**Day 7 Write simple script to simulate fog nodes & users (just dummy requests).**

**✅ Simple Script: Fog Nodes + Dummy Users**

import simpy

import random

# Define a Fog Node

class FogNode:

def \_\_init\_\_(self, env, name, capacity):

self.env = env

self.name = name

self.capacity = capacity

self.resource = simpy.Resource(env, capacity=capacity)

# Define a User Request process

def user(env, user\_id, fog, service\_time):

print(f"[{env.now}] User-{user\_id} arrives at {fog.name}")

with fog.resource.request() as req:

yield req

print(f"[{env.now}] User-{user\_id} starts service at {fog.name}")

yield env.timeout(service\_time) # Simulate processing time

print(f"[{env.now}] User-{user\_id} finishes at {fog.name}")

# Simulation runner

def run\_simulation():

random.seed(1)

env = simpy.Environment()

# Create two fog nodes

fogA = FogNode(env, "Fog-A", capacity=2)

fogB = FogNode(env, "Fog-B", capacity=1)

# Generate 5 users with random service times

for i in range(5):

fog = random.choice([fogA, fogB]) # Pick a fog randomly

service\_time = random.randint(3, 7)

env.process(user(env, i, fog, service\_time))

# Run the simulation

env.run(until=20)

# Run

if \_\_name\_\_ == "\_\_main\_\_":

run\_simulation()

**🔎 What this does**

* **FogNode** = like a mini data center (with capacity = number of requests it can serve at once).
* **User** = arrives instantly at time 0, requests some processing, and leaves after service\_time.
* **env.timeout()** = simulates work being done.
* At the end, you’ll see logs like:
* [0] User-0 arrives at Fog-B
* [0] User-1 arrives at Fog-A
* [0] User-1 starts service at Fog-A
* [3] User-1 finishes at Fog-A
* ...

👉 This is the **toy base model**.  
From here, we can extend later with:

* arrivals at different times,
* metrics like latency/cost,
* game-theory dynamic pricing.