Dapr in Action

Event-Driven AI Agents with Dapr

Dana Arsovska
Community Manager @Dapr
Marc Duiker
Community Manager @Dapr

Agenda

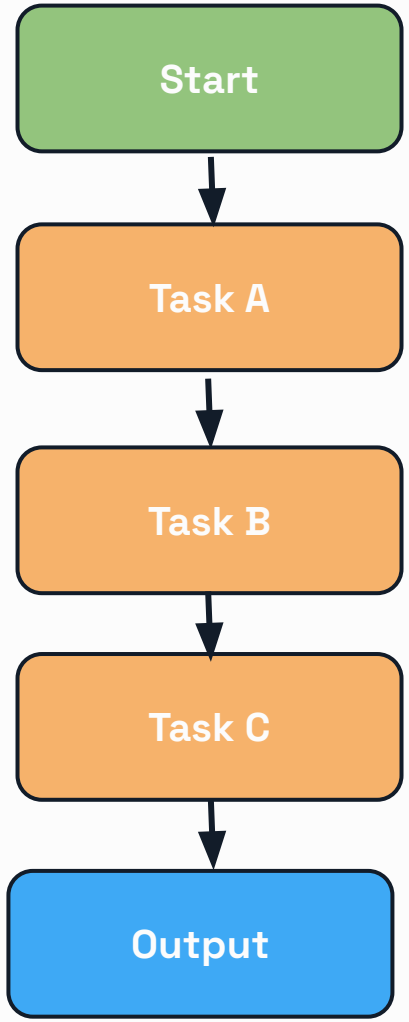
- What is Agentic AI?
- Challenges of productionizing Agentic AI
- Dapr & Dapr Agents
- Workshop
- Q&A

Agentic AI

What are Agentic Systems?

Agentic systems are systems where Large Language Models (LLMs), with varying degrees of autonomy, maintain control over how they accomplish complex tasks.

Rule-Based Automation

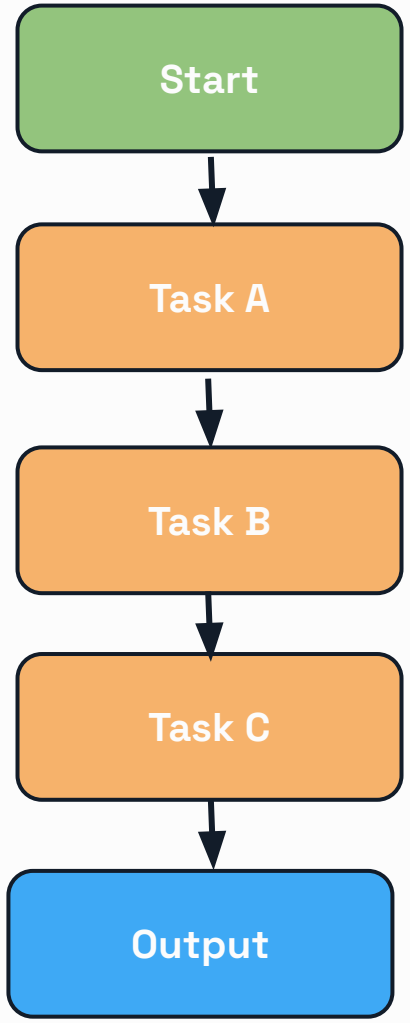


System that executes predefined, rule-based tasks automatically

✅ Highly deterministic

❌ Cannot adapt to new scenarios automatically

Rule-Based Automation



Document Parser

INVOICE

Invoice #: 12345

Date: 03/15/2024

Amount: $1,250.00

Due: 04/15/2024

Fixed Rules

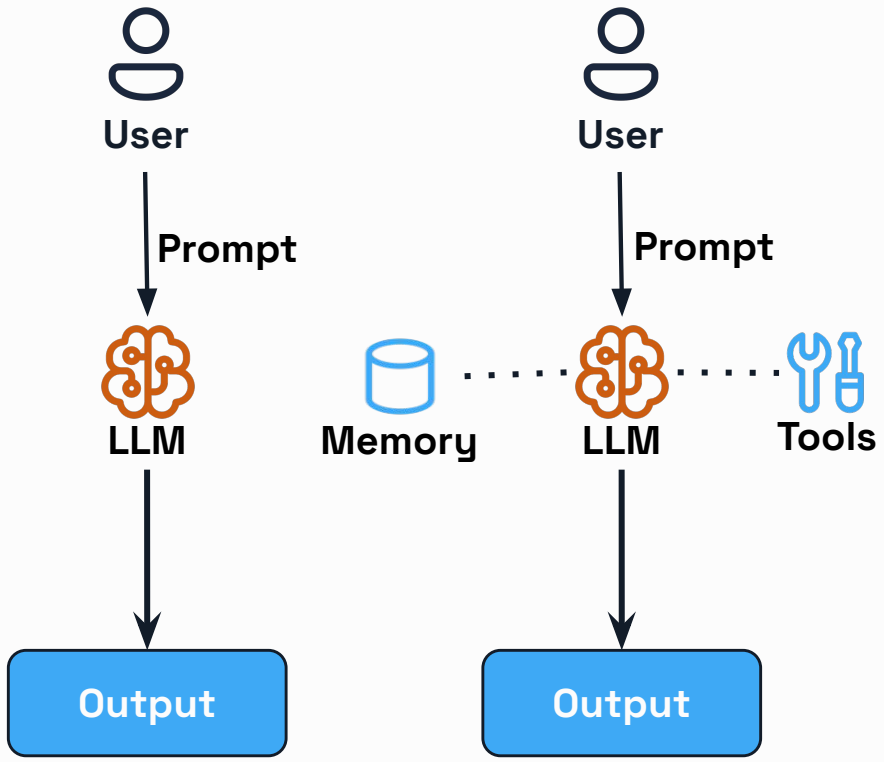
Invoice #:\s*(\d+)

Date:\s*(\d{2}/\d{2}/\d{4})

