[tml-multi-agenticai-iot-3f10-ml_agenticai] Details

Generated On: 2025-09-28 23:50:27 UTC

TML Solution DAG Parameters' Details: User Chosen Parametets

STEP 1: Get TML Core Params: tml_system_step_1_getparams_dag

User Parameter	Chosen Value
solutionname	tml-multi-agenticai-iot-3f10-ml_agenticai
solutiontitle	TML Multi-Agentic AI Solution for IoT
solutiondescription	This is an awesome TML Multi-Agentic AI real-time solution for IoT device monitoring built by TSS
brokerhost	127.0.0.1
brokerport	9092
cloudusername	None
ingestdatamethod	LOCALFILE

STEP 2: Create Kafka Topics: tml_system_step_2_kafka_createtopic_dag

User Parameter	Chosen Value
companyname	Otics
myname	Sebastian
myemail	Sebastian.Maurice
mylocation	Toronto
replication	1
numpartitions	1
enabletls	1
microserviceid	
raw_data_topic	iot-raw-data,agent-responses,team-lead-responses ,supervisor-responses,all-agents-responses
preprocess_data_topic	iot-preprocess,iot-preprocess2
ml_data_topic	ml-data
prediction_data_topic	iot-ml-prediction-results-output

STEP 3: Produce to Kafka Topics

User Parameter	Chosen Value
PRODUCETYPE	LOCALFILE
inputfile	/rawdatademo/IoTData.txt

TOPIC	iot-raw-data
PORT	_39399
IDENTIFIER	TML Multi-Agentic Solution,/rawdatademo/IoTData.txt
HTTPADDR	https://
FROMHOST	('seb', '127.0.1.1')
TOHOST	0.0.0.0
CLIENTPORT	Not Applicable
TSS_CLIENTPORT	Not Applicable
TML_CLIENTPORT	Not Applicable
docfolder	
doctopic	
chunks	3000
docingestinterval	0

STEP 4: Preprocesing Data: tml-system-step-4-kafka-preprocess-dag

User Parameter	Chosen Value
raw_data_topic	iot-raw-data,agent-responses,team-lead-responses,supervisor-responses,all-agents-responses
preprocess_data_topic	iot-preprocess,iot-preprocess2
preprocessconditions	
delay	70
maxrows	800
array	0
saveasarray	1
topicid	-999
rawdataoutput	1
asynctimeout	120
timedelay	0
preprocesstypes	anomprob,trend,avg
pathtotmlattrs	--pathtotmlattrs--
identifier	IoT TML Multi-Agentic AI device performance and failures
jsoncriteria	uid=metadata.dsn,filter:allrecords~subtopics=metadata.property_name~values=datapoint.value~identifiers=metadata.display_name~datetime=datapoint.updated_at~msgid=datapoint.id~latlong=lat:long

STEP 4a: Preprocesing Data: tml-system-step-4a-kafka-preprocess-dag

User Parameter	Chosen Value

raw_data_topic	--raw_data_topic1--
preprocess_data_topic	--preprocess_data_topic1--
preprocessconditions	--preprocessconditions1--
delay	--delay1--
maxrows	--maxrows1--
array	--array1--
saveasarray	--saveasarray1--
topicid	--topicid1--
rawdataoutput	--rawdataoutput1--
asynctimeout	--asynctimeout1--
timedelay	--timedelay1--
preprocesstypes	--preprocesstypes1--
pathtotmlattrs	--pathtotmlattrs1--
identifier	--identifier1--
jsoncriteria	--jsoncriteria1--

STEP 4b: Preprocesing Data: tml-system-step-4b-kafka-preprocess-dag

User Parameter	Chosen Value
raw_data_topic	--raw_data_topic2--
preprocess_data_topic	--preprocess_data_topic2--
preprocessconditions	--preprocessconditions2--
delay	--delay2--
maxrows	--maxrows2--
array	--array2--
saveasarray	--saveasarray2--
topicid	--topicid2--
rawdataoutput	--rawdataoutput2--
asynctimeout	--asynctimeout2--
timedelay	--timedelay2--
preprocesstypes	--preprocesstypes2--
pathtotmlattrs	--pathtotmlattrs2--
identifier	--identifier2--
jsoncriteria	--jsoncriteria2--

STEP 4c: Preprocesing Data: tml-system-step-4c-kafka-preprocess-dag

User Parameter	Chosen Value
raw_data_topic	--raw_data_topic3--
preprocess_data_topic	--preprocess_data_topic3--

delay	--delay3--
maxrows	--maxrows3--
array	--array3--
saveasarray	--saveasarray3--
topicid	--topicid3--
rawdataoutput	--rawdataoutput3--
asynctimeout	--asynctimeout3--
timedelay	--timedelay3--
searchterms	--rtmssearchterms--
rtmsstream	--rtmsstream--
identifier	--identifier3--
rememberpastwindows	--rememberpastwindows--
patternwindowthreshold	--patternwindowthreshold--
localsearchtermfolder	--localsearchtermfolder--
localsearchtermfolderinterval	--localsearchtermfolderinterval--
rtmsscorethreshold	--rtmsscorethreshold--
rtmsscorethresholdtopic	--rtmsscorethresholdtopic--
attackscorethreshold	--attackscorethreshold--
attackscorethresholdtopic	--attackscorethresholdtopic--
patternscorethreshold	--patternscorethreshold--
patternscorethresholdtopic	--patternscorethresholdtopic--
rtmsfoldername	--rtmsfoldername--
rtmsmaxwindows	--rtmsmaxwindows--
RTMS Output Github Link	Output Data URL

STEP 5: Entity Based Machine Learning : tml-system-step-5-kafka-machine-learning-dag

User Parameter	Chosen Value
preprocess_data_topic	iot-preprocess,iot-preprocess2
ml_data_topic	ml-data
modelruns	100
offset	-1
islogistic	1
networktimeout	600
modelsearchtuner	90
processlogic	classification_name=failure_prob:Power_preprocessed_AnomProb=55,n
dependentvariable	failure
independentvariables	Power_preprocessed_AnomProb

rollbackoffsets	500
topicid	-999
consumefrom	
fullpathtotrainingdata	/Viper-ml/viperlogs/iotlogistic
transformtype	
sendcoefto	
coeftoprocess	
coefsubtopicnames	
ML Output Github Link	Output Data URL

STEP 6: Entity Based Predictions: tml-system-step-6-kafka-predictions-dag

User Parameter	**Chosen Value**
preprocess_data_topic	iot-preprocess,iot-preprocess2
ml_prediction_topic	iot-ml-prediction-results-output
streamstojoin	Power_preprocessed_AnomProb
inputdata	
consumefrom	ml-data
offset	-1
delay	70
usedeploy	1
networktimeout	600
maxrows	800
topicid	-999
pathtoalgos	/Viper-ml/viperlogs/iotlogistic

STEP 7: Real-Time Visualization: tml-system-step-7-kafka-visualization-dag

User Parameter	**Chosen Value**
vipervizport	49689
topic	all-agents-responses,iot-preprocess,iot-ml-prediction-results-output
dashboardhtml	dashboard-agenticai.html
secure	1
offset	-1
append	0
chip	amd64
rollbackoffset	400

STEP 8: tml_system_step_8_deploy_solution_to_docker_dag

User Parameter	Chosen Value
Docker Container	maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64 (https://hub.docker.com/r/maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64)
Docker Run Command	**docker run -d --net=host -p 5050:5050 -p 4040:4040 -p 6060:6060** --env TSS=0 --env SOLUTIONNAME=tml-multi-agenticai-iot-3f10-ml_agenticai --env SOLUTIONDAG=solution_preprocessing_ml_agenticai_dag-tml-multi-agenticai-iot-3f10 --env GITUSERNAME=<Enter Github Username> --env GITPASSWORD='<Enter Github Password>' --env GITREPOURL=<Enter Github Repo URL> --env SOLUTIONEXTERNALPORT=5050 -v /var/run/docker.sock:/var/run/docker.sock:z -v /your_localmachine/foldername:/rawdata:z --env CHIP=amd64 --env SOLUTIONAIRFLOWPORT=4040 --env SOLUTIONVIPERVIZPORT=6060 --env DOCKERUSERNAME='' --env EXTERNALPORT=39399 --env KAFKABROKERHOST=127.0.0.1:9092 --env KAFKACLOUDUSERNAME='<Enter API key>' --env KAFKACLOUDPASSWORD='<Enter API secret>' --env SASLMECHANISM=PLAIN --env VIPERVIZPORT=49689 --env MQTTUSERNAME='' --env MQTTPASSWORD='' --env AIRFLOWPORT=9000 --env READTHEDOCS='<Enter Readthedocs token>' maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64

