of ownership (TCO).

B. Set up the foundational infrastructure on GCP, decide which workload to migrate first, and run a proof-of-concept.

C. Run a performance benchmark for your existing applications, codify the target infrastructure, and deploy a landing zone.

D. Define the project structure on GCP, educate the team on cloud-native technologies, and modernize all applications ahead of the migration.

Questions Breakdown

You are responsible for **planning a migration** of your company's workloads to Google Cloud. What actions should you do **first**?

A. Run a thorough discovery and assessment of the current environment, identify app dependencies, and calculate total cost of ownership (TCO).

B. Set up the foundational infrastructure on GCP, decide which workload to migrate first, and run a proof-of-concept.

C. Run a performance benchmark for your existing applications, codify the target infrastructure, and deploy a landing zone.

D. Define the project structure on GCP, educate the team on cloud-native technologies, and modernize all applications ahead of the migration.

Questions Breakdown

Your company wants to migrate an on-premises MySQL deployment to a managed offering on Google Cloud. You need to minimize downtime and performance impact during the migration. Which approach should you recommend?

A. Use MySQL tools to create a dump of the existing server, then shut down the server, upload the dump file to Cloud Storage, and load it into a new Cloud SQL instance.

B. Provision a Google Compute Engine (GCE) virtual machine and install MySQL. Set up replication on the on-premises server until cut-over.

C. Provision a Cloud SQL for MySQL instance. Set up asynchronous replication on the on-premises database with the Cloud SQL instance as the replica until cut-over.

D. Use the Database Migration Service to set up continuous data replication until cut-over.

Questions Breakdown

Your company wants to migrate an on-premises **MySQL** deployment to a **managed offering** on Google Cloud. You need to minimize downtime and performance impact during the migration. Which approach should you recommend?

A. Use MySQL tools to create a dump of the existing server, then shut down the server, upload the dump file to Cloud Storage, and load it into a new Cloud SQL instance.

B. Provision a Google Compute Engine (GCE) virtual machine and install MySQL. Set up replication on the on-premises server until cut-over.

C. Provision a Cloud SQL for MySQL instance. Set up asynchronous replication on the on-premises database with the Cloud SQL instance as the replica until cut-over.

D. Use the Database Migration Service to set up continuous data replication until cut-over.

Questions Breakdown

You are responsible for migrating VMware-based workloads to Google Cloud. Your company's CTO decided to adopt container technologies and modernize applications away from VMs and into native containers. You need to plan a solution for migrating VMs to GCP while minimizing downtime. Which approach should you recommend?

A. Create a virtual machine image from each existing VM and use a third-party tool to convert them to Docker images. Deploy the Docker images into a Google Kubernetes Engine (GKE) cluster.

B. Upload all application source codes to Cloud Source Repositories. Use Cloud Build to create a continuous deployment pipeline that deploys the source code into Cloud Run.

C. Run a discovery exercise to identify existing workloads and assess migration readiness. Configure the deployment environment for the migrated containers, and use Anthos Migrate to Containers.

D. Use Migrate for Compute Engine to first migrate applications to virtual machines on GCP. Use Anthos Migrate to Containers to migrate all VMs to containers.

Questions Breakdown

You are responsible for migrating **VMware-based workloads to Google Cloud**. Your company's CTO decided to **adopt container** technologies and modernize applications **away from VMs** and into native containers. You need to plan a solution for migrating VMs to GCP while **minimizing downtime**. Which approach should you recommend?

A. Create a virtual machine image from each existing VM and use a third-party tool to convert them to Docker images. Deploy the Docker images into a Google Kubernetes Engine (GKE) cluster.

B. Upload all application source codes to Cloud Source Repositories. Use Cloud Build to create a continuous deployment pipeline that deploys the source code into Cloud Run.

C. Run a discovery exercise to identify existing workloads and assess migration readiness. Configure the deployment environment for the migrated containers, and use Anthos Migrate to Containers.

D. Use Migrate for Compute Engine to first migrate applications to virtual machines on GCP. Use Anthos Migrate to Containers to migrate all VMs to containers.

Segment 7: Designing a solution infrastructure that meets technical requirements

Objectives

- Designing for high availability and failover
- Elasticity of cloud resources, quotas and limits
- Designing for scalability
- Designing for performance

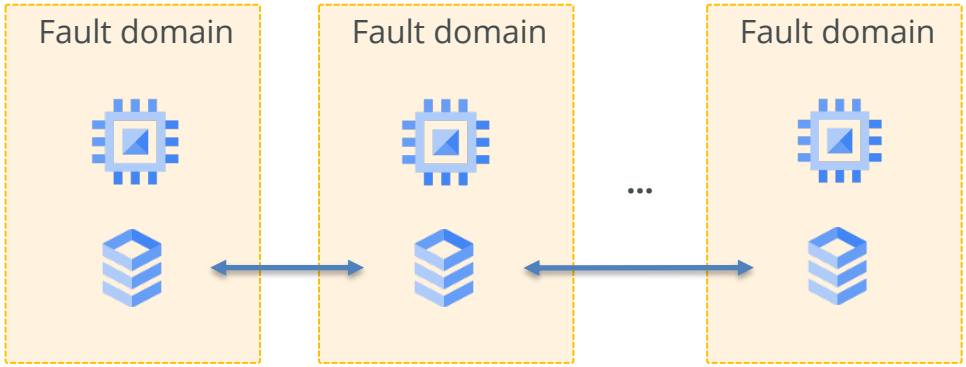
Segment 7: Designing a solution infrastructure that meets technical requirements

Objectives

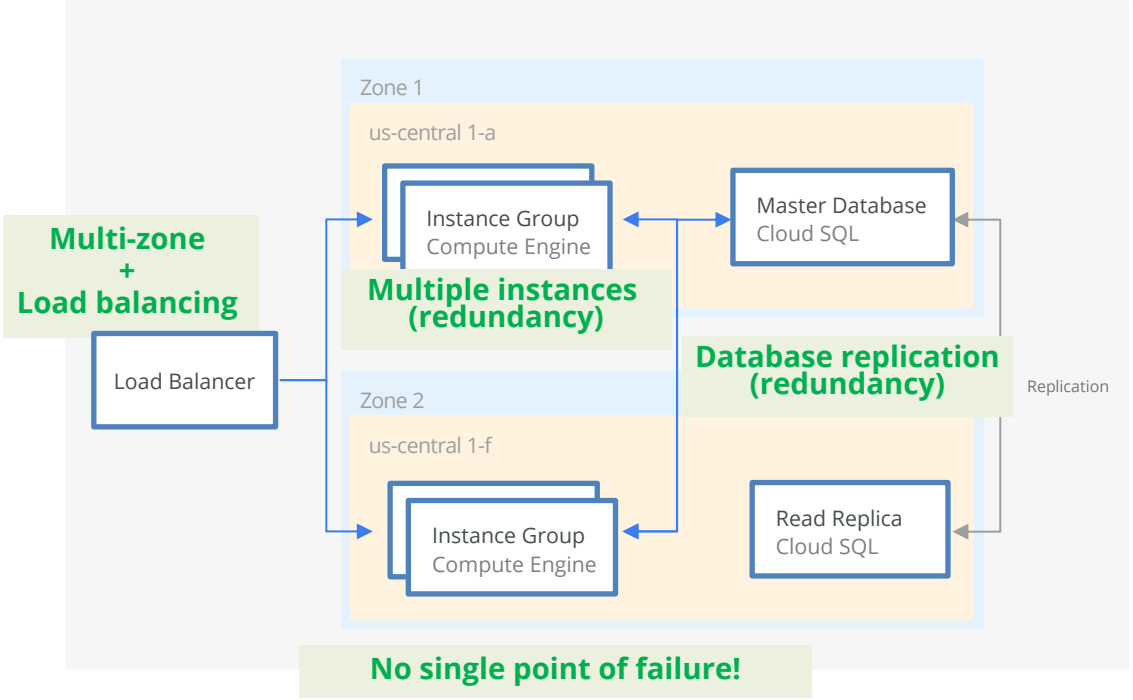
- Designing for high availability and failover
- Elasticity of cloud resources, quotas and limits
- Designing for scalability
- Designing for performance

High availability (HA) Design Principles

- Create redundancy
- Eliminate single points of failure
- Replicate resources across multiple fault domains