Limitations

- Breaks if format changes
- Manual updates for new document types

Evolution of Agentic AI: LLMs

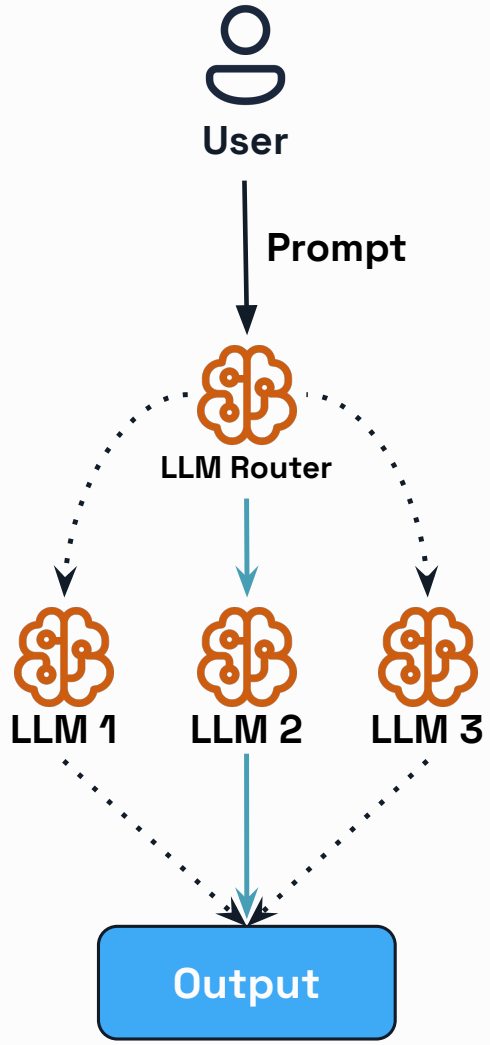


LLMS: Brain of the Agent (pretrained knowledge)

Tools: Real-time, proprietary, or specialized data

Memory: Use past data to improve decision making

Evolution of Agentic AI: Workflows



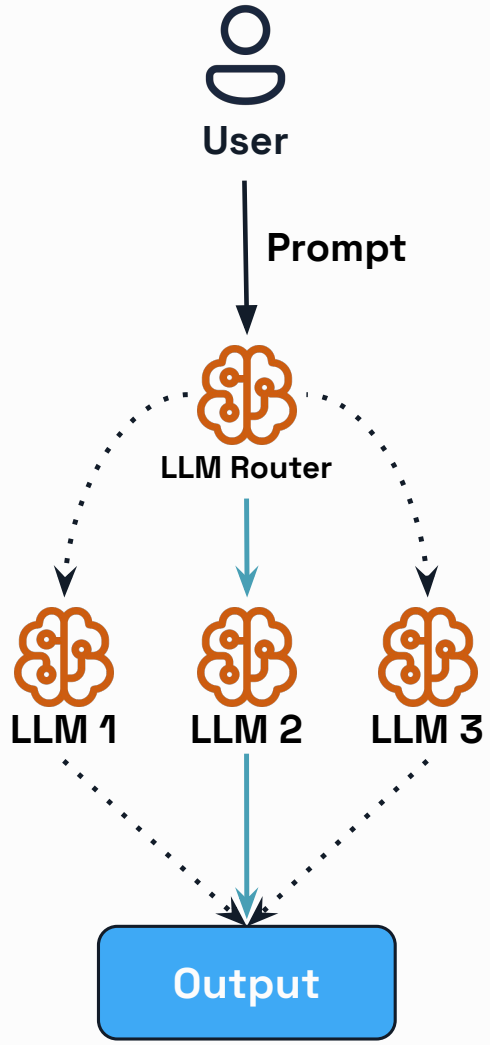
Systems where LLMs and tools are orchestrated through predefined code paths

✅ Deterministic tasks requiring flexibility

✅ They are goal-driven, not just rule-driven

❌ Not adaptive, cannot reason about unexpected situations

Evolution of Agentic AI: Workflows



Incoming Support Ticket

Message: "Hi, I keep getting error code 503 when trying to pay my invoice through your portal."

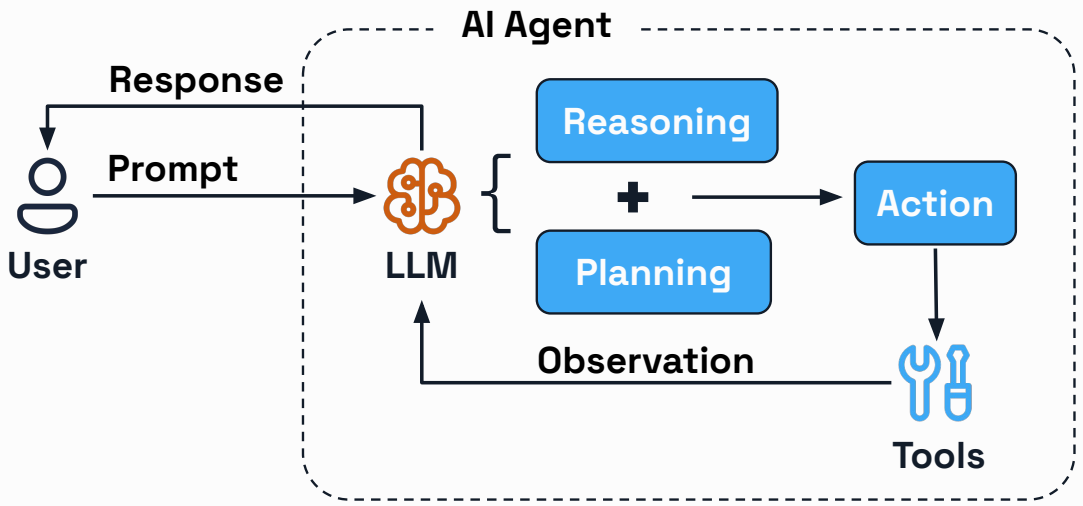
Classification: IT ISSUE

→ Routed to Account Support Team

Limitations

- Predefined path: extract → classify → route

Evolution of Agentic AI: AI Agents



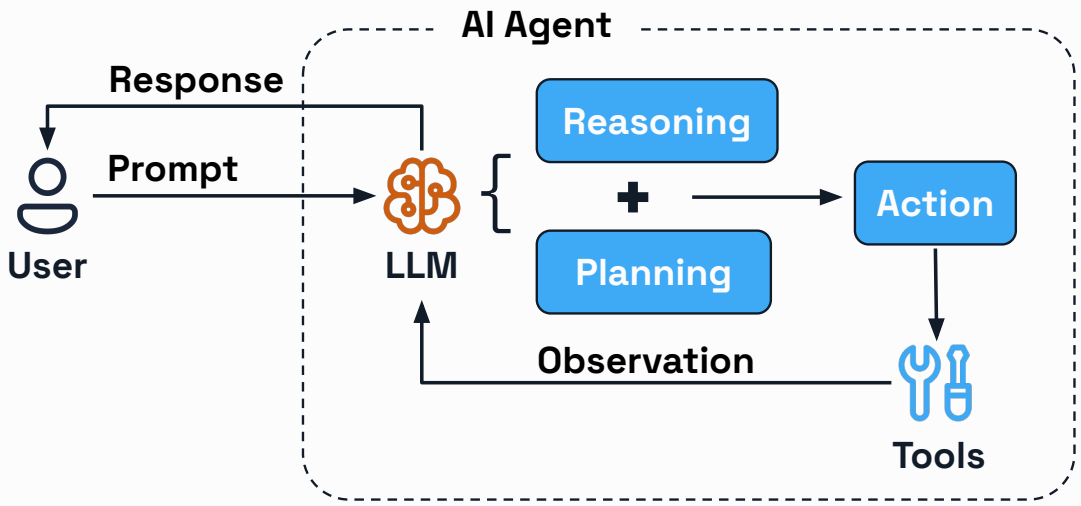
Systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks*

