Interfaces funcionales, despliegue de microservicios en Cloud

BOOTCAMP JAVA-MICROSERVICIOS | Perú 2022

Por: Froilan Ichpas Buendia

fichpasb@emeal.nttdata.com

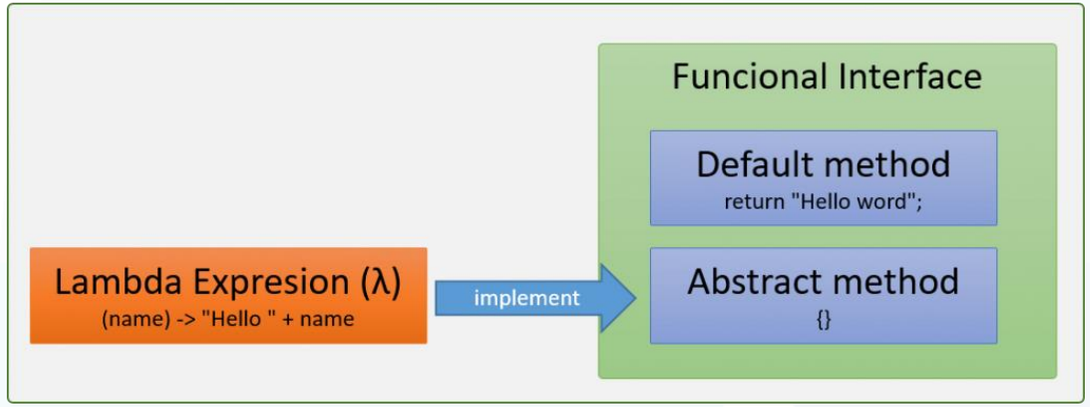
AGENDA

INTERFACES FUNCIONALES

DESPLIEGUE DE MICROSERVICIOS EN CLOUD

Interfaces funcionales

Interfaces funcionales



- ✓ *Las **interfaces funcionales** son interfaces que tienen un único método abstracto a implementar, pudiendo implementar uno o varios métodos default o static. Esto significa que cada interfaz creada que respeta esta premisa se convierte automáticamente en una interfaz funcional.*

https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html

Interfaces Funcionales

```java
@FunctionalInterface
public interface IGreeting {
    void SayHello();
}
```

```java
import pe.com.java.bootcamp.interfaces.IGreeting;

public class Ejercicios07 {

    public static void main(String[] args) {
        IGreeting greeting = () -> {System.out.println("Hello World");};
        greeting.SayHello();
    }
}
```

Interfaces Funcionales (Parámetros)

```java
@FunctionalInterface
public interface IGreetingV2 {
    void SayHello(String usuario);
}
```

```java
6
7       public static void main(String[] args) {
8
9           IGreetingV2 greeting = usuario -> {
10              if (usuario == "Bootcamp")
11                  System.out.println("Hello " + usuario);
12              else
13                  System.out.println("Hello World " + usuario);
14          };
15          greeting.SayHello("Bootcamp");
16      }
17  }
```

Interfaces Funcionales (Retornar Valores)

```java
@FunctionalInterface
public interface IAddition {
    int add(int a , int b);
}
```

```java
import pe.com.java.bootcamp.interfaces.IAddition;

public class Ejercicios10 {

    public static void main(String[] args) {

        IAddition addition =  (a, b) -> {
            int c = a * 2;
            return c + b;
        };
        int resultado = addition.add(4, 5);
        System.out.println(resultado);

    }
}
```

Interfaces Funcionales Genéricas

- Supplier
- Consumer y BiConsumer
- Predicate y BiPredicate
- Function y Bifunction
- UnaryOperator y BinaryOperator

Supplier

```
Supplier<LocalDateTime> s = () -> LocalDateTime.now();
LocalDateTime time = s.get();

System.out.println(time);
```

*Contiene el método **get** que no recibe argumentos y produce un valor de tipo <T>. A menudo se usa para crear un objeto colección en donde se colocan los resultados de la operación de un flujo.*

