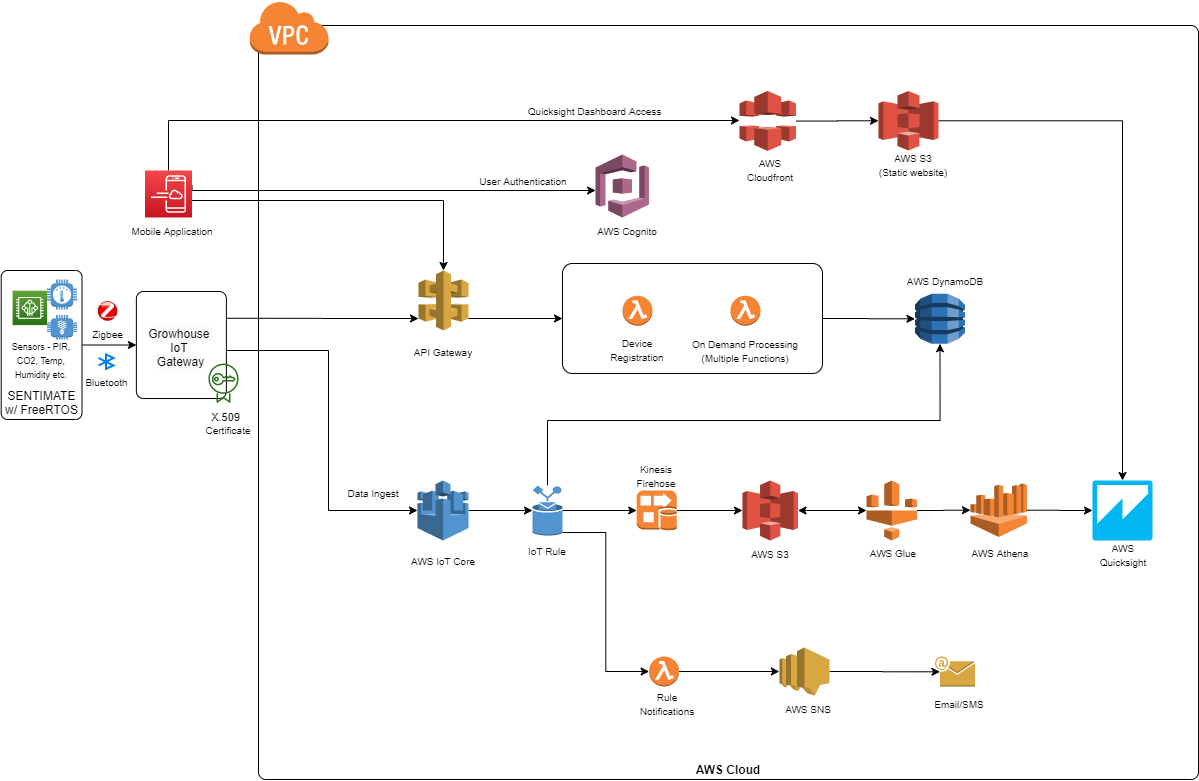
### EFR32 IoT Gateway Cloud Design Document



1. Gateway Registration
   * The gateway receives the request to register the gateway from the mobile app over BLE.
   * The gateway calls the AWS API Gateway 'creatething' API to register.
   * The API invokes a lambda function that will:
     + create a thing in AWS IoT Core
     + generate necessary certificates
     + create SNS subscription for SMS and Email
     + create a gateway entry in the AWS DynamoDB table
   * All of the necessary details are sent in the API response to the gateway which stores them for future use.
   * Gateway sends the "success/failure" response to the mobile application over BLE
2. Sensor Registration
   * The gateway receives the request to register the sensor from the mobile app over BLE.
   * The gateway will search for the available Zigbee devices and send the list to the mobile application.
   * User will select which sensor to provision and the payload will be sent to the gateway.
   * The gateway calls the AWS API Gateway 'creatething' API to register.
   * The API invokes a lambda function that will:
     + create a thing in AWS IoT Core
     + create a sensor entry in the AWS DynamoDB table
   * All of the necessary details are sent in the API response to the gateway which stores them for future use.
   * Gateway sends the "success/failure" response to the mobile application over BLE
3. Gateway Deletion
   * The gateway to be deleted is selected by the user in the mobile app.
   * The mobile app searches for the selected gateway over BLE.
   * Once the gateway is found, the mobile application calls the AWS API 'deletething' to delete that gateway from the cloud
   * The API will invoke a lambda function that will:
     + remove any certificates associated with that gateway
     + delete any sensors associated with that gateway
     + remove the entries from the AWS DynamoDB table
   * The response is collected by the mobile application
   * The mobile application then sends a 'gateway deletion' request to the gateway.
   * The gateway will clear all the details it had stored when registering the gateway
   * If the gateway is not available over BLE, the mobile application will call the API 'deletething' and then the gateway needs to be hard-reset.
4. Sensor Deletion
   * The sensor to be deleted is selected by the user in the mobile app.
   * The mobile app searches for the gateway which is associated with sensor over BLE.
   * Once the gateway is found, the mobile application calls the AWS API 'deletething' to delete that sensor from the cloud
   * The API will invoke a lambda function that will:
     + delete any sensors associated
     + remove the entries from the AWS DynamoDB table
   * The response is collected by the mobile application
   * The mobile application then sends a 'sensor deletion' request to the gateway.
   * The gateway will clear all the details it had stored when registering the gateway
   * If the gateway is not available over BLE, the mobile application will call the API 'deletething' and then the gateway needs to be hard-reset.
5. Telemetry
   * The gateway connects to the AWS IoT Core securely using the certificates it stored while gateway registration
   * Once connected, the gateway sends the telemetry data to AWS IoT Core using MQTT protocol.
   * When the data reaches cloud, IoT Rules are invoked, which check for value thresholds and send notifications via Email/SMS if required.
   * Another rule is invoked to write this data in the DynamoDB table.
   * A separate rule is invoked which puts this data received in the Kinesis Firehose stream.
6. IoT Rule for Threshold value
   * Rule will be invoked once cloud will receive data on MQTT topic from provisioned sensor.
   * Rule will execute Lambda function which will check current threshold value for specific sensor.
   * Lambda function will also check notification value for Email and SMS.
   * Based on threshold value and notification value user will get alert notification via Email/SMS if value would be out of range.
7. User Management
   * For the user management, AWS Cognito service is used which allows user sign/signup flow easily.
   * When the user opens the Mobile App, he/she can create an account which whose details will be stored in the Cognito User Pool.
8. Quicksight Dashboard
   * Quicksight dashboard is used to prepared History chart for specific sensor.
   * Dashboard further getting publish using aws public domain url and same url is being used by mobile application to display the data.
   * IoT Rule will be invoked when sensor data comes to aws cloud and that data get pass to kinesis firehose to store in S3 bucket.
   * S3 bucket data will be read by Glue to convert into specific format and this converted data again publish to Athena.
   * Athena will store data into workgroup which will be used by Quicksight to prepare dashboard.
   * Once this setup done, need to create quicksight analyses for different kind of dashboard look.