# **[Project 1]**

# **IoT analytics using predictive models and integration with edge devices to send commands based on prediction outcomes**

Internet of Things (IoT) has evolved tremendously in all spheres of our lives like Industrial applications, Social interactions, Remote management of facilities and equipment to name a few. In general application areas, IoT data collected by Sensors can be used for monitoring as well as predicting the outcomes. If any deviation from the norm is detected, corrective action can be prescribed either manually or by an automated process. Such actions may come out of rule-based anomaly detection or a Statistical Change point detection or a Predictive model that predicts a faulty condition ahead of time. This approach goes a long way in implementing Predictive maintenance which is a more prudent approach than Scheduled Preventive maintenance which is periodic in nature.

The end to end process steps for applying Analytics on IoT data are listed below:

1. Collect IoT data from the sensor
2. Changepoint detection using IoT Sensor data. Refer code pattern [Detect change points in IoT sensor data](https://developer.ibm.com/code/patterns/detect-change-points-in-iot-sensor-data/) for more details.
3. Predicting equipment failure using IoT Sensor data. Refer code pattern [Predict equipment failure using IoT sensor data for](https://developer.ibm.com/code/patterns/predict-equipment-failure-using-iot-sensor-data/) more details.
4. Sending decisions based on Analytics insights to the edge for automated action

This IBM Code Pattern is a composite pattern that demonstrates the building of a complete IoT analytics solution. When you complete this code pattern, you will learn how to:

* Send events from an edge device (we use Raspberry Pi for demonstration) to the Watson IoT Platform
* Store the events in a DB2 database on IBM Cloud
* Invoke a predictive model on Watson Studio for IoT events using the below code patterns:
  + [Predict equipment failure using IoT sensor data](https://developer.ibm.com/code/patterns/predict-equipment-failure-using-iot-sensor-data/)
  + [Orchestrate data science workflows using Node-RED](https://github.com/IBM/node-red-dsx-workflow)
* Send a command back to the edge (we use Raspberry Pi for demonstration) based on the outcome of the predictive model

This pattern uses a sample equipment sensors data. This data is sent to the Watson IoT platform and stored in a DB2 database. A predictive model is built using the data in the DB2 database. The predictive model then takes the sensor events from Watson IoT platform as input and returns the state of the equipment as Running or Failing. If the equipment is failing, then a shutdown command is sent back to the edge device which is a Raspberry Pi.

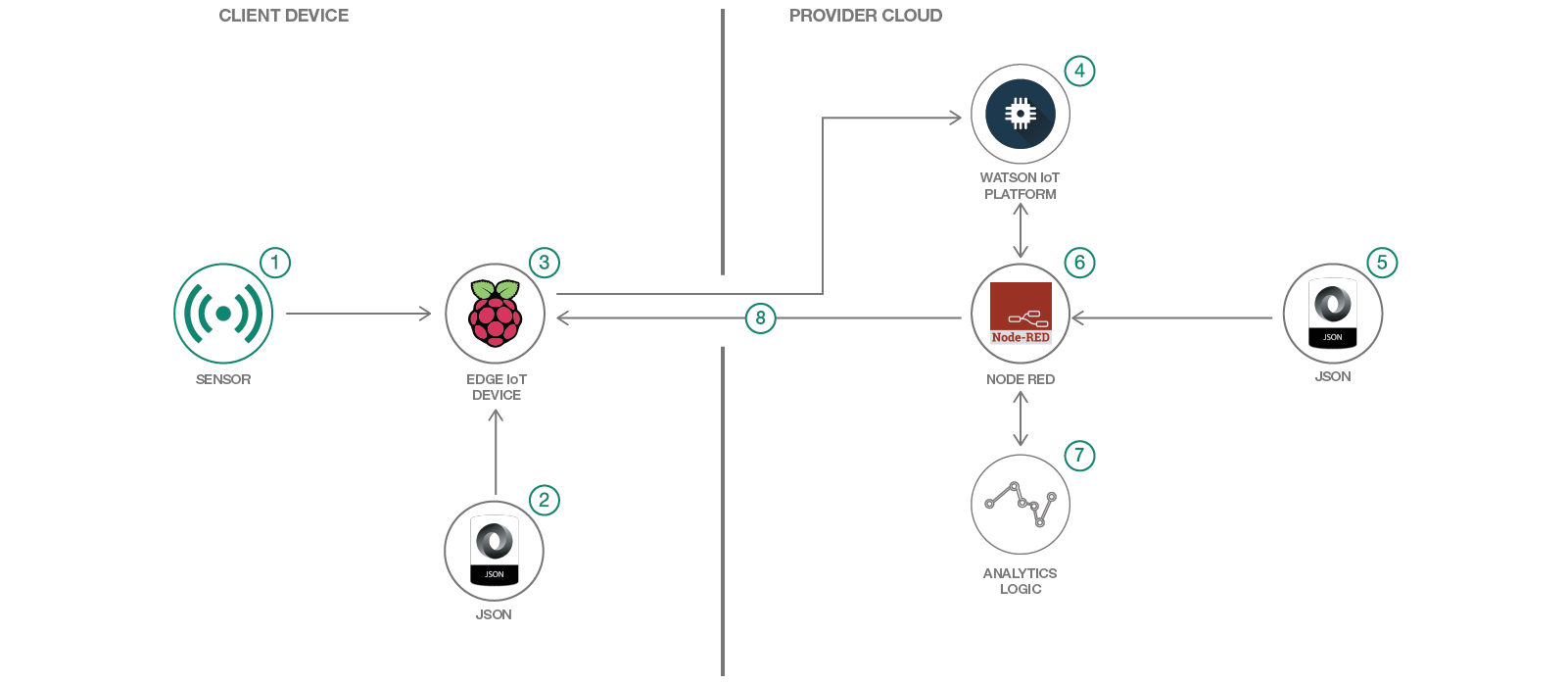
This pattern uses [Node-RED](https://nodered.org/) at both device and cloud for building the solution:

* Implementing device client on Raspberry Pi to send events to Watson IoT platform
* Consuming events from Watson IoT platform on IBM Cloud and storing the events in a DB2 database
* Invoke the predictive model on Watson Studio and get a response back for an IoT event
* Send a command back to the Raspberry Pi through the Watson IoT platform

## **Prerequisites:**

* [Connect Raspberry Pi to the network and Note IP address for accessing the Pi](https://www.raspberrypi.org/documentation/)
* [Running Node-RED on Raspberry Pi](https://nodered.org/docs/hardware/raspberrypi)

## **Flow**

****

1. The Raspberry Pi gets events from the sensors. In the absence of sensors, the sensor events are read from a file.
2. The Node-RED flows are invoked on the Raspberry Pi.
3. The sensor events are sent to the Watson IoT platform.
4. The Watson IoT platform receives the events and sends them to all subscribing applications.
5. The Node-RED flows on IBM Cloud are triggered. The sensor events are received and stored in a database.
6. The predictive model on Watson Studio is triggered. The outcome of the model execution is sent back to the Node-RED through web sockets.
7. Based on the outcome, the Node-RED flow sends a command with the action to be taken to the edge device(Raspberry Pi) through the Watson IoT platform
8. The Node-RED flow on Raspberry Pi receives the command

## **Included Components**

* [IBM Cloud](https://console.bluemix.net/catalog/): IBM's innovative cloud computing platform or IBM Cloud (formerly Bluemix) combines  
  platform as a service (PaaS) with infrastructure as a service (IaaS) and includes a rich catalog of cloud services that can be easily integrated with PaaS and IaaS to build business applications rapidly.
* [IBM Watson IoT Platform](https://internetofthings.ibmcloud.com/): IBM Watson™ IoT Platform for IBM Cloud gives you a versatile  
  toolkit that includes gateway devices, device management, and powerful application access. By using  
  Watson IoT Platform, you can collect connected device data and perform analytics on real-time data  
  from your organization.
* [IBM Watson Studio](https://www.ibm.com/bs-en/marketplace/data-science-experience): Analyze data using Python, Jupyter Notebook  
  and RStudio in a configured, collaborative environment that includes IBM value-adds, such as managed Spark.
* [DB2 Warehouse](https://console.bluemix.net/catalog/services/db2-warehouse): IBM Db2 Warehouse on Cloud is a fully-managed, enterprise-class, cloud data warehouse service.

## **Featured Technologies**

* [Analytics](https://developer.ibm.com/code/technologies/analytics?cm=IBMCode-_--_-featured_technologies-_-analytics): Finding patterns in data to derive information.
* [Data Science](https://developer.ibm.com/code/technologies/data-science?cm=IBMCode-_--_-featured_technologies-_-data-science): Systems and scientific methods to analyze structured and unstructured data in  
  order to extract knowledge and insights.
* [Internet of Things](https://en.wikipedia.org/wiki/Internet_of_things)

# 

# **[Project 2]**

# **Discovery customer sentiment from product reviews**

In this code pattern, we walk you through a working example of a web application that queries and manipulates data from the Watson Discovery Service. With the aid of a custom model built with Watson Knowledge studio, the data will have additional enrichments that will provide improved insights for user analysis.

This web app contains multiple UI components that you can use as a starting point for developing your own Watson Discovery and Knowledge Studio service applications.

The main benefit of using the Watson Discovery Service is its powerful analytics engine that provides cognitive enrichment and insights into your data. This app provides examples of how to showcase these enrichments through the use of filters, lists, and graphs. The key enrichments that we will focus on are:

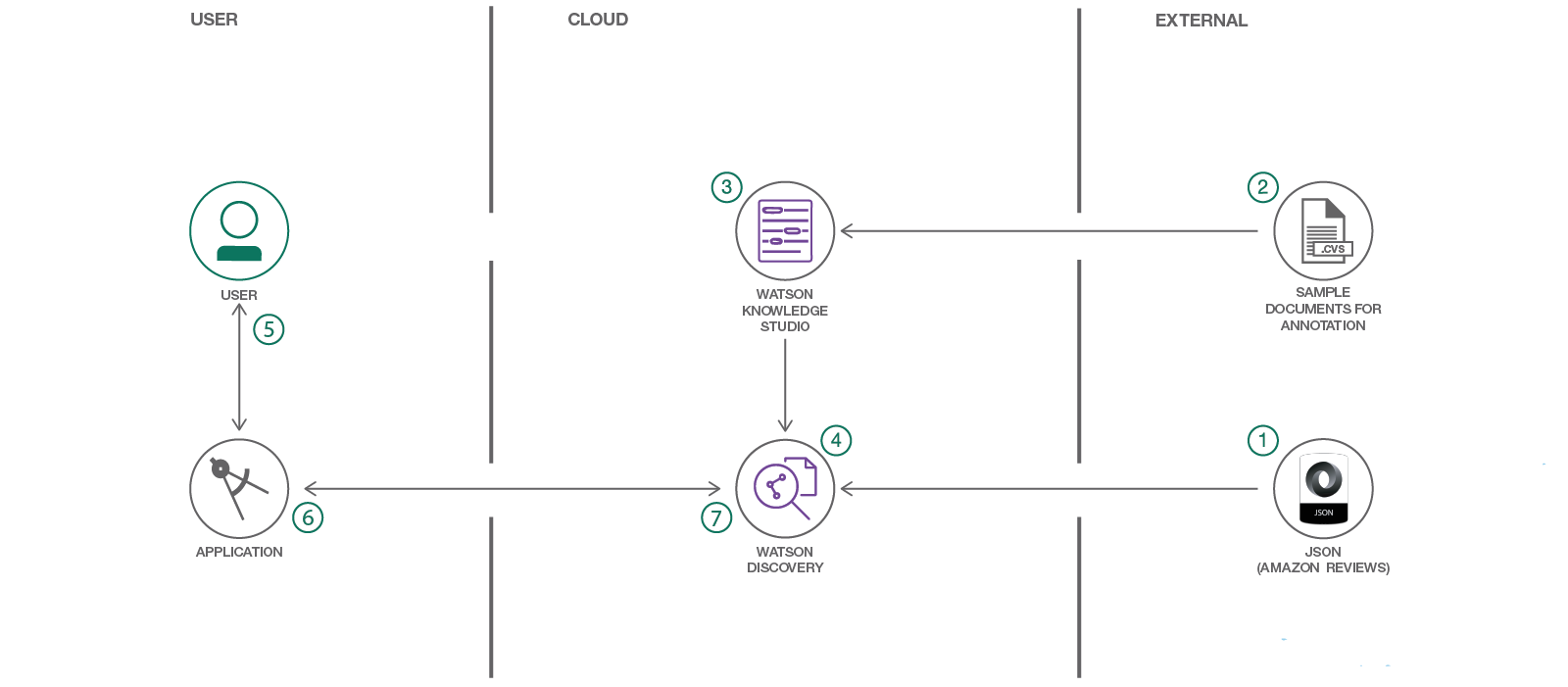
* Entities: people, companies, organizations, cities, and more.
* Categories: classification of the data into a hierarchy of categories up to 5 levels deep.
* Concepts: identified general concepts that aren't necessarily referenced in the data.
* Keywords: important topics typically used to index or search the data.
* Entity Types: the classification of the discovered entities, such as a person, location, or job title.
* Sentiment: the overall positive or negative sentiment of each document.

