Week 2/24-2/28 Notes

- Downloaded Kafka, mySQL, and Docker onto personal computer

My Task:

Create Python Function for Randomization of Kafka Topic Name for login/registry:

Conditions:

- Make sure name doesn’t already exist in database (can’t have two of the same name)

Note: we want to use random words (not numbers) for this name to make it easier to understand

Right now:

```
171     topic_name = None
172     # Special case: If entity is an "Agent", create a Kafka topic
173     if entity_type == "Agent":
174         topic_name = data["name"].replace(" ", "_").lower()
175         create_kafka_topic(topic_name)
176         send_kafka_message(topic_name, {"message": f"New Agent registered: {data['name']}"})
177
```

Potential Libraries:

random-word:

- Library used to generate random words (simply/easy)
  - Appends random names to Kafka Topic Name

faker:

- Library used to generate random words, names, addresses
  - Appends random names to Kafka Topic Name
  - Can generate wider variety of random data

Consider:

If name doesn’t already exist in database:

- Keep name as is (ex. “example_agent_1”)

OR

- **Append random words to all names so that all Kafka Topic Names are randomized**

For Next Week:

- Check status of my team
- Share my own progress
- Come up with next task
- Status review

Progress

- Registered myself in the system and generated my private key so that I can register an Agent (Test my code)

- Randomization of Kafka Topic Name
  - Stored in Database

Next steps:

- Add code to newest version

- Working on Next Task:
  - Tests need to be done by 3/20
  - Researching how we can host the HSML API
    - Needs to be able to access it and call it from BL Plugin
    - Idea: takes input, authenticates and provides access to Kafka topic (question: should it be in EC2 container/inside local server? How to securely host it?

Note:

- Verification Codes Doc in GitHub (functions & missing functions)

Running in Kafka/Docker Environment (as a package)

Pros:

- Everything is centralized
- Easier to manage
- Can integrate Kafka for logging in/failed logins

Cons:

- Issues with scaling (if there’s high traffic)
- Security might need additional attention (making sure Docker network is secure)

Running in EC2 Container

Pros:

- Easier to scale
- Integrates with AWS services
- Requires less management

Cons:

- Less direct control

*Likely the more secure option

Additional:

Want full control:

- Kubernetes (EKS) or EC2 with Docker

Kubernetes: auto-scaling, self-healing, better security, high availability

Less management:

- AWS Fargate (check next slide)

Additional:

- AWS ECS (Elastic Container Service) with Fargate (could be better choice)
  - Works well for Kafka-based API authentication system
  - High availability
  - Auto-scaling

“AWS Fargate is a serverless compute engine that allows users to run containers without managing servers. It works with Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS). “

EC2: best option

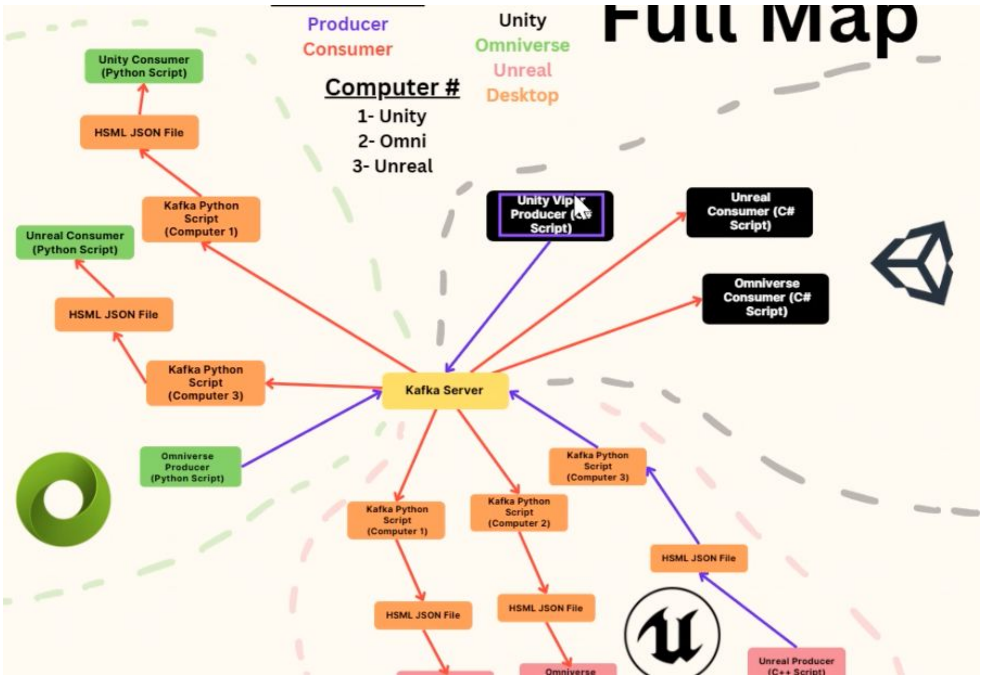
- Full control
- Comfortable managing Docker & security ourselves
- Direct, simple, Kafka integration
- Don’t need auto-scaling

