Request for Quotation for AFLCMC / Detachment 12 Kessel Run, Data Science Services Contract

Installation Guide

January 8, 2020

Solicitation No. HBBK-2019-0003, FA8730-20-F-0047

KESSELRUN

Prepared For:

Air Force Life Cycle Management Center/DET-12
Attn: Ms. Sara Corsetti and Mr. Joshua Naim
25 Randolph Road, Building #1110
Hanscom, AFB 01731

Prepared By:

Pinakin Patel, Chief Executive Officer
IT Concepts, Inc.
1600 Spring Hill Road, Suite 305
Vienna, VA 22182
(571) 918-9987
www.useitc.com
DUNS: 968872213
CAGE: 6GX40

SDVOSB

SBA 8(a) Certified

CMMISVC/3

CMMIDEV/3

Intertek Intertek Intertek

IT CONCEPTS INCORPORATED | DRIVEN TO DELIVER.

Table of Contents

List of Figures

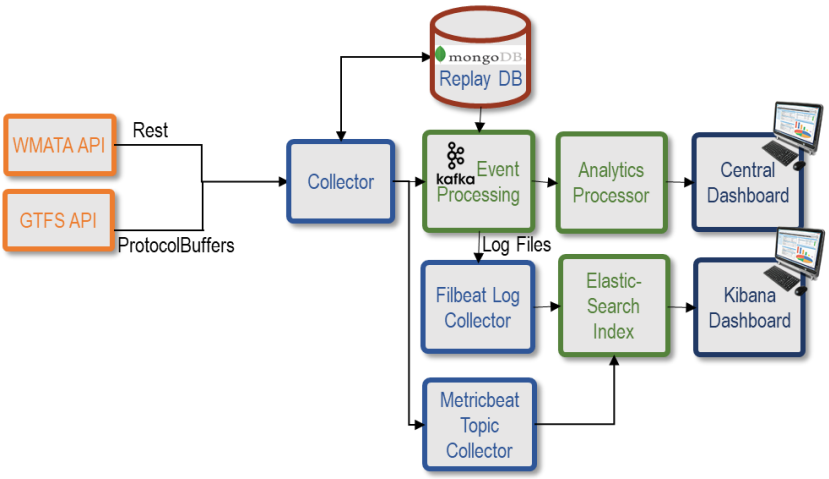
1 Installation Guide

This document serves as the installation guide for the Take Home challenge. All code referenced is available in the Kessel Run Take Home Challenge Git repository at: https://github.com/itconceptsinc/kr

1.1 Architecture

The diagrams below give a high level overview of the architecture designed for the Take Home Challenge solution. **Figure 1-1** is the data flow diagram for this effort. **Figure 1-2** shows the physical infrastructure as laid out in AWS.

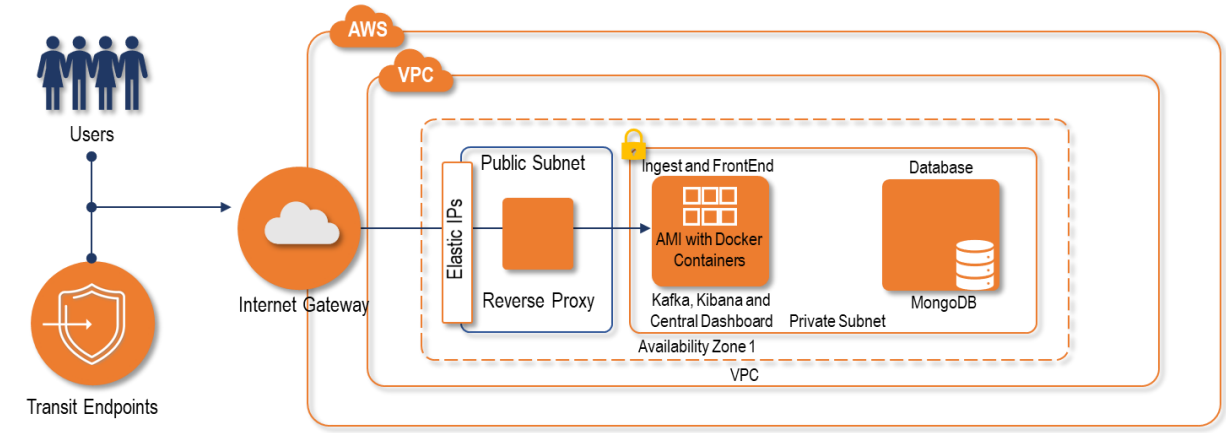
Figure 1-1: ITC Kessel Run Take Home Challenge Architecture