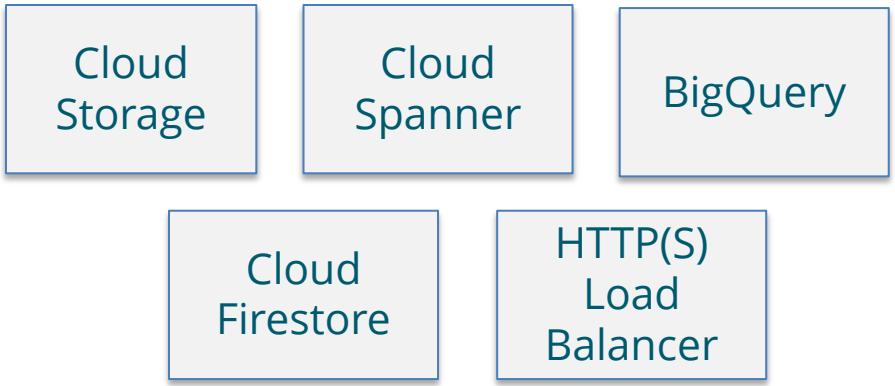
Multi-zone HA Architecture



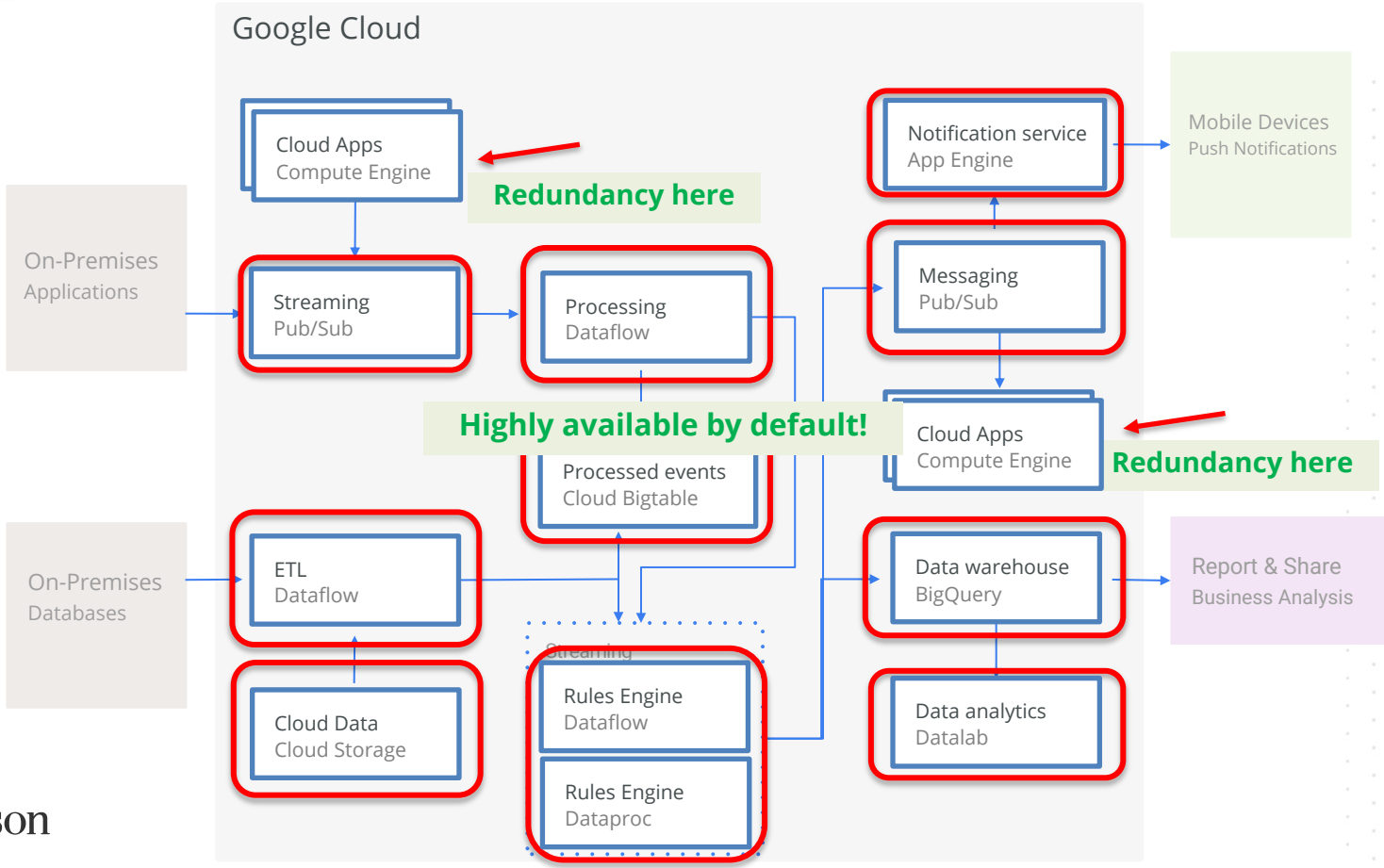
Pearson

Leverage Managed Services

- Most managed services are regional (i.e., resilient against zone outages)
- Some are global or multi-region (i.e., resilient against regional outages), for example:



High Availability Design



Pearson

Availability of GCP Services

Service	Scope	HA design
Pub/Sub	Global topics, regional messages	Resilient against zonal outages with synchronous message replication.
Dataflow	Zonal	Can avoid data loss in a zone or region outage by reusing the same subscription to the Pub/Sub topic (Dataflow only acks the messages in Pub/Sub if they were persistent in the destination) and failover to another Dataflow job in another zone or region.
Dataproc	Regional control plane, zonal clusters	In a zonal outage, you can recreate a new instance of the cluster in another zone. Once cluster is available, data processing can resume.

Availability of GCP Services

Service	Scope	HA design
App Engine	Regional	App runs redundantly across all of the zones within the selected region.
Cloud Functions	Regional	Data and traffic are automatically load balanced across zones within a region. Both control and data planes are resilient against zonal failures.
Cloud Run	Regional	Data and traffic are automatically load balanced across zones within a region. Both control and data planes are resilient against zonal failures.
External IP addresses	Regional or Global	Resilient against zone outages (Regional) and region outages (Global)

Availability of GCP Services

Service	Scope	HA design
Google Kubernetes Engine	Zonal or regional	Mitigation of regional outage requires deployment across multiple regions, which is not offered as a built-in capability. However, multiple regional clusters can be deployed and traffic controlled with multi-cluster ingress.
Compute Engine VMs	Zonal	For a high-availability VM deployment, use Compute Engine instance templates and/or managed instance groups to run and scale identical VM instances in multiple zones.
Persistent Disks	Zonal or Regional (with synchronous inter-zone replication)	In the event of an outage in your virtual machine's zone, you can force attach a regional Persistent Disk to a VM instance in the disk's secondary zone.

Availability of GCP Services

Service	Scope	HA design
Cloud Firestore	Regional or Multi-Regional	Regional availability SLA is 99.99% and Multi-Region is 99.999%. Strong consistency.
Cloud BigQuery	Regional or Multi-Regional	Data written to BigQuery is written to both the primary and secondary zones.
Cloud Bigtable	Zonal clusters	To configure a Bigtable instance for HA, create a new app profile that uses multi-cluster routing. Deploy clusters in multiple zones and/or regions as needed. Failover will be automatic. Note: eventual consistency.

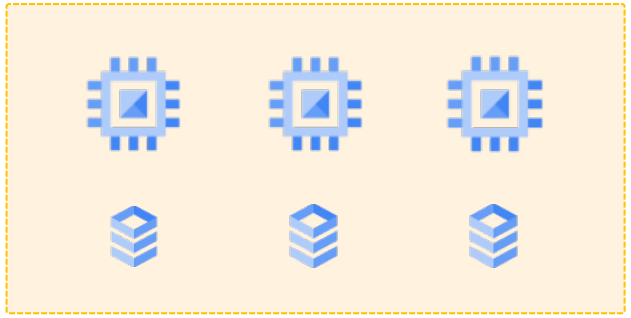
Segment 7: Designing a solution infrastructure that meets technical requirements

Objectives

- Designing for high availability and failover
- Elasticity of cloud resources, quotas and limits
- Designing for scalability
- Designing for performance

Elasticity

- Does the solution scale when busy, so that the service's availability and performance remain intact?
- Does the solution scale back when demand is low, so that infrastructure cost is reduced?



Design Strategies for Elasticity

- Autoscaling and load balancing
- Managed services (serverless)

Autoscaling and Load Balancing: Compute Engine VMs

Managed instance groups (MIGs)

- Specify an instance template
- Define an autoscaling policy
  - Target utilization metric
  - Schedules

Average CPU utilization
HTTP load balancing serving capacity, based on either utilization or requests per second
Cloud Monitoring metrics

Autoscaling and Load Balancing: GKE

Kubernetes Engine containers: GKE Autoscaling

- Pod autoscaling
- Cluster autoscaling

Autoscaling and Load Balancing

GKE Autoscaling: Cluster autoscaling

- Automatically resizes the number of nodes in a given node pool based on the demands of workloads.
- You specify a minimum and maximum size for the node pool
- If Pods are unschedulable because there are not enough nodes in the pool, cluster autoscaler will also add nodes.
- Cluster autoscaler will automatically balance nodes across availability zones (for regional clusters)

Quotas and Limits

GCP-enforced quotas

- Used to restrict how much of a particular shared Google Cloud resource you can use
- Enforced to protect community from unforeseen spikes in usage and overloaded services

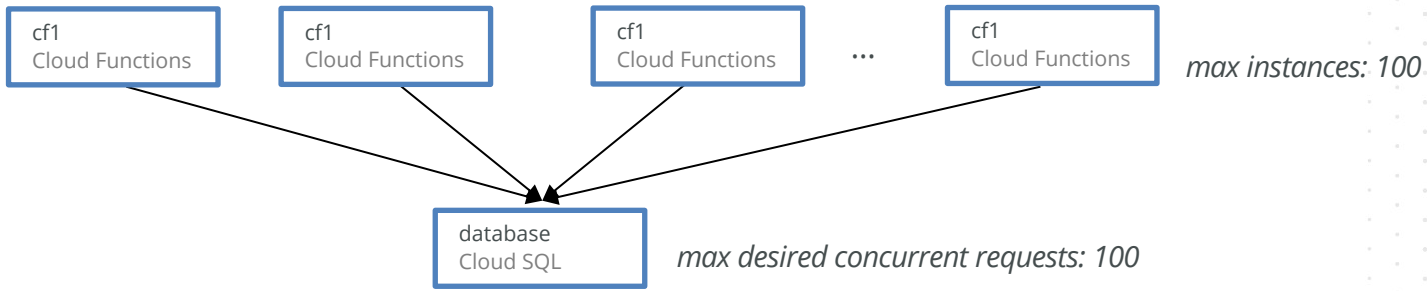
User-enforced quotas

- You can set your own limits on service usage to avoid unexpected bills

Quotas and Limits

Leverage rate-limiting tools to scale without hitting limits:

- Example: Limit concurrent connections to Cloud SQL database
- With Cloud Functions, use **max instances** setting to limit how many concurrent instances of the function are running and establishing database connections