✅ Open-ended problems

❌ Less reliable

*Anthropic - Building effective agents: https://www.anthropic.com/engineering/building-effective-agents

Evolution of Agentic AI: AI Agents



Incoming Support Ticket

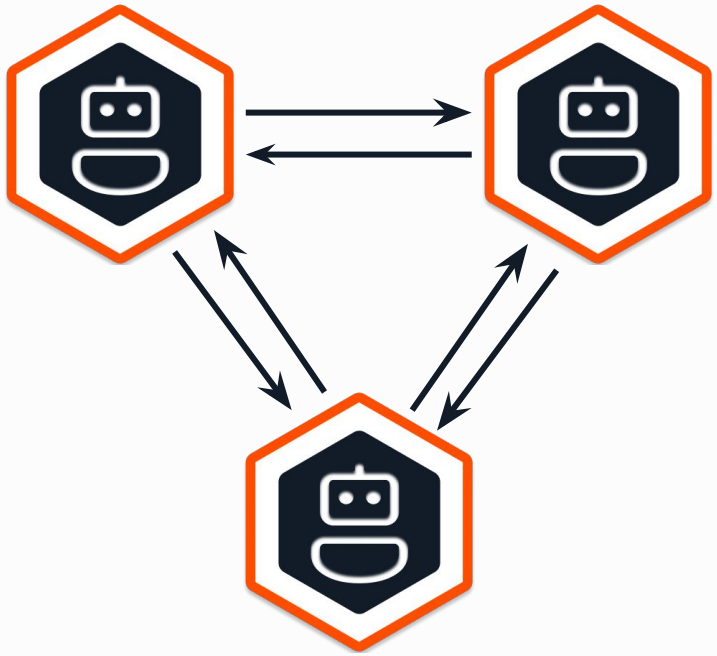
Message: "Hi, Your system charged my card 5,000 instead of 500, now my account is locked, and I can't access the reports."

→ Routed to multiple teams

Limitations

- Possibility of cascading mistakes

Evolution of Agentic AI: Agent Mesh



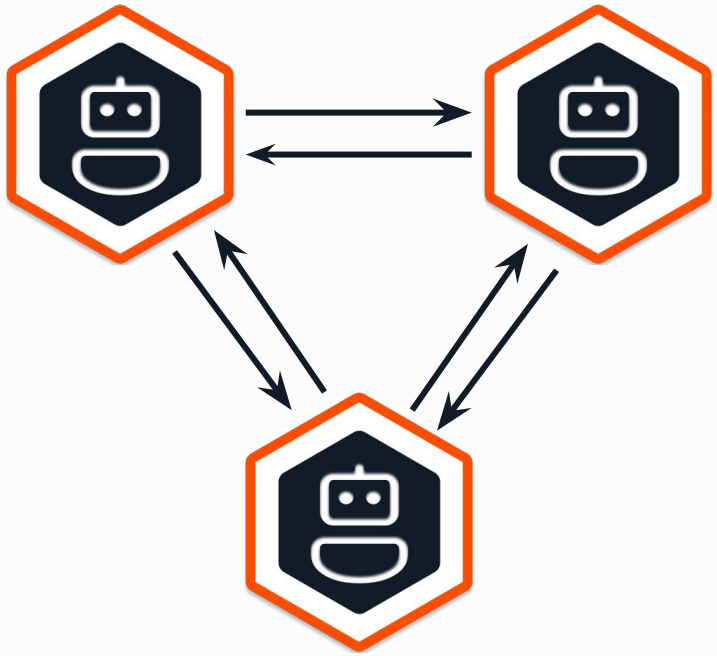
Group of Agents with varying autonomy collaborating to achieve complex goals

✅ Can solve complex, interdependent tasks

❌ Complexity in coordination

❌ Harder to debug and predict outcomes

Evolution of Agentic AI: Agent Mesh

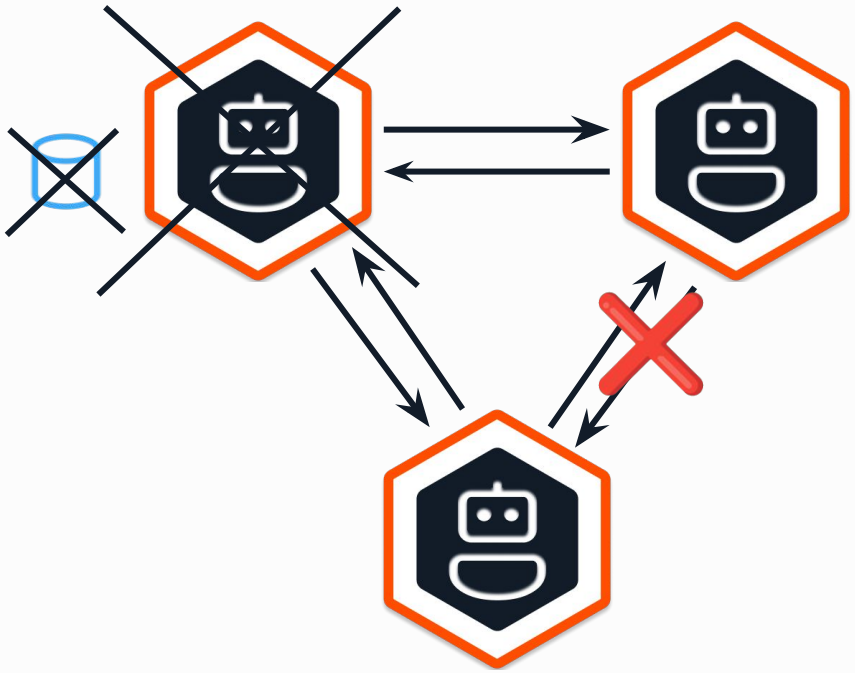


- Intake Agent gathers customer history, previous tickets, account status
- Technical Agent runs initial diagnostics, checks system logs
- Account Agent pulls billing history, subscription details
- Resolver agent suggests solution

