Seneca
POLYTECHNIC

How Transactional Machine Learning (TML) Processing and Machine Learning Works?

Dr. Sebastian Maurice
August 2023

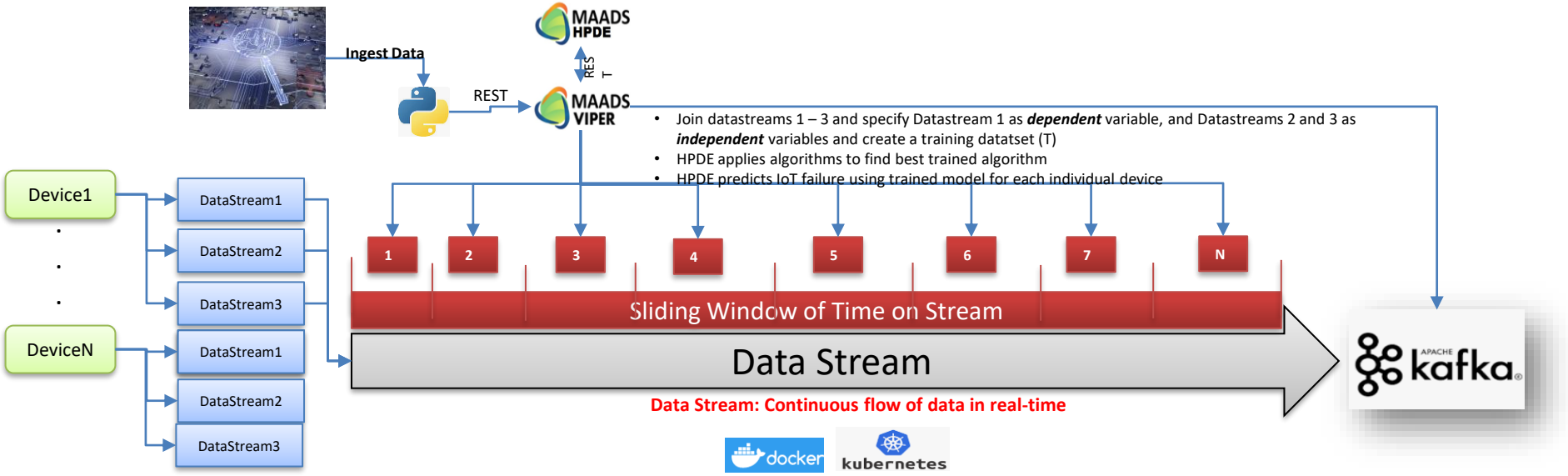
How TML Preprocessing works?

- All data streams from devices flow into Apache Kafka to a Kafka Topic
- TML performs **in-memory** processing of data in the Kafka Topic using TWO components across all **sliding time windows**
  - Python Script that uses the MAADSTML python library functions
  - MAADS-VIPER binary that can run in Linux, Mac, Windows (or other operating systems) on any Chip (32 or 64 bit) architecture (AMD, ARM, PPC, S390x, etc.)
- REST API connect MAADSTML python script to MAADS-VIPER
- 35+ different processing types: min, max, dataage, timediff, variance, anomaly prediction, outlier detection, etc…
- Apache Kafka is the central source of both input and output data – ***no external real-time database needed***
- *Our technology can process unlimited number of devices (billions at high speed)*
- All TML solutions are containerized with Docker and scale with Kubernetes

How TML Machine Learning works?

- All data streams from devices flow into Apache Kafka to a Kafka Topic
- TML performs **in-memory** machine learning of data in the Kafka Topic by **joining data streams** using THREE components across all **sliding time windows**:
  - Python Script that uses the MAADSTML python library functions
  - MAADS-VIPER binary that can run in Linux, Mac, Windows (or other operating systems) on any Chip (32 or 64 bit) architecture (AMD, ARM, PPC, S390x, etc.)
  - MAADS-HPDE binary that can run in Linux, Mac, Windows (or other operating systems) on any Chip (32 or 64 bit) architecture (AMD, ARM, PPC, S390x, etc.)

- REST API connect MAADSTML python script to MAADS-VIPER and MAADS-HPDE
- 5 different algorithm types: logistic regression, linear regression, gradient boosting, neural networks, ridge regression
- Apache Kafka is the central source of both input and output data for estimated parameters – ***no external real-time database needed***
- ***TML auto-creates individual machine learning models for each Device at the "entity" level and joins datastreams 1-3 for each device and user specifies "Dependent" variable streams, and "Independent" variables streams***
- *Our technology can build unlimited machine learning models (billions at high speeds) for unlimited number of devices (billions at high speed)*
- All TML solutions are containerized with Docker and scale with Kubernetes



How TML Processes JSON data in real-time?

- TML uses json paths (fields) to extract data from JSONs
- It processsees a group of JSONs in a sliding time window by using a field called **Jsoncriteria – which requires a user to indicate how they want to extract data from a grouped or aggregate json messages in sliding time window**

