ARQUITECTURAS DE NUBE PARA Internet of Things

REDTPI4.0

CYTED

PROGRAMA IBEROAMERICANO DE CIENCIA
Y TECNOLOGÍA PARA EL DESARROLLO

Agenda

Sesión 1: 2 horas sincrónicas + 4 horas independientes

1. Introducción a IoT y arquitecturas
2. Introducción y principios de Nube
3. Manejo del laboratorio
4. Ejercicios de introducción (trabajo independiente)

Sesión 2: 2 horas sincrónicas + 4 horas independientes

1. Arquitecturas de Nube – Generalidades
2. El Agente/Orquestador/Broker
3. Sistemas de almacenamiento
4. Sistemas de ETL
5. Sistemas de Toma de decisiones
6. Sistemas de Visualización
7. Ejercicios de Agentes y escritura de datos (trabajo independiente)

Sesión 3: 2 horas sincrónicas + 4 horas independiente

1. Sistemas de Almacenamiento – Modos de almacenamiento/arquitecturas con ventajas y desventajas

Sesión 4: 2 horas sincrónicas + 4 horas independiente

1. Sistemas de ETL
2. Visualización de datos
3. Taller de ETL y visualización de datos – Ejercicio PM2.5 (trabajo independiente)

Sesión 5: 2 horas sincrónicas + 4 horas independiente

1. Sistemas de toma de decisiones
2. Interfaces de usuario y desarrollo de apps
3. Taller de toma de decisiones y desarrollo de apps (preventivos y reactivos)

Sesión 6: 2 horas sincrónicas + 4 horas independiente

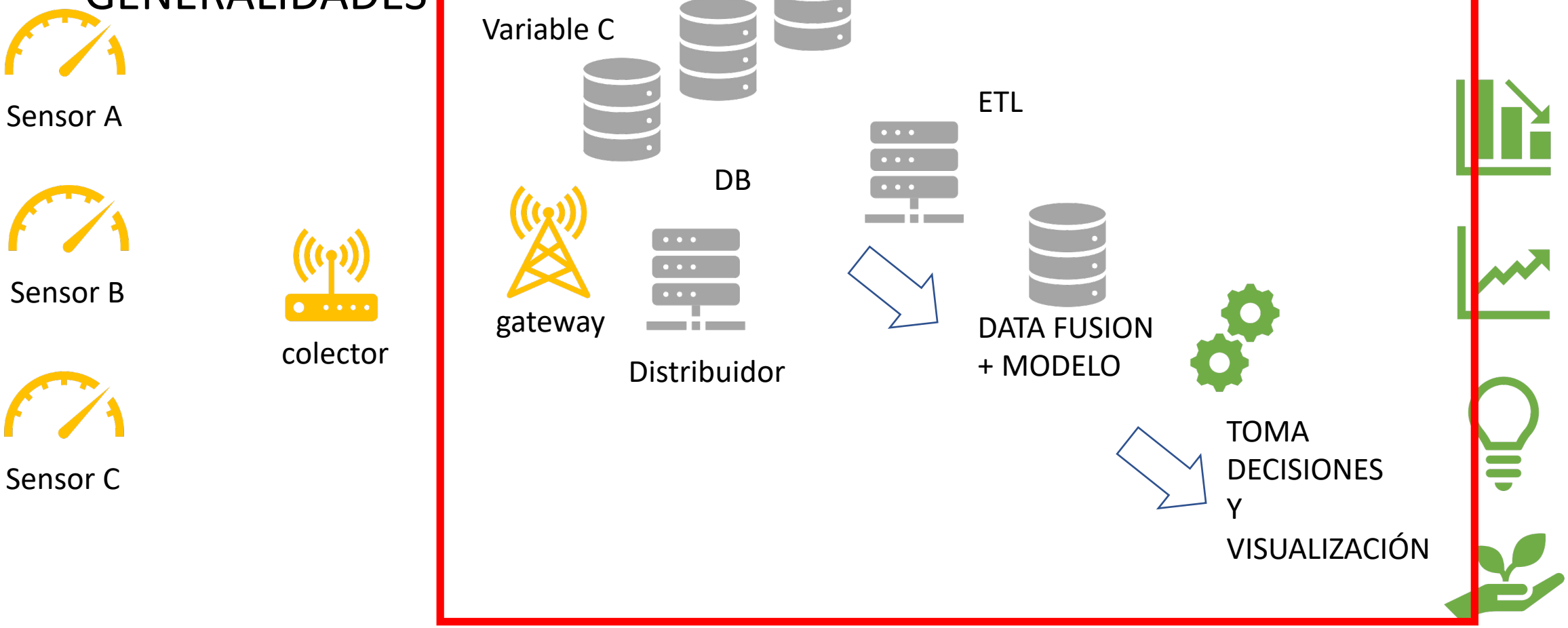
1. Integración de la arquitectura con FiWARE

