Envision
Engineering
AMERICAS PROTOTYPING

HVAC real time monitoring and observability

Table of Contents

Overview

Octank Beauty is a leading company of beauty and healthcare products sold all over the world. Their products range from lipsticks and eyelashes to skin moisturizers, toothpaste and shampoos. Octank beauty operates several factories around America and many of its manufacturing processes involve controlling the environmental conditions of the rooms where manufacturing is taking place.

Currently, Octank operates an industrial grade HVAC (Heating, Ventilation and Air Conditioning) to maintain some environmental conditions in several rooms of the manufacturing plant. Furthermore, Octank saves measurements of around 800 sensors installed throughout all of the components in the HVAC system.

Octank has reported that HVAC accounts for around x% of the total energy consumption in the America facilities, furthermore, failures in the HVAC system have to be taken care of in a timely manner to ensure the operation of certain processes is not affected.

In the current document, we detail a prototype, created by AWS Envision Engineering team, that aims to help Octank, improve the operation of their HVAC system. The prototype is made up of three major components: A Business Intelligence (BI) solution tailored for the upper management of Octank, a real time monitoring solution that helps the operations team monitor in real-time the behavior of Octank's HVAC system, and an anomaly detection module for automatically detecting anomalies in the operation of the HVAC system.

AWS team

Envision Engineering

- David Laredo (Prototyping Architect – razodav@amazon.com)

Account team for Octank

- Leticia Santos (CTO – sletic@amazon.com)
- Lidio Ramalho (CIO - lidior@amazon.com)

Scope

This prototype provides data insights generated by analyzing the measurements of the sensor data from the HVAC system at Octank's America facility. These sensor readings follow three main flows depending on the type of outcome they provide:

1. Business Intelligence dashboards tailored for the upper management of the manufacturing plan. These dashboards provide aggregated information of the overall operation of the HVAC system.
2. Real time dashboards of the operation of the HVAC, aimed at the operations team, that will help them monitor, in real-time, the behavior of the HVAC system. Furthermore, these dashboards can help the operations team identify and isolate failures in the system in a timely manner.
3. An anomaly detection module that analyzes trends in the data collected by the sensors and detects if the components (or the system itself) is at an anomalous state given the conditions of the moment. This module sends notifications to the operations team based on the number of anomalies detected over a configurable period of time.

The anomaly detection process is, in general, as follows:

1. Sensor data is pulled from the system at configurable time periods
2. Data is transformed using the rolling window approach
3. Data is passed into a custom anomaly detection model
4. The model returns whether the records are anomalous or not
5. Based on the number or anomalies detected over the time, alerts are sent out via email

These data insights are stored alongside the date/time of the recording and component identifiers. With the date and component identifiers, it's possible to filter the data in different ways.

Visualization and aggregation of the data insights is possible via an interactive dashboard, accessed via web browser. Optionally, a summary could be emailed on a daily basis to selected users.

Not in scope

Following are the features that are **not** in the scope of this prototype:

1. Forecasting of the data points.
2. Failure tracing.
3. Recommendations for fixing the failure

Octank’s HVAC system

Heating, Ventilation and Air Conditioning (HVAC) is the technology of indoor and vehicular environmental comfort. Its goal is to provide thermal comfort and acceptable indoor air quality. A usual HVAC system is made up of the following major components: air handling unit (AHU), staged air volume (SAV), variable air volume (VAV) and Thermafusers. Furthermore, AHUs, SAVs, VAVs and Thermafusers may contain the following components: dampers, filters, heat exchanging coils (HECs) and Fans.

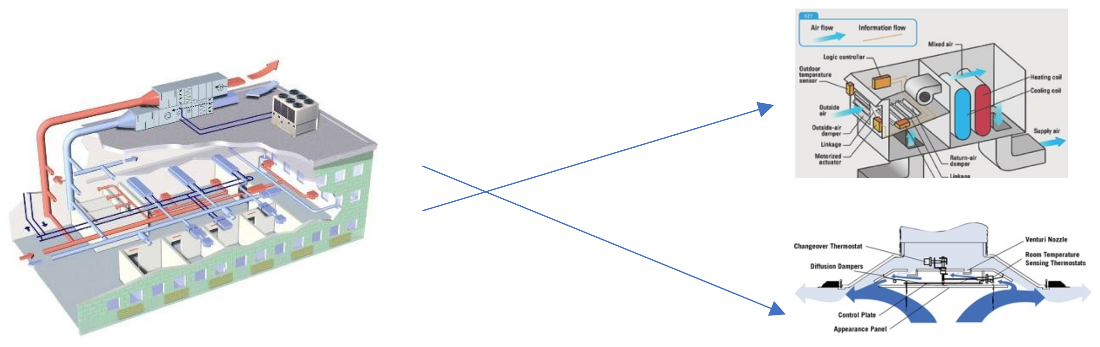


Figure 1: An HVAC systema and its major component; an AHU (left) distributes air to the rooms via pipes that deliver the air though thermafusers. The air is processed in the AHU using heat exchangers, fans, dampers and filters.

As with any system, there are several relationships between each of the components of the system and their relationships among each other affect the overall behavior of the system. Knowing these relationships is important for pinpointing the location of the failure and tracing it to its root. The following illustration displays how each of the HVAC’s components are related to each other.

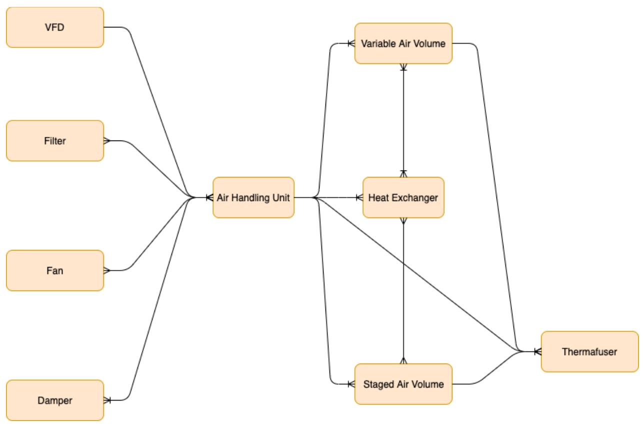


Figure 2: An entity-relation diagram of Octank’s HVAC system

Octank America’s HVAC is composed of one AHU and more than 100 different thermafusers distributed all over the factory. Together, these components contain around 50 components such as dampers, heat exchangers, filters and fans. Altogether, all of the components have more than 800 sensors in total.

