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

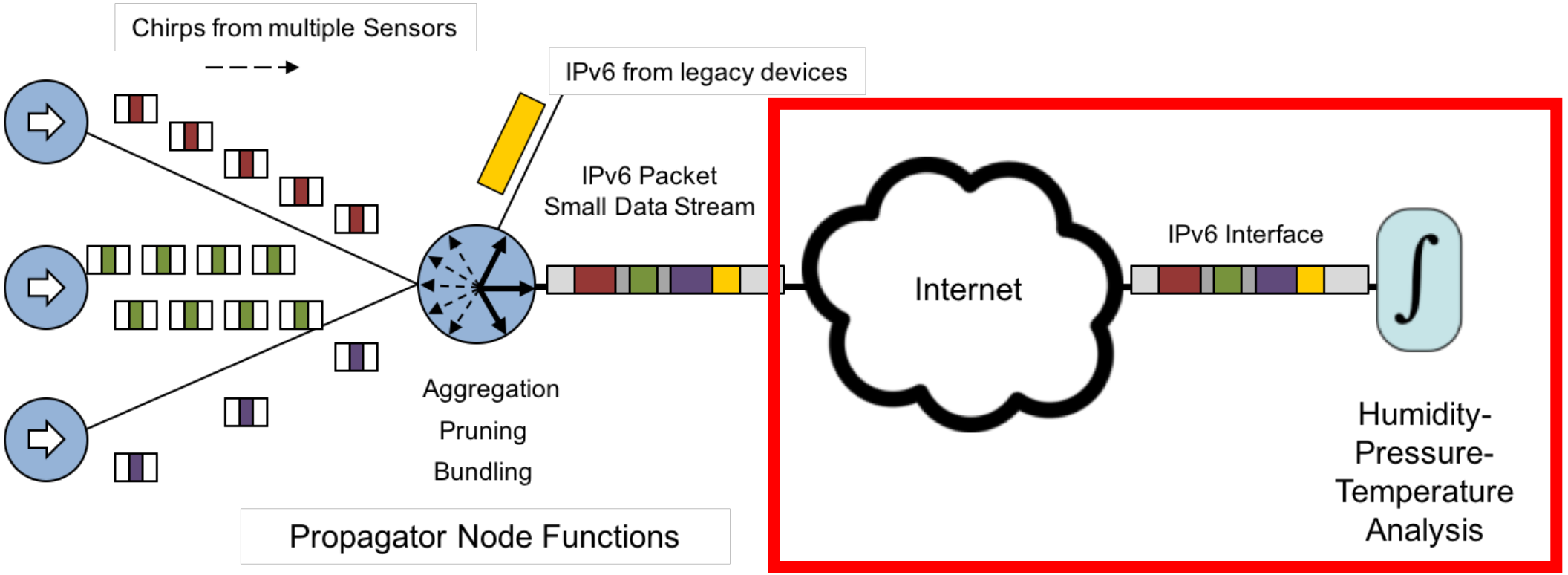
1. Integración de la arquitectura con FiWARE