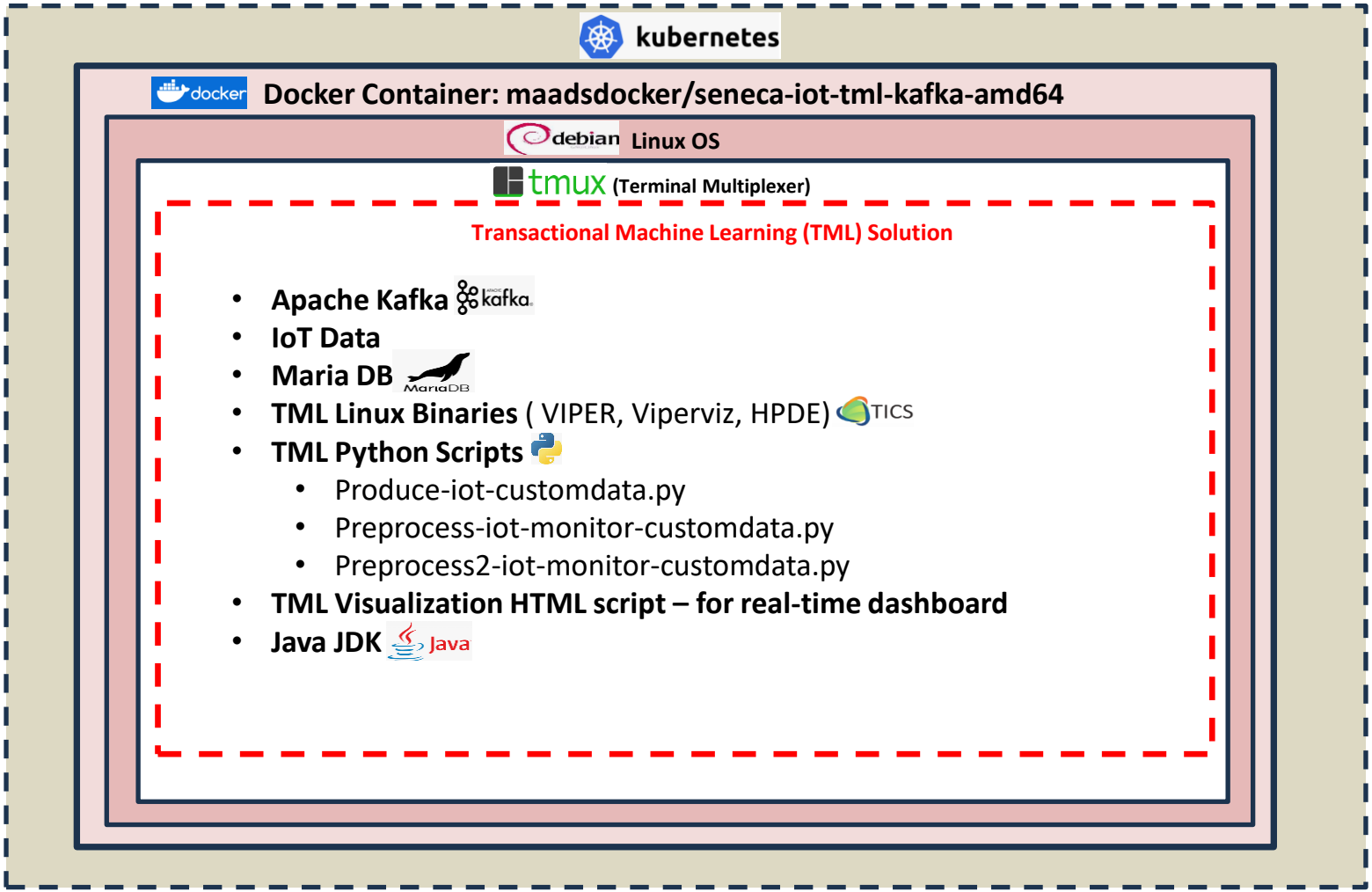
- **A jsoncriteria has 7 fields:**
- jsoncriteria=
- '**uid=**,filter:allrecords~\ ← **uid**: This is the json field to group by for example DSN or Device SerialNumber
- **subtopics=**~\ ← **subtopics:** This is the json field to the name of the field you want to process
- **values=**~\ ← **values:** This is the json field containing the value of the subtopic
- **identifiers=**~\ ← **identifier:** This is the json field containing any label or identifier for the values
- **datetime=**~\ ← **datetime:** This is the json field containing datetime, must be in UTC format – i.e. 2006-01-02T15:04:05
- **msgid=**~\ ← **msgid:** this is the json field containing further details about the values
- **latlong=**‘ ← This is json field contain latitude and longitude. You can use a “:” to combine lat:long

How TML Processes JSON data in real-time? Example

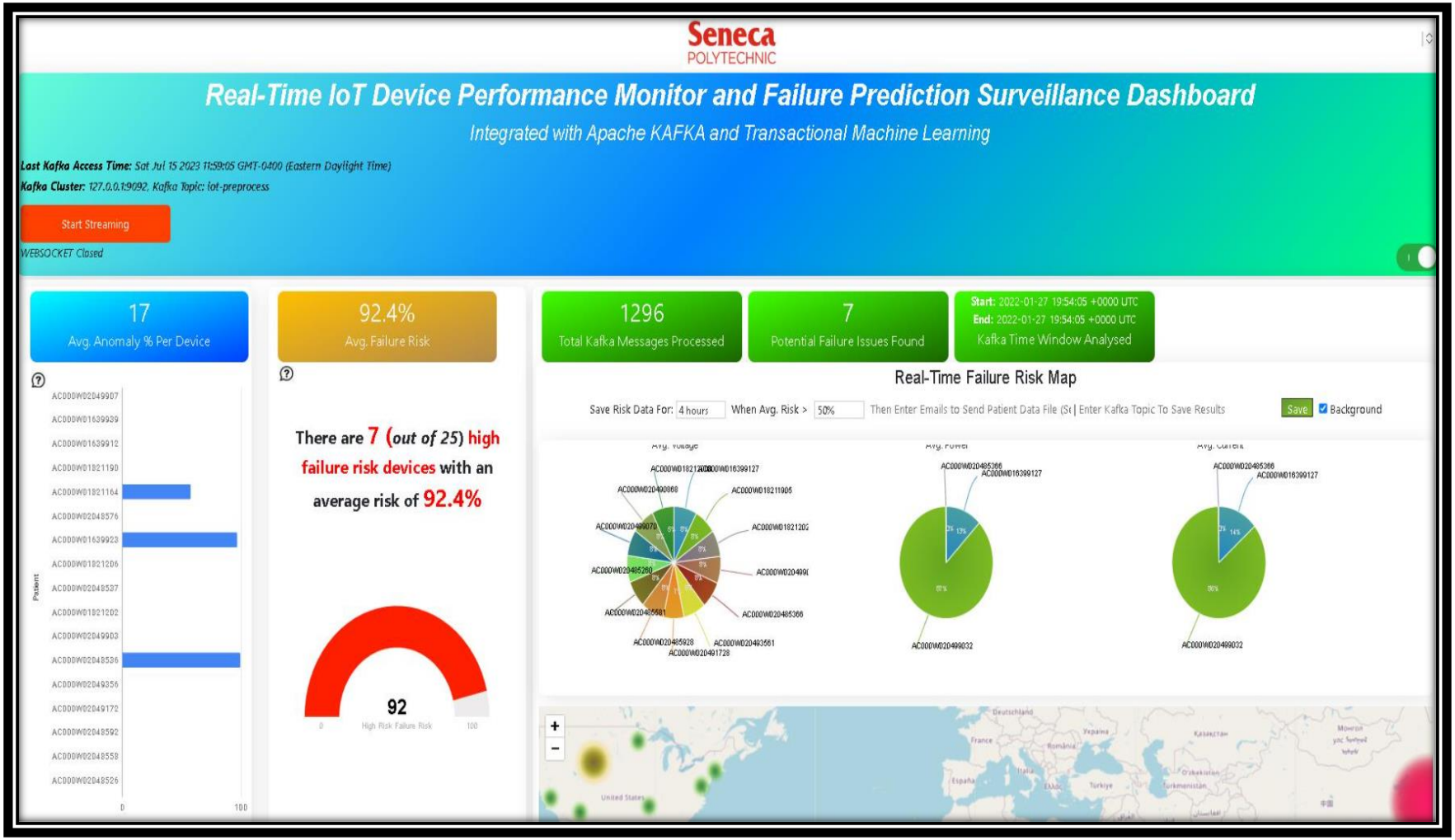
- If I have:

- **{"metadata":{"oem_id":"32795e59","oem_model":"SQR141U1XXW","dsn":"AC000W016399396","property_name":"Power","display_name":"Power (mW)","base_type":"integer","event_type":"datapoint"},"datapoint":{"id":"de3e8f0e-7faa-11ec-31cb-6b3a1eb15a96","updated_at":"2022-01-27T19:53:59Z","created_at":"2022-01-27T19:53:59Z","echo":false,"closed":false,"value":"0","metadata":{},"created_at_from_device":"2022-01-27T19:51:40Z","user_uuid":"f4d3b326-da9a-11eb-87af-0a580ae966af","discarded":false,"scope":"user","direction":"output"} , "lat": 29.22, "long": -141.22}**
- **{"metadata":{"oem_id":"32795e59","oem_model":"SQR141U1XXW","dsn":"AC000W016399396","property_name":"Current","display_name":"Current (mA)","base_type":"integer","event_type":"datapoint"},"datapoint":{"id":"de422f10-7faa-11ec-3925-f218ec2b4e1d","updated_at":"2022-01-27T19:53:59Z","created_at":"2022-01-27T19:53:59Z","echo":false,"closed":false,"value":"0","metadata":{},"created_at_from_device":"2022-01-27T19:51:40Z","user_uuid":"f4d3b326-da9a-11eb-87af-0a580ae966af","discarded":false,"scope":"user","direction":"output"} , "lat": 28.22, "long": -140.22}**
- **{"metadata":{"oem_id":"32795e59","oem_model":"SQR441U1XXW","dsn":"AC000W016399127","property_name":"EnergyUsed","display_name":"Energy Used (mWh)","base_type":"integer","event_type":"datapoint"},"datapoint":{"id":"de3f833c-7faa-11ec-b4ba-126e4b986056","updated_at":"2022-01-27T19:53:59Z","created_at":"2022-01-27T19:53:59Z","echo":false,"closed":false,"value":"2668340","metadata":{},"created_at_from_device":"2022-01-27T19:51:31Z","user_uuid":"c4d88504-64b4-11eb-902d-0a580ae9bff0","discarded":false,"scope":"user","direction":"output"} , "lat": 24.22, "long": -149.22}**
- **{"metadata":{"oem_id":"32795e59","oem_model":"SQR441U1XXW","dsn":"AC000W016399127","property_name":"EnergyUsed24hr","display_name":"Energy Used 24hr (mWh)","base_type":"integer","event_type":"datapoint"},"datapoint":{"id":"de475850-7faa-11ec-dfce-f2bfc16ef579","updated_at":"2022-01-27T19:53:59Z","created_at":"2022-01-27T19:53:59Z","echo":false,"closed":false,"value":"0","metadata":{},"created_at_from_device":"2022-01-27T19:51:31Z","user_uuid":"c4d88504-64b4-11eb-902d-0a580ae9bff0","discarded":false,"scope":"user","direction":"output"}, "lat": 23.22, "long": -143.22}**

- I can extract, group and process them by specifying the following JSON criteria:
- jsoncriteria=
- 'uid= metadata.dsn,filter:allrecords~\ ← **uid**: This is the json field to group by for example DSN or Device SerialNumber
- subtopics= metadata.property_name~\ ← **subtopics:** This is the json field to the name of the field you want to process
- values= datapoint.value~\ ← **values:** This is the json field containing the value of the subtopic
- identifiers= metadata.display_name~\ ← **identifier:** This is the json field containing any label or identifier for the values
- datetime= datapoint.updated_at~\ ← **datetime:** This is the json field in UTC format – i.e. 2006-01-02T15:04:05
- msgid= datapoint.id ~\ ← **msgid:** this is the json field containing further details about the values
- latlong=lat:long' ← This is json field contain latitude and longitude. You can use a ":" to combine lat:long


kubernetes
Docker Container: maadsdocker/seneca-iot-tml-kafka-amd64
Linux OS
tmux (Terminal Multiplexer)
Transactional Machine Learning (TML) Solution
Apache Kafka
IoT Data
Maria DB
TML Linux Binaries (VIPER, Viperviz, HPDE)
TICS
TML Python Scripts
Produce-iot-customdata.py
Preprocess-iot-monitor-customdata.py
Preprocess2-iot-monitor-customdata.py
TML Visualization HTML script – for real-time dashboard
Java JDK
Java

TML IOT Dashboard



TML LOG STREAMING

Seneca
POLYTECHNIC

VIPER LOG STREAM: *viperlogs*

Last Kafka Access Time: Sat Aug 19 2023 11:16:11 GMT-0400 (Eastern Daylight Time)

Kafka Cluster: 127.0.0.1:9092, Kafka Topic: viperlogs

Stop Streaming

Download Table as CSV | Download JSON

Status: WEBSOCKET OPEN..Receiving Kafka messages from VIPERviz (RUNNING..)