Segment 7: Designing a solution infrastructure that meets technical requirements

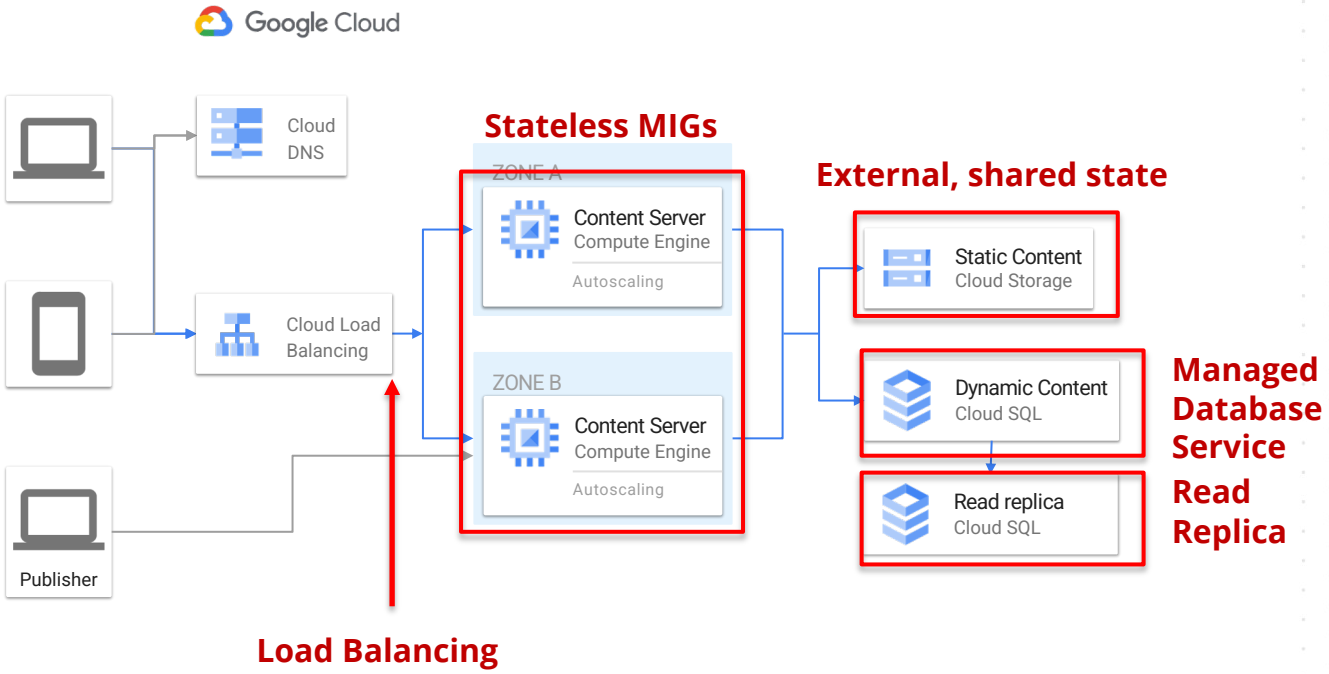
Objectives

- Designing for high availability and failover
- Elasticity of cloud resources, quotas and limits
- Designing for scalability
- Designing for performance

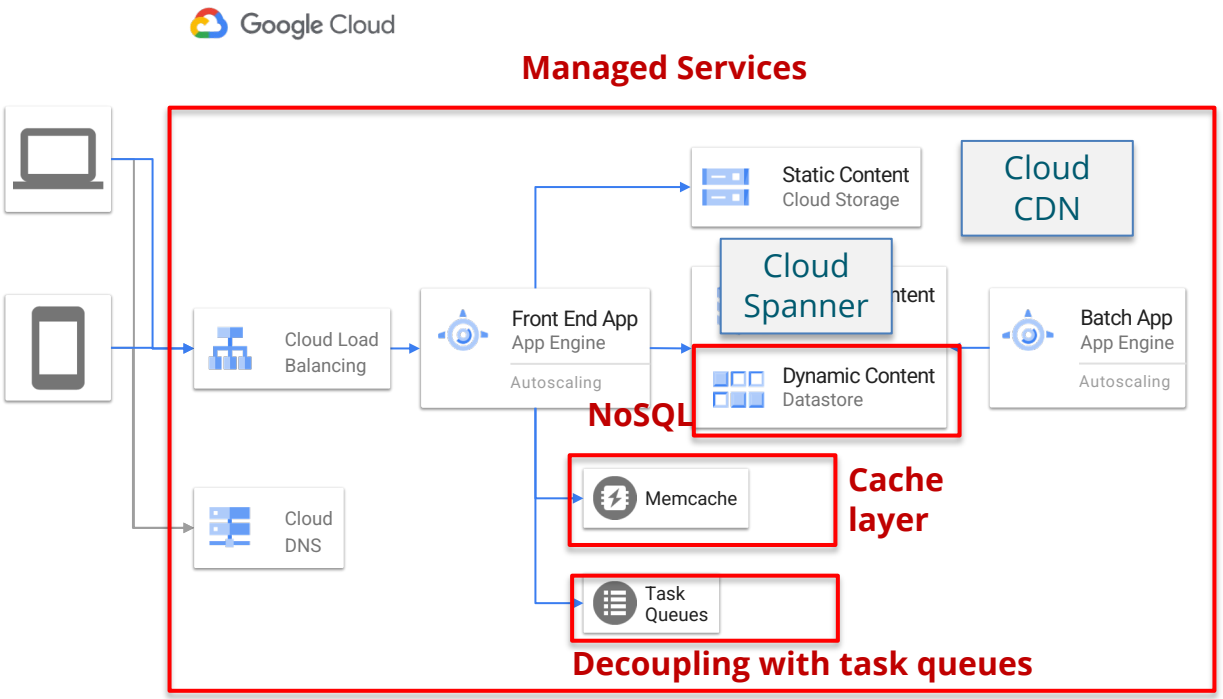
Scalability for Growth

- More than just autoscaling to cover temporary demand fluctuations.
- Think about scalable design patterns:
  - Aim for statelessness
  - Leverage serverless platform and scalable, managed services for consistent performance
  - Leverage Cloud Monitoring to make data-driven scaling decisions
  - Leverage native load balancing and multi-zone/multi-region architectures to withstand failures
  - Leverage CI/CD through native tools to help automate building and deploying apps (+ incorporate automated testing)
  - Leverage automation and loose coupling

Scalable Architecture Example #1



Scalable Architecture Example #2



More scaling opportunities?

Segment 7: Designing a solution infrastructure that meets technical requirements

Objectives

- Designing for high availability and failover
- Elasticity of cloud resources, quotas and limits
- Designing for scalability
- Designing for performance

Performance Optimization Best Practices

- Monitor and analyze performance
- Architect workloads for optimal resource placement
- Implement content caching
- Isolate read and write traffic
- Tune apps

Performance Efficiency Design Patterns

- Autoscale resources
- Use GPUs/TPUs
- Pick the appropriate machine family for the workload
- Cache frequently accessed data
- Consider the cost-performance tradeoff between SSD and HDD storage
- Choose appropriate capacity tiers for block and file storage
- Deploy application close to your users
- Leverage Cloud CDN

Performance Optimization: Identify Apps to Tune

Use tools to inspect and analyze performance:

- **Cloud Trace** and **OpenTelemetry**: helps you instrument code to identify latency and find bottlenecks in inter-service communications
- **Cloud Debugger**: helps you inspect and analyze production code behavior in real time without affecting its performance
- **Cloud Profiler**: helps you identify and address performance by continuously analyzing CPU and memory consumption

Demo: Designing a Solution Infrastructure that Meets Technical Requirements

Technical Requirements

i. Provisioning must be done through Infrastructure-as-Code process
ii. Code review and approvals must be in place for infrastructure provisioning. The review process should be decentralized and owned by each team
iii. All API access should be private and use of external IP addresses restricted
iv. All API communications over HTTPS, enforced SHA-256 encryption on all data stores
v. Microservices architecture

Design and Implementation

i. Provisioning must be done through Infrastructure-as-Code process
ii. Cloud Source Repositories, one source repo per product/service, pull request mechanism enforced
iii. Private Google Access, VPC Service Controls, Organization Service Policy constraints
iv. Cloud-first development leveraging Google's default encryption at rest and in transit. Anthos Service Mesh with mutual TLS on all communications.
v. Kubernetes, Anthos

Questions Breakdown

Your company has a web-based application hosted in a single data center. As the customer base grows, customers from distant locations often complain that the website is slow. Your company has decided to move the application to Google Cloud to benefit from its global footprint.

How should you design the solution to reduce latency for customers?

A. Deploy the application to a set of virtual machines and use DNS-based load balancing
B. Use zonal managed instance groups in different zones with a regional load balancer
C. Use regional managed instance groups in different regions with a global load balancer
D. Deploy the application to a regional App Engine instance with a global load balancer.

Questions Breakdown

Your company has a web-based application hosted in a **single data center**. As the customer base grows, customers from distant locations often complain that the **website is slow**. Your company has decided to move the application to Google Cloud to benefit from its **global footprint**.

How should you design the solution to **reduce latency** for customers?

A. Deploy the application to a set of virtual machines and use DNS-based load balancing

B. Use zonal managed instance groups in different zones with a regional load balancer

C. Use regional managed instance groups in different regions with a global load balancer

D. Deploy the application to a regional App Engine instance with a global load balancer.

Questions Breakdown

Your company has acquired another company that has a containerized web application running on-premises. You need to move the application to Google Cloud and redesign the solution to accommodate a larger number of users and scale automatically with usage. What should you do?

A. Host the application on Google Kubernetes Engine and enable Horizontal Pod Autoscaler and cluster autoscaling
B. Host the application on Google Kubernetes Engine and enable Vertical Pod Autoscaler and cluster autoscaling
C. Host the application on Compute Engine instances. Perform a load test and use Google Cloud's machine type recommender to identify the most appropriate machine type
D. Host the application on a managed instance group with an autoscaling policy. Use Google Cloud's managed instance group machine type recommender to identify the most appropriate machine type

Questions Breakdown

Your company has acquired another company that has a **containerized** web application running on-premises. You need to move the application to Google Cloud and redesign the solution to accommodate a **larger number of users** and **scale automatically** with usage. What should you do?

