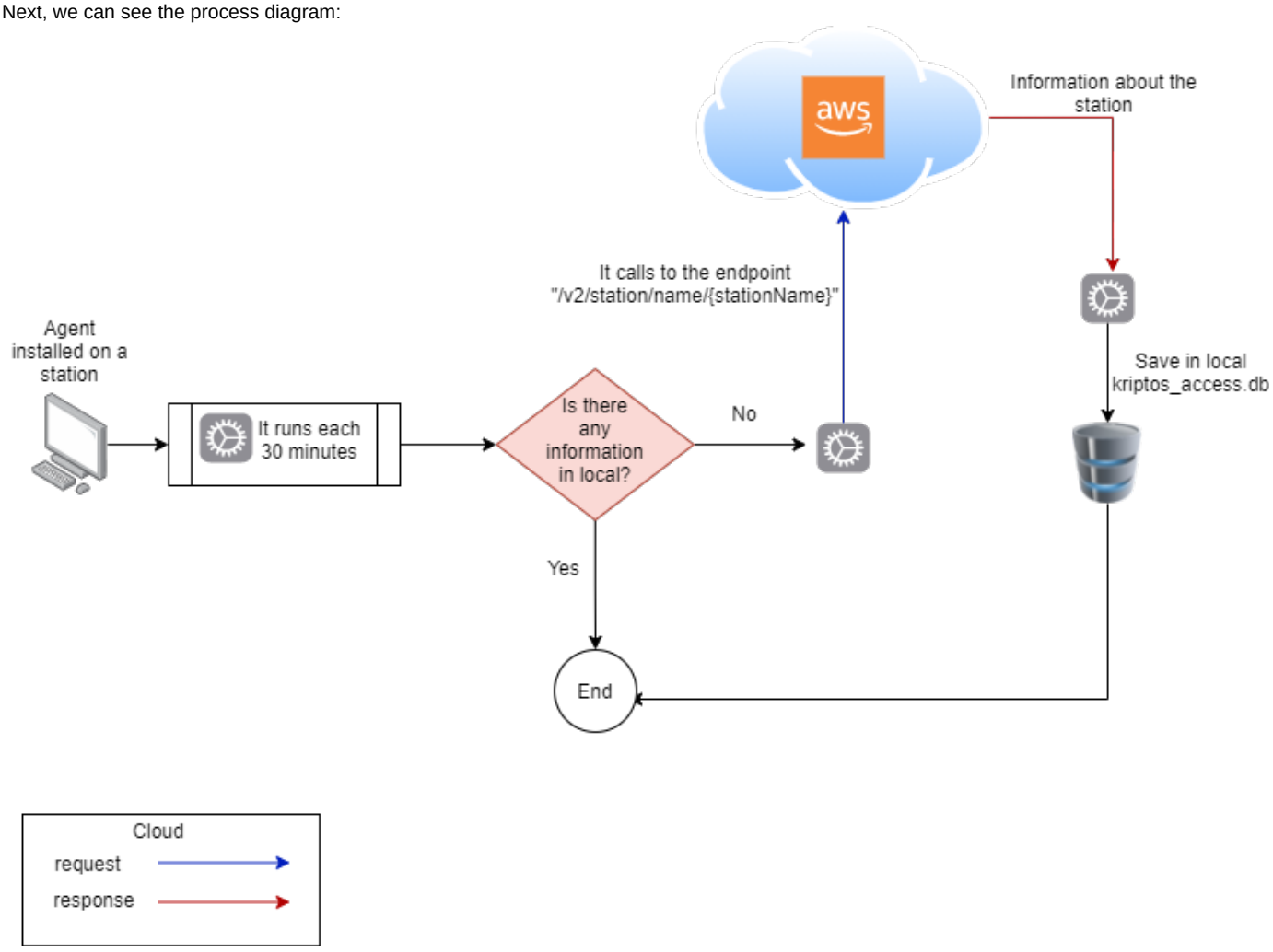
Access information process

This process allows the agent to get all information about a station, this information is:

- Enterprise ID which the station belongs
- Unique station ID generated in the backend
- Area ID which the station belongs
- Vectorizer version for this station

The agent sends the hostname trough an endpoint, and it receives all information about that hostname. This information we use to call another endpoints, because it's mandatory on them. All of this information is saved in a database called **kriptos_access.db**.

Next, we can see the process diagram:



This process is the first that runs as soon as the agent is installed, and after this, it runs each 30 minutes every day and it always verifies if the information is saved in local and if there's no information, this process calls the endpoint again to get the information, but if the information exists in local, nothing happens and the process finishes until the next verification (after 30 minutes).

