Temporal

Durable MCP

JOIN OUR SLACK CHANNEL

Feel free to ask questions or go there for helpful links!

https://t.mp/nov-18-slack-channel

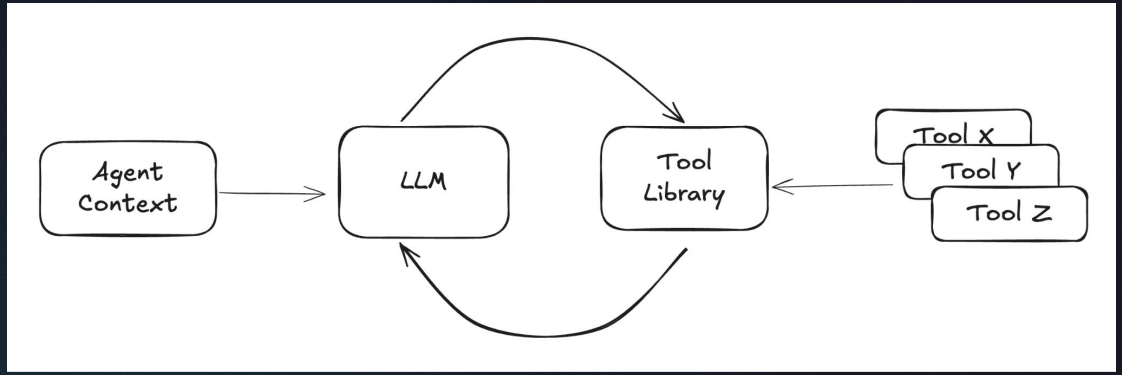
LEARNING OBJECTIVES

- ✅ Explore the role that MCP plays in AI applications
- ✅ Review MCP fundamentals
- ✅ Build durability into your MCP tools with Temporal
- ✅ Debug and monitor MCP tools with Temporal Web UI
- ✅ Implement long running MCP tools with Temporal (queries, signals)

Temporal

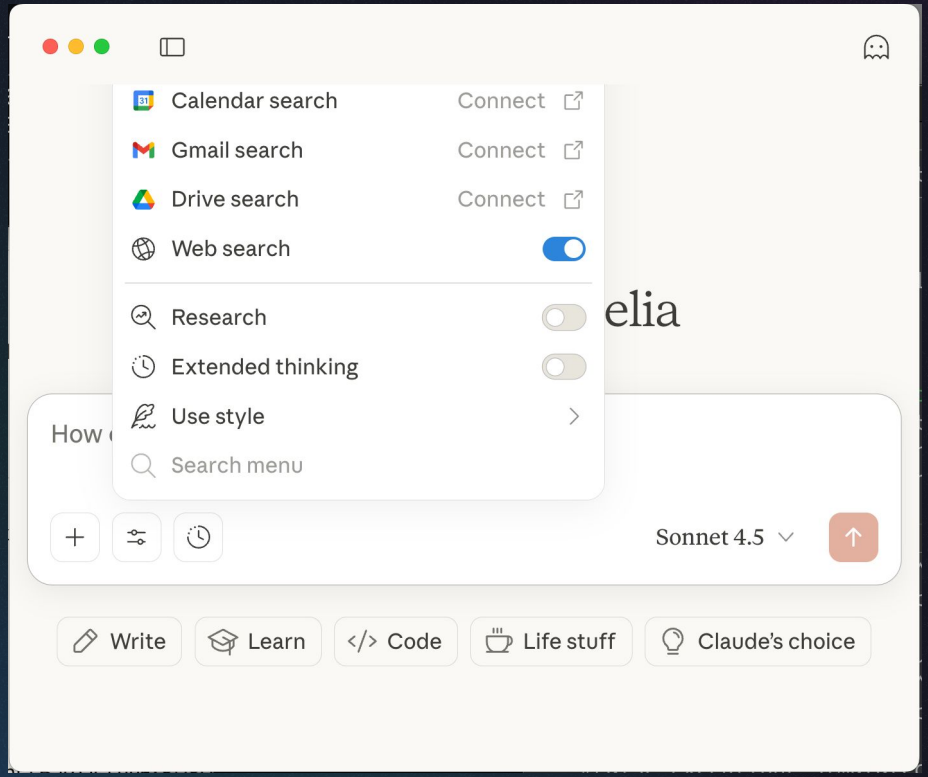
Foundations: Building MCP Servers with Temporal

WHAT ARE WE TRYING TO ACHIEVE?