Next Steps:

- Tomorrow: integrate and test my code in the lab

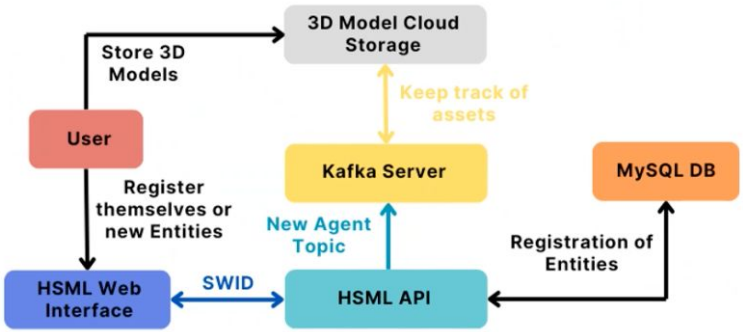
Next 2 Tasks I’m working on:

- Test time it takes to register an entity and authenticate a person
  - Implement timestamp from when they submit request
- Adding to this function: “Creator” property – mandatory in Domain/Agent) – needs “swid” inside for the Person (return error if not)

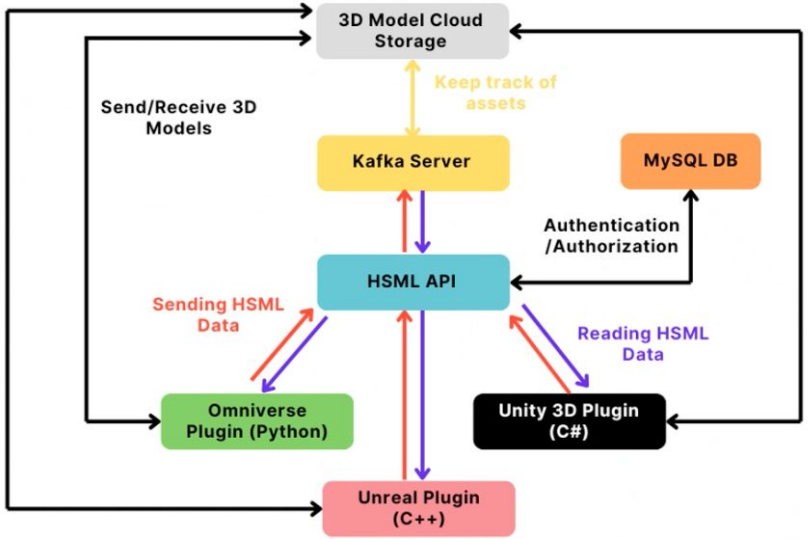

Full Map
Producer
Consumer
Unity
Omniverse
Unreal
Desktop
Computer #
1- Unity
2- Omni
3- Unreal
Unity Consumer (Python Script)
HSML JSON File
Kafka Python Script (Computer 1)
Unreal Consumer (Python Script)
HSML JSON File
Kafka Python Script (Computer 3)
Omniverse Producer (Python Script)
Unity Vip Producer (C# Script)
Unreal Consumer (C# Script)
Omniverse Consumer (C# Script)
Kafka Server
Kafka Python Script (Computer 1)
Kafka Python Script (Computer 2)
Kafka Python Script (Computer 3)
HSML JSON File
HSML JSON File
HSML JSON File
Unreal Producer (C++ Script)

