Cloud Agnostic Architecture / Beauty of being Cloud Agnostic by utilizing cloud services

by **Remya Jose & Shemeer PP** | April 16, 2020

Nowadays, most of the organizations utilize at least one of the public cloud solutions, such as cloud computing services such as storage, software, virtual machines etc. which are provided by third parties over the internet like AWS, Azure, Google. Public cloud services are very popular due to various advantages like high scalability, less costs, high availability, high security, enhanced collaboration, early to market etc. Cloud providers are also fiercely competing, new services are launched almost every month.

While designing a cloud application, organizations can either opt for Cloud Native Architecture or Cloud Agnostic Architecture. Cloud-native architecture involves services and components tied to the Cloud Service Provider (CSP) itself. However, a cloud native approach will provide faster development and utilize all the benefits of public cloud services. Cloud Agnostic Architecture builds using open source tools and standards which can be moved to/from any on-premises infrastructure or any public cloud platform.

A major Solution provider from North America for the industrial and military sector wanted to develop an easy to use, efficient, and effective Maintenance instruction rendering solution, so that the maintenance personnel are prompted in real-time the steps to follow as well as how to perform complex tasks correctly without interruptions. They were looking for an application platform which can be deployed on-premise for some of their customers due to security/process/government constraints and cloud based solution for regular customers.

Hence, Gadgeon designed a Cloud Agnostic application where we utilize the maximum benefits of the cloud services.

What is meant by Cloud Agnostic?

Cloud Agnostic Application are the applications which can be moved seamlessly between any public cloud platform or on-premise infrastructure quite easily. Cloud Agnostic design is getting popular due to the below key advantage -

- No Cloud Lock-in - Limitless portability between various cloud platforms. This assured a consistent and standard performance whatever platform the application is deployed on.
- Easy On Premise deployment – There is no need to re-architecture the same solution for on premise deployment, with minor changes in some of the base components it can be easily ported to on-premise environment that meets critical infrastructure regulations and security constraints set by governments or to remote areas like mines where internet access is limited.

While designing applications as Cloud Agnostic sounds great, however the implementation will be more complex and the application will be able to use services that are offered by all the major public cloud providers.

How do we architect the application?

There are 2 two approaches to be cloud agnostics.

1. Do not depend on any SaaS services of cloud providers
2. Cautiously select the services while architecting the platform.

We decided to pursue the second option since this will combine the benefits of public cloud services and the flexibility of being cloud agnostic. We have architected the Maintenance instruction rendering solution by considering the below aspects

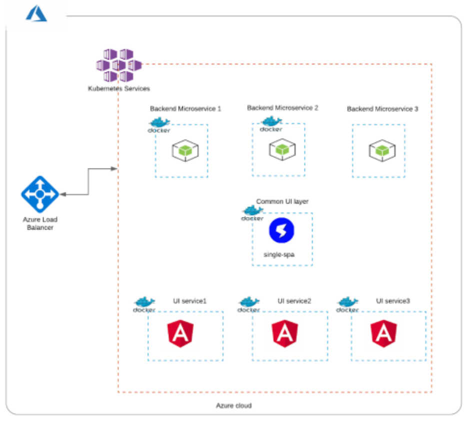
Microservices & Micro Frontend architecture

Microservice-based architecture is a methodology in which complex applications are divided into smaller units, each serving a specific business function and interacting with each other.

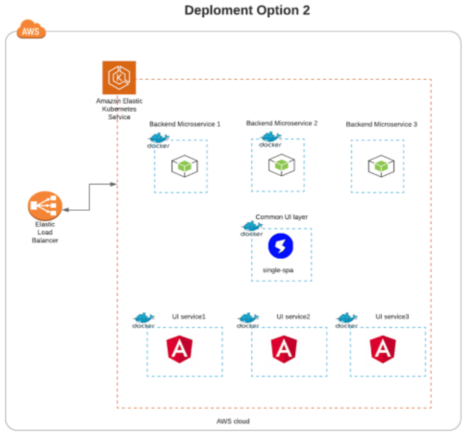
Microservices characteristics includes: It offers a minimal and complete set of features, It is portable and platform independent, It uses standard data formats etc.

