

Prueba técnica: Data pipeline

Candidato: Martha Domínguez Orduño

Descripción: Desarrollar un pipeline de análisis de datos utilizando los datos abiertos de la Ciudad de México correspondientes a la ubicación de las unidades del metrobús durante la última hora para obtener un histórico de la posición en la que se encuentra cada unidad que pueda ser consultado mediante un API Rest filtrando por unidad o por alcaldía.

Arquitectura de la solución:



Stack tecnológico: Elastic Stack

ELK Stack (o Elastic Stack) es un conjunto de herramientas open source desarrolladas por Elasticsearch que permite recoger datos de cualquier tipo de fuente y en cualquier formato para realizar búsquedas, análisis y visualización de los datos en tiempo real.

Docker Compose

Con Compose puedes crear diferentes contenedores y al mismo tiempo, en cada contenedor, diferentes servicios, unirlos a un volumen común, iniciarlos y apagarlos, etc. Es un componente fundamental para poder construir aplicaciones y microservicios.

Repositorio del proyecto: <https://github.com/Marth4011/pipeline.git>

Rama master

Prueba técnica: Data pipeline

Fuente

De la fuente obtenemos los datos crudos en formato json sobre las ubicaciones del Metrobús de la Ciudad de México, los datos lucen de la siguiente forma:

```
{
  "nhits": 207,
  "parameters": {
    "dataset": "prueba_fetchdata_metrobus",
    "timezone": "UTC",
    "rows": 10,
    "format": "json",
    "records": [
      {
        "datasetid": "prueba_fetchdata_metrobus",
        "recordid": "2772aca1f8d450e3b98080905a79d093f395b007f",
        "fields": {
          "vehicle_id": "170",
          "date_updated": "2020-09-30 22:09:15",
          "position_longitude": "-99.18779754638672",
          "trip_schedule_relationship": 2,
          "position_speed": 16,
          "position_latitude": 19.3174991607666,
          "trip_route_id": "367",
          "vehicle_label": "112",
          "position_odometer": 231,
          "vehicle_current_status": 2,
          "geographic_point": [19.317499160767, -99.1877975463871],
          "geometry": {
            "type": "Point",
            "coordinates": [-99.187797546387, 19.317499160767]
          },
          "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
          "datasetid": "prueba_fetchdata_metrobus",
          "recordid": "8c34c423a11634dc8450557cf7bcceff728f10084",
          "fields": {
            "vehicle_id": "177",
            "trip_start_date": "20200428",
            "date_updated": "2020-09-30 22:09:15",
            "position_longitude": "-99.17749786376953",
            "trip_schedule_relationship": 0,
            "position_speed": 13,
            "position_latitude": 19.292600631713867,
            "trip_route_id": "367",
            "vehicle_label": "119",
            "position_odometer": 0,
            "trip_id": "9732304",
            "vehicle_current_status": 1,
            "geographic_point": [19.292600631714, -99.17749786377],
            "geometry": {
              "type": "Point",
              "coordinates": [-99.17749786377, 19.292600631714]
            },
            "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
            "datasetid": "prueba_fetchdata_metrobus",
            "recordid": "5891a34451e73729d26f5a2b69bb05090055e4a",
            "fields": {
              "vehicle_id": "128",
              "date_updated": "2020-09-30 22:09:15",
              "position_longitude": "-99.04730224609375",
              "trip_schedule_relationship": 2,
              "position_speed": 0,
              "position_latitude": 19.39080047607422,
              "vehicle_label": "221",
              "position_odometer": 46,
              "vehicle_current_status": 2,
              "geographic_point": [19.390800476074, -99.047302246094],
              "geometry": {
                "type": "Point",
                "coordinates": [-99.047302246094, 19.390800476074]
              },
              "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
              "datasetid": "prueba_fetchdata_metrobus",
              "recordid": "19.46489906311035",
              "vehicle_label": "317",
              "position_odometer": 0,
              "vehicle_current_status": 1,
              "geographic_point": [19.46489906311, -99.059501647949],
              "geometry": {
                "type": "Point",
                "coordinates": [-99.059501647949, 19.46489906311]
              },
              "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
              "datasetid": "prueba_fetchdata_metrobus",
              "recordid": "eddbb539c14cd27710f469c6e59035af05302ebf",
              "fields": {
                "vehicle_id": "1018",
                "trip_start_date": "20200428",
                "date_updated": "2020-09-30 22:09:15",
                "position_longitude": "-99.1295013427344",
                "trip_schedule_relationship": 0,
                "position_speed": 9,
                "position_latitude": 19.3978004455564,
                "trip_route_id": "301",
                "vehicle_label": "339",
                "position_odometer": 492,
                "trip_id": "9737162",
                "vehicle_current_status": 2,
                "geographic_point": [19.397800445557, -99.129501342733],
                "geometry": {
                  "type": "Point",
                  "coordinates": [-99.129501342733, 19.397800445557]
                },
                "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                "datasetid": "prueba_fetchdata_metrobus",
                "recordid": "bdc687fcbab66d0bc7d11183a64285e901b7ab5",
