Application Development with Cloud Run

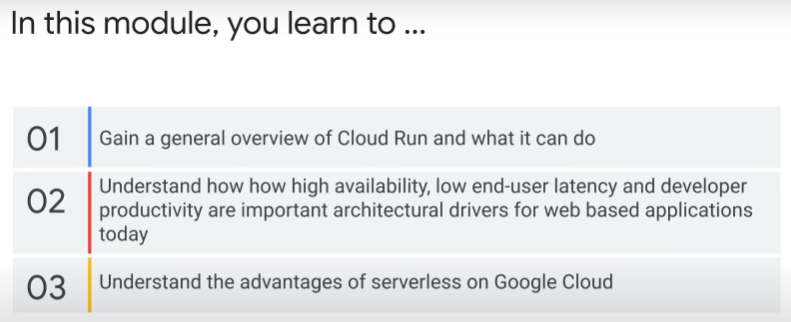
### Course Introduction

1. Introduction to modern cloud native application development
   * fundamentals
   * practices
   * tools
   * capabilities
2. Fully managed compute environment, for deploying and scaling severless containers

* no need to provision machines
* no need to configure clusters or auto-scaling
* write codes in any language (java, js, py, go, php, rb, sh, cs)

1. Learn to
   * design
   * implement
     + secure service-to-service communications
     + monolithic to microservices migration for various languages
   * deploy
     + for high availability, low latency
   * secure
     + based on service identities and granting only permissions needed
   * manage
     + data connection and persistence
   * scale ... applications

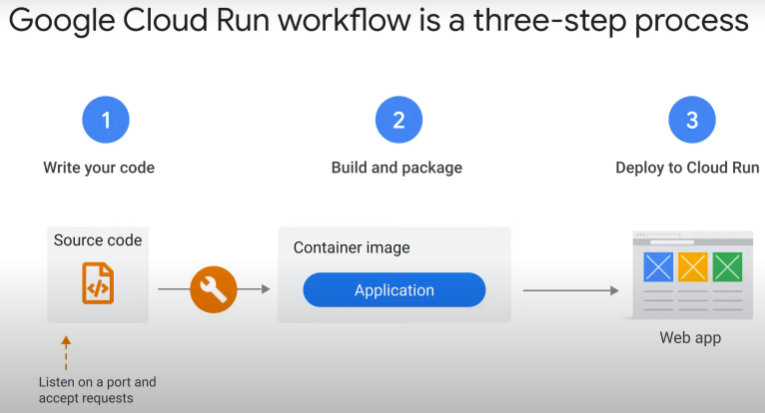
### Introduction to Cloud Run



* Intro to containers
* Intro to container workflow
* Uses cases
* Cloud Run characteristics: use disposable containers, request-driven, auto-scale quickly
  + How to build reliable, scalable, cost-effective applications
  + What this means? Why important?

### What is Cloud Run?

A 3-step developer workflow



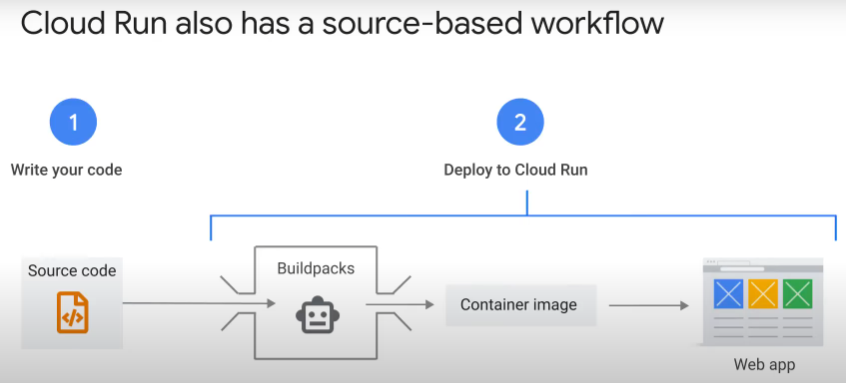
Steps:

1. Write application in preferred programming language
   * application should start a web server that listens on a port for incoming requests
2. Build and package application into container image
3. Deploy container image to Cloud Run
   * gets unique HTTPS URL on successful deployment

