**Project Name: Borewell Water & Utility Monitoring System**

**🧱 1. Architecture Overview**

**📐 High-Level Design (HLD)**

* **Data Source**: IoT Edge Devices → Flow Meters
* **Ingestion**: Azure IoT Hub → Stream Analytics
* **Processing**: Azure Functions → Azure Stream Analytics → Azure Machine Learning
* **Storage**: Azure SQL + Azure Data Lake
* **Visualization**: Power BI Dashboards
* **Automation**: Azure Functions + Power Automate
* **DevOps**: Azure DevOps, Terraform, YAML pipelines
* **Security**: Azure NSGs, Firewalls, Azure Defender, Key Vault
* **Monitoring**: Azure Monitor, Application Insights, Log Analytics

**⚙️ Low-Level Design (LLD)**

IoT Devices → Azure IoT Hub → Azure Stream Analytics →

├─→ Azure SQL (Processed Data)

└─→ Azure Data Lake (Raw Data)

Azure Functions (Alerts) → Email/Teams

Azure ML (Forecasting) → Azure SQL

Power BI → Azure SQL & Data Lake

Power Automate → Email Reports

**🧱 2. Service Types**

| **Service Type** | **Example in Project** |
| --- | --- |
| **Monolithic** | Power BI dashboard, Power Automate workflows |
| **Microservices** | IoT Data Processor, Alert System, ML Predictor APIs |
| **Backend** | Azure Functions, Azure Stream Analytics, REST APIs |
| **Frontend** | Power BI Dashboards |
| **Database** | Azure SQL Database, Azure Data Lake |

**☁️ 3. Azure Infrastructure (Provisioned via Terraform)**

* **Compute**: Azure Functions, Azure IoT Edge runtime
* **Network**:
  + Virtual Network (VNet)
  + Subnets for IoT, Backend, DB
  + Network Security Groups (NSGs)
  + Application Gateway / Firewall
* **Data**:
  + Azure SQL (structured)
  + Data Lake Gen2 (raw + semi-structured)
* **Monitoring**:
  + Azure Monitor
  + Log Analytics
  + App Insights
* **Security**:
  + Azure Defender for Cloud
  + Azure Key Vault (Secrets, Connection Strings)
  + Azure Firewall
* **DevOps**:
  + Azure DevOps YAML Pipelines
  + Git Repositories
  + Terraform (Infrastructure as Code)

**🔁 4. CI/CD Pipeline - YAML (Azure DevOps)**

yaml

trigger:

branches:

include:

- main

variables:

azureSubscription: 'AzureServiceConnection'

environment: 'prod'

stages:

- stage: Terraform\_Deployment

jobs:

- job: terraform

steps:

- checkout: self

- task: TerraformInstaller@0

inputs:

terraformVersion: '1.3.0'

- script: |

terraform init

terraform validate

terraform plan -out=tfplan

terraform apply tfplan

displayName: 'Run Terraform'

- stage: Deploy\_Functions

jobs:

- job: deployFunctions

steps:

- task: AzureFunctionApp@1

inputs:

appType: 'functionApp'

azureSubscription: '$(azureSubscription)'

appName: 'iot-water-alerts'

package: '$(System.DefaultWorkingDirectory)/function.zip'

**🔒 5. Security Architecture**

| **Layer** | **Implementation** |
| --- | --- |
| **Network** | VNet + NSGs + Private Endpoints + Azure Firewall |
| **Secrets** | Azure Key Vault |
| **Access Control** | RBAC + Azure AD Identity for Apps |
| **Threat Protection** | Azure Defender, DDoS Protection, Microsoft Sentinel |

**📊 6. Monitoring & Alerting**

| **Tool** | **Purpose** |
| --- | --- |
| Azure Monitor | Infrastructure and service-level metrics |
| App Insights | Function-level telemetry and performance |
| Log Analytics | Query logs across all services |
| Alerts | Triggered via Azure Monitor/Functions |