BI dashboards

Following there are screenshots of the BI dashboard aimed for the upper management of Octank America. The following plots are generated using Amazon Quicksight, which is a scalable, serverless, embeddable, machine learning-powered business intelligence (BI) service built for the cloud.

The management can oversee aggregates of all of the HVAC component, these aggregates and visualizations can be filtered by component or type of component, and they can be grouped by several dimensions such as day, month, hour, etc.

We are proposing one set of plots per component. In the following we present some of the most relevant plots. These plots can be easily customized to fit the needs of the management team.

The AHU dashboard presents a view of the main sensors within the air handling unit. We present an average of heating and cooling requests by month and grouped by day. This visualization works well for comparing only two months, including more months will make the visualizations very crowded and difficult to interpret.

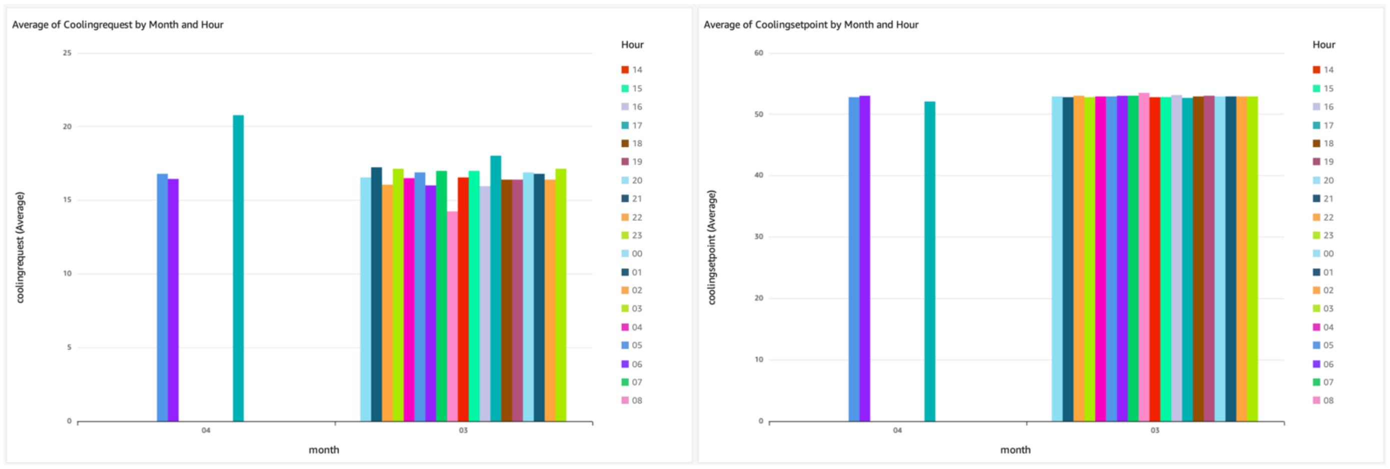


Figure 2: Average cooling and heating setpoints by month and grouped by day.

Next, we show plots of CO2 levels (left) and air temperature (right) for the AHU sensors. These two plots can help visually identify whether the AHU is operating normally or abnormally based on CO2 and temperature levels.

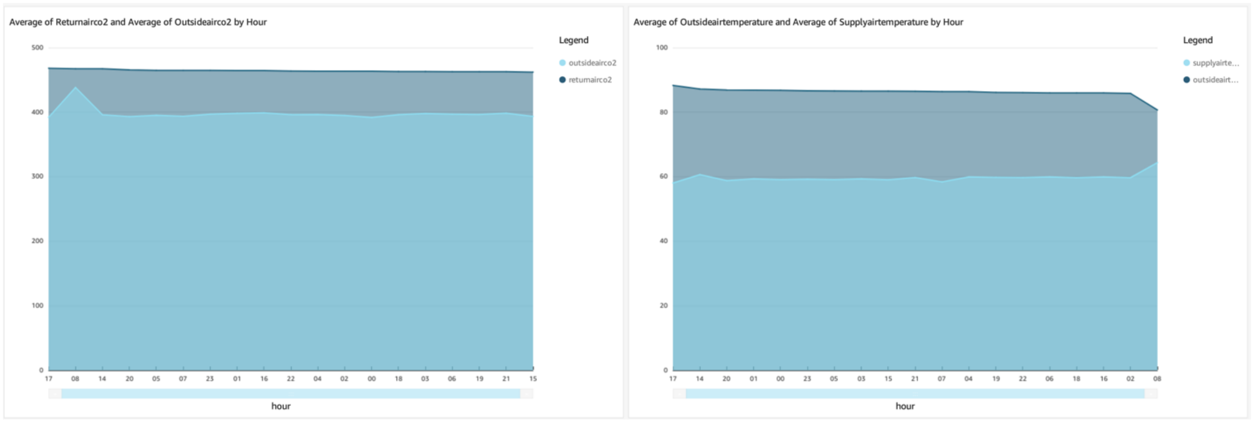


Figure 3: CO2 and temperature plots. These plots can help visualize identify anomalies

We have created a set of plots specifically for the anomalies in the system. In the following plot we can easily visualize the count of anomalies per day grouped by hour, we also created scatter plots of the sensors within the thermafuser to make the identification of anomalies visually easier.