Data Flow from Washington Metropolitan Area Transit Authority (WMATA) and General Transit Feed Specification (Google)

- Collect and send data through our Kafka workflow where it is transformed into JSON
- Our Analytics Processor performs trending analysis, force directed categorization, and Bayesian analysis for prediction
- Data is output into a Plot.ly dashboard for end user consumption
- Monitor Kafka logs with FilBeat and ingest into ElasticSearch for display in a Kibana dashboard

Figure 1-2: Physical Architecture



The ITC Team Take Home Challenge is contained within a single AWS VPN and exposed through security groups through a public subnet to end users. ITC has 2 EC2s to contain the demonstration applications

1: Configured with Kafka, Kibana, Filebeat, Elastic Search all within their own Docker containers

In a future iteration, each container will be separated for scaling

2: Contains the MongoDB Docker container

The MongoDB is used to store streaming data for replay as the data stream volume may not have been enough

In a future iteration, we will offload the database into AWS databases – e.g., RDS, Dynamo, etc.

For simplicity of deployment, we are using a single MongoDB

2 Installation

2.1 AWS Environment

The AWS environment consists of two EC2s in a private VPC that is exposed via security groups to the public internet. Currently it is configured with a t2.small for MongoDB and Analytics Dashboard. A t2.xlarge is used for the Kafka and ELK stack. SSH access is available upon request at jon.hammond@useitc.com and delivery of an SSH key for access to the EC2s. Creation of EC2s can be accomplished using script or manually creating them using the Amazon Linux AMI - **Amazon Linux 2 AMI (HVM), SSD Volume Type-ami-00068cd7555f543d5 (64-bit x86)**. We installed with 100GB of EBS storage for the demonstration. Each was installed into the same VPC and availability zones and each has its own security group. In the future the environment will be defined using a cloud formation template. Upon request ssh access can be granted by emailing jon.hammond@useitc.com.

2.1.1 Security Groups

EC2 #1 – MongoDB and Analytics Dashboard – Inbound Rules defined and Outbound left open. In the future, the Instance would be secured more thoroughly but is left open for the demonstration with direct ssh access.

Figure 2-1: EC2 # 1 – MongoDB and Analytics Dashboard – Inbound Rules Defined/Outbound Open

Edit inbound rules

Type	Protocol	Port Range	Source		Description
HTTP	TCP	80	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP F	TCP	8050	Custom	0.0.0.0/0	Plotly-Dash
Custom TCP F	TCP	8050	Custom	::/0	Plotly-Dash
Custom TCP F	TCP	27017	Custom	0.0.0.0/0	Mongo Port
Custom TCP F	TCP	27017	Custom	::/0	Mongo Port
Custom TCP F	TCP	9300	Custom	0.0.0.0/0	Elastic
HTTPS	TCP	443	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom	::/0	e.g. SSH for Admin Desktop
Custom TCP F	TCP	9200	Custom	0.0.0.0/0	Elastic

EC2 #2 – Kafka, ELK Stack – Inbound Rules defined and Outbound left open. In the future the Instance would be secured more thoroughly but is left open for the demonstration with direct ssh access.