With Watson Knowledge Studio (WKS), a machine learning annotator can be trained to recognize mentions of the custom entity and relation types which can then be incorporated into the Discovery application enrichment process.

For this code pattern, we will be using data that contains food reviews from Amazon, see the [Kaggle dataset](https://www.kaggle.com/snap/amazon-fine-food-reviews) for further information.

When the reader has completed this code pattern, they will understand how to:

* Use Watson Knowledge Studio to create a custom annotator.
* Deploy a WKS model to Watson Discovery.
* Load and enrich data in the Watson Discovery Service.
* Query and manipulate data in the Watson Discovery Service.
* Create UI components to represent enriched data created by the Watson Discovery Service.
* Build a complete web app that utilizes popular JavaScript technologies to feature Watson Discovery Service data and enrichment.



## **Flow**

1. A sample set of review documents are loaded into WKS for annotation.
2. A WKS model is created.
3. The WKS model is applied to a Watson Discovery service instance.
4. The food review JSON files are added to the Discovery collection.
5. The user interacts with the backend server via the app UI. The frontend app UI uses React to render search results and can reuse all of the views that are used by the backend for server-side rendering. The frontend is using semantic-UI-react components and is responsive.
6. User input is processed and routed to the backend server, which is responsible for server-side rendering of the views to be displayed on the browser. The backend server is written using express and uses the express-react-views engine to render views written using React.
7. The backend server sends user requests to the Watson Discovery Service. It acts as a proxy server, forwarding queries from the frontend to the Watson Discovery Service API while keeping sensitive API keys concealed from the user.

NOTE: see [DEVELOPING.md](https://github.com/technoindianjr/watson-discovery-food-reviews/blob/master/DEVELOPING.md) for project structure.

## **Included components**

* [Watson Discovery](https://www.ibm.com/watson/services/discovery/): A cognitive search and content analytics engine for applications to identify patterns, trends, and actionable insights.
* [Watson Knowledge Studio](https://www.ibm.com/watson/services/knowledge-studio/): Teach Watson the language of your domain with custom models that identify entities and relationships unique to your industry, in unstructured text. Use the models in Watson Discovery, Watson Natural Language Understanding, and Watson Explorer.

# **[Project 3]**

# **Industrial Visual Analysis**

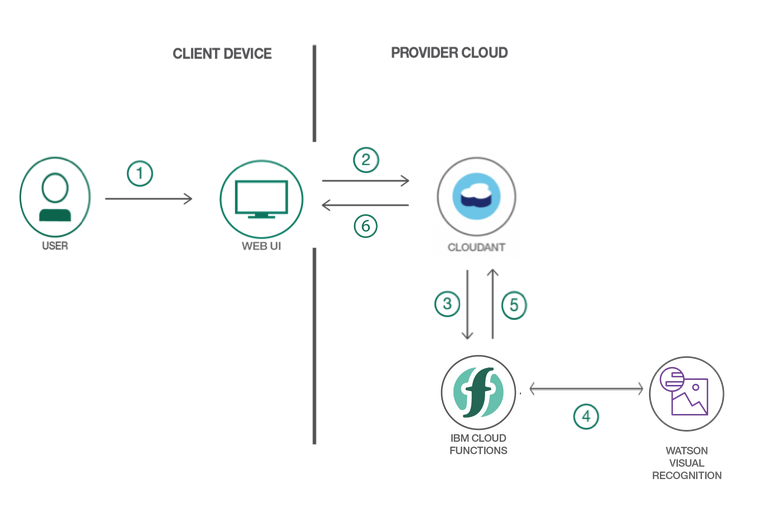
In this code pattern, we will identify industrial equipment for various damages upon visual inspection by using machine learning classification techniques. Using Watson Visual Recognition, we will analyze the image against a trained classifier to inspect oil and gas pipelines with six identifiers - Normal, Burst, Corrosion, Damaged Coating, Joint Failure, and Leak. For each image, we will provide a percent match with each of them, on how closely the image matches one of the damaged identifiers or the Normal identifier. This data can then be used to create a dashboard to the pipelines needing immediate attention to no attention.