	Generated	Message	Service	Service Host	Service Port	Kafka Cluster	Offset	Partition
1	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found=Voltage. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,845	0
2	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found=Voltage. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,851	0
3	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found=Voltage. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,854	0
4	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found=Voltage. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,855	0
5	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found=EnergyUsed24hr. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,856	0
6	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found=EnergyUsed. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,846	0
7	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found=EnergyUsed. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,852	0
8	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found=EnergyUsed. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,853	0
9	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found=Current. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,849	0
10	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5368 UTC] INFO [parsesubtopics Record(s) found in Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,859	0
11	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5367 UTC] INFO [parsesubtopics Record(s) found=Voltage. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,842	0
12	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5367 UTC] INFO [parsesubtopics Record(s) found=Voltage. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,850	0
13	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5367 UTC] INFO [parsesubtopics Record(s) found=Power. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,829	0
14	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5367 UTC] INFO [parsesubtopics Record(s) found=Power. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,832	0
15	2023-08-19T15:16:02.629+00:00	[Sat, 19 Aug 2023 15:16:01.5367 UTC] INFO [parsesubtopics Record(s) found=Power. In Topic=iot-mainstream - Viper writing results to preprocesstopic=iot-preprocess. YOU ARE STREAMING!]	VIPER	172.17.0.2	41,575	127.0.0.1:9092	16,835	0

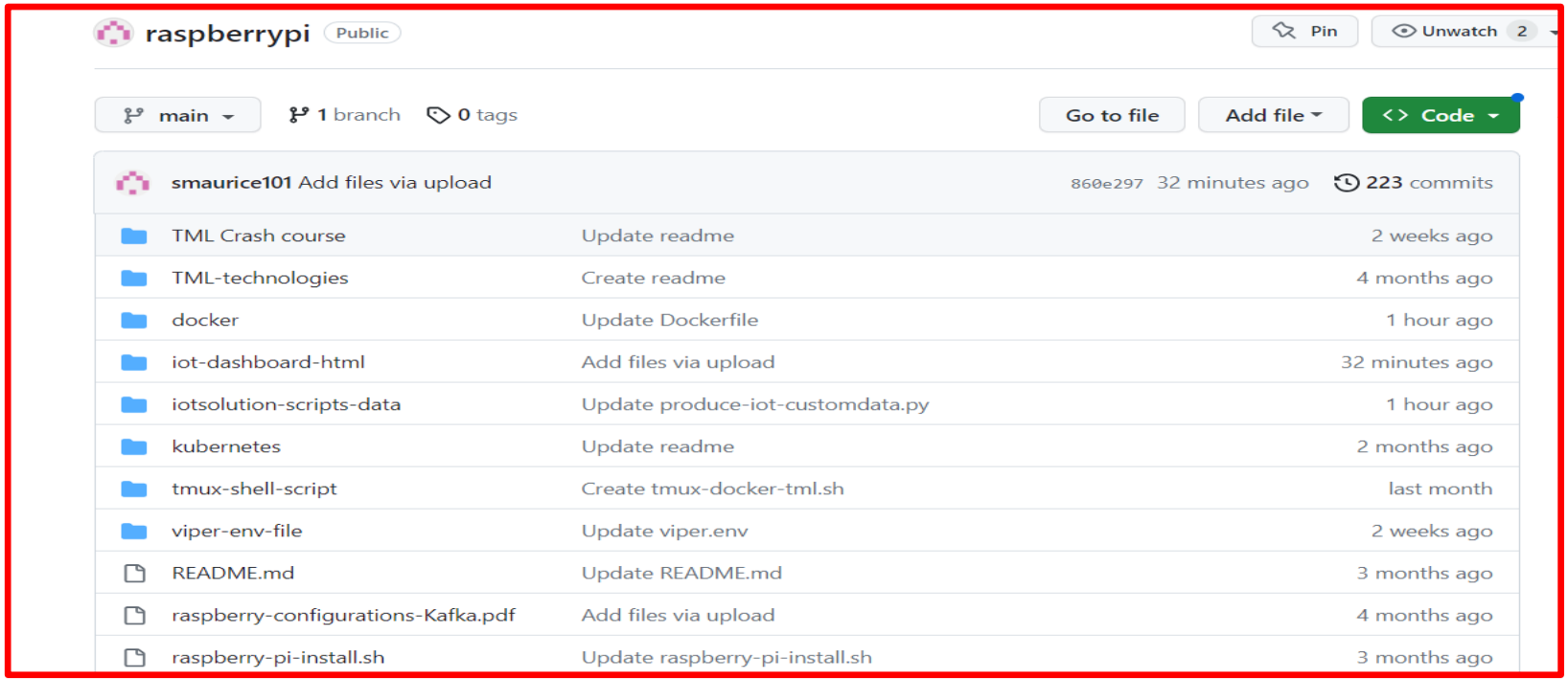
STEPS TO RE-CREATING TML IOT SOLUTION FOR STUDENTS

TML Student Solution: Re-Creating TML Solution

- Students can build their own streaming solution
- Before building your own solution – students re-create the solution **in Slide 6 and 7 to learn the components**

STEPS TO TAKE TO RE-CREATE IOT SOLUTION:

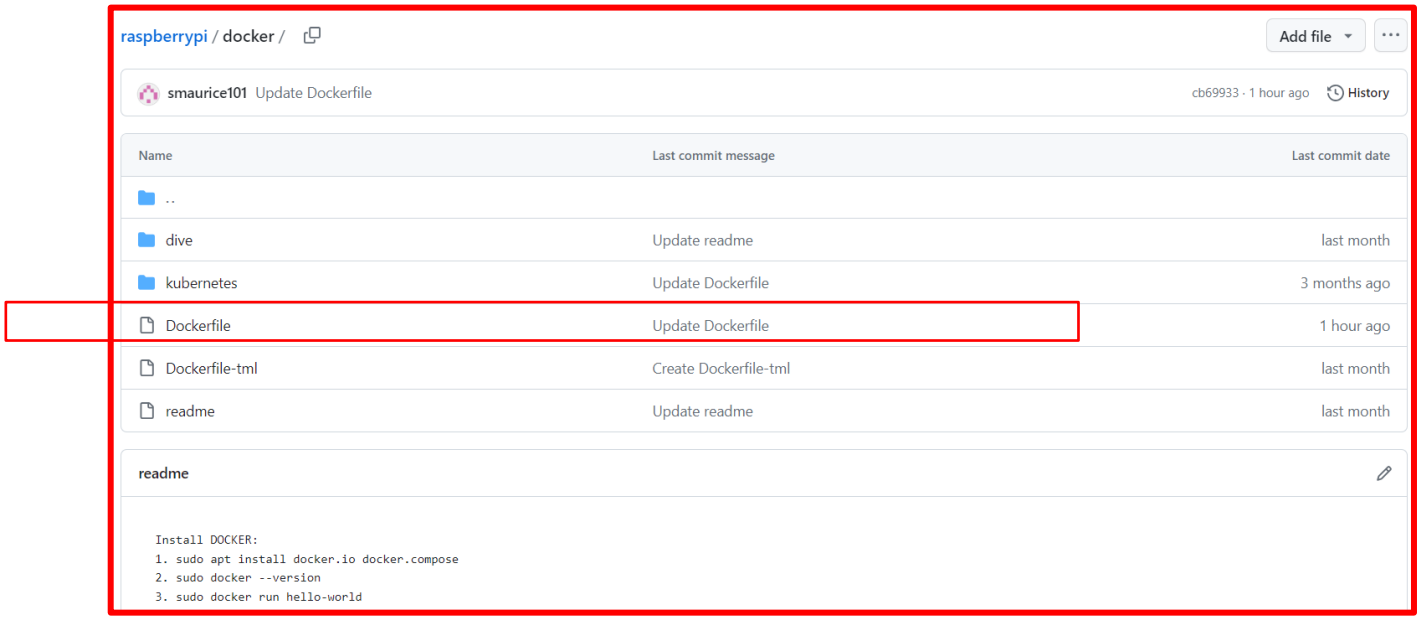
1. **Create your own Git Repository (DO NOT ADD readme.md) by cloning:**
   a) **https://github.com/smaurice101/raspberrypi.git (you should see image below in YOUR OWN Github Account)**



TML Student Solution: Re-Creating TML Solution

STEPS TO TAKE TO RE-CREATE IOT SOLUTION:

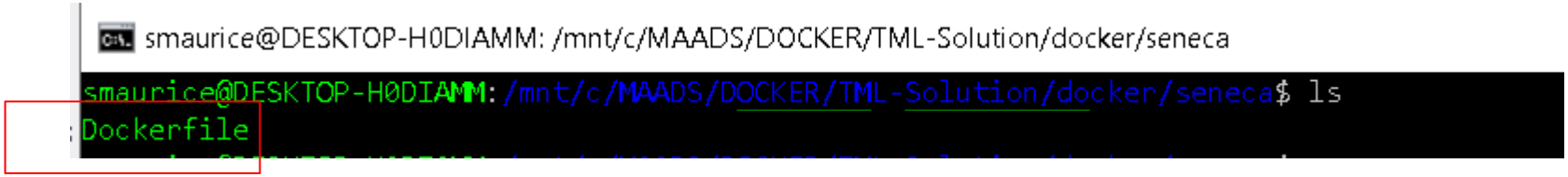
2. **Install docker in Linux VM or WSL (WSL is recommended):**
    1. **Run:** sudo apt install docker.io docker.compose
3. In your raspberry pi repo – GOTO **docker** folder
    1. **Copy** the **Dockerfile to your LOCAL computer** **(NOTE: File name MUST be exactly Dockerfile – no file extensions)**



TML Student Solution: Re-Creating TML Solution

STEPS TO TAKE TO RE-CREATE IOT SOLUTION:

4. Go to the location where you stored Dockerfile on your LOCAL computer
    1. **Confirm Dockerfile exists**

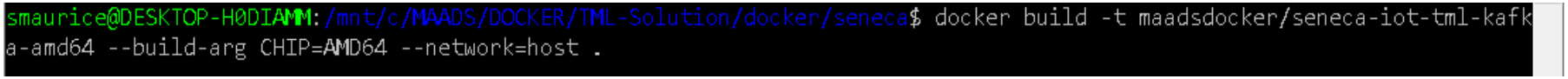


TML Student Solution: Re-Creating TML Solution

STEPS TO TAKE TO RE-CREATE IOT SOLUTION:

5. **Create** a Dockerhub Account: **https://hub.docker.com/**
   - My account is: **maadsdocker** **(REPLACE WITH YOUR OWN DOCKER HUB ACCOUNT)**

6. **RUN** docker build in the SAME folder where Dockerfile is saved:
   1. **Run: docker build -t maadsdocker/seneca-iot-tml-kafka-amd64 --build-arg CHIP=AMD64 --network=host .**
   2. **NOTE: The " . " at the end – this must be there**
   3. **NOTE: DO NOT USE YOUR GITHUB Account in docker build command**
   4. You can choose any container name you wish