→ Customer support gets: Complete customer profile + preliminary analysis instead of starting from scratch

Productionizing Agentic AI

Why Productionizing Agentic AI is Hard



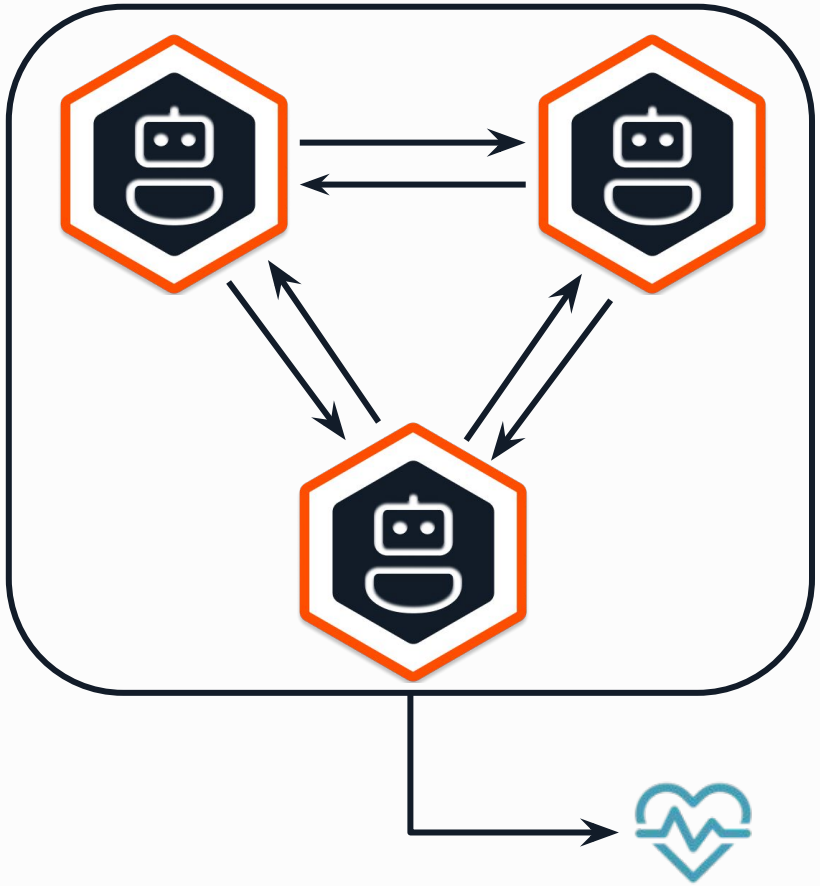
Scalability: run hundreds of agents reliably

Fault tolerance: recovering from network or task failures

Collaboration: asynchronous communication, agent discoverability

State: persist memory across conversations and sessions

Why Productionizing Agentic AI is Hard



Scalability: run hundreds of agents reliably

Fault tolerance: recovering from network or task failures

Collaboration: asynchronous communication, agent discoverability

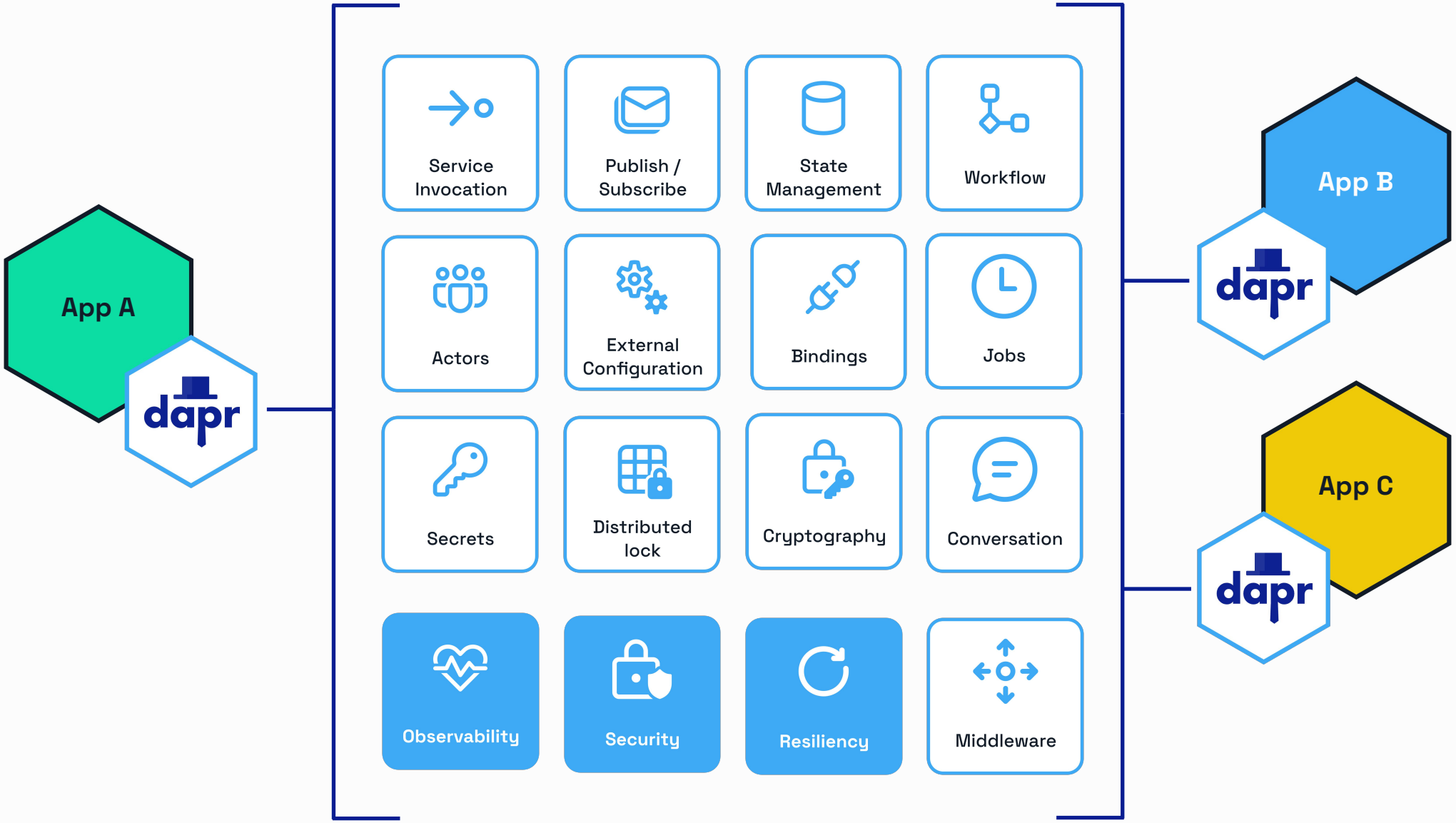
State: persist memory across conversations and sessions

