STACKERY

Enterprise Serverless Use Cases

EXECUTIVE SUMMARY

Over the past several years, there has been accelerating adoption of serverless application architectures within modern enterprise organizations. Benefits such as decreased infrastructure costs and improved time to market are major factors driving this trend. Adoption of micro-service architecture patterns and DevOps are further accelerating the shift toward serverless technology.

As the number of companies taking advantage of serverless technology has increased, a growing list of common enterprise use cases has emerged. For these common enterprise needs, serverless application architecture offers major cost and time to market benefits over traditional architectures. Leaders looking to bring the advantages of serverless into their own organizations generally have questions about where serverless patterns are best applied and how to navigate integration with existing systems. In this document we provide an overview of the qualities that define serverless application architectures and discuss common enterprise use cases and patterns where leading edge enterprises are gaining a competitive advantage through the use of serverless technology.

OVERVIEW OF SERVERLESS TECHNOLOGY

At its core, serverless technology is an infrastructure abstraction which enables developers to focus on writing software and managing application performance, rather than deploying, maintaining, and scaling the underlying server infrastructure -- tasks which are offloaded to the cloud vendor.

Serverless Application Architectures have several key characteristics:

1. Emphasis on using ephemeral compute and external services as the building blocks of the system.
2. Close correspondence between resources used and resources billed. For example, the pay-per-use pricing of Functions-as-a-Service (FaaS) platforms such as AWS Lambda.
3. Managed execution environment that enables on-demand scaling and high availability by default, with reduced operational responsibility for the enterprise.

Typical serverless applications will use one or more FaaS functions to provide compute and integrate with data storage systems, distributed queues, and/or HTTP endpoints. These technical building blocks can be flexibly combined to meet a wide variety of business use cases where low operational costs, immediate scalability, and fast time to market are desirable.

1 INGESTION AND PROCESSING OF ANALYTICS DATA

Overview

All modern enterprises ingest large amounts of analytics data from web properties and other sources. This data is often siloed across multiple internal and 3rd party analytics providers, leading to inefficiency and blind spots. Serverless technology is a fit for building the highly scalable, event-driven data ingest pipelines used for event collection and data processing of analytics data. A common enterprise scenario would be to create an application to handle the collection of a new data source (e.g. telemetry data from a new product), or to create an application which provides a unified analytics pipeline which collects all analytics events, performs any necessary processing, and forwards to one or more third party or in-house tools for analysis.

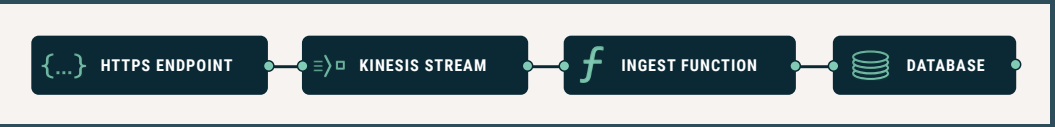
Benefits

- On-demand scaling is an ideal fit for data collection workloads.
- Provides flexibility to rapidly integrate various data collection and analysis tools to meet evolving business needs.
- Pay-per-use billing model is generally more cost effective than traditional pay-to-provision models.
- Resilient to large traffic spikes with minimal risk of data loss.

Considerations

- Cold start and concurrent execution limits should be considered when evaluating system reliability needs.
- Architecture should generally aim to decouple network ingest from data processing to avoid data loss and effectively respond to upstream system backpressure.
- Event-streaming technologies like AWS Kinesis Streams play a central role in these data pipelines.

Example Architecture



2 API INTEGRATION LAYER

Overview

Modern enterprises frequently need to integrate data and functionality from multiple existing applications, services and APIs. Often it is desirable to provide a consistent API frontend which interacts with multiple backend service APIs. This pattern is commonly used when it is desirable to provide multiple downstream teams with a simpler, more consistent integration point. This pattern is also commonly invoked when rewriting or replacing legacy APIs, by putting an API proxy layer in front of existing APIs, and incrementally augmenting or replacing existing functionality.

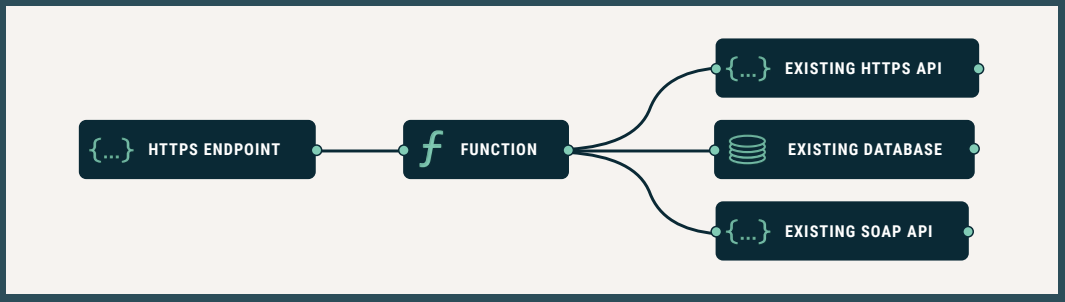
Benefits

- Centralization of complex system integration can accelerate application development.
- API proxy pattern decreases project risk in costly legacy rewrite projects by enabling incremental delivery.
- Pay per use billing model is generally very cost effective in these use cases.

Considerations

- Integration of multiple distributed systems requires careful reasoning about how upstream failures are handled.