STEP 9: tml_system_step_9_privategpt_qdrant_dag

User Parameter	Chosen Value
PrivateGPT Container	--pgptcontainername--
PrivateGPT Run Command	--privategptrun--
Qdrant Container	--qdrantcontainer--
Qdrant Run Command	--qdrantrun--
Consumefrom	
pgpt_data_topic	--pgpt_data_topic--
offset	-1
rollbackoffset	400
topicid	-999
enabletls	1
partition	--partition--
prompt	--prompt--

context	--context--
jsonkeytogather	--jsonkeytogather--
keyattribute	--keyattribute--
keyprocesstype	--keyprocesstype--
vectordbcollectionname	--vectordbcollectionname--
concurrency	--concurrency--
CUDA_VISIBLE_DEVICES	--cuda--
pgpthost	--pgpthost--
pgptport	--pgptport--
hyperbatch	--hyperbatch--
docfolder	--docfolder--
docfolderingestinterval	--docfolderingestinterval--
useidentifierinprompt	--useidentifierinprompt--
searchterms	--searchterms--
streamall	--streamall--
temperature	--temperature--
vectorsearchtype	--vectorsearchtype--
llm	--llmmodel--
embedding	--embedding--
vectorsize	--vectorsize--
contextwindowsize	--contextwindowsize--
vectordimension	--vectordimension--
mitrejson	--mitrejson--

STEP 9b: tml_system_step_9b_agenticai_dag

User Parameter	Chosen Value
rollbackoffset	1
ollama-model	llama3.1
deletevectordbcount	10
vectordbpath	/rawdata/vectordb
temperature	0.1
topicid	-999
enabletls	1
partition	-1
vectordbcollectionname	tml-llm-model-v2
ollamacontainername	maadsdocker/tml-privategpt-with-gpu-nvidia-amd64-llama3-tools
mainip	http://127.0.0.1
mainport	11434

embedding	nomic-embed-text
agenttopic	agent-responses
agents_topic_prompt	iot-preprocess:Are there any issues or anomalies in the JSON data values in the hyperprediction field? Specifically, the hyprepreddiction field indicates the value of the Preprocesstype field, if the Preprocesstype is Avg, then the hyprepredictions are the average values of the first value in the Identifier field, which is Current for the device name in the mainuid field. For example, if the hypreprediction=154646, and if Preprocesstype=Avg, and the first value in Identifier=Current, and mainuid=AC000W020496398 then 154646 is the average of electrical current for device name AC000W020496398. Determine if the trends electrical current values are normal or abnormal by looking at other similar jsons. Do NOT focus on data quality, just focus on the hyperprediction, Identifier, and mainuid fields.
teamlead_topic	team-lead-responses
teamleadprompt	Are there any issues or major concerns in the data? The data are from IoT devices that are being monitored by individual agents. If you find issues or concerns, then highlight the devices name (i.e. in the mainuid field) with details on whether the device failure probabilities are increasing or greater than 0.70.
supervisor_topic	supervisor-responses
supervisorprompt	Are there any major issues or concerns in the data? This data is IoT data being monitored for potential failure from IoT devices. If you find a major concern or major issues in the failure probabilities of IoT devices indicated in the hyperprediction values, then use the send_email tool to send an email message that highlights devices with the issues that need investigation Do NOT send too many emails, and do not send duplicate emails with same device names.
agenttoolfunctions	**send_email:send_email: You are an email-sending agent. Use smtp parameters to send emails when there is an anomaly in the data, make sure to** indicate the device name in the mainuid field. do not write a smtp script, actually send the email using the SMTP parameters smtp_server='' smtp_port=0 username='' password='' sender='' recipient='' subject='' body=''
agent_team_supervisor_topic	all-agents-responses
concurrency	2
CUDA_VISIBLE_DEVICES	0

STEP 10: tml_system_step_10_documentation_dag

User Parameter	Chosen Value
Solution Documentation URL	https://tml-multi-agenticai-iot-3f10-ml_agenticai.readthedocs.io

[tml-multi-agenticai-iot-3f10-ml_agenticai] Operating Details

Generated On: 2025-09-28 23:50:27 UTC

Note

THIS DOCUMENTATION CREATION WAS AUTOMATICALLY TRIGGERED BY: TSS Development Environment Container

If this documentation was triggered by the TSS - then your solution is running in the TSS Development environment. All ports and links will point to your TSS development environment.

If this documentation was triggered by your TML solution - then your solution is running in your TML solution container. All ports and links will point to your TML solution container.

If you have NOTHING running - most of the links and ports below WILL NOT WORK.

If you have BOTH RUNNING - all of the links and ports below WILL WORK.

Also, this documentation is updated by the **LATEST** run of either the TSS or TML.

Important

These are the operating details for your TML solution.

It is important to note that this documentation will

dynamically update with the **Runtime Ports** for your solution.

The ports below were valid for the last run of your solution, and will

not be valid if your [tml-multi-agenticai-iot-3f10-ml_agenticai] container is NOT running.

Tip

You must have your [tml-multi-agenticai-iot-3f10-ml_agenticai] container running before connecting to the Visualization and Airflow URLs.

Github Logs

This is your main TSS Github logs. All TSS processes are committed to Github and logged.

Important

https://github.com/smaurice101/raspberrypitss/blob/main/tml-airflow/logs/logs.txt

TSS Docker Run Command

This is the TML Solution Studio Docker Run command. Note for MAC users change amd64 to arm64 in the container name.

Note

```
docker run -d --net=host --env AIRFLOWPORT=9000 -v <change to your local
folder>:/dagslocalbackup:z -v /var/run/docker.sock:/var/run/docker.sock:z -v
/your_localmachine/foldername:/rawdata:z --env
GITREPOURL=https://github.com/smaurice101/raspberrypitss.git --env CHIP=amd64 --env TSS=1
--env SOLUTIONNAME=TSS --env EXTERNALPORT=39399 --env VIPERVIZPORT=49689 --env
GITUSERNAME='smaurice101' --env DOCKERUSERNAME='maadsdocker' --env
MQTTUSERNAME='smaurice' --env KAFKACLOUDUSERNAME='' --env
KAFKACLOUDPASSWORD='<Enter your API secret>' --env READTHEDOCS='<Enter your
readthedocs token>' --env GITPASSWORD='<Enter personal access token>' --env
DOCKERPASSWORD='<Enter your docker hub password>' --env MQTTPASSWORD='<Enter your
mqtt password>' --env UPDATE=1 maadsdocker/tml-solution-studio-with-airflow-amd64
```

TSS Docker Run Command: Parameter Explanation

Parameter	Explanation
docker	This calls the docker engine
-d	This runs your container in **detached** mode
--net=host	This give your container access to your host operating system
-v	This stands for **volume mapping**. It maps a local folder in your host machine to the folder in the container. The value **z** means the container has **shared access** to your local folder. For example, -v /mylocal/folder:/dagslocalbackup:z, means map /mylocal/folder (on my host machine) to **/dagslocalbackup** in the contaner. This allows files generated in the container to be automatically written to your local folder.

--env GITREPOURL	This is your Github repo, that you cloned from **https://github.com/smaurice101/raspberrypi**
--env CHIP=AMD64	This is the chip if your are running the TSS on windows/linux. If you are running MAC, use **CHIP=ARM64**
--env TSS=1	This is the TSS value and MUST be 1.
--env AIRFLOWPORT=9000	This is the airflow port for TSS. Connect to TSS from your browser: http://localhost:9000
--env SOLUTIONNAME=TSS	This is the solution name.
--env VIPERVIZPORT=49689	This is the port the Viperviz binary will listen on for connections. Note: If VIPERVIZPORT=-1, a random free port is selected by TSS.
--env EXTERNALPORT=39399	This is the external port that will be assigned to your TSS solution for external access. You will need this port in the REST, and gRPC clients. Note: if EXTERNALPORT=-1, TSS will choose a free port randomly. This external port is used by Viper binary: Viper will be listening on this port for a connection as shown here: :ref:`Your Solution TML Binaries` In the TMUX window **Viper-produce**: :ref:`Your Solution TMUX Windows`
--env READTHEDOCS	This is the readthedocs API token you created. Refer to: Set up readthedocs
--env GITUSERNAME	This is your Githib username.
--env GITPASSWORD	This is the Github Personal Access Token you created. Refer to: Creating Github Token:
--env DOCKERUSERNAME	This is your Docker Hub username.
--env DOCKERPASSWORD	This is your Docker Hub password.
--env MQTTUSERNAME	This is your MQTT username. See Set up HiveMQ
--env MQTTPASSWORD	This is your MQTT password.
--env KAFKACLOUDUSERNAME	This is your API key from Confluent Cloud
--env KAFKACLOUDPASSWORD	This is your API Secret from Confluent Cloud.
maadsdocker/tml-solution-studio-with-airflow-amd64	This is the TSS container name for AMD64 If using MAC/Unix use: maadsdocker/tml-solution-studio-with-airflow-arm64

TSS Dashboard URL

This is the visualization URL for your TSS dashboard. Note ports may change at runtime. The solution documentation will update automatically.

Important

http://localhost:49689/dashboard-agenticai.html?topic=all-agents-responses,iot-preprocess,iot-ml-prediction-results-output&offset=-1&groupid=&rollbackoffset=400&topictype=prediction&append=0&secure=1

TSS Airflow Port

This is the airflow port in your TSS solution container.