Figure 2-2: EC2 # 2 – Kafka, ELK Stack – Inbound Rules Defined/Outbound Open

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP F	TCP	8050	Custom	0.0.0.0/0	Dashboard
Custom TCP F	TCP	8050	Custom	::/0	Dashboard
Custom TCP F	TCP	32769	Custom	0.0.0.0/0	Kafka Broker
Custom TCP F	TCP	32769	Custom	::/0	Kafka Broker
Custom TCP F	TCP	5601	Custom	0.0.0.0/0	kibana
Custom TCP F	TCP	2181	Custom	0.0.0.0/0	zookeeper
Custom TCP F	TCP	9300	Custom	0.0.0.0/0	elastic
Custom TCP F	TCP	9200	Custom	0.0.0.0/0	elastic
Custom TCP F	TCP	9092	Custom	0.0.0.0/0	Kafka
Custom TCP F	TCP	9092	Custom	::/0	Kafka

In addition, for each instance, required installation includes docker and git.

sudo yum update

sudo amazon-linux-extras install docker

sudo yum install git

git clone https://github.com/itconceptsinc/kr.git

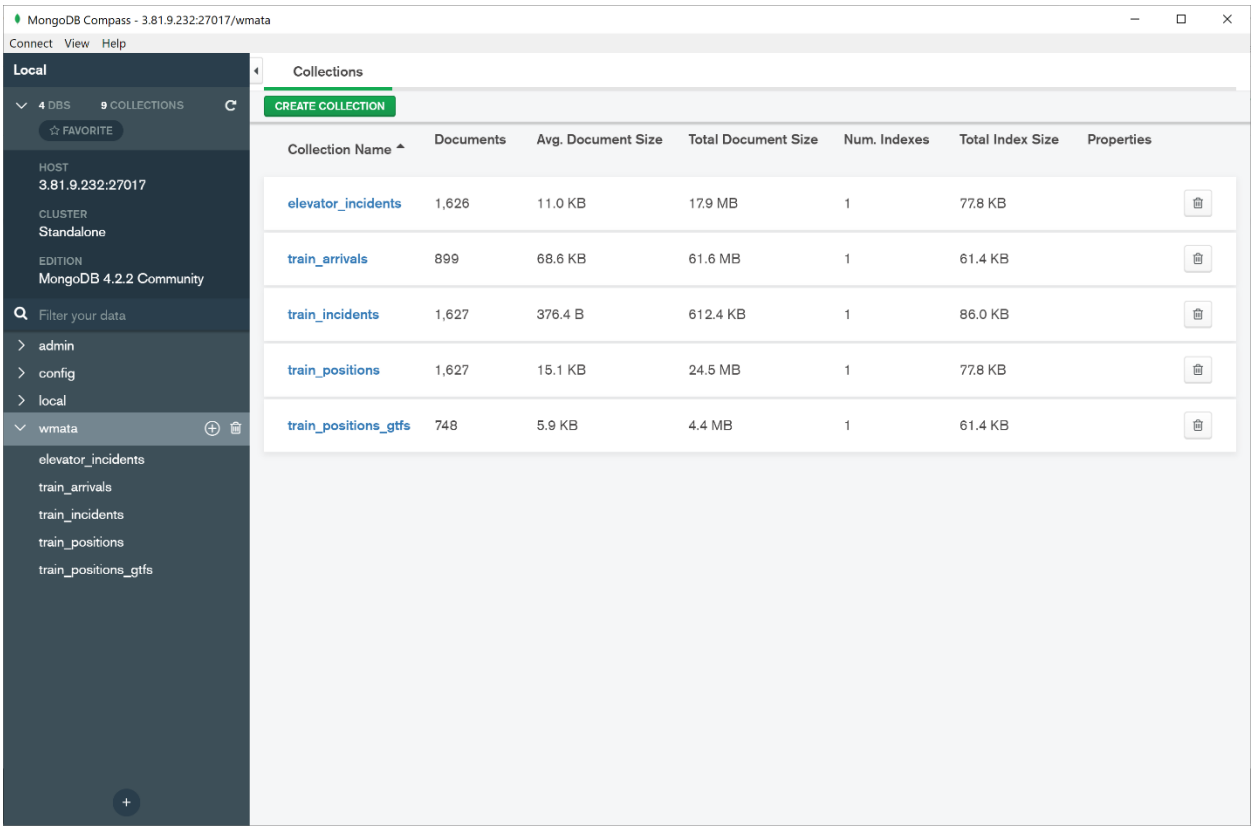
sudo ./conda.sh (from ec2_scripts)

sudo ./setup_python.sh (from ec2_scripts)

2.2 MongoDB

The purpose of the MongoDB instance is to store streaming data, so the data scientists have a non-streaming data set to design models against. It is configurable for the data collectors to directly feed Kafka or for MongoDB to feed Kafka. The MongoDB contains 5 collections although the demonstration currently does not use all 5. The collections may be browsed using an explorer tool such as Compass. Available at https://www.mongodb.com/download-center/compass. Other database explorer tools may also be used. The connect string is:
mongodb://ks_admin:ks_password@3.81.9.232:27017/?authSource=admin&readPreference=primary&appname=MongoDB%20Compass&ssl=false.