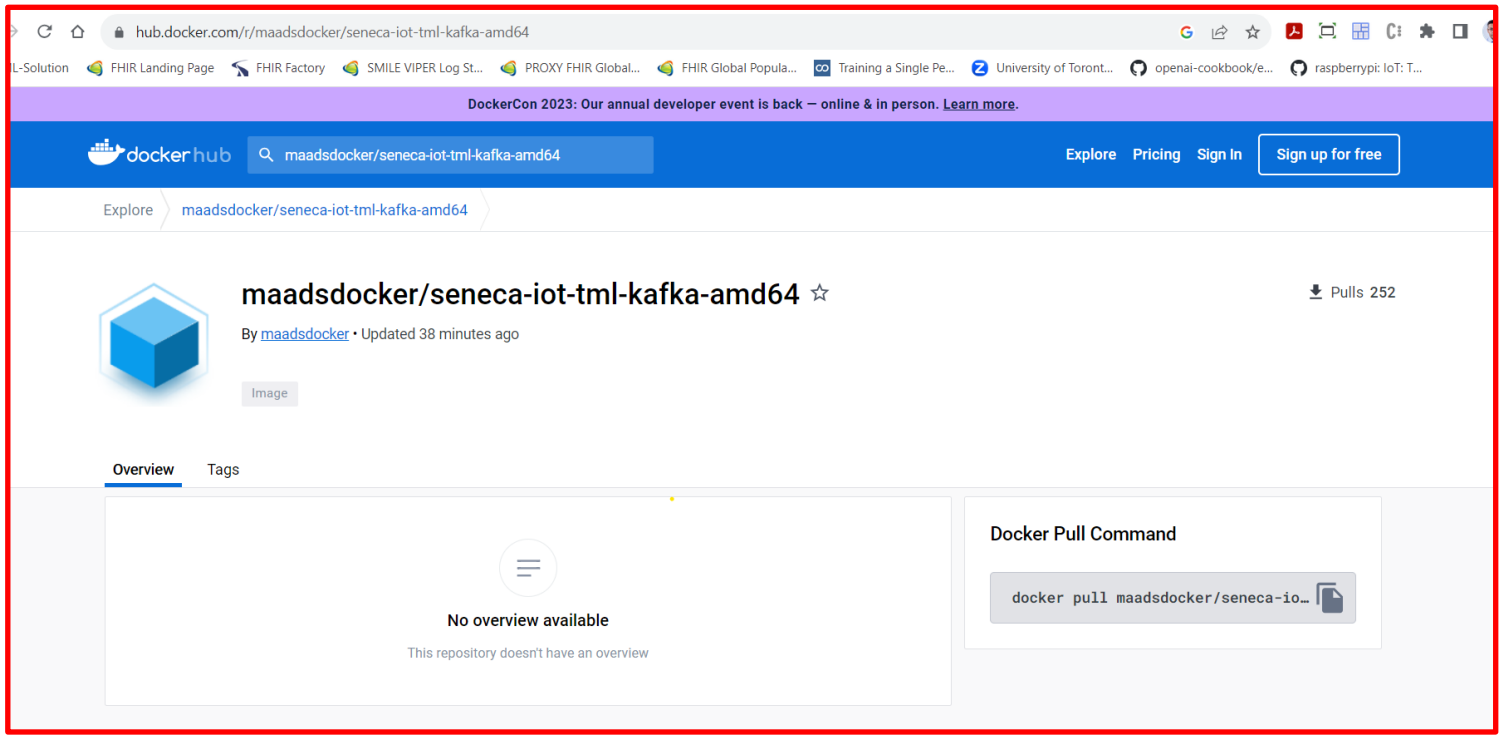


TIP: After you did your normal build – you can use the FASTER Docker Build command:
docker build -t maadsdocker/seneca-iot-tml-kafka-amd64 --build-arg CHIP=AMD64 --build-arg CACHEBUST=$(date +%s) --network=host .

TML Student Solution: Re-Creating TML Solution

STEPS TO TAKE TO RE-CREATE IOT SOLUTION:

7. If your Docker build is successful you now have a Docker Container called: **seneca-iot-tml-kafka-amd64**
8. **You can now PUSH your container to your Docker Hub account:**
   - **Run: docker push maadsdocker/seneca-iot-tml-kafka-amd64**
9. If your Push is successful you will see your container in Docker Hub under your account



TML Student Solution: Re-Creating TML Solution

STEPS TO TAKE TO RE-CREATE IOT SOLUTION:

10. **Run your container:**
    1. **Run: docker run -p 9005:9005 maadsdocker/seneca-iot-tml-kafka-amd64**
    2. **NOTE: The " -p" this will FORWARD Port 9005 and map HOST Port 9005 to CONTAINER Port 9005**
    3. **You MUST port forward for TML Dashboard to work**

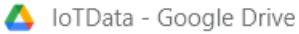
```
smaurice@DESKTOP-H0DIAMM:/mnt/c/MAADS/DOCKER/TML-Solution/docker/seneca$ docker run -p 9005:9005 maadsdocker/seneca-iot-
tml-kafka-amd64
```

TML Student Solution: Re-Creating TML Solution

STEPS TO TAKE TO RE-CREATE IOT SOLUTION:

11. RAW DATA FOR SOLUTION:

1. https://docs.google.com/uc?export=download&id=1yRgDYrWnHu74NYX9GMAVDjR10ZyfoZvh



Students can change this path to their own data.