It can be accessed by entering: http://localhost:9000

TSS Log File Dashboard

This is the log file dashboard for your development TML solution running in TSS.

Important

http://localhost:49689/viperlogs.html?topic=viperlogs&append=0

Note

It should be noted that your solution is running in the TSS Development Environment. This gives TML developers a very good way to test their TML solutions before deploying it.

The solution ports and links below may not work because they will require your to RUN your solution container first. After, you run your solution container the links and ports will automatically update in the documentation.

Your Solution Docker Container

Important

maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64 (https://hub.docker.com/r/maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64)

Your Solution Docker Run Command

This is the Docker Run command for your solution container. Note ports may change at runtime. The solution documentation will update automatically.

```
docker run -d --net=host -p 5050:5050 -p 4040:4040 -p 6060:6060 \
       --env TSS=0 \
        --env SOLUTIONNAME=tml-multi-agenticai-iot-3f10-ml_agenticai \
        --env SOLUTIONDAG=solution_preprocessing_ml_agenticai_dag-tml-multi-agenticai-iot-3f10 \
        --env GITUSERNAME=<Enter Github Username> \
        --env GITPASSWORD='<Enter Github Password>' \
        --env GITREPOURL=<Enter Github Repo URL> \
        --env SOLUTIONEXTERNALPORT=5050 \
        -v /var/run/docker.sock:/var/run/docker.sock:z \
        -v /your_localmachine/foldername:/rawdata:z \
        --env CHIP=amd64 \
        --env SOLUTIONAIRFLOWPORT=4040 \
        --env SOLUTIONVIPERVIZPORT=6060 \
        --env DOCKERUSERNAME='' \
        --env EXTERNALPORT=39399 \
        --env KAFKABROKERHOST=127.0.0.1:9092 \
        --env KAFKACLOUDUSERNAME='<Enter API key>' \
        --env KAFKACLOUDPASSWORD='<Enter API secret>' \
        --env SASLMECHANISM=PLAIN \
        --env VIPERVIZPORT=49689 \
        --env MQTTUSERNAME='' \
        --env MQTTPASSWORD='' \
        --env AIRFLOWPORT=9000 \
        --env READTHEDOCS='<Enter Readthedocs token>' \
        maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64
```

Tip

You can use DEMO credentials for testing and quickly seeing the power of TSS and TML.

The demo credentials can be found here: Demo Credentials

Your Solution Docker Run Command: Parameter Explanation

Important

Please ask the developer of this solution.

Parameter	**Explanation**
docker	This calls the docker engine
-d	This runs your container in **detached** mode
--net=host	This give your container access to your host operating system
--env TSS=0	Internal TSS variable. MUST be 0.
--env SOLUTIONNAME	This is the name of your TML solution.
--env SOLUTIONDAG	This is the name of the DAG that comprises your solution. This DAG is triggered automatically when you run this container.
--env SOLUTIONVIPERVIZPORT=TBD	This is the port Viperviz is listening. You point your browser to this port. See :ref:`Your Solution Dashboard URL`

--env CLIENTPORT=Not Applicable	Use this port if you are externally connecting to the TML/TSS solution using REST API or gRPC clients. You will need this port in the REST, and gRPC clients. This external port is used by Viper binary: Viper will be listening on this port for a connection as shown here: :ref:`Your Solution TML Binaries` In the TMUX window **Viper-produce**: :ref:`Your Solution TMUX Windows`
--env VIPERVIZPORT=49689	This is the port Viperviz is listening in TSS. You point your browser to this port. See :ref:`Your Solution Dashboard URL`
--env AIRFLOWPORT=9000	This is the port for Airflow in TSS solution studio container.
--env SOLUTIONAIRFLOWPORT=TBD	This is the port for Airflow in TML solution container. Note: This is provided mainly for debugging and testing purposes only.
--env GITUSERNAME	This is your Github username.
--env GITPASSWORD	This is the Github Personal Access Token you created. Refer to: Creating Github Token
--env GITREPOURL	This is your Github repo, that you cloned from **https://github.com/smaurice101/raspberrypi**
--env DOCKERUSERNAME	This is your Docker username.
--env READTHEDOCS	This is the readthedocs API token you created. Refer to: Set up readthedocs
--env CHIP=amd64	This is the chip family of your OS.
--env EXTERNALPORT=39399	This is the external port that you can use when making an external connection to your TML solution running in TSS Dev environment.
--env SOLUTIONEXTERNALPORT=TBD	This is the external port that you can use when making an external connection to your TML solution for external data ingestion. if SOLUTIONEXTERNALPORT=-1, TSS selects a free port randomly.
--env MQTTUSERNAME	This is your MQTT username
--env MQTTPASSWORD	This is your MQTT password.
--env KAFKACLOUDUSERNAME	This is your API key from Confluent Cloud
--env KAFKACLOUDPASSWORD	This is your API Secret from Confluent Cloud.
maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64	Your solution container name.

Your Solution Airflow Port

This is the airflow port in your solution container.

It can be accessed by entering: http://localhost:TBD

Important

TBD

Note: This port will change when SOLUTIONAIRFLOWPORT=-1, you can set it to

particular number.

Your Solution External Port

This is the Docker Run command for your solution container. Note ports may change at runtime. The solution documentation will update automatically.

Important

TBD

This is the external port that you can use when making an external connection to your TML solution for external data ingestion. You will need this port in the REST, and gRPC clients.

Note: if SOLUTIONEXTERNALPORT=-1, TSS will choose a free port randomly.

This external port is used by Viper binary: Viper will be listening on this port

for a connection as shown here :ref:`Your Solution TML Binaries`

In the TMUX window **Viper-produce**: :ref:`Your Solution TMUX Windows`

Non-Solution vs Solution Ports

Non-solution ports are only for TSS, this is because TSS includes a TML Dev environment to allow TML solution developers to test their solutions.

Solution ports are for your TML solution that you created and will deploy.

Important

It is important to note the difference between the following ports:

- AIRFLOWPORT and SOLUTIONAIRFLOWPORT
- EXTERNALPORT and SOLUTIONEXTERNALPORT
- VIPERVIZPORT and SOLUTIONVIPERVIZPORT

The reason is because TSS includes a Development environment for TML

solutions, many times you will want to run your solution in Dev and run

it in its own solution container for testing before you deploy your

solution. But, since ONLY ONE application can listen on a port,

we must assign a different port to the solutions so there is no

port conflict between applications in DEV and PROD.

However, if you set all port to -1, TSS will randomly choose
free ports for you. The reason for setting the ports with an
actual number that is NOT -1, is if you want to scale your TML solution
with Kubernetes and producing data using REST or gRPC and do not want
ports to keep changing and breaking your app.

Your Solution Dashboard URL

This is the visualization URL for your TML dashboard. Note ports may change at runtime. The solution documentation will update automatically.

> Important
>
> This will appear AFTER you run Your Solution Docker Container

Your Solution Log File Dashboard

This is the log file dashboard for your TML solution running.

> Important
>
> This will appear AFTER you run Your Solution Docker Container

Your Solution Dashboard URL: Parameter Explanation

Parameter	**Explanation**
http://localhost:TBD/<html file>	This is the URL pointing to an html file running inside your solution container. Refer to: TML Real-time dashboards
SOLUTIONVIPERVIZPORT=TBD	This is the port Viperviz is listening on.
topic	This is the topic that the TML binary Viperviz is reading (consuming) in Apache Kafka and sending it to your broweser over websockets.
offset	This value tells the Viperviz binary to read the latest real-time data. **offset=-1**, means to go to the end of the data stream and get the latest record.
groupid	This can be empty.

rollbackoffset	This is the number of offsets to **rollback** the data stream from the **offset** value. Note: If you increase this number, Viperviz will send more data to your browser. But be carefull, too much data may crash your browser or computer.
topictype	Leave as is.
append	This tells your html file whether to append or not the data streaming to your browser. If append=0, the html will not apend, if append=1, then data will accumulate in your browser.
secure	This tells Viperviz whether to encrypt your data to the browser. If secure=1, data are encrypted, secure=0 no encryption.

[tml-multi-agenticai-iot-3f10-ml_agenticai] Github Repo

This is the Github repo for all your solution code

> Important
>
> https://github.com/smaurice101/raspberrypitss/tree/main/tml-airflow/dags/tml-solutions/tml-multi-agenticai-iot-3f10

Readthedocs URL

This is this URL.

> Important
>
> https://tml-multi-agenticai-iot-3f10-ml_agenticai.readthedocs.io

Solution Trigger DAG

This is the name of the solution DAG you chose to trigger.

> Important
>
> solution_preprocessing_ml_agenticai_dag-tml-multi-agenticai-iot-3f10

Your Solution TML Binaries

These are the ports the TML binaries are listening on.