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

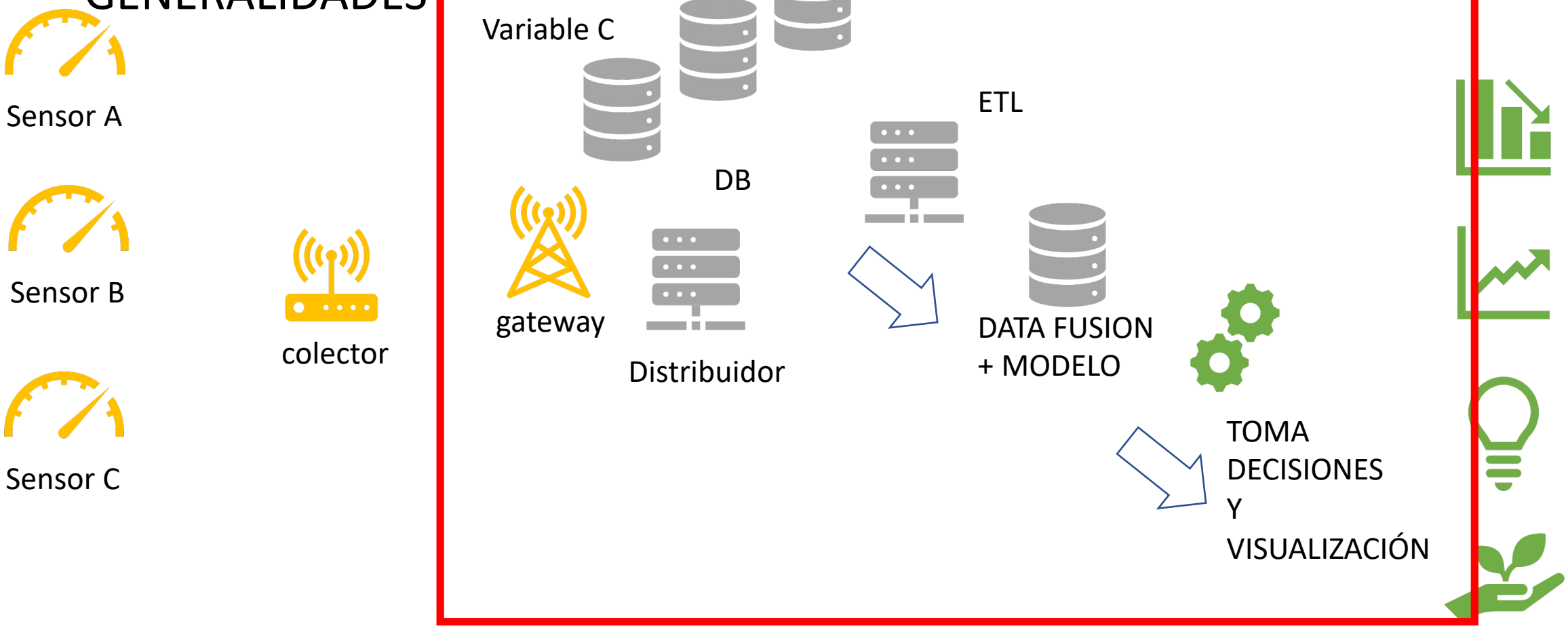
ARQUITECTURAS DE NUBE

GENERALIDADES



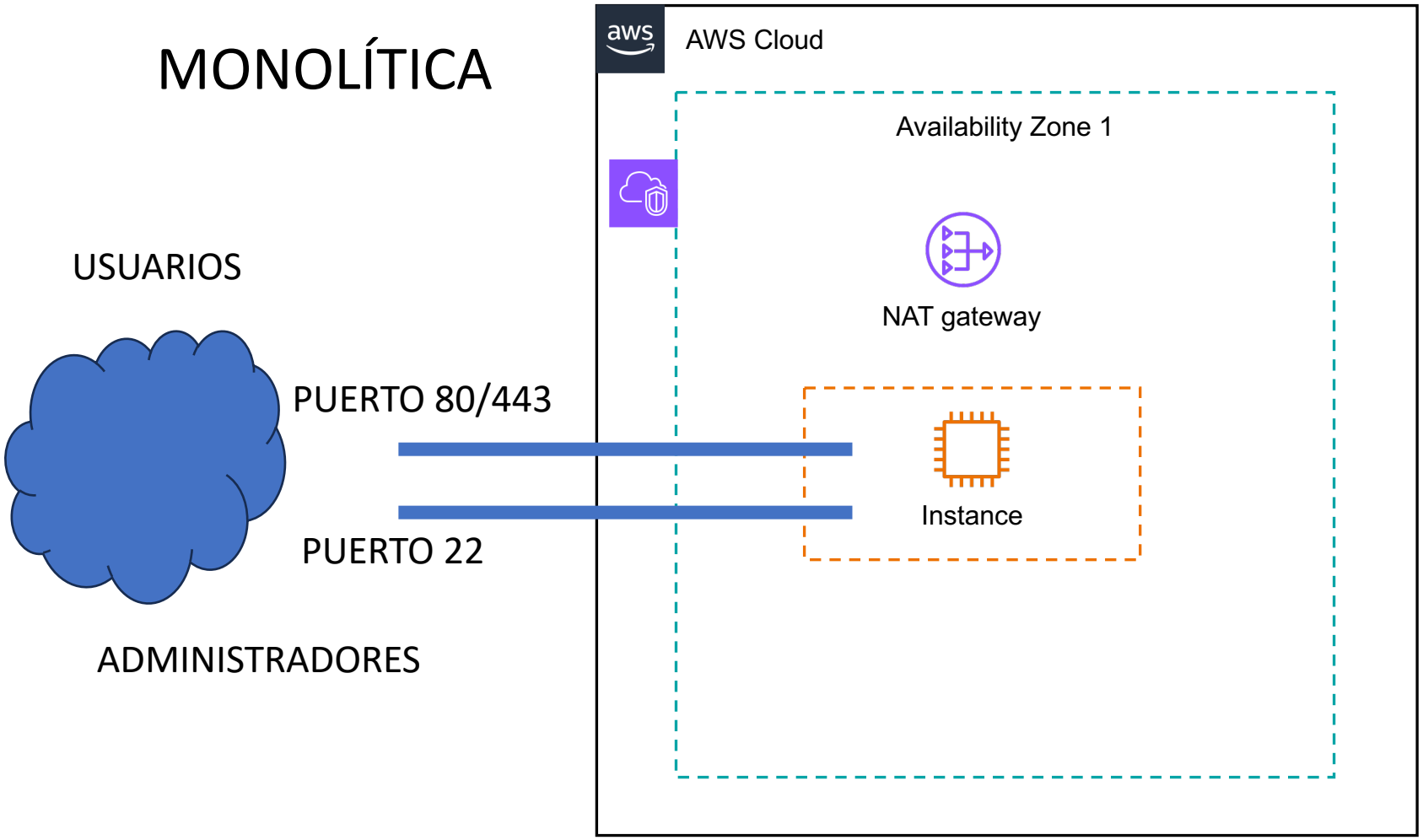
ARQUITECTURAS DE NUBE

GENERALIDADES



ARQUITECTURAS DE NUBE