Figure 2-3: MongoDB Instance



All collections except train_positions_gtfs as JSON encoded. The GTFS collection is base64 encoded Google Protocol Buffers in the Google defined GTFS format. The JSON formats are described in Appendix A – Data Formats. The GTFS format specifications are available at: https://gtfs.org/. In addition to data stored in MongoDB, there also exists static data which is stored in the Git repository.

After cloning the git repository, MongoDB may be installed using sudo mongo.sh from the ec2_scripts directory.

2.3 Kafka with Zookeeper

Kafka routes the data though a simple workflow to our analytics processor. Kafka and Zookeeper are installed by running a sudo docker compose up from docker_instance directory after cloning the Git repository.

2.4 Elasticsearch

Elastic search is used to store Kafka log data and Kafka topic data so that we can visualize the operation and performance of the Kafka tool. The ELK stack is installed by install elasticseach followed by kibana, filebeats and metricbeats. Currently all are installed on the same instance as Kafka. This is due to filebeats reading the exported volume for logs from Kafka. In a future iteration, this can be reconfigured to separate these instances.

The installation of ElasticSearch, Kibana, and filebeat can be installed by running sudo ./filebeat.sh from the elk directory of the cloned repository. Filebeat requires access to a shared directory of the kafkalogs.

It particular it is based about the following commands:

sudo docker pull docker.elastic.co/elasticsearch/elasticsearch:7.5.1

sudo docker run -p 9200:9200 -p 9300:9300 --name "elastic" --ulimit nofile=65536:65536 -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:7.5.1 &

2.5 Kibana

Kibana is used as the dashboard to visualize the Kafka logs and topic data. This is to provide insight into the operational aspects of the data pipeline. Kibana was installed in the previous step however focuses on the following commands:

1) Pull the container
    a. sudo docker pull docker.elastic.co/kibana/kibana:7.5.1
2) Run the container and link it to Elasticsearch
    a. sudo docker run --link elastic:elasticsearch -p 5601:5601 --name "kibana" docker.elastic.co/kibana/kibana:7.5.1 &

There are two primary dashboards for kibana that are relevant to this effort. The first (Filebeat Kafka) displays the log files for the kafka server and gives insights into the running status including WARN, INFO, DEBUG, and ERROR messages. The second visualization (Metricbeat Kafka) displays the status of the topics and partitions withing kafka. It displays the current indexes, lags and topic details.

