OSAMA MUSTAFA وَمَا تَوْفِيقِي إِلَّا بِاللهِ
TECHNOLOGY GEEK

For the people who think differently Welcome aboard

Migrating to Serverless

Posted on October 11, 2021 by Osama Mustafa in Cloud
we'll look at considerations for migrating existing applications to serverless and common ways for extending the serverless

At a high level, there are three migration patterns that you might follow to migrate your legacy your applications to a serverless model.

Leapfrog

As the name suggests, you bypass interim steps and go straight from an on-premises legacy architecture to a serverless cloud architecture

Organic

You move on-premises applications to the cloud in more of a "lift and shift" model. In this model, existing applications are kept intact, either running on Amazon Elastic Compute Cloud (Amazon EC2) instances or with some limited rewrites to container services like Amazon Elastic Kubernetes Service (Amazon EKS)/Amazon Elastic Container Service (Amazon ECS) or AWS Fargate.

Developers experiment with Lambda in low-risk internal scenarios like log processing or cron jobs. As you gain more experience, you might use serverless components for tasks like data transformations and parallelization of processes.

At some point in the adoption curve, you take a more strategic look at how serverless and microservices might address business goals like market agility, developer innovation, and total cost of ownership.

You get buy-in for a more long-term commitment to invest in modernizing your applications and select a production workload as a pilot. With initial success and lessons learned, adoption accelerates, and more applications are migrated to microservices and serverless.

Strangler

With the strangler pattern, an organization incrementally and systematically decomposes monolithic applications by creating APIs and building event-driven components that gradually replace components of the legacy application.

Distinct API endpoints can point to old vs. new components, and safe deployment options (like canary deployments) let you point back to the legacy version with very little risk.

New feature branches can be "serverless first," and legacy components can be decommissioned as they are replaced. This pattern represents a more systematic approach to adopting serverless, allowing you to move to critical improvements where you see benefit quickly but with less risk and upheaval than the leapfrog pattern.

Migration questions to answer:

- What does this application do, and how are its components organized?
- How can you break your data needs up based on the command query responsibility (CQRS) pattern?
- How does the application scale, and what components drive the capacity you need?
- Do you have schedule-based tasks?
- Do you have workers listening to a queue?
- Where can you refactor or enhance functionality without impacting the current implementation?

Application Load Balancer vs. API Gateway for directing traffic to serverless targets

Application Load Balancer	**Amazon API Gateway**
Easier to transition existing compute stack where you are already using an Application Load Balancer	Good for building REST APIs and integrating with other services and Lambda functions
Supports authorization via OIDC-capable providers, including Amazon Cognito user pools	Supports authorization via AWS Identity and Access Management (IAM), Amazon Cognito, and Lambda authorizers
Charged by the hour, based on Load Balancer Capacity Units	Charged based on requests served
May be more cost-effective for a steady stream of traffic	May be more cost-effective for spiky patterns
	Additional features for API management: Export SDK for clients Use throttling and usage plans to control access Maintain multiple versions of an APICanary deployments

Consider three factors when comparing costs of ownership:

- The **infrastructure cost** to run your workload (for example, the costs for your provisioned EC2 capacity vs. the per-invocation cost of your Lambda functions)
- The **development effort** to plan, architect, and provision resources on which the application will run
- The costs of your team's **time to maintain** the application once it is in production

Reference

- Choosing (https://pages.awscloud.com/NAMER-field-GC-Deloitte-TCO-whitepaper-2019-learn.html)serverless (https://pages.awscloud.com/NAMER-field-GC-Deloitte-TCO-whitepaper-2019-learn.html)or traditional infrastructure? (https://pages.awscloud.com/NAMER-field-GC-Deloitte-TCO-whitepaper-2019-learn.html)

Cheers

Osama

Tagged AWS, AWS Services, Migrating to Serverless, Osama, Osama Cloud, Osama mustafa, Osama mustafa blog, osama oracle

Published by Osama Mustafa

Osama considered as one of the leaders in Cloud technology, DevOps and database in the Middle-East. I have more than ten years of experience within the industry. moreover, certfied 4x AWS , 4x Azure and 6x OCI, have also obtained database certifications for multiple providers. In addition to having experience with Oracle database and Oracle products, such as middle-ware, OID, OAM and OIM, I have gained substantial knowledge with different databases. Currently, I am architecting and implementing Cloud and DevOps. On top of that, I'm providing solutions for companies that allow them to implement the solutions and to follow the best practices.
View all posts by Osama Mustafa