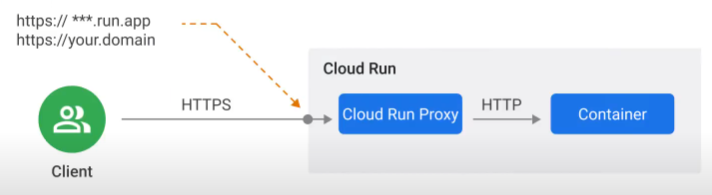
* Cloud Run starts containers on demand to handle requests.
  + Ensures all incoming requests are handled.
    - by dynamically adding and removing containers
  + serverless, i.e. no need to configure and maintain infrastructure

2 types of workflow

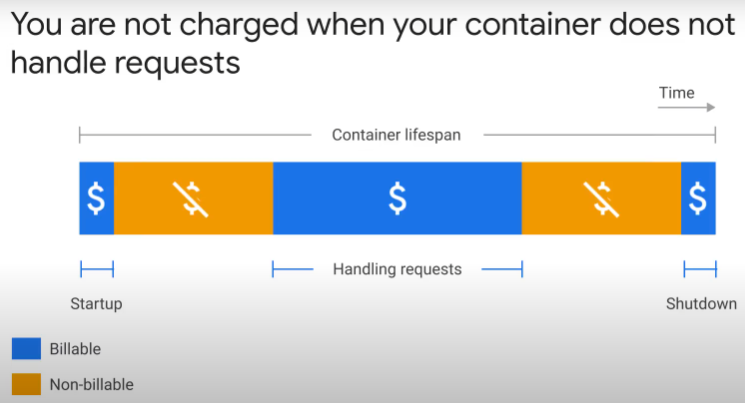
* container-based
* source-based
* Benefits of container-based workflow
  + full control of all files, dependencies, libraries that go into container image
  + customisable
  + portable
  + transparency
  + flexibility

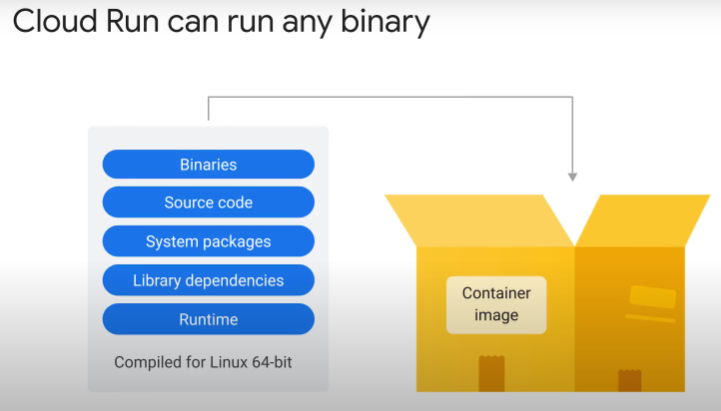


* What is source-based workflow
  + https://cloud.google.com/run/docs/deploying-source-code
  + deploy source code for new services and new revisions
  + *gcloud run deploy* with *–-source* flag
  + *gcloud run deploy* defaults to source deployment if no *–-image* or *–-source* flags
  + command uses buildpacks and Cloud Build to automatically build container images
  + stores images in Artifact Registry in region Cloud Run deployed to.
  + If no Dockerfile in source code repo, buildpacks detect language and download dependencies of code, use secure base image managed by Google
* Benefits of source-based workflow
  + No need to worry about containerization
  + secure – base image managed by Google has needed security fixes
  + consistent and well-configured build if developer does not customise with Dockerfile
  + (-) Will only work in regions that support Artifact Registry
  + continuous deployment from GitHub, Bitbucket, Cloud Source Repository
  + [https://cloud.google.com/run/docs/continuous-deployment-with-cloud-build#setup-cd](https://cloud.google.com/run/docs/continuous-deployment-with-cloud-build" \l "setup-cd)

