# The Architect’s Guide to Serverless on Google Cloud

## Mastering Modern Design Patterns for the Professional Cloud Architect Exam

# The Mission: Focus on Code, Not Infrastructure

**Before**

**Abstraction**

**After**

**Core Statement:** Serverless computing abstracts away infrastructure management, enabling teams to build faster and scale seamlessly. Your role as an architect is to select the right tool for the job.

### Abstraction

No servers to provision or manage.

### Scalability

Automatically scales with demand, including to zero.

### Pay-per-use

Billed only for the resources consumed during execution.

### Introduction to the Google Cloud Spectrum:

- **Cloud Run:** For containers.
- **App Engine:** For applications.
- **Cloud Functions:** For events.

# The Architect's Palette: Your Core Compute Choices

## Cloud Run
The Container-First Choice

- **What it is:** A fully managed platform for running stateless containers.
- **Best for:** Modern web services, APIs, and jobs where you need the flexibility of custom containers.

## App Engine
The All-in-One Platform

- **What it is:** A fully managed Platform-as-a-Service (PaaS) for web and mobile applications.
- **Best for:** Rapid development of applications where the platform manages everything from scaling to versioning.

## Cloud Functions
The Event-Driven Specialist

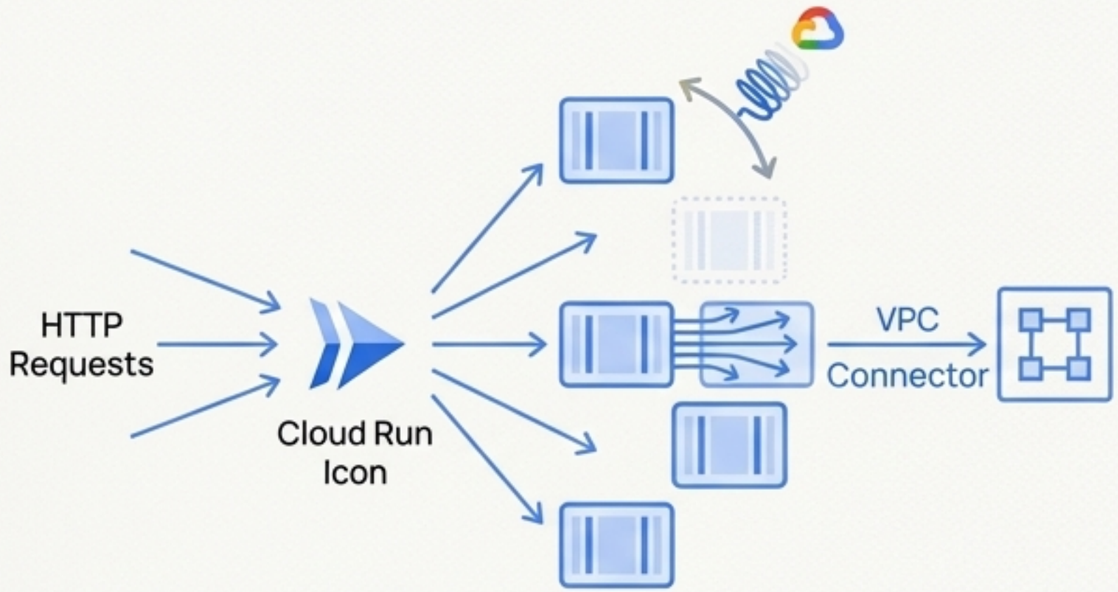
- **What it is:** A serverless Function-as-a-Service (FaaS) for running code in response to events.
- **Best for:** Glueing services together, data processing pipelines, and event-driven automation.

# The Modern Workhorse: Managed Containers with Cloud Run

**Combines the flexibility of containers with the simplicity of serverless.**

## Key Architectural Concepts

- **Stateless by Design:** Each request is handled in isolation. No local state persists between requests.
- **Autoscaling:** Scales from zero to thousands of instances based on incoming requests. Be mindful of cold starts.
- **Concurrency:** A single container instance can handle multiple requests simultaneously (default: 80). Tuning this is a key performance/cost lever.
- **Networking:** Integrates with HTTPS Load Balancers via Serverless NEGs for custom domains and global routing. Use a VPC Connector for private egress.




| | |
|---|---|
| **Services** | Respond to HTTP/gRPC requests. The typical choice for APIs and websites. |
| **Jobs** | Run-to-completion tasks. Ideal for batch processing, data migration, or ETL workloads. |