Sesión 7: 8 horas presenciales

1. Montaje del proyecto presencial
2. Arquitecturas de alta disponibilidad en Nube
3. Ejercicios prácticos de montaje con sensores vía WiFi
4. Dimensionamiento de procesamiento y aspectos financieros de soluciones de Nube

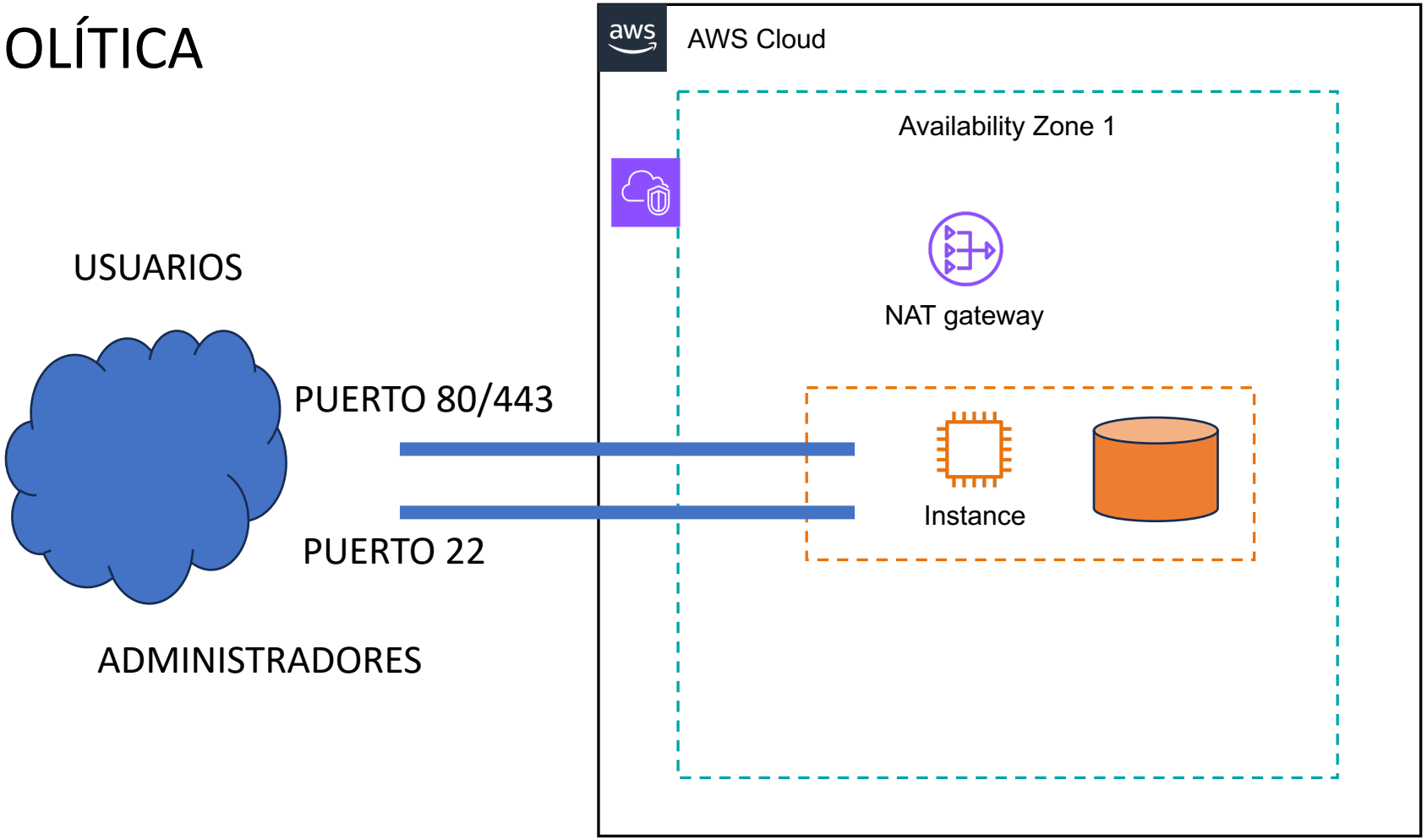
SISTEMAS DE ALMACENAMIENTO

GENERALIDADES



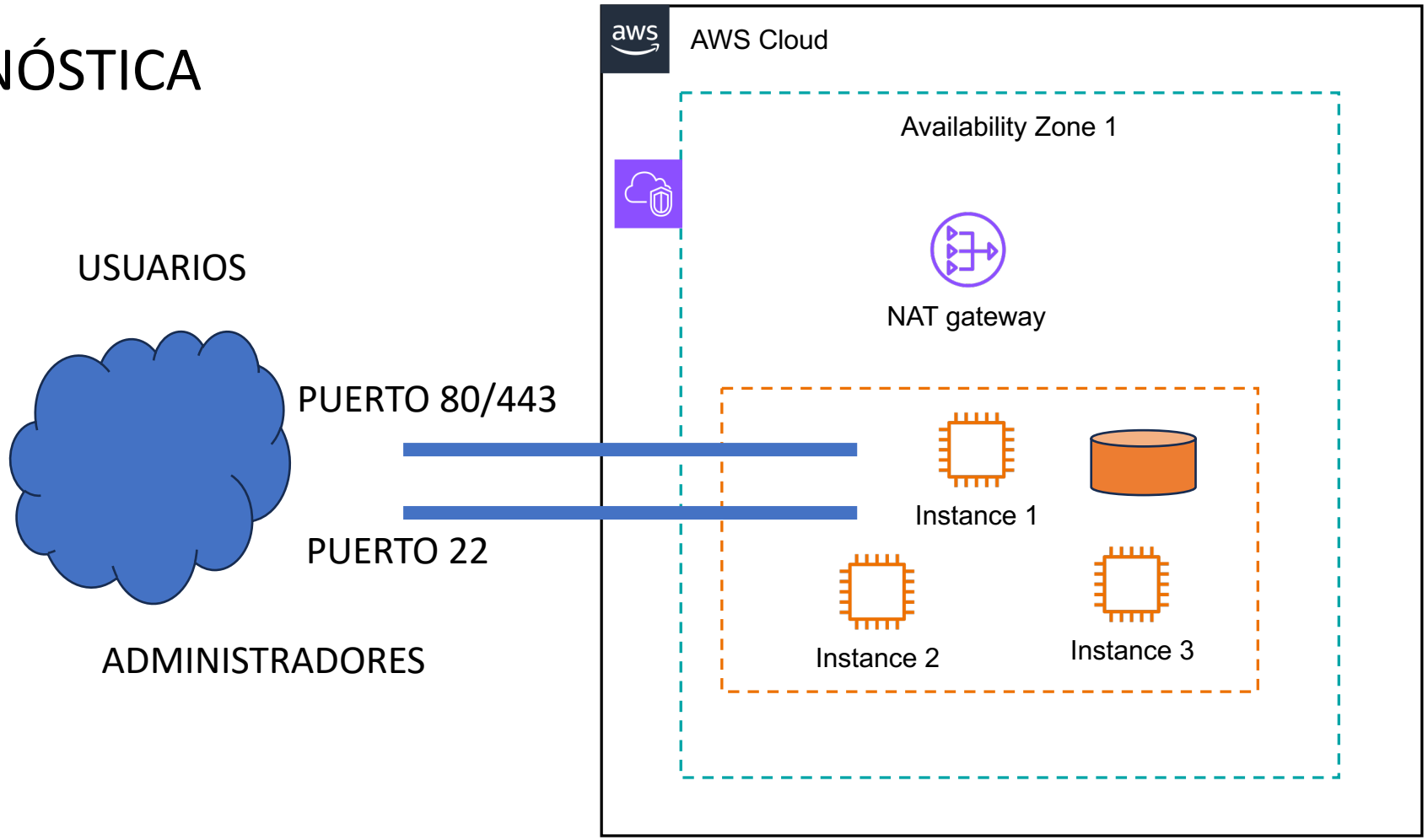
ARQUITECTURAS DE NUBE

MONOLÍTICA



ARQUITECTURAS DE NUBE

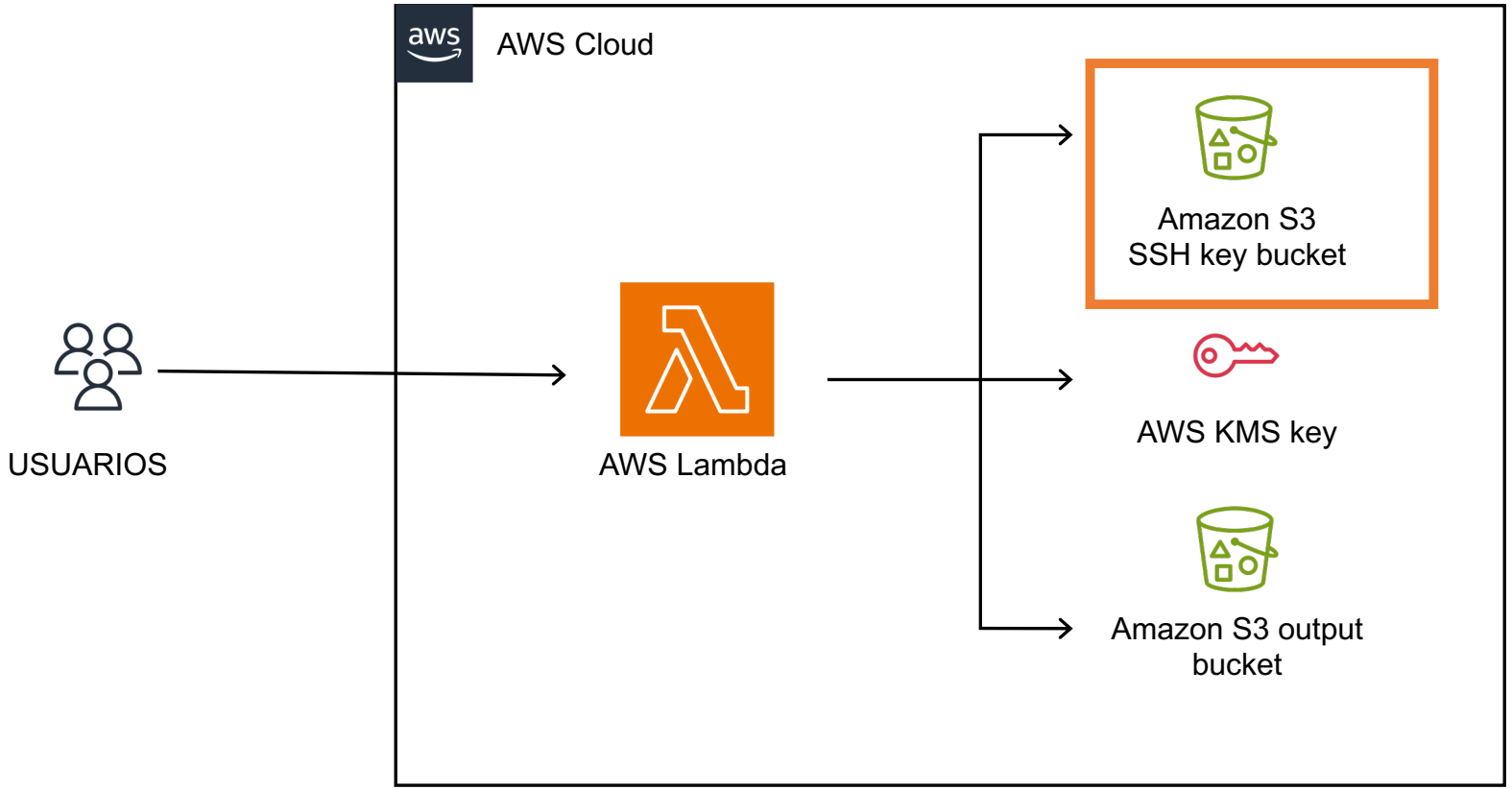
AGNÓSTICA



TODOS LOS SERVICIOS DE FORMA DISCRETA