Service URL is secured by SSL

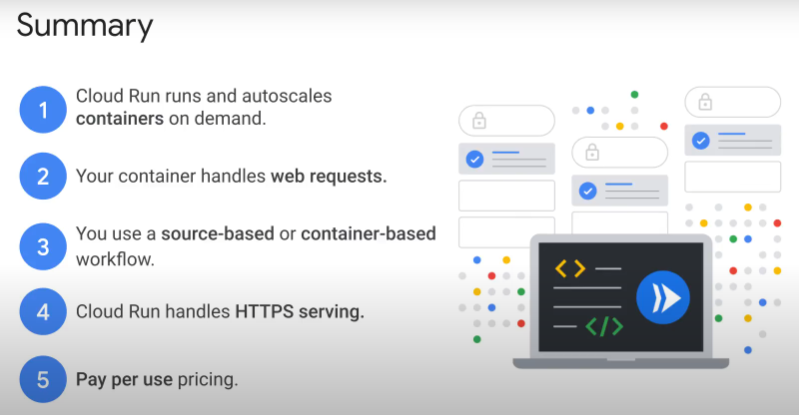
* your application is exposed on unique subdomain of global run.app domain, or your custom domain
* Cloud Run generates valid SSL certificate, configures SSL termination correctly with secure settings
  + handles incoming HTTPS requests, decrypts them and forwards to application

Pricing model of Cloud Run

* Pay for system resources used
* only pay for start, stop and when container is handling HTTPS requests.
* Small fee for every 1 million requests served.
* Container lifespan has granularity of 100ms
* No cost when idle
* cost of container time increase with CPU and memory allocated to container
* max is 4 vCPUs and 8Gb of RAM
* other compute products, e.g. Compute Engine, charge as long as they are running. Even if not used. User paying for idle server capacity.

Cloud Run can run any binary, as long as it is compiled for Linux 64-bit.

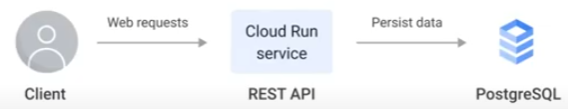
* Means can run apps written in java, python, node.js, PHP, Go, C++ ..
* app must be able to handle web requests



### Typical Use Cases

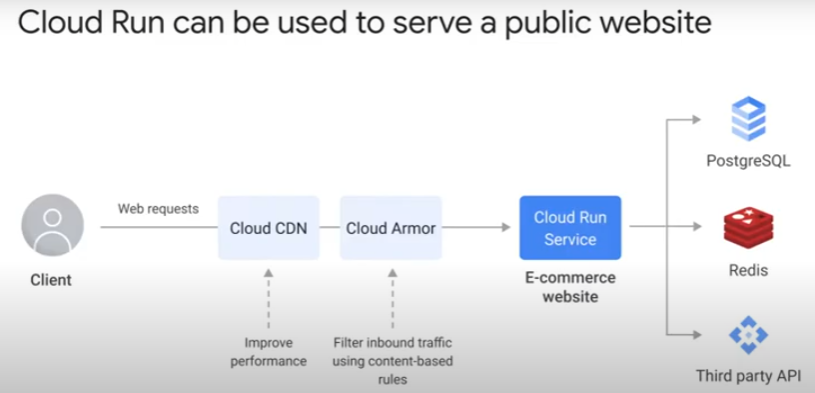
1. Website, Web application

* serve a REST API
* Cloud Run Service connected to a database
* can serve ML predictions without database connection
* no limit on container image size. Memory size limited to 8Gb.



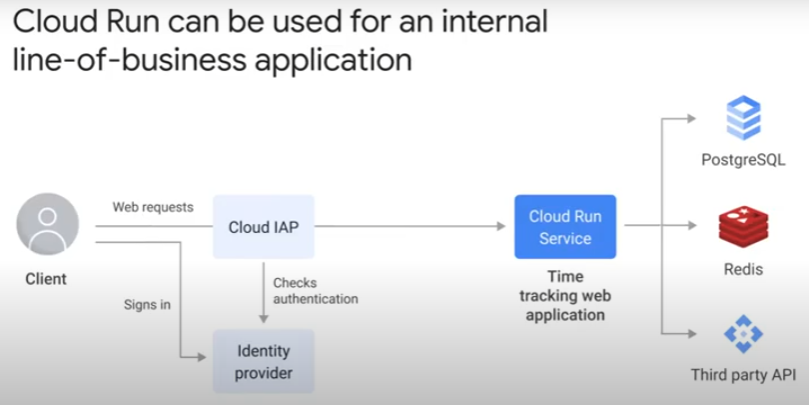
2. More complex website, e.g. e-commerce

* add CDN (content delivery network). Distributed network of servers at edge locations that cache assets needed for website loading. Decrease page load time.
* add Cloud Armor.
  + Filter out malicious traffic using content-based policies
  + Network security to protect against DDoS and web application attacks (Web Application Firewall).
* connect to different types of databases (in-memory Redis store for session info, relational, non-relational). Connect to 3rd party APIs.