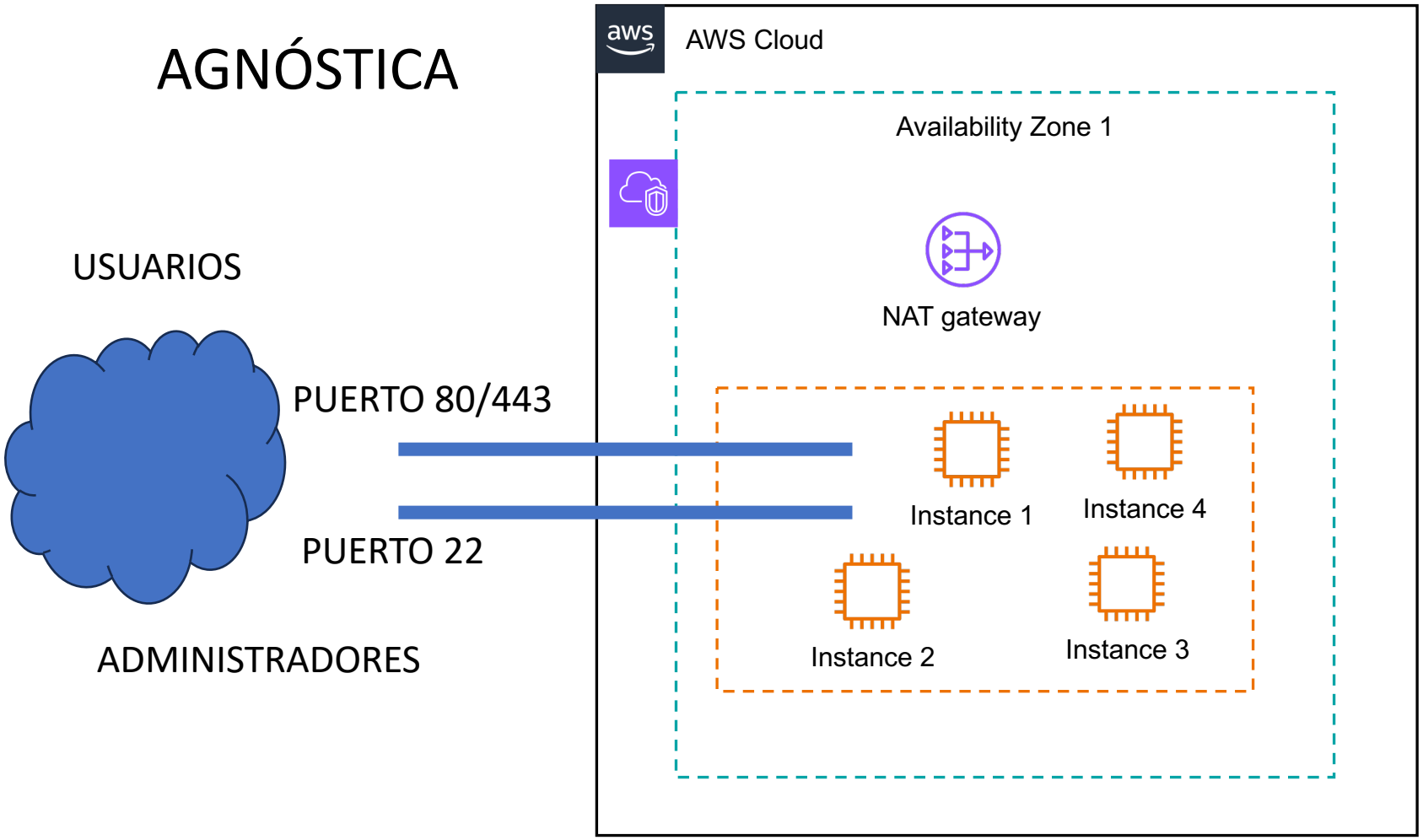
MONOLÍTICA



- PARA PRUEBA DE CONCEPTO
- BARATA
- RAPIDA DE MONTAR
- FACIL DE GESTIONAR
- FACIL DE HACKEAR
- REQUIERE COMPONENTES MINIMOS
- RECUERDE QUE HAY PLANO DE CONTROL Y DE USUARIO