Observability: understanding outcomes and performance

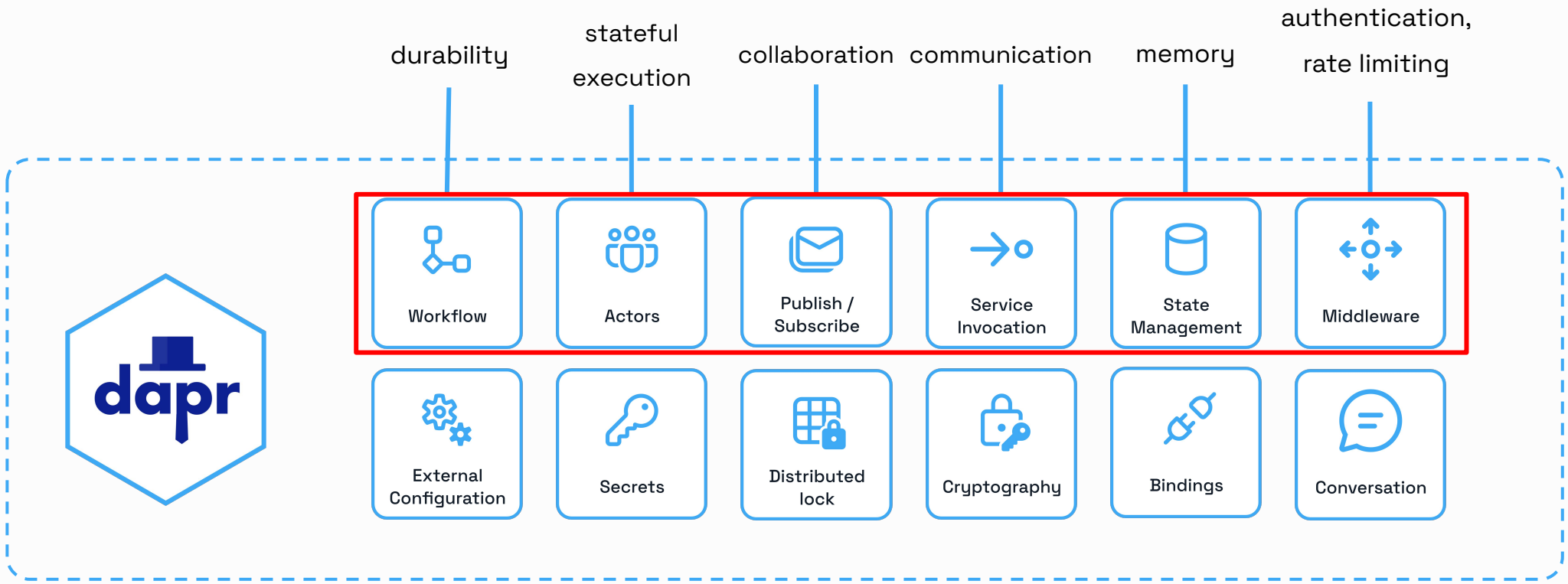
Dapr Agents

https://dapr.github.io/dapr-agents/

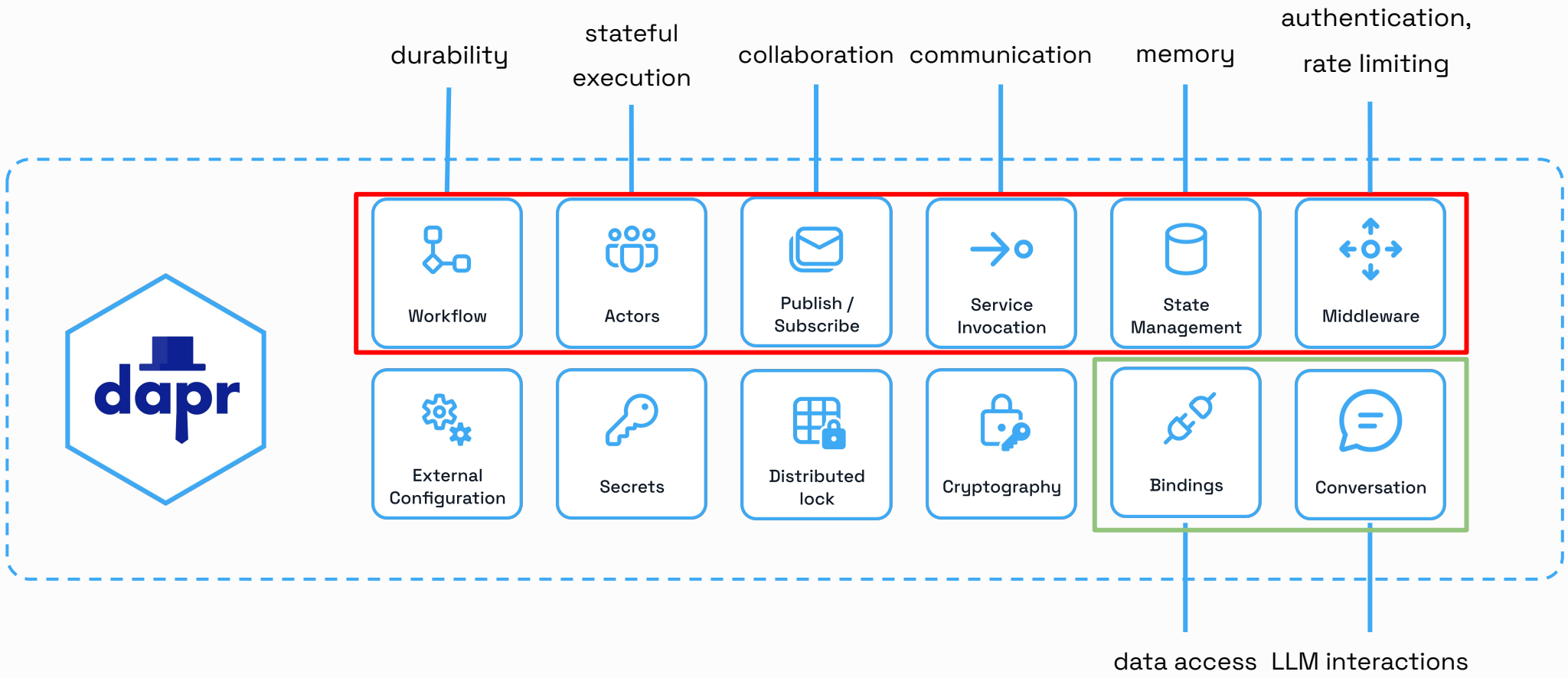
Dapr – Application Developer Platform



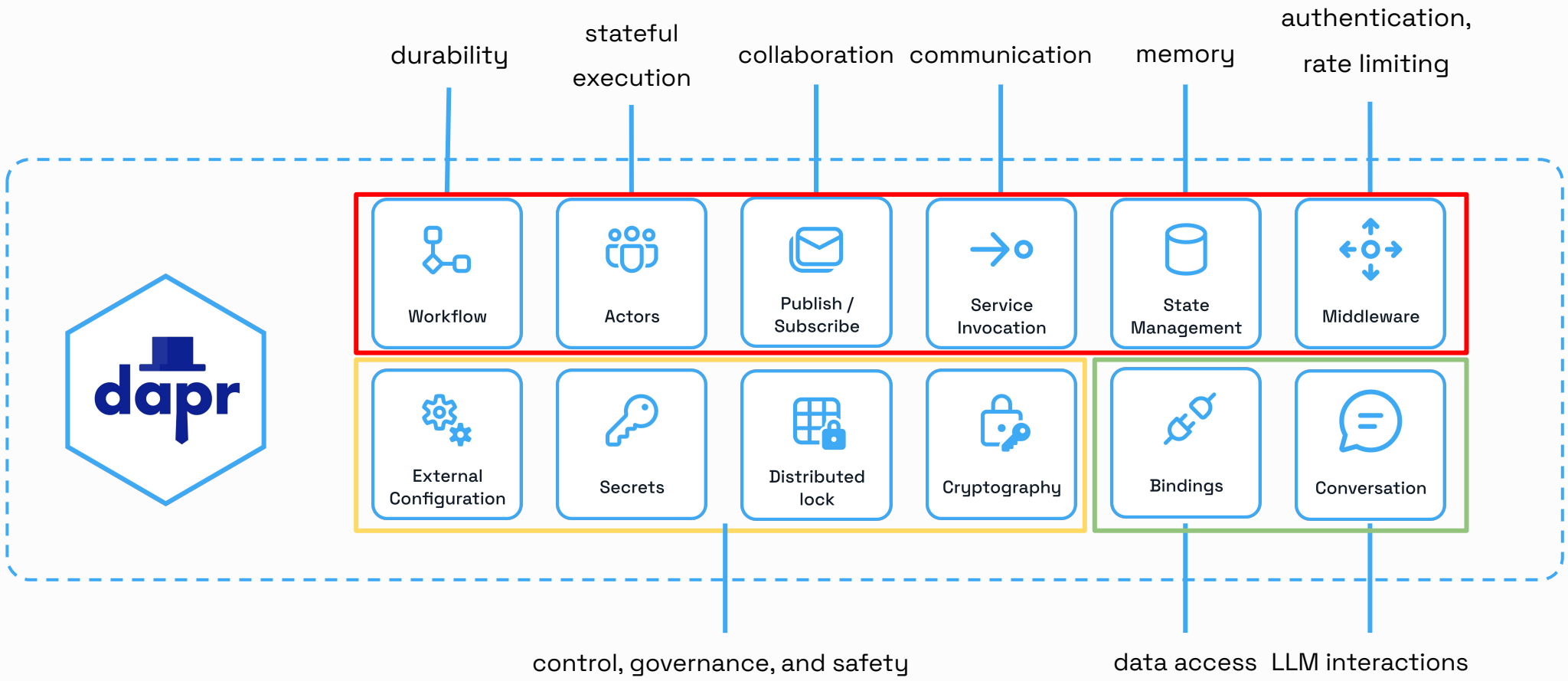
Where does Dapr fit in?



Where does Dapr fit in?



Where does Dapr fit in?



Dapr Agents

Dapr Agents is a developer framework designed to build production-grade resilient AI agent systems that operate at scale.

- Run thousands of agents efficiently on a single core
- Automatic retries
- Direct databases integration
- Observable by default
- Vendor-neutral & open source
- Built-in RBAC and access scopes

https://github.com/dapr/dapr-agents

```python
import dapr.ext.workflow as wf
from openai import OpenAI
```

```python
wfr = wf.WorkflowRuntime()  # Initialize Workflow runtime
client = OpenAI()  # Initialize OpenAI client

def call_openai(prompt: str, model: str = "gpt-4o") -> str:
    """Reusable function to call OpenAI's chat completion API."""
    response = client.chat.completions.create(
        messages=[{"role": "user", "content": prompt}],
        model=model,
    )
    return response.choices[0].message.content.strip()
```

Initializations

```python
# Activity 1: Pick a random LOTR character
@wfr.activity(name="pick_character")
def pick_character(ctx):
    character = call_openai(
"Return a random Lord of the Rings character's name." )
    return character
```

Create an activity

```python
# Define Workflow logic
@wfr.workflow(name="lotr_workflow")