                "fields": {
                  "vehicle_id": "398",
                  "trip_start_date": "20200428",
                  "date_updated": "2020-09-30 22:09:15",
                  "position_longitude": "-99.18679809570312",
                  "trip_schedule_relationship": 0,
                  "position_speed": 0,
                  "position_latitude": 19.40220069885254,
                  "trip_route_id": "301",
                  "vehicle_label": "340",
                  "position_odometer": 735,
                  "trip_id": "9736450",
                  "vehicle_current_status": 2,
                  "geographic_point": [19.402200698853, -99.186798095703],
                  "geometry": {
                    "type": "Point",
                    "coordinates": [-99.186798095703, 19.402200698853]
                  },
                  "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                  "datasetid": "prueba_fetchdata_metrobus",
                  "recordid": "96ff0e95339e3c6501b375b20983ae189b3e97d",
                  "fields": {
                    "vehicle_id": "409",
                    "trip_start_date": "20200428",
                    "date_updated": "2020-09-30 22:09:15",
                    "position_longitude": "-99.17240142822266",
                    "trip_schedule_relationship": 0,
                    "position_speed": 12,
                    "position_latitude": 19.403900146484375,
                    "trip_route_id": "301",
                    "vehicle_label": "351",
                    "position_odometer": 131,
                    "trip_id": "9736261",
                    "vehicle_current_status": 2,
                    "geographic_point": [19.403900146484, -99.172401428223],
                    "geometry": {
                      "type": "Point",
                      "coordinates": [-99.172401428223, 19.403900146484]
                    },
                    "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                    "datasetid": "prueba_fetchdata_metrobus",
                    "recordid": "b3f4fc731c85701e1a945e0f059a39cbe230c2",
                    "fields": {
                      "vehicle_id": "449",
                      "trip_start_date": "20200428",
                      "date_updated": "2020-09-30 22:09:15",
                      "position_longitude": "-99.17040252685547",
                      "trip_schedule_relationship": 0,
                      "position_speed": 5,
                      "position_latitude": 19.403499603271484,
                      "trip_route_id": "301",
                      "vehicle_label": "391",
                      "position_odometer": 0,
                      "trip_id": "9736737",
                      "vehicle_current_status": 1,
                      "geographic_point": [19.403499603271, -99.170402526855],
                      "geometry": {
                        "type": "Point",
                        "coordinates": [-99.170402526855, 19.403499603271]
                      },
                      "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                      "datasetid": "prueba_fetchdata_metrobus",
                      "recordid": "551cfed918802a5a306aa9a1bd3de30e37edfe82",
                      "fields": {
                        "vehicle_id": "1",
                        "trip_start_date": "20200428",
                        "date_updated": "2020-09-30 22:09:15",
                        "position_longitude": "-99.14900207519531",
                        "trip_schedule_relationship": 0,
                        "position_speed": 3,
                        "position_latitude": 19.427000045776367,
                        "trip_route_id": "1",
                        "vehicle_label": "401",
                        "position_odometer": 668,
                        "trip_id": "9685594",
                        "vehicle_current_status": 2,
                        "geographic_point": [19.427000045776, -99.149002075195],
                        "geometry": {
                          "type": "Point",
                          "coordinates": [-99.149002075195, 19.427000045776]
                        },
                        "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                        "datasetid": "prueba_fetchdata_metrobus",
                        "recordid": "eebf53918acf00db3b0f0315334f92dlccfa6c",
                        "fields": {
                          "vehicle_id": "22",
                          "trip_start_date": "20200428",
                          "date_updated": "2020-09-30 22:09:15",
                          "position_longitude": "-99.15409851074219",
                          "trip_schedule_relationship": 0,
                          "position_speed": 1,
                          "position_latitude": 19.419599533081055,
                          "trip_route_id": "1",
                          "vehicle_label": "422",
                          "position_odometer": 560,
                          "trip_id": "9685769",
                          "vehicle_current_status": 2,
                          "geographic_point": [19.419599533081, -99.154098510742],
                          "geometry": {
                            "type": "Point",
                            "coordinates": [-99.154098510742, 19.419599533081]
                          },
                          "record_timestamp": "2020-10-01T03:09:16.345000+00:00"
                        }
                      ]
                    }
                  ]
                }
              ]
            }
          ]
        }
      ]
    }
  }
```