Demo System Architecture

Registration



High Level Design



Modified from Jared's Figure

Progress Update 3/20

- 3/14 (in lab): added checking if Kafka Topic name exists in database
  - New method: uploading my code as a .py file
- 3/17: small group meeting
  - Alicia integrated and tested the grade and it is now part of version 7
- Currently working on:
  - Test time it takes to register an entity and authenticate a person
    - Implement timestamp from when they submit request
- Next:
  - Adding to this function: “Creator” property – mandatory in Domain/Agent) – needs “swid” inside for the Person (return error if not)

Progress Update 3/27

- Finished incorporating time stamp
  - Coming into lab tomorrow 3/27 to test the code and record the amount of time the program currently takes to register and authenticate an entity

- Adding to this function: “Creator” property – mandatory in Domain/Agent) – needs “swid” inside for the Person (return error if not)

Progress Update 3/28 (in lab)

Testing Latency for Registering myself:

```
Generated unique SWID: did:key:6MkmYpSZtZVmhau2ZonbV444vZCv139ebbzzsEpzEH8hHMV
It took 0.52 seconds to register.
Private key saved to: C:/Users/nnamian/OneDrive - JPL/Desktop/Digital Twin Int
eroperability/Codes/HSML Examples/registeredExamples\private_key.pem
Updated JSON saved to: C:/Users/nnamian/OneDrive - JPL/Desktop/Digital Twin In
teroperability/Codes/HSML Examples/registeredExamples\Niki_Namian.json
{'status': 'success', 'message': 'Entity registered successfully', 'did_key':
'did:key:6MkmYpSZtZVmhau2ZonbV444vZCv139ebbzzsEpzEH8hHMV', 'private_key_path':
 'C:/Users/nnamian/OneDrive - JPL/Desktop/Digital Twin Interoperability/Codes/
HSML Examples/registeredExamples\\private_key.pem', 'updated_json_path': 'C:/U
sers/nnamian/OneDrive - JPL/Desktop/Digital Twin Interoperability/Codes/HSML E
xamples/registeredExamples\\Niki_Namian.json'}
PS C:\Users\nnamian\Documents\GitHub\hsml_schema\scripts\verification>
```

It took **0.52 seconds** to register myself. This is from the time I input the JSON file with my information to the time it generated my private key.

Progress Update 3/28 (in lab)

Testing Latency for Authenticating myself:

```
PS C:\Users\nnamian\Documents\GitHub\hsml_schema\scripts\verification> python .\Re
gistration_API_v4.py
Must be registered in the Spatial Web to register a new Entity. Type 'new' to regi
ster or 'login' if already registered:ll
ogin
Provide your private_key.pem path: C:\Users\nnamian\OneDrive - JPL\Desktop\Digital
 Twin Interoperability\Codes\HSML Examples\registeredExamples\private_key_Niki.pem

It took 0.01 seconds to authenticate.
Welcome Niki Namian, you can now register your new Entity.
Enter the directory to your HSML JSON to be registered:
```

It took **0.01 seconds** to authenticate myself. This is from the time I input the path to my private key to the time that the system ensured that I was already registered (authenticated me).