ARQUITECTURAS DE NUBE

AGNÓSTICA

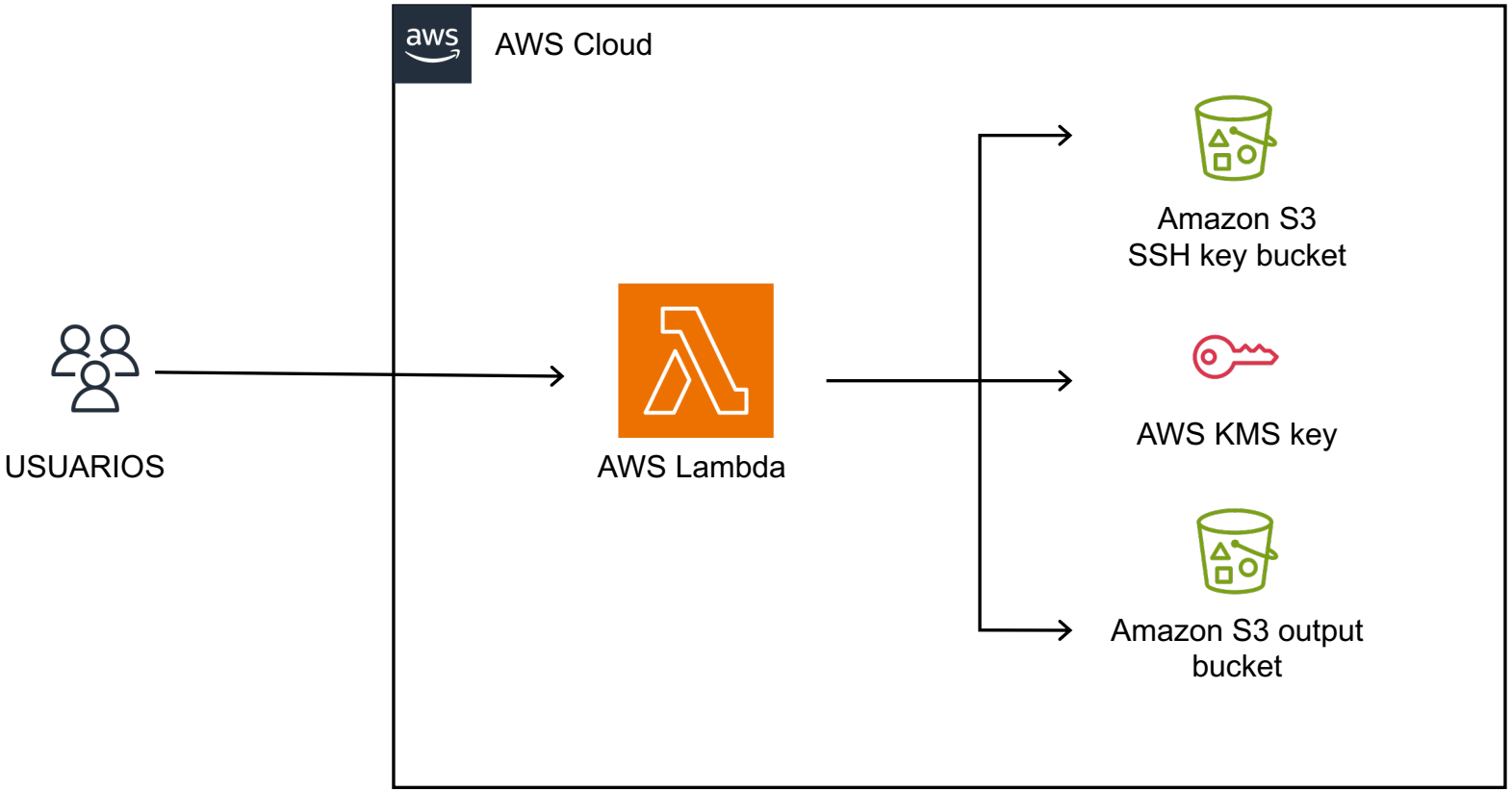


- PARA PRUEBA DE CONCEPTO
- BARATA
- RAPIDA DE MONTAR
- MEDIO DE GESTIONAR
- FACIL DE HACKEAR
- REQUIERE COMPONENTES MINIMOS
- RECUERDE QUE HAY PLANO DE CONTROL Y DE USUARIO
- DEDICAR MUCHO TIEMPO A LA GESTIÓN DE LA INSTANCIA