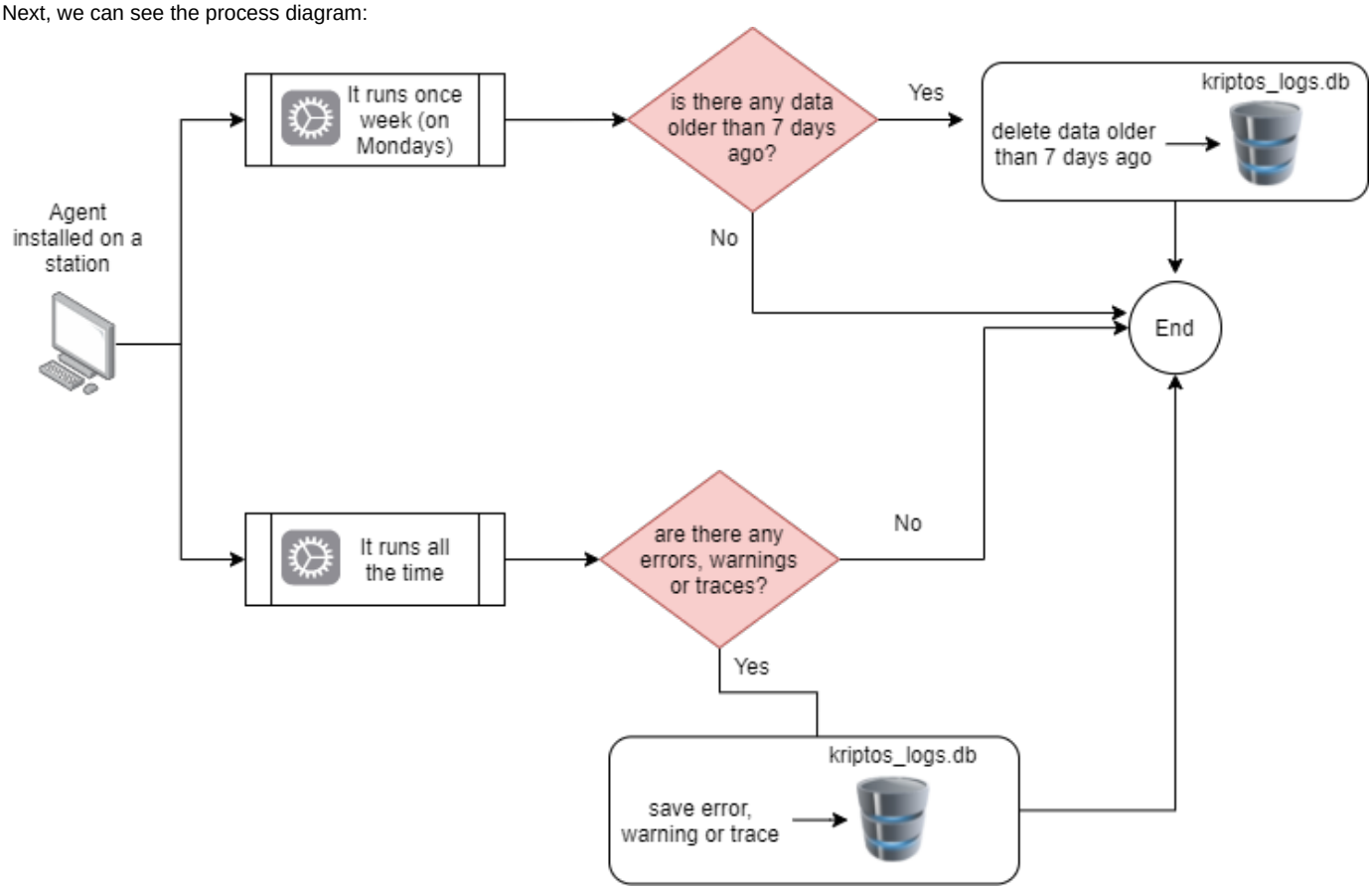
Local logs process

This process consists on save all errors, all information and all warnings we want to catch from the agent, and all of these information we'll save in a local database called **kriptos_logs.db**.

Once week (maybe on Mondays), the agent will delete the old logs, keeping only the last week logs (logs from the last 7 days ago) and this process will work automatically, we've decided to do this process, because it's important to avoid the size of the database grows without limits.

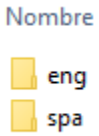
All the time, this process will be monitoring for catching errors, warnings and traces we want to save in the database **kriptos_logs.db**.

Next, we can see the process diagram:



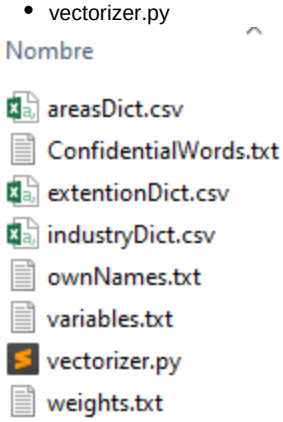
Machine learning process

This process downloads all machine learning files that agent needs to vectorize documents. First, the process makes a call to an endpoint to get the urls (AWS S3) and it uses these urls to download all files, it’s important to do this process very fast, because the urls expire in a few seconds. After downloading all files, it creates a folder called “**ml**“ and save all files on this one. The files are organized in two folders (inside ml folder): **spa** and **eng**. Each folder has all files about its respective language.