Consumer y BiConsumer

```java
// Consumer to display a number
Consumer<Integer> display = a -> System.out.println(a);
display.accept(10);

// BiConsumer to add two numbers
BiConsumer<Integer, Integer> addTwo = (x, y) -> System.out.println(x + y);
addTwo.accept(1, 2);
```

Contiene el método **accept** que recibe como argumento <T> y devuelve void. Realiza una tarea con su argumento <T>,como mostrar el objeto en pantalla, invocar a un método del objeto, etc

Predicate y BiPredicate

```
 // Creating predicate
Predicate<Integer> lesserthan = i -> (i < 18);

// Calling Predicate method
System.out.println(lesserthan.test(10));

// Creating biPredicate
BiPredicate<String, Integer> filter = (x, y) -> {
    return x.length() == y;
};

// Calling BiPredicate method
boolean result = filter.test("mkyong", 6);
System.out.println(result);  // true
```

*Contiene el método **test** que recibe un argumento T y devuelve un boolean.*

UnaryOperator y BinaryOperator

```java
public static void main(String args[]) {

    // Creating UnaryOperatos
    UnaryOperator<Integer> func2 = x -> x * 2;

    // Calling  method
    Integer result2 = func2.apply(2);
    System.out.println(result2);

    // Creating BinaryOperator
    BinaryOperator<Integer> func3 = (x1, x2) -> x1 + x2;

    // Calling  method
    Integer result3 = func3.apply(2, 3);
    System.out.println(result3);

}
```

NTT DATA

Despliegue de Microservicios en Cloud

Que es la nube:

La nube son instalaciones en las cuales cada una tiene de forma independiente energía eléctrica, refrigeración y seguridad,

son llamados Centro de Datos. Dentro de ellos se encuentran cientos de equipos conectados a Internet para consumir los

servicios ofrecidos en la Nube.

Los Centros de Datos se encuentran distribuidos a lo largo del mundo.

Azure es el servicio Cloud que tiene mas DataCenters desplegados a nivel mundial.

Azure (Microsoft) busca que sus DataCenters sean 100% sustentables utilizando Energias Limpias

¿Para qué o Por qué?

Datacenters

Instalaciones que cuentan de manera independiente con energía eléctrica, refrigeración y seguridad

Distribuidos a lo largo del mundo

AZURE es quien tiene mas datacenters distribuidos a lo largo del mundo

- Con energía sustentables

Servicios

- Cómputo
- Servidores
- Almacenamiento y bases de datos
- Redes
- AI
- Software y +

Todo lo que puedas hacer en un pc se puede hacer en la nube pero más: **barato, ágil y seguro**

Ventajas

Modelo de consumo: Servicios

No inviertes en infraestructura,
Reduce costos operativos
Escala según necesidades