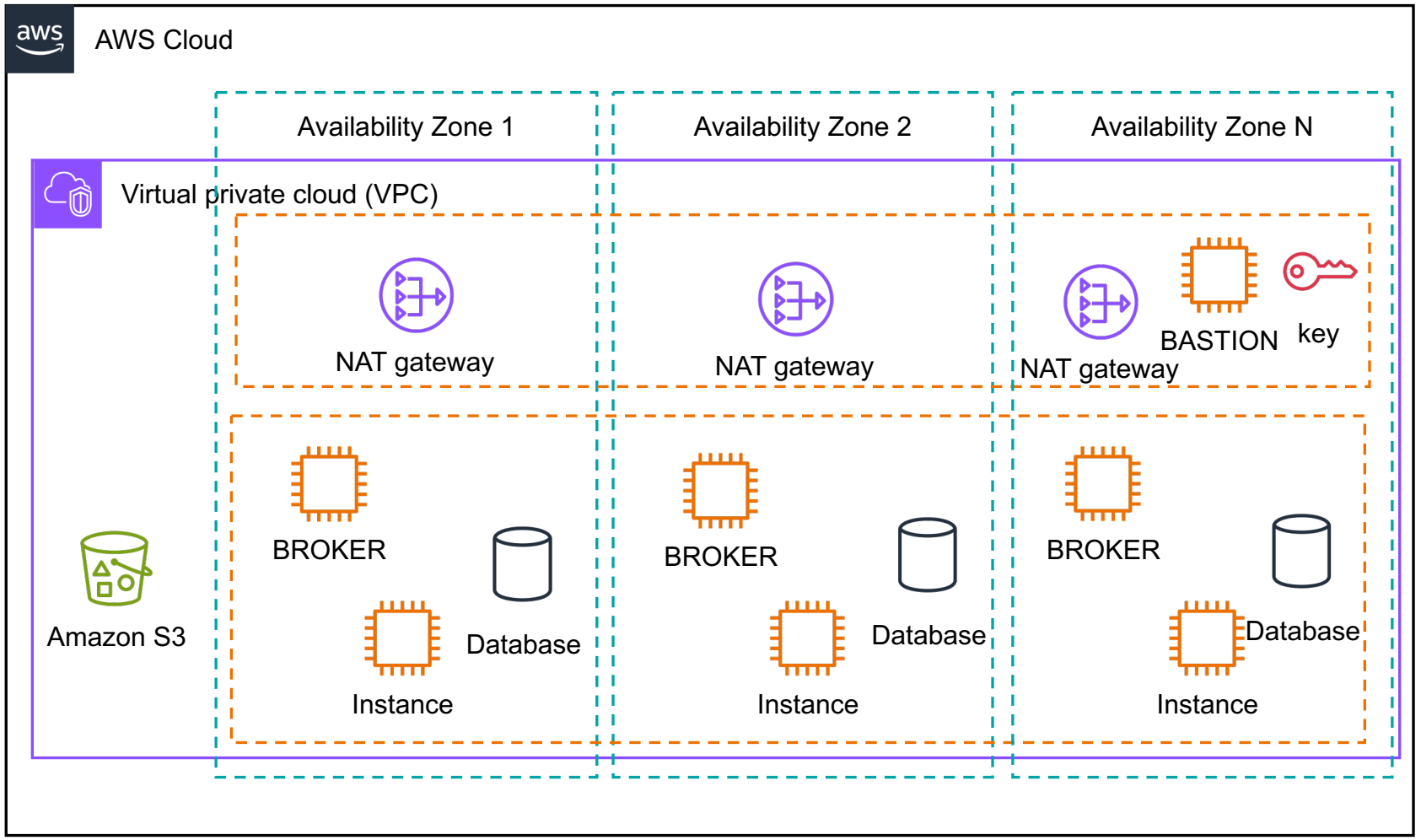
ARQUITECTURAS DE NUBE

DEPENDIENTE DE LA NUBE



ARQUITECTURAS DE NUBE

LA COMPLETA



SISTEMAS DE ALMACENAMIENTO

GENERALIDADES

En un archivo

En un motor

En una entidad de almacenamiento de nube

SISTEMAS DE ALMACENAMIENTO

- LOCAL — usar una base de datos local en la misma máquina
- MOTOR LOCAL — usar un motor de base de datos en la misma máquina de carga de datos
- MOTOR EXTERNO — usar un motor de base de datos en una máquina remota con acceso remoto