A. Host the application on Google Kubernetes Engine and enable Horizontal Pod Autoscaler and cluster autoscaling

B. Host the application on Google Kubernetes Engine and enable Vertical Pod Autoscaler and cluster autoscaling

C. Host the application on Compute Engine instances. Perform a load test and use Google Cloud's machine type recommender to identify the most appropriate machine type

D. Host the application on a managed instance group with an autoscaling policy. Use Google Cloud's managed instance group machine type recommender to identify the most appropriate machine type

Questions Breakdown

Your company is running a stateless web application on two Compute Engine instances in two availability zones. The application receives a lot of traffic during business hours and little traffic otherwise. During peak hours, several users are complaining that the application is slow and sometimes crashing. You need to redesign the solution to improve performance. What should you do?

A. Deploy the application to two more instances in a separate Google Cloud region

B. Deploy the application to an extra instance in a new availability zone. Configure startup and shutdown scripts so that the extra instance only runs during business hours.

C. Set up a Cloud Monitoring alert that triggers a Cloud Function to create a new instance if the average CPU utilization is high.

D. Create an instance template and deploy the application to a managed instance group with an autoscaling policy.

Questions Breakdown

Your company is running a **stateless** web application on two Compute Engine instances in two availability zones. The application receives **a lot of traffic during business hours** and little traffic otherwise. **During peak hours**, several users are complaining that the **application is slow and sometimes crashing**. You need to redesign the solution to **improve performance**. What should you do?

A. Deploy the application to two more instances in a separate Google Cloud region

B. Deploy the application to an extra instance in a new availability zone. Configure startup and shutdown scripts so that the extra instance only runs during business hours.

C. Set up a Cloud Monitoring alert that triggers a Cloud Function to create a new instance if the average CPU utilization is high.

D. Create an instance template and deploy the application to a managed instance group with an autoscaling policy.

Segment 8: Case studies solutioning – pen-and-paper exercises

Objectives

- Case Study Breakdown: EHR Healthcare
- Case Study Breakdown: Helicopter Racing League
- Case Study Breakdown: TerramEarth
- Case Study Breakdown: Mountkirk Games

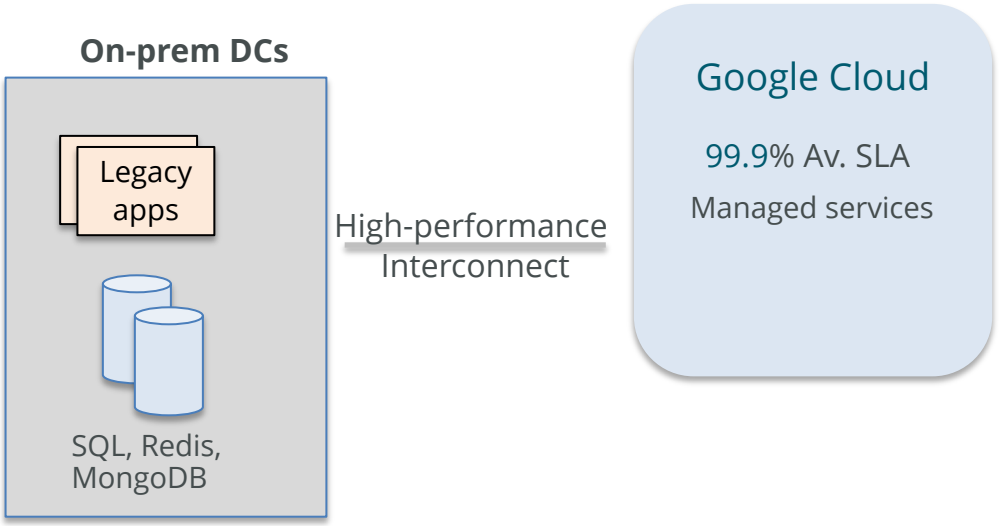
Segment 8: Case studies solutioning – pen-and-paper exercises

Objectives

- Case Study Breakdown: EHR Healthcare
- Case Study Breakdown: Helicopter Racing League
- Case Study Breakdown: TerramEarth
- Case Study Breakdown: Mountkirk Games

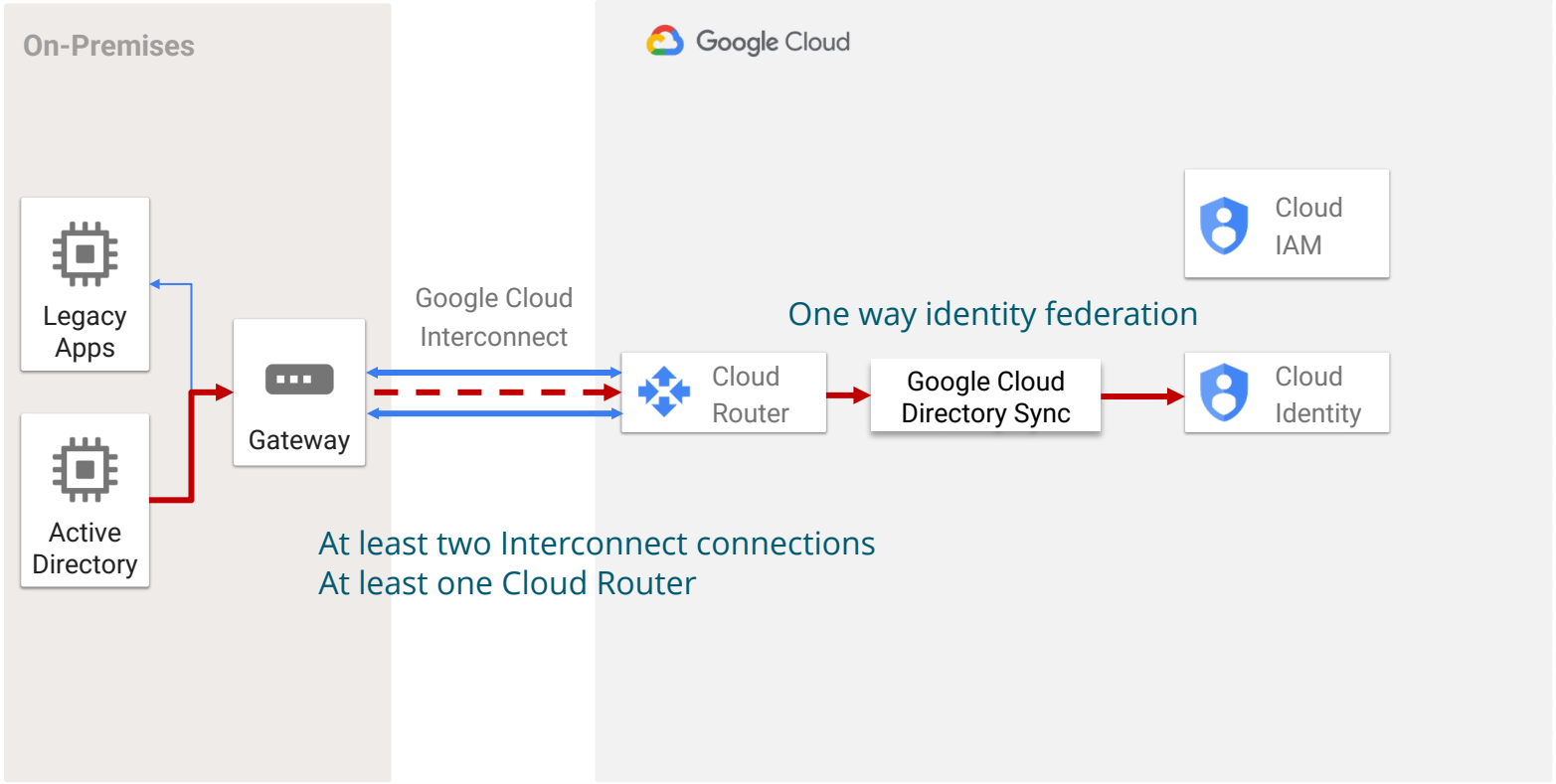
Exam Case Study: EHR Healthcare

Case Study Summary



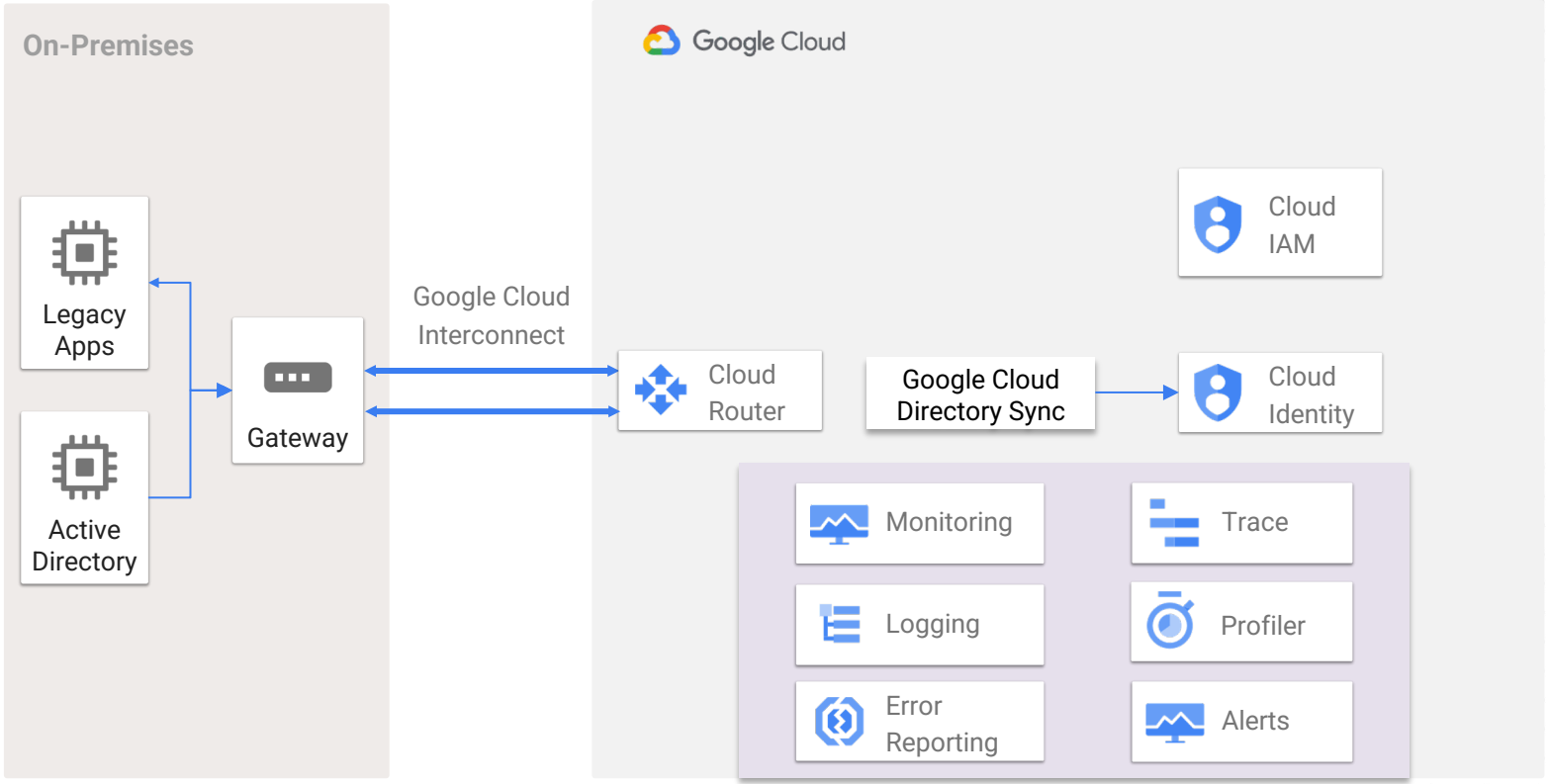
- SaaS Product, web-based, Kubernetes
- MySQL+MS SQL Server, Redis, MongoDB
- Hosted on multiple on-prem locations
- Legacy integrations with partners – remain on-prem
- Requirements: reduce latency, low infrastructure admin cost, centralized monitoring, high-performance interconnect.