https://datos.cdmx.gob.mx/api/records/1.0/search/?dataset=prueba_fetchdata_metrobus&q=

Ingesta

Logstash, uno de los productos principales del Elastic Stack, se usa para agregar y procesar datos y enviarlos a Elasticsearch. Logstash es una pipeline de procesamiento de datos open source y del lado del servidor que te permite ingestar datos de múltiples fuentes simultáneamente y enriquecerlos y transformarlos antes de que se indexen en Elasticsearch

Logstash se basa en el uso de plugins que soportan los distintos elementos de la herramienta mencionados: entradas, códecs, filtros y salidas.

Logstash dispone de un gran conjunto de plugins para la entrada de datos:

Beats: datos de Elastic Beats

Elasticsearch: resultados de consultas a Elasticsearch

Exec: salida de comandos

File: ficheros

http: datos recibidos por http o https

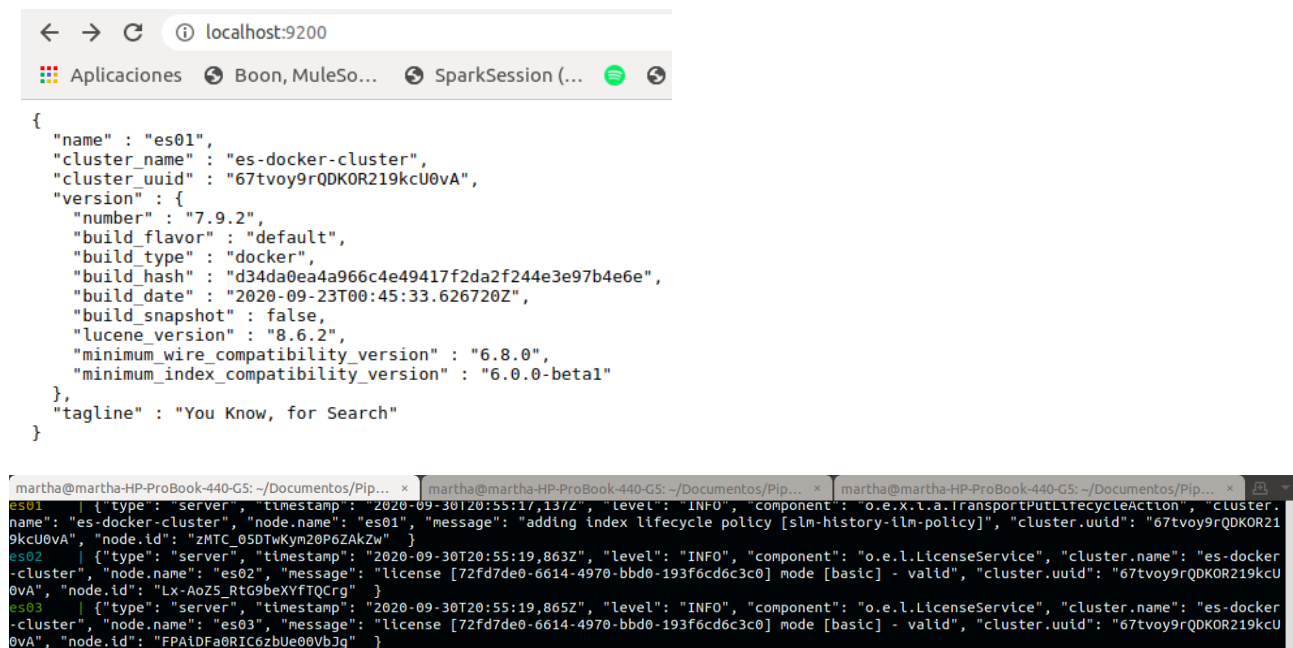
jdbc: datos de BBDD

pipe: datos de un comando pipe

tcp: datos recibidos por un socket tcp

Prueba técnica: Data pipeline

Para esta solución se levantó un clúster con tres nodos en un contenedor de Docker como a continuación se muestra:



```
{
  "name" : "es01",
  "cluster_name" : "es-docker-cluster",
  "cluster_uuid" : "67tvoy9rQDK0R219kcU0vA",
  "version" : {
    "number" : "7.9.2",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "d34da0ea4a966c4e49417f2da2f244e3e97b4e6e",
    "build_date" : "2020-09-23T00:45:33.626720Z",
    "build_snapshot" : false,
    "lucene_version" : "8.6.2",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

```
es01 | {"type": "server", "timestamp": "2020-09-30T20:55:17.137Z", "level": "INFO", "component": "o.e.x.t.a.transport.PutLifecycleAction", "cluster.name": "es-docker-cluster", "node.name": "es01", "message": "adding index lifecycle policy [slm-history-ilm-policy]", "cluster.uuid": "67tvoy9rQDK0R219kcU0vA", "node.id": "zMTC_05DTwKym20P6ZakZw" }