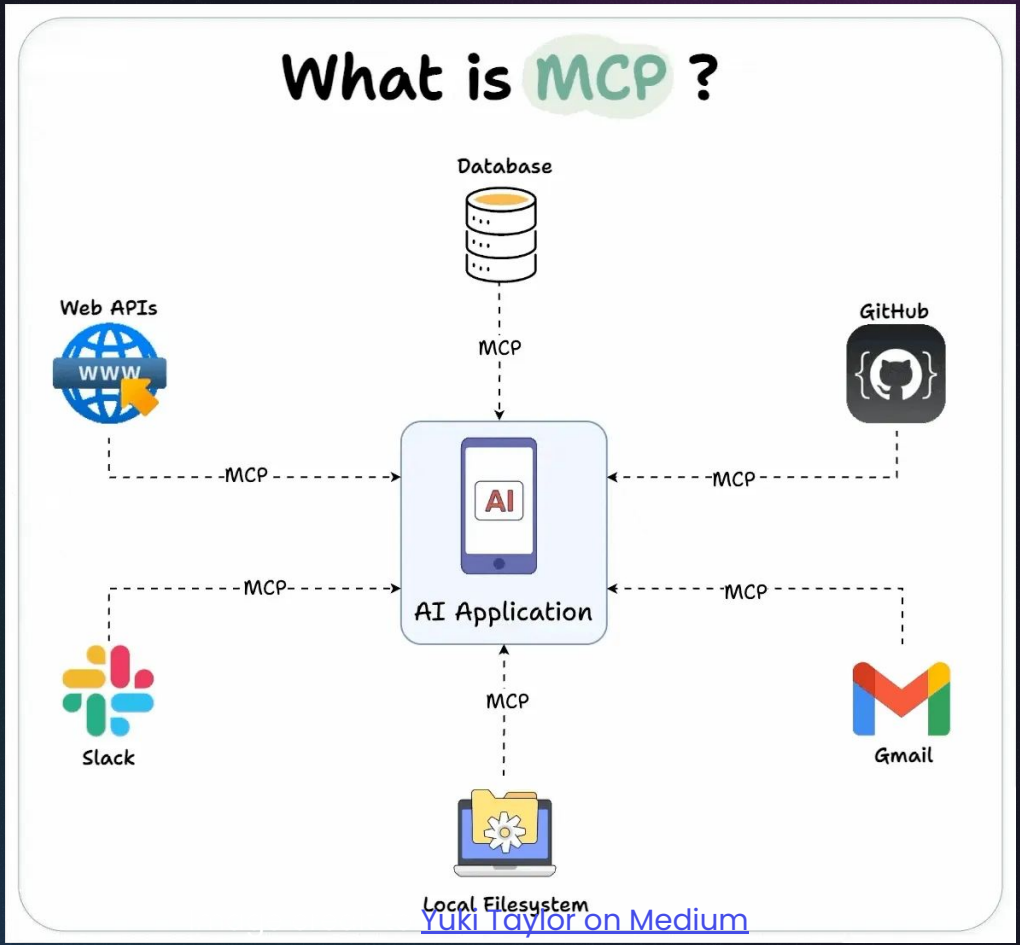
- Instead of building for-purpose agents...
- ... define agents through their instructions and a set of tools they are supplied