Important

VIPERHOST_PRODUCE=0.0.0.0, VIPERPORT_PRODUCE=5050, VIPERHOST_PREPOCESS=127.0.1.1, VIPERPORT_PREPROCESS=39719, VIPERHOST_PREPOCESS2=127.0.1.1, VIPERPORT_PREPROCESS2=39577, VIPERHOST_PREPOCESS_PGPT=127.0.1.1, VIPERPORT_PREPROCESS_PGPT=46321, VIPERHOST_ML=127.0.1.1, VIPERPORT_ML=33085, VIPERHOST_PREDCT=127.0.1.1, VIPERPORT_PREDICT=41879, HPDEHOST=127.0.1.1, HPDEPORT=45199, HPDEHOST_PREDICT=127.0.1.1, HPDEPORT_PREDICT=39591

Your Solution TMUX Windows

Important

python-produce-8296-tml-multi-agenticai-iot-3f10-ml_agenticai,solution_preprocessing_ml_agenticai_dag-tml-multi-agenticai-iot-3f10, python-preprocess-2805-tml-multi-agenticai-iot-3f10-ml_agenticai,solution_preprocessing_ml_agenticai_dag-tml-multi-agenticai-iot-3f10, python-ml-8666-tml-multi-agenticai-iot-3f10-ml_agenticai,solution_preprocessing_ml_agenticai_dag-tml-multi-agenticai-iot-3f10, python-predict-9344-tml-multi-agenticai-iot-3f10-ml_agenticai,solution_preprocessing_ml_agenticai_dag-tml-multi-agenticai-iot-3f10, python-agenticai-7610-tml-multi-agenticai-iot-3f10-ml_agenticai,solution_preprocessing_ml_agenticai_dag-tml-multi-agenticai-iot-3f10, viper-produce, viper-preprocess, viper-preprocess-pgpt, viper-ml, viper-predict

- Your solution is running in these

  TMUX windows:

  - To view windows, type:

    tmux ls
  - To go inside window, type:

    tmux a -t <window name>
  - To exit window, type:

    CTLR+b, d
  - To scroll window, type:

    CTLR+b, [
  - To un-scroll window, type:

    CTLR+[

Transactional Machine Learning Solution Studio (TSS) Usage

TSS Container Run Procedure

Important

All TSS details can be found on the main site: https://tml.readthedocs.io

For TSS Container Details Go Directly here.

Scaling [tml-multi-agenticai-iot-3f10-ml_agenticai] With Kubernetes

Generated On: 2025-09-28 23:50:27 UTC

You can scale your solution with Kubernetes. To do so, will will need to apply the following YAML files to your Kubernetes cluster.

Tip

Refer to TML documentation for more information on scaling with Kubernetes.

Watch the YouTube Video: here.

You can also run the YAML files locally in a 1 node kubernetes cluster called minikube. Refer to Installing minikube

Important

Below assumes you have a Kubernetes cluster and **kubectl** installed in your Linux environment.

Attention!

Make sure to STOP the TSS Container and other containers before running Kubernetes/Minikube.

If you get the following WARNING from Kubernetes:

> Warning FailedScheduling default-scheduler 0/1 nodes are available: 1 Insufficient nvidia.com/gpu. preemption: 0/1 nodes are available: 1 No preemption victims found for incoming pod. **Make sure no other application is using the GPU.** You can check by executing in terminal the command: **nvidia-smi**
>
> Oherwise, Issue the commands below:

```
sudo apt update && sudo apt install -y nvidia-docker2

sudo nvidia-ctk runtime configure --runtime=docker

sudo systemctl restart docker
```

Based on your TML solution [tml-multi-agenticai-iot-3f10-ml_agenticai] - if you want to scale your application with Kubernetes - you will need to apply the following YAML files.

YML File	Description
:ref:`tml-multi-agenticai-iot-3f10-ml_agenticai.yml`	This is your main solution YAML file. It MUST be applied to your Kubernetes cluster.

:ref:`secrets.yml`	You MUST store your passwords in base64 format in this file. For instructions on how to convert plain text passwords to base64 refer to instructions
:ref:`mysql-storage.yml`	This is storage allocation for MySQL DB. It MUST be applied to your Kubernetes cluster.
:ref:`mysql-db-deployment.yml`	This is the MySQL deployment YAML. It MUST be applied to your Kubernetes cluster.
:ref:`kafka.yml`	This is the Kafka deployment YAML. This is MANDATORY if using kafka locally or on-premise.
:ref:`privategpt.yml`	This is the privateGPT deployment YAML. This is OPTIONAL. However, it must be applied if using Step 9 DAG.
:ref:`qdrant.yml`	This is the Qdrant deployment YAML. This is OPTIONAL. However, it must be applied if using Step 9 DAG.
:ref:`nginx-ingress-tml-multi-agenticai-iot-3f10-ml_agenticai.yml`	If you are scaling your TML solution you must apply the nginx-ingresstml-multi-agenticai-iot-3f10-ml_agenticai.yml; this yaml is auto-generated for every TML solution. For more details see section :ref:`Scaling with NGINX Ingress and Ingress Controller`

kubectl Apply command

Important

To apply the YAML files below to your Kubernetes cluster simply run this command:

```
kubectl apply -f kafka.yml -f secrets.yml -f mysql-storage.yml -f mysql-db-deployment.yml -f tml-multi-agenticai-iot-3f10-ml_agenticai.yml
```

tml-multi-agenticai-iot-3f10-ml_agenticai.yml

Important

Copy and Paste this YAML file: tml-multi-agenticai-iot-3f10-ml_agenticai.yml - and save it locally.

Attention!

MAKE SURE to update any tokens and passwords in the **secrets.yml** file:

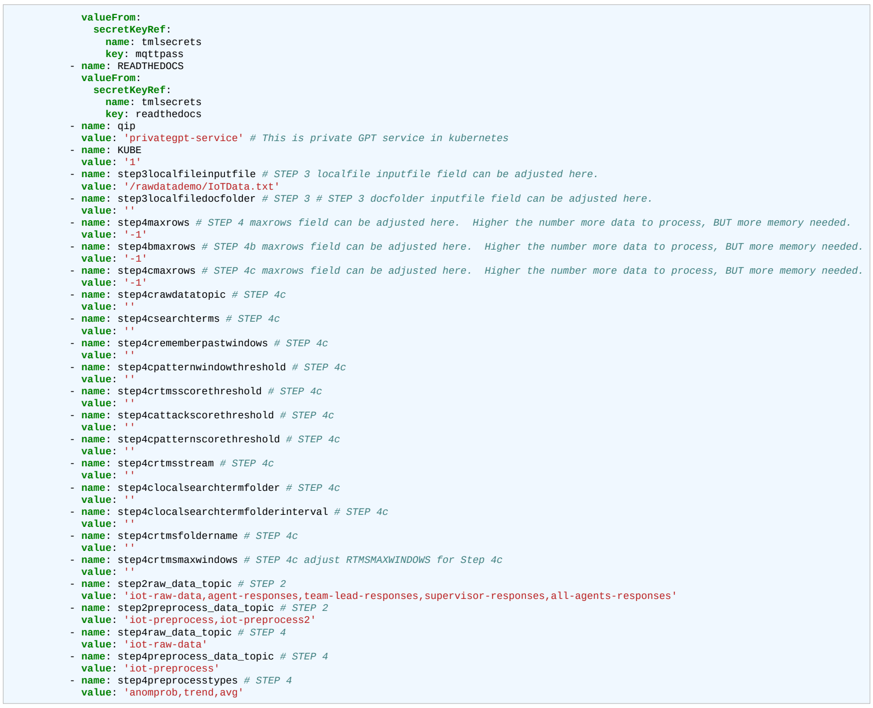
1. GITPASSWORD (MANDATORY)
2. READTHEDOCS (MANDATORY)
3. KAFKACLOUDPASSWORD (OPTIONAL)
4. MQTTPASSWORD (OPTIONAL)