Figure 2-4: Filebeat Kafka Dashboard

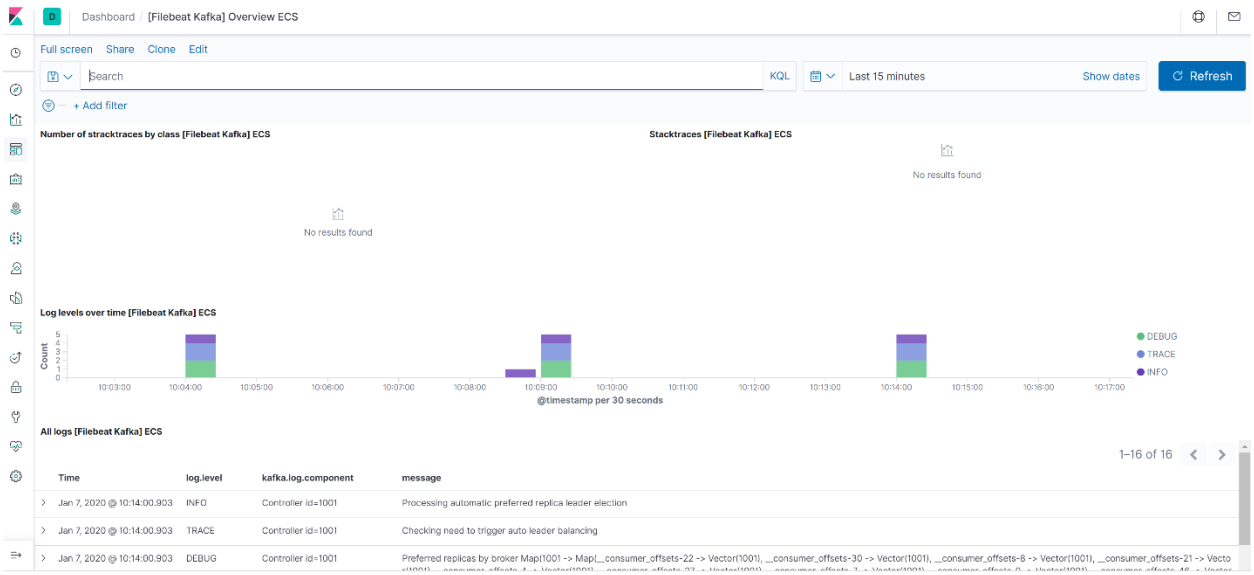
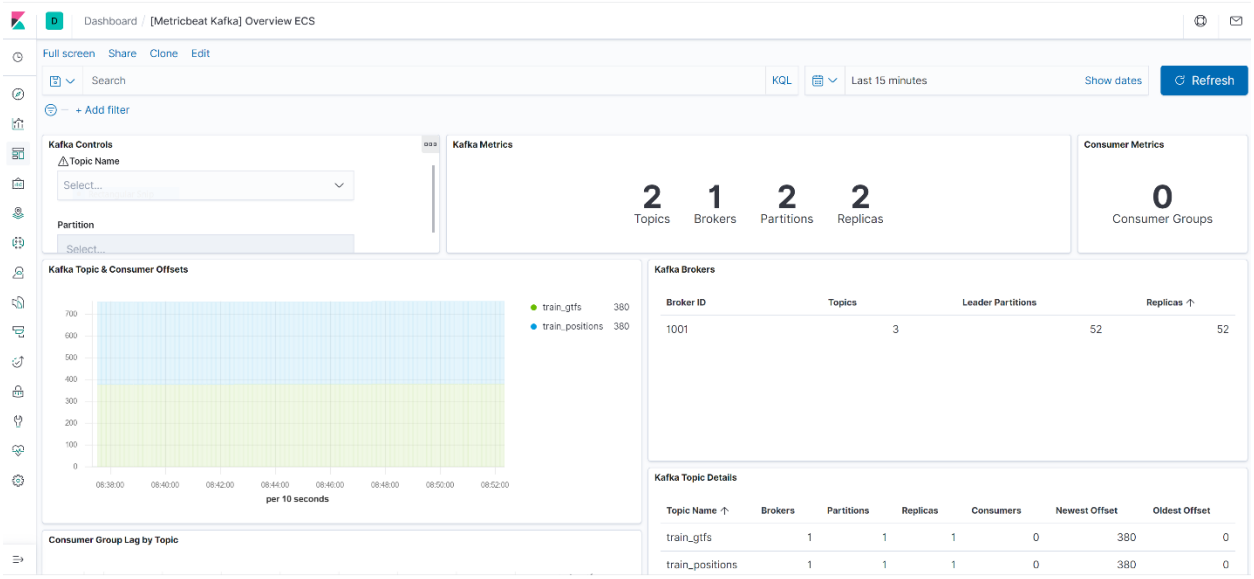


Figure 2-5: Metricbeat Kafka Dashboard



2.6 Filebeat

Filebeat is used to read log data from kafka and load into ElasticSearch so that Kibana can visualize it. Filebeat ses a shared volume from the Kafka docker container that is currently located in the home directory. Filebeat was installed in elasticsearch install but involves:

1) Pull the container
    a. sudo docker pull docker.elastic.co/beats/filebeat:7.5.1
2) Run the dashboard setup

   a. sudo docker run --name "filebeat" docker.elastic.co/beats/filebeat:7.5.1 setup -E setup.kibana.host=172.31.20.86:5601 -E output.elasticsearch.hosts=["172.31.20.86:9200"] &