def task_chain_workflow(ctx: wf.DaprWorkflowContext):
    character = yield ctx.call_activity(pick_character))
    return quote
```

Create a durable workflow

```python
import dapr.ext.workflow as wf
from openai import OpenAI

client = OpenAI()  # Initialize OpenAI client
wfr = wf.WorkflowRuntime()  # Initialize Workflow runtime

def call_openai(prompt: str, model: str = "gpt-4o") -> str:
    """Reusable function to call OpenAI's chat completion API."""
    response = client.chat.completions.create(
        messages=[{"role": "user", "content": prompt}],
        model=model,
    )
    return response.choices[0].message.content.strip()
```

```python
# Activity 1: Pick a random LOTR character
@wfr.activity(name="pick_character")
def pick_character(ctx):
    character = call_openai(
"Return a random Lord of the Rings character's name." )
    return character
```

```python
# Define Workflow logic
@wfr.workflow(name="lotr_workflow")
def task_chain_workflow(ctx: wf.DaprWorkflowContext):
    character = yield ctx.call_activity(pick_character))
    return quote
```

```python
from dapr_agents.workflow import WorkflowApp, workflow, task
from dapr.ext.workflow import DaprWorkflowContext
```

```python
# Task 1: Pick a random LOTR character
@task(description="Return a random Lord of the Rings
character's name.")
def get_character() -> str:
    pass
```

```python
# Define Workflow logic
@workflow(name='lotr_workflow')
def task_chain_workflow(ctx: DaprWorkflowContext):
    character = yield ctx.call_activity(get_character)
    return character
```