For instructions on how to do this, refer to instructions

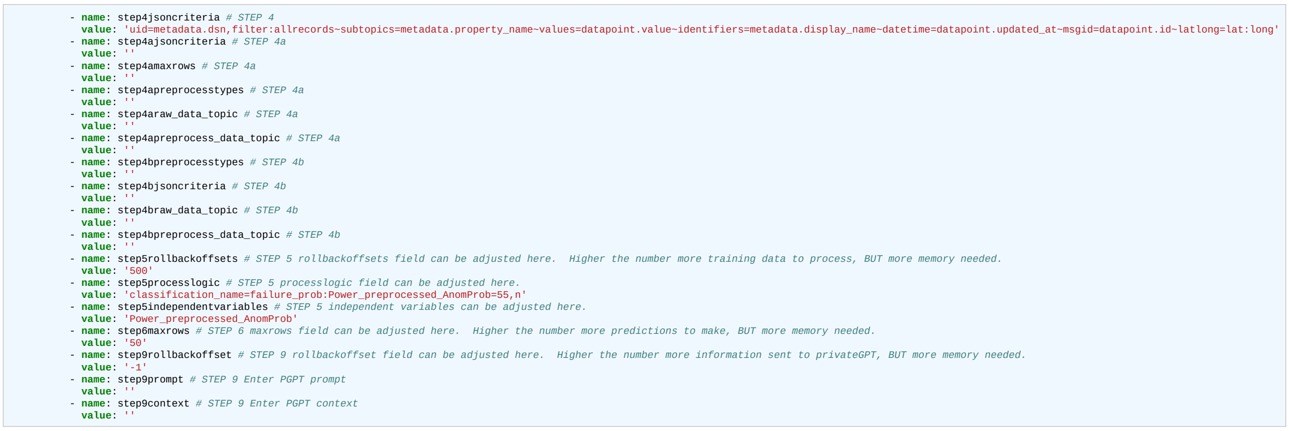
```
################# tml-multi-agenticai-iot-3f10-ml_agenticai.yml

  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: tml-multi-agenticai-iot-3f10-ml_agenticai
  spec:
    selector:
      matchLabels:
        app: tml-multi-agenticai-iot-3f10-ml_agenticai
    replicas: 3 # tells deployment to run 1 pods matching the template
    template:
      metadata:
        labels:
          app: tml-multi-agenticai-iot-3f10-ml_agenticai
      spec:
        containers:
        - name: tml-multi-agenticai-iot-3f10-ml_agenticai
          image: maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64:latest
          volumeMounts:
          - name: dockerpath
            mountPath: /var/run/docker.sock
          - name: rawdata
            mountPath: /rawdata   # container folder where the local folder will be mounted
          ports:
          - containerPort: 5050
          - containerPort: 4040
          - containerPort: 6060
          env:
          - name: TSS
            value: '0'
          - name: PROJECTNAME
            value: 'tml-multi-agenticai-iot-3f10'
          - name: SOLUTIONNAME
            value: 'tml-multi-agenticai-iot-3f10-ml_agenticai'
          - name: SOLUTIONDAG
            value: 'solution_preprocessing_ml_agenticai_dag-tml-multi-agenticai-iot-3f10'
          - name: GITUSERNAME
            value: 'smaurice101'
          - name: GITREPOURL
            value: 'https://github.com/smaurice101/raspberrypitss.git'
          - name: SOLUTIONEXTERNALPORT
            value: '5050'
          - name: CHIP
            value: 'amd64'
          - name: SOLUTIONAIRFLOWPORT
```

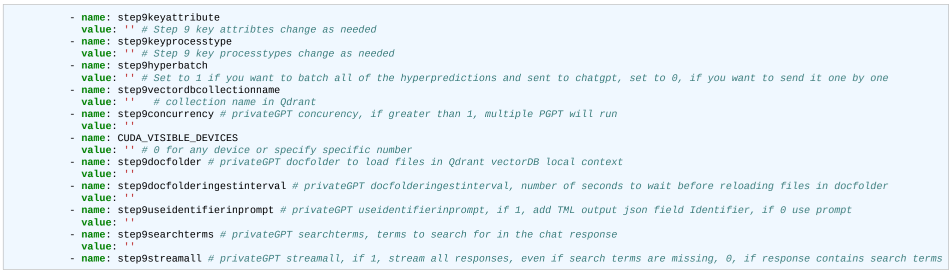
```
            value: '4040'
          - name: SOLUTIONVIPERVIZPORT
            value: '6060'
          - name: DOCKERUSERNAME
            value: 'maadsdocker'
          - name: CLIENTPORT
            value: '0'
          - name: EXTERNALPORT
            value: '39399'
          - name: KAFKACLOUDUSERNAME
            value: ''
          - name: VIPERVIZPORT
            value: '49689'
          - name: MQTTUSERNAME
            value: 'smaurice'
          - name: AIRFLOWPORT
            value: '9000'
          - name: GITPASSWORD
            valueFrom:
              secretKeyRef:
               name: tmlsecrets
               key: githubtoken
          - name: KAFKACLOUDPASSWORD
            valueFrom:
              secretKeyRef:
               name: tmlsecrets
               key: kafkacloudpassword
          - name: MQTTPASSWORD
```



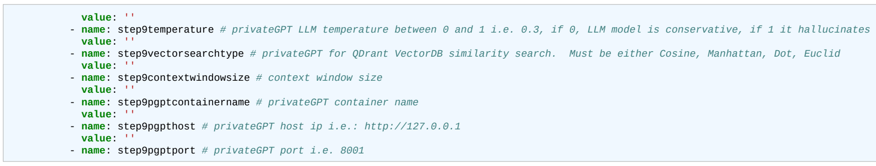
```yaml
          valueFrom:
            secretKeyRef:
              name: tmlsecrets
              key: mqttpass
        - name: READTHEDOCS
          valueFrom:
            secretKeyRef:
              name: tmlsecrets
              key: readthedocs
        - name: qip
          value: 'privategpt-service' # This is private GPT service in kubernetes
        - name: KUBE
          value: '1'
        - name: step3localfileinputfile # STEP 3 localfile inputfile field can be adjusted here.
          value: '/rawdatademo/IoTData.txt'
        - name: step3localfiledocfolder # STEP 3 # STEP 3 docfolder inputfile field can be adjusted here.
          value: ''
        - name: step4maxrows # STEP 4 maxrows field can be adjusted here.  Higher the number more data to process, BUT more memory needed.
          value: '-1'
        - name: step4bmaxrows # STEP 4b maxrows field can be adjusted here.  Higher the number more data to process, BUT more memory needed.
          value: '-1'
        - name: step4cmaxrows # STEP 4c maxrows field can be adjusted here.  Higher the number more data to process, BUT more memory needed.
          value: '-1'
        - name: step4crawdatatopic # STEP 4c
          value: ''
        - name: step4csearchterms # STEP 4c
          value: ''
        - name: step4crememberpastwindows # STEP 4c
          value: ''
        - name: step4cpatternwindowthreshold # STEP 4c
          value: ''
        - name: step4crtmsscorethreshold # STEP 4c
          value: ''
        - name: step4cattackscorethreshold # STEP 4c
          value: ''
        - name: step4cpatternscorethreshold # STEP 4c
          value: ''
        - name: step4crtmsstream # STEP 4c
          value: ''
        - name: step4clocalsearchtermfolder # STEP 4c
          value: ''
        - name: step4clocalsearchtermfolderinterval # STEP 4c
          value: ''
        - name: step4crtmsfoldername # STEP 4c
          value: ''
        - name: step4crtmsmaxwindows # STEP 4c adjust RTMSMAXWINDOWS for Step 4c
          value: ''
        - name: step2raw_data_topic # STEP 2
          value: 'iot-raw-data,agent-responses,team-lead-responses,supervisor-responses,all-agents-responses'
        - name: step2preprocess_data_topic # STEP 2
          value: 'iot-preprocess,iot-preprocess2'
        - name: step4raw_data_topic # STEP 4
          value: 'iot-raw-data'
        - name: step4preprocess_data_topic # STEP 4
          value: 'iot-preprocess'
        - name: step4preprocesstypes # STEP 4
          value: 'anomprob,trend,avg'
```