Case Study Technical Solution: Hybrid Identity

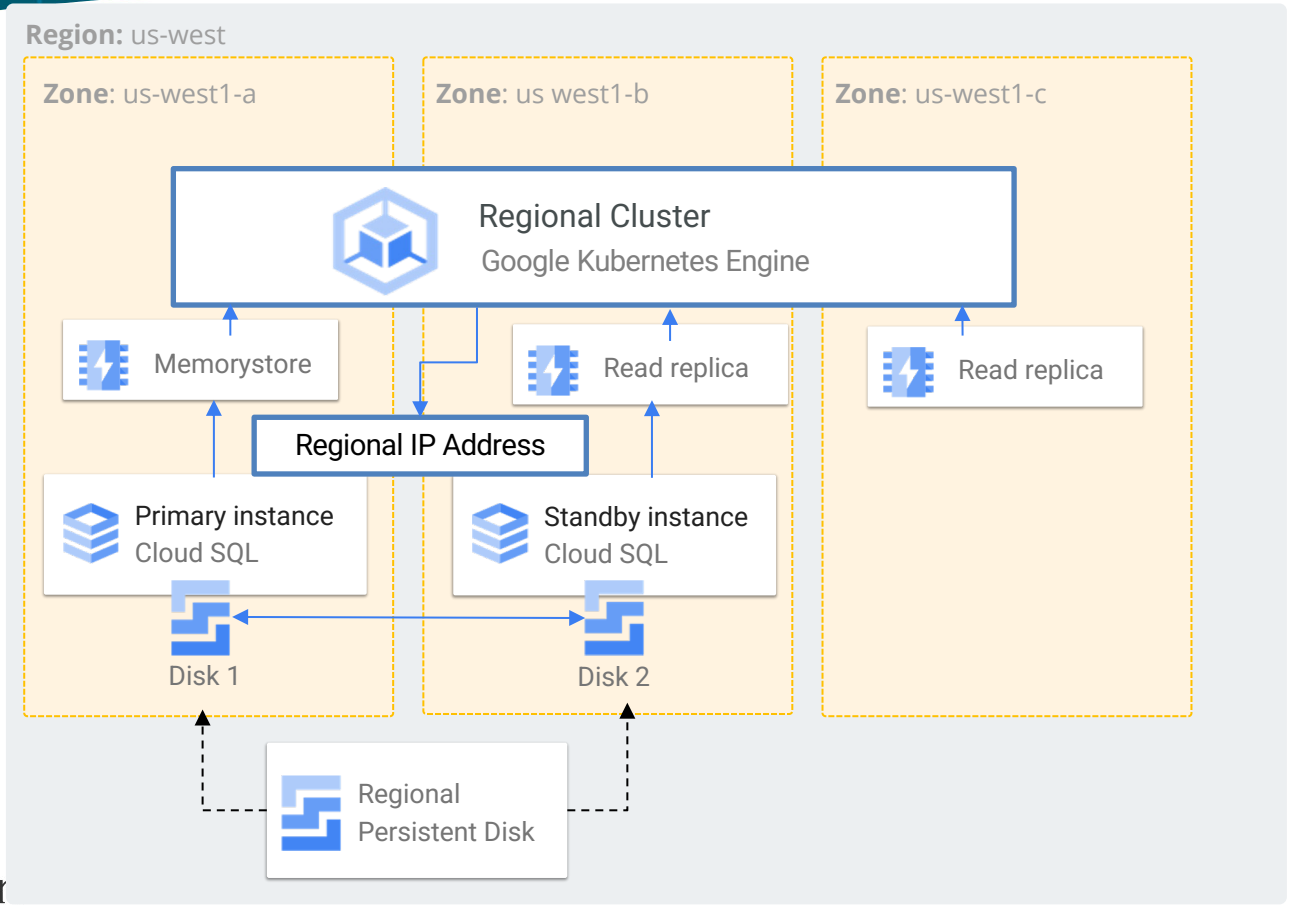


Pearson

Case Study Technical Solution: Monitoring

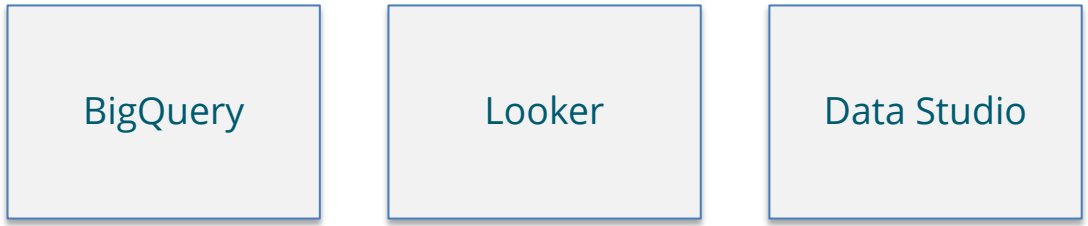


Case Study Technical Solution: Web Apps

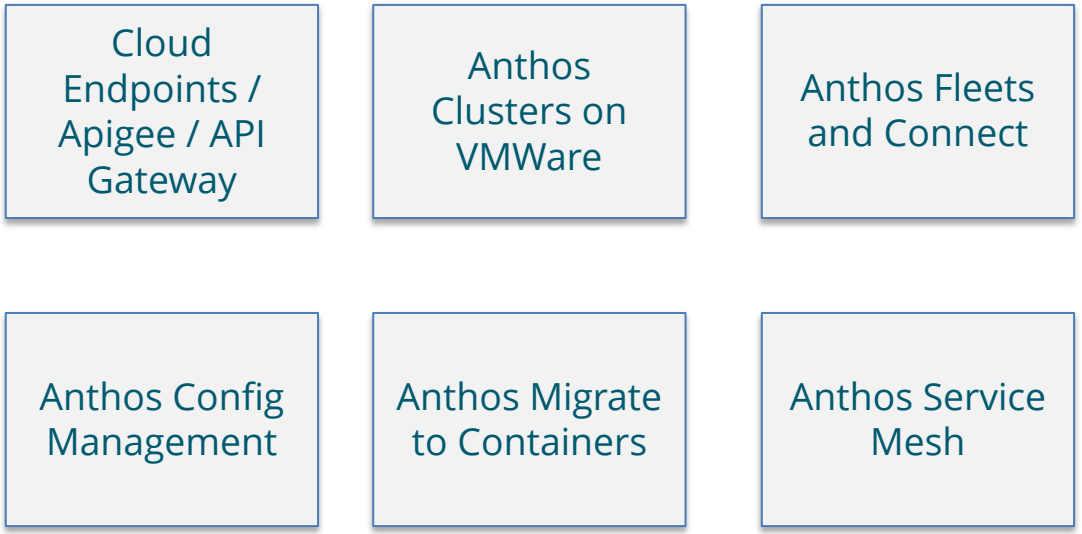


Case Study Solution: CI/CD Building Blocks

Case Study Solution: Analytics & Reports Building Blocks



Case Study Solution: Integrations Building Blocks



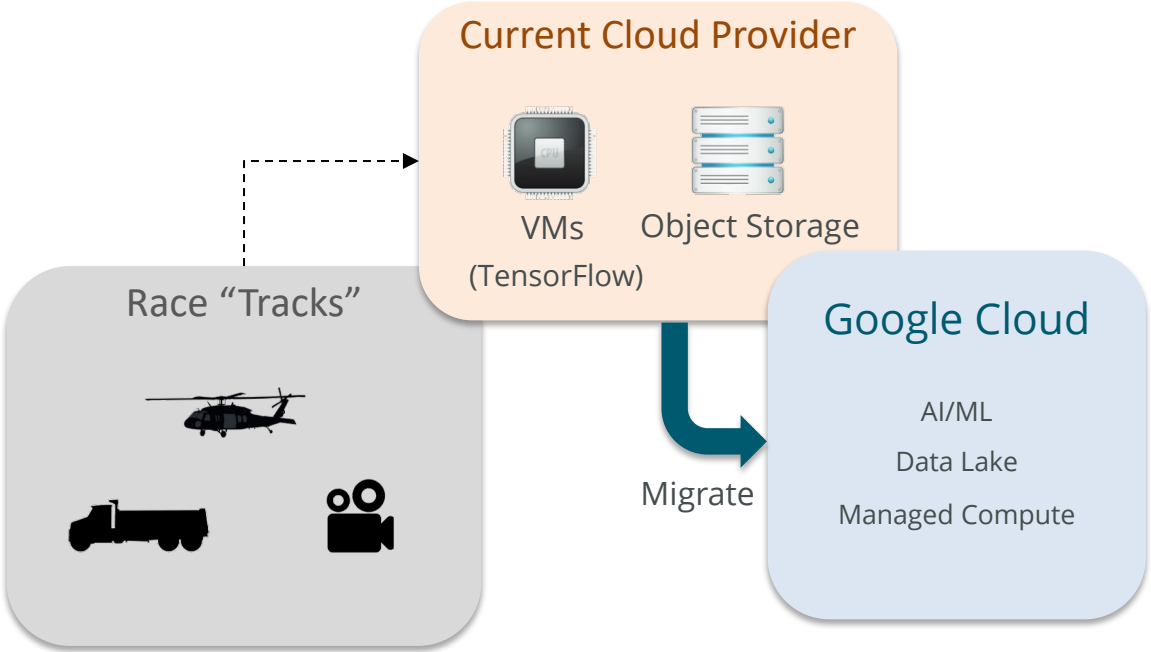
Segment 8: Case studies solutioning – pen-and-paper exercises