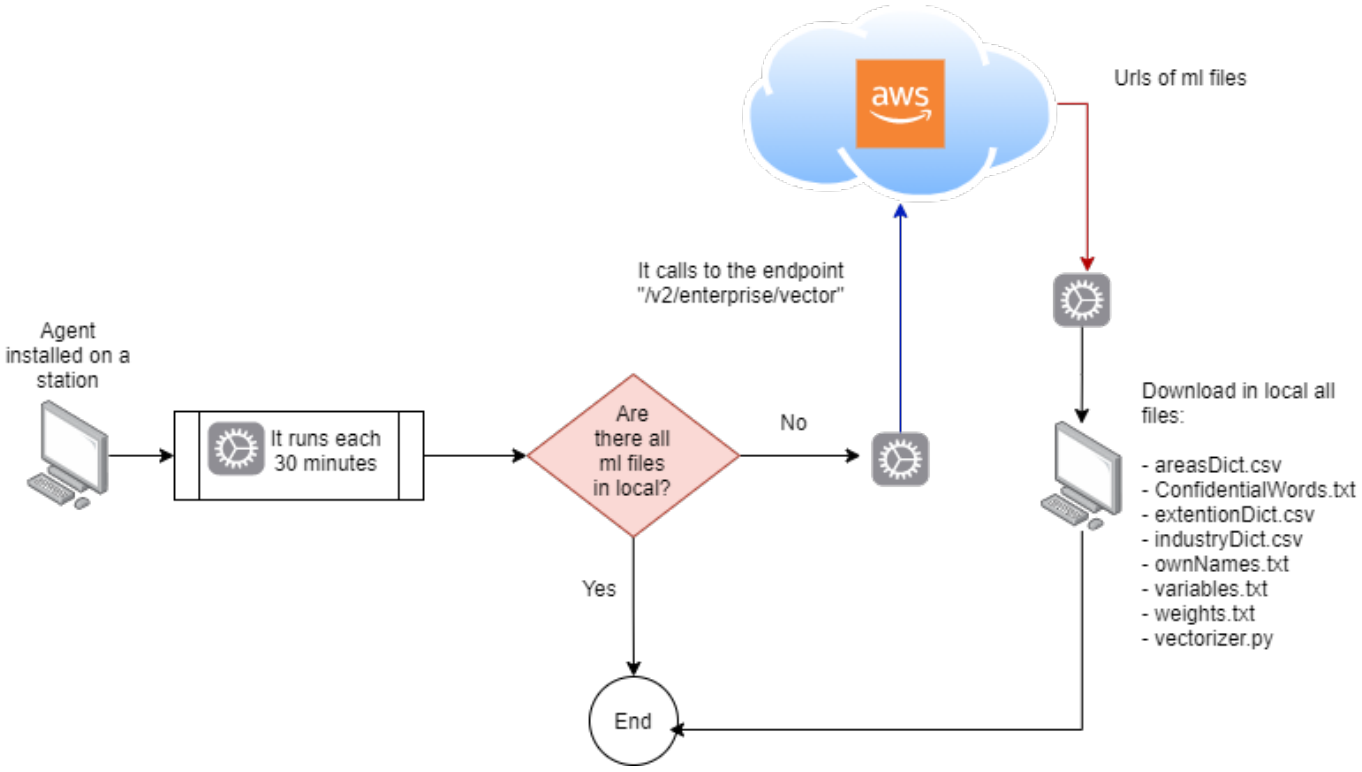
The files downloaded are:

- areasDict.csv
- ConfidentialWords.txt
- extentionDict.csv
- industryDict.csv
- ownNames.txt
- variables.txt
- weights.txt
- vectorizer.py



This process runs each 30 minutes every day to verify if there are all ml files, if there aren't the ml files, it tries again to download all files through the endpoint.