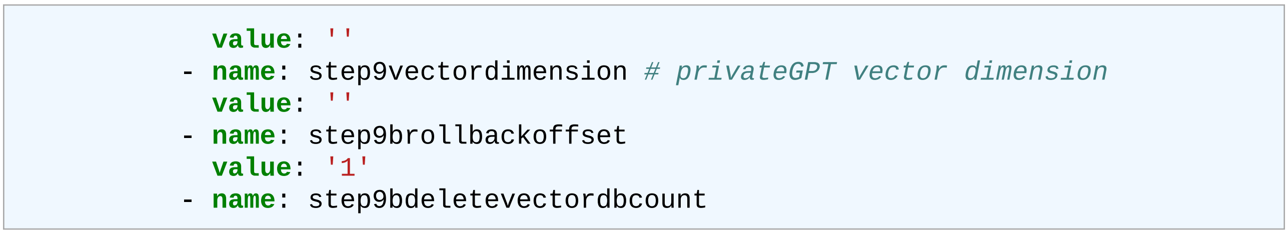
```yaml
        - name: step4jsoncriteria # STEP 4
          value: 'uid=metadata.dsn,filter:allrecords~subtopics=metadata.property_name~values=datapoint.value~identifiers=metadata.display_name~datetime=datapoint.updated_at~msgid=datapoint.id~latlong=lat:long'
        - name: step4ajsoncriteria # STEP 4a
          value: ''
        - name: step4amaxrows # STEP 4a
          value: ''
        - name: step4apreprocesstypes # STEP 4a
          value: ''
        - name: step4araw_data_topic # STEP 4a
          value: ''
        - name: step4apreprocess_data_topic # STEP 4a
          value: ''
        - name: step4bpreprocesstypes # STEP 4b
          value: ''
        - name: step4bjsoncriteria # STEP 4b
          value: ''
        - name: step4braw_data_topic # STEP 4b
          value: ''
        - name: step4bpreprocess_data_topic # STEP 4b
          value: ''
        - name: step5rollbackoffsets # STEP 5 rollbackoffsets field can be adjusted here.  Higher the number more training data to process, BUT more memory needed.
          value: '500'
        - name: step5processlogic # STEP 5 processlogic field can be adjusted here.
          value: 'classification_name=failure_prob:Power_preprocessed_AnomProb=55,n'
        - name: step5independentvariables # STEP 5 independent variables can be adjusted here.
          value: 'Power_preprocessed_AnomProb'
        - name: step6maxrows # STEP 6 maxrows field can be adjusted here.  Higher the number more predictions to make, BUT more memory needed.
          value: '50'
        - name: step9rollbackoffset # STEP 9 rollbackoffset field can be adjusted here.  Higher the number more information sent to privateGPT, BUT more memory needed.
          value: '-1'
        - name: step9prompt # STEP 9 Enter PGPT prompt
          value: ''
        - name: step9context # STEP 9 Enter PGPT context
          value: ''
```



```yaml
        - name: step9keyattribute
          value: '' # Step 9 key attribtes change as needed
        - name: step9keyprocesstype
          value: '' # Step 9 key processtypes change as needed
        - name: step9hyperbatch
          value: '' # Set to 1 if you want to batch all of the hyperpredictions and sent to chatgpt, set to 0, if you want to send it one by one
        - name: step9vectordbcollectionname
          value: ''   # collection name in Qdrant
        - name: step9concurrency # privateGPT concurency, if greater than 1, multiple PGPT will run
          value: ''
        - name: CUDA_VISIBLE_DEVICES
          value: '' # 0 for any device or specify specific number
        - name: step9docfolder # privateGPT docfolder to load files in Qdrant vectorDB local context
          value: ''
        - name: step9docfolderingestinterval # privateGPT docfolderingestinterval, number of seconds to wait before reloading files in docfolder
          value: ''
        - name: step9useidentifierinprompt # privateGPT useidentifierinprompt, if 1, add TML output json field Identifier, if 0 use prompt
          value: ''
        - name: step9searchterms # privateGPT searchterms, terms to search for in the chat response
          value: ''
        - name: step9streamall # privateGPT streamall, if 1, stream all responses, even if search terms are missing, 0, if response contains search terms
```



```yaml
          value: ''
        - name: step9temperature # privateGPT LLM temperature between 0 and 1 i.e. 0.3, if 0, LLM model is conservative, if 1 it hallucinates
          value: ''
        - name: step9vectorsearchtype # privateGPT for QDrant VectorDB similarity search.  Must be either Cosine, Manhattan, Dot, Euclid
          value: ''
        - name: step9contextwindowsize # context window size
          value: ''
        - name: step9pgptcontainername # privateGPT container name
          value: ''
        - name: step9pgpthost # privateGPT host ip i.e.: http://127.0.0.1
          value: ''
        - name: step9pgptport # privateGPT port i.e. 8001
```



```yaml
          value: ''
        - name: step9vectordimension # privateGPT vector dimension
          value: ''
        - name: step9brollbackoffset
          value: '1'
        - name: step9bdeletevectordbcount
```

```
            value: '1'
          - name: step9bvectordbpath
            value: ''
          - name: step9btemperature
            value: '/rawdata/vectordb'
          - name: step9bvectordbcollectionname
            value: ''
          - name: step9bollamacontainername
            value: ''
          - name: step9bCUDA_VISIBLE_DEVICES
            value: 'maadsdocker/tml-privategpt-with-gpu-nvidia-amd64-llama3-tools'
          - name: step9bmainip
            value: ''
          - name: step9bmainport
            value: 'http://127.0.0.1'
          - name: step9bembedding
            value: '11434'
          - name: step9bagents_topic_prompt
            value: 'nomic-embed-text'
          - name: step9bteamlead_topic
            value: '
      iot-preprocess:Are there any issues or anomalies in the JSON data values in the hyperprediction field? Specifically, the hyprepredicton field indicates the value of the Preprocesstype field,
      if the Preprocesstype is Avg, then the hyprepredictions are the average values of the first value in the Identifier field, which is Current for the
      device name in the mainuid field.
      For example, if the hyprepredicton=154646, and if Preprocesstype=Avg, and the first value in Identifier=Current, and mainuid=AC000W020496398
      then 154646 is the average of electrical current for device name AC000W020496398.  Determine if the trends electrical current values are normal or abnormal by looking at other similar jsons.
      Do NOT focus on data quality, just focus on the hyperprediction, Identifier, and mainuid fields.
```

Tip

In the solution YAML file above, you can adjust the **replicas** field. Currently, **replicas: 3** for demonstration purposes.

secrets.yml

Important

You MUST store base64 passwords in this file and apply it to the Kubernetes cluster.

Refer to instructions.

```
###################secrets.yml
apiVersion: v1
kind: Secret
metadata:
  name: tmlsecrets
type: Opaque
data:
  readthedocs: <enter your base64 password>
  githubtoken: <enter your base64 password>
  mqttpass: <enter your base64 password>
  kafkacloudpassword: <enter your base64 password>
```

mysql-storage.yml

Important

Copy and Paste this YAML file: mysql-storage.yml - and save it locally.

```yaml
################# mysql-storage.yml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 20Gi
```

mysql-db-deployment.yml

> Important
>
> Copy and Paste this YAML file: mysql-db-deployment.yml - and save it locally.

```yaml
################# mysql-db-deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
```

```
      - image: maadsdocker/mysql:latest
        name: mysql
        resources:
         limits:
          memory: "512Mi"
          cpu: "1500m"
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: "raspberry"
        - name: MYSQLDB
          value: "tmlids"
        - name: MYSQLDRIVERNAME
          value: "mysql"
        - name: MYSQLHOSTNAME
          value: "mysql:3306"
        - name: MYSQLMAXCONN
          value: "4"
        - name: MYSQLMAXIDLE
          value: "10"
        - name: MYSQLPASS
          value: "raspberry"
        - name: MYSQLUSER
          value: "root"
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim

---
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
spec:
  ports:
  - port: 3306
  selector:
    app: mysql
```

kafka.yml

This is the Kafka service needed by TML pods - if using Kafka locally or on-premise.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kafka
spec:
  selector:
    matchLabels:
      app: kafka
  replicas: 1 # tells deployment to run 1 pods matching the template
  template:
    metadata:
      labels:
        app: kafka
    spec:
      containers:
      - name: kafka
        image: maadsdocker/kafka-amd64  # IF you DO NOT have NVIDIA GPU use: maadsdocker/tml-privategpt-no-gpu-amd64
        env:
        - name: KAFKA_HEAP_OPTS
          value: "-Xmx512M -Xms512M"
        - name: PORT
          value: "9092"
        - name: TSS
          value: "0"
        - name: KUBE
          value: "1"
        - name: KUBEBROKERHOST
          value: "kafka-service:9092"
---
apiVersion: v1
kind: Service
metadata:
  name: kafka-service
spec:
  ports:
  - port: 9092
  selector:
    app: kafka
```

privategpt.yml

Note

This YAML is Optional - Use Only If Step 9 Dag is used

Important

Copy and Paste this YAML file: privategpt.yml - and save it locally.

Note

By default this assumes you have a Nvidia GPU in your machine and so it using the Nvidia privateGPT container:

> **image: maadsdocker/tml-privategpt-with-gpu-nvidia-amd64**

if you DO NOT have a Nvidia GPU installed then change image to:

> **image: maadsdocker/tml-privategpt-no-gpu-amd64**

```
################# privategpt.yml
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  name: privategpt
spec:
  selector:
    matchLabels:
      app: privategpt
  replicas: 1 # tells deployment to run 1 pods matching the template
  template:
    metadata:
      labels:
        app: privategpt
    spec:
      containers:
      - name: privategpt
        image: --kubeprivategpt-- # IF you DO NOT have NVIDIA GPU use: maadsdocker/tml-privategpt-no-gpu-amd64
        imagePullPolicy: IfNotPresent  # You can also use Always, Never
        env:
        - name: NVIDIA_VISIBLE_DEVICES
          value: all
        - name: DP_DISABLE_HEALTHCHECKS
          value: xids
        - name: WEB_CONCURRENCY
          value: "--kubeconcur--"
        - name: GPU
          value: "1"
        - name: COLLECTION
          value: "--kubecollection--"
        - name: PORT
          value: "8001"
        - name: CUDA_VISIBLE_DEVICES
          value: "0"
        - name: TOKENIZERS_PARALLELISM
          value: "false"
        - name: temperature
          value: "--kubetemperature--"
        - name: vectorsearchtype
          value: "--kubevectorsearchtype--"
        - name: contextwindowsize
          value: "--kubecontextwindowsize--"
        - name: vectordimension
          value: "--kubevectordimension--"
        - name: mainmodel
          value: "--kubemainmodel--"
        - name: mainembedding
          value: "--kubemainembedding--"
        - name: TSS
          value: "0"
        - name: KUBE
          value: "1"
        resources:             # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU
          limits:              # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU
            nvidia.com/gpu: 1  # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU
        ports:
        - containerPort: 8001
      tolerations:             # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU
      - key: nvidia.com/gpu    # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU
        operator: Exists       # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU
        effect: NoSchedule     # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU
---
apiVersion: v1
kind: Service
metadata:
  name: privategpt-service
  labels:
    app: privategpt-service
spec:
  type: NodePort #Exposes the service as a node ports
  ports:
  - port: 8001
    name: p1
    protocol: TCP
    targetPort: 8001
  selector:
    app: privategpt
```

ollama.yml

Note

This YAML is Optional - Use Only If Step 9b Dag is used for Multi-Agentic AI

Important

Copy and Paste this YAML file: ollama.yml - and save it locally.

Note

By default this assumes you have a Nvidia GPU in your machine and so it using the Nvidia Ollama container:

> **image: maadsdocker/tml-privategpt-with-gpu-nvidia-amd64-llama3-tools**

if you DO NOT have a Nvidia GPU installed then the Ollama container will use CPU but will be much slower.

