Google Cloud

Whitepaper
Nov 8 2022

Google Cloud and Oracle Cloud Infrastructure – making the most of multicloud

Table of Contents

Table of Contents

Google Cloud

Introduction

This document describes an approach for building composite multicloud systems that combine resources into a single, fully integrated platform. Specifically, this use case uses a private interconnect to combine components running in Oracle Cloud Infrastructure (OCI) with resources running on Google Cloud. While the actual implementation details can vary significantly, we hope that this information will provide a blueprint that you can follow to connect components running across a variety of public clouds.

Multicloud

When creating architectures for cloud applications, administrators often co-locate applications and databases in the same zone or region of the same cloud. This is similar to the common practice of locating all tiers in the same on-premises data center.

Back in 2018, Gartner found that most enterprises were already using more than one cloud provider. They also discovered that companies rated the ability to use the best-of-market capabilities from multiple cloud providers as one of the main reasons for adopting a multicloud strategy. Gartner further defined two different types of multicloud architectures:

- A redundant architecture, where a single cloud contains the components for an application. To achieve redundancy, you host additional, self-contained instances of your application on another cloud of your choice.
- A composite architecture, where a single workload uses components from more than one cloud.

In a more specific example of a composite architecture, an organization looking to modernize applications might select Google Cloud for their future workloads because of the capabilities and options available in Google Compute Engine (GCE), Google Kubernetes Engine (GKE), serverless, and other Google Cloud services. But that workload might still depend on an Oracle database. To reduce operational overhead, the organization might consider Oracle Autonomous Database on OCI. Selecting both Google Cloud and OCI gives an organization the best of both worlds.

---

1. *Gartner Research: Technology Insight for Multicloud Computing, August 16th 2018*

However, when evaluating a composite architecture, the major concerns for users include added latency and complexity. Many applications are sensitive to latency between the application and database tiers, which leads to the practice of co-locating these resources as close to each other as possible. Furthermore, there is a lot of technical complexity involved in connecting different clouds with different networking technologies, physical and logical resource hierarchies, security, identity services, and more.

It's important to consider your options when you implement a composite architecture, find ways to reduce complexity, and assess the impact of this multicloud strategy on your application performance.

Reasons for adopting composite multicloud architectures

While the most popular reason that companies choose a multicloud approach is to avoid vendor lock-in, there are several more technically-oriented goals that drive multicloud adoption. The primary drivers include resiliency and the desire to build best of breed systems that combine unique or differentiated public cloud offerings. This gives you the freedom to select any vendor that provides these services. The advantage of this approach is the potential to do things that are simply impossible on a single cloud.

Reasons for not adopting composite multicloud architectures

On the other hand, the major downsides of systems containing components that span more than one cloud are the inherent laws of physics and the latencies introduced by such an architecture. Manageability can also be problematic in such systems, as the cloud vendors generally do not offer monitoring or management solutions in a single pane of glass that can view components running on multiple clouds.

What we tested

We tested several common use cases to address potential issues seen in a composite architecture. We explored how to handle the latency issues when you spread components of a single system across two or more clouds. We also created a rudimentary single pane of glass monitoring solution to provide visibility to the resources housed in both clouds.

Private interconnect between Google Cloud and OCI

Using a highly-available, private, low-latency connection provided by a supported partner such as Equinix, you can extend your Google Cloud network to reach other clouds. Once the Equinix Fabric connection is in place, you can use Oracle's FastConnect to connect OCI to Google Cloud, and use Google Cloud's Partner Interconnect to connect Google Cloud to OCI. The Partner Interconnect **allows your traffic to completely bypass the public internet.** In the following sections, we review the major concepts that enable the interconnect and provide a dive deep on how to set up an interconnect between Google Cloud and OCI using Equinix.