Next, we can see the process diagram:


aws
Urls of ml files
It calls to the endpoint
"/v2/enterprise/vector"
Agent
installed on a
station
It runs each
30 minutes
Are
there all
ml files
in local?
No
Yes
End
Download in local all
files:
- areasDict.csv
- ConfidentialWords.txt
- extentionDict.csv
- industryDict.csv
- ownNames.txt
- variables.txt
- weights.txt
- vectorizer.py

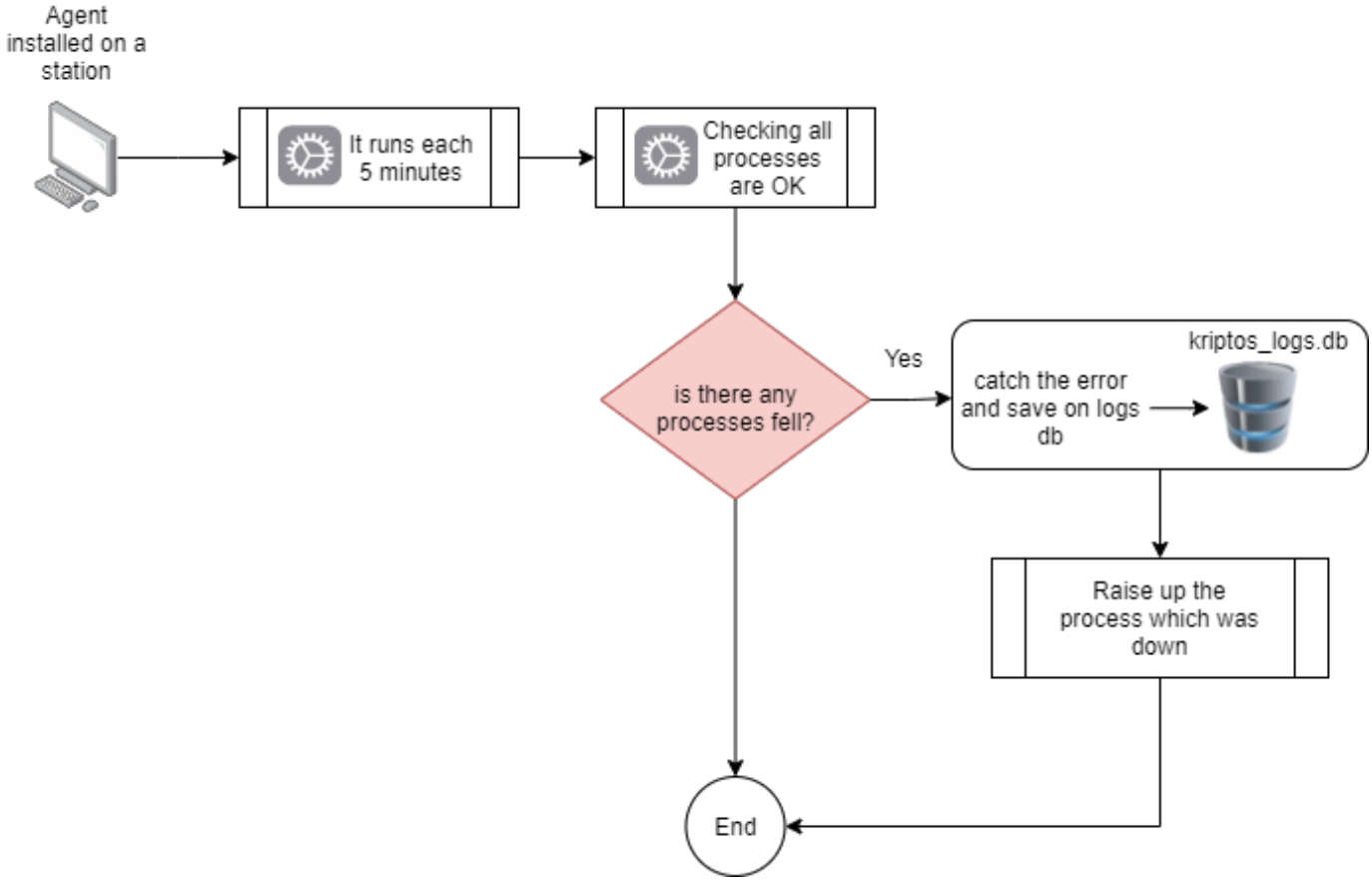
Monitoring process

This process runs all the time and checks the next:

- Checking all processes are running, it runs each 5 minutes, the processes are:
  - Vectorizer
  - Scanning
  - Real time scanning
  - Access information
  - Machine learning
  - Remote handling
  - Updater
  - Cleaner
  - Telemetry
  - Writer

If any processes is fell, the monitoring process must find the reasons why that process failed. After this, the monitoring process must raise up the process which was down.