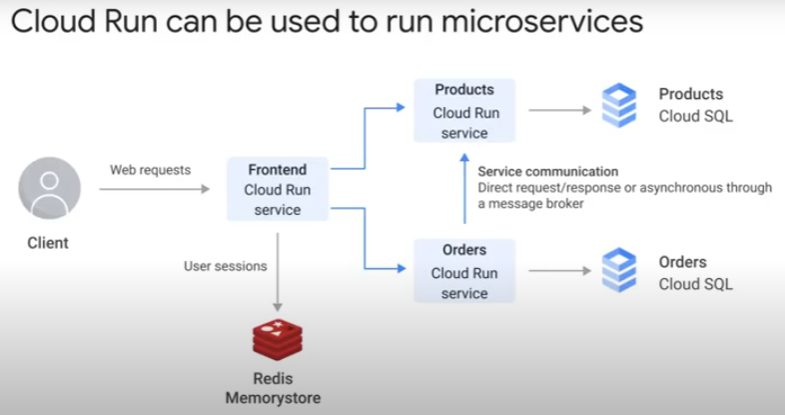
3. Internal business applications

* Cloud Identity Aware Proxy (IAP) to handle authentication using Federated Identity Management(FIM).
  + Use existing directory of users.
    - Users’ credentials provided to and stored with identity provider.
  + Single login, multiple access.
  + e.g. with Google WorkSpace account
* the Cloud Run Service for applications that don’t need high availability



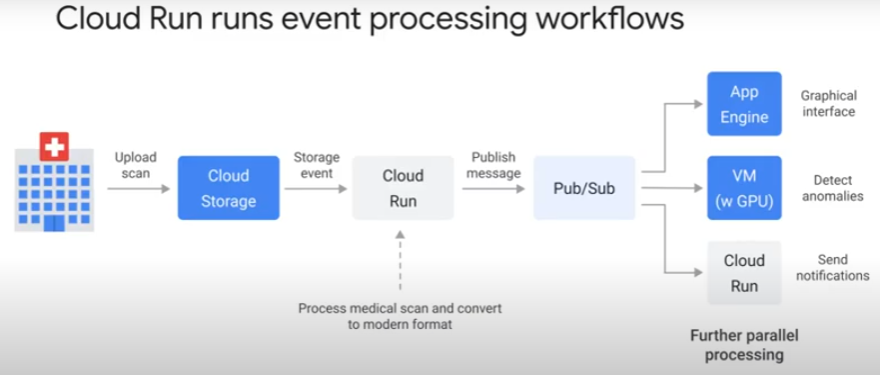
4. Microservice architecture

* break up application into services
* up to 1,000 services in a project
* each service can have its own database
* multiple options for service-service communications: direct RPC calls, REST APIs, gRPCs, asynchronous messages via PubSub.
  + PubSub <-> Cloud Run service
    - Push subscriptions
    - messages are HTTP/S requests
    - guaranteed delivery
    - event notifications



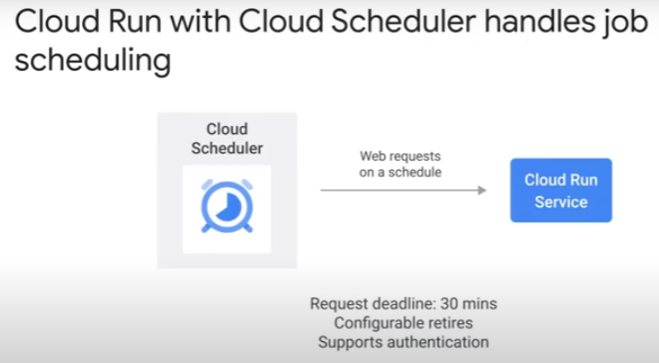
5. Business workflows triggered by events