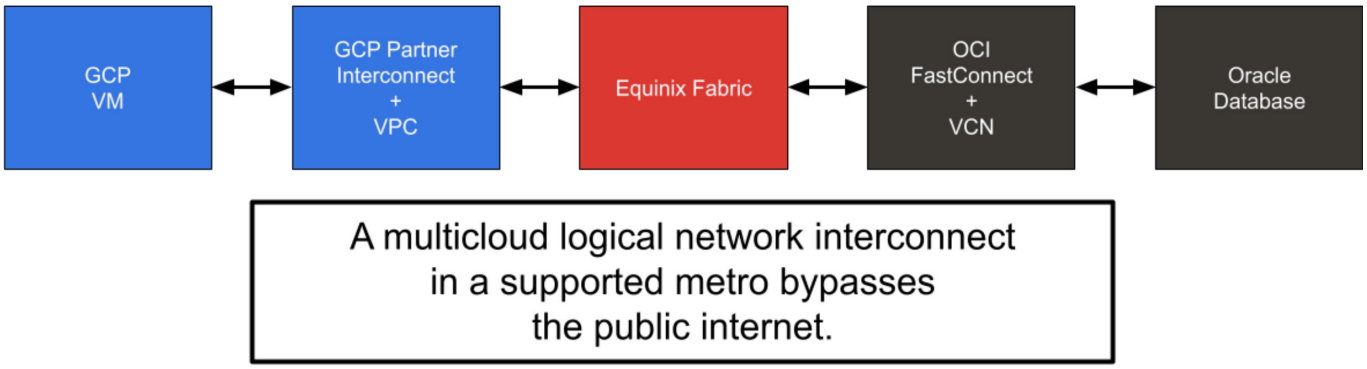
Google Cloud Platform (GCP)

A Google Cloud Partner Interconnect enables you to connect your Google Cloud Virtual Private Cloud (VPC) to your OCI Virtual Cloud Network (VCN) so that your VCN's internal IP addresses are accessible directly from Google Cloud and vice-versa. You do not need to use a NAT device or VPN tunnel to reach internal IP addresses.
Figure 1 shows a high-level overview of how an Equinix Fabric connects Google Cloud and OCI.

---

2. as defined in the Google Cloud customer agreements

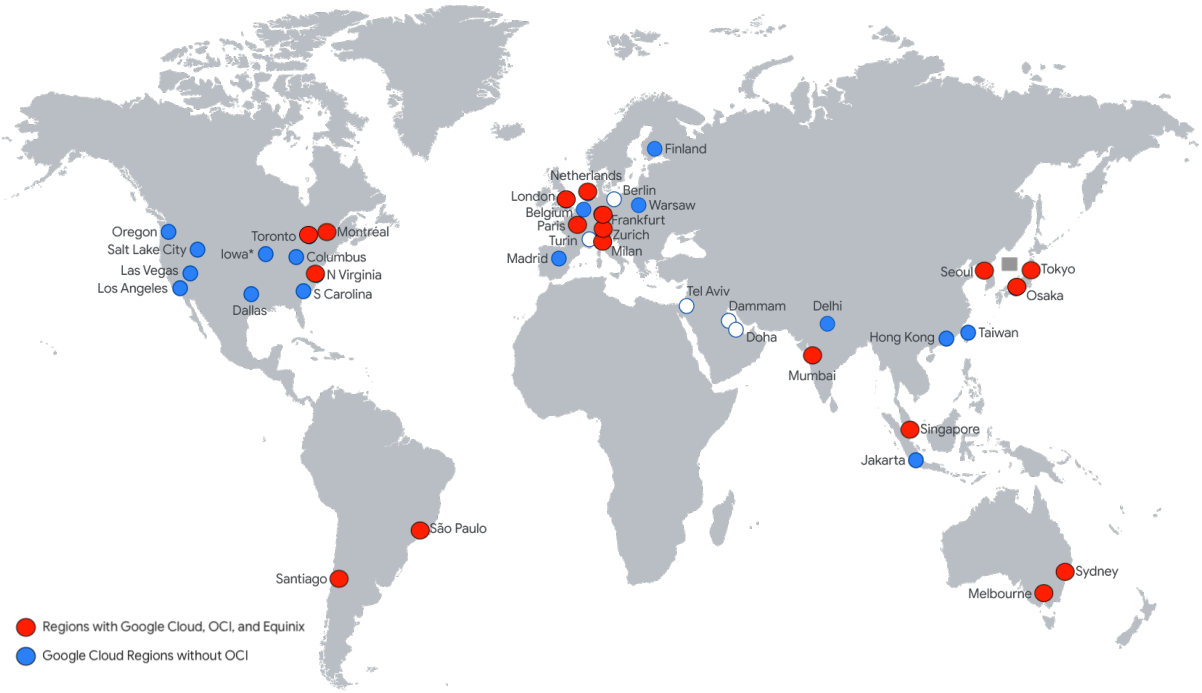
Figure 1: Connecting Google Cloud and OCI across an Equinix Fabric