3) Clean the container
   a. sudo dokcer rm filebeat
4) Run Filebeat
   a. sudo docker run --name "filebeat" --volume="filebeat2.docker.yml:/usr/share/filebeat/filebeat.yml" --volume="kafka.yml:/usr/share/filebeat/modules.d/kafka.yml" --volume="/var/lib/docker/containers:/var/lib/docker/containers:ro" --volume="/var/run/docker.sock:/var/run/docker.sock:ro" --volume="$KAFKA_LOG_DIR:/opt/kafka/logs" docker.elastic.co/beats/filebeat:7.5.1 filebeat -e -strict.perms=false -E setup.kibana.host=172.31.20.86:5601 -E output.elasticsearch.hosts=["172.31.20.86:9200"] &

2.7 Metricbeat

Metricbeat is used to talk with the Kafka broker in order to collect data on topics and partitions. The data is pushed into ElasticSearch so Kibana can visualize it. Metricbeat can be installed by running sudo ./metricbeat.sh from the elk directory of the clone repository. Metricbeat may need the Kafka container to grant access to read the topics. It may be necessary to run the following commands on the Kafka container.

/opt/kafka/bin/kafka-acls.sh --authorizer-properties zookeeper.connect=172.31.20.86:2181 --add --allow-principal User:'*' --operation Read --topic '*'

/opt/kafka/bin/kafka-acls.sh --authorizer-properties zookeeper.connect=172.31.20.86:2181 --add --allow-principal User:'*' --operation Describe --group '*'

Essentially the install requires the following commands:

1) Pull the Container
   a. sudo docker pull docker.elastic.co/beats/filebeat:7.5.1
2) Run the dashboard setup
   a. sudo docker run --name "metricbeat" docker.elastic.co/beats/metricbeat:7.5.1 setup -E setup.kibana.host=172.31.20.86:5601 -E output.elasticsearch.hosts=["172.31.20.86:9200"] &
3) Clean the container
   a. docker rm metricbeat
4) Run metricbeat
   a. Sudo docker run --name "metricbeat" --volume="metricbeat.docker.yml:/usr/share/metricbeat/metricbeat.yml" --volume="metricbeat.kafka.yml:/usr/share/metricbeat/modules.d/kafka.yml" --volume="/var/lib/docker/containers:/var/lib/docker/containers:ro" --volume="/var/run/docker.sock:/var/run/docker.sock:ro" docker.elastic.co/beats/metricbeat:7.5.1 metricbeat -e -strict.perms=false -E setup.kibana.host=172.31.20.86:5601 -E output.elasticsearch.hosts=["172.31.20.86:9200"] &