The image's data is stored in a Cloudant database which makes it easier to connect remote devices (including drones) to capture images. The database can store different properties of the images like location and description. This code pattern demonstrates IBM Cloud Functions (OpenWhisk) to trigger microservice as an image is added to the Cloudant database. The microservice performs the Visual Recognition analysis and updates the Cloudant database with the analysis data.

When the reader has completed this code pattern, they will understand how to:

* Train Visual Recognition to classify images
* Configure Cloudant database to store and retrieve image data
* Set up IBM Cloud Functions to trigger Visual Recognition analysis and store result in Cloudant database
* Launch a web app to view a dashboard of the Visual Recognition analysis, and deploy to IBM Cloud

# **Architecture Flow**

****

1. User uploads the image through the web UI
2. The image data is sent to the Cloudant database
3. As the image is added into the database, the Cloud Functions triggers microservice
4. The microservice analyzes the image using the trained Watson Visual Recognition service
5. The analyzed data is fed back into the Cloudant database
6. The dashboard on the web UI displays the Visual Recognition analysis and images requiring attention

## **Included Components**

* [Visual Recognition](https://www.ibm.com/watson/services/visual-recognition/)
* [Cloudant](https://www.ibm.com/analytics/us/en/technology/cloud-data-services/cloudant/)
* [IBM Cloud Functions](https://console.bluemix.net/openwhisk)

# **[Project 4]**

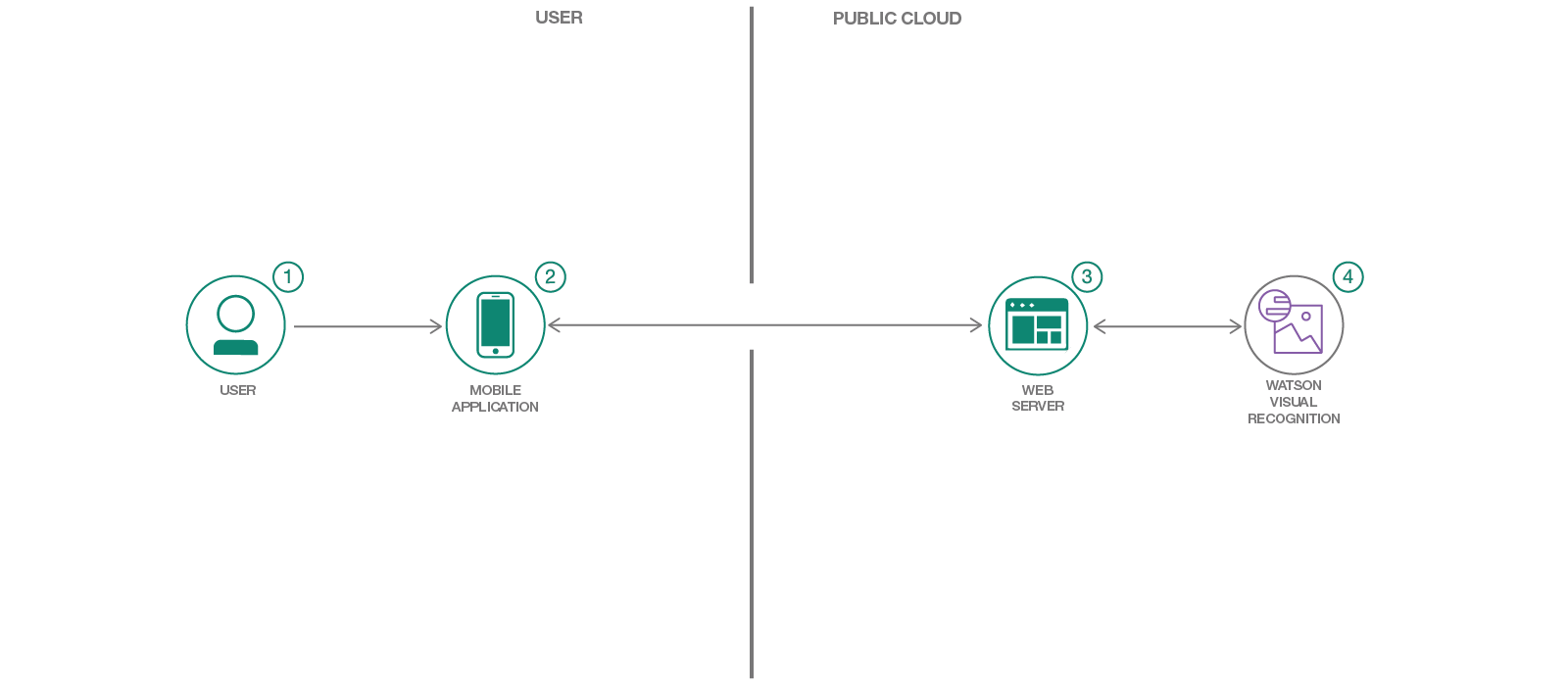
# **Create a custom Visual Recognition classifier for analyzing vehicle damage**

In this developer code pattern, we will create a mobile app using Apache Cordova, Node.js and Watson Visual Recognition. This mobile app sends pictures of auto and motorcycle accidents and issues to be analyzed by a server app, using Watson Visual Recognition.

The server application will use pictures of auto accidents and other incidents to train Watson Visual Recognition to identify various classes of issues, i.e. vandalism, broken windshield, motorcycle accident, or flat tire. A developer can leverage this to create their own custom Visual Recognition classifiers for their use cases.

When the reader has completed this Code Pattern, they will understand how to:

* Create a Node.js server that can utilize the Watson Visual Recognition service for classifying images.
* Have a server initialize a Visual Recognition custom classifier at startup.
* Create a Visual Recognition custom classifier in an application.
* Create an Android mobile application that can send pictures to a server app for classification using Visual Recognition.



