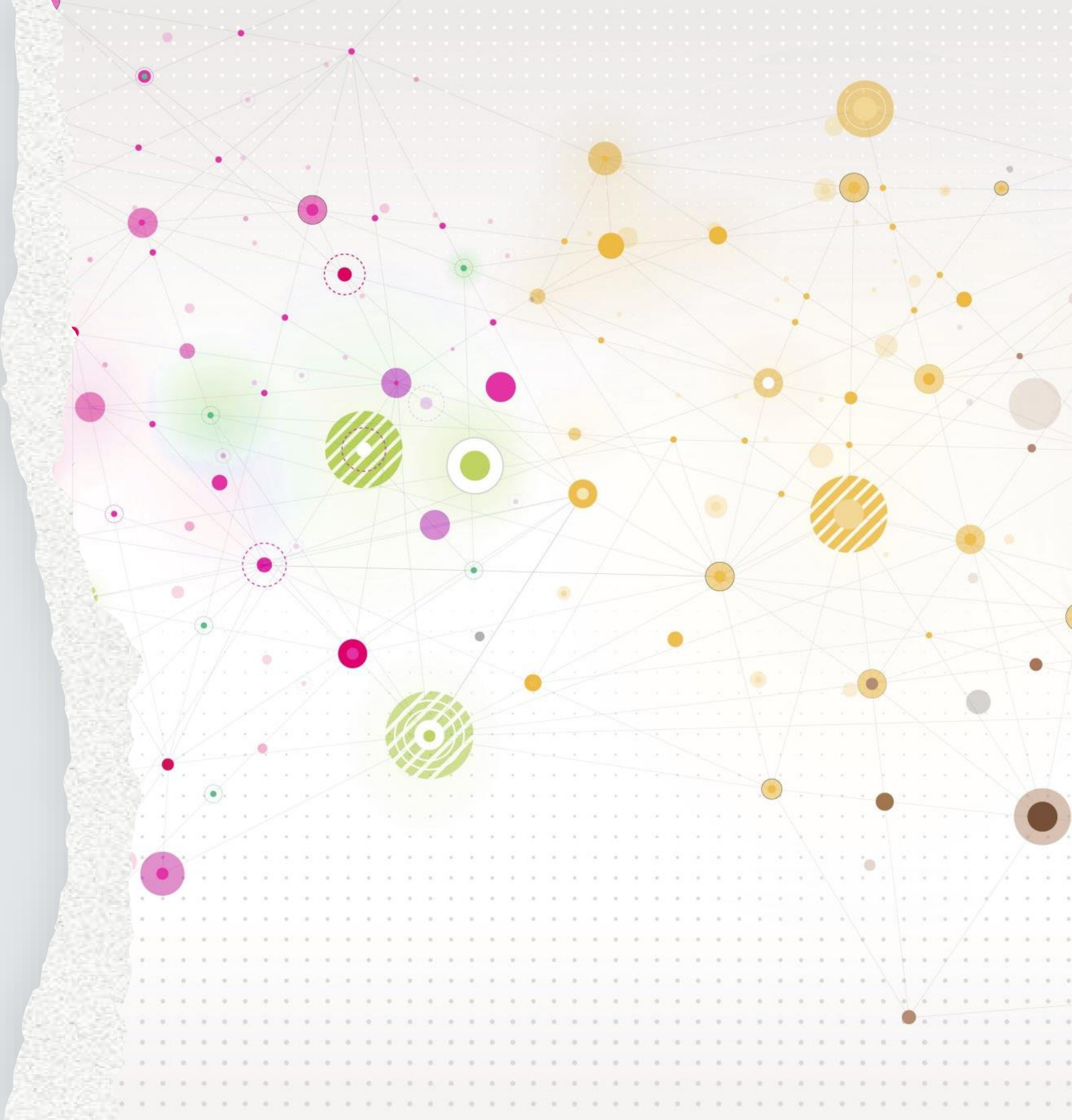
Service Architecture

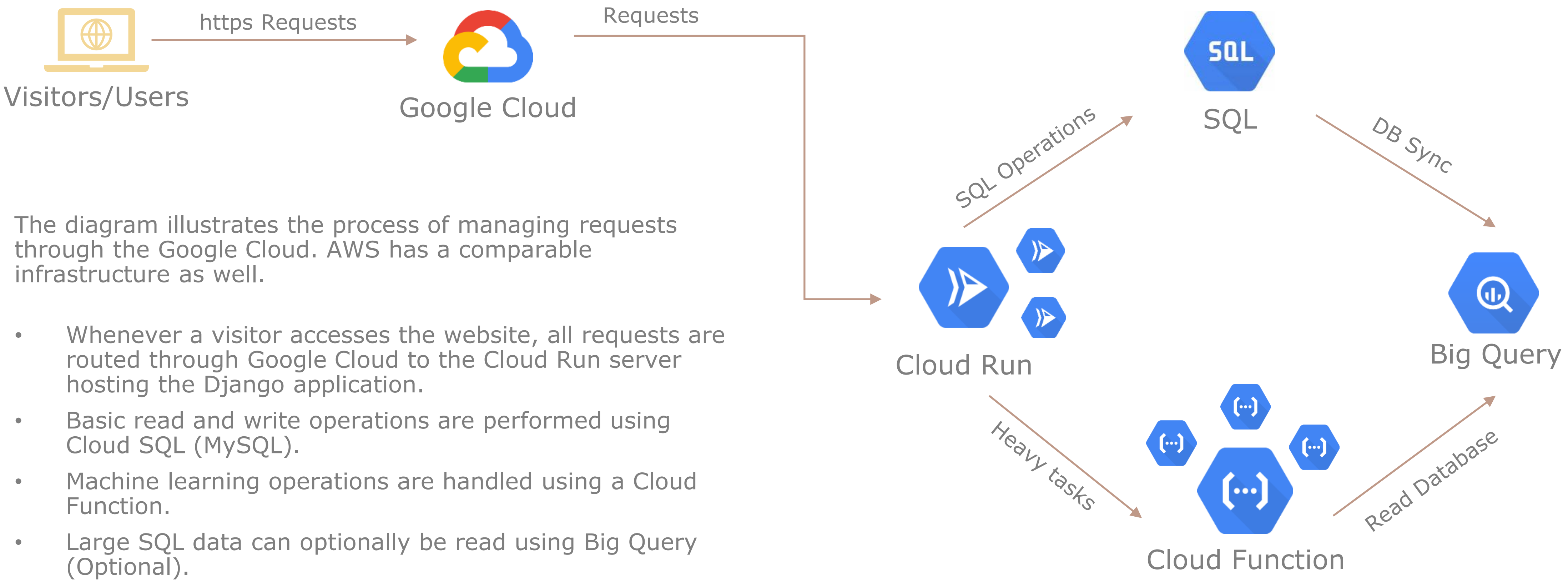
Py-Measurement

It aims to estimate waist size based on input data. While it may not be entirely accurate initially, with consistent input of correct information over time, it becomes increasingly