**📈 7. Additional Systems**

**💡 Electricity Monitoring**

* Energy Meters → Azure IoT Hub
* Process: Similar Stream Analytics + Power BI + Alerts
* Benefits: Peak load analysis, load balancing

**⚙️ Motor Runtime Monitoring**

* Vibration/Current Sensors → Azure IoT Hub
* Triggers Predictive Maintenance via ML and Alerts

**✅ Key Outcomes**

* 🔹 Real-time monitoring of 35+ borewells
* 🔹 Predictive water demand using ML
* 🔹 Automated alerts for over-usage
* 🔹 Energy optimization via dashboard
* 🔹 Reduced manual errors and improved visibility
* 🔹 IaC-enabled scalable infrastructure

**Interview Answer: Borewell Water Consumption Monitoring System**

**Project Overview:**

In my current organization, I worked on a project called **“Borewell Water Consumption Monitoring System”**, aimed at automating the tracking of water usage from 35 borewells across multiple industrial plants. Earlier, the readings were taken manually, which often led to errors and delays in decision-making.

**Problem Statement:**

Each borewell could pump up to **65 KL/hour**, and manual tracking was inefficient and inaccurate. There was a clear need to monitor water usage in **real-time**, generate **alerts**, and implement **predictive analytics** for better resource planning.

**My Role as a DevOps Engineer:**

I was responsible for:

* **End-to-end infrastructure automation** using **Terraform**
* Designing and implementing **CI/CD pipelines** with **Azure DevOps (YAML-based)**
* Setting up **secure and scalable Azure resources**
* Ensuring monitoring, alerting, and system reliability
* Supporting backend and data engineers in deployment and scaling

**Solution Architecture:**

We installed **Ultrasonic and Electromagnetic Flow Meters**, which connected to **IoT Edge devices**. These sent real-time data every minute to **Azure IoT Hub**. The data then flowed through:

1. **Azure Stream Analytics** – for real-time data processing
2. **Azure SQL Database** and **Data Lake Gen2** – for structured and unstructured storage
3. **Azure Functions** – to trigger alerts if water usage exceeded limits
4. **Azure Machine Learning** – for forecasting future water needs
5. **Power BI** – for live dashboards
6. **Power Automate** – for sending daily usage reports via email

**DevOps & Infrastructure Details:**

* **Infrastructure as Code (IaC)** using **Terraform**:
  + Provisioned Azure IoT Hub, Stream Analytics, Function Apps, SQL, VNet, NSGs, Azure Firewall, and Storage
* **CI/CD Pipelines** using **Azure DevOps (YAML)**:
  + Automated deployment of function apps and backend microservices
* **Security**:
  + Implemented **NSGs**, **Azure Key Vault**, and **RBAC**
  + All services deployed within a **secured VNet** with **private endpoints**
* **Monitoring**:
  + Used **Azure Monitor**, **Log Analytics**, and **Application Insights**
  + Set up custom alerts and dashboards

**Service Architecture:**

* **Microservices**: For data processing, alert generation, ML predictions (via APIs)
* **Monolithic**: Power BI and Power Automate workflows (centralized dashboards)
* **Frontend**: Power BI Dashboards
* **Backend**: Azure Functions, APIs
* **Database**: Azure SQL, Data Lake

**Additional Utility Monitoring:**

* Created an **Electricity Monitoring Dashboard** by integrating energy meter data
* Implemented **Motor Running Hours Monitoring** using IoT data to plan **predictive maintenance**
* These helped reduce downtime and optimize energy consumption

**Impact:**

* ✅ Real-time monitoring of water and electricity usage
* ✅ Reduced manual errors and improved response times
* ✅ Forecasting helped optimize borewell scheduling and energy use
* ✅ Infrastructure provisioning time reduced significantly via Terraform
* ✅ Improved operational efficiency and resource management

**Closing:**

This project allowed me to apply my DevOps skills in real-world industrial automation, leveraging cloud-native tools, IaC, CI/CD, and monitoring to deliver a secure and scalable solution that improved visibility and decision-making.