While the backend is designed using Microservices architecture based on node.js, java/springboot etc, the frontend UI components is based on+ micro frontend (using single-spa framework).This architecture methodology supports deploying the part of application into multiple environments; e.g., secure components can be deployed on premise, stateless services can be shared across multiple customers on public cloud etc.

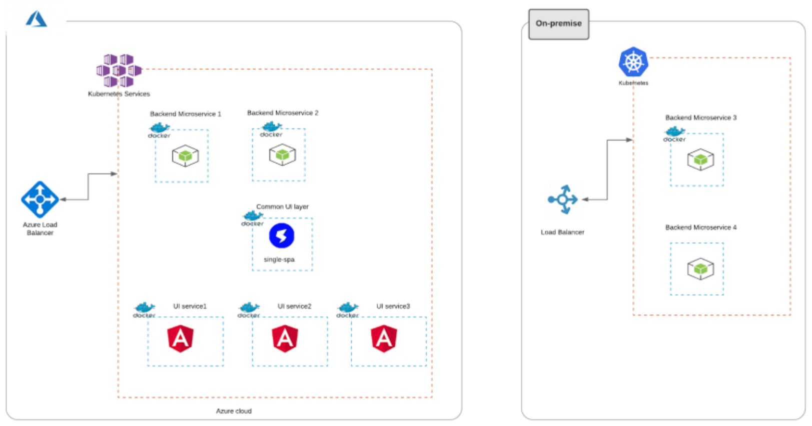
Deploment Option 1



Deploment Option 2



Deploment Option 3



Containerizing the application

Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they run. This decoupling allows container-based applications to be deployed easily and consistently, regardless of the target environment.

We had used Docker for build, package and deployment of our application and Kubernetes for container orchestration. The Dockers and Kubernetes are supported by most of the public cloud providers as well as open source implementation is also available..

- Elastic Kubernetes Service – Amazon Elastic Kubernetes Service (Amazon EKS) is a fully managed Kubernetes service. EKS is integrated with services other AWS services such as Amazon CloudWatch, Auto Scaling Groups, AWS Identity and Access Management (IAM), and Amazon Virtual Private Cloud (VPC), providing you a seamless experience to monitor, scale, and load-balance your applications.
- Azure Kubernetes Service - The fully managed Azure Kubernetes Service (AKS) makes deploying and managing containerized applications easy. Creation of azure Kubernetes service will create a set of resources which includes(VM, network group etc) and can easily integrate with Advanced identity and access management, Azure DevOps and Azure Monitor etc. to monitor, secure and scale the applications.
- Kubernetes - Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications. Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

Compare cloud platforms and choose the right services

Select services and frameworks which are crucial for cloud agnostic applications. The services selected should be portable to other deployment environments with minimum/zero changes. Utilizing the services offered by Cloud service providers helps to save time to market.

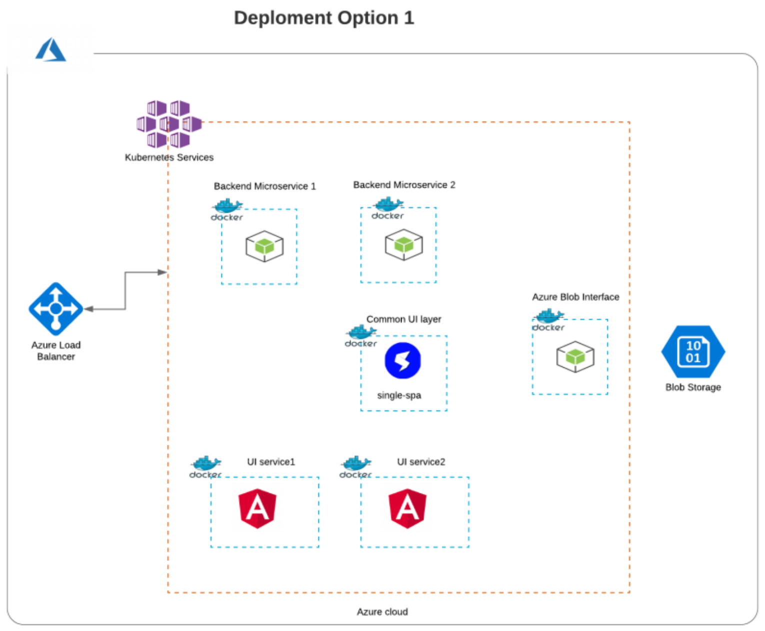
Database as service: Managing databases is hard,but cloud vendor's managed database services support to save time and money, and we can focus on the core business logic. However, if we didn't select a proper database service which will create vendor lock-in.