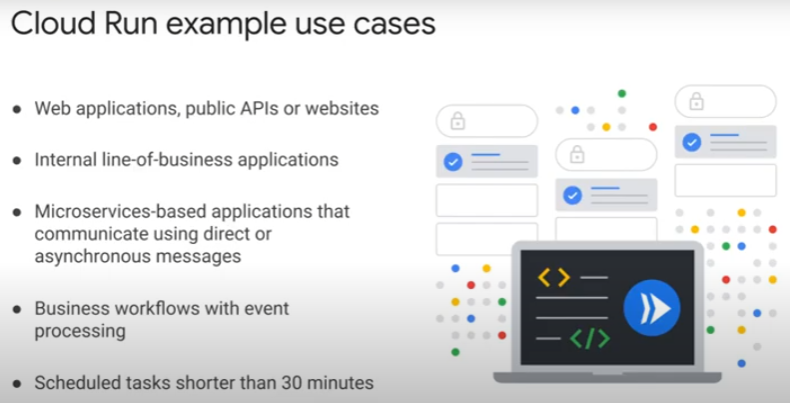
* event notifications via PubSub to Cloud Run services
* can be triggered by events from GCS, Big Query, Cloud Build, PubSub topic, Cloud Log etc
* example: Convert old medical images from outdated binary formats to modern formats. Medical imaging devices have 20+ years life-cycle. Cloud Run on demand.
* Example: ordering inventory in e-commerce system



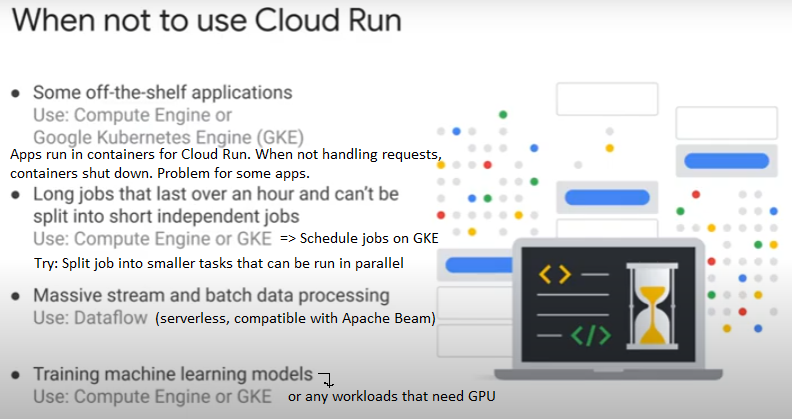
6. Invoke jobs on a schedule

* example: generate invoices, rebuild search index
* container lifetime only guaranteed when it is handling requests
  + if tasks scheduled to start later, container may have auto shut down before task starts/finishes
  + if taks lasts > 30 minutes, request may time-out





When NOT to use Cloud Run?

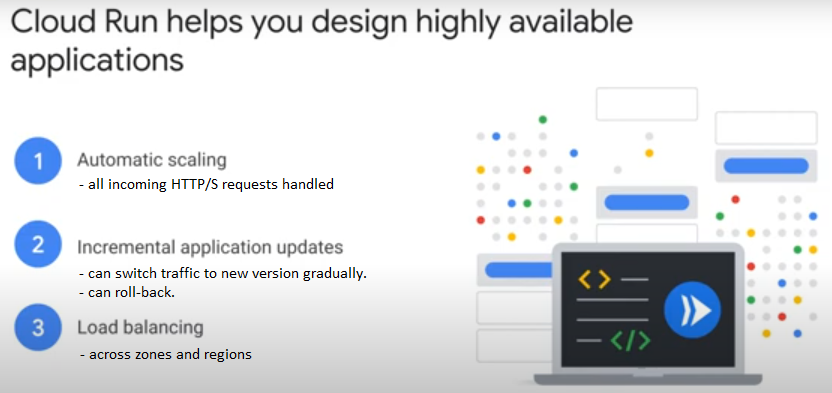


### High Availability

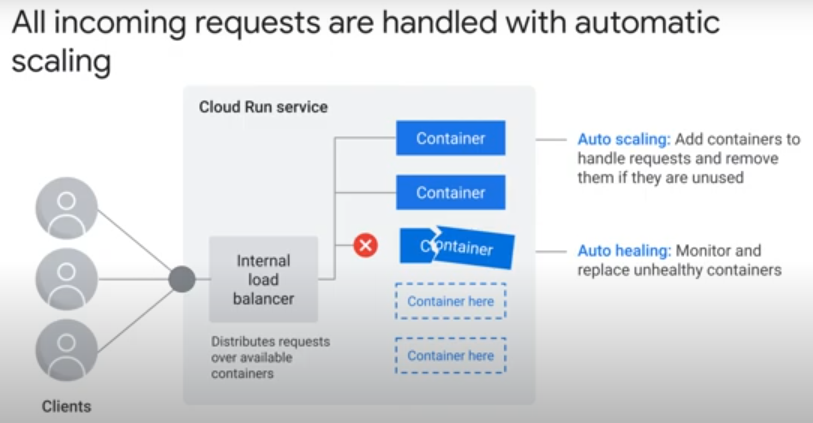
Importance:

* for applications critical to a business. App must be available for business to operate.
  + Loss $$ and reputation if down.
* Planned maintenance downtime was standard.
  + Service unavailable during night for batch processing.