Next, we can see the process diagram:



Real time scanning process

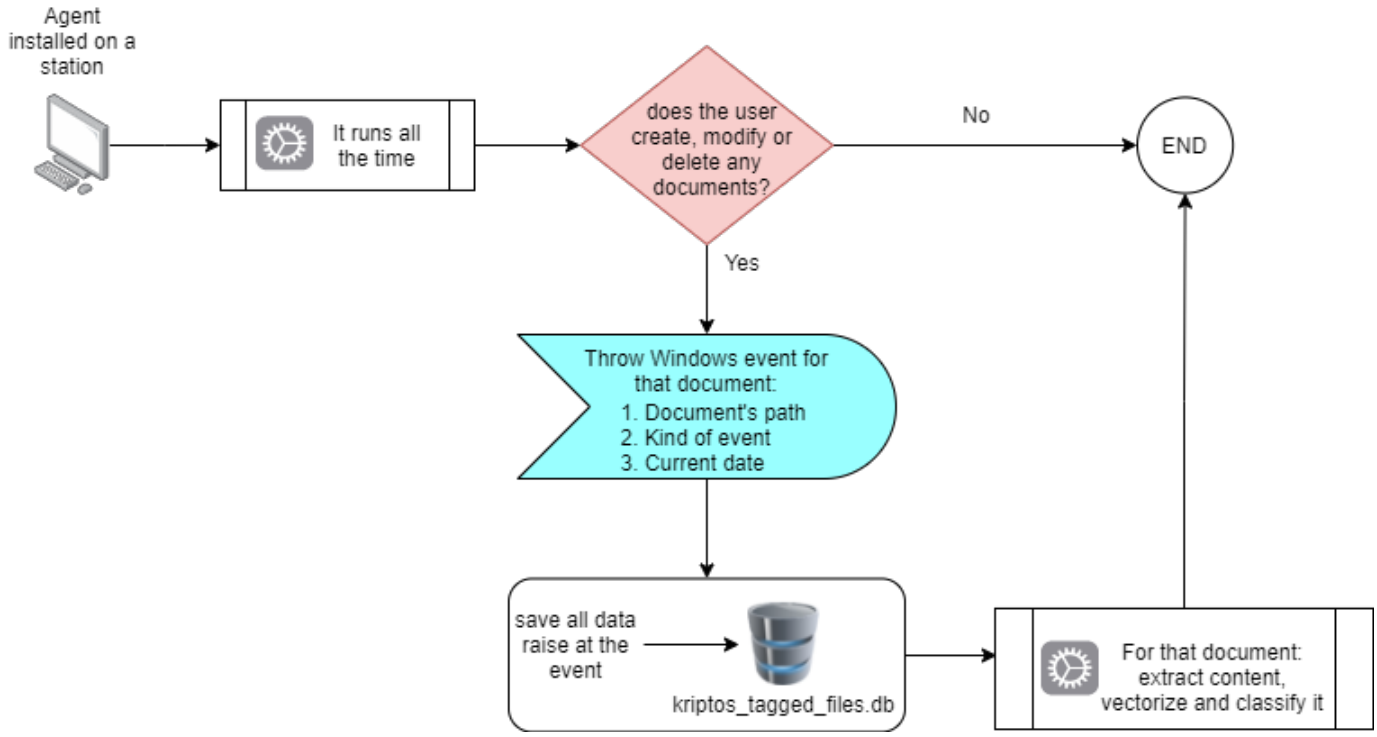
This process is watching all the time if there's a new event related to: creating, modifying or deleting a document on this computer. If an event occurred, the process catches this event which raise the next information:

- Path of the document
- Kind of event: creating, modifying or deleting
- Current date of the event

So, this information is saved on a database called **kriptos_tagged_files.db** and inmediately, this document is vectorized (with the document content) and classified in real time. The process uses a high priority for this kind of documents, because these documents are processed in real time.

> The compatible formats are: doc, docx, xls, xlsx, ppt, pptx and pdf.

Next, we can see the process diagram:



Remote handling process

This process is used for the agent to execute certain actions ordered from the backend, there are actions that are executed only once and others that the agent will be executing indefinitely until the opposite is indicated in the backend.

Actions which are executed only once, are the next:

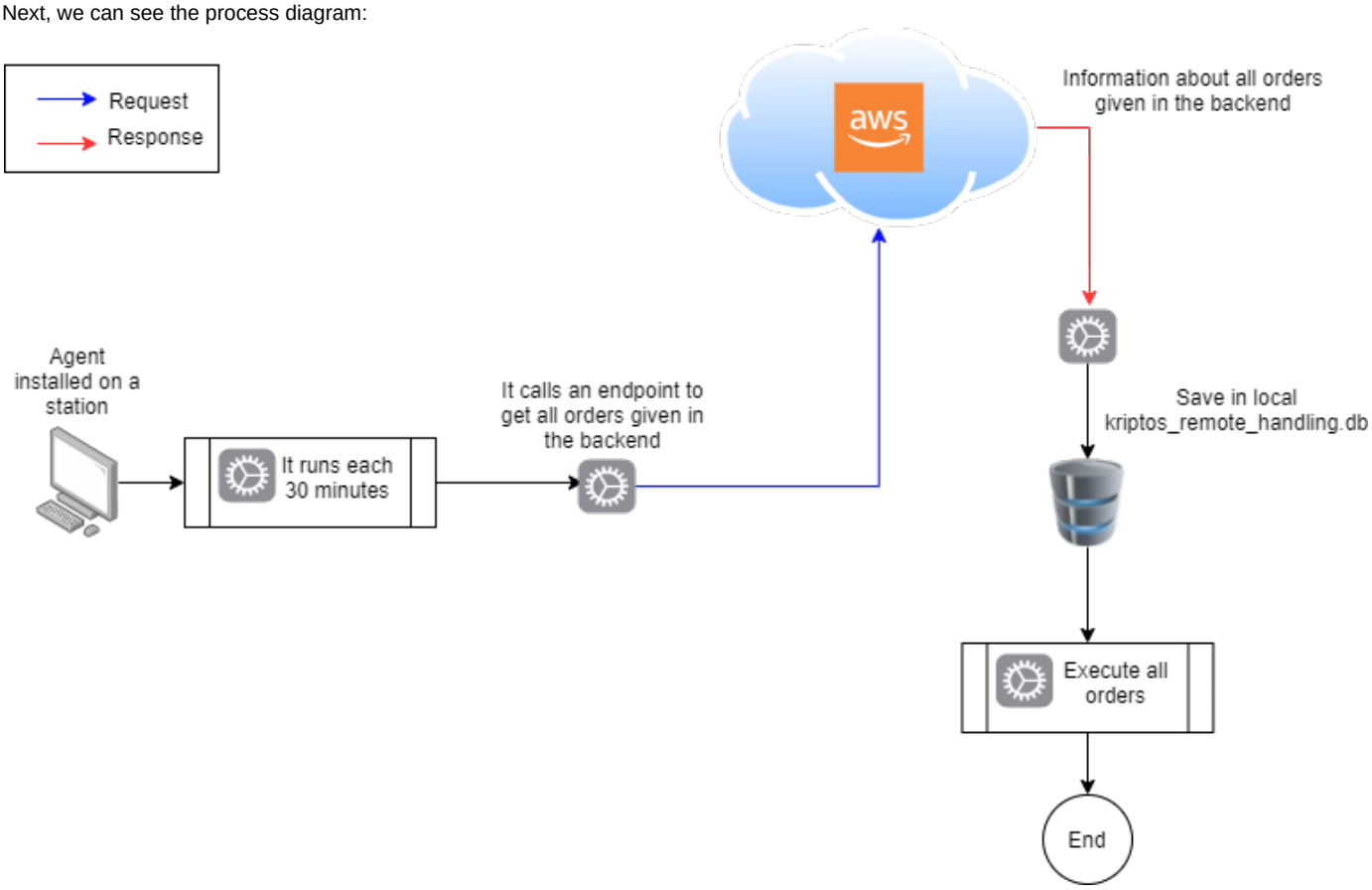
1. Rescan of all documents from the station
2. Sending all logs from the station (errors, warnings, crashes)
3. Reclassification of certain documents indicated by the customer, the customer can indicate the documents he wants to reclassify with a specific tag (we do similar in the cloud agent)

Actions which are executed indefinitely until the opposite is indicated in the backend, are the next:

1. Stop of the agent in the station
2. Activate the telemetry
3. Change the interval to do requests for this process, by default is 30 minutes
4. Change the urls of all endpoints, this case can be used when the backend is migrated to other infraestructure

This information is saved on a database called **kriptos_remote_handling.db** in json format.

Next, we can see the process diagram:



Scanning process

This is the biggest process of the agent, because it contains the next subprocesses:

- Hard disk scanning
- Vectorization
- Language detection
- Send documents to the server queue

Hard disk scanning consists in review every folder of all units from the hard disk (except special folders), while this subprocess is running, it detects all documents with the formats we support: doc, docx, xls, xlsx, ppt, pptx, pdf. All of these documents are saved on a database called **krip tos_untagged.db**. This subprocess runs once a week to detect all new documents that real time process cannot detect (for many reasons).

Vectorization consists to have a work runs all the time, while this happens we read the documents from the database **kriptos_untagged.db** and we vectorize each document for parts, for example, in excel documents, we vectorize each cell until we finish vectorize all cells, another example is in pdf documents, we vectorize each page until we finish all pages. So, doing this vectorization, we save many hardware resourses from the station.