SISTEMAS DE ALMACENAMIENTO

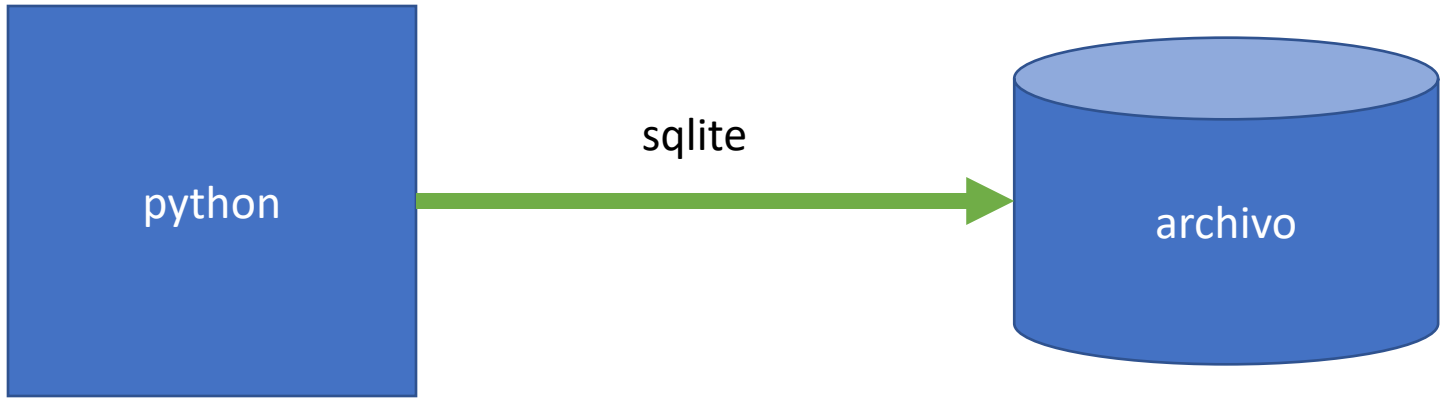
- LOCAL — usar una base de datos local en la misma máquina
- MOTOR LOCAL — usar un motor de base de datos en la misma máquina de carga de datos
- MOTOR EXTERNO — usar un motor de base de datos en una máquina remota con acceso remoto

SISTEMAS DE ALMACENAMIENTO

- LOCAL

```
import sqlite3

referenciabasedatos = 'Basedatos.db'
```

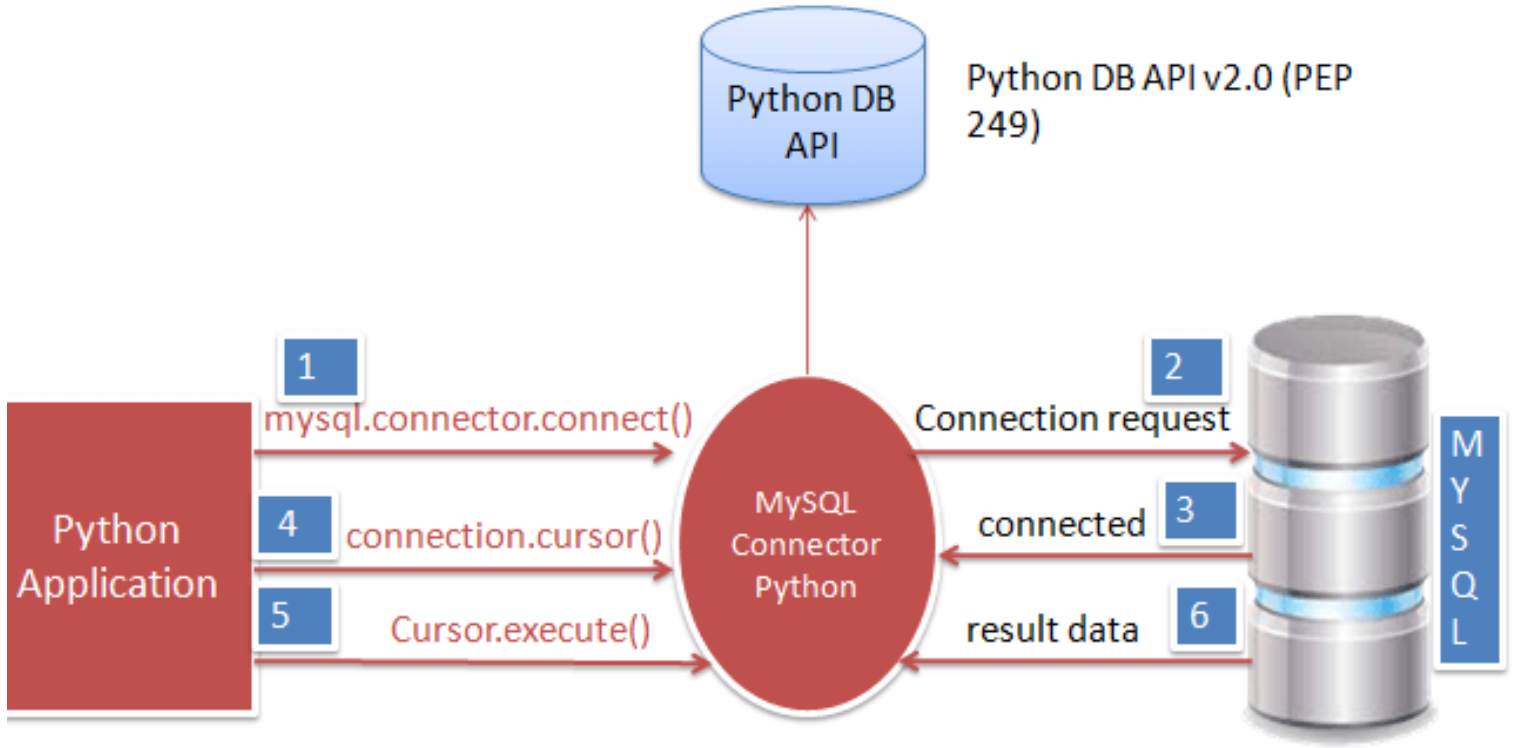



```
con = sqlite3.connect(referenciabasedatos)

cur = con.cursor()
cur.execute("INSERT INTO sensores VALUES("+id+","+temp");")
con.commit()
con.close()
```