3 features of Cloud Run useful for design of high-availability applications.

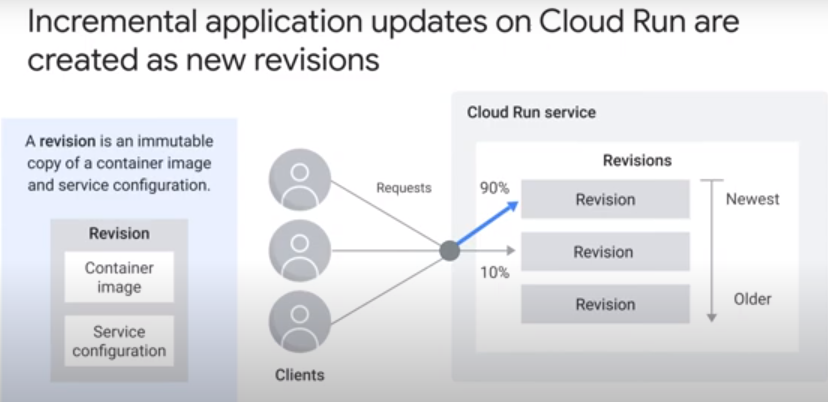


Feature 1: Auto-scaling



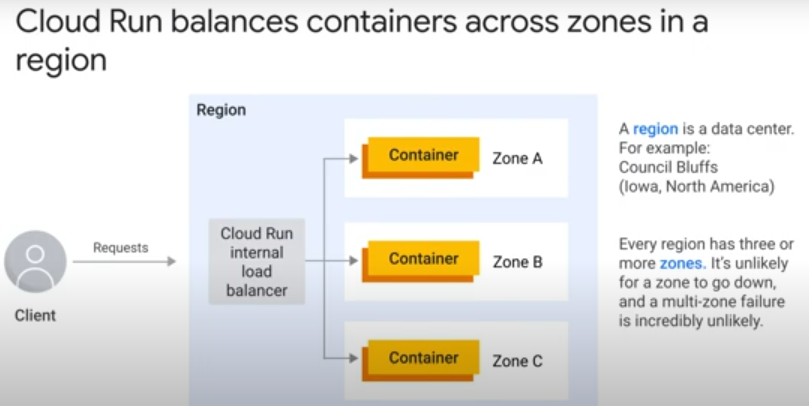
* Cloud Run ↑↓ capacity to handle ALL incoming requests. No drops.
* Auto-scaling:
  + Each service has internal load balancer.
    - distributes traffic over available containers
  + If load > X % for all containers, add containers
  + if load < Y %, stops sending traffic to some containers and shut them down
  + max 1,000 containers in 1 Cloud Run service.
    - Can increase if send request to Google Support
* Auto-healing:
  + if container fails health check, replaced

Feature 2: App remains available with rollout of updates



* Application updates are events that can affect app availability
* Cloud Run supports immutable deployment
* Any change to application or service configuration >new revision > new container image created.
  + Does not modify existing images
* Canary rollout. Incremental.
  + Maintain service availability during roll-out.
  + Shift traffic to new revision gradually to reduce impact if there is a failure.
    - Rollback if there are issues.
* Or, blue/green deployment – 100% switch to image of new revision.

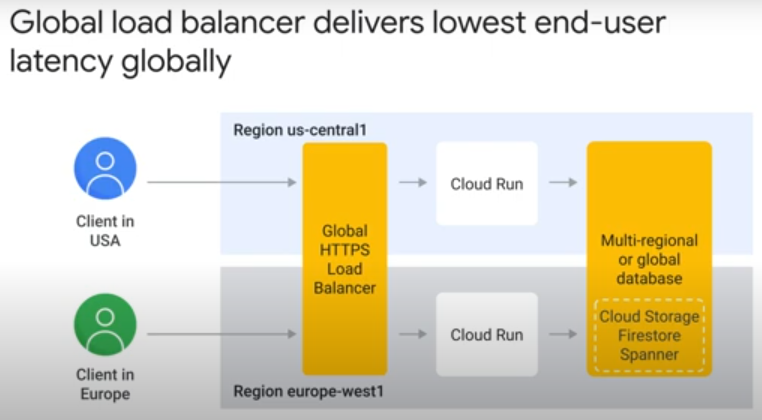
Feature 3: Cloud Run load balances across zones in a region