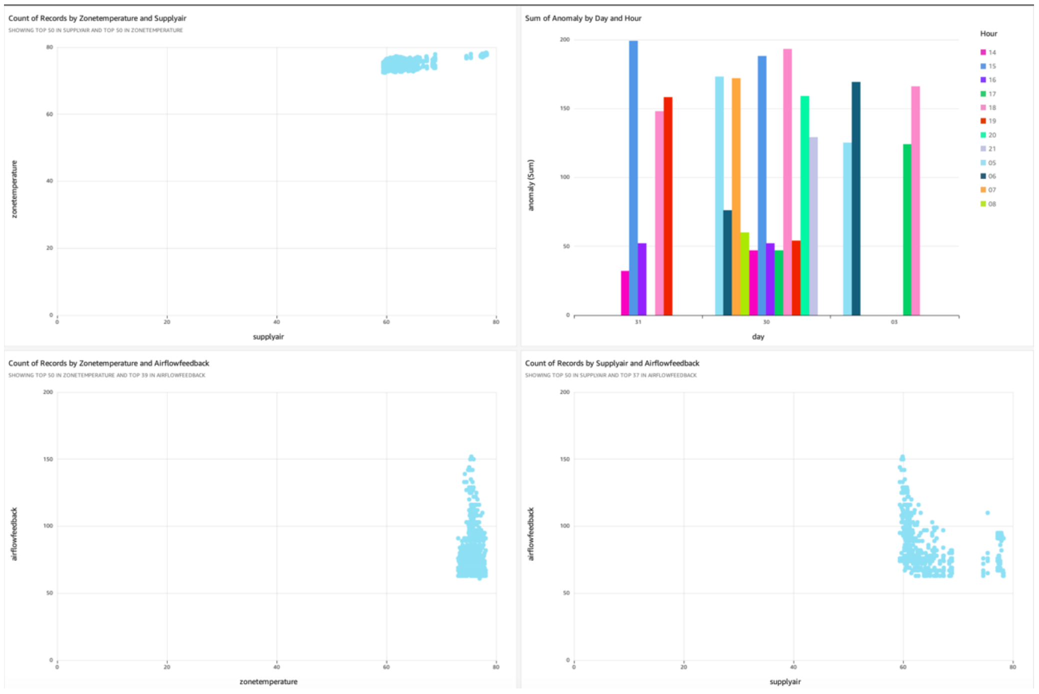


Figure 4: Anomaly detection plots used for easily spotting when the anomalies happened and what was the likely cause.

Real-time operations dashboard

Following there are screenshots of the real-time operations dashboard aimed for the operations and maintenance team of Octank America. The following plots are presented using the managed Grafana service, which is a fully managed service that is developed together with Grafana Labs and based on open source Grafana.

Using these visualizations, the operations team can easily oversee, in real time, the current status of the several components that make up the HVAC system of Octank America. Such plots include line plots, gauge plots, bar plots, etc.

We present the plots for the most relevant sensors in a single dashboard, the final user is free to easily adjust these dashboards to better suit their needs.

As mentioned earlier, users can generate gauge plots to measure a single dimension. In this case, we are measuring the angular speed of the two fans inside the AHU.

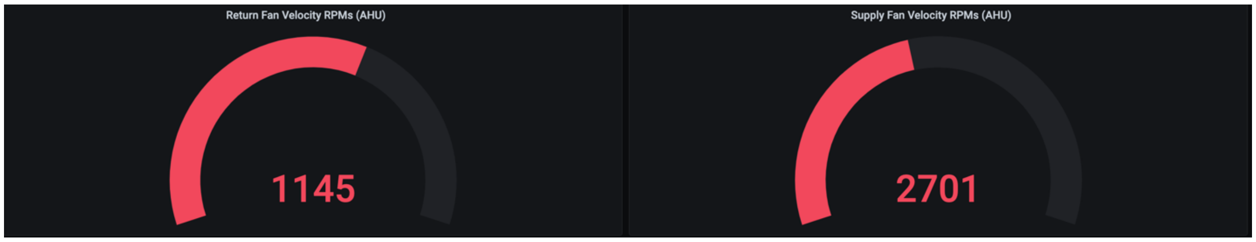

Return Fan Velocity RPMs (AHU)
1145
Supply Fan Velocity RPMs (AHU)
2701

Figure 5: Angular speed (in RPMs) of return and supply air fan inside the AHU.

Next, we present measurements for some other relevant sensors. For the following plots we decided to display such information as line plots with time on the x-axis, thus presenting them as time series. Using such visualizations makes it very easy to identify sudden peaks/drops in the data as well as easily identifying trends in the behavior of the system. These kinds of visualizations are well suited when we try to detect anomalous behaviors in the system.

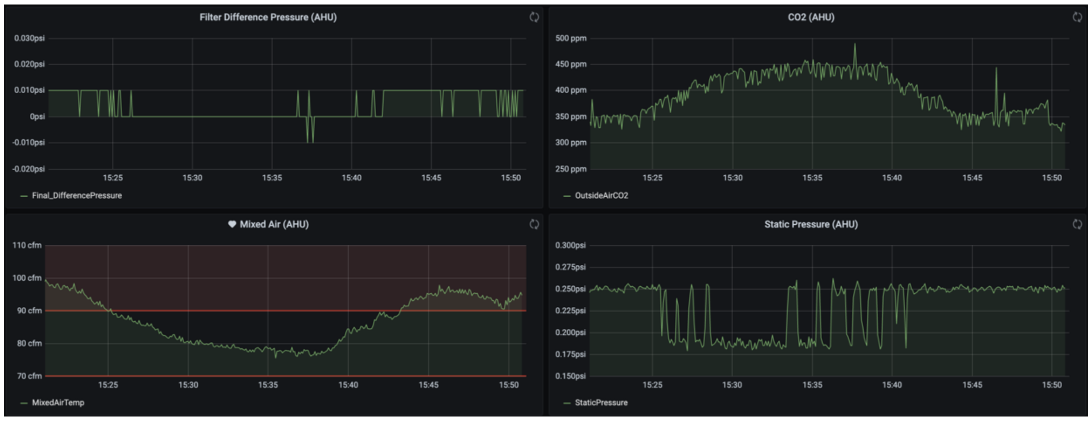
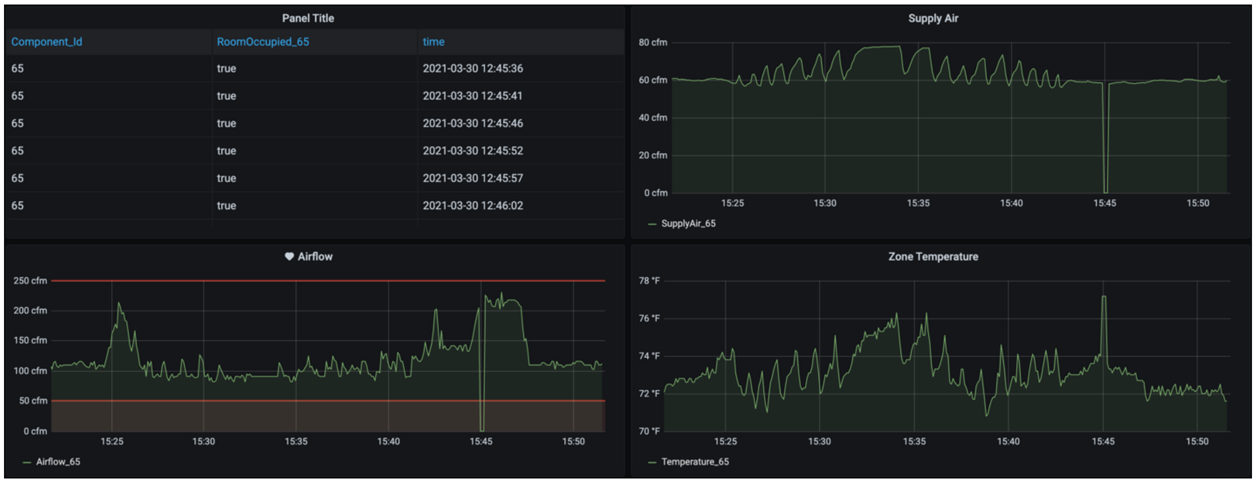


Figure 7: Timeseries measurements of some of the sensors within the AHU.