intelligent in predicting the correct waist size.

V1.0



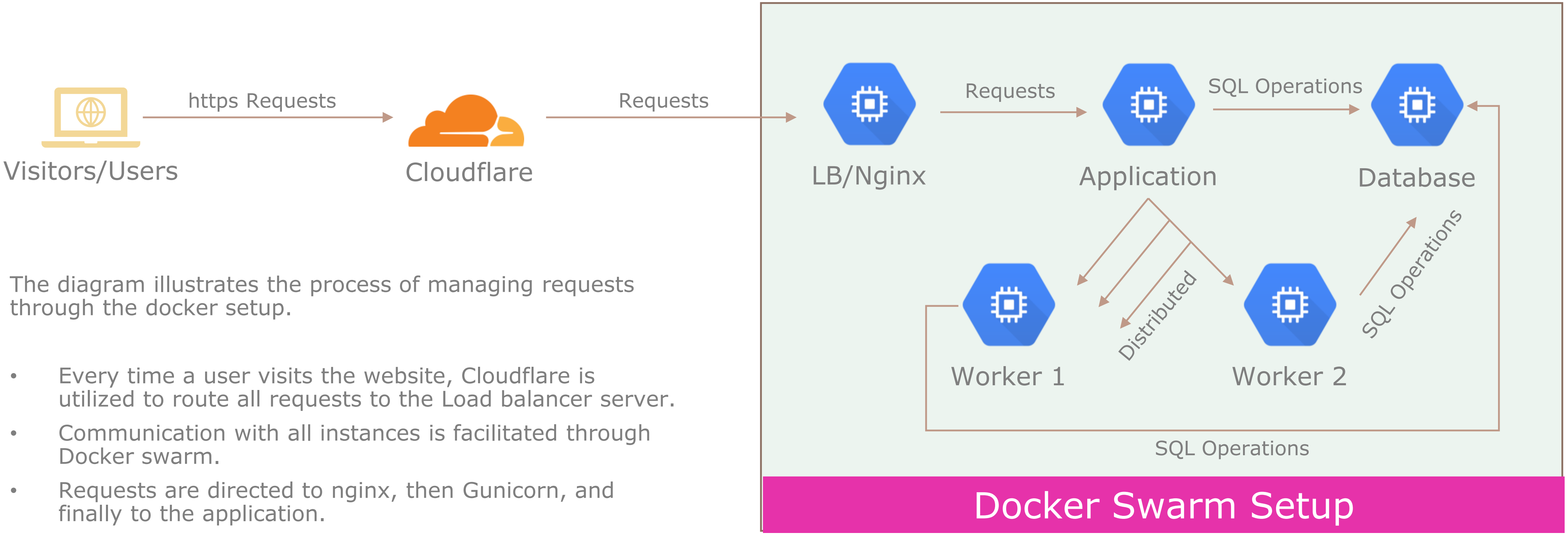
Cloud Deployment Diagram (GCP Serverless) – Primary approach



The diagram illustrates the process of managing requests through the Google Cloud. AWS has a comparable infrastructure as well.