Objectives

- Case Study Breakdown: EHR Healthcare
- Case Study Breakdown: Helicopter Racing League
- Case Study Breakdown: TerramEarth
- Case Study Breakdown: Mountkirk Games

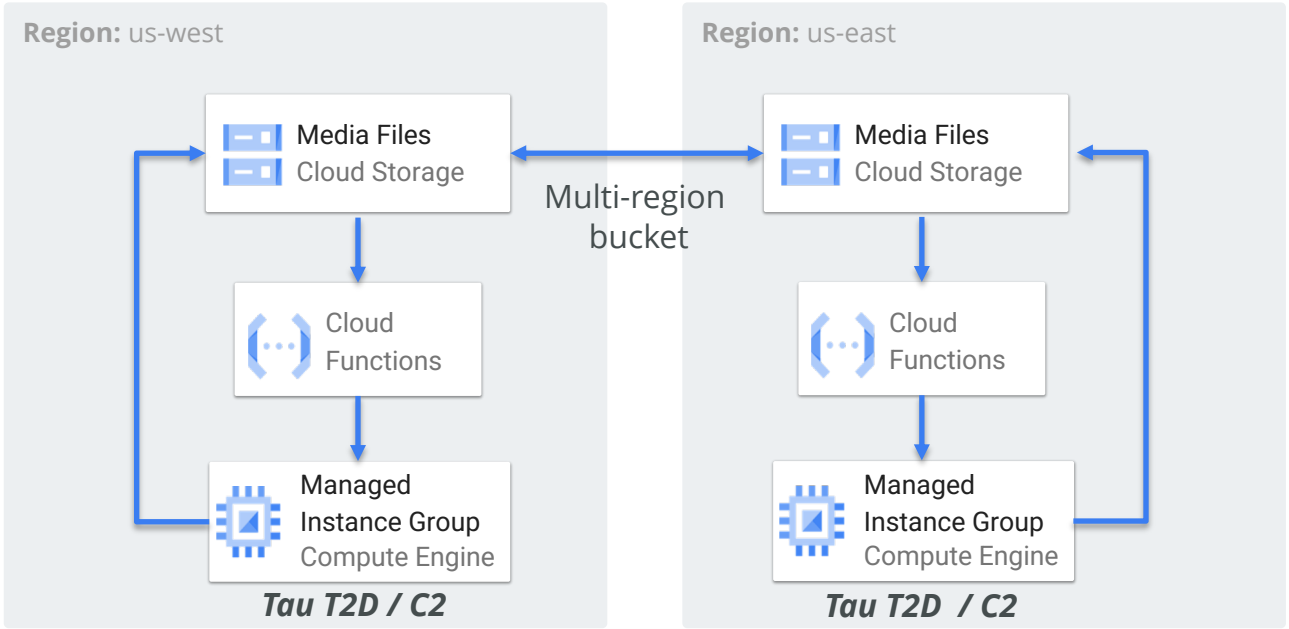
Exam Case Study: Helicopter Racing League

Case Study Summary



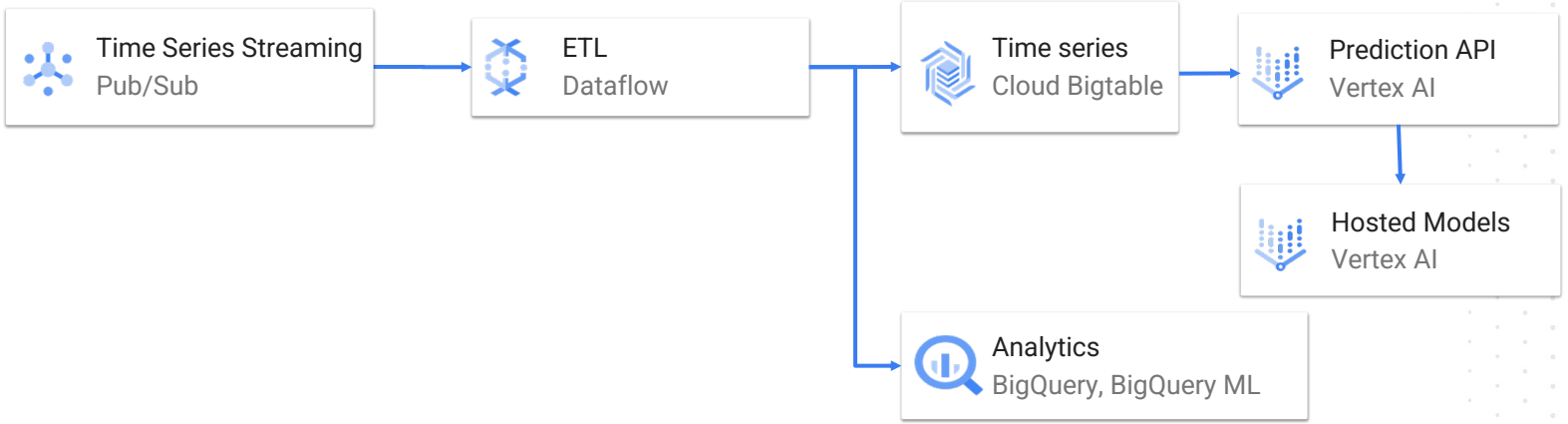
- Global event streaming, live telemetry
- AI/ML for predictions
- Requirements: Global availability, lower latency, real-time analytics, serve predictions to partners, low ops, improve prediction throughput and accuracy
- Data lake for large volumes of race data

Case Study Solution: Media Storage and Transcoding

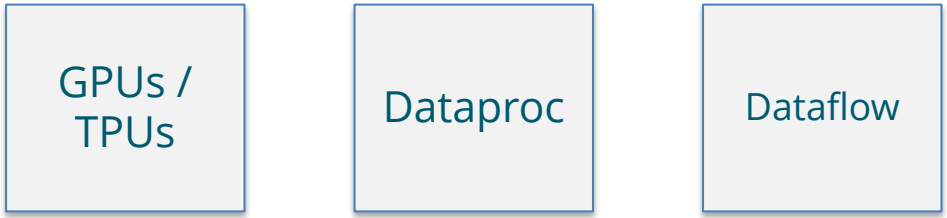


Also consider:
Transcoder API

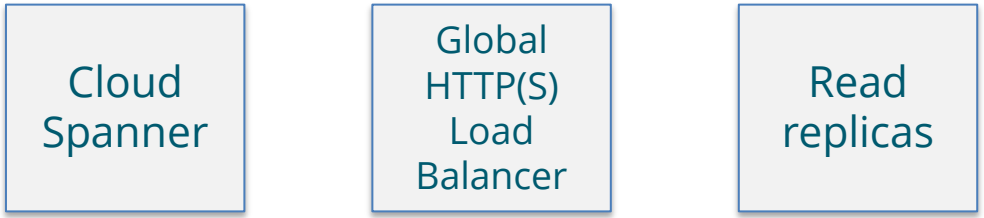
Case Study Solution: Telemetry and Analytics



Case Study Solution: Data Processing Building Blocks



Case Study Solution: Global Footprint Building Blocks



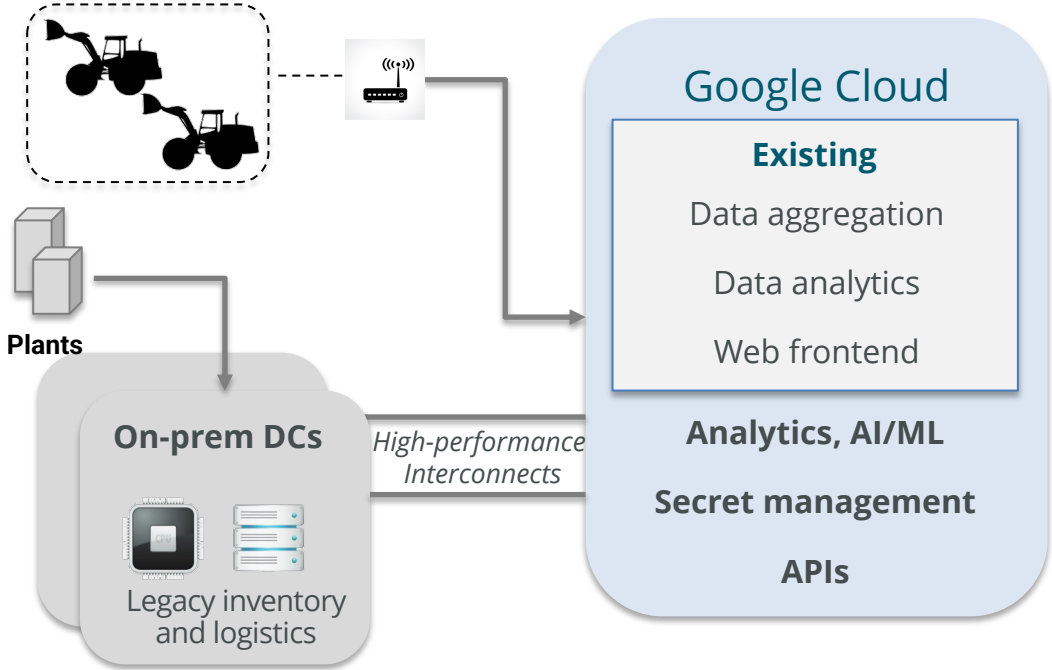
Segment 8: Case studies solutioning – pen-and-paper exercises