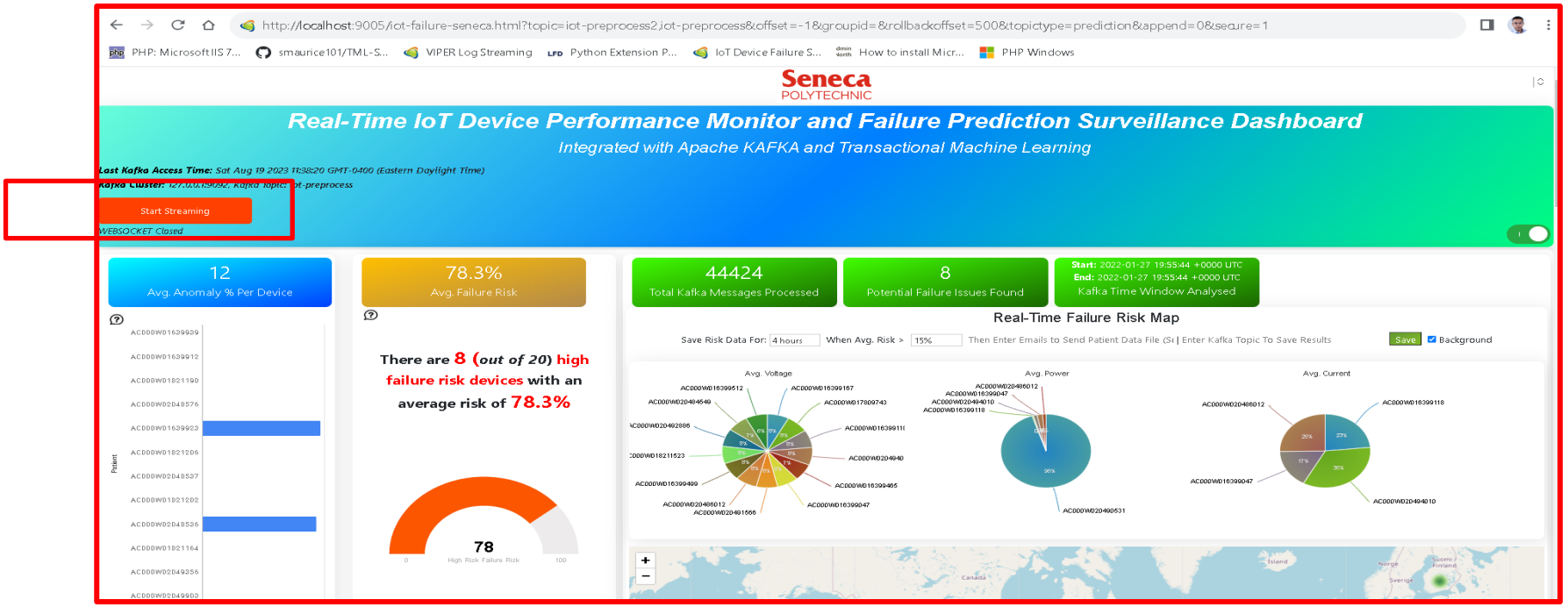
- Insert your file ID into this URL (https://drive.google.com/uc?export=download&id=), then surround the URL with quotes so that Bash doesn't misinterpret the &, like so:
- Get file ID by going to share -> copy link -> then get id from COpy link: https://drive.google.com/file/d/1mGcHQC7IxiTFYeUSFof3fDppVSC4rq3v/view?usp=drive_link
- Specifically, you will need to use this URL: https://drive.google.com/uc?export=download&id=
- YOU WILL NEED TO ADD THE id FOR YOUR FILE - THIS CAN BE FOUND BY RIGH-CLICKING ON YOUR FILE IN GOOGLE DRIVE - CHOOSE SHARE -> THEN COPY LINK -THEN COPY THE TEXt BETWEEN /d and /view.
- For example, here is a similar link: https://drive.google.com/file/d/1mGcHQC7IxiTFYeUSFof3fDppVSC4rq3v/view?usp=drive_link
- The id is 1mGcHQC7IxiTFYeUSFof3fDppVSC4rq3v. The download url will be: https://drive.google.com/uc?export=download&id=1mGcHQC7IxiTFYeUSFof3fDppVSC4rq3v

This url will need to be replaced in Dockerfile to download your IoTData.zip

TML Student Solution: Re-Creating TML Solution

STEPS TO TAKE TO RE-CREATE IOT SOLUTION:

12. **RUN TML DASHBOARD:**
    1. **Open a Browser on the machine running the container**
    2. ***PASTE This URL in your browser:*** ***http://localhost:9005/iot-failure-seneca.html?topic=iot-preprocess2,iot-preprocess&offset=-1&groupid=&rollbackoffset=500&topictype=prediction&append=0&secure=1***
    3. ***CLICK START STREAMING BUTTON***