Finally, we present plots for the most relevant sensors installed inside a thermafuser. The following plots show time-series plots for airflow, temperature and supply air of one of the thermafusers of Octank's HVAC. Furthermore, for the airflow plot we set up operational limits and send alerts whenever these limits have been exceeded for the last 5 minutes. This is possible due to Grafana's built in alert system which can easily be integrated with Amazon's Simple Notification Service (SNS).



Component_Id	RoomOccupied_65	time
65	true	2021-03-30 12:45:36
65	true	2021-03-30 12:45:41
65	true	2021-03-30 12:45:46
65	true	2021-03-30 12:45:52
65	true	2021-03-30 12:45:57
65	true	2021-03-30 12:46:02

Figure 8: Time-series plots for supply air, airflow and zone temperature of a thermafuser within the HVAC system.

Anomaly detection module and notifications system

As mentioned earlier, our proposed solution can detect anomalies happening in the thermafusers. A custom model was trained using Amazon Sagemaker, a fully-managed service that enables data scientists and developers to quickly and easily build, train, and deploy machine learning models at any scale.

The anomaly detection works as follows:

1. Sensor data is pulled from the system at configurable time periods
2. Data is transformed using the rolling window approach
3. Data is passed into a custom anomaly detection model
4. The model returns whether the records are anomalous or not
5. Based on the number or anomalies detected over the time, alerts are sent out via email

This process is performed for every sensor at configurable time intervals.

Rolling (moving) window data transformation

As was observed in Figures 7 and 8, the data sensor data coming from the HVAC are indeed time series. Consider the following plot.

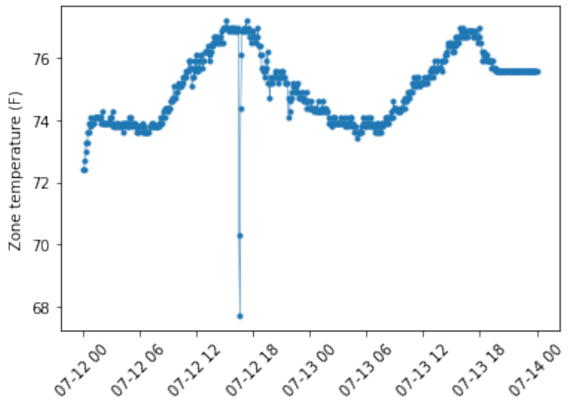


Figure 9: Time series plot of the zone temperature measured at a thermafuser.

As can be observed some points move drastically in a very short period of time. Take for instance the large dip in the sensor readings just before 07-12 at 18 hours. This could be due to wrong sensor readings or due to an anomaly going on in the system. When looking for anomalies in time series it is often useful to look for trends rather than looking at individual points in time.

Moving average smoothing is a naive and effective technique in time series forecasting. Smoothing is a technique applied to time series to remove the fine-grained variation between time steps thus allowing us to look at the overall trend instead of focusing our attention in individual readings. Consider now the following plot.

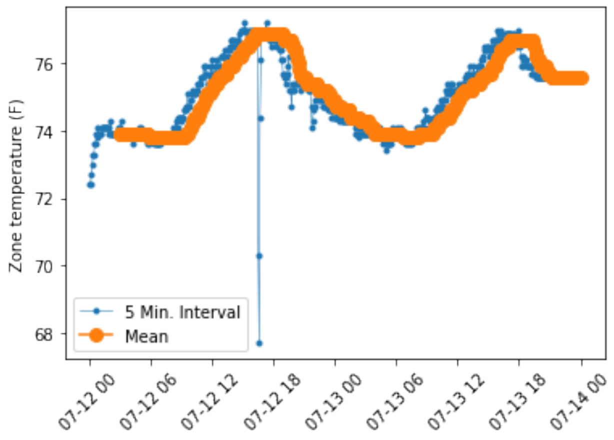


Figure 10: Time series plot of the zone temperature measured at a thermafuser (blue) and the moving window over 36 samples (orange).

By smoothing the data, we get a clearer view of the overall trend of the data. The hope of smoothing is to remove noise and better expose the signal of the underlying causal processes. Moving averages are a simple and common type of smoothing used in time series analysis and time series forecasting.

Calculating a moving average involves creating a new series where the values are comprised of the average of raw observations in the original time series. A moving average requires that we specify a window size called the window width. This defines the number of raw observations used to calculate the moving average value.

The “moving” part in the moving average refers to the fact that the window defined by the window width is slid along the time series to calculate the average values in the new series. Take a look at Figure 11. Here we present how to create several moving windows of size “m” from a sample of size “T” and a stride of 1.