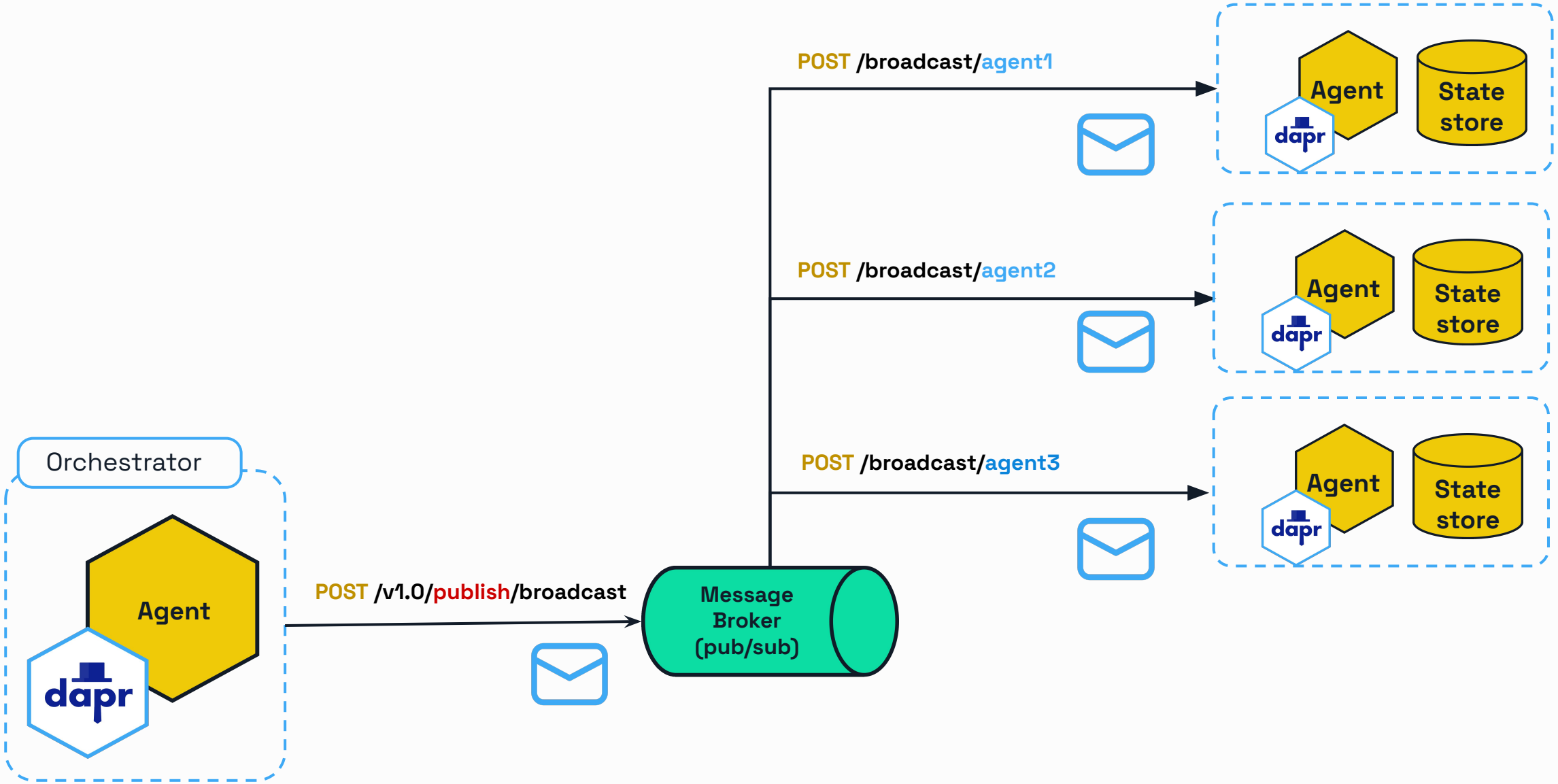
Durable Agents

Define a Durable Agent

```
from dapr_agents import DurableAgent


weather_agent = DurableAgent(
    role="Weather Assistant",
    name="Stevie",
    goal="Help humans get weather and location info using smart tools.",
    instructions=[
        "Respond clearly and helpfully to weather-related questions.",
        "Use tools when appropriate to fetch real-time weather data."],
    tools=tools
  )
```