SISTEMAS DE ALMACENAMIENTO

- MOTOR LOCAL
- MOTOR EXTERNO



SISTEMAS DE ALMACENAMIENTO

MOTOR LOCAL

1. Instale un servidor de base de datos (MySQL o MariaDB)

sudo apt-get install mariadb-server mariadb-client

sudo systemctl stop mariadb.service

sudo systemctl start mariadb.service

sudo systemctl enable mariadb.service

sudo mysql_secure_installation

2. Inicialice el motor

Cree una base de datos y una tabla

sudo mysql -u root -p

CREATE DATABASE prueba;

CREATE USER 'user'@'localhost' IDENTIFIED BY 'new_password_here';

GRANT ALL ON prueba.* TO 'user'@'localhost' IDENTIFIED BY 'user_password_here' WITH GRANT OPTION;

FLUSH PRIVILEGES;

EXIT;

SISTEMAS DE ALMACENAMIENTO

MOTOR LOCAL y REMOTO

3. Conéctese usando Python

pip install mysql-connector-python

import mysql.connector

```
import mysql.connector
from mysql.connector import Error

try:
   connection = mysql.connector.connect(host='localhost',
                                        database='Electronics',
                                        user='pynative',
                                        password='pynative@#29')
   if connection.is_connected():
      db_Info = connection.get_server_info()
      print("Connected to MySQL Server version ", db_Info)
      cursor = connection.cursor()
      cursor.execute("select database();")
      record = cursor.fetchone()
      print("You're connected to database: ", record)

except Error as e:
   print("Error while connecting to MySQL", e)
finally:
   if connection.is_connected():
      cursor.close()
      connection.close()
      print("MySQL connection is closed")
```

SISTEMAS DE ALMACENAMIENTO

MOTOR LOCAL y REMOTO

3. Cargue datos en una tabla desde Python

```
query = """
INSERT INTO tablapeliculas (title, release_year, genre,
collection_in_mil) VALUES ("Forrest Gump", 1994, "Drama", 330.2)
"""
with connection.cursor() as cursor:
    cursor.execute(query)
    connection.commit()
```

EJERCICIOS

- Usar Python para MONTAR:
  - 1. una arquitectura local escribiendo en un archivo con SQLITE – sustentar al profesor
  - 2. una arquitectura con una base de datos de motor local
  - 3. una arquitectura con una base de datos remota – use un RDS y un EC2
  - 4. una arquitectura con una base de datos en un contenedor

  Emplee el protocolo http para hacer un post y transformar esos datos para ser cargados en una base de datos local o remota con su sensor escribiendo en el sistema

- VER LOS EJEMPLOS
  - Distribuidor.py -> LOCAL
  - Diapositivas -> MOTOR LOCAL o EXTERNO