Equinix links Google Cloud and OCI through their Network Edge service. This service allows businesses to create virtual routers that serve as the connection points between OCI's FastConnect and Google Cloud's Partner Interconnect. Equinix offers a variety of vendors for these virtual routers and allows customers to bring their own license or use a license provided by Equinix.

Equinix operates their Network Edge service in a large number of metropolitan areas. To achieve latencies as low as 2 milliseconds, you need to select metro areas where Google Cloud and OCI are co-located. In the following map shown in Figure 2, regions marked in red depict regions where Google Cloud, Equinix, and OCI are co-located as of October 2022.

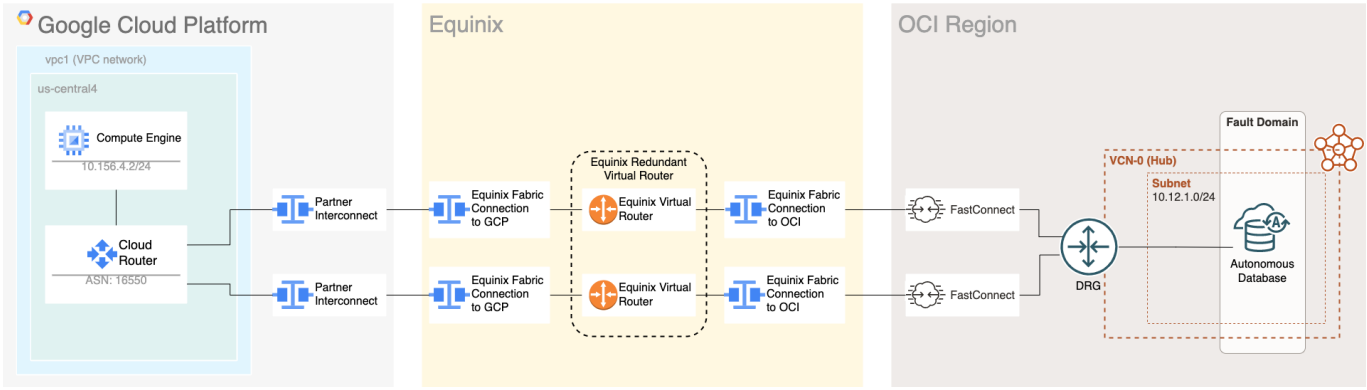
Figure 2: Regions where Google Cloud, OCI, and Equinix are co-located



Cross-cloud connectivity between Google Cloud and OCI uses each cloud's edge routing devices: Cloud Routers (Google Cloud) and a Dynamic Routing Gateway (OCI). Each routing device advertises routes to their managed subnets by way of the Border Gateway Protocol (BGP).

Enterprise-level connectivity requires more than a single path between clouds. Equinix can automatically create redundant virtual routing devices. You should configure redundant Partner Interconnects in Google Cloud, and multiple FastConnect circuits in OCI. Figure 3 shows a high-level representation of a redundant architecture between Google Cloud and OCI.

Figure 3: Cloud interconnect redundancy between Google Cloud and OCI



Testing composite architecture applications

After connecting the cloud platforms, you can run the databases and the application components that connect to those databases across different cloud platforms. However, the nature of your application will determine the technical viability of your plan.

To evaluate the performance of a multicloud application, we performed infrastructure tests using a popular database load generator to generate an application workload in two distinct infrastructure architectures:

1. Oracle Database (OCI) ⟷ Simulated Application (OCI)

2. Oracle Database (OCI) ⟷ Simulated Application (Google Cloud)

We chose these configurations to create a head-to-head test showing the impact of additional latency when the application was not co-located with the database.