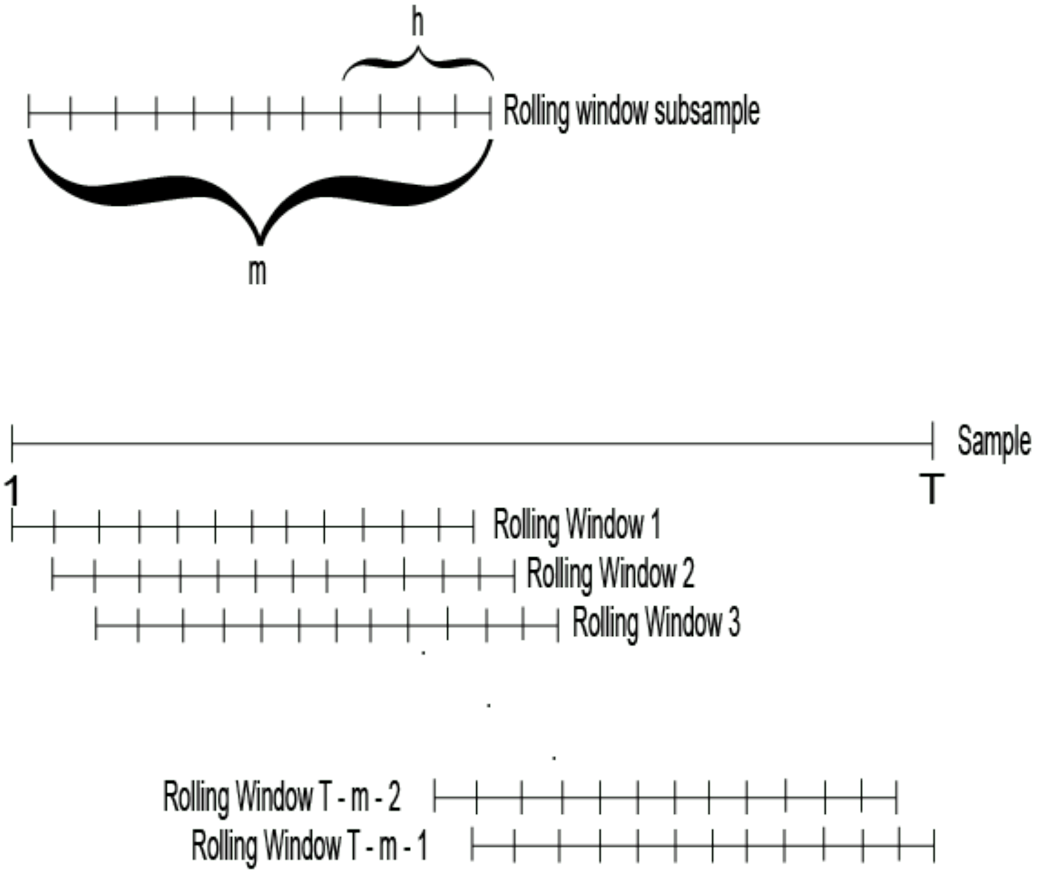


Figure 11: Illustration of a Rolling window from a sample of size T and a window size of m.

For this prototype we are calculating a moving average with a window of 12 observations (1 hour) and a stride of 1. These averaged windows are then passed to an isolation forest classifier.

An isolation forest is an unsupervised algorithm for detecting outliers within a data set. In a very general way, an isolation forest works by creating and ensemble of decision trees, each decision tree then partitions the data randomly. The intuition behind the anomaly detection is that data points that can be isolated very early in the process are very likely outliers (anomalies) in the data. For further details about isolations forests please see.

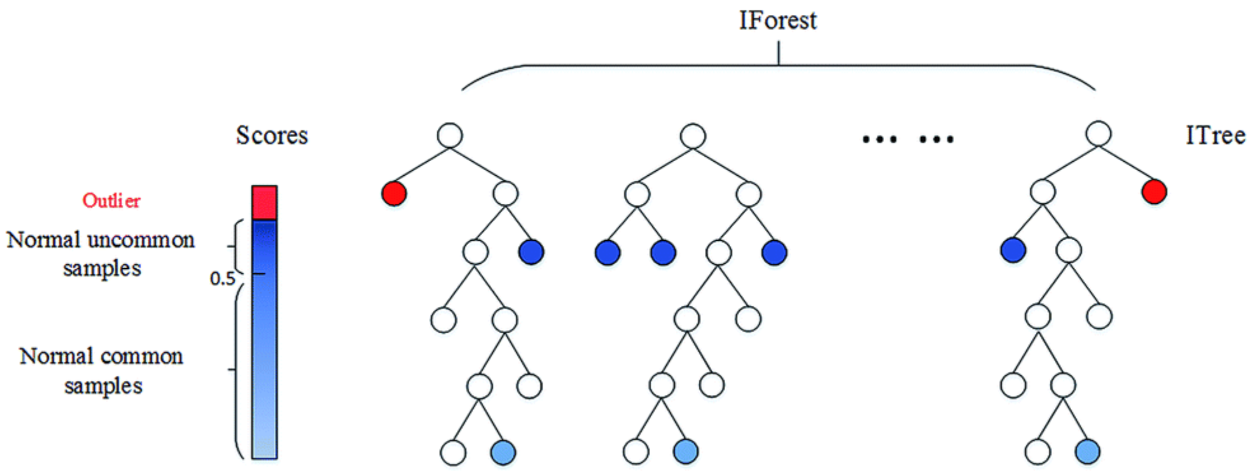


Figure 11: An isolation forest creates an ensemble of trees. Each tree tries to isolate the records, records which can be easily isolated i.e., early in the process, are considered outliers in the data.

Solution architecture

Next, we present the architecture diagram of the solution.