## 

## 

## 

## **Flow**

1. The user interacts with the mobile app and captures an image.
2. The image on the mobile phone is passed to the server application running in the cloud.
3. The server sends the image to Watson Visual Recognition Service for analysis.
4. Visual Recognition service classifies the image and returns the information to the server.

## **Included components**

* [Watson Visual Recognition](https://www.ibm.com/watson/developercloud/visual-recognition.html): Visual Recognition understands the contents of images - tag images, find human faces, approximate age, and gender, and find similar images in a collection.

# 

# **[Project 5]**

# **Compose domain specific bots using an agent bot**

A domain-specific bot addresses queries related to a specific domain or topic. Some examples are - Travel bot(Travel related conversations), Weather bot (Weather related conversations).

If a user wants to have a cross-domain conversation, then the user will have to switch between the different bots.

There are scenarios where a user wants to have a conversation involving multiple domains. For example - When I want to travel to a place, I want to query for the weather and then book a cab or flight. I might have to end up switching between two bots, weather bot and travel bot. What if I could just have one interface bot which will redirect my messages to a specific bot and get answers to me?

Well, this code pattern showcases an implementation of this approach.

The solution here is to have an agent bot (or an interface bot) and a few other bots which can handle conversations for a specific domain - let's call these specific bots. The agent bot knows about the specific bots and also about which domain each of them can handle. When a user initiates conversation with agent bot, the agent bot will understand the intent of user query and it will redirect the user query to a specific bot. Subsequent requests from the user are redirected to specific bot. When the conversation with the specific bot is over or when the specific bot is not able to handle the request, the control is given back to agent bot which will then redirect the messages to appropriate bot.

This approach provides a seamless experience for users. It can be used by organisations which provide a host of services to its customers like financial services, tours and travel agencies, news agencies etc..

Advantages of this approach are:

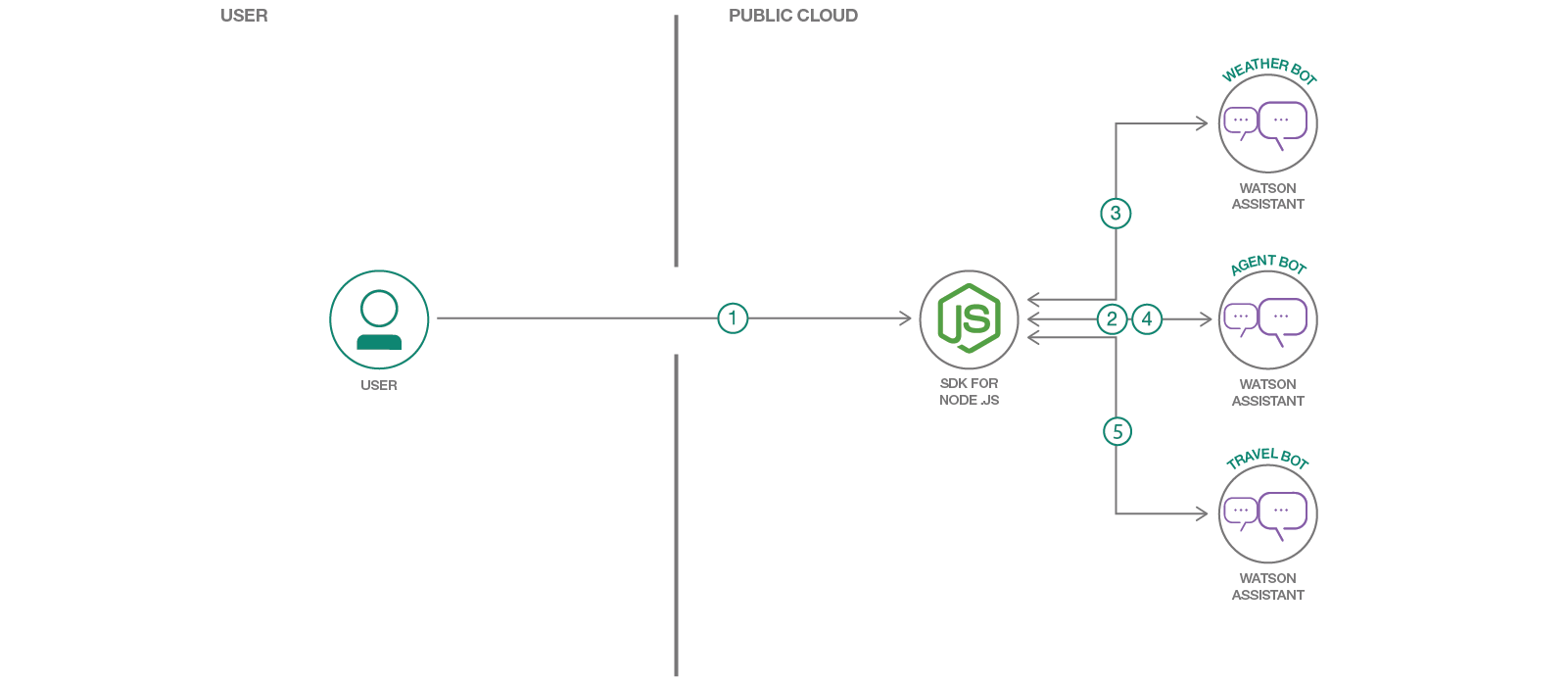
* Plug and play the bots
* Modular approach facilitates Bots composition
* Come up with new services by composing two or more Bots
* Easy to maintain, make changes, add/remove functionalities
* Easy to troubleshoot issues
* Transparent to user

In this code pattern, we will use Watson assistant Bot for building Bots and Node.js application as the orchestration layer.