Escalabilidad

Vertical: en RAM/CPU a una maquina virtual

Horizontal: aumentando en instancias a recursos

Elasticidad

Siempre tendrán los recursos necesarios

Confiabilidad y alta disponibilidad

Recuperación anti desastres

Agilidad

Distribución geográfica

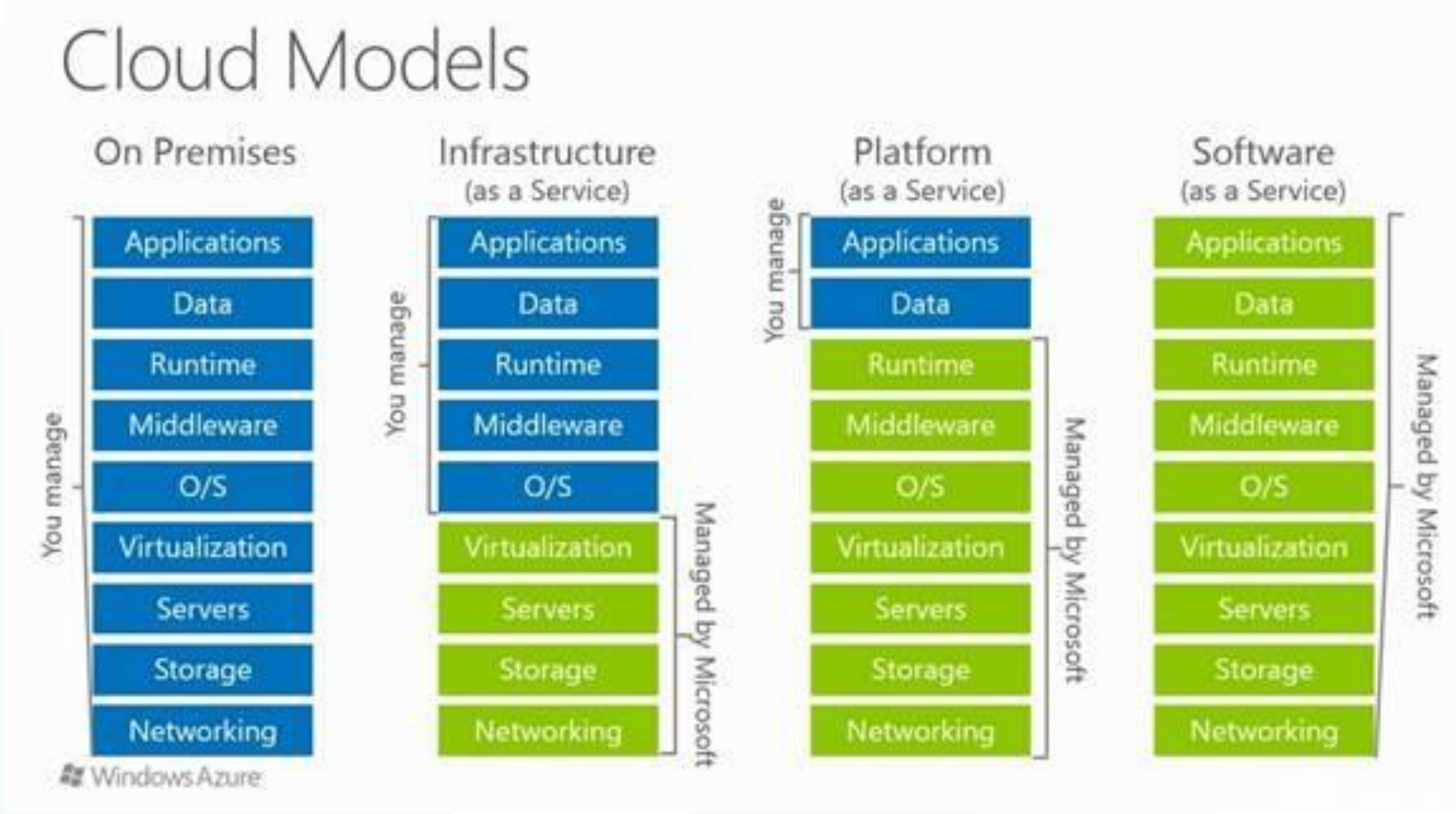
CapEx

Gastos de Capital
Inversión en infraestructura física, deducible a largo plazo

OpEx

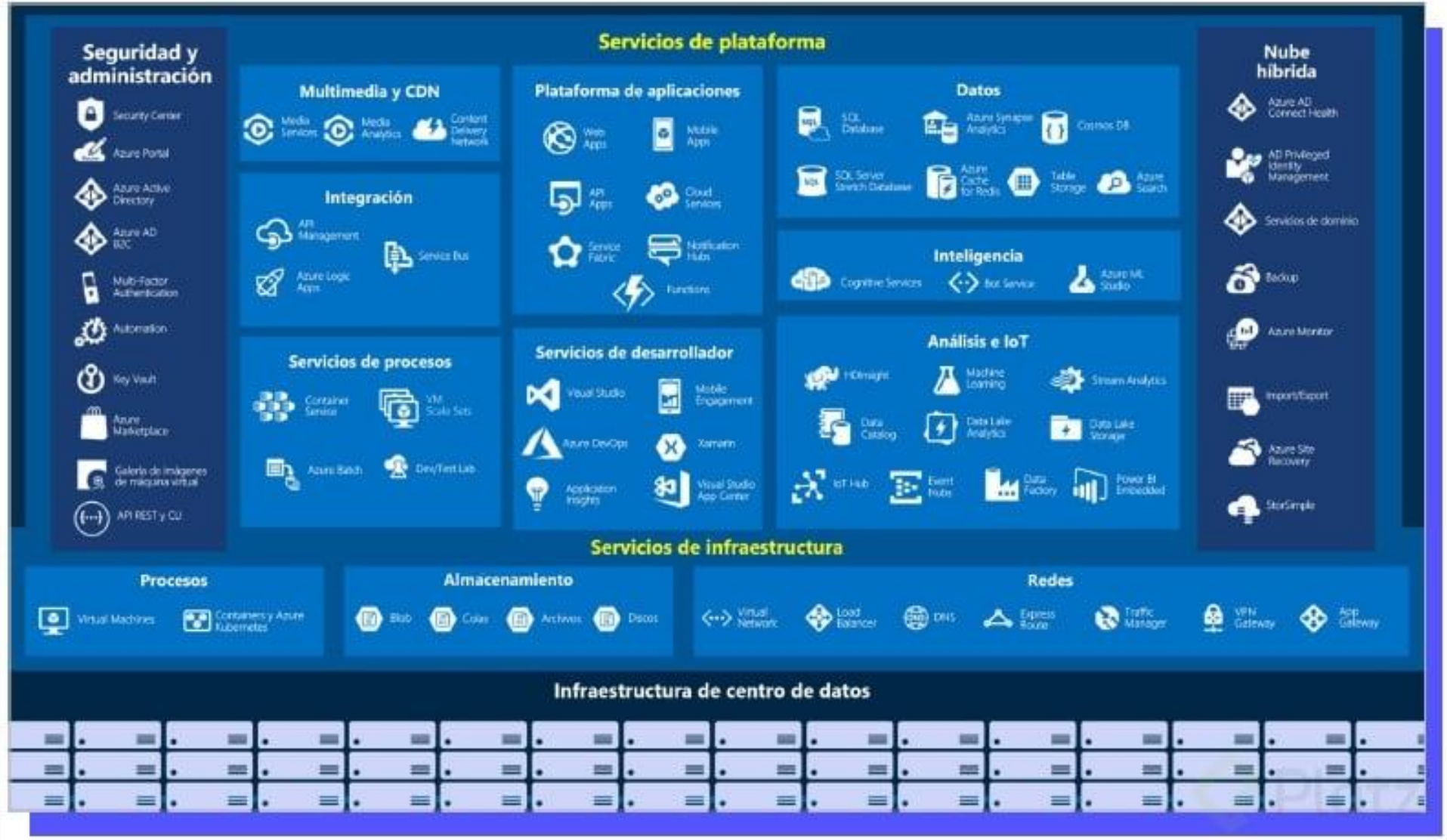
Gastos Operativos
Inversión en servicios o productos facturados al momento

¿Modelo de nube?