```yaml
################# ollama.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ollama
spec:
  selector:
    matchLabels:
      app: ollama
  replicas: 1 # tells deployment to run 1 pods matching the template
  template:
    metadata:
      labels:
        app: ollama
    spec:
      containers:
      - name: ollama
        image: maadsdocker/tml-privategpt-with-gpu-nvidia-amd64-llama3-tools # IF you DO NOT have NVIDIA GPU then CPU will be used
        imagePullPolicy: IfNotPresent  # You can also use Always, Never
        env:
        - name: NVIDIA_VISIBLE_DEVICES
          value: all
        - name: DP_DISABLE_HEALTHCHECKS
          value: xids
        - name: WEB_CONCURRENCY
          value: "2"
        - name: GPU
          value: "1"
        - name: COLLECTION
          value: "tml-llm-model-v2"
```

```yaml
        - name: PORT
          value: "11434"
        - name: CUDA_VISIBLE_DEVICES
          value: "0"
        - name: TOKENIZERS_PARALLELISM
          value: "false"
        - name: temperature
          value: "0.1"
        - name: rollbackoffset
          value: "1"
        - name: ollama-model
          value: "llama3.1"
        - name: deletevectordbcount
          value: "10"
        - name: vectordbpath
          value: "/rawdata/vectordb"
        - name: topicid
```

```
          value: "-999"
        - name: enabletls
          value: "1"
        - name: partition
          value: "-1"
        - name: vectordbcollectionname
          value: "tml-llm-model-v2"
        - name: ollamacontainername
          value: "maadsdocker/tml-privategpt-with-gpu-nvidia-amd64-llama3-tools"
        - name: mainip
          value: "http://127.0.0.1"
        - name: mainport
          value: "11434"
        - name: embedding
          value: "nomic-embed-text"
        - name: agents_topic_prompt
          value: "
  iot-preprocess:Are there any issues or anomalies in the JSON data values in the hyperprediction field? Specifically, the hyprepredicition field indicates the value of the Preprocesstype field,
  if the Preprocesstype is Avg, then the hyprepredictions are the average values of the first value in the Identifier field, which is Current for the
  device name in the mainuid field.
  For example, if the hyprepredicition=154646, and if Preprocesstype=Avg, and the first value in Identifier=Current, and mainuid=AC000W020496398
  then 154646 is the average of electrical current for device name AC000W020496398.  Determine if the trends electrical current values are normal or abnormal by looking at other similar jsons.
  Do NOT focus on data quality, just focus on the hyperprediction, Identifier, and mainuid fields."
        - name: teamlead_topic
          value: "team-lead-responses"
        - name: teamleadprompt
          value: "
Are there any issues or major concerns in the data? The data are from IoT devices that are being monitored by individual agents. If you find issues or concerns, then
highlight the devices name (i.e. in the mainuid field) with details on whether the device failure probabilities are increasing or greater than 0.70.
```

"

- name: supervisor_topic value: "supervisor-responses"
- name: supervisorprompt value: "

Are there any major issues or concerns in the data? This data is IoT data being monitored for potential failure from IoT devices. If you find a major concern or major issues in the failure probabilities of IoT devices indicated in the hyperprediction values, then use the send_email tool to send an email message that highlights devices with the issues that need investigation Do NOT send too many emails, and do not send duplicate emails with same device names."

- name: agenttoolfunctions value: "

send_email:send_email: You are an email-sending agent. Use smtp parameters to send emails when there is an anomaly in the data, make sure to

indicate the device name in the mainuid field. do not write a smtp script, actually send the email using the SMTP parameters smtp_server='' smtp_port=0 username='' password='' sender='' recipient='' subject='' body=''"

- name: agent_team_supervisor_topic value: "all-agents-responses"
- name: TSS value: "0"
- name: KUBE value: "1"

resources: # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU

limits: # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU

nvidia.com/gpu: 1 # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU

ports: - containerPort: 11434

tolerations: # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU - key: nvidia.com/gpu # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU

operator: Exists # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU effect: NoSchedule # REMOVE or COMMENT OUT: IF you DO NOT have NVIDIA GPU

--- apiVersion: v1 kind: Service metadata:

name: ollama-service labels:

app: ollama-service

spec:

type: NodePort #Exposes the service as a node ports ports: - port: 11434 name: p1 protocol: TCP targetPort: 11434

selector:

app: ollama

qdrant.yml

Note

This YAML is Optional - Use Only If Step 9 Dag is used

Important

Copy and Paste this YAML file: qdrant.yml - and save it locally.

```yaml
################# qdrant.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: qdrant
spec:
  selector:
    matchLabels:
      app: qdrant
  replicas: 1
  template:
    metadata:
      labels:
        app: qdrant
    spec:
      #hostNetwork: true
      containers:
      - name: qdrant
        image: qdrant/qdrant
        ports:
        - containerPort: 6333
        volumeMounts:
        - mountPath: /qdrant/storage
          name: qdata
      volumes:
      - name: qdata
        hostPath:
          path: /qdrant_storage
---
apiVersion: v1
kind: Service
metadata:
  name: qdrant-service
  labels:
    app: qdrant-service
spec:
  type: NodePort #Exposes the service as a node ports
  ports:
  - port: 6333
    name: p1
    protocol: TCP
    targetPort: 6333
  selector:
    app: qdrant
```

Tip

The number of replicas can be changed in the **cybersecuritywithprivategpt-3f10.yml** file: look for **replicas**. You can increase or decrease the number of replicas based on the amout of real-time data you are processing.

Kubernetes Dashboard Visualization

To visualize the dashboard you need to forward ports to your solution **deployment in Kubernetes**. For this solution, the port forward command would be:

```
kubectl port-forward deployment/tml-multi-agenticai-iot-3f10-ml_agenticai 6060:6060
```

After you forward the ports then copy/paste the viusalization URL below and run your dashboard.

```
http://localhost:6060/dashboard-agenticai.html?topic=all-agents-responses,iot-preprocess,iot-ml-prediction-results-output&offset=-1&groupid=&rollbackoffset=400&topictype=prediction&append=0&secure=1
```

Scaling with NGINX Ingress and Ingress Controller

All TML solutions will scale with NGINX ingress to perform load-balancing. But, before you can use ingress - ingress MUST be enabled in Kubernetes cluster. Follow these steps:

Important

STEP 1: To turn on ingress in minikube type:

```
minikube addons enable ingress
```

```
minikube addons enable ingress-dns
```

STEP 2: In Linux Add tml.tss domain name to /etc/hosts file

a. Edit your **/etc/hosts** file

b. add an entry to **/etc/hosts**:

```
127.0.0.1 tml.tss
```

c. Save the file

STEP 2b: In Windows Add tss.tml domain name to C:\Windows\System32\drivers\etc

a. Edit your **C:\Windows\System32\drivers\etc\hosts** file (Note: You may need to COPY the hosts file to another directory, then edit the file, then copy it back to **C:\Windows\System32\drivers\etc\hosts**

b. add an entry:

```
127.0.0.1 tml.tss
```

c. Save the file

d. copy it back to **C:\Windows\System32\drivers\etc\hosts**

STEP 3: In a new Linux terminal you MUST turn on minikube tunnel type:

```
minikube tunnel
```

STEP 4: Apply nginx-ingress-tml-multi-agenticai-iot-3f10-ml_agenticai.yml to your kubernetes cluster. First you need to save it locally then apply it:

nginx-ingress-tml-multi-agenticai-iot-3f10-ml_agenticai.yml

```
############# nginx-ingress-tml-multi-agenticai-iot-3f10-ml_agenticai.yml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tml-ingress
  annotations:
    nginx.ingress.kubernetes.io/use-regex: "true"
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  ingressClassName: nginx
  rules:
    - host: tml.tss
      http:
        paths:
          - path: /viz(/|$)(.*)
            pathType: ImplementationSpecific
            backend:
              service:
                name: tml-multi-agenticai-iot-3f10-ml_agenticai-visualization-service
                port:
                  number: 80
---
apiVersion: v1
kind: ConfigMap
apiVersion: v1
metadata:
  name: ingress-nginx-controller
  namespace: ingress-nginx
data:
  allow-snippet-annotations: "true"
```

```
kubectl apply -f nginx-ingress-tml-multi-agenticai-iot-3f10-ml_agenticai.yml
```

You are now ready to run the Dashboard using Ingress load balancing.

Ingress Dashboard Visualization

Copy and paste this URL below in your browser and start streaming. Because you are now using INGRESS, Kubernetes will perform load balancing on the streaming data.