When the reader has completed this Code Pattern, they will understand how to:

* How to configure a bot to make it Agent Bot
* How to configure a specific bot to return control to Agent Bot
* How to build an orchestration layer to stitch Agent Bot and specific Bots

## **Flow**

****

1. A user accesses web application and types in a message. Nodejs application, an orchestration later, sends user message to agent bot
2. Agent bot determines the intent of the message and responds with the specific bot details, to which the message needs to be redirected.
3. The node.js application sends a message to the specific bot (Weather Bot, in this case). Specific bot responds. continues between a user and specific bot.
4. When the conversation with the specific bot is over, a user message is then sent to agent bot to determine the intent.
5. The node.js application sends a message to the specific bot (Travel Bot, in this case). Specific bot responds. The conversation continues between a user and a specific bot.

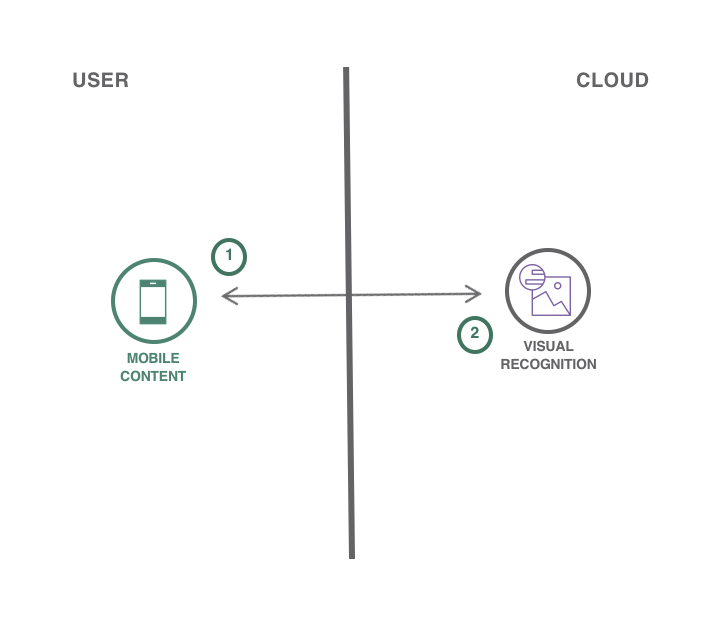
# **[Project 6]**

# **Create an Android application in Java which uses image recognition**

In this code pattern, you will create an Android app that showcases computer vision by labelling what the device's camera sees. You will provide a Visual Recognition service where you can either leverage a demo model or train your own custom model.

When you have completed this code pattern, you will understand how to:

* Customize Watson Visual Recognition for your unique use case
* View the labels related to a picture and the estimated accuracy of that label



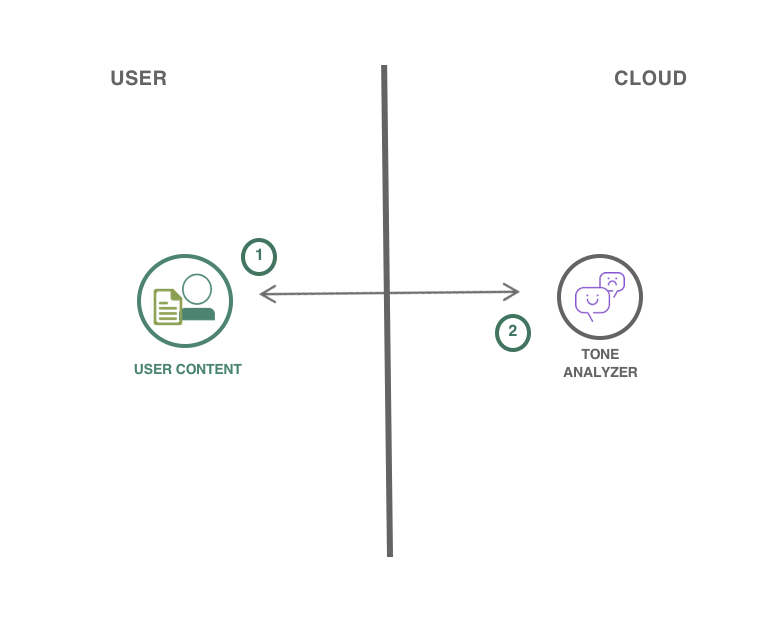
# **[Project 7]**

# **Create an Android application in Java which analyzes the emotion and tone of natural language**

In this code pattern, you will create an Android application using the Watson Tone Analyzer service. You will be able to analyze emotions and tones in what people write online, such as in tweets or reviews, and then predict whether they are happy, sad, confident, and more. Monitor customer service and support conversations so you can respond to your customers appropriately and at scale. Enable your chatbot to detect customer tones so you can build dialogue strategies to adjust the conversation accordingly.

When you have completed this code pattern, you will understand how to:

* Build a mobile application that uses Watson Tone Analyzer
* Provision and integrate additional IBM Cloud services
* Learn how to analyze emotions and tones in text



# **[Project 8] Data Science Process Pipeline in action to solve Employee Attrition Problem**

# This code pattern is a high-level overview of what to expect in a data science pipeline and the tools that can be used along the way. It starts from framing the business question, to building and deploying a data model. The pipeline is demonstrated through the employee attrition problem.

# Employees are the backbone of any organization. Its performance is heavily based on the quality of the employees and retaining them. With employee attrition, organizations are faced with a number of challenges:

# Expensive in terms of both money and time to train new employees

# Loss of experienced employees

# Impact on productivity

# Impact on profit

# The following solution is designed to help address the employee attrition problem. When the reader has completed this code pattern, they will understand:

# The Process involved in solving a data science problem

# How to create and use Watson Studio instance

# How to mitigate bias by transforming the original dataset through use of the AI Fairness 360 (AIF 360) toolkit

# How to build and deploy the model in Watson Studio using various tools

# The dataset used in the code pattern is supplied by [Kaggle](https://www.kaggle.com/) and contains HR analytics data of employees that stay and leave. The types of data include metrics such as education level, job satisfactions, and commute distance.

