```
from clusterops import (
    list_available_gpus,
    execute_on_gpu,
)
from swarms import Agent, AgentRearrange
from swarm_models import OpenAIChat
import os
import logging

from dotenv import load_dotenv

load_dotenv()

# Get the OpenAI API key from the environment variable
api_key = os.getenv("OPENAI_API_KEY")

# Create an instance of the OpenAIChat class
model = OpenAIChat(
    openai_api_key=api_key,
    model_name="gpt-4o-mini",
    temperature=0.1,
    max_tokens=2000,
)


# Function for the director agent
```

```
def director_task(task: str):
    logging.info(f"Running Director agent for task: {task}")
    director = Agent(
        agent_name="Director",
        system_prompt="Directs the tasks for the workers",
        llm=model,
        max_loops=1,
        dashboard=False,
        streaming_on=True,
        verbose=True,
        stopping_token="<DONE>",
        state_save_file_type="json",
        saved_state_path="director.json",
    )
    return director.run(task)


# Function for worker 1
def worker1_task(task: str):
    logging.info(f"Running Worker1 agent for task: {task}")
    worker1 = Agent(
        agent_name="Worker1",
        system_prompt="Generates a transcript for a youtube video on what swarms are",
        llm=model,
        max_loops=1,
        dashboard=False,
        streaming_on=True,
        verbose=True,
        stopping_token="<DONE>",
        state_save_file_type="json",
        saved_state_path="worker1.json",
    )
    return worker1.run(task)



# Function for worker 2
def worker2_task(task: str):
    logging.info(f"Running Worker2 agent for task: {task}")
    worker2 = Agent(
        agent_name="Worker2",
        system_prompt="Summarizes the transcript generated by Worker1",
        llm=model,
        max_loops=1,
        dashboard=False,
        streaming_on=True,
        verbose=True,
        stopping_token="<DONE>",
        state_save_file_type="json",
        saved_state_path="worker2.json",
    )
    return worker2.run(task)
```

```
# GPU Assignment Example
def assign_tasks_to_gpus():
    # List available GPUs
    gpus = list_available_gpus()
    logging.info(f"Available GPUs: {gpus}")

    # Example: Assign Director to GPU 0
    logging.info("Executing Director task on GPU 0")
    execute_on_gpu(
        0, director_task, "Direct the creation of swarm video format"
    )

    # Example: Assign Worker1 to GPU 1
    logging.info("Executing Worker1 task on GPU 1")
    execute_on_gpu(
        1,
        worker1_task,
        "Generate transcript for youtube video on swarms",
    )

    # Example: Assign Worker2 to GPU 2
    logging.info("Executing Worker2 task on GPU 2")
    execute_on_gpu(
        2,
        worker2_task,
        "Summarize the transcript generated by Worker1",
    )



# Flow Management using AgentRearrange (optional)
def run_agent_flow():
    # Initialize the agents
    director = Agent(
        agent_name="Director",
        system_prompt="Directs the tasks for the workers",
        llm=model,
        max_loops=1,
        dashboard=False,
        streaming_on=True,
        verbose=True,
        stopping_token="<DONE>",
        state_save_file_type="json",
        saved_state_path="director.json",
    )

    worker1 = Agent(
        agent_name="Worker1",
        system_prompt="Generates a transcript for a youtube video on what swarms are",
        llm=model,
        max_loops=1,
        dashboard=False,
        streaming_on=True,

        verbose=True,

        stopping_token="<DONE>",

        state_save_file_type="json",

        saved_state_path="worker1.json",

    )


    worker2 = Agent(

        agent_name="Worker2",

        system_prompt="Summarizes the transcript generated by Worker1",

        llm=model,

        max_loops=1,

        dashboard=False,

        streaming_on=True,

        verbose=True,

        stopping_token="<DONE>",

        state_save_file_type="json",

        saved_state_path="worker2.json",

    )


    # Define agent list and flow pattern

    agents = [director, worker1, worker2]

    flow = "Director -> Worker1 -> Worker2"


    # Use AgentRearrange to manage the flow

    agent_system = AgentRearrange(agents=agents, flow=flow)
```

```
    output = agent_system.run(
            "Create a format to express and communicate swarms of llms in a structured manner for
youtube"
    )
    print(output)



if __name__ == "__main__":
    logging.info(
        "Starting the GPU-based task assignment for agents..."
    )
    assign_tasks_to_gpus()

    logging.info("Starting the AgentRearrange task flow...")
    run_agent_flow()
```