AND AI “BROWSERS” HAVE AGENTIC BEHAVIOR

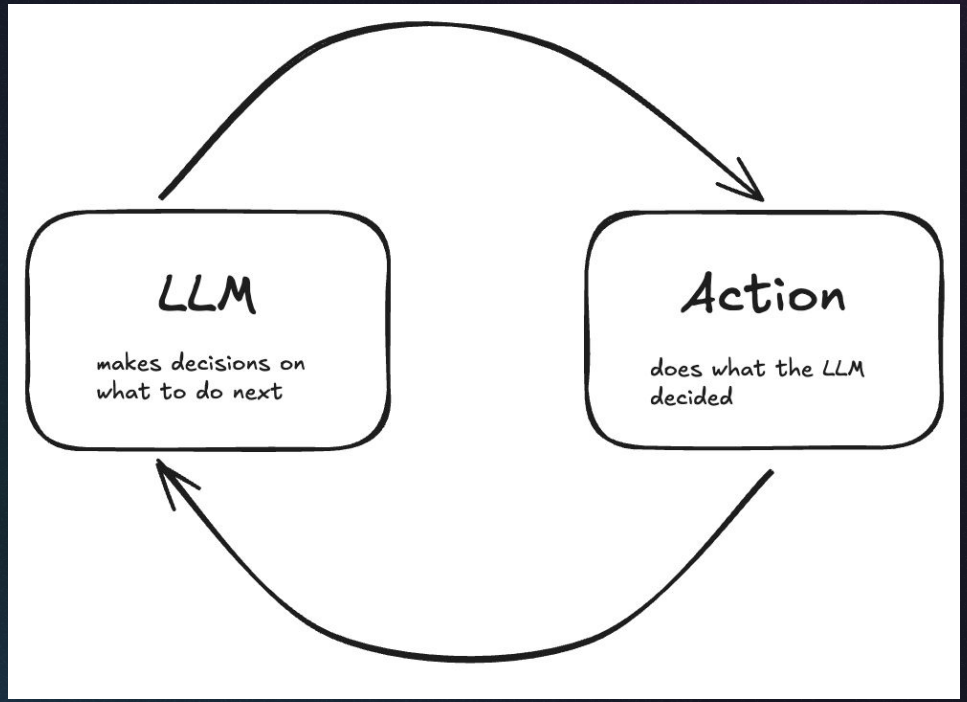


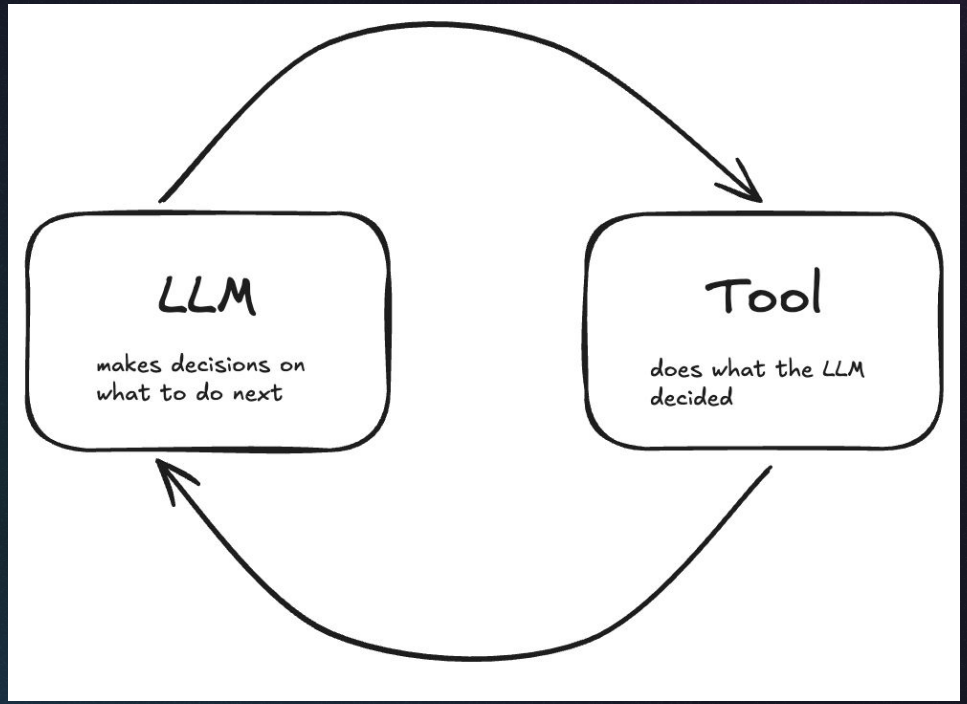
MCP IS THE DEFACTO STANDARD FOR AGENT EXTENSION AND CONFIGURATION

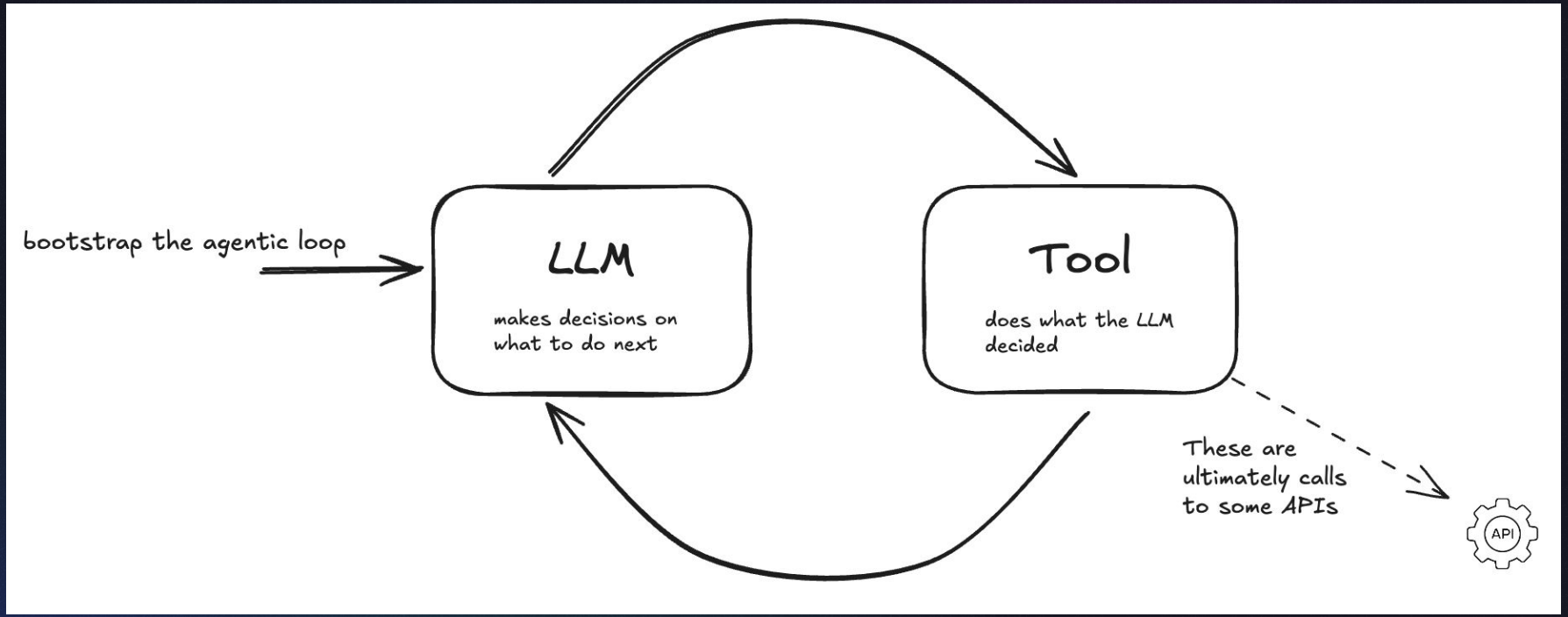
IT ALSO ALLOWS
FOR-PURPOSE AGENTS
TO CONNECT TO EXTERNAL SERVICES



Yuki Taylor on Medium


LLM
makes decisions on what to do next
Action
does what the LLM decided


LLM
makes decisions on what to do next
Tool
does what the LLM decided


bootstrap the agentic loop
LLM
makes decisions on what to do next
Tool
does what the LLM decided
These are ultimately calls to some APIs
API

MCP OVERVIEW

MCP SERVER, IN PRINCIPLE

- Placeholder slide - what I want to do here is establish a way of thinking about an MCP server - what it represents, not just drill into the pieces and parts.
- I.e. ChatGPT applications

(maybe not in the notebooks)