# The Managed Platform: All-in-One PaaS with App Engine

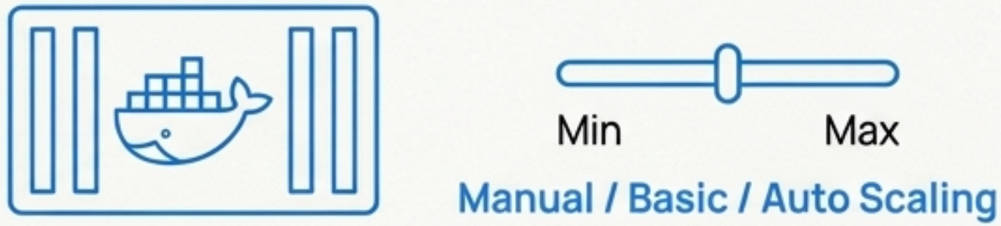
A fully managed platform for building and running applications without managing infrastructure.

## Critical Choice

### App Engine Standard


- **Runtime:** Sandboxed environment with limited, specific runtimes (e.g., Python, Java, Go).
- **Scaling:** Scales to zero, very fast scaling.
- **Use Case:** Ideal for traditional web apps and APIs with predictable runtime needs.

### App Engine Flexible




- **Runtime:** Runs your code in a Docker container, allowing custom runtimes and binaries.
- **Scaling:** Does not scale to zero. More control with manual/basic/auto scaling options.
- **Use Case:** Applications requiring custom libraries, background processes, or more CPU/memory.

## Built-in Capabilities

Memcache

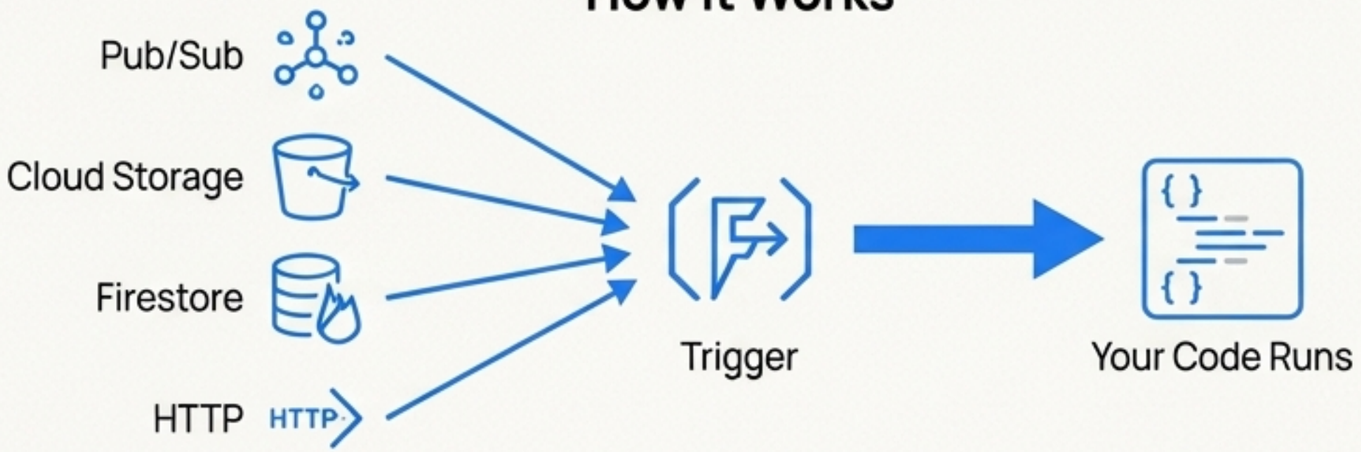
Task Queues

Cron Jobs

# The Event Specialist: Serverless Logic with Cloud Functions

A serverless Function-as-a-Service (FaaS) for lightweight, event-driven workloads.

## How it Works



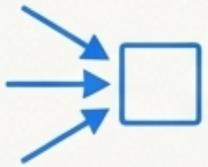

## Key Evolution: Gen 1 vs. Gen 2

**Gen 1**
Traditional FaaS experience.

**Gen 2**
Built on Cloud Run, offering longer request processing times, larger instances, and concurrency. ****For the exam, assume Gen 2 unless specified.****