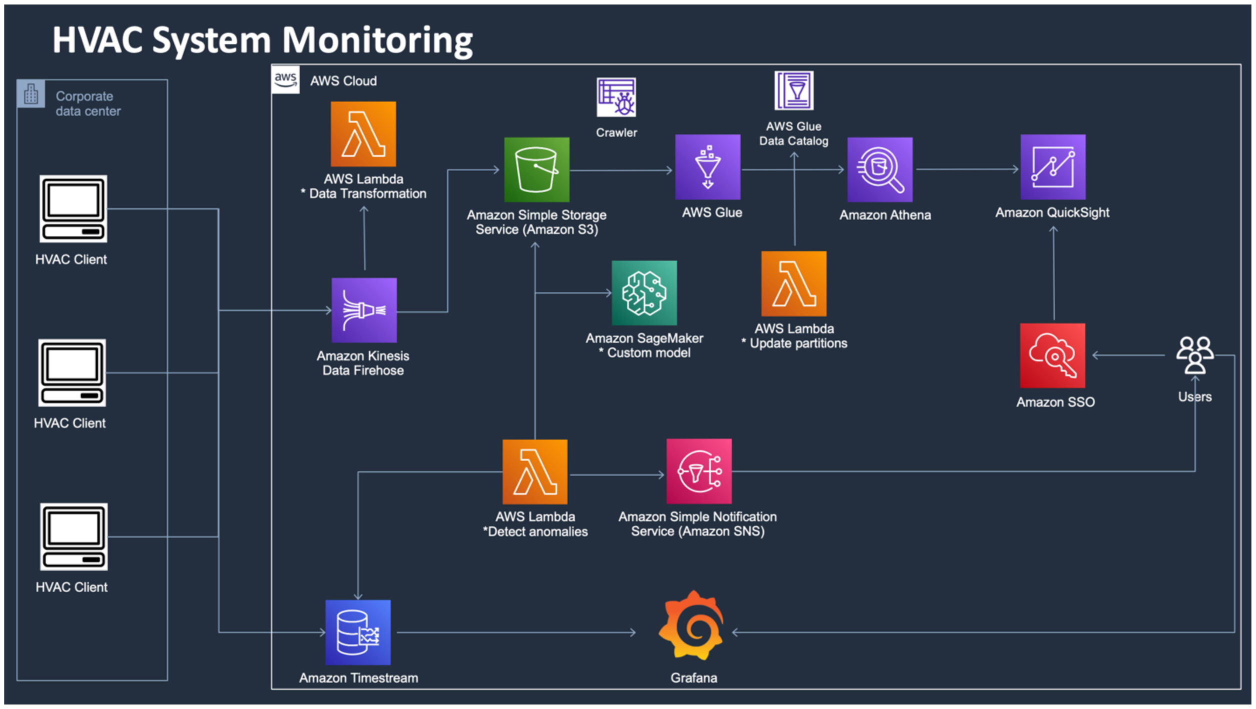


Figure 11: System's architecture

As can be observed in Figure 11 the sensor readings follow two main flows. In the upper flow the data is ingested through Amazon Kinesis Data Firehose streams, there is one stream per HVAC component. The name of the streams complies with the following nomenclature:

{component_name}{integer}-{year}{month}{day}

where year, month and day specify the date of creation of the stream. The following is a list of all the available streams in this prototype:

- ahu1-2021031
- damper1-2021031
- filter1-2021031
- fan1-2021031
- sav1-2021031
- vav1-2021031
- vfd1-2021031
- hec1-2021031
- thermafuser1-2021031
- anomalies1-2021031

Amazon Kinesis Data Firehose is a fully managed service that automatically provisions, manages and scales compute, memory, and network resources required to process and load your streaming data. Once set up, Kinesis Data Firehose loads data streams into your destinations continuously as they arrive. A field based on the timestamp of the record is extracted using a Lambda function and then the data delivered to Amazon S3, an object storage service that offers industry-leading scalability, data availability, security, and performance. There is one bucket (folder) per component. The S3 buckets are named using the following nomenclature:

octank-{region}-hvac-{component}

The following buckets were created for this prototype:

- octank-america-hvac-ahu
- octank-america-hvac-damper
- octank-america-hvac-fan
- octank-america-hvac-filter
- octank-america-hvac-hec
- octank-america-hvac-sav
- octank-america-hvac-vav
- octank-america-hvac-thermafuser
- octank-america-hvac-vfd
- octank-america-hvac-anomalies

There is an additional bucket called octank-america-component-relationships that stores the relationships between the components in the HVAC system.

It is important to highlight that we have enabled data compression and partitioning to make querying the data more efficient. The data is compressed using the apache parquet columnar format while partitioning follows the next nomenclature:

{Year}/{Month}/{Day}/{Hour}

Data is then queried using Amazon Athena, an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run. Nevertheless, for Athena to be able to access the data within the S3 buckets we need to define a schema for the data. For such task we leverage AWS Glue Data Catalog which is a metadata store that lets us store, annotate, and share metadata in the AWS Cloud. The data catalogs are creating using AWS Glue Crawlers. It is important to mention that partitions are not updated automatically by AWS Glue or Athena, therefore we make use of the lambda function “Update partitions” to update them every time a new partition is created.

Finally, data is visualized using Amazon Quicksight as explained in section …

On the bottom flow the data is directly sent from the client computers to Amazon Timestream, a fast, scalable, and serverless time series database service for IoT and operational applications that makes it easy to store and analyze trillions of events per day up to 1,000 times faster and at as little as 1/10th the cost of relational databases. We have created a database called "octank-america-hvac" which contains the following tables:

- ahu_readings
- damper_readings
- fan_readings
- filter_readings
- hec_readings
- sav_readings
- vav_readings
- vfd_readings
- thermafuser_readings

To make the use of Timestream cost-efficient we store the data for the last 7 days in memory and then move it to magnetic storage for up to 30 days.

The data stored in timestream is the presented in real-time through Grafana as explained in Section …

Additionally, the data for the last 10 minutes of thermafuser readings is retrieved from Timestream every 10 minutes, this data is the preprocessed, passed to an inference endpoint and postprocessed using the lambda function "Detect anomalies".

In case more than 50 anomalies have been detected in the thermafuser for the evaluation period the Lambda function sends a notification to the "ThermafuserAlarm" topic in the Simple Notification Service, Amazon publish/subscribe like service for sending and receiving notifications. Depending on how SNS is configured it can send the alerts to a specific user's email and/or phone number via SMS.

Finally, the anomaly detection model is described in Section … and the user may find the code and all the data transformations in the GitHub repository under …

It is important to highlight that the solution has SSO enabled so any user with the right permissions can access the solution using its corporate credentials.

Deployment

Walk through the next sections, in order, to deploy the application in your AWS account:

1. Prerequisites

1. Configure the AWS Credentials in your environment. Refer to *Configuration and credential file settings*.

2. Download and install AWS SAM CLI. Refer to *Installing the AWS SAM CLI*.

2. Deploy the AWS resources

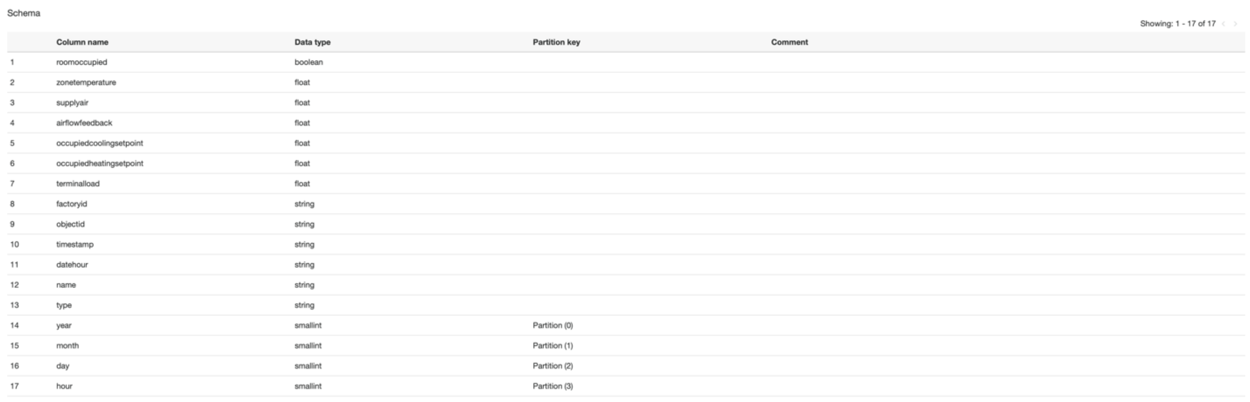
The following instructions assume we will create the flow for a Thermafuser. Apply the same instructions when creating a flow for any other HVAC component