TODOS LOS SERVICIOS DE FORMA DISCRETA

ARQUITECTURAS DE NUBE

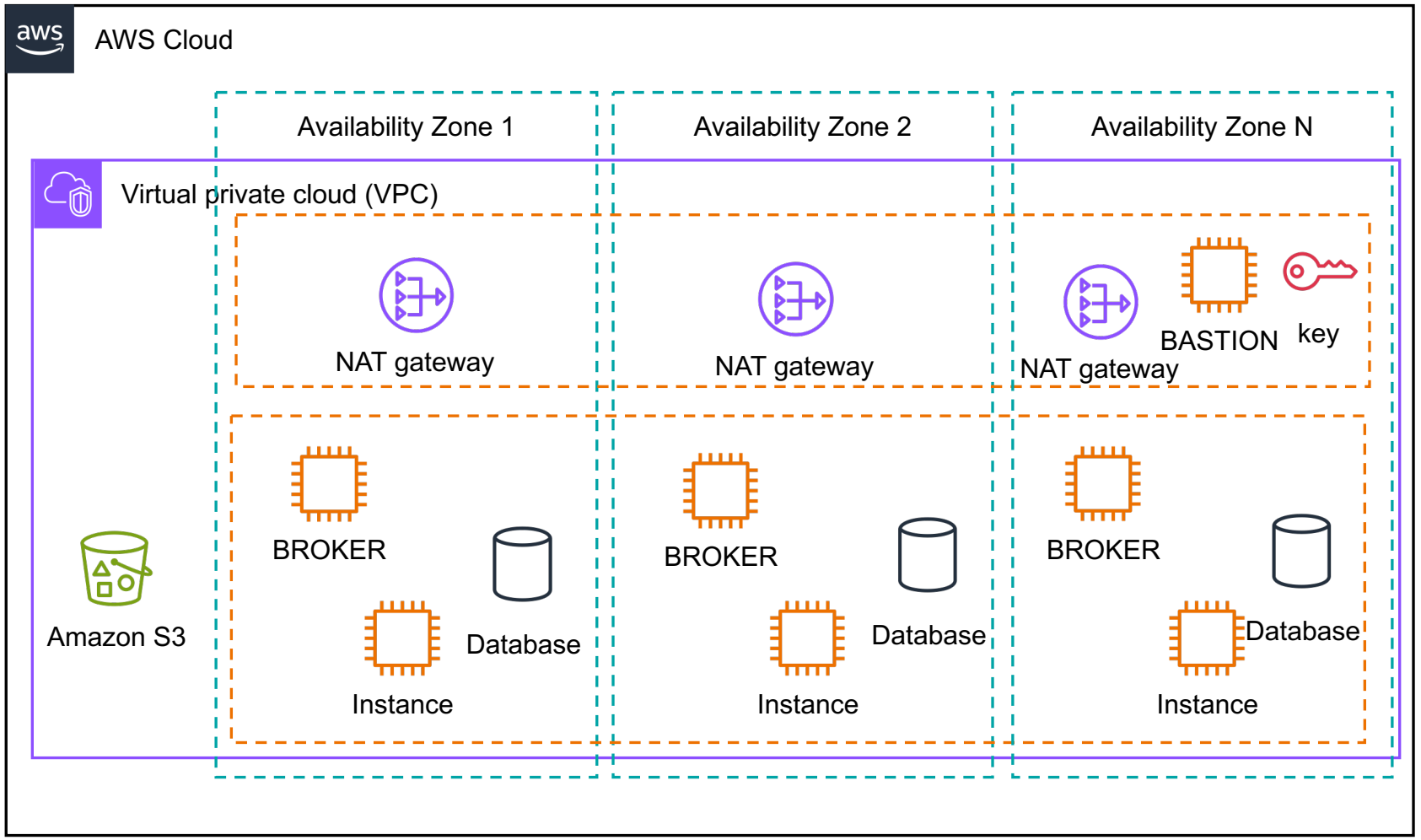
DEPENDIENTE DE LA NUBE – VENDOR LOCKIN



- PARA PRODUCCIÓN
- BARATA
- RAPIDA DE MONTAR
- FÁCIL DE GESTIONAR
- DIFÍCIL DE HACKEAR
- REQUIERE COMPONENTES MINIMOS
- DEDICAR POCO TIEMPO A LA GESTIÓN DE LA INSTANCIA
- DEPENDO TOTALMENTE DEL PROVEEDOR

ARQUITECTURAS DE NUBE

LA MÁS PROFESIONAL



EL AGENTE U ORQUESTADOR (BROKER O DISTRIBUIDOR)

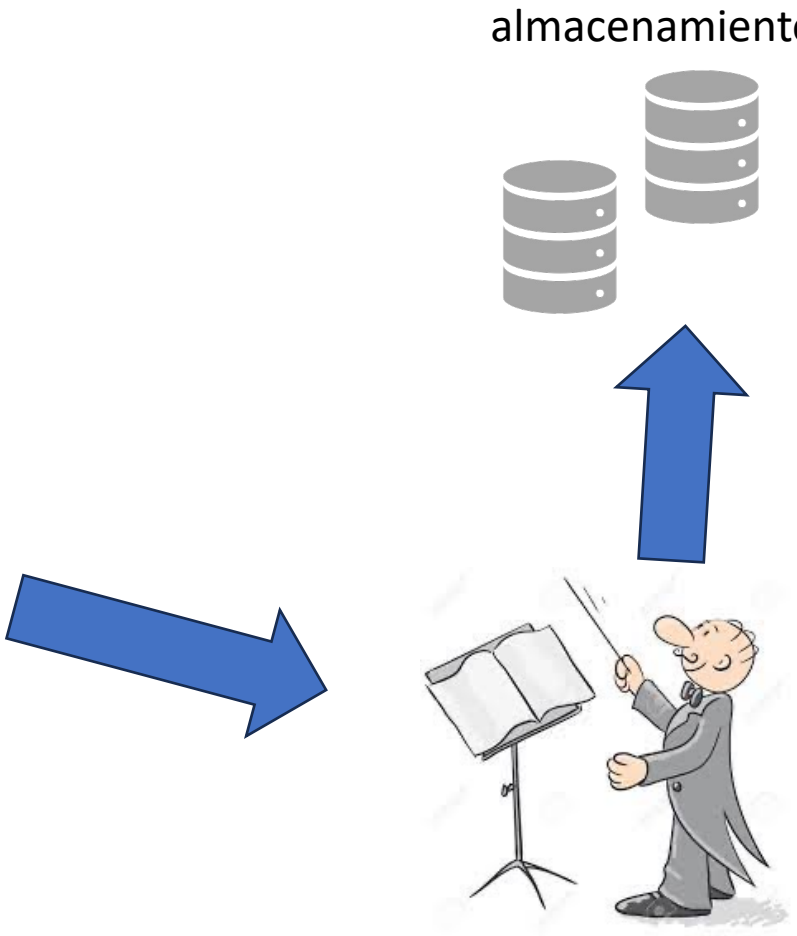
GENERALIDADES

Un orquestador es una ENTIDAD que gestiona el ciclo de los datos y acciones en una solución de IoT – Crea procesos/Jobs/eventos – administra los datos – genera notificaciones

