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)

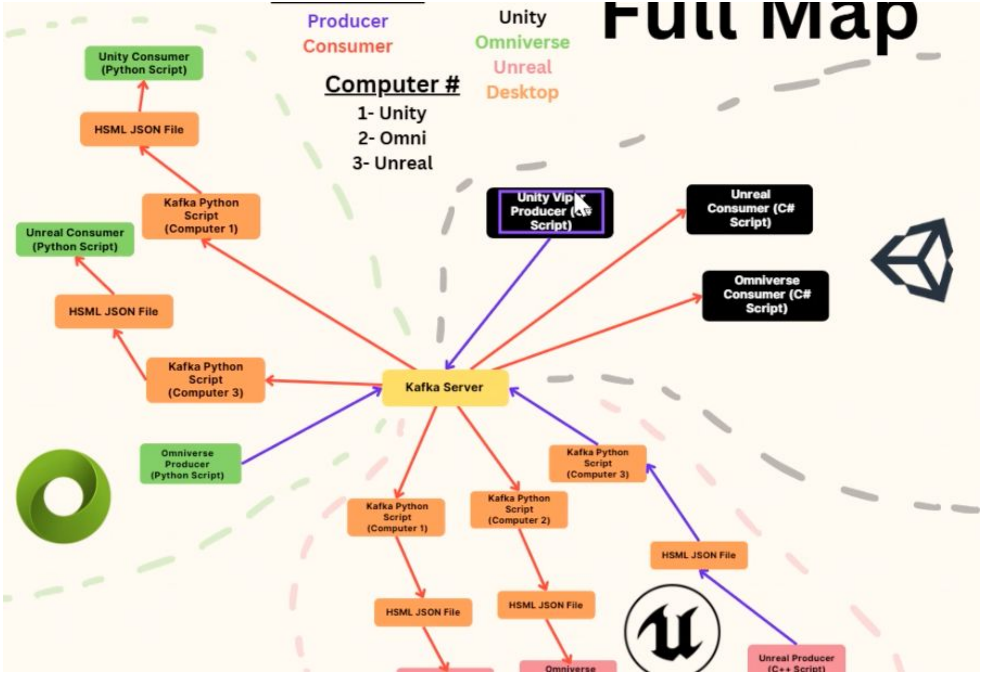
Hosting HSML API

- 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). “

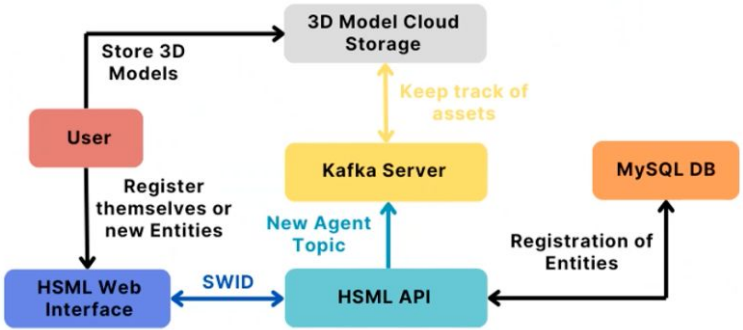
EC2: best option

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

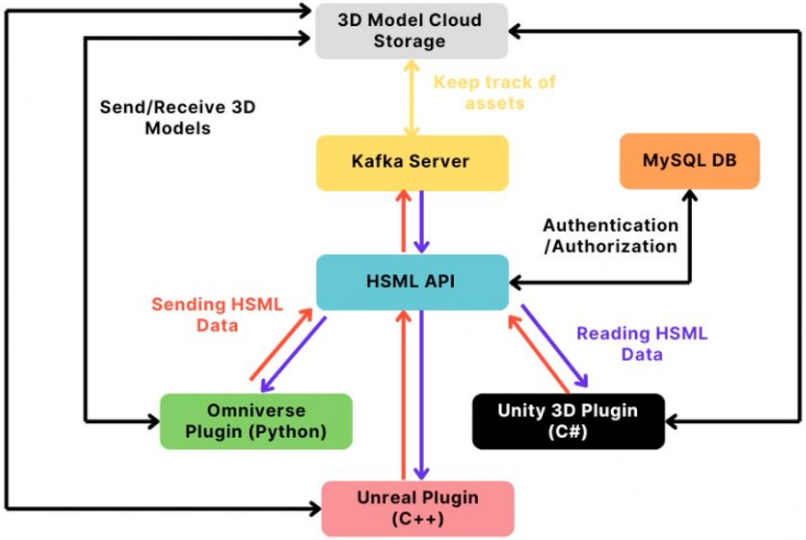

Full Map
Producer
Consumer
Unity
Omniverse
Unreal
Desktop
Computer #
1- Unity
2- Omni
3- Unreal
Unity Consumer (Python Script)
HSML JSON File
Kafka Python Script (Computer 1)
Unreal Consumer (Python Script)
HSML JSON File
Kafka Python Script (Computer 3)
Omniverse Producer (Python Script)
Unity Vip Producer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

Demo System Architecture

Registration



High Level Design



Modified from Jared's Figure