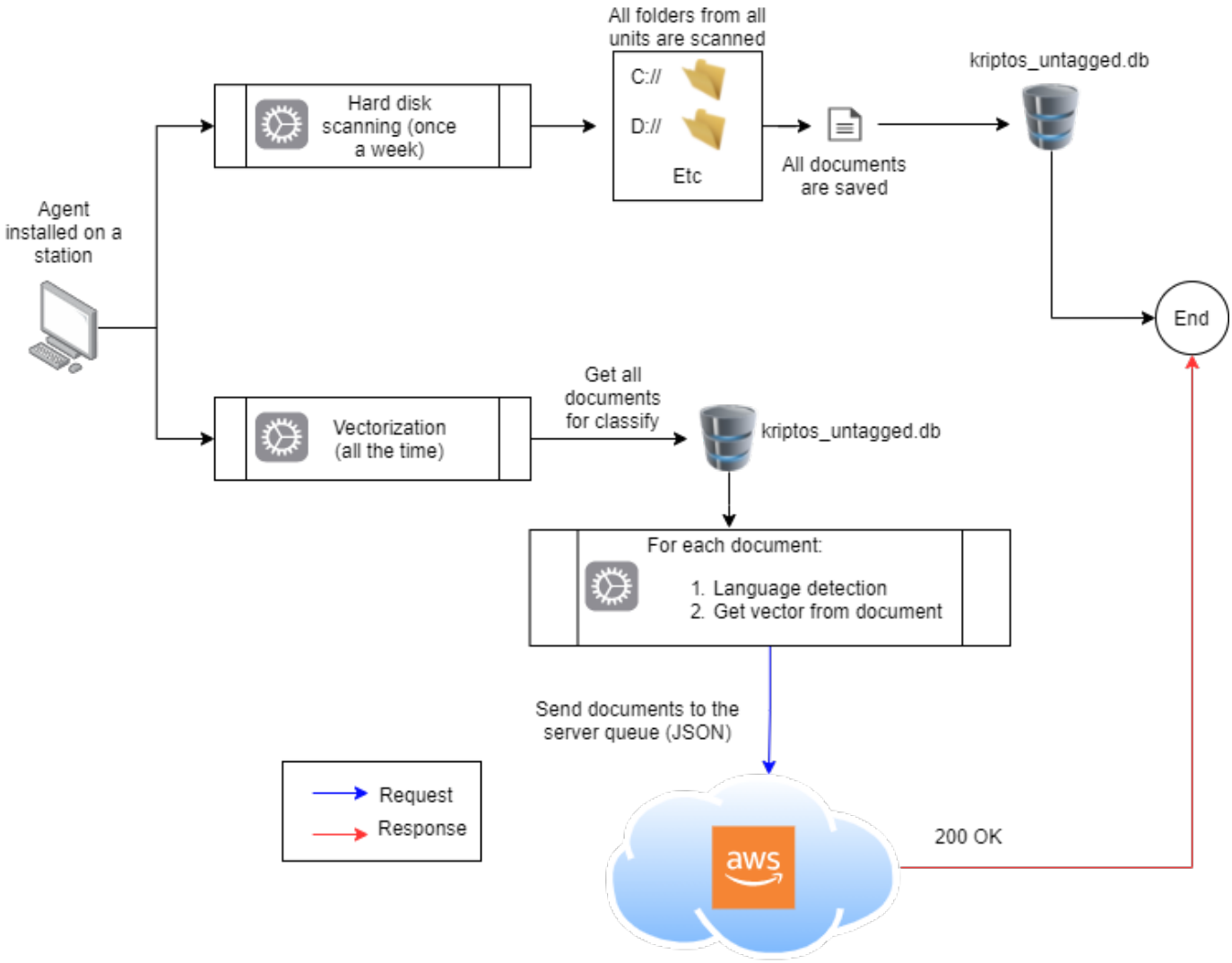
Language detection occurs while we vectorize the documents, at this moment we detect the language, but if the language is a strange language, we send to detect the language a second time and we choose the best option.

Send documents to the server queue consists on send all documents scanned and vectorized with priority 2, to the server and that’s all. These documents are sent in JSON format and we send per document the next information:

- ID
- Path
- Scanning date
- Vector
- Modification date

After this, the server must be classify all of these documents, and the agent will ask for them after a few minutes.

The process diagram is the next:


All folders from all units are scanned
C://
D://
Etc
kriptos_untagged.db
Hard disk scanning (once a week)
All documents are saved
Agent installed on a station
End
Get all documents for classify
Vectorization (all the time)
kriptos_untagged.db
For each document:
1. Language detection
2. Get vector from document
Send documents to the server queue (JSON)
Request
Response
aws
200 OK

Telemetry process

When the telemetry is activated for a station in the backend, this process starts to register data on a local database called **kriptos_telemetry.db**, and to send these data to the backend through an endpoint. These data is going to send to the backend each minute, for simulate a real time data on the web platform, and it will still sending data until from the backend is order to deactivate the telemtry in this station.

If the hostname is changed, the monitoring module detect it when it starts, so it informs to the backend about the hostname change, if the backend responds with an OK, the agent updates the hostname on its local database called kriptos_access.db.