MCP HAS TWO PARTS

Primitives

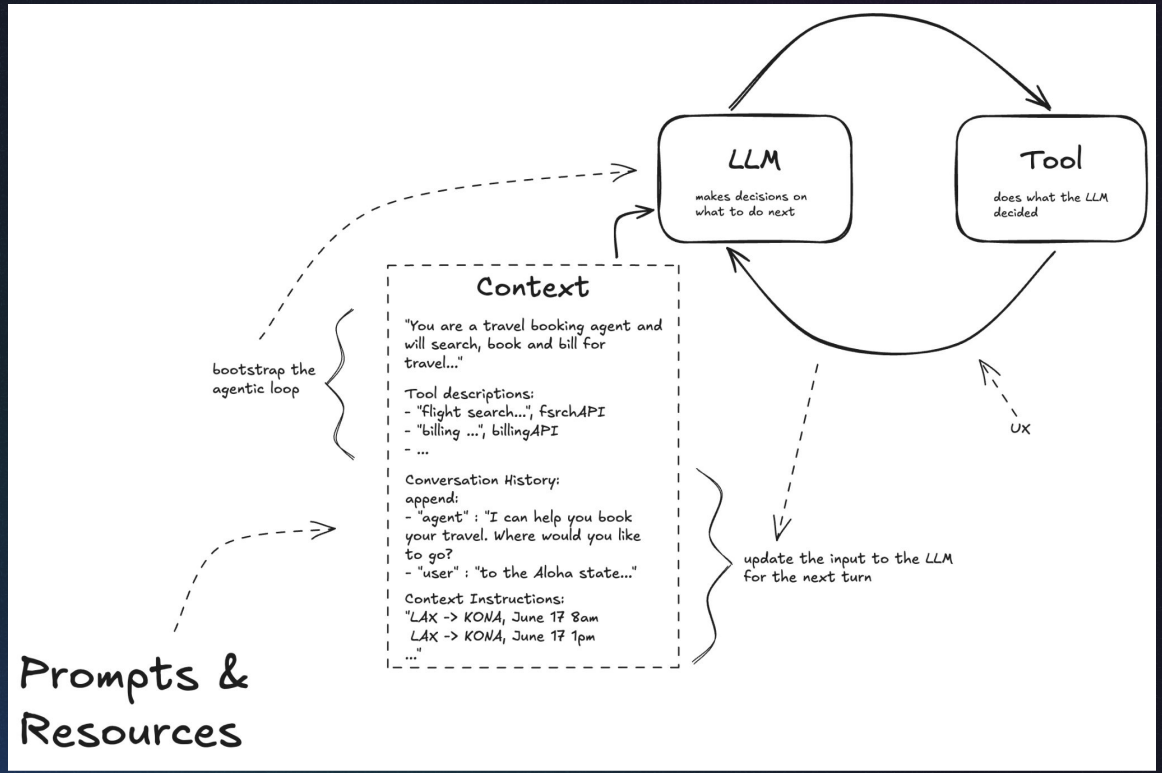
- Prompts
- Resources (i.e. files)
- Tools (i.e. Agent-ready APIs)

Client Server Architecture

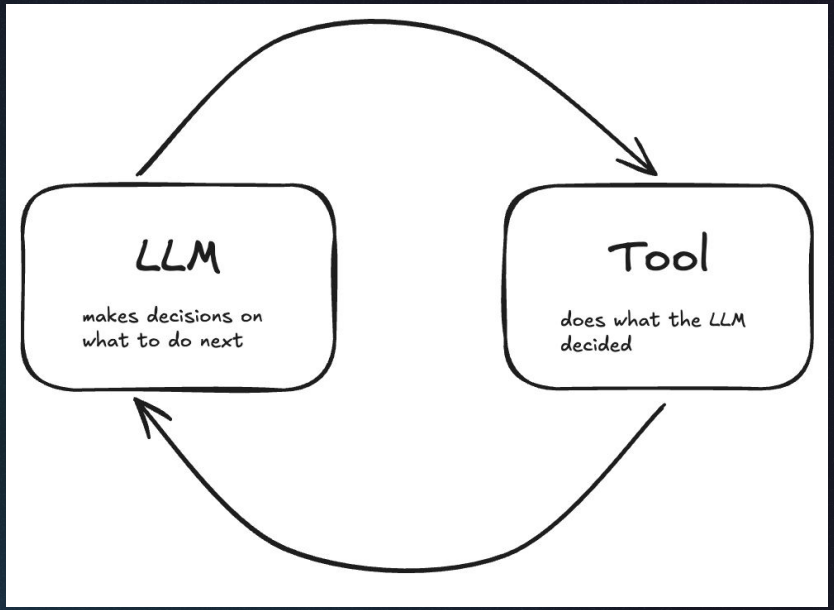
- MCP Clients
- MCP Servers
- A transport protocol between them