## Critical Exam Concepts

### Error Handling
Design for retries and configure dead-letter topics for failed invocations.



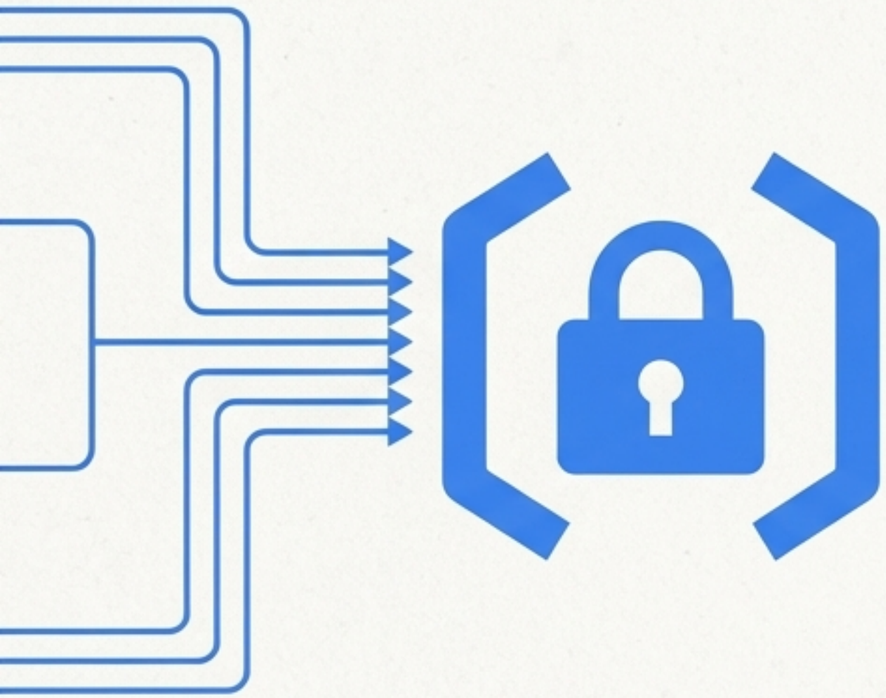
### Idempotency
Ensure your function can safely be called multiple times with the same input.

### Security
Use IAM invoker roles to control who can trigger the function.

# The Architect's Decision Matrix: Choosing Your Compute

| Criterion | Cloud Run | App Engine Standard | App Engine Flexible | Cloud Functions (Gen 2) |
|---|---|---|---|---|
| **Unit of Deployment** | Container Image | Source Code | Source Code (Containerized) | Source Code |
| **Primary Use Case** | APIs, Web Services, Jobs | Standard Web Apps | Apps with custom dependencies | Event-driven "glue code" |
| **Runtime** | Any Language/Binary | Specific Runtimes | Any Language/Binary | Specific Runtimes |
| **Scale to Zero** | Yes | Yes | No | Yes |
| **Concurrency** | Multiple requests/instance | 1 request/instance | Multiple requests/instance | Multiple requests/instance |
| **Max Timeout** | 60 mins (HTTP), 24h (Jobs) | 10 mins (auto), 24h (basic) | 60 mins | 60 mins |
| **Networking** | VPC Connector | Built-in Services | VPC Connector | VPC Connector |

# The Secure Foundation: Your Artifact Supply Chain



Artifact Registry is the evolution of Container Registry. It is a single, managed registry for all your software artifacts.

## Supported Formats

- Docker container images
- Maven (Java) packages
- npm (Node.js) packages
- PyPI (Python) packages
- APT and YUM (Linux) packages

## Repository Structure

Google Cloud Project

- us-central1
- europe-west1
- us-central1

A single Google Cloud project can host multiple repositories, each with a specific format and region, allowing for granular organization.

# Fortifying the Supply Chain: Security and Governance

A secure application starts with secure artifacts.

### 1. Access Control (IAM)

Use fine-grained IAM roles to control who can push (write) and pull (read) from each repository. Example: CI/CD service accounts can write, production runtimes can only read.

### 2. Vulnerability Scanning

Automatically scan container images for known vulnerabilities (CVEs) upon push. This provides critical security insights before deployment.

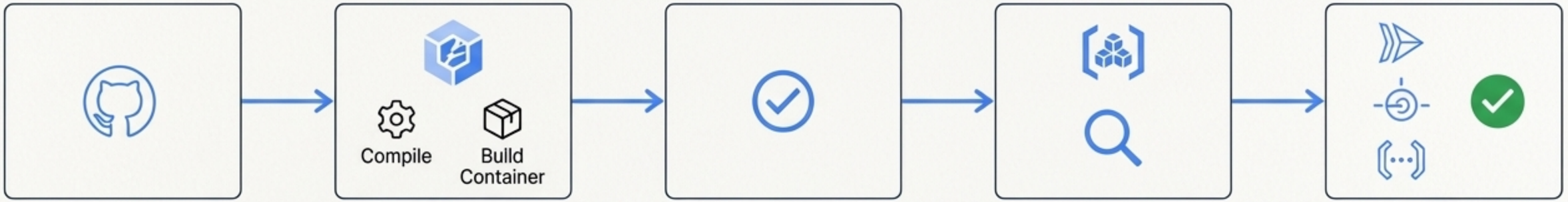
### 3. Provenance & Integrity (Binary Authorization)

Use image signing with attestations to ensure that only trusted, verified images are allowed to be deployed in your environments (e.g., GKE, Cloud Run).

