```
import os

from fastapi import FastAPI, Request
from fastapi.middleware.cors import CORSMiddleware
from swarms_cloud.schema.agent_api_schemas import (
    ParallelSwarmAPIInput,
    ParallelSwarmAPIOutput,
)
from swarms_cloud.schema.swarm_schema import SwarmAPISchema, AllSwarmsSchema
from swarms_cloud.utils.create_agent import create_agent_sync

# Create a FastAPI app
app = FastAPI(
    debug=True,
    title="Parallel Swarm API",
    version="0.1.0",
)

# Load the middleware to handle CORS
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

```
@app.get("/")
def read_root():
    return {"Hello": "World"}


@app.get("/health")
def health():
    return {"status": "ok"}


@app.get("/version")
def version():
    return {"version": "0.1.0"}


@app.post("v1/swarms/parallel/create/{swarm_id}", response_model=SwarmAPISchema)
def create_parallel_swarm(request: Request, swarm_input: ParallelSwarmAPIInput):
    task = swarm_input.task

    created_agents = []

    # Parse the schema for all the agents
    for agent in swarm_input.agents:
        created_agents.append(create_agent_sync(agent))
```

```
    # Now execute all the agents in parallel
    import concurrent.futures

    with concurrent.futures.ThreadPoolExecutor() as executor:
        futures = [executor.submit(agent.run, task) for agent in created_agents]

        # Wait for all the tasks to complete
        [future.result() for future in concurrent.futures.as_completed(futures)]

    #



@app.post(
    "v1/swarms/parallel/{swarm_id}/completions", response_model=ParallelSwarmAPIOutput
)
def run_parallel_swarm_completions(
    request: Request, swarm_input: ParallelSwarmAPIInput
):
    task = swarm_input.task

    created_agents = []

    # Parse the schema for all the agents
    for agent in swarm_input.agents:
        created_agents.append(create_agent_sync(agent))
```

```
    # Now execute all the agents in parallel
    import concurrent.futures

    with concurrent.futures.ThreadPoolExecutor() as executor:
        futures = [executor.submit(agent.run, task) for agent in created_agents]

        # Wait for all the tasks to complete
        [future.result() for future in concurrent.futures.as_completed(futures)]

    # log_entry = ParallelSwarmAPIOutput(
    #     completions=MultipleAgentOutputs(
    #         agents =
    #     )
    # )


@app.post("v1/swarms", response_model=AllSwarmsSchema)
def get_all_swarms(
    request: Request,
    Swa,
):
    return AllSwarmsSchema(
        swarms=[
            SwarmAPISchema(
                id="1",
```

```
            swarm_name="Swarm API",
            swarm_description="Swarm API description",
            created_at=1628584185,
            owned_by="TGSC",
            tags=["tag_1", "agent"],
            use_cases={
                "use_case_1": "Use case 1 description",
                "use_case_2": "Use case 2 description",
            },
        )
    ]
)



if __name__ == "__main__":
    import uvicorn

    uvicorn.run(
        app,
        host="0.0.0.0",
        port=os.getenv("AGENT_PORT"),
        use_colors=True,
        log_level="info",
    )
```