Progress Update 3/28 (in lab)

Testing Latency for Registering an Entity:

```
Welcome Niki Namian, you can now register your new Entity.
Enter the directory to your HSML JSON to be registered: C:\Users\nnamian\Documents
\GitHub\hsml_schema\examples\registeredExamples\Example_Entity.json
HSML JSON accepted.
Warning: Object not linked to any other Entity. It will be registered under this u
ser's SWID.
Warning: SWID 'did:key:6MktDWufSfhLrYfcAWnstadnsUQQryRN25c55D8jbvbRPxZ' in JSON fi
le will be overwritten.
Subprocess output:
Generated DID:key: did:key:6MktnTgZWX7fDFJPzkYhUaWbzRTVrsNudzWjvkK7fv7q7PH
Private key saved to private_key.pem

Generated unique SWID: did:key:6MktnTgZWX7fDFJPzkYhUaWbzRTVrsNudzWjvkK7fv7q7PH
It took 0.46 seconds to register.
Private key saved to: C:/Users/nnamian/OneDrive - JPL/Desktop/Digital Twin Interop
erability/Codes/HSML Examples/registeredExamples\private_key.pem
Updated JSON saved to: C:/Users/nnamian/OneDrive - JPL/Desktop/Digital Twin Intero
perability/Codes/HSML Examples/registeredExamples\Example_Entity.json
{'status': 'success', 'message': 'Entity registered successfully', 'did_key': 'did
:key:6MktnTgZWX7fDFJPzkYhUaWbzRTVrsNudzWjvkK7fv7q7PH', 'private_key_path': 'C:/Use
rs/nnamian/OneDrive - JPL/Desktop/Digital Twin Interoperability/Codes/HSML Example
s/registeredExamples\\private_key.pem', 'updated_json_path': 'C:/Users/nnamian/One
Drive - JPL/Desktop/Digital Twin Interoperability/Codes/HSML Examples/registeredEx
amples\\Example_Entity.json'}
PS C:\Users\nnamian\Documents\GitHub\hsml_schema\scripts\verification>
```

Continuation of Last Slide

After authenticating myself, I registered an entity. From the time I entered the path to the JSON file to the time that it took to register the entity was **0.46 seconds**.

Note: the time it took to generate the did key for the entity is included in this time

Progress Update 3/28 (in lab)

Breaking down steps in registering an entity:

```
It took 0.01 seconds to authenticate.

Welcome Niki Namian, you can now register your new Entity.
Enter the directory to your HSML JSON to be registered: C:\Users\nnamian\Documents
\GitHub\hsml_schema\examples\registeredExamples\example_Agent.json

It took 0.00 seconds to accept the HSML JSON file.

HSML JSON accepted.

It took 0.00 seconds to generate warning for registration/check required fields.


It took 0.10 seconds to check if swid already exists.

Warning: SWID 'did:key:6MktkcpjWjsKwfngBbkpgT5qF3WgR9FAatXXE8N2CXne5E4' in JSON fi
le will be overwritten.
```

It took **0.10 seconds** to check if the swid already exists in the database.

Progress Update 3/28 (in lab)

Kafka Topic Name:

```
Subprocess output:
Generated DID:key: did:key:6MkhYhaCAKF8DGhxUcVE6DutMEtgZ7g7KMtyN7Qt51GiaF2
Private key saved to private_key.pem

Generated unique SWID: did:key:6MkhYhaCAKF8DGhxUcVE6DutMEtgZ7g7KMtyN7Qt51GiaF2
Kafka topic 'example_agent_3fv5am' created successfully.

It took 0.03 seconds to create the Kafka topic name.

Message sent to Kafka topic 'example_agent_3fv5am': {'message': 'New Agent registered: example Agent'}

It took 0.90 seconds to send the message.
```

It took **0.03 seconds** to create the Kafka topic name after while registering and **0.90 seconds** to send a message.

Table of Time Measurements

Process	**Time Taken (seconds)**
Registering a New User	0.52 seconds
Authenticate a User (trying to login)	0.01 seconds
Register a New Entity	0.46 seconds
Create a Kafka Topic Name	0.03 seconds
Send a Message	0.90 seconds

Broken Down Steps Within Registration

Process	**Time Taken (seconds)**
Accept HSML JSON File	0.00 seconds
Check of all required fields	0.00 seconds
Check if swid already exists	0.10 seconds