**[Exam Trigger]** A corporate policy mandates that no container with a 'CRITICAL' vulnerability can be deployed to production.

Enable Artifact Registry vulnerability scanning and integrate a check in your Cloud Build pipeline that fails the build if critical CVEs are found.

# The Automated Factory: Your CI/CD Assembly Line




**Core Tool:** Cloud Build, a fully managed, serverless CI/CD platform.

**A Typical Serverless CI/CD Pipeline:**

1. **Trigger:** Code is pushed to a Cloud Source Repository or GitHub/Bitbucket.
2. **Build:** Cloud Build executes steps defined in `cloudbuild.yaml` to compile code and build a container image.
3. **Test:** Run unit and integration tests.
4. **Scan & Push:** Push the image to Artifact Registry, triggering a vulnerability scan.
5. **Deploy:** If tests and scans pass, deploy the new version to Cloud Run, App Engine, or Cloud Functions.

**Key Features:** Use substitutions (`_VARIABLE`) for environment-specific configurations (dev vs. prod). Integrate with Cloud Deploy for advanced progressive delivery, approvals, and rollbacks.

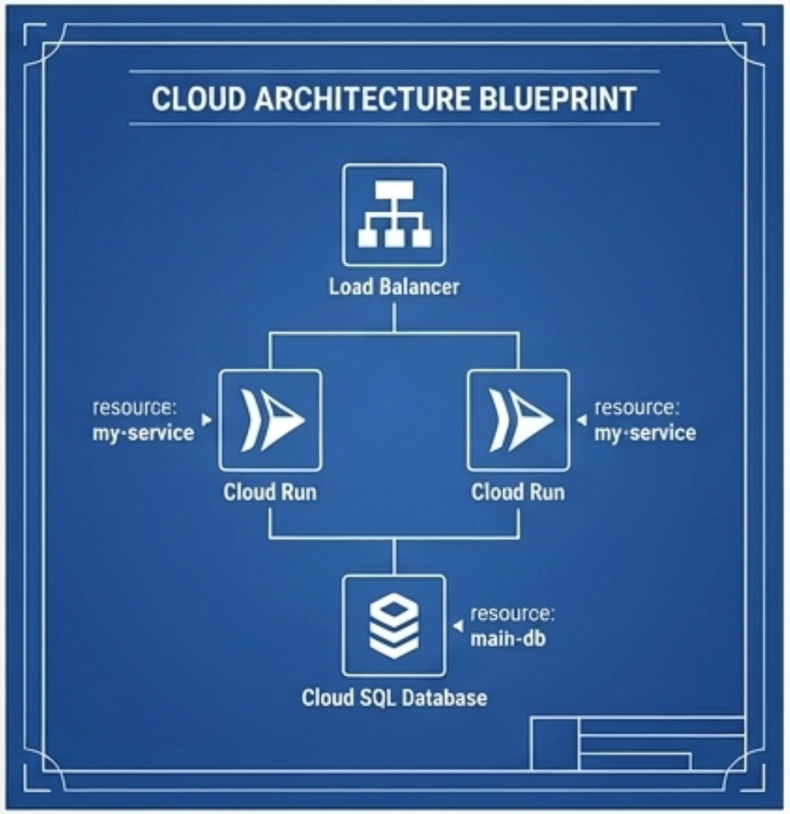
# The Blueprint: Defining Your World with Infrastructure as Code

**Core Concept**
Manage your cloud resources declaratively, just like application code. This ensures consistency, repeatability, and version control for your entire environment.

**Introducing Cloud Deployment Manager (CDM)**
Google Cloud's native IaC service.

**Core Components**

- **Templates**: Define resources in YAML, with optional Jinja2 or Python for logic.
- **Configurations**: A YAML file that lists the templates and resources to be deployed.
- **Preview Mode**: A critical safety feature. Run `gcloud deployment-manager deployments update --preview` to see what will change before applying it.

