Varsity College

# ICE TASK 1 – CLDV 6212

CREATED BY TemplateLAB

ST10075585

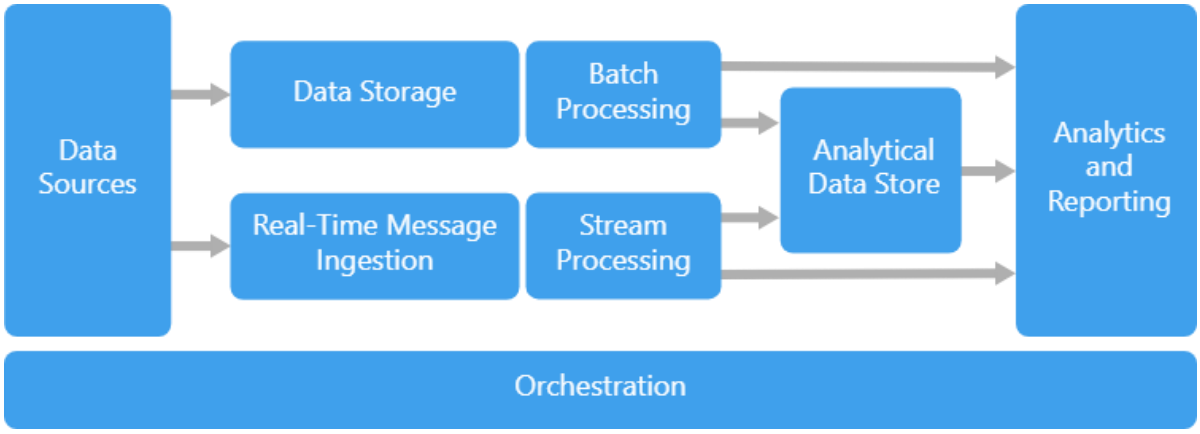
ST10075585

NSIKELELO KUMALO

ICE TASK 1- CLDV 6212

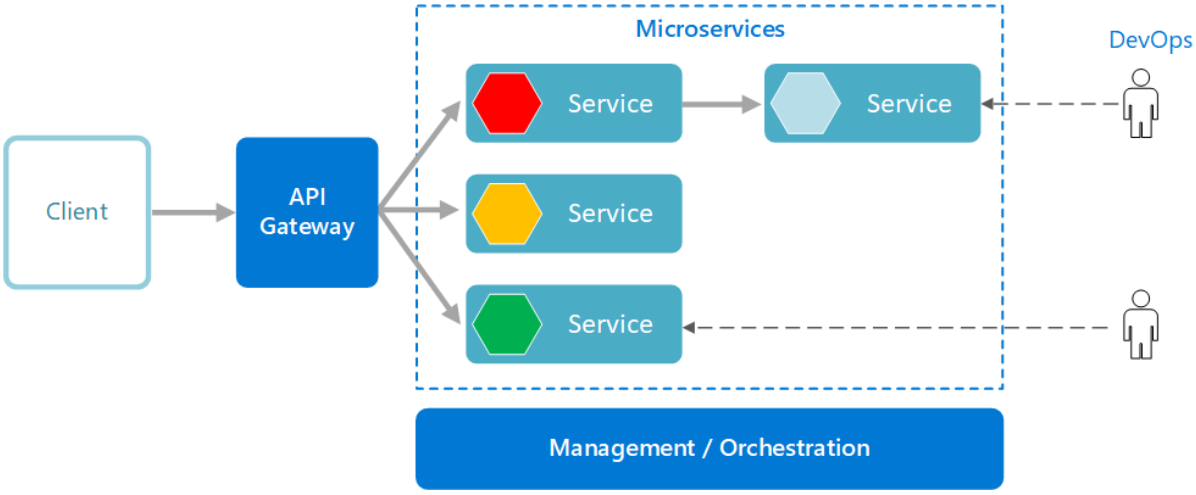
**What is an architecture style:**

An architecture style refers to a group of architectures that exhibit specific common characteristics. For instance, N-tier is a prevalent architecture style, and recently, microservice architectures have gained popularity. These styles do not mandate the use of specific technologies, but certain technologies align well with particular architectures. For example, containers are a natural and suitable choice for microservices.