es02 | {"type": "server", "timestamp": "2020-09-30T20:55:19.863Z", "level": "INFO", "component": "o.e.l.LicenseService", "cluster.name": "es-docker-cluster", "node.name": "es02", "message": "license [72fd7de0-6614-4970-bbd0-193f6cd6c3c0] mode [basic] - valid", "cluster.uuid": "67tvoy9rQDK0R219kcU0vA", "node.id": "Lx-AoZ5_RtG9bexYfTQCrg" }
es03 | {"type": "server", "timestamp": "2020-09-30T20:55:19.865Z", "level": "INFO", "component": "o.e.l.LicenseService", "cluster.name": "es-docker-cluster", "node.name": "es03", "message": "license [72fd7de0-6614-4970-bbd0-193f6cd6c3c0] mode [basic] - valid", "cluster.uuid": "67tvoy9rQDK0R219kcU0vA", "node.id": "FPA1DFa0RIC6zbUe0vBjg" }
```

El siguiente paso es ingestar el archivo json en Logstash mediante su API.

Almacenamiento

Para el almacenamiento de los datos se utiliza la herramienta elasticsearch la cual proporciona una API RESTful para la comunicación con las aplicaciones del cliente. Por lo tanto, las llamadas REST se usan para ingestar datos, realizar búsquedas y análisis de datos, así como también administrar el cluster y sus índices. Internamente, todos los métodos descritos recurren a esta API para ingestar datos en Elasticsearch, en el clúster antes descrito se tienen tres nodos los cuales entre otras funcionalidades nos brindan el almacenamiento de los datos para posteriormente realizar el procesamiento de los datos con Kivana Dev Tools.

Prueba técnica: Data pipeline

Análisis y visualización

Para la consulta de datos se puede utilizar la herramienta gráfica de Kibana llamada Dev Tools con la cual se pueden lanzar comandos GET, POST, PUT y DELETE como se puede apreciar en la siguiente imagen de muestra.