Objectives

- Case Study Breakdown: EHR Healthcare
- Case Study Breakdown: Helicopter Racing League
- Case Study Breakdown: TerramEarth
- Case Study Breakdown: Mountkirk Games

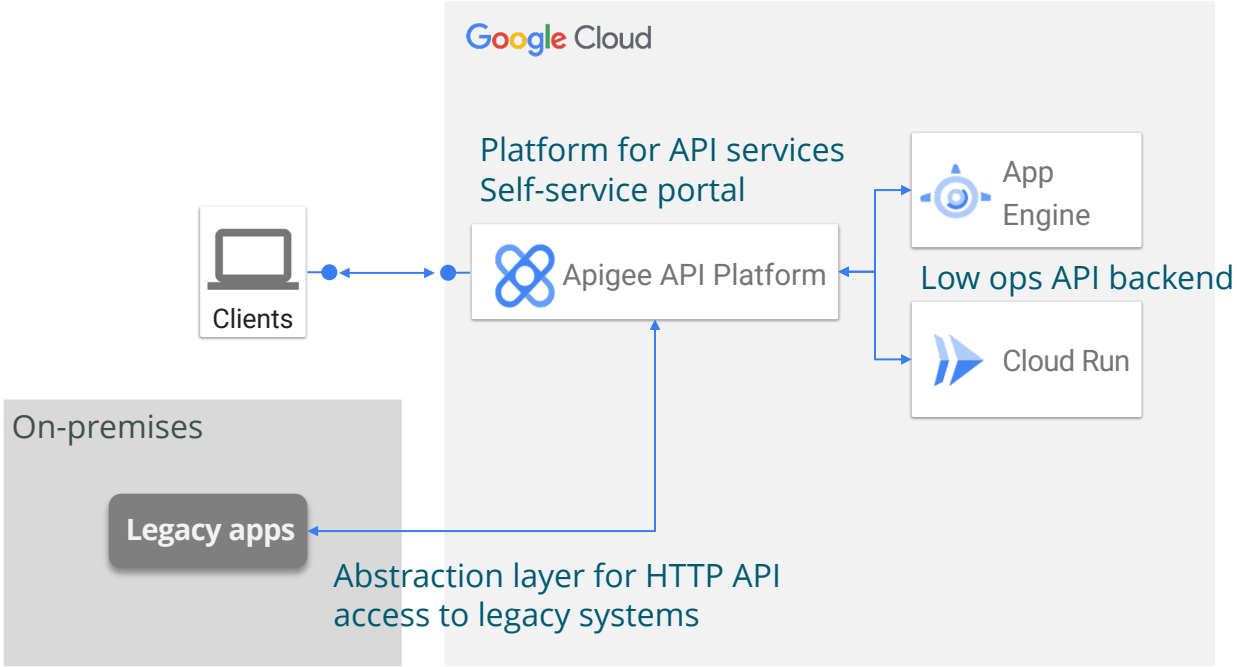
Exam Case Study: TerramEarth

Case Study Summary

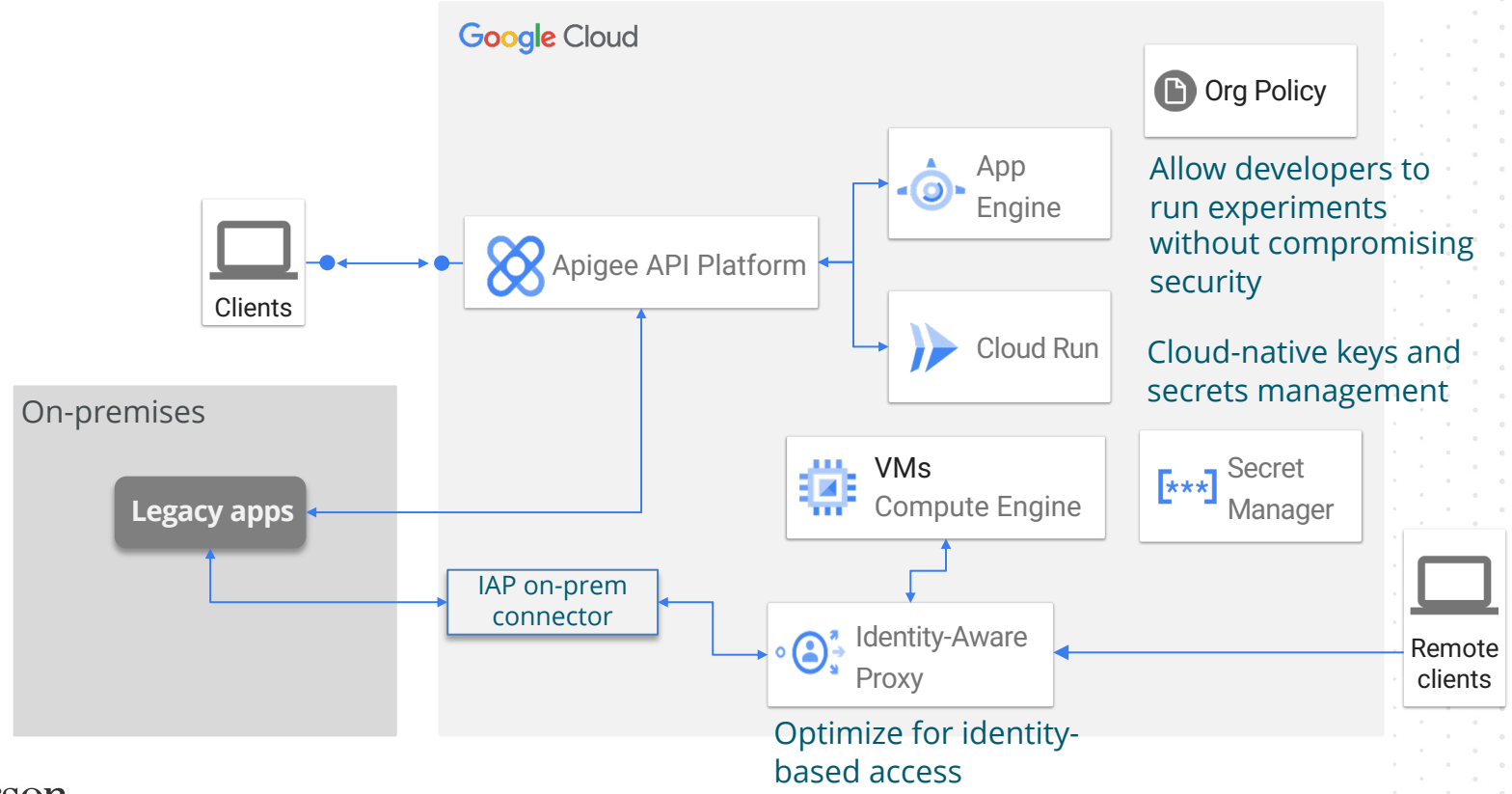


- Manufacturing, IoT, global
- 2 million vehicles (and growing), 200-500MB of data per day
- Real-time and batch telemetry data
- Multiple interconnects already present.
- Remote developers
- Requirements: predictions (AI/ML), elasticity, low ops. Increase development speed and reliability (CI/CD), platform for API services, self-service portal, cloud-native key and secrets management, enable partners to access data.

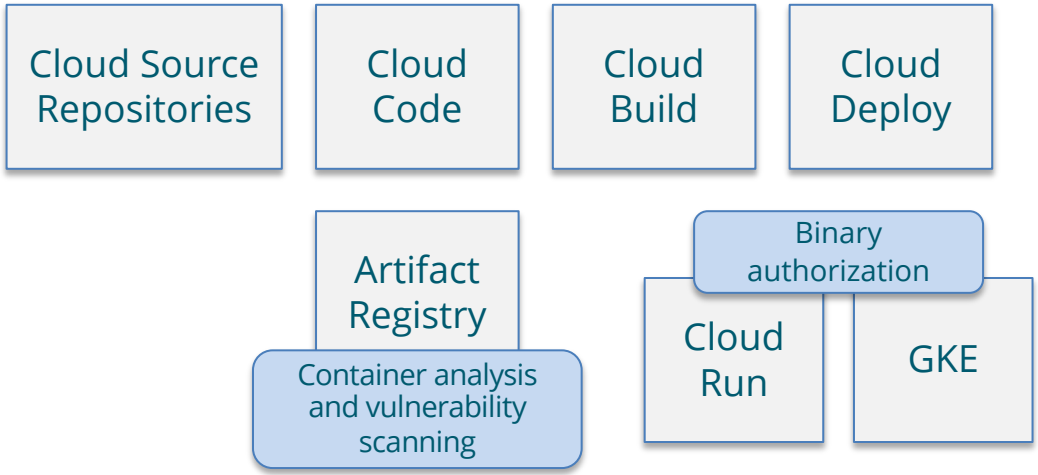
Case Study Solution: API



Case Study Solution: Security



Case Study Solution: CI/CD Building Blocks



Case Study Solution: Predictions Building Blocks

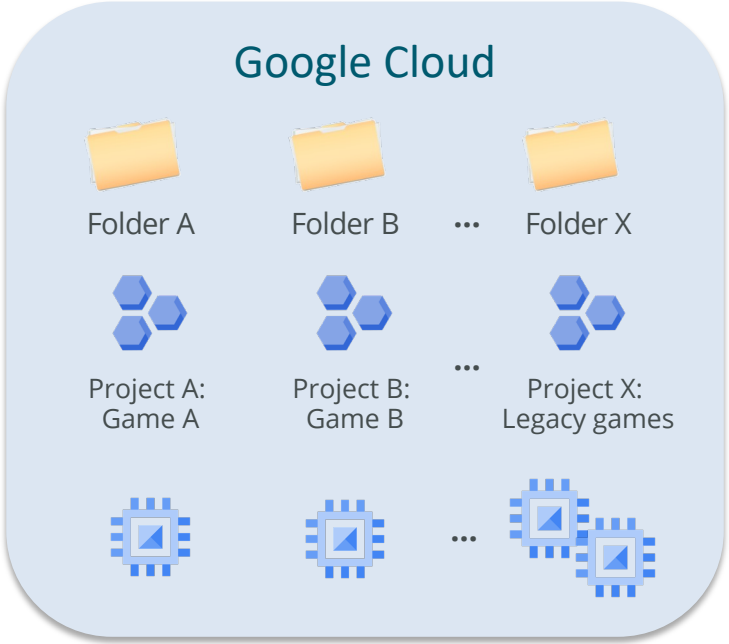
Segment 8: Case studies solutioning – pen-and-paper exercises

