maintained in the source code repository.

## Technology Preview

J2EE deployment has given way to “container” based deployment (i.e., Docker.) I am currently working on AWS Lambda service deployment, which leverages AWS services to support my REST-based JSON web services, including security.

I am currently migrating from SOA to REST on Cloud, and this has required adding supporting mechanisms because of increased API end-points. These additional mechanisms include:

API Gateway: An API gateway handles authentication, edge routing and public API exposure

Service Mesh: As service end-points proliferate moving from SOA to Micro services; it is no longer feasible to configure a service’s end-point dependencies. A *service mesh* offers the ability to provide this capability. AWS and Google each offer their own service mesh iterations: AWS’ is called App Mesh and Google uses Istio.

Note: A service mesh's primary purpose is to manage internal service-to-service communication, while an API Gateway is primarily meant for external client-to-service communication.

Data Grid: Service invocations introduce latency and require more time to execute. As applications become more distributed, and use remote dependencies to compute values, the remote access becomes more of a performance problem. Many of us have used various caching techniques to combat this problem. Distributed in-memory and in-process caching mechanisms help. I have used Redis and Ehcache for these two approaches. However, for data science processing, moving the processing to the data via Hadoop/HDFS greatly increases processing performance.

APP MESH

Airflow: <https://aws.amazon.com/blogs/machine-learning/build-end-to-end-machine-learning-workflows-with-amazon-sagemaker-and-apache-airflow/>.