* Note: there are a few other concepts, but these are the core ones

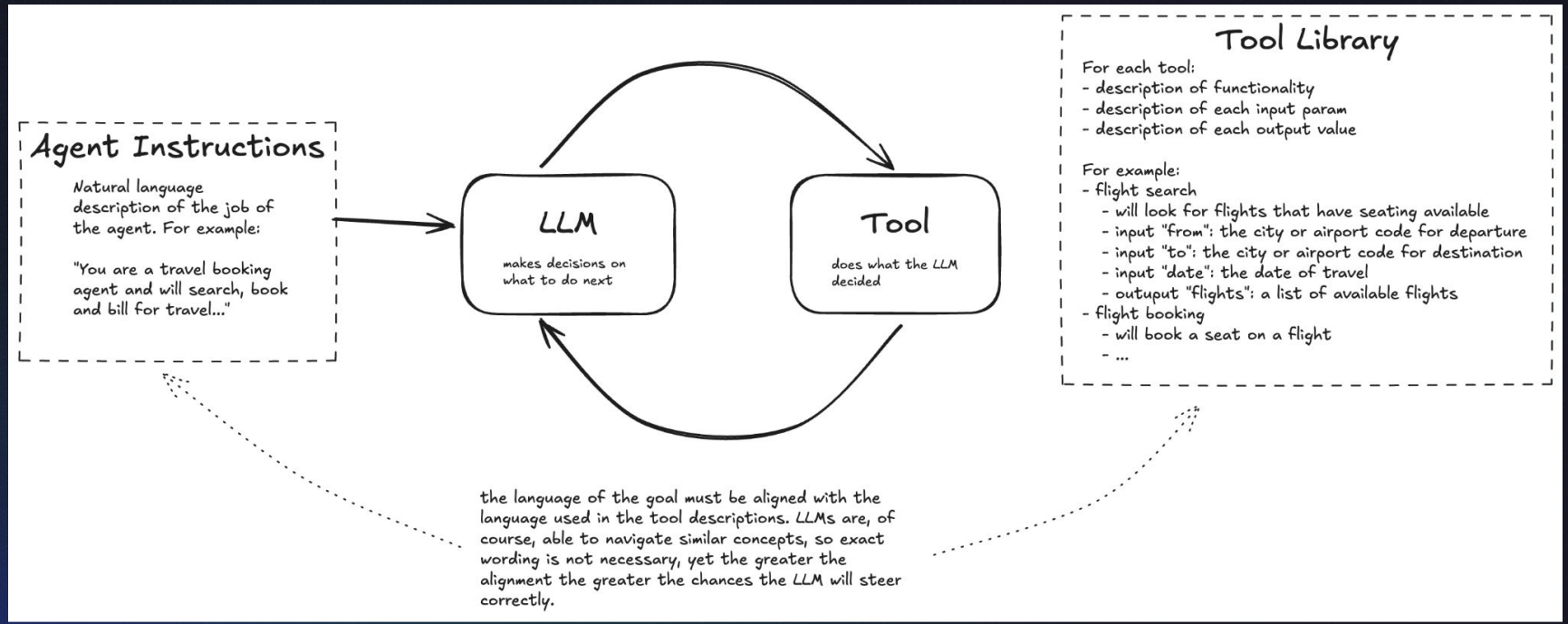
PROMPTS AND RESOURCES → CONTEXT WINDOW



TOOLS = AGENT-READY APIS



"AGENT-READY" APIS MEANS LANGUAGE ALIGNED



MCP HAS TWO PARTS

Primitives

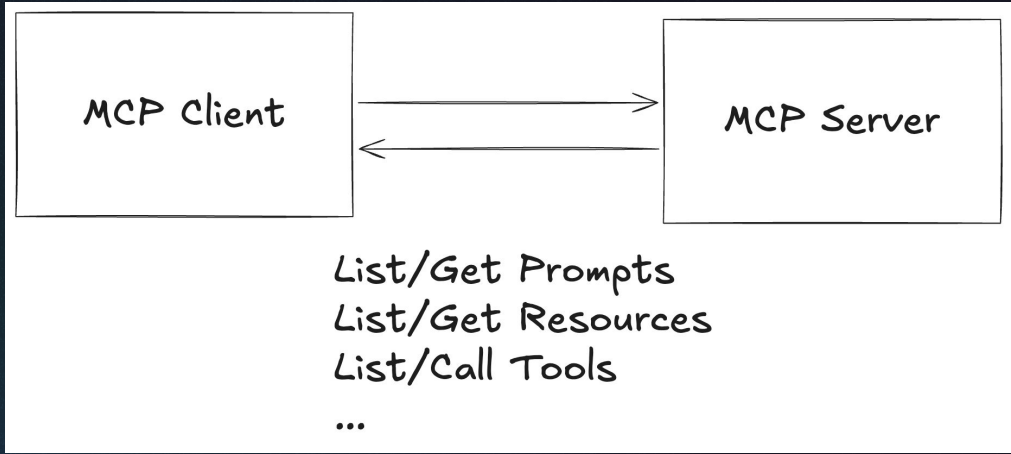