- Lo desarrollamos a código propio = máquina de estados
- Lo adaptamos de una plataforma (ORION por ejemplo)

EL AGENTE U ORQUESTADOR (BROKER O DISTRIBUIDOR)

GENERALIDADES



RECIBE LOS DATOS DE LOS SENSORES
VALIDA FUENTE
QUITA CIFRADO
PUEDE AGREGAR CAMPOS (FECHA)
MANDA A ALMACENAR

EL AGENTE U ORQUESTADOR (BROKER O DISTRIBUIDOR)

GENERALIDADES

SISTEMAS DE ALMACENAMIENTO

GENERALIDADES

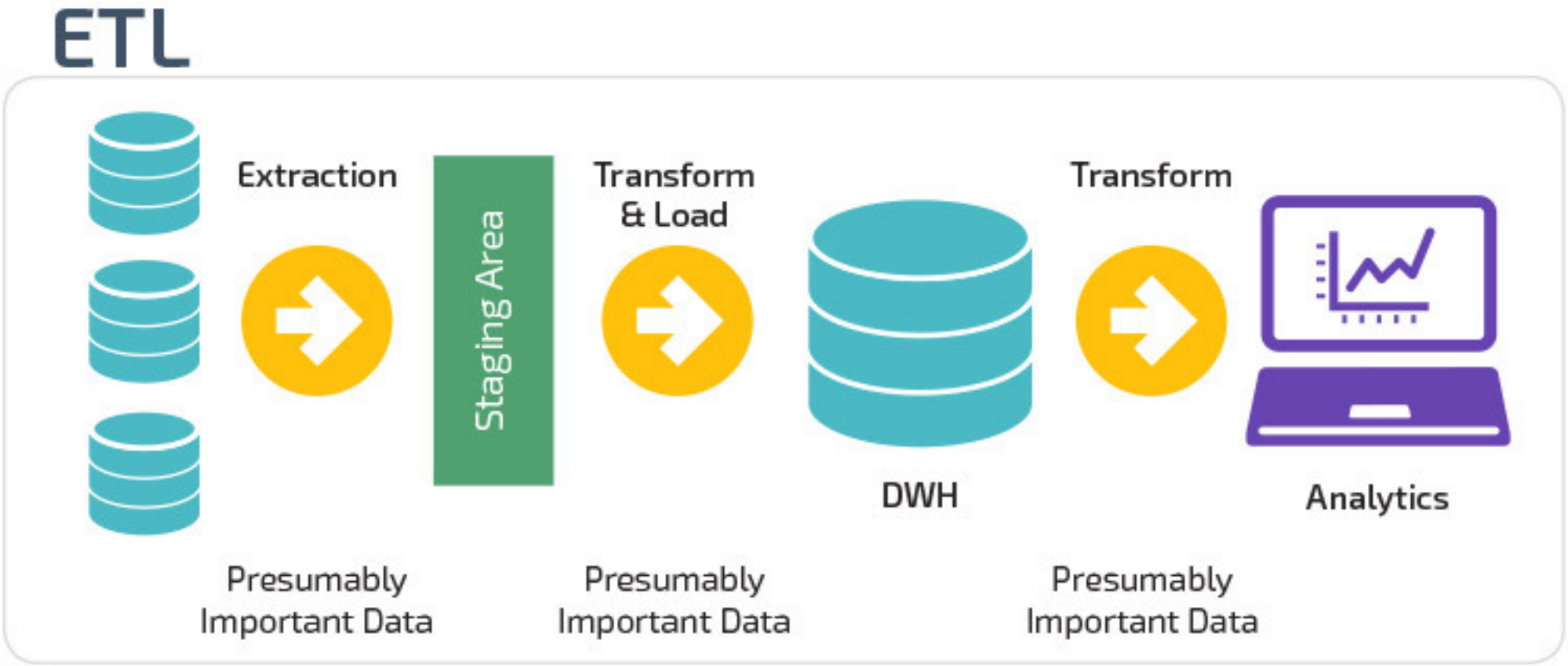
En un archivo

En un motor

En una entidad de almacenamiento de nube

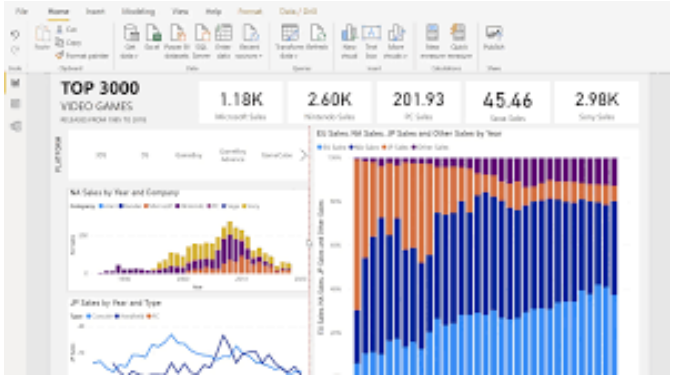
SISTEMAS DE ETL

GENERALIDADES



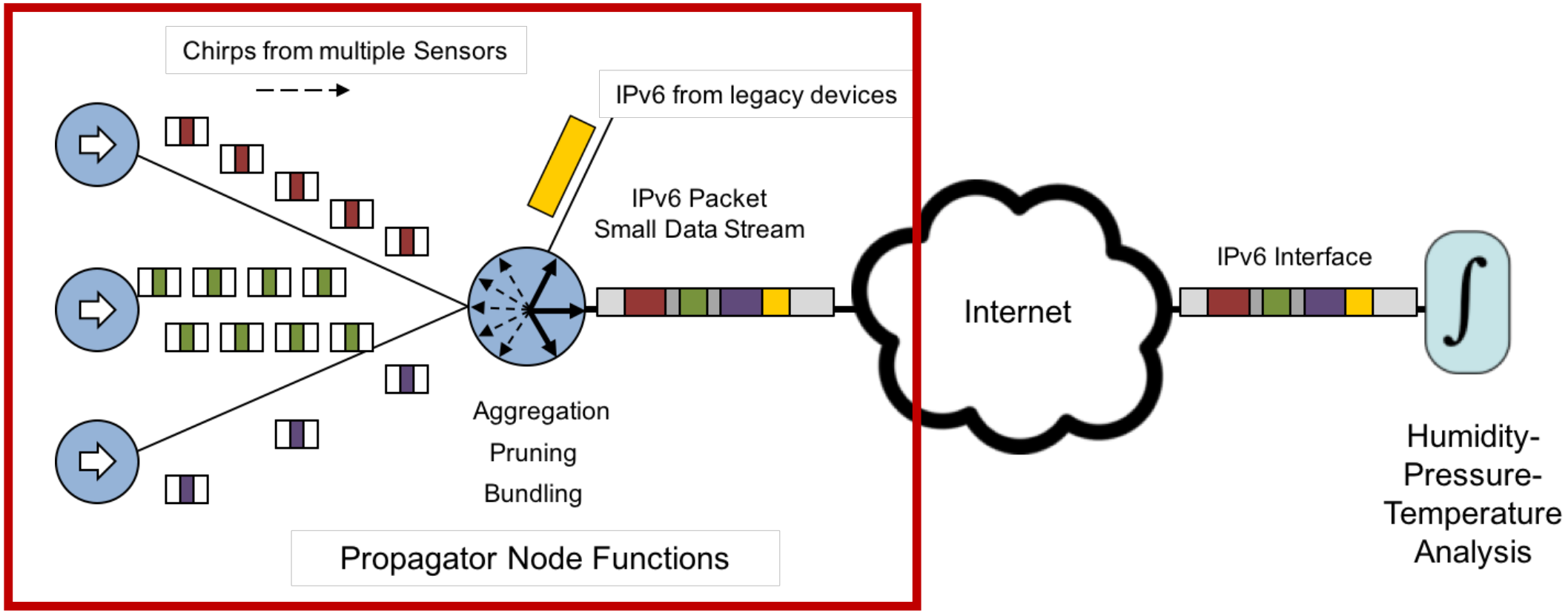
SISTEMAS DE VISUALIZACIÓN

GENERALIDADES



EJERCICIO DEL AGENTE – CONTRUYAMOS UN AGENTE BASICO Y UNO INTERMEDIO

Capa de datos y sensado



EJERCICIO DEL AGENTE – CONTRUYAMOS UN AGENTE BASICO Y UNO INTERMEDIO

Capa de datos y sensado

- AGREGATION: proceso de tomar el chirp de muestras de un sensor, y concentrarlo en un procesador de un end device
- PRUNING: proceso de reducir el número de muestras del chirp de medidas, aplicando técnicas de estimación, corrección/calibración, teorema central del límite, para determinar el valor medido en el proceso de sensado
- BUNDLING: proceso de construir la información para enviar por IPv6 o IPv4 hacia o desde el Gateway, esta función la puede o no hacer el End Device, también se puede hacer en el collector o en el mismo gateaway, todo depende de la arquitectura de hardware escogida (puede ser un único dispositivo el que hace todo, o puede estar distribuidas las funciones en la arquitectura)