* Cloud Run is a regional service
* a region is a data center
* Each region is divided into 3 or more zones
* single point of failure (single zone) is unlikely. Entire region to fail, ...nah
* Cloud Run’s internal load balancer distributes containers across multiple zones in a region.
  + Resilient against failure of a single zone.

Same principle for high availability of a global service

* Use a global load-balancer that exposes a single IP address globally
* connect with multiple regional Cloud Run services
* load balancer ensures client traffic always routed to Cloud Run service in nearest region
* decrease latency too
* faster webpage loading for clients. Less likely to abandon HTTPS requests.



### Concerns about serverless

1. Auto-scaling costs

* If deploy Cloud Run service that can scale up to 1,000 containers, may need to pay for 1,000 containers if receive that volume of incoming requests.
* Manage by configuring scaling limits
* Can get commited-use discount of up to 70%, If usage is predictable.

2. Scaling mismatch with downstream systems

* downstream systems may not be able to handle if Cloud Run service really scales up to 1,000 instances within seconds.
* Need design and test throughput

3. Reliant on vendor

* if run into problem with Cloud Run, can only file support ticket

4. Portability \*\*

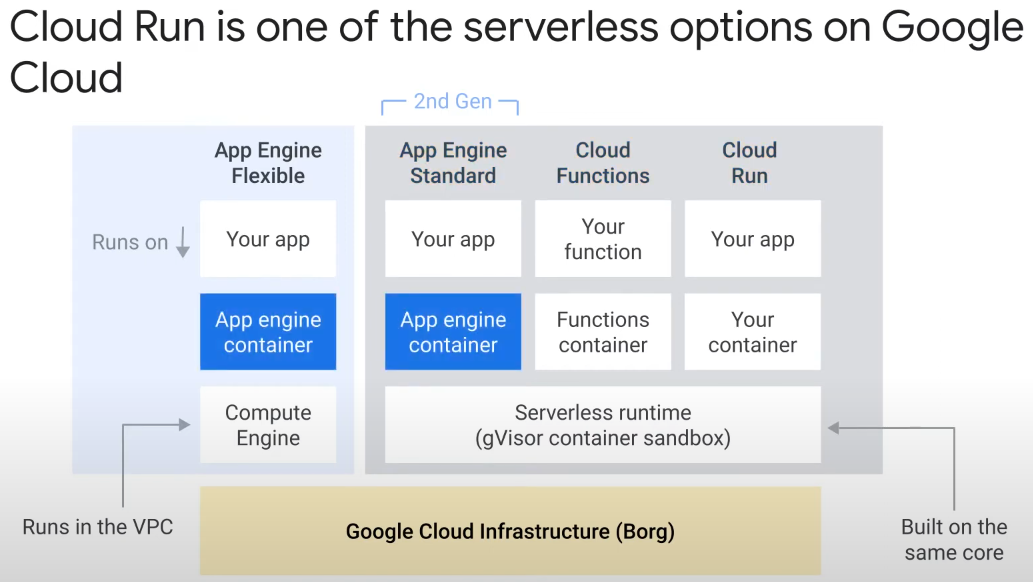
* important to developers
  + do not want vendor tie-in
  + need multi-cloud, on premise – cloud hybrid arrangement
* crucial in some use cases
  + data sovereignty
    - regulatory requirement for application to handle data in specific geographical region.
    - Cloud Run service not available in that region

\*\* Applications on Cloud Run are portable

1. packaged in containers
2. Cloud Run platform is API-compatible with open-source Knative (can cross-talk)
   * but Cloud Run is NOT managed Knative
   * a different implementation of same Application Programming Interface (API)
   * Cloud Run and Knative implement same container runtime contract

### Positioning on Google Cloud

To understand the different serverless products on Google Cloud



* App Engine Flexible is different from the rest
  + runs on top of Compute Engine
  + cannot scale to zero
  + runs within VPC network
* The remaining three services are all implemented on top of gVisor container sandbox.
  + Cloud Run has the most configuration options.