# 

# 

# 

# 

# 

# 

# The data is made available under the following license agreements:

### **Dataset License Details**

|  |  |  |
| --- | --- | --- |
| **Asset** | **License** | **Source Link** |
| [**Employee Attrition Data - Database License**](https://github.com/IBM/employee-attrition-aif360/blob/master/data/emp_attrition.csv) | [**Open Database License (ODbL)**](https://opendatacommons.org/licenses/odbl/1.0/) | [**Kaggle**](https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset/home) |
| [**Employee Attrition Data - Content License**](https://github.com/IBM/employee-attrition-aif360/blob/master/data/emp_attrition.csv) | [**Database Content license (DbCL)**](https://opendatacommons.org/licenses/dbcl/1.0/) | [**Kaggle**](https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset/home) |

## **Flow**

# **architecture**

# Create and login to the IBM Watson Studio.

# Upload the jupyter notebook and start running it.

# Notebook downloads the dataset and imports fairness toolkit (AIF 360) and Pygal data visualization library.

# Pandas is used for reading the data and perform initial data exploration.

# Matplotlib, Seaborn, Plotly, Bokeh and Pygal (from step-3) are used for visualizing the data.

# Scikit-Learn and AIF 360 (from step-3) are used for model development.

# Use the IBM Watson Machine Learning feature to deploy and access the model to generate employee attrition classification.

## **Included Components**

# [IBM Watson Studio](https://www.ibm.com/bs-en/marketplace/data-science-experience): Analyze data using RStudio, Jupyter, and Python in a configured, collaborative environment that includes IBM value-adds, such as managed Spark.

# [IBM Watson Machine Learning](https://cloud.ibm.com/catalog/services/machine-learning): a set of REST APIs to develop applications that make smarter decisions, solve tough problems, and improve user outcomes.

# [Jupyter Notebook](http://jupyter.org/): An open source web application that allows you to create and share documents that contain live code, equations, visualizations, and explanatory text.

# **[Project 9] Use Watson Studio and PyTorch to create a machine learning model to recognize handwritten digits**

## **Overview**

# Recognizing handwritten numbers is a piece of cake for humans, but it's a non-trivial task for machines. Nowadays, with the advancement of machine learning, people have made machines more and more capable of performing this task. We now have mobile banking apps that can scan checks in seconds and accounting software that can extract dollar amounts from thousands of contracts in minutes. If you are interested in knowing how this all works, please follow along with this code pattern as we take you through the steps to create a simple handwritten digit recognizer in Watson Studio with PyTorch.

# **What is Watson Studio?**

# [Watson Studio](https://dataplatform.cloud.ibm.com/) is an integrated environment for data scientists, developers and domain experts to collaboratively work with data to build, train and deploy models at scale. If you are new to Watson Studio, the best way to understand it is to [see it in action](https://medium.com/ibm-watson/ibm-watson-studio-in-10-videos-6acf96cc608a)

## **What is PyTorch?**

# [PyTorch](https://pytorch.org/) is a relatively new deep learning framework. Yet, it has begun to gain adoption especially among researchers and data scientists. The strength of PyTorch is its support of dynamic computational graph while most deep learning frameworks are based on static computational graph. In addition, its strong NumPy like GPU accelerated tensor computation has allowed Python developers to easily learn and build deep learning networks for GPUs and CPUs alike.

# In this code pattern, you will use Jupyter Notebook in Watson Studio and access preinstalled and optimized PyTorch environments through the Python client library of the [Watson Machine Learning](https://cloud.ibm.com/catalog/services/machine-learning) service, which has a set of REST APIs in its core that allows users to submit training jobs, monitor status, and store and deploy models.

# 

# 

# 

# 

# When you have completed this code pattern, you will understand how to:

# Create a project in Watson Studio and use Jupyter Notebooks in the project.

# Use the Python client of Cloud Object Storage to create buckets and upload data to buckets.

# Submit PyTorch training jobs to Watson Machine Learning service.

# Use the trained PyTorch model to predict handwritten digits from images.

## **Flow**

# **architecture**

# Log into IBM Watson Studio

# Run the Jupyter notebook in Watson Studio

# Use PyTorch to download and process the data

# Use Watson Machine Learning to train and deploy the model

# 

# **[Project 10]**

# **Live streaming of devices/sensor data using custom widgets within Cognos dashboard**

IBM Cognos Analytics integrated reporting, modelling, analysis, dashboards, stories, and event management so that you can understand your organization data, and make effective business decisions. Just take an example in the Automobile manufacturing unit, one of the key challenging tasks is to monitor the devices and taking an accurate decision of bringing them for planned or unplanned maintenance. From the traditional Cognos analytics (previous to 11.x version) dashboard, it was NOT possible to show the volatility and the non-volatile data in a single dashboard. Meaning, the real-time data display was not possible directly. Now with the latest features of Cognos Analytics 11.x, custom widgets can be built to display volatile data and custom widgets can be used through Extensions in Cognos Dashboards.

With the latest feature of Cognos Analytics 11.x Extensions, we have the ability to add and remove elements in the IBM Cognos Analytics user interface for a perspective. An extension is a zip file that contains spec.json, optional images and js folders. You can create extensions that add functions to the IBM Cognos Analytics user interface.