EJERCICIO DEL AGENTE – CONTRUYAMOS UN AGENTE BASICO Y UNO INTERMEDIO

Capa de datos y sensado

DISEÑO DE UNA TRAMA

DATAGRAMA IP

# SEC	ID SENSOR	FORMATO	DATA	CHECKSUM	FIRMA

# SEC	ID SENSOR	DATA

# SEC	ID SENSOR	TIMESTAMP	LOCATION	SENSOR 1 DATA	…	SENSOR N DATA	CHECKSUM	FIRMA

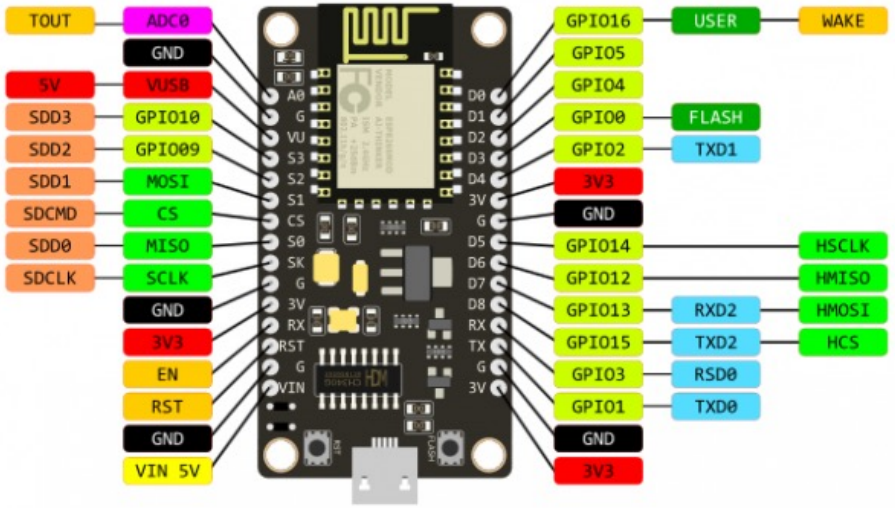
CABECERA

CUERPO / DATOS

CIERRE/DEAMBULO/SEGURIDAD

EJERCICIO DEL AGENTE – CONTRUYAMOS UN AGENTE BASICO Y UNO INTERMEDIO

Capa de datos y sensado



REAL	PLACA		SENSOR
GPIO5	D1	⟷	SCL
GPIO4	D2	⟷	SDA
3V	3V	⟷	3V
G	G	⟷	G

SI NO TIENES MICRO, PUEDES USAR POSTMAN PARA EMULAR LOS PAQUETES QUE VAN AL SERVIDOR – VER VIDEO

EJERCICIO DEL AGENTE – CONTRUYAMOS UN AGENTE BASICO Y UNO INTERMEDIO

Capa de datos y sensado

TAREA
CREE UNA FUNCIÓN QUE MANDE LOS DATOS A LA NUBE EN SU ORQUESTADOR

<usa la librería wifi y hace un post a la dirección del agente>

```
#include <ClosedCube_HDC1080.h>
#include <Wire.h>

ClosedCube_HDC1080 sensor;

void setup() {
sensor.begin(0x40);
Serial.begin(9600);
}

void loop() {
  double temperatura = sensor.readTemperature();
  double humedad = sensor.readHumidity();

  Serial.print("Temperatura = ");
  Serial.print(temperatura);
  Serial.print("ºC  Humedad = ");
  Serial.print(humedad);
  Serial.println("%");
  delay(2000);
}
```

EJERCICIO DEL AGENTE – CONTRUYAMOS UN AGENTE BASICO Y UNO INTERMEDIO

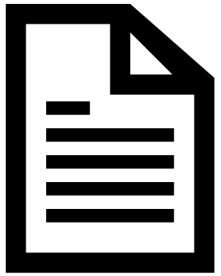
```
void senddata()
{
  String PostData = "";
  PostData = String("id="+String(id)+"; temperatura="+String(temperatura,7)+"; longitud="+String(longitud,7)+"; latitud="+String(latitud,7));
 if ( client.connect(server,80))
  {
    Serial.println("conectado");
    client.print("POST /sensor_send_data HTTP/1.1\n");
    // poner la direccion IP del servidor
    client.print("Host: 192.168.0.101 \n");
    client.println("User-Agent: Arduino/1.0");
    client.println("Connection: close");
    client.println("Content-Type: application/x-www-form-urlencoded;");
    client.print("Content-Length: ");
    client.println(PostData.length());
    client.println();
    client.println(PostData);
  } else {
    Serial.println("error de conexion");
  }
}
```

EJERCICIO DEL AGENTE – CONTRUYAMOS UN AGENTE BASICO Y UNO INTERMEDIO

AGENTE BÁSICO

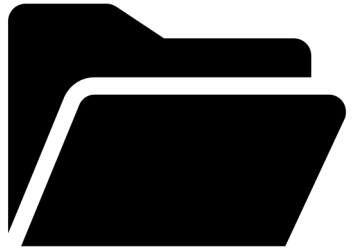
APP01.py

SOLO VERIFICAMOS QUE LLEGAN LOS DATOS



APP02.py

USAMOS UNA BASE DE DATOS LOCAL EN ARCHIVO



TAREA

USAMOS UNA BASE DE DATOS LOCAL EN ARCHIVO Y CON INTERFAZ VISUAL

EJERCICIO DEL AGENTE – CONTRUYAMOS UN AGENTE BASICO Y UNO INTERMEDIO

AGENTE BÁSICO

```
from flask import Flask, render_template, jsonify, request

app = Flask(__name__)

@app.route('/')
def home():
return 'hola mundo desde mi broker simple'
@app.route('/sensor_send_data', methods=['POST'])
def sensor_send():
values = request.data
print(values)
return "datos recibidos ok",201

if __name__ == '__main__':
app.run(debug=True,host='0.0.0.0',port=80)
```