Google Cloud Run - Why, When and How?

Ryan Deering - X00144631

What is it?

- A *fully managed* cloud service ran by Google Cloud
- Runs Docker container images as a auto-scaling stateless HTTP service.
- Built on the Knative platform, a Kubernetes-based platform to deploy and manage serverless workloads. Open-source and primarily maintained by Google themselves.

Fully managed -- Google essentially handles all the complicated bits of managing such architecture. You can choose not to have fully managed by using the Google Kubernetes Cluster, won't be covered in the scope of the lecture but it is easy to switch.

Stateless - Remind audience of the definition. "a communications protocol that treats each request as an independent transaction that is unrelated to any previous request

Why is it relevant?

Autoscaling - Cloud Run automatically scales your application up and down for you. Management of instances is abstracted away. Supports scaling to zero.

Fully Managed - With it being serverless, Cloud Run handles the frustrating parts of managing IT infrastructure. Resources are allocated on demand.

Variety of Options - Capable of working with any programming language, Cloud Run enables people to use programming languages like Python, C++, Perl (!), and Java. You can attach system libraries to your container image and use them in Cloud Run.

Autoscaling - Mention Google Cloud Run can scale all the way down to zero. If the service is not being used, it can incur no costs! Although if it hasn't been used in a while, there will be cold start latency -- service will take a while to start. Can use Google Cloud Scheduler to periodically request every few mins to keep warm. Worth mentioning free tier as well. 2 million requests from memory. Enables very fine-grained billing.

Fully Managed - You can set the maximum amount of instances in creating a new revision of the service. Scales up to 1000 containers by default, and they can handle 80 requests by default.

Variety of Options - Users have even gotten 22 year old Python running on Cloud Run:
https://dev.to/googlecloud/ministry-of-silly-runtimes-vintage-python-on-cloud-run-3b9d

Enables a wide variety of runtimes to be used. Even very very legacy stuff. Mention the lab, and how the container has ancient binaries from Libreoffice for the actual PDF conversion.

Deployment Process

- Deployment to Google Cloud Run is a simple process
- Primarily need a Docker container image, needs to be hosted on Google Container Registry or Docker Hub. (requires Docker account)
- Container Image requires a HTTP server to run on the image.

Cloud Run for Anthos:

- Visible in the deployment process, same product as Cloud Run but run in different places.

Mention that we'll see in the lab that we need to put our docker image onto the Google Container Registry before we can deploy on Cloud Run.

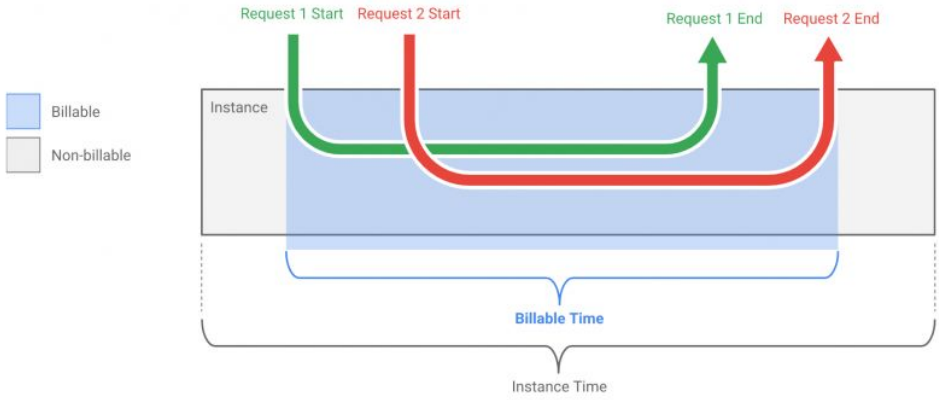
Cost and Idle Resources

- You are **ONLY** billed when a request is being handled on your container instance.
- An application that is not receiving traffic is free of charge. You incur no costs.
- Pricing is based on CPU and memory allocated during billable time, rounded up to the nearest 100 milliseconds.
- If your service handles multiple requests simultaneously, you do not pay for them separately.

Free Tier - 180,000 vCPU seconds per month, 360,000 GiB-seconds per month, 2 million requests per month

You only pay for the CPU, memory and traffic.

Cost and Idle Resources



The only time you are charged when a request is NOT being made is when a container instance is starting or shutting down.
Maximum concurrent requests for a single instance is 80.

When to use it?

If your application is:

- Publicly accessible: APIs, web applications or webhooks
- Private microservices: Internal microservices, data transformation, background jobs

- Applications that do processing when idle or storing in-memory are NOT suited for Google Cloud Run. Applications are throttled down to zero CPU usage when not handling requests.

Private microservices: Mention how there's settings to enable authentication and security.

Cloud Run vs. Cloud Functions

Cloud Functions - Best for event-driven, single purpose workloads. Source-based. Limited in terms of programming languages you can use. Workloads can run up to 9 mins. Lets you deploy small snippets of code.

Cloud Run - Best when all workloads are stateless. Workloads can run up to 15 mins. Container-based - lets you deploy with any stack.

Cloud Functions - Reminder of the labs where the functions were triggered by user actions, specifically mentioning the likes of the Pub/Sub lab and the Hello World HTTP request.

Cloud Run - Mention Cloud Run can do PubSub and Storage events as well. Re: Containers, mention the portability of containers. Allows easy switching to Google's non-managed version of Cloud Run, GKE. Also allows switching to other cloud providers very very easy. Containers mean you're not cooked into Google's cloud architecture. Allows for flexibility but may introduce more tedium due to the greater complexity. Bring up ancient Python again.

Shared - Both services auto-scale your code and manage the infrastructure your code runs on. They both run on the Google Cloud Platform serverless infrastructure.

Cloud Run vs The Competition

AWS Fargate - Fargate allows for a variety of container workloads, not limited to HTTP servers, background and long running tasks. More configuration needed than Cloud Run - subnets, load balancers, auto-scaling. Does not support scale-to-zero or concurrency control. Pay for CPU/Memory while container is running. Exposes container instances.

AWS Lambda - Lambda only recently added container support. More strict than Cloud Run, only allows for AWS-provided images or you have to code your own Runtime API translation layer to get the same kind of functionality as Cloud Run. Doesn't support multiple requests per instance.

Fargate - Make a big point of how on Cloud Run, you only pay while a request is being handled. On Fargate, a service that is receiving no traffic still incurs costs. Make a point on how Cloud Run is more abstract, you do not have access to container instances and their life cycles, it is abstracted away from you.
Cloud Run more suitable for rabid prototyping in this case.

Lambda - Cloud Run can only run container images with HTTP servers. Cloud Run can handle multiple requests simultaneously, not charged separately.

Cloud Run vs The Competition

Azure Container Instances - ACI is more suited towards long-running containers, so the price model differs compared to Cloud Run. Cloud Run supports auto-scaling and scaling to zero, but ACI does not. ACI enables running more than just HTTP servers.

No scaling at all, could be potentially more expensive as it's not switching off when not used.

Sources:

- https://github.com/ahmetb/cloud-run-faq
- https://technologyconversations.com/2020/08/04/google-cloud-run-gcr-vs-azure-container-instances-aci-vs-aws-ecs-with-fargate/
- https://cloud.google.com/run#section-13
- https://medium.com/google-cloud/cloud-run-and-cloud-function-what-i-use-and-why-12bb5d3798e1
- https://cloud.google.com/run/pricing?utm_campaign=CDR_ahm_aap-severless_cloud-run-faq_&utm_source=external&utm_medium=web