Steps to deploying a Web Service to a cloud service
Candidate: Lalitha Viswanathan
Date: June 17 2021
Assumption : Google Cloud

The chemistry team finds that your program works perfectly and shares the news about this tool with the entirety of the Bio-Rad organization, prompting great demand for it from a number of users that are not familiar with working on the command line and are dispersed around the globe.

In order to meet this demand, the company determines that your program should be deployed in the cloud so that it is usable via a web browser.

Your Assignment: In no more than 500 words, please describe the steps necessary to deploy this program in the cloud. Assume that Bio-Rad has a web development team that will produce an interface that will pass the input files and any command line arguments to the executable.

• How would you approach packaging the code and environment to ensure consistent deployment? •

The key steps in Google cloud are as below:

1. Docker can be used to package the code and enviroment to ensure consistent deployment. It is an open platform for developing, shipping, and running applications. Docker enables separation of applications from infrastructure so you can deliver software quickly. This can significantly reduce the delay between writing code and running it in production.
2. Docker provides the ability to package and run an application in a loosely isolated environment (container). The isolation and security allows running of many containers simultaneously on a given host.
3. Docker provides tooling and a platform to manage the lifecycle of containers:
    1. Develop your application and its supporting components using containers.
    2. The container becomes the unit for distributing and testing your application.
4. Docker ensures fast, consistent deliver of applications, streamlines the development lifecycle by allowing developers to work in standardized environments using local containers which provide the user's applications and services and support CI/CD workflows.
5. Docker's container-based platform allows for highly portable workloads. Docker containers can run on a developer's local laptop, on physical or virtual machines in a data center, on cloud providers, or in a mixture of environments.

Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system.The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers. The Docker daemon listens for Docker API requests and manages Docker objects. The client is the primary way end users can interact with Docker. DockerHub is a public registry for storing

Docker images.

A Docker image can be deployed to GKE. by uploading the image to the Container Registry, deploy the app to the cluster, managing the autoscaling for the deployment, expose the app to the internet. GKE allows to run a large number of microservices, have faster container startup time, supports automated orchestration, node repair, node upgrade and auto scaling.

• How would you ensure that sufficient compute resources are available so that users get the quickest possible turnaround time on their analyses? •

Google Cloud uses quotas to restrict how much of a particular shared Google Cloud resource that one can use. Each quota represents a specific countable resource, such as API calls to a particular service, the number of load balancers used concurrently by the project, or the number of projects that one can create.

There are two types of quotas:

Allocation quotas are used to restrict the use of resources that don't have a rate of usage, such as the number of VMs. Allocation quotas don't reset over time. Some quotas are global while others are regional or zonal. For example, there are separate limits for how many Compute Engine VM instances that you can create in each Google Cloud region.

Quotas can be viewed in the Cloud Console, using the gcloud command-line tool, using the Service Uage API and as quota metrics in Cloud Monitoring. Specific IAM permissions are needed to view project quotas.

From the command line, the below commands can be used

```
gcloud alpha services quota list \

--service=SERVICE_NAME.googleapis.com \

--consumer=projects/PROJECT_ID
```

To view the same service's quota details for an organization:

```
gcloud alpha services quota list \

--service=SERVICE_NAME.googleapis.com \

--consumer=organizations/ORG_ID
```

Use cloud monitoring / set up Dashboards on GCP to monitor resource usage

• How would you approach ensuring that the same analyses aren't repeated by multiple users? If you are aware of specific resources from well-known cloud service providers (AWS, Google Cloud, Microsoft Azure, etc.) that could be leveraged for this task, please discuss their implementation.

1. **Google Cloud provides a robust set of billing and management tools,** that can give one the visibility and insights needed to keep up with cloud deployment.Billing reports give an at-a-glance view of the costs. Custom dashboards can be built for more granular cost views.
2. Only pay for the compute used using the steps below
3. **Identify idle VMs (and disks**
4. **Schedule VMs to auto start and stop**
5. **Rightsize VMs**:
6. **Leverage preemptible Vms**: These are highly affordable compute instances that live up to 24

hours and that are up to 80% cheaper than regular instances. Preemptible VMs are a great fit for fault tolerant workloads such as big data, genomics, media transcoding, financial modelling and simulation.

7. **Optimize Cloud Storage costs and performance**
8. **Cloud storage offers multiple storage classes:** standard, nearline, coldline and archival, all with varying costs and their own best fit use cases.
9. **Lifecycle policies**: Money can be saved automatically with object lifecycle management. By configuring a lifecycle policy, one can programmatically set an object to adjust its storage class based on a set of conditions—or even delete it entirely if it's no longer needed.
10. **Deduplication to identify workflows that are being repeated by end users**
11. **Tune the data warehouse:** Usage of BigQuery
12. **Enforce controls**: To limit query costs, use the maximum bytes billed.
13. **Use partitioning and clustering**: Partioning and clusterring tables whenever possible, can help greatly reduce the cost of processing queries, as well as improve performance. BigQuery automatically drops the price of data stored by 50% for each partition or table that hasn't been edited over some time. It is more cost-effective and convenient to keep data in BigQuery rather than going through hassles of migrating it to lower tiered storage.
14. **Check for streaming inserts**: Load data into BigQuery as a batch job.
15. **Use Flex Slots**: By default, BigQuery charges variable on-demand pricing based on bytes processed by queries. For high-volume customer with stable workloads, it may be more cost effective to switch from on-demand to flat rate pricing. Flex Shots is a new way to purchase BigQuery slots for duration as short as 60 seconds, on top of monthly and annual flat-rate commitments.
16. **Cloud Platform SKUs** is a quick way to identify how much one is spending on a given Google Cloud service. Network Topology is a module of Network Intelligence Center and provides comprehensive visibility into the global GCP deployment and its interaction with the public internet, including an organization-wide view of the topology, and associated network performance metrics. This allows idedntification of inefficient deployments
17. **Network Service Tiers**: Google Cloud lets one choose between two network service tiers
18. **Cloud Logging**: Control over network traffic visibility by filtering out logs. The same applies to Data Access Audit Logs, which can be quite large and incur additional costs.