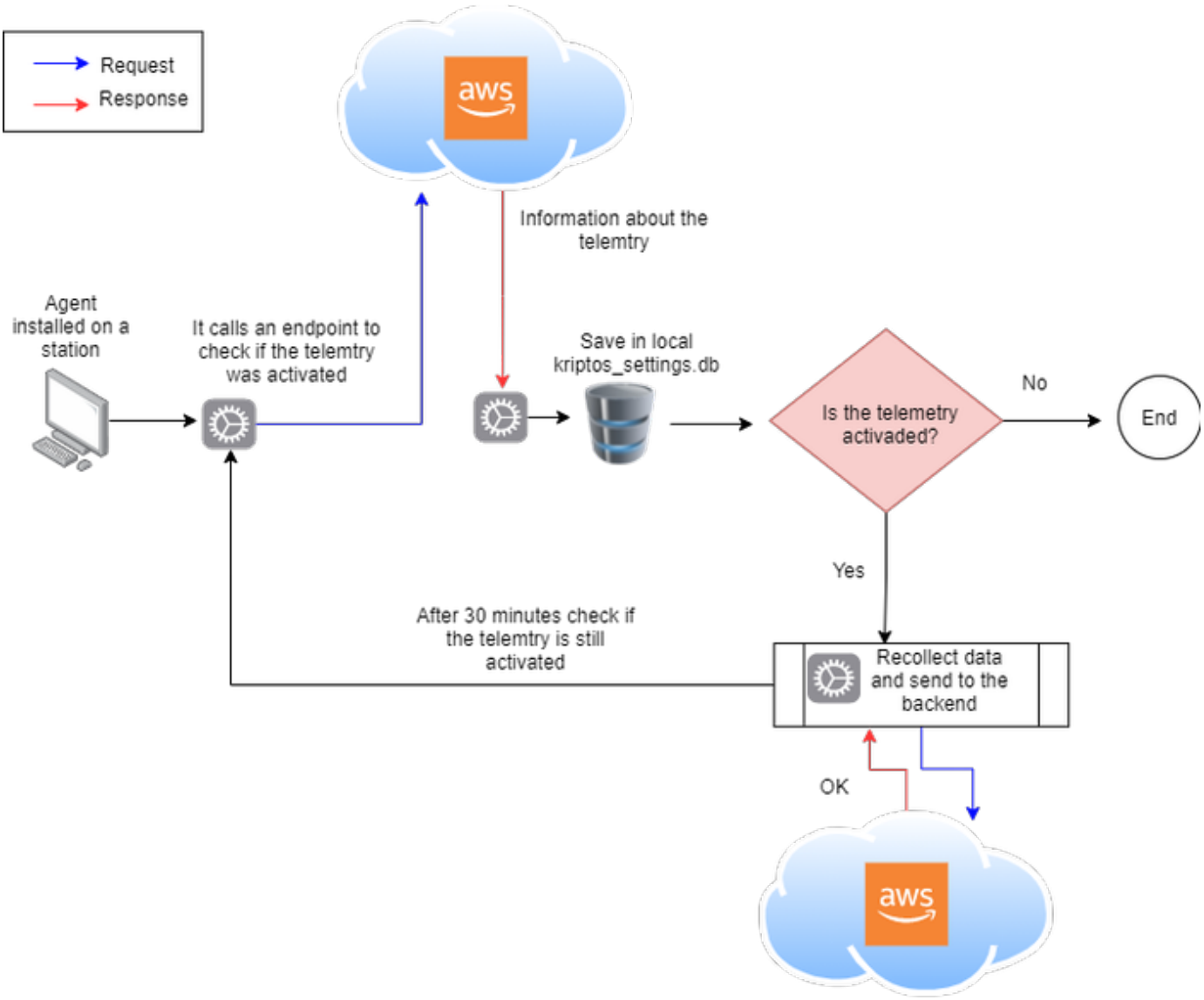
The telemetry data we can get is the next:

- What ml version is the agent using or downloaded?
- How many strange languages were detected in the idiom detection process?
- Are the processes activated or are they in stop?
- Was the computer shutdown?
- How many errors were generated in the content extraction process and what documents were?
- Was the hostname changed?
- Was the agent updated and what version is?
- Is the Windows original?
- What errors have generated on the computer when the scanning is running?
- What hardware and software features does the computer have?
- are there any problems to add metadata on the documents?

At the moment, only these features we can catch on this process.

After 30 minutes, the remote handling module checks if the telemetry is still activated in the backend, and it depends of this to stop sending data to the backend. If we want to have a good telemetry, we recommend to have the agent sends data at least for 1 day to certain station, because if we activate the telemetry for every stations, it will be very expensive in our infraestructure.

Next, we can see the process diagram:


Request
Response
aws
Information about the telemtry
Agent installed on a station
It calls an endpoint to check if the telemtry was activated
Save in local kriptos_settings.db
Is the telemetry activaded?
No
End
Yes
After 30 minutes check if the telemtry is still activated
Recollect data and send to the backend
OK
aws