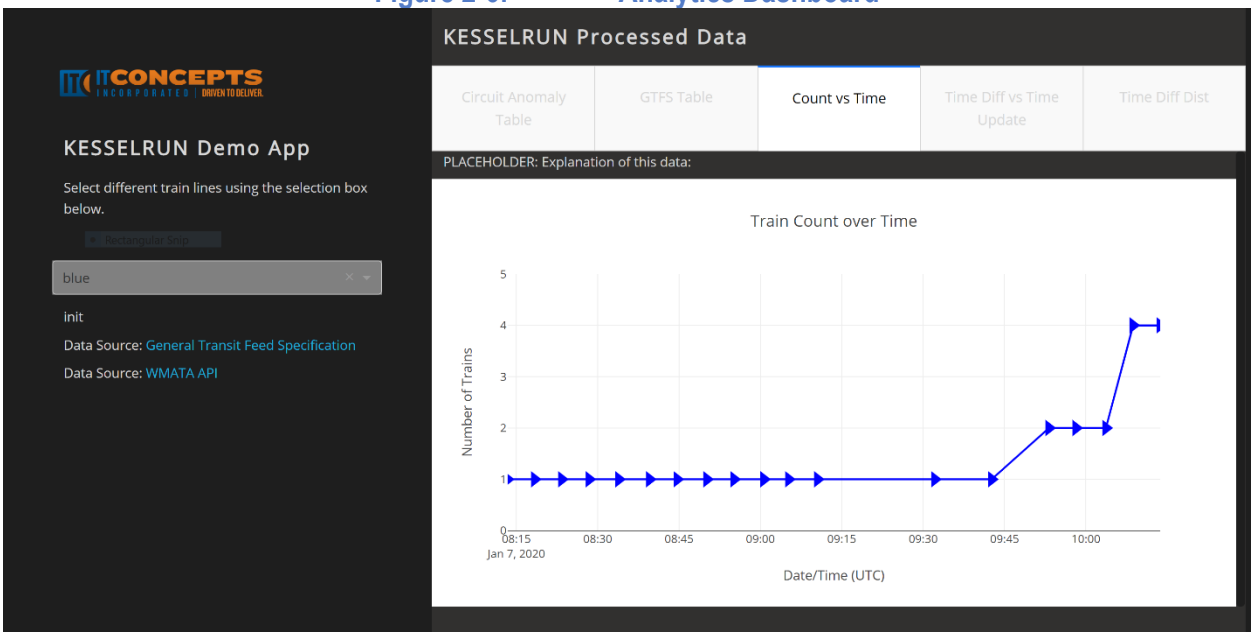
2.8 Analytics Dashboard

The Analytics Dashboard receives data from Kafka and performs several analytics. The analytic are then visualized using Plot.ly. To start the Analytics Dashboard run sudo python central_control_dashboard.py from the analysis_dashboard directory of the git repository clone.

The dashboard can be reached at: http://3.81.9.232:8050/

In order to use the dashboard, choose a train line from the dropdown on the left and choose an analytics visualization tab.

Figure 2-6: Analytics Dashboard



3 Development Support

3.1 Issue Tracking

Currently, ITC is tracking all issues within JIRA using a Kanban Agile process. JIRA is available at: https://useitc.atlassian.net/browse/KRTHC. Accounts may be requested by emailing jon.hammond@useitc.com.

4 JSON Data Formats

4.1 Elevator_incidents


```
{
  "_id": {
    "$oid": "5e0eeba764d2c719ac305a93"
  },
  "epoch_time": {
    "$numberDouble": "1578036135.772734"
  },
  "data": {
    "ElevatorIncidents": [{
      "UnitName": "A02W02",
      "UnitType": "ESCALATOR",
      "UnitStatus": null,
      "StationCode": "A02",
      "StationName": "Farragut North, L Street Entrance, West of Connecticut, south side L Street",
      "LocationDescription": "Escalator between street and mezzanine",
      "SymptomCode": null,
      "TimeOutOfService": "1117",
      "SymptomDescription": "Minor Repair",
      "DisplayOrder": {
        "$numberDouble": "0"
      },
      "DateOutOfServ": "2019-12-29T11:17:00",
      "DateUpdated": "2019-12-31T21:44:52",
      "EstimatedReturnToService": "2020-01-03T23:59:59"
    }, {
      "UnitName": "A07X06",
      "UnitType": "ESCALATOR",
```

"UnitStatus": null,
"StationCode": "A07",
"StationName": "Tenleytown-AU",
"LocationDescription": "Escalator between middle landing and mezzanine",
"SymptomCode": null,
"TimeOutOfService": "2115",
"SymptomDescription": "Preventive Maintenance Inspection",
"DisplayOrder": {
"$numberDouble": "0"
},
"DateOutOfServ": "2020-01-02T21:15:00",
"DateUpdated": "2020-01-02T22:30:33",
"EstimatedReturnToService": "2020-01-04T23:59:59"
}, {
"UnitName": "A08N02",
"UnitType": "ELEVATOR",
"UnitStatus": null,
"StationCode": "A08",
"StationName": "Friendship Heights, Western Avenue Entrance",
"LocationDescription": "Elevator between mezzanine and platform",
"SymptomCode": null,
"TimeOutOfService": "0128",
"SymptomDescription": "Inspection Repair",
"DisplayOrder": {
"$numberDouble": "0"
},
"DateOutOfServ": "2020-01-03T01:28:00",
"DateUpdated": "2020-01-03T01:56:27",
"EstimatedReturnToService": "2020-01-10T23:59:59"
}]
}
}