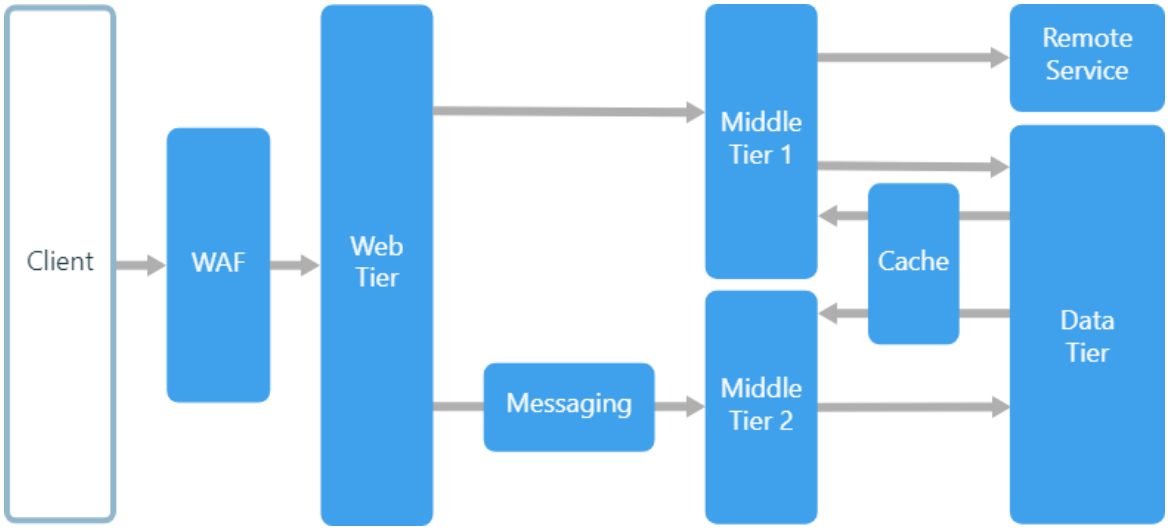

**Big Data:** Big Data is an architecture style designed for handling large datasets in specific scenarios. It involves breaking down extensive datasets into smaller parts and processing them simultaneously for analysis and reporting. On the other hand, Big Compute, also known as high-performance computing (HPC), involves performing parallel computations using a vast number of cores. It is commonly used in domains like simulations, modelling, and 3-D rendering.

In the context of a large e-commerce platform, Big Data architecture is used to analyse customer behaviour and purchase patterns by processing massive amounts of data through parallel processing and applying various algorithms. Simultaneously, the platform leverages Big Compute (HPC) to manage real-time inventory efficiently, handling numerous product updates and shipping calculations in parallel. The combination of Big Data and Big Compute enables the platform to perform large-scale data analysis while maintaining real-time inventory management, enhancing overall performance and user satisfaction.




**Microservices:** Microservices represent an architectural approach where an application is structured as a collection of individual services. These services are designed to focus on specific business capabilities and can be developed and tested independently, allowing them to be written in different programming languages and deployed separately. In the context of an e-commerce application developed using microservices, each microservice would handle a distinct function such as managing the shopping cart, facilitating search functionality, or handling customer reviews. These microservices can be implemented in diverse languages, hosted on separate infrastructures, and maintained by different teams.

Communication between these microservices is facilitated through a lightweight protocol. In contrast, the traditional 3-tier architecture employs the Model View Controller (MVC) framework. For microservices, supporting frameworks like Sidecar, Ambassador, and Adapter come into play, enabling the efficient implementation and management of this architecture.




**N-tier:** N-tier architecture is a conventional approach commonly used in enterprise applications. It manages dependencies by dividing the application into distinct layers responsible for specific functions, such as presentation, business logic, and data access. Each layer is only allowed to communicate with the layers beneath it. However, this horizontal division can pose challenges as it becomes difficult to

introduce changes to one part of the application without affecting the rest of the system. Consequently, frequent updates become problematic, leading to limitations in the speed of adding new features.