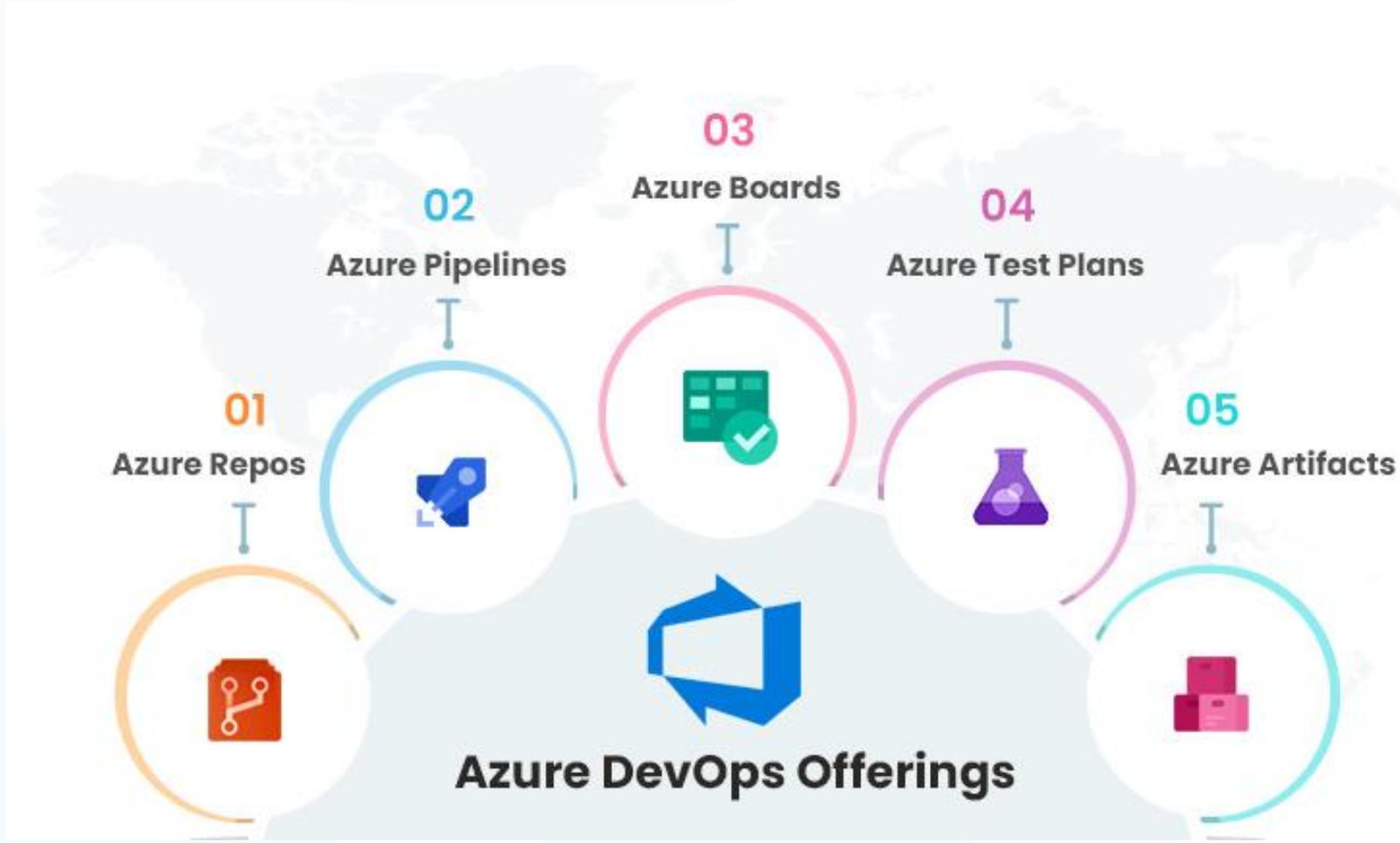


¿Que es Azure?

Microsoft Azure es un servicio de computación en la nube creado por Microsoft para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos



Azure devOps



Links de Azure:

Soluciones de Azure | Microsoft Azure

https://azure.microsoft.com/es-es/solutions/

Microsoft Azure

https://portal.azure.com/

Examinar todo - Learn | Microsoft Docs

https://docs.microsoft.com/es-es/learn/browse/?products=azure&resource_type=learning%20path

Cuenta gratuita

https://azure.microsoft.com/es-es/free/


NTT DATA
Gracias