```
http://tml.tss/viz/dashboard-agenticai.html?topic=all-agents-responses,iot-preprocess,iot-ml-prediction-results-output&offset=-1&groupid=&rollbackoffset=400&topictype=prediction&append=0&secure=1
```

Making Secure TLS Connection with gRPC

You can make secure connection through the NGINX controller to your TML service using TLS encryption. TML solutions utilizing gRPC have a built-in gRPC server for secure and fast connections.

> Note
>
> Note that the REST API service is unencrypted, but very useful if you don't have the need for security inside your VM.

Here are the Steps to follow.

Step 1: Get Server Certificates

To utilize the secure gRPC service you must have valid security certificates for each server. You can self-sign your own certificates by following the steps here: Steps to Self-Signing Certificates

> Tip
>
> You can download the all the certificates from the certs repo

Step 2: Apply the Server Certificates to Kubernetes Cluster

> Attention!
>
> These certificates are for the **tml.tss** sever. Follow the steps to add this host to your /etc/hosts file: :ref:Scaling with NGINX Ingress and Ingress Controller`
>
> If you have a different host, then you will need to re-generate these certificates for your new host, replace tls.crt and tls.key with your your new keys. **Note these keys are in base64 format for security.** To replace with your own hosts, just update the the san.cnf file

Step 3: Apply the secret-tls to Kubernetes Cluster

Save the Yaml in Step 2 locally, then apply it to the cluster.

```
kubectl apply -f secret-tls.yml
```

You are done! You know have a secure connection betweenn your client gRPC application and the TML solution.

Using gRPcurl to Write Data to the TML gRPC Server

gRPcurl is a utility for writing data to your gRPC solution.

Tip

You can install gRPCurl from here.

Important

You must download four (4) files to your local machines:

1. ca.crt
   a. Get it from here
2. tml_grpc.proto, tml_grpc_pb2_grpc.py, tml_grpc_pb2.py
   a. Get them from here

Run the gRPCurl Commands

Once your TML solution using gRPC is running in Kubernetes you can test it by sending data with these commands:

```
grpcurl -insecure -H "client-api-protocol: 1,1" -cacert ca.crt -import-path . -proto tml_grpc.proto  tml.tss:443 list tmlproto.Tmlproto
```

```
grpcurl -insecure -H "client-api-protocol: 1,1" -cacert ca.crt -import-path . -proto tml_grpc.proto  tml.tss:443 list
```

```
grpcurl -insecure -H "client-api-protocol: 1,1" -cacert ca.crt -import-path . -proto tml_grpc.proto  tml.tss:443 describe tmlproto.Tmlproto.GetServerResponse
```

```
grpcurl -insecure -H "client-api-protocol: 1,1" -cacert ca.crt -import-path . -proto tml_grpc.proto  tml.tss:443 describe .tmlproto.Message
```

```
grpcurl -insecure -H "client-api-protocol: 1,1" -cacert ca.crt -import-path . -proto tml_grpc.proto -msg-template tml.tss:443 describe .tmlproto.Message
```

Send data to the sever:

```
grpcurl -insecure -H "client-api-protocol: 1,1" -cacert ca.crt -import-path . -proto tml_grpc.proto -d '{"message":"admin yeah!!"}' tml.tss:443 tmlproto.Tmlproto/GetServerResponse
```

Kubernetes Pod Access Commands

To go inside the pods, you can type command:

```
kubectl exec -it <pod name> -- bash
```

Note: replace **<pod name>** with actual pod name..use this command to get the pod name

```
kubectl get pods -A
```

To list service pods type:

```
kubectl get svc -A
```

To list deployment pods type:

```
kubectl get deployments -A
```

To Horizontally AUTO-SCALE Deployments type:

```
kubectl autoscale deployment  <deployment name> --cpu-percent=50 --min=1 --max=100
```

> Important
>
> The above command instructs Kubernetes to scale pods based on 50% CPU utilization to a minimum number of pods of 1 (small workload) to a maximum of 100 pods for large world loads. Of course, you can easily change these min and max numbers.
>
> This auto-scaling is very important to scale up and down your solution, while efficiently managing cloud computing costs.

To list deployments being auto-scaled type:

```
kubectl get hpa -A
```

To delete the pods:

```
kubectl delete all --all --all-namespaces
```

To get information on a pod type:

```
kubectl describe pod <pod name>
```

Start minikube with NVIDIA GPU Access:

```
minikube start --driver docker --container-runtime docker --gpus all --cni calico --memory 8192
```

> Note
>
> Note you may need to type: **./minikube**

Start minikube with NO GPU:

```
minikube start --driver docker --container-runtime docker --cni calico --memory 8192
```

DELETE minikube:

```
minikube delete
```

Tip

Adjust the **--memory 8192** as needed.

Latest Logs From Latest Build

Generated On: 2025-09-28 23:50:27 UTC

These are the latest logs generated from your latest build.

Tip

Complete logs from all builds can be found here on GitHub

```
[INFO 2025-09-28_23:45:40] STEP 1: completed - TML system parameters successfully gathered

[INFO 2025-09-28_23:45:45] STEP 2: Create topics started

[INFO 2025-09-28_23:48:07] STEP 2: Completed

[INFO 2025-09-28_23:48:16] STEP 3: producing data started

[INFO 2025-09-28_23:48:20] STEP 4: Preprocessing started

[INFO 2025-09-28_23:48:22] STEP 3: reading local file..successfully

[INFO 2025-09-28_23:48:25] STEP 4: Preprocessing started

[INFO 2025-09-28_23:48:30] STEP 7: Visualization started

[INFO 2025-09-28_23:48:31] STEP 5: Machine learning started

[INFO 2025-09-28_23:48:46] STEP 6: Predictions started

[INFO 2025-09-28_23:48:48] STEP 7: /Viperviz/viperviz-linux-amd64 0.0.0.0 49689

[INFO 2025-09-28_23:48:55] STEP 9b: Starting ollama server

[INFO 2025-09-28_23:49:12] STEP 8: Starting docker push for: maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64

[INFO 2025-09-28_23:49:23] STEP 9b: Ollama container.  Here is the run command: docker run -d -p 11434:11434 --net=host --gpus all -v /var/run/docker.sock:/var/run/docker.sock:z --env PORT=11434 --env TSS=1 --env GPU=1 --env COLLECTION=tml-llm-model-v2 --env WEB_CONCURRENCY=2 --env CUDA_VISIBLE_DEVICES=0 --env TOKENIZERS_PARALLELISM=false --env temperature=0.1 --env LLAMAMODEL="llama3.1" --env mainembedding="nomic-embed-text" --env OLLAMASERVERPORT="http://127.0.0.1:11434" maadsdocker/tml-privategpt-with-gpu-nvidia-amd64-llama3-tools, v=0

[INFO 2025-09-28_23:49:23] STEP 9b: Success starting Agentic AI.  Here is the run command: docker run -d -p 11434:11434 --net=host --gpus all -v /var/run/docker.sock:/var/run/docker.sock:z --env PORT=11434 --env TSS=1 --env GPU=1 --env COLLECTION=tml-llm-model-v2 --env WEB_CONCURRENCY=2 --env CUDA_VISIBLE_DEVICES=0 --env TOKENIZERS_PARALLELISM=false --env temperature=0.1 --env LLAMAMODEL="llama3.1" --env mainembedding="nomic-embed-text" --env OLLAMASERVERPORT="http://127.0.0.1:11434" maadsdocker/tml-privategpt-with-gpu-nvidia-amd64-llama3-tools

[ERROR 2025-09-28_23:49:36] STEP 9b: Agentic AI Step 9b DAG in tml_system_step_9b_agenticai_dag-tml-multi-agenticai-iot-3f10.py model "llama3.1" not found, try pulling it first (status code: 404)  Aborting after 10 consecutive errors.

[INFO 2025-09-28_23:50:15] STEP 8: Docker Container created and optimized.  Will push it now.  Here is the commit command: docker commit 2676ff0e07dc maadsdocker/tml-multi-agenticai-iot-3f10-ml_agenticai-amd64 - message=0

[INFO 2025-09-28_23:50:27] STEP 10: Started to build the documentation

[INFO 2025-09-28_23:50:26] STEP 10: Documentation successfully built on GitHub..Readthedocs build in process and should complete in few seconds
```