An example of N-tier architecture can be seen in a web-based e-commerce application. The presentation layer deals with the user interface, where customers browse products and place orders. The business logic layer handles the processing of orders, inventory management, and pricing calculations. The data access layer interacts with the database, fetching and storing product information, order details, and customer data. Each layer operates independently, with the presentation layer relying on the business logic layer, which in turn depends on the data access layer for retrieving and storing data. However, changes to the database structure or business rules may necessitate modifications in both the business logic and presentation layers, making the application update process complex and time-consuming. Despite its challenges, N-tier architecture is still preferred for migrating existing applications with established layered structures, especially in infrastructure as a service (IaaS) environments or when utilizing a mix of IaaS and managed services.

**What are Technology choices:**

Technology choices involve picking particular tools, software, programming languages, frameworks, and platforms to construct and execute a solution or system. These choices are critical as they shape the architecture and features of an application and impact its ability to scale, perform, and stay secure, ultimately influencing its overall success.

**Technology Choice:**

Messaging facilitates communication between various components or services within a distributed application. It allows asynchronous communication, enabling components to exchange messages independently without direct connections. By decoupling components, messaging ensures their independent functioning and efficient scalability.

Messaging can be implemented using different approaches:

a. Message Queues: In this approach, messages are sent by applications and stored in queues until received by the intended recipients. This ensures orderly message delivery.

b. Publish-Subscribe (Pub/Sub): Publishers disseminate messages to specific topics, and multiple subscribers can receive messages related to those topics.

c. Event Bus: An event bus acts as a central hub for handling events and distributing them to relevant subscribers.

Well-known messaging technologies include Apache Kafka, RabbitMQ, and Azure Service Bus.

**Application Architecture Pattern: Reference Architecture:**

A Reference Architecture serves as a pre-defined blueprint or model that offers guidelines and best practices for constructing a particular type of application or system. It presents a recommended structure, design principles, and technology selections that align with the organization's goals and industry standards. The purpose of reference architectures is to accelerate development, mitigate risks, and foster uniformity across various projects.

As an illustration, a cloud-native reference architecture could outline the adoption of microservices, containerization, Kubernetes for orchestration, and cloud-specific services such as AWS Lambda or Azure Functions for serverless computing.


**Azure Well-Architected Framework: Security:**

The Azure Well-Architected Framework focuses on establishing guidelines and best practices provided by Microsoft to create secure, dependable, high-performance, and efficient applications and workloads on the Microsoft Azure cloud platform.

In terms of security, the framework highlights the following principles:

a. Identity and Access Management: Ensuring the presence of proper authentication and authorization methods to regulate access to Azure resources. This involves utilizing Azure Active Directory, RBAC (Role-Based Access Control), and Multi-Factor Authentication (MFA).

b. Network Security: Implementing measures such as network segmentation, virtual networks, Network Security Groups (NSGs), and Azure Firewall to control and monitor network traffic.

c. Data Security: Utilizing encryption both during data transmission (e.g., TLS/SSL) and while data is at rest (e.g., Azure Disk Encryption, Azure Storage Service Encryption) to safeguard sensitive information.

d. Threat Protection: Making use of Azure Security Centre and Azure Sentinel to identify and respond to threats through monitoring and incident response.

e. Compliance and Governance: Adhering to regulatory requirements and industry standards by employing compliance services like Azure Policy, Azure Blueprints, and audit logging.

f. Disaster Recovery and Business Continuity: Planning for data backups, replication, and recovery mechanisms to ensure business continuity in the event of disruptions.

g. Secure DevOps: Implementing security practices throughout the software development lifecycle, which includes activities like vulnerability scanning, code analysis, and continuous security testing.

**Linking the Principles Together:**

The principles discussed above are interconnected and should be considered collectively when designing and building an application.

For instance, in the context of a microservices-based application hosted on Azure:

- The Big Data architecture may come into play if the application deals with large volumes of data that need to be processed and analysed in real-time.
- The N-tier architecture can be utilized to organize the microservices into different tiers, such as the presentation tier, application tier, and data tier, enabling scalability and maintainability.
- Messaging technology may be used to allow communication and data exchange between the microservices in an asynchronous manner.
- The reference architecture can provide guidelines on how to design and implement each microservice securely and consistently, following the Azure Well-Architected Framework's security principles.

By applying these principles in a cohesive manner, developers and architects can ensure the application is secure, scalable, and adheres to industry best practices, resulting in a robust and well-architected solution on the Azure platform.