For each of these architectures, we ran the load generator with both a low (10 ms) and a high (100 ms) maximum "think time." The term "think time" refers to the amount of time between actions performed by the simulated application user. Think time is added to a test suite in order to mimic the real-world use of an application.

We selected the load generator's standard Client Side Order Entry workload for a reason. By having application logic on the client side (Java) rather than in the database (PL/SQL), the workload becomes more sensitive to network latency between the database and the application tier, and creates a strenuous test. We expect such network latency to be slightly higher in a multicloud architecture.

We used a single-core Oracle Autonomous Database and sized the application tier in each cloud with sufficient CPU to ensure acceptable performance. We found through testing that an increase in the application user count above 40 users produced no additional improvement in transactional throughput. As a result, we used a standard user count of 40 users for all tests.

Table 1 shows the key metrics for low and high think-time application configurations in the application tier for each deployment model.

Table 1: Single cloud compared to composite multicloud across multiple metrics

Maximum think time	10 ms		100 ms	
Component deployment model	Single cloud	Composite multicloud	Single cloud	Composite multicloud
Total transactions	129,979	108,236	17,353	17,357
Average transactions per second	433.26	360.79	57.84	57.86
Average response time (ms)	31.30	49.88	37.10	51.94
Minimum response time (ms)	2	4	2	4
Maximum response time (ms)	1499	467	619	349

The results show the impact of increased network latency on transaction throughput for demanding, low think-time environments. As shown in the table, when application demand reduces (think time increases), the impact of network latency on transaction throughput falls to zero. This occurs at a surprisingly low think-time setting. For many applications, such a small increase in latency has little or no effect on the application.

Migrating data

One of the most common use cases for databases is data integration. To explore this, we executed some tests focused on data extraction from the Oracle environment. The goal of these tests was to show how to stream data from a database in OCI to BigQuery and Cloud Spanner for analytics and AI purposes. While our tests focused on moving data into Google databases, you can just as easily share data from a Google database to Oracle.

Because Autonomous Data Warehouse (ADW) instances do not officially support the LogMiner and XStream APIs, we used Oracle GoldenGate with the BigData Adapter as our change data capture (CDC) method for these database types. One of the goals of this test configuration was to ensure that you can replicate data directly from ADW over the private interconnect to BigQuery, Cloud Spanner and Google Cloud Storage.

For both of these tests, we used a standard installation of Oracle GoldenGate (OGG) 21c. Once we configured the necessary supplemental logging within ADW, replication worked as expected and we were able to load data into BigQuery. Because OGG cannot run directly on Oracle Autonomous Database, we used a hub infrastructure configuration. You can place the OGG hub infrastructure (typically a single VM) in either cloud. However, to minimize the amount of data transferred, we placed the hub in OCI so that OGG compression could be used across the interconnect.

2. *According to the dedicated instance feature list: https://docs.oracle.com/en/cloud/paas/autonomous-database/dedicated/adbdg/index.html#articletitle*

Figures 4 and 5: Initial load extract

```
GGSCI (test-ogg) 6> !
info e_init

Extract    E_INIT    Last Started 2022-08-11 18:58   Status RUNNING
Checkpoint Lag       Not Available
Process ID           23523
Log Read Checkpoint  Table HR.EMPLOYEES
                     2022-08-11 18:58:02  Record 1
Task                 SOURCEISTABLE


GGSCI (test-ogg) 7> !
info e_init

Extract    E_INIT    Last Started 2022-08-11 18:58   Status STOPPED
Checkpoint Lag       Not Available
Log Read Checkpoint  Table HR.CUSTOMERS
                     2022-08-11 18:58:04  Record 55500
Task                 SOURCEISTABLE
```

```
Database Version:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production.

2022-08-11 18:57:56  INFO    OGG-25341
Database Language and Character Set:
NLS_LANGUAGE     = "AMERICAN"
NLS_TERRITORY    = "AMERICA"
NLS_CHARACTERSET = "AL32UTF8".

2022-08-11 18:57:56  INFO    OGG-26025
Database Unique Name: eyz1pod.

2022-08-11 18:57:57  INFO    OGG-06509  Using the following key columns for source table HR.EMPLOYEES: EMPLOYEE_ID.

2022-08-11 18:57:57  INFO    OGG-06509  Using the following key columns for source table HR.CUSTOMERS: CUST_ID.

2022-08-11 18:58:02  INFO    OGG-01888  TCP network is configured as

                OS DEFAULT          SPECIFIED           ACTUAL VALUE
        IP_DSCP    0                  N/A                 0
         IP_TOS    0                  N/A                 0
    TCP_NODELAY    0                  N/A                 0
    TCP_QUICKACK   1                  N/A                 1
       TCP_CORK    0                  N/A                 0
      SO_SNDBUF    8192               N/A                 8192
      SO_RCVBUF    43690              N/A                 43690.

2022-08-11 18:58:02  INFO    OGG-01478  Output file /opt/ogg/gghome1/dirdat/il is using format RELEASE 19.1/21.1.

2022-08-11 18:58:02  INFO    OGG-02911  Processing table HR.EMPLOYEES.

2022-08-11 18:58:03  INFO    OGG-02911  Processing table HR.CUSTOMERS.
```