4.2 Train_arrivals


```
{
  "_id": {
    "$oid": "5e102c875654942c9aa23925"
  },
  "epoch_time": {
    "$numberDouble": "1578118279.980057"
  },
  "data": {
    "Trains": [{
      "Car": "8",
      "Destination": "Glenmont",
      "DestinationCode": "B11",
      "DestinationName": "Glenmont",
      "Group": "1",
      "Line": "RD",
      "LocationCode": "A01",
      "LocationName": "Metro Center",
      "Min": "BRD"
    }, {
      "Car": "8",
      "Destination": "Shady Gr",
      "DestinationCode": "A15",
      "DestinationName": "Shady Grove",
      "Group": "2",
      "Line": "RD",
      "LocationCode": "A01",
      "LocationName": "Metro Center",
      "Min": "BRD"
    } ]
  }
}
```

4.3 Train_incidents


```
{
  "_id": {
    "$oid": "5e0ee9aadae76ab937aa2179"
  },
  "epoch_time": {
    "$numberDouble": "1578035626.937173"
  },
  "data": {
    "Incidents": []
  }
}
```


4.4 Train_positions


```
{
  "_id": {
    "$oid": "5e0ee9aadae76ab937aa217a"
  },
  "epoch_time": {
    "$numberDouble": "1578035626.979605"
  },
  "data": {
    "TrainPositions": [{
      "TrainId": "031",
      "TrainNumber": "PM34",
      "CarCount": {
        "$numberInt": "0"
      },
      "DirectionNum": {
        "$numberInt": "1"
      },
      "CircuitId": {
        "$numberInt": "1323"
```

```
        },
        "DestinationStationCode": null,
        "LineCode": null,
        "SecondsAtLocation": {
            "$numberInt": "1550"
        },
        "ServiceType": "Unknown"
    }, {
        "TrainId": "045",
        "TrainNumber": "PM50",
        "CarCount": {
            "$numberInt": "0"
        },
        "DirectionNum": {
            "$numberInt": "1"
        },
        "CircuitId": {
            "$numberInt": "2925"
        },
        "DestinationStationCode": null,
        "LineCode": null,
        "SecondsAtLocation": {
            "$numberInt": "6040"
        },
        "ServiceType": "Unknown"
    } ]
  }
}
```


4.5 Train_positions_gtfs

After Base64 decoding, this format requires gtfs-realtime-bindings to decode. Decoded specification is at https://gtfs.org/reference/realtime/v2/.


```
{
```

```
  "_id": {
    "$oid": "5e10ea601513386a7654707d"
  },
  "epoch_time": {
    "$numberDouble": "1578166880.831649"
  },
  "data": {
    "$binary": {
      "base64": "Cg0KAzIuMBAAGMPUw/…….",
      "subType": "00"
    }
  }
}
```


4.6 Static Data Formats

Full static data descriptions are available at: https://gtfs.org/reference/static

4.6.1 Agency

Information about WMATA

agency_id,agency_name,agency_url,agency_timezone,agency_lang,agency_phone,agency_fare_url

4.6.2 Calendar Dates

Links service Ids to the Schedules.

service_id,date,exception_type

4.6.3 Feed Info

Describes the feed publisher information.

feed_publisher_name,feed_publisher_url,feed_lang,feed_start_date,feed_end_date

4.6.4 Routes

Train Line Information. For example “RED” line.

route_id,agency_id,route_short_name,route_long_name,route_type,route_url,route_color

4.6.5 Shapes

Defines the positions of each line.

shape_id,shape_pt_lat,shape_pt_lon,shape_pt_sequence,shape_dist_traveled

4.6.6 Stop Times

Scheduled times for each trip.

trip_id,arrival_time,departure_time,stop_id,stop_sequence,pickup_type,drop_off_type,shape_dist_traveled

4.6.7 Stops

Positional data and description for each stop

stop_id,stop_code,stop_name,stop_desc,stop_lat,stop_lon,zone_id

4.6.8 Trips

Links the Lines, Services, and Trips in a given direction.

route_id,service_id,trip_id,trip_headsign,direction_id,block_id,shape_id,scheduled_trip_id