1. Create S3 buckets

    1. On the AWS Console navigate to S3
    2. Click create bucket
    3. Name the bucket (octank-america-hvac-thermafuser)
    4. Enable the default encryption and use the default (SSE-S3) encryption key to enable single sign on encryption

2. Create AWS Glue data catalogs for data compression

    1. On the AWS Console navigate to AWS Glue
    2. If it doesn't exist create a new database called HVAC
    3. Select the HVAC database and go to "Tables in HVAC"
    4. Click on the "Add Tables Button" and select "Add Table Manually"
    5. Enter the name of the table (thermafuser_reading_raw)
    6. Select the HVAC database
    7. Select S3 as the source
    8. Select "Specified Path in my Account" and enter the path of the S3 bucket (s3://octank-america-hvac-thermafuser/)
    9. Choose the JSON data format
    10. Create the following schema (this depends on the component)


Schema

Showing: 1 - 17 of 17

	Column name	Data type	Partition key	Comment
1	roomoccupied	boolean		
2	zonetemperature	float		
3	supplyair	float		
4	airflowfeedback	float		
5	occupiedcoolingsetpoint	float		
6	occupiedheatingsetpoint	float		
7	terminalload	float		
8	factoryid	string		
9	objectid	string		
10	timestamp	string		
11	datehour	string		
12	name	string		
13	type	string		
14	year	smallint	Partition (0)	
15	month	smallint	Partition (1)	
16	day	smallint	Partition (2)	
17	hour	smallint	Partition (3)	

11. Add the combination of year and month as an index
12. Click Finish

3. Create the Lambda Functions. These instructions work for the three functions used in the prototype

    1. On the AWS console navigate to Lambda
    2. Click on the “Create Function” button
    3. Select “Author from scratch”
    4. Type the function name accordingly
    5. Select Python 3.8 as the runtime
    6. Click create function
    7. Navigate to the Code tab and double click on the lambda_function.py file
    8. Replace the code in the code editor window by the code of the corresponding function. The code for the functions can be located in the GitHub project under the /dev/python/lambdas/ folder.
    9. Click on the Deploy button

4. Create Kinesis Firehose streams

    1. On the AWS Console navigate to Kinesis Data Firehose
    2. Select “Create delivery stream”
    3. Enter the name of the delivery stream “thermafuser-20210310”
    4. Choose “Direct Put or other sources” as the Source
    5. Enable server-side encryption for source records in delivery stream
    6. Enable “Data Transformation” and select the “extractPartitionKeys” function that you created in Step 3
    7. Enable “Record format conversion”
    8. Select Apache Parquet as Output format
    9. Select US Oregon as the AWS Glue Region

10. Select “hvac” as the AWS Glue database
11. Choose the table named “thermafuser_reading_raw” that you created in step 2
12. Choose the S3 bucket named “octank-america-hvac-thermafuser” that you created in step 1 as the destination S3 bucket
13. Enable S3 encryption and use the default key
14. Click the “Create delivery stream button”

5. Configure AWS Glue Crawlers

    1. Navigate to AWS Glue from the AWS Console
    2. Click the button “Add Crawler”
    3. Enter the name of the crawler (thermafuser_crawler)
    4. Leave the default options and click Next
    5. Leave the default options and use the path “s3://octank-america-hvac-thermafuser” as the S3 bucket path to crawl
    6. Click Next and in the following screen create a new IAM Role
    7. Leave the “Run On demand” option selected and click Next
    8. Choose “hvac” as the destination database
    9. Add a prefix like “readings-octank”
    10. Click Finish
    11. Run your crawler and wait for a couple of minutes until the operation is complete

6. Setup Quicksight

    1. Navigate to AWS Quicksight from the AWS Console
    2. If this is the first time you open Quicksight follow the instructions to create your Quicksight account
    3. Click on the name of your user in the top right corner
    4. Click on Manage Quicksight
    5. On the menu on the left click on “Security & Permissions”
    6. In the “Quicksight access to AWS Services” click on “Add or remove”
    7. Check the Amazon Athena checkbox
    8. Click “Update”
    9. Navigate back to Quicksight main screen
    10. On the menu on the left navigate to “Datasets”
    11. Click on the “New Dataset” button
    12. Choose the Athena option
    13. In the name of the data source field enter “thermafuser_readings” and click “Next”
    14. Choose AWSDataCatalog as the Catalog and “hvac” as the Database
    15. Select the table named “readings_octank_america_hvac_thermafuser” (this is the table that was created by the crawler and may have a different name)
    16. Click “Select”. Your table is now can now be queried by Quicksight

7. Create Timestream DB and Tables

    1. Navigate to Timestream from the AWS Console
    2. Click on the “Create Database” button
    3. Leave the default values and enter the name “octank-america-hvac”. Click on the “Create Database” button
    4. Wait for the database to be created
    5. On the right-side menu click on the “Tables button”
    6. Click on the “Create Table” button
    7. Choose the “octank-america-hvac” as the database and enter the name “thermafuser_readings” as the name of the table
    8. Choose 7 days in the “Memory store retention” and 30 days in the “Magnetic Store retention”
    9. Click “Create Table”
    10. Your table is now created

8. Create Sagemaker workspace

    1. Navigate to Sagemaker from the AWS Console
    2. On the left side menu expand “Notebook” and then click on “Notebook instances”
    3. Click on the “Create Notebook” instance
    4. Leave the default settings and enter the name “octank-anomalyDetection” for the name of the notebook instance
    5. Wait for your instance to be created
    6. Once your instance is created click on the “Open Jupyter” button
    7. Replace the contents of the Jupyter notebook with the contents found in the GitHub folder under /dev/python/notebooks/anomaly_detection