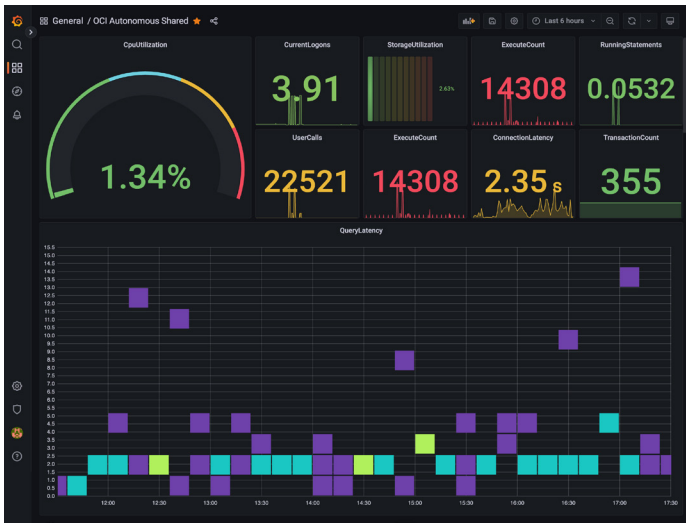
To replicate data into Cloud Spanner, there is a bit more complexity because there is no native GoldenGate adapter for Cloud Spanner. We used Confluent for these tests because Google Cloud Marketplace has a Confluent image readily available, which minimized the infrastructure required to get started. In this configuration, we used OGG 21c to extract data from ADW and the OGG Big Data Adapter, and we used Kafka Connect to connect to Cloud Spanner.

The technical constraints of ADW required us to use OGG for data extraction. Note that OGG is not a requirement for other Oracle database services, such as ExaCS, DBCS, or Cloud at Customer.

Unified monitoring of multicloud components

The increased availability multicloud offers comes at the price of increased operational and maintenance overhead. Proper monitoring becomes a critical component to manage your multicloud environment. Fortunately, with modern monitoring tools such as Prometheus, OpenTelemetry, and Grafana, you can create a unified dashboard that monitors components across both clouds. Figure 6 shows an example of a dashboard we created. minimize the amount of data transferred, we placed the hub in OCI so that OGG compression could be used across the interconnect.