- Whenever a visitor accesses the website, all requests are routed through Google Cloud to the Cloud Run server hosting the Django application.
- Basic read and write operations are performed using Cloud SQL (MySQL).
- Machine learning operations are handled using a Cloud Function.
- Large SQL data can optionally be read using Big Query (Optional).

Cloud Deployment Diagram (Docker on VM) - Secondary approach



The diagram illustrates the process of managing requests through the docker setup.

- Every time a user visits the website, Cloudflare is utilized to route all requests to the Load balancer server.
- Communication with all instances is facilitated through Docker swarm.
- Requests are directed to nginx, then Gunicorn, and finally to the application.
- Workers are established to perform distributed tasks with queue management and can be scaled up or down as needed.
- A dedicated database instance is provided to ensure optimal performance.

Note: This configuration is compatible with most high-quality cloud services available.

Scalability

Primary approach

1. Whenever a request is directed to Cloud Run, it will automatically spawn a new instance if existing instances are overloaded. Cloud Run can support up to 80 concurrent requests per second.
2. Cloud SQL offers advanced computing capabilities, security features, backup functionalities, and migration options. It allows deployment for both MySQL and PostgreSQL databases. *(Cloud SQL is not serverless)*
3. Cloud Functions are utilized for executing distributed tasks, and each request triggers the launch of a new instance due to its concurrency limit of one.
4. By handling read operations for large sets of data, Big Query helps reduce the load on SQL instances and can be synchronized with the main database at any time.

Secondary approach

1. When a request is directed to DNS, Cloudflare communicates with the load balancer instance, which distributes the requests to available Gunicorn instances. Gunicorn loads a new app to serve many requests.
2. To handle distributed tasks, a new worker can be created and torn down as needed. Celery and RabbitMQ are excellent tools for this purpose.
3. To ensure optimal performance, a dedicated database instance is provided and can be further optimized by implementing master (write) and slave (read) instances.
4. With the use of Docker swarm, all instances are connected to the manager and can be automatically scaled based on the docker-compose rules.
5. Caching: Using a caching layer such as Memcached or Redis can help reduce the load on the database and improve response times.

Same for both

1. Database optimization: Optimizing the database can improve performance by reducing query times, optimizing indexes, and implementing read replicas or sharding.

2. Code optimization: Optimizing the code can help reduce server load and improve response times by reducing the amount of processing required for each request. This can include optimizing database queries, reducing the number of queries, and implementing caching.

Security

Primary approach

1. Google handles all security measures.
2. Only essential ports, such as 443 or 8000, are open.
3. The serverless architecture doesn't require any VMs to serve applications or workers.
4. Cloud SQL provides advanced computing capabilities, security features, backup functionality, and migration options.
5. Cloud functions and Big Query are only accessible locally and require a service account for certain actions.
6. This approach is ideal for startups focused on application development, as it enables them to leverage the power of the serverless approach without worrying about security concerns.

Secondary approach

1. Cloudflare is employed to safeguard the web application and prevent the load balancer's IP from being exposed.
2. A new instance should be launched for connecting to other instances via SSH, as opening SSH on main servers can pose a security risk.
3. To minimize security risks, only specific ports on the VM should be opened.
4. Docker includes an internal network that helps to secure the network.
5. This approach is beneficial when there is a need for multiple applications.
6. Overall, this architecture necessitates a centralized workflow managed by the organization to safeguard the VMs.

Note: Deploying services over the cloud always entails security concerns, and it is crucial to choose the appropriate solutions for your application, write quality code, and monitor regularly to mitigate risks.

Pricing

Primary approach

1. Cloud Run/Cloud function is a serverless compute platform offered by Google Cloud Platform (GCP) that enables you to run containerized applications. The cost of running applications on Cloud Run is based on the number of vCPU-seconds, memory-seconds, and network egress (outbound traffic) used by the application.
2. The pricing for Cloud Run/Cloud function is based on a pay-per-use model, where you are charged only for the resources consumed by your application. There are no upfront costs or minimum fees.
3. A SQL instance cost around approx. Rs. 4000 every month (2CPU/4GB/10GB SSD).
4. BigQuery is a cloud-based data warehousing and analytics platform provided by Google Cloud Platform (GCP). The pricing for BigQuery is based on the amount of data processed and the amount of data stored in the platform.

Secondary approach

1. Each instance, which consists of 4 virtual CPUs, 8 GB of memory, and a 128 GB SSD, costs approximately Rs. 7000-8000.
2. Cloudflare Pro incurs a cost of about Rs. 2100.
3. Cloud provider fees for network usage, backup, and restoration are charged separately.

Conclusion

A primary approach that is both easy and cost-effective would be to start with a Minimum Viable Product (MVP). This involves creating a simplified version of your service that includes only the essential features needed to attract early adopters and test the market.

Thank You ☺