9. Create and deploy custom anomaly detection model

    1. Navigate to the “octank-anomalyDetection” Jupyter notebook
    2. Open the “Thermafuser_anomaly_detection_scikit.ipynb” notebook
    3. On the notebook menu click on “Kernel” and the select “Restart and run all”
    4. The notebook takes care of all the steps for training and deploying the anomaly detection model

10. Configure “Anomaly detect” Lambda function

    1. Navigate to your Lambda function for performing anomaly detection (you should have created this one in step 3)
    2. In the function overview click on the “Add trigger” button
    3. Select EventBridge (CloudWatch events) in the dropdown menu for selecting a trigger
    4. Select create a new rule

5. Enter the rule name "ThermafuserAnomalyDetect"
6. Enter a rule description
7. In the "schedule expression" text box enter: rate(10 minutes) and click "Add"

11. Setup Amazon SNS topic and subscribe to it

    1. Navigate to Simple Notification System (SNS) from the AWS Console
    2. On the left side menu select "Topics"
    3. Click on the "Create Topic" button
    4. Leave the default values and name the topic "ThermafuserAlarm1"
    5. Click on the "Create topic" button
    6. Navigate to your newly created topic and click on its name
    7. Click on the "Create subscription" button
    8. Choose your topic ARN on the dropdown menu
    9. For protocol select Email on the dropdown menu
    10. Enter a desired email address on the "endpoint" text field
    11. Click on "Create subscription"
    12. The subscription is not active until the subscriber confirms it.

12. Create a Grafana workspace

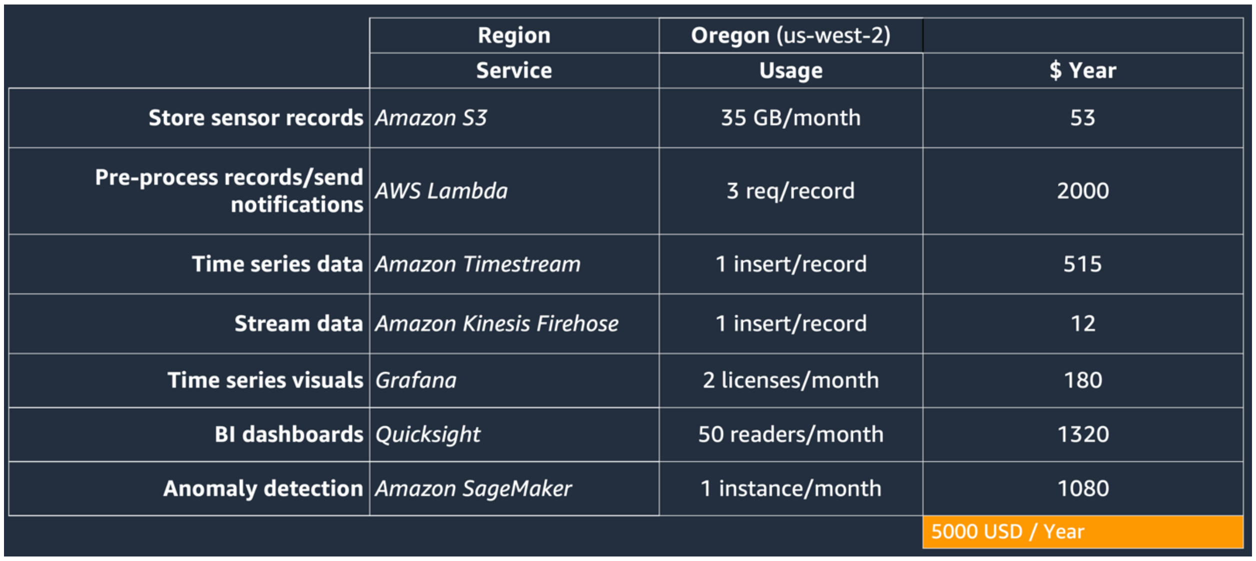
    1. Navigate to Managed Grafana Service from the AWS Console
    2. Click on the "Create workspace" button
    3. Wait for your workspace to be created
    4. Once your workspace is created navigate to your Grafana workspace by clicking in the "Grafana Workspace URL" link
    5. Once in your Grafana workspace go to the "Configuration" section (the little gear icon at the left menu)
    6. In the Configuration section click on "Add data source"
    7. Select the "Amazon Timestream" option
    8. Leave the default options and change the "Default Region" to us-west-2 (Assuming your architecture is being deployed in the Oregon region)
    9. Click on "Save & Test" button. You should see a message indicating the connection was successful

13. Configure Single Sign On according to your setup. For a tutorial on how to set up SSO using Okta please read https://aws.amazon.com/blogs/big-data/federate-amazon-quicksight-access-with-okta/

Cost estimation

Below there's an approximate cost estimation based on the following assumptions:

1) Assuming data is recorded every 5 minutes
2) Assuming 35 GB of data per month: Largest object is 1KB (5 KB) * 800 * 12 * 24 * 30 = 34.5 GB/Month.
3) Configuring lifecycle policies
4) 7 days of data in-memory and 1 month of magnetic storage



	Region	**Oregon** (us-west-2)	
	Service	**Usage**	**$ Year**
Store sensor records	*Amazon S3*	35 GB/month	53
Pre-process records/send notifications	*AWS Lambda*	3 req/record	2000
Time series data	*Amazon Timestream*	1 insert/record	515
Stream data	*Amazon Kinesis Firehose*	1 insert/record	12
Time series visuals	*Grafana*	2 licenses/month	180
BI dashboards	*Quicksight*	50 readers/month	1320
Anomaly detection	*Amazon SageMaker*	1 instance/month	1080
			5000 USD / Year

Please keep in mind that this calculation is just an approximated and implementation costs may vary on practice. To create an estimation using Free Tier and Tier pricing, please use data from chart "Input Data" and please refer to **AWS Pricing Calculator** (https://calculator.aws/).

Improvements

Due to the limited timeframe of the prototype development, it doesn’t contain many features that may be valuable to users. It also could be improved by following all the best practices needed for operating reliable, secure, efficient, and cost-effective systems in the cloud. This section details some areas that should be improved to achieve a production solution based on the prototype. Most of the points are aligned with the AWS Well-Architected Framework.

Bibliography