Example Architecture



3 CONNECTED DEVICE (IOT) BACKEND

Overview

Many modern enterprises deploy applications that interact with customer devices such as mobile phones or Internet of Things (IoT) devices. For example, a coffee retailer may allow customers to submit orders via a mobile app, or a smart home device manufacturer may allow customers to control their home thermostat via an Alexa skill. Often, traffic levels fluctuate dramatically based on factors such as customer usage patterns and regional events. The scale-on-demand characteristics of serverless application architectures lends itself well to this use case, as it allows the system to incur minimal infrastructure cost during low load periods and rapidly scale to meet demand of high load periods. Additionally, pay-per-use billing can allow for increased cost modeling and reduce the risk of supporting these devices over the product lifespan.

Benefits

- Scale-on-demand architecture enables rapid scale up and high-availability, while incurring minimal costs during low traffic periods.
- Deliberate use of cloud provider regions, CDNs, and related cloud provider capabilities can decrease latency and improve customer experience.

Considerations

- Design of protocol for device/backend communication should address mechanisms for upgrading client software and managing changes in functionality over time.

Example Architecture

4 DIGITAL ASSET PROCESSING

Overview

Modern enterprises are often responsible for managing digital assets submitted by users. For example many applications provide collaboration or social media functionality where users upload photos, videos, and other media. Often, user submitted media must be put through a variety of processing steps, for example to resize images or transcode video bitrates. In some cases it may be desirable to use Artificial Intelligence (AI) APIs (e.g. AWS Lex, AWS Rekognition) to perform tasks such as tagging images, speech-to-text conversion, or flagging inappropriate content. These event driven workflows are an ideal fit for serverless application architecture, which provides mechanisms for functions to be invoked in response to an external event such as a file upload to an object storage service such as AWS S3.

Benefits

- Event-driven digital asset processing pipelines are a natural fit for serverless architectures and challenging to implement in traditional architectures.
- Compute capacity is spun up on demand in response to processing needs making costs low and predictable.

Considerations

- Data locality should be considered during architecture phase. Ideally compute and storage resources should be located within close geographic proximity (e.g. within the same AWS region) to minimize latency and network transfer costs.

Example Architecture

5 INTERNAL BUSINESS TOOLING

Overview

Every modern enterprise follows a set of internal business processes to coordinate projects, manage approval decisions, and collect and distribute information internally. These processes require coordination between multiple employees and often are unique to an organization. In many organizations teams exist to provide technical tooling around internal processes. For example an application that manages employee attendance of an annual security training course or an executive dashboard surfacing relevant information to key decision makers on a continuous basis.

Benefits

- Modeling business workflows that consist of a series of transactional steps (e.g. requesting and receiving approval) dovetails well with the transactional and event driven qualities of serverless application architecture.
- Functions-as-a-Service platforms are well suited to integrating diverse systems and generating a centralized view of data across an organization which is the key characteristic of Executive Dashboard applications.
- Generally, internal business tools have low computational requirements making pay-per-use billing model attractive.

Considerations

- In many cases 3rd-party SaaS solutions can be used to solve internal business tooling challenges without incurring ongoing development and maintenance costs. Serverless technologies are an attractive choice in this space when custom development is deemed necessary.

Example Architecture

6 PUBLIC FACING WEBSITE

Overview

Every modern enterprise manages multiple web properties which generally play critical business roles. While a variety of traditional architectures and PaaS solutions are appropriate solutions for this extremely common need, it is straightforward to develop public facing web sites and web applications using serverless technology. It is worth considering these benefits when planning new development.

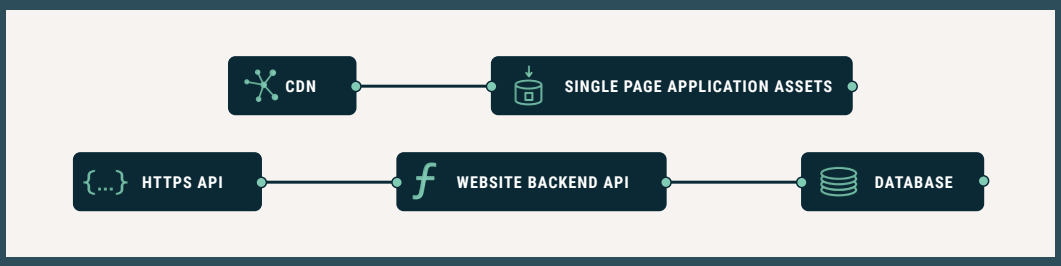
Benefits

- Hosting costs scale directly with site traffic and are often orders of magnitude cheaper than comparable pay-to-provision systems.
- The infrastructure abstraction provided by serverless architectures simplifies hosting setup process and can result in faster time to market.
- Easy integration with cloud provider's Content Delivery Network (CDN), SSL certificate management, and security features such as DDoS protection can provide significant benefits over alternatives.

Considerations

- There may be organizational benefits to avoiding introduction of new web hosting architectural patterns.
- Implementing new websites using serverless architecture can be used as a low risk way to "test the waters" and bring an engineering team up to speed on serverless best practices.

Example Architecture



7 MANAGED SOA PLATFORM

Overview

Many organizations have been migrating towards micro-service architecture patterns and DevOps process. With micro-service patterns, managing and orchestrating many independent services becomes more challenging. This has led to the rise of complex orchestration platforms (e.g. kubernetes, mesos, docker swarm) to provide features like dynamic scaling, fault tolerance, and service discovery. Serverless architecture components such as FaaS provide a lower management alternative with similar benefits, enabling larger organizations to provide internal engineering teams with a managed hosting and orchestration platform on which to deploy their applications.

Benefits

- Serverless technology provides similar benefits to container orchestration platforms such as Kubernetes with a lower infrastructure management burden and higher level of abstraction for application developers.

Considerations

- Vendor specific limits on execution time, memory, and disk space mean certain applications need significant changes to run in a serverless architecture. It is generally necessary to provide a variety of hosting options to accommodate a modern enterprises' full suite of application infrastructure needs.

Example Architecture