YASH Technologies

More than what you think.

AWS Microservices

Info@yash.com | www.yash.com

Abstract:

Microservices as a broad term is an approach to software development, to facilitate deployment cycles, and scaling through a defined ownership model. Software, consisting of micro independent services, owned by small teams communicate over well defined APIs.

Reputed management & consulting firms like McKinsey have recognised microservices as a process to increase the technology organization's ability to provide cross-unit and cross-application functions.

The Boston Consulting Group rightly points out the of microservices being like readymade packaged food, where you don't know where it is made, but it gets the job done.

In this white paper we brief AWS microservices basics, the benefits of AWS microservices deployment, the challenges it encounters, and how AWS helps address them.

Introduction:

As an industry that is fast paced, trends in product as well as application developments are an industry norm. With enterprises looking forward to turning leaner, more agile and move towards a DevOps approach, micro services fits the bill perfectly, with its focus on building single-function modules with distinct interfaces and operations.

Microservices architectures are not a radically different approach to the SLDC, but can be viewed as a collection of best practices in Agile methodology, API first design and Continuous delivery.

Microservices – Components & features:

Since microservices is a developmental, architectural approach, it is challenging to define its characteristics. However, all microservices, have distinct features as follows:

Independent :

Components in the architecture can be upgraded, changed or removed without affecting the functioning of other services or components. Also, understandably, the teams function independent of each other.

Expertise:

A microservice component is designed specifically to cater to a specific capability, that focuses only on a particular domain. If the component's code builds up to a certain level that exceeds the "micro" stage and gets complex, it is broken down into smaller services.

Build-Run:

DevOps is a principle where the team responsible for building a service is also responsible for maintaining, its up-keep and operating it, while maintaining close contact with the end users and constantly upgrading services to meet customer needs/ expectations.

Black box:

Components are designed such that they hide the complexity of their own design – like a black box. Communication happens via definite APIs and does not show any hidden dependencies between the components.

Decentralised:

These architectures are distributed systems with built in decentralised data management, with each service having its own miniature database to suit its own data model. They are a hyper local version of services – in every way of development, deployment and management.

Customised:

Microservices follow the principle of tailor making their services to suit requirements specific to each need. It therefore follows a heterogenous approach to every aspect including operating systems, databases, and tools – thus giving developers freedom to operate – a system popularly called polyglot persistence and programming.

Benefits of AWS Microservices:

One of the major reasons enterprises use microservices is to address the issues of scalability, complexity and agility that are associated with traditional on-premises or monolithic deployments. The major factors that turn into business drivers for deploying AWS Microservices are:

Agility:

With determined, and well established boundaries, and small teams that take entire ownership of services, teams work independently and fast, with quicker response times and smaller working cycles.

Innovation:

Microservices give the flexibility of choice to the developers where they can choose technologies, databases and frameworks thus giving room for creativity and innovation.

Quality:

Quality of code improves immensely with microservices due to a simple fact that the focus is on small bite sized modules with enhanced reusability, accountability and composition.

Scalability:

When an enterprise chooses to deploy microservices model, it means even large scale systems are broken down into appropriate modules, for optimal usage. The services can be scaled vertically or horizontally based on requirement.

Challenges of AWS Micro Services:

Although AWS Micro Services offers exciting possibilities, and all advantages as detailed above, as with all architectural methodologies, this approach comes with its own set of challenges. Some trade-offs that are inherent to the micro services are

• **Organization** – Since Microservices at its core is all about smaller teams, leaner organization, organising resources to form an effective team structure, streamlining activities that follow a DevOps approach is a challenge.

• **Migration** – When an enterprise considers migration from a traditional monolithic architecture to a microservices architecture, it is important to examine the right boundaries and resonsibility placed on micro services. This process is not easy, because it requires that the huge complex existing system be broken down into proper segments without causing disruption.

• **Distributed Systems** – While Microservices promises an effectively distributed system, it comes inherently with a set of issues collectively called the Fallacies of Distributed Computing

• **Versions** – Versioning for microservices can be a big problem, because the team needs to incrementally deploy new versions of a service in such a way that both old and new versions of a service contract are running simultaneously. Therefore, it's important to have a strategy for service versioning. There are several best practices like routing-based versioning, which can be applied at the API level.

AWS has a bouquet of benefits that address most of the above detailed challenges of microservices architectures:

• **On-demand resources** – At its very basic DNA, the AWS is a platform where resources are available on demand. Where traditional infrastructures are limited, AWS has no limit on resources. This address the challenge of provisioning and scaling resources.

• **Innovation** – Because AWS means the enterprise only pays for how much and what it uses, it allows room for innovation while not compromising on the risk exposure factor. New services and features can be tweaked, replaced, brought in or phased out if not successful.

• **Continuous Deliveries** – The cloud in general and the AWS platform in particular allows automation of the provisioning and deployment process. Simultaneous and continuous Integration with the development part of the application lifecycle is also extended to the operations part of the lifecycle. This aspect enables the adoption of Continuous Deployment and Delivery.

• **Managed services** – Managed services is one of the biggest offerings of the cloud. Managed services ease the provisioning virtual servers, configuring and optimizing software, and having reliable backups. Features like monitoring, security, availability and scalability are in built into those services. This reduces the operational complexity of running microservices.

• **Infrastructure as code** – In addition to using scripts to manage an infrastructure, AWS allows for management of whole architecture as a code in a version control system, just like in application code. Which automatically means any part of the infrastructure can be redeployed at any time. Rollbacks are no longer limited to the application—they can include the whole infrastructure.

• **Service orientation** – AWS follows a service-oriented structure where each AWS service focuses on solving a specific issue, and communicates with other services using definite APIs. This allows for a wonderful possibility where the enterprise can put together creations it wants from building blocks.

Conclusion:

Microservices as an architecture is a distributed approach, designed to overcome challenges and limitations of traditional SLDC models.

AWS offers a large spectrum of managed services, that help development teams choose better, build faster while minimising operational complexity.

Resources:

• AWS Whitepapers • AWS Architecture Blog • AWS Answers • AWS Documentation

AWS Glossary:

For updated AWS terminology, please see AWS Glossary in the AWS General Reference.