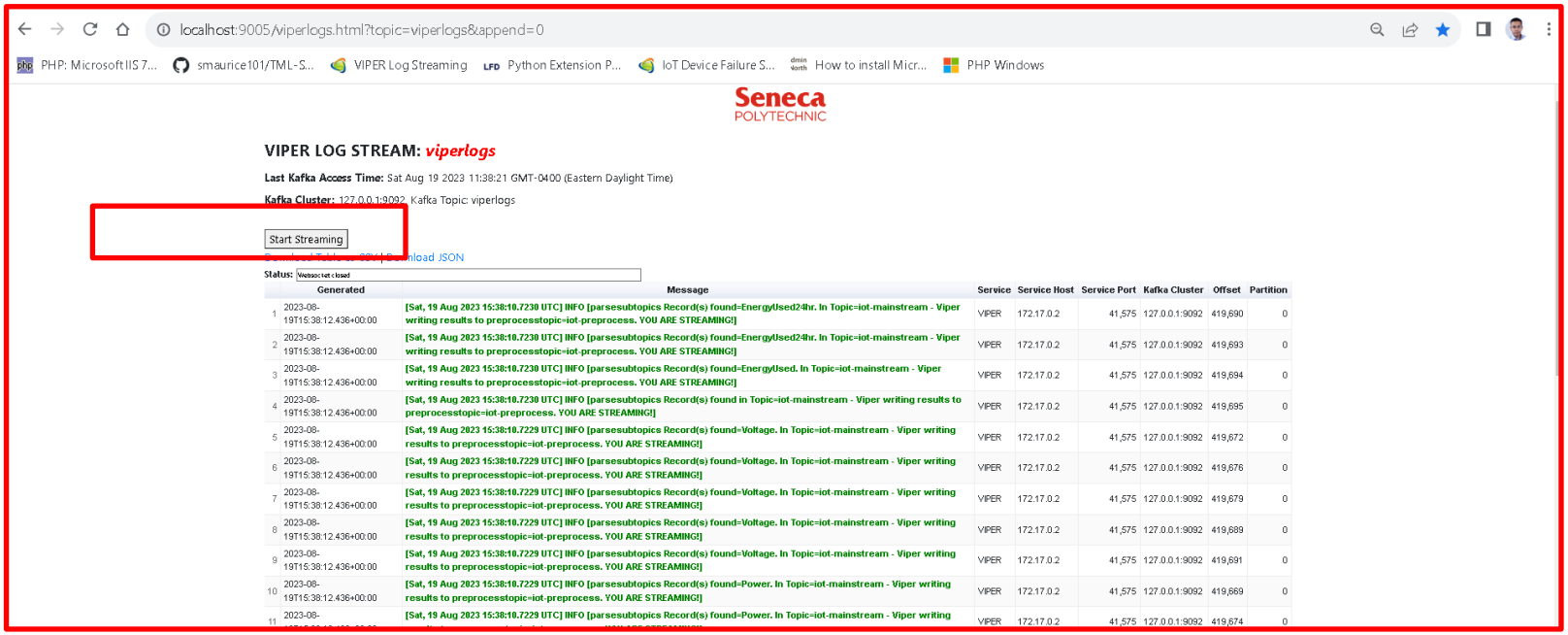


TML Student Solution: Re-Creating TML Solution

STEPS TO TAKE TO RE-CREATE IOT SOLUTION:

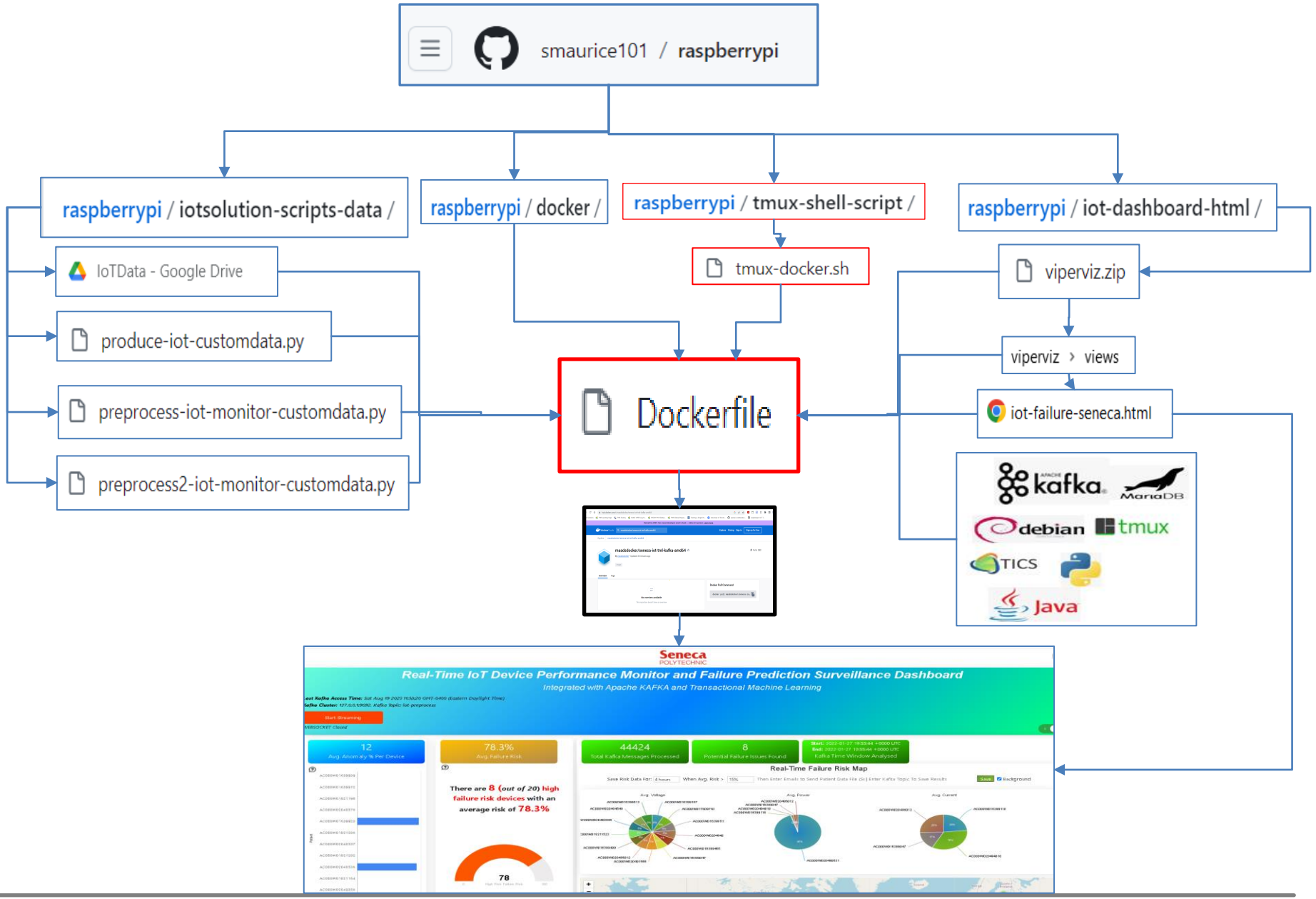
13. RUN TML LOG STREAMING:

1. **Open a Browser on the machine running the container**
2. ***PASTE This URL in your browser:*** ***http://localhost:9005/viperlogs.html?topic=viperlogs&append=0***
3. ***Click Start Streaming button***



YOU ARE NOW STREAMING!

CREATING YOUR OWN TML SOLUTION PROCESS



HAPPY STREAMING WITH TML!