Objectives

- Case Study Breakdown: EHR Healthcare
- Case Study Breakdown: Helicopter Racing League
- Case Study Breakdown: TerramEarth
- Case Study Breakdown: Mountkirk Games

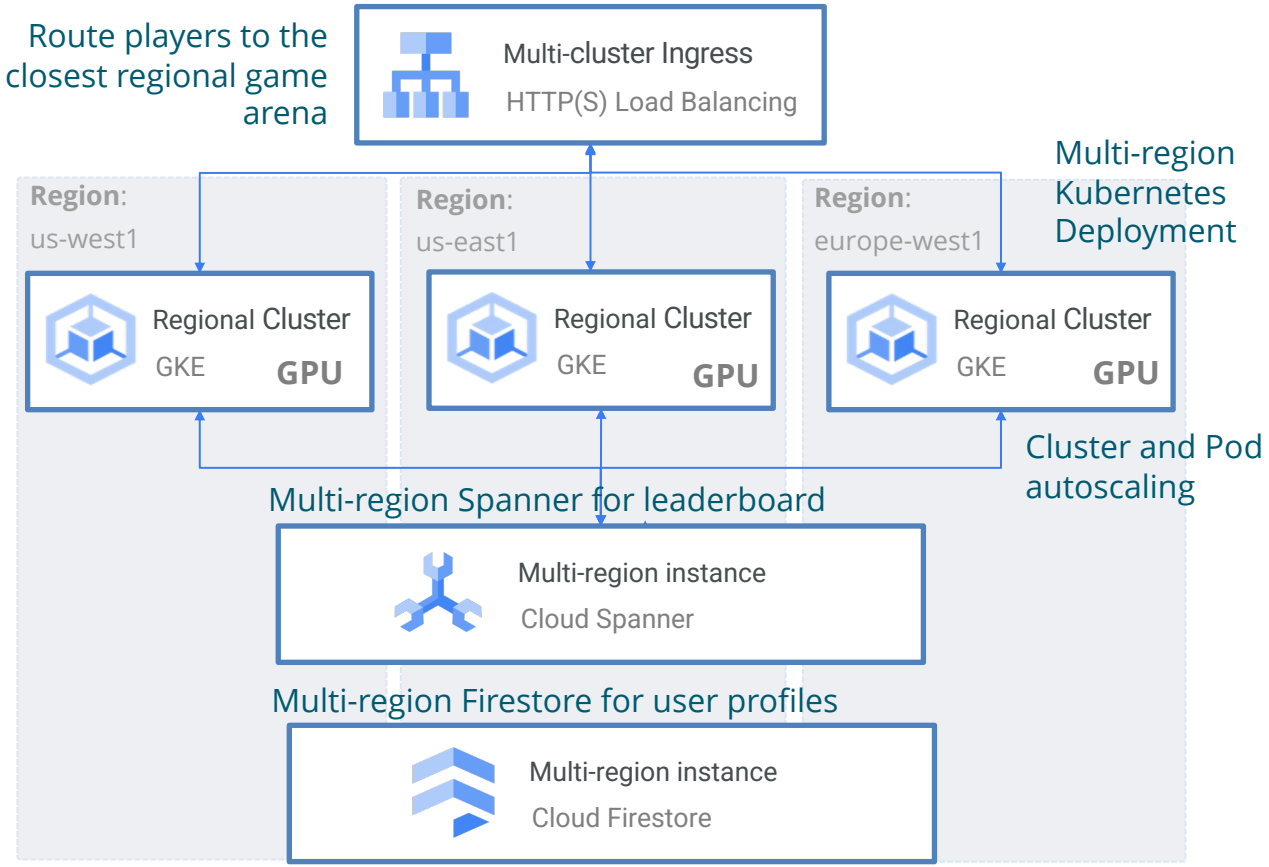
Exam Case Study: Mountkirk Games

Case Study Summary



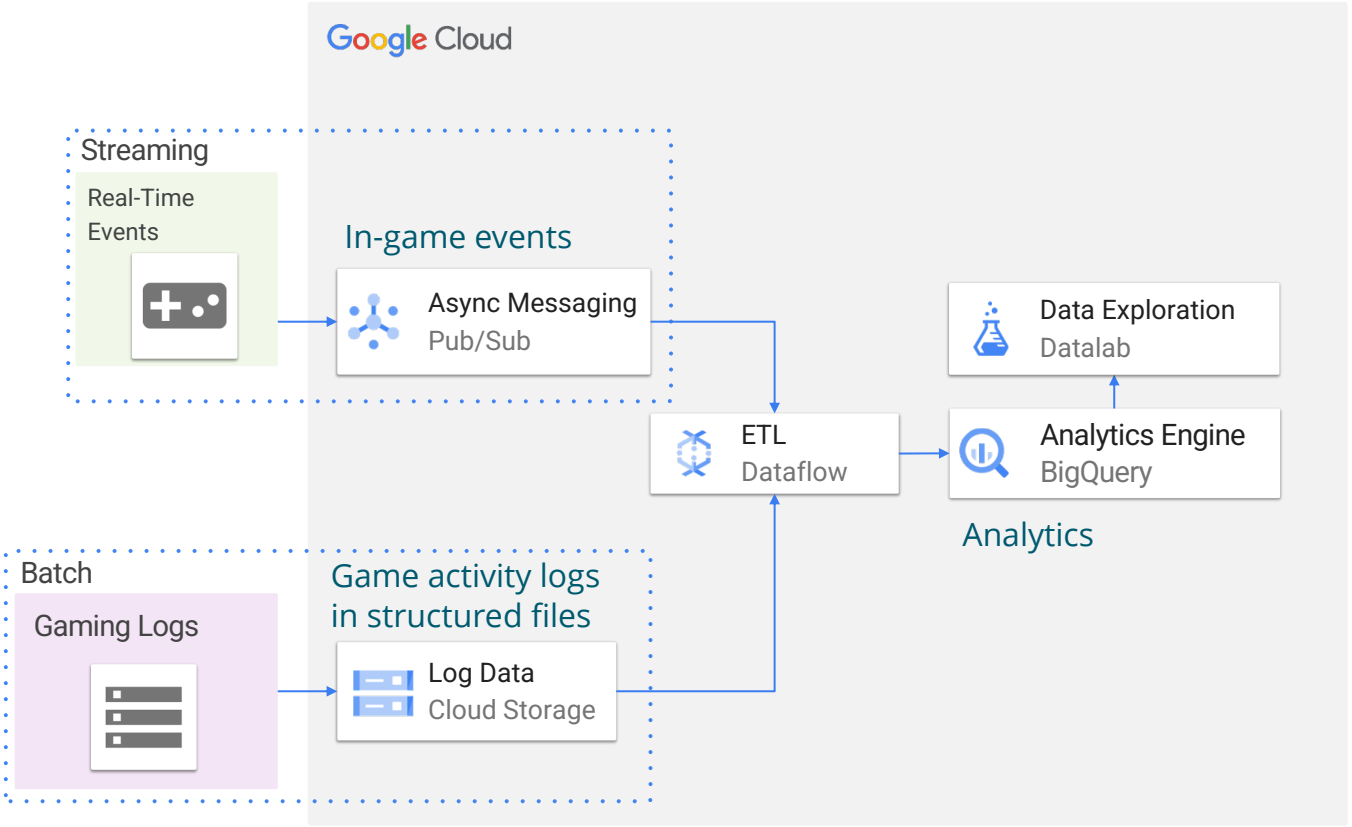
- Online mobile gaming, global, near real-time leaderboard
- Fully on Google Cloud, but still running mostly VMs. Separate dev/test environments
- Plan to use GKE, Global Load Balancer, and multi-region Spanner for leaderboard
- Each game in an isolated project
- Legacy games bundled in single project
- Requirements: Multi-region, low latency (top priority), elastic, rapid development, multiple gaming platforms, managed services and pooled resources, GPU processing, game activity logs in structured files for batch analysis.

Case Study Solution: Game Backend



Pearson

Case Study Solution: Analytics



Case Study Solution: Other Considerations

- Different **GCP projects** for dev/test/prod environments
- **Load testing** with production-like traffic
- **Canary** releases
- **Migration planning**: app inventory, impact analysis, staged migration, testing/proof-of-concepts, etc.
- **Anthos migrate to containers**
- IAM strategy that takes into account **policy inheritance**

Segment 9: Wrap-up

Objectives

- Q&A
- Course wrap-up
- Exam tips
- Next steps

Exam Tips and Next Steps

- Sign up for free trial and get hands-on with GCP
- Use spatial repetition (flashcards) apps to remember concepts (e.g. **Anki**)
- Address major weaknesses first
- Schedule the exam for a time when you're most alert
- Don't skip over the fundamentals

Thank You!

linkedin.com/in/vicdan

Pearson