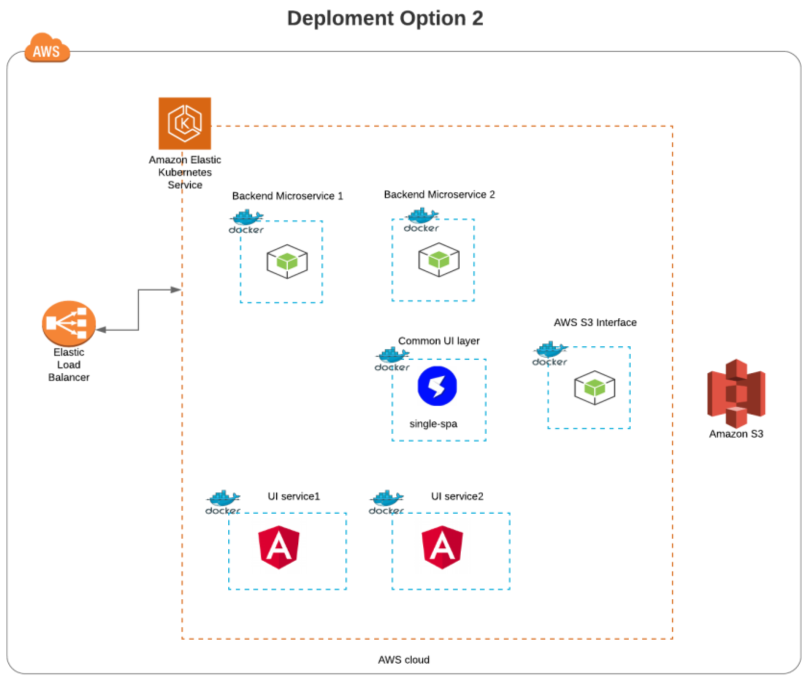
If the application requires a Relational database, we can select a database as a service from the cloud providers such as AWS Aurora or Azure Database. This support switches between various cloud providers or even to the on-premise infrastructure since the underlying database engine is open source/commercial database like MySQL, postgresql etc. Similarly, we can opt for Azure Cosmos DB's mongodb interface while going for NoSQL database requirements. Popular cloud vendors are also providing database migration services which support moving the database from one infrastructure to another, e.g. AWS Database Migration Service.

Load Balancer and other security services: Utilizing cloud vendors load balancers and security services support the application to be highly scalable, available and secure. The public cloud providers offer many inbuilt security features. Since these features are outside the application logic, we can port the application from one cloud vendor to another without any friction.

Interface Layers: For cloud agnostic application, we can choose services which are available in multiple cloud vendors. Even though multiple vendors provide similar services, the interface to these services may differ slightly. Hence, in order to reduce the friction to port between various services the application design can have separate service layers.

For example, storing the file objects all the major cloud host providers are providing various storage services like AWS S3, Azure Blob, Google cloud storage etc. These public services are cost effective and have optimized performance. Hence, we utilize these services and use multiple interface layers for different cloud providers.





Automate all the aspects

DevOps is a crucial part for any cloud agnostic application development. For the Maintenance instruction rendering solution, we started the design with an automation first mentality. Infrastructure creation, Deployment, CI/CD pipeline for easy integration and automation testing are part of this application. The services/applications used for the automation process also need careful selection process for being cloud agnostic.

HashiCorp Terraform helps define infrastructure as code for the application. Terraform supports spinning up the infrastructure and deploying the application on top of it. For Cloud agnostic devOps, there will be multiple terraform abstraction layers that will interact whichever cloud platform we need to spin up.

Continuous Integration and Deployment pipelines as well as Automation Testing can also be cloud agnostics by using open source tools like Jenkins, Robot framework, Selenium etc.

Conclusion

Cloud agnostic architecture had many advantages. However, we need to take detailed analysis in the initial stage of an application design and carefully select the services and system design principles to develop cloud agnostic applications.