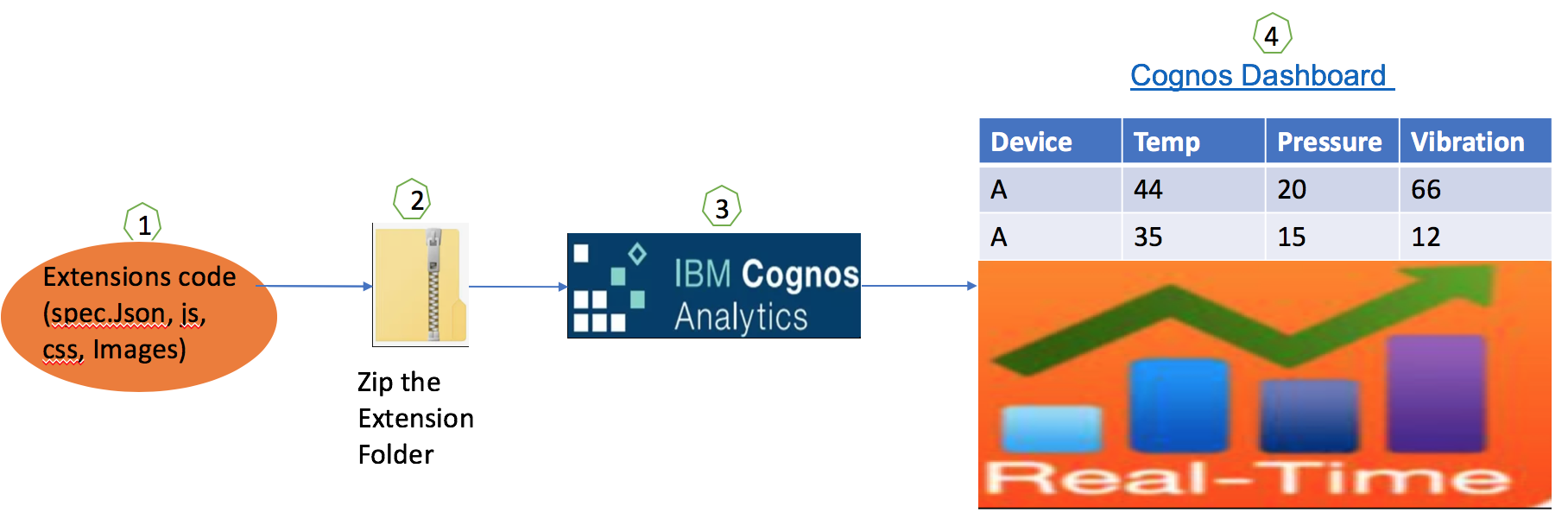
In Automobile manufacturing unit, a plant engineer's job is to continuously monitor the health of the device and keep them intact. It is quite a tedious job for him to monitor constantly those devices and take a call to bring the devices down for maintenance. When there is a corrective maintenance task, where equipment is repaired or replaced after wear, malfunction or break down, at that moment going for maintenance is a fair ask. But, if there is preventive/predictive maintenance required based on the previous break down activities, the system/models would have predicted for the device to bring down for maintenance. However, if you really look at the live (real-time) metrics of the devices, he would notice that devices health is absolutely fine and intact, and the plant engineer would not go to an unnecessary (overhead) maintenance. In order to see such live/real-time metrics from the Cognos dashboard, it is not a straight way to build. This code pattern helps us to achieve this functionality with the help of Custom widgets which is built using d3 and js(java scripts).

In this code pattern, we will build a Cognos add-on to consume highly volatile streaming (real-time) data. The real-time Dashboard displays a mix of data (volatile and non-volatile) on a single dashboard. It shows a dancing chart that captures volatile data and incrementally updates itself. This pattern demonstrates displaying live-insights of the devices health metrics within Cognos Dashboard.

When the reader has completed this Code Pattern, they will understand how to:

* Build Cognos Custom Widgets
* Integrate JavaScript built Extension within Cognos Dashboard
* Display mix of Historical and Live Streaming IoT data in Cognos Dashboard
* Interact with widgets within the same Cognos Dashboard

## **Flow**

****

1. Develop the code (includes spec.json, js, css, Images) to build Cognos Custom Widget (Extensions).
2. Bundle the code as a zip file.
3. Upload the zipped files into Cognos using Extensions.
4. Use the built custom widget into Cognos Dashboard.

# 

# 

1. The Node-RED flow on Raspberry Pi receives the command

## **Included Components**

* [IBM Cloud](https://console.bluemix.net/catalog/): IBM's innovative cloud computing platform or IBM Cloud (formerly Bluemix) combines  
  platform as a service (PaaS) with infrastructure as a service (IaaS) and includes a rich catalog of cloud services that can be easily integrated with PaaS and IaaS to build business applications rapidly.
* [IBM Watson IoT Platform](https://internetofthings.ibmcloud.com/): IBM Watson™ IoT Platform for IBM Cloud gives you a versatile  
  toolkit that includes gateway devices, device management, and powerful application access. By using  
  Watson IoT Platform, you can collect connected device data and perform analytics on real-time data  
  from your organization.
* [IBM Watson Studio](https://www.ibm.com/bs-en/marketplace/data-science-experience): Analyze data using Python, Jupyter Notebook  
  and RStudio in a configured, collaborative environment that includes IBM value-adds, such as managed Spark.
* [DB2 Warehouse](https://console.bluemix.net/catalog/services/db2-warehouse): IBM Db2 Warehouse on Cloud is a fully-managed, enterprise-class, cloud data warehouse service.

## **Featured Technologies**

* [Analytics](https://developer.ibm.com/code/technologies/analytics?cm=IBMCode-_--_-featured_technologies-_-analytics): Finding patterns in data to derive information.
* [Data Science](https://developer.ibm.com/code/technologies/data-science?cm=IBMCode-_--_-featured_technologies-_-data-science): Systems and scientific methods to analyze structured and unstructured data in  
  order to extract knowledge and insights.
* [Internet of Things](https://en.wikipedia.org/wiki/Internet_of_things)