Figure 6: Monitoring dashboard using common tools

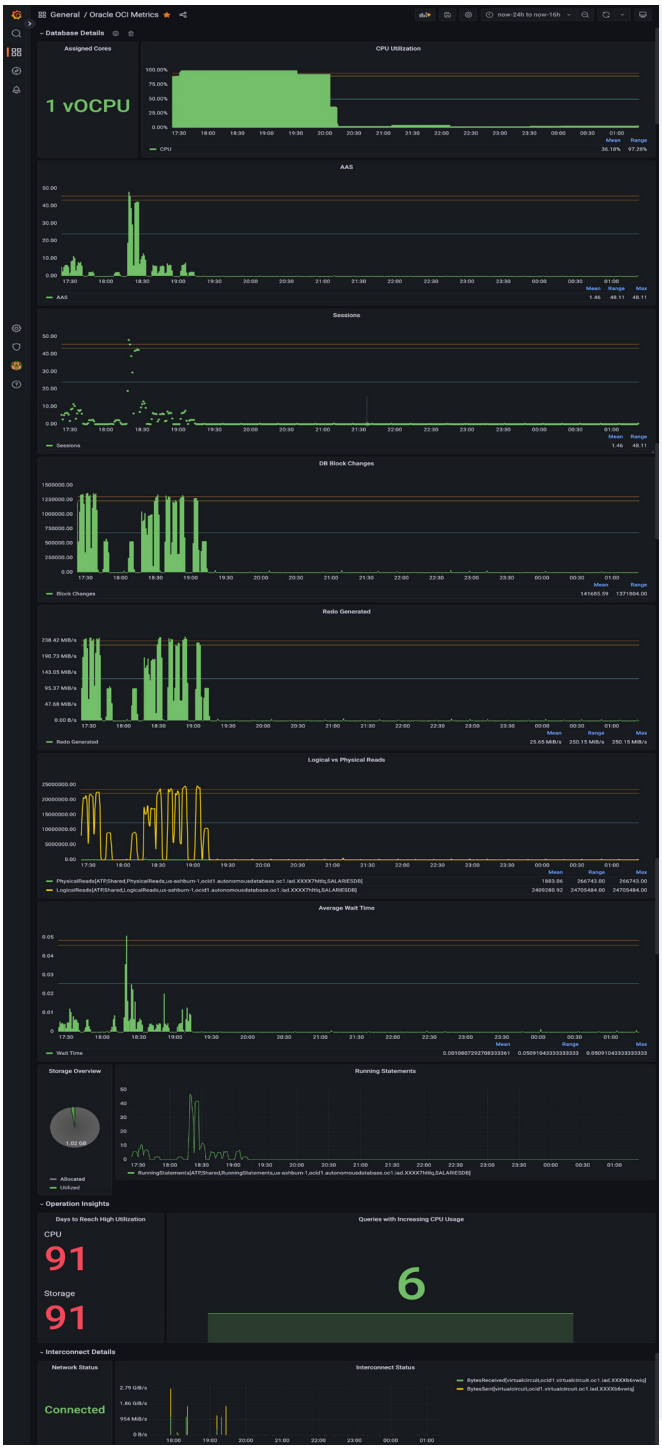


3. *https://grafana.com/grafana/dashboards*

You can use the public Oracle OCI API to gather metrics for databases running in OCI, including ADW.
This option offers a “zero-impact” way to fetch and collect performance data and other metrics. You can also use an OCI plugin that works directly within Grafana to gather these metrics. There are also a number of pre-built dashboards for monitoring most OCI components available on the Grafana marketplace.[3]

Figure 7 shows a more comprehensive dashboard running in Google Cloud that we used to monitor the environment during our load tests.

Figure 7: Extended multicloud monitoring dashboard



Conclusion

Google Cloud supports users in their choice of hybrid cloud or multicloud architectures that work best for them. Google Cloud aims to be the most open public cloud, offering a combination of open-source software and the ability to configure low-latency connectivity to other clouds. This combination enables the deployment of robust multicloud systems and helps minimize vendor lock-in.

In this document, we have shown an approach to building multicloud deployments that use currently available technologies. One of the main building blocks is the use of a private interconnect, such as the Equinix Network Edge. Once you provision the interconnect, you can connect components of systems across multiple clouds in a seamless manner. We addressed several common use cases, including:

- Streaming data from one cloud to another for further processing, as shown in our Oracle ADW to Google BigQuery example
- Backing up databases from one cloud to another
- Monitoring components running on multiple clouds in a single pane of glass

We also demonstrated a true multicloud system, where the application and database tiers were split between two clouds (OCI and Google Cloud). We compared the performance of an application configured in a multicloud environment to the same application configured with all the components located in a single cloud. As expected, the multicloud deployment added some latency. However, for many applications, such a small increase in latency is more than acceptable to gain the additional flexibility, best of breed functionality, and new opportunities opened up by this multicloud approach.