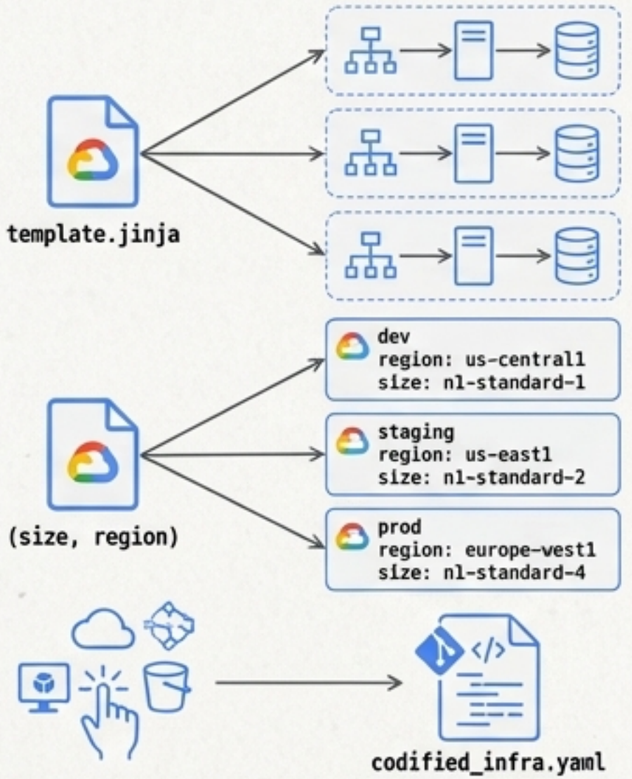



**Why use it?** Move from manual or scripted setups to a repeatable, version-controlled process.

# CDM in Practice: Migration and Multi-Environment Patterns

## Primary Use Cases

- **Standardization:** Create reusable templates for common architectures (e.g., a standard web service with a load balancer and database).
- **Multi-Environment Deployments:** Use parameterized templates to deploy consistent infrastructure for `dev`, `staging`, and `prod` environments, changing only the variables (e.g., machine size, region).
- **Migration:** Codify existing, manually-created infrastructure into CDM templates to bring it under version control.




**[Exam Trigger]**
You are asked to recommend a GCP-native tool to automate the creation of repeatable, version-controlled environments for a new project.
**Solution:** Cloud Deployment Manager.

**Consider Alternatives**: Briefly mention that while CDM is the GCP-native choice, Terraform is a common answer for multi-cloud scenarios or if advanced state management and drift detection are required.

# Mastering the Gauntlet: High-Impact Exam Triggers

| If you need... | Choose... |
| --- | --- |
| to run a custom container with any binary. | **Cloud Run** or **App Engine Flexible**. |
| to react to a file upload in Cloud Storage. | **Cloud Functions**. |
| a fully managed platform with built-in task queues and cron jobs. | **App Engine Standard**. |
| to run a script to completion on a schedule (e.g., nightly report). | **Cloud Run Jobs**. |
| to gradually roll out a new feature to 10% of users. | **Traffic Splitting** in Cloud Run or App Engine. |
| a serverless service to access resources in a private VPC. | a **VPC Connector**. |
| to store and scan Python, Java, and Docker artifacts in one place. | **Artifact Registry**. |

# Architectural Traps to Avoid: Common PCA Exam Pitfalls

Storing secrets or API keys directly in source code or environment variables.

Integrate with **Secret Manager** and grant the service's identity access to the secret.

Granting broad project-level permissions to Artifact Registry.

Apply fine-grained **IAM policies per-repository** (e.g., Reader, Writer).

Assuming a serverless service can reach private resources by default.

Always configure a **Serverless VPC Access Connector** for private network egress.

Deploying applications that require custom system libraries to App Engine Standard.

Use **App Engine Flexible** or **Cloud Run**, which support custom container environments.

Ignoring latency from cold starts for a user-facing, latency-sensitive application.

Configure a **minimum number of instances** (`min-instances`) in Cloud Run or App Engine.

# The Modern Architect's Mindset: Agility and Security

**Success on the PCA exam and in modern architecture is not about memorizing every service feature. It is about understanding the trade-offs and building systems that are both agile and secure.**

### Choose the Right Abstraction

Match the service (Container, PaaS, FaaS) to the workload.

### Automate Everything

Use CI/CD and IaC to create repeatable, secure deployment processes.

### Secure the Supply Chain

Treat your artifacts as first-class citizens with scanning, access control, and provenance.

### Design for Failure

Understand cold starts, configure error handling, and plan for scalability.