What it is?
Cloud containerization offers a more modern solution to traditional virtualization by offering a lightweight, portable and efficient approach. Containers are aptly named, offering a packaged solution with only the desired application and its dependencies. As per Google's own definition, a container is a method that "bundles an application's code with all the files and libraries it needs to run" (Google Cloud, 2025)

Unlike virtualization, containers do not need their own guest OS and instead use the host machines shared OS while keeping their own processes, packages and networking isolated. Additionally, containers offer a faster start up time, lower resource usage and much more portability. In comparison to bare metal, containers are the clear winner offering more flexibility, ease of deployment, isolation and reproducibility.

Cloud Containerization
In distributed systems, containers play a crucial role in enabling multi-cloud and hybrid-cloud architectures. Their lightweight design, operational agility, and cost-efficiency make it possible to deploy services across cloud providers with minimal overhead. Among these advantages, **image portability** stands out as one of the core enablers of modern cloud-native systems.

Containers encapsulate an application and its dependencies into a consistent image format, ensuring that the same artifact can run reliably across different environments, such as: public clouds, private data centers, edge nodes, or developer machines. In the broader body of research, including systematic studies on multi-cloud containerization, it is repeatedly identified that image portability is a dominant theme in cloud-native architectures. For instance, Waseem et al. (2025) identify portability as a foundational motivator for multi-cloud strategies, emphasizing how consistent container images ease migration and interoperability across cloud ecosystems.

For performance, there is not a one size fits all approach when it comes to virtualization and containerization. However, many studies bolster the argument in favor of containers noting that they exhibit significantly improved performance and a reduced overhead in cloud settings (Pahl et al. 2019).

Cloud and serverless (a hybrid approach)
We are using a mixture of both serverless and containers. This allows us to maximize the advantages in each respective system. Containers allow us to have full control over the environment as well as manage long-running and stateful workloads. Serverless helps maintain a simple, stable and low maintenance system compared to a full server approach and is ideal for our event-driven and stateless tasks.

The other alternative to a cloud container approach would be a bare metal system, which is both costly and lack the flexibility to scale compared to the cloud equivalent. They do however tend to outperform containers in more specialized or performance-critical workloads.

Cloud Run Containerization Job

To execute the aforementioned hybrid architecture, we utilize **Cloud Run containerization jobs** on Google Cloud Platform (GCP) to manage and run our operational workflows. These jobs are fully serverless and leverage the inherent **portability** of containerized workloads. Cloud Run automatically manages resource scaling; including CPU allocation, concurrency handling, and traffic-based autoscaling (scaling down to **zero** when idle and up rapidly during bursts in demand) making it highly effective for distributed systems requiring elasticity and variable workloads (Google Cloud, 2025) . Cloud Run integrates natively with other GCP services such as **Pub/Sub** for event-driven execution and **BigQuery** for data processing pipelines, enabling seamless interoperability across components in our system.

A Cloud Run job executes a container image to run short-lived compute processes without provisioning or managing any underlying servers. Because Cloud Run is fundamentally based on **containerization**, it inherits the core strengths of container-based systems—environmental consistency, deployment agility, and platform portability. These attributes are widely recognized in the containerization literature and are repeatedly emphasized as key advantages for cloud-native and distributed system designs (Cherukuri, 2024) ; (Waseem et al., 2025) . Consequently, Cloud Run serves as an ideal execution model for our distributed system use case, where reliability, rapid scaling, and hybrid-cloud interoperability are critical.

Challenges

Containers (both cloud and bare-metal) do however come with their own set of challenges, largely around security. Though containers are isolated, they do share the host OS/kernel which introduces risks such as privilege escalation, container breakout attacks and vulnerable base images. If for example, you are using the same container image and build on 100 different servers, and it has a backdoor (via your code or potentially a package backdoor) then you now have a security vulnerability across all of your servers. Beyond security, there are also challenges that exist around scalability when it pertains to multi-cloud, both in a performance capacity (effective resource use, latency handling) and in an availability aspect (reliability across multi-cloud) (Waseem et al., 2025). For our use case however, performance for a single system remains a non-issue and the challenges around scalability would be addressed when or if multiple instances are executed at once.

Future Iterations

As we are currently only looking at one exchange with our build, we could expand this solution to look at multiple exchanges or markets based in different countries. If such an approach was taken, we can use our existing build template and then look at using CDN to handle a more worldwide or multi-country approach. For example, we may want to use our strategy to allocate funds in the NASDAQ but also look at the same approach on assets listed on the Hong-Kong exchange. The time difference of operation, as well as latency risk by running our US hosted build makes our current build non-viable without configuration changes. Our containerized approach however will make it easy to mimic the behavior in a more locally suited area (such as in China, or close proximity) and control what time we want to execute our operations (via cloud run) to align with the opening of each respective market.

Key Points for Slides

- Cloud Run containerization jobs execute fully serverless, containers, enabling rapid, on-demand compute without managing any underlying infrastructure.
- Automatic scaling in Cloud Run such as down to zero when idle and instantly up during bursts makes it ideal for distributed and event-driven architectures.
- Because Cloud Run runs standard container images, it inherits the portability, consistency, and agility of containerized systems across hybrid and multi-cloud environments.
- Native integrations with Pub/Sub, BigQuery, and other GCP services streamline data pipelines and distributed workflows.
- Cloud Run's short-lived, stateless job model is highly cost-efficient for operational workloads that do not require persistent servers.

References

Cherukuri, B. R. (2024). *Containerization in cloud computing: comparing Docker and Kubernetes for scalable web applications*. International Journal of Science and Research Archive, 13(01), 3302–3315.

Google Cloud. (2025). *What is containerization? How does it work?*. https://cloud.google.com

Pahl, C., Brogi, A., Soldani, J., & Jamshidi, P. (2019). *Cloud container technologies: A state-of-the-art review*. IEEE Transactions on Cloud Computing, 7(3), 677–694.

Waseem, M., Ahmad, A., Liang, P., Akbar, M. A., Khan, A. A., Ahmad, I., Setälä, M., & Mikkonen, T. (2025). *Containerization in Multi-Cloud Environment: Roles, Strategies, Challenges, and Solutions for Effective Implementation*. arXiv:2403.12980.