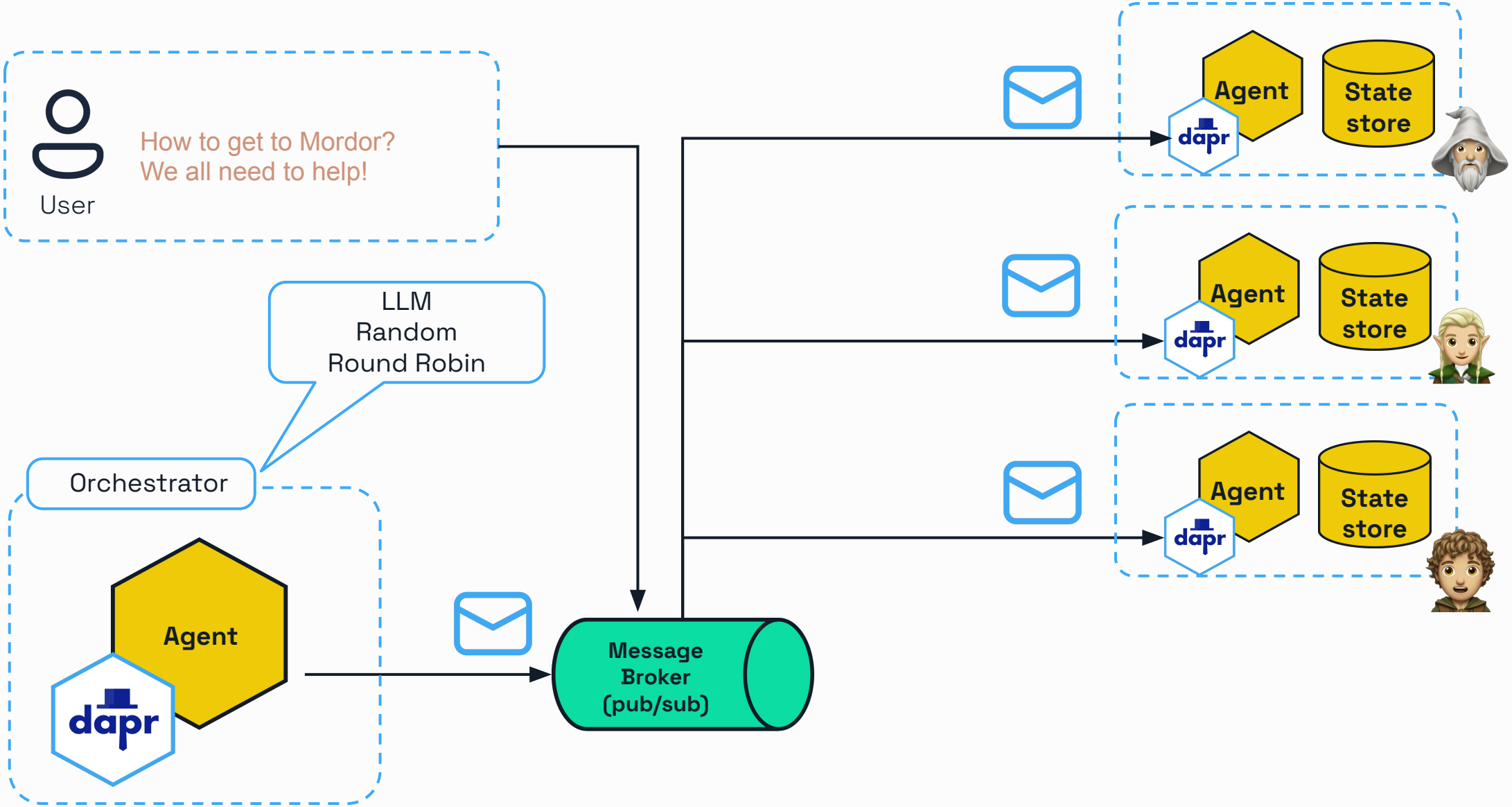
Multi-Agent Collaboration



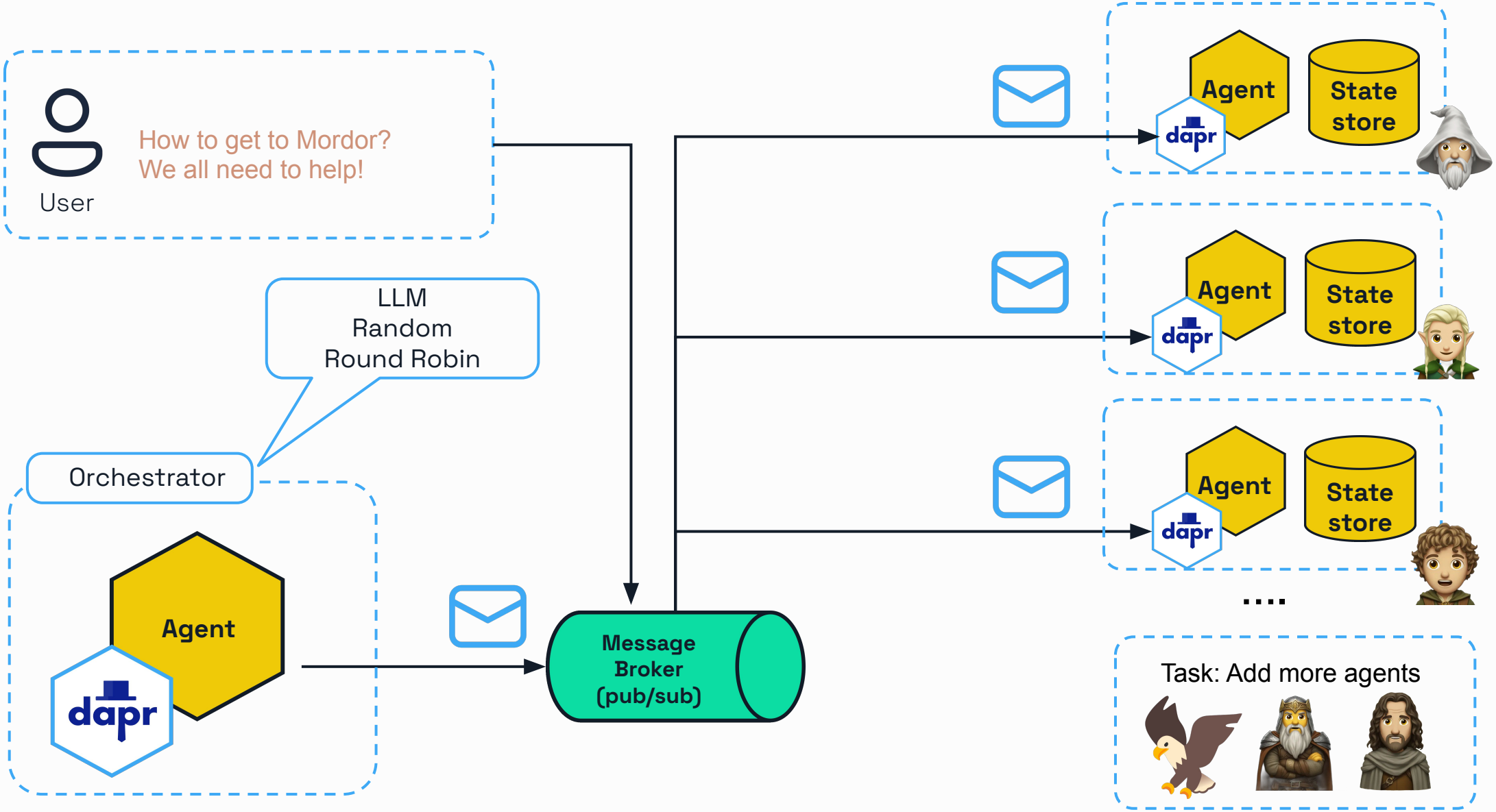
Workshop Overview

https://github.com/Dzvezdana/pydata_workshop_september2025

Collaborative Multi-Agent Workflow



Collaborative Multi-Agent Workflow



Our Stack

Dapr Agents

- Framework to build AI Agent Systems

HuggingFace API

- API access to various LLMs

Collaborative Multi-Agent Workflow

Local setup

- Clone this git repo
- Follow the instructions in the README.
- You are ready to start!

Github repo

Slides

Privatebin password:

pydata_ams2025_dapr

Dapr Resources

dapr.io

bit.ly/dapr-youtube

bit.ly/dapr-quickstarts

bit.ly/dapr-discord

@daprdev

@daprdev.bsky.social

dapr.io

Claim the Dapr Community Supporter badge!

bit.ly/dapr-supporter