- Prompts
- Resources (i.e. files)
- Tools (i.e. Agent-ready APIs)

Client Server Architecture

- MCP Clients
- MCP Servers
- A transport protocol between them

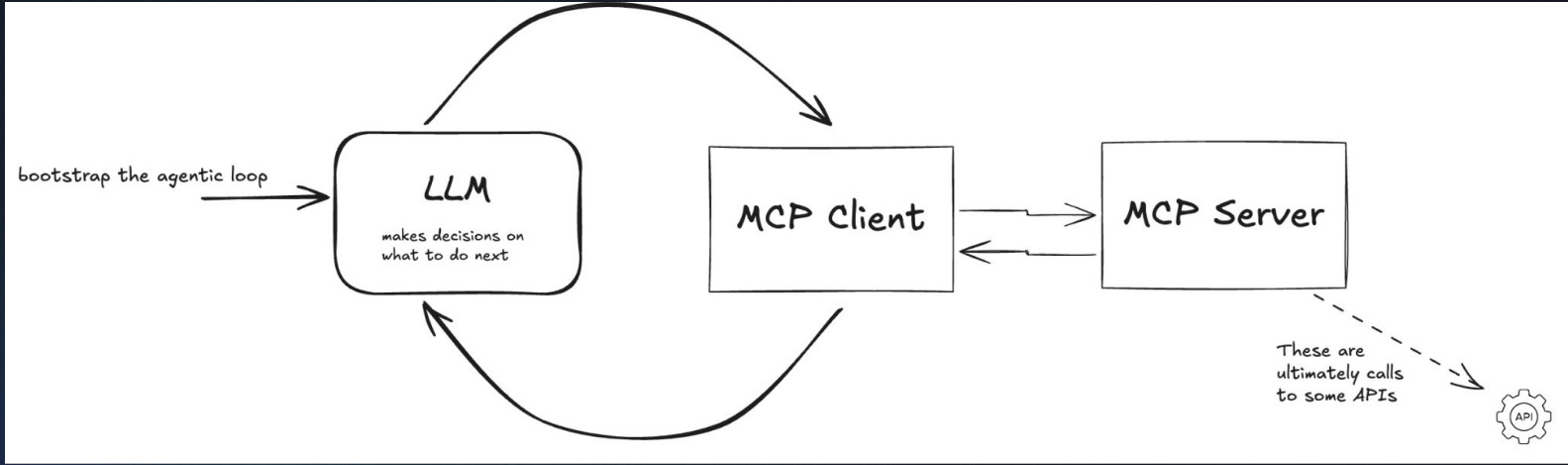
* Note: there are a few other concepts, but these are the core ones

MCP CLIENT/SERVER ARCHITECTURE

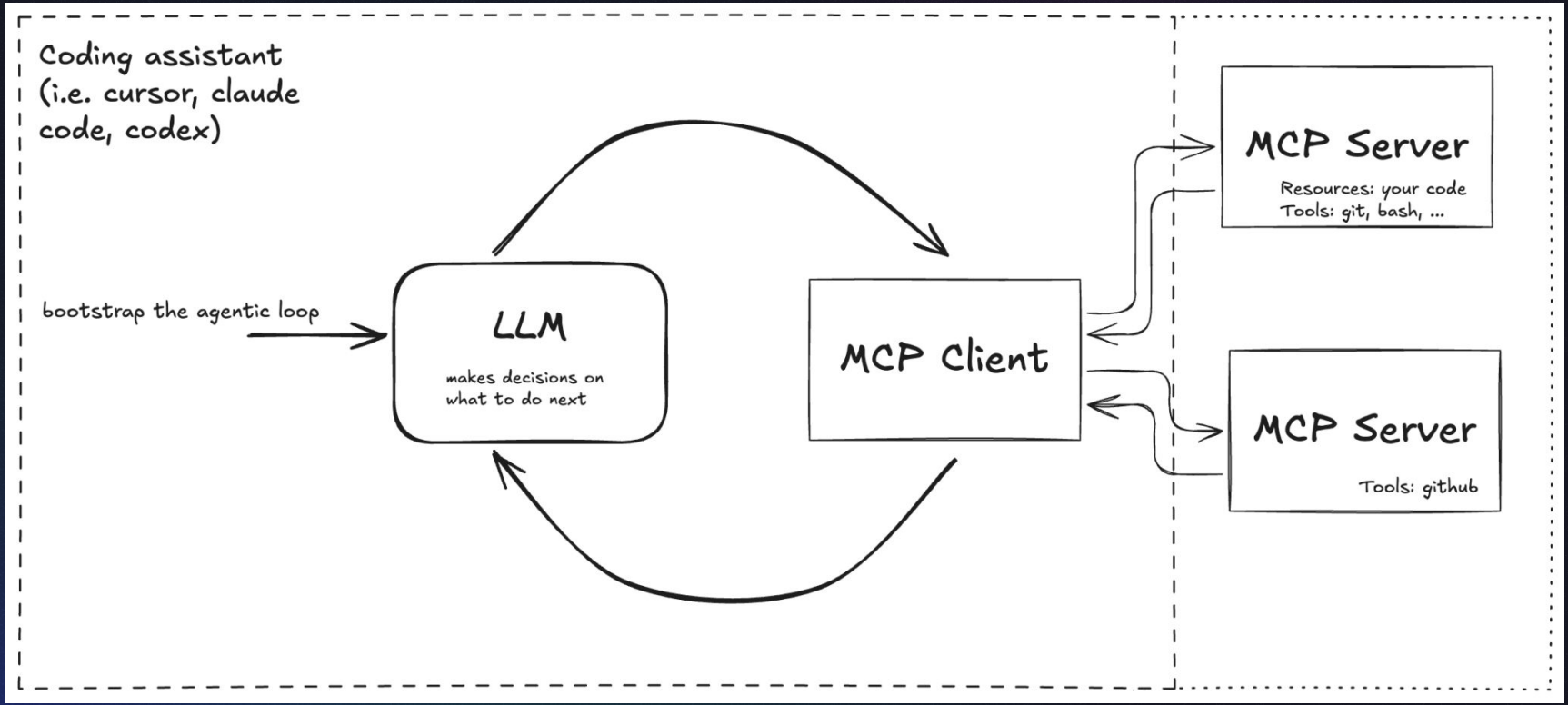


* Note: there are additional operations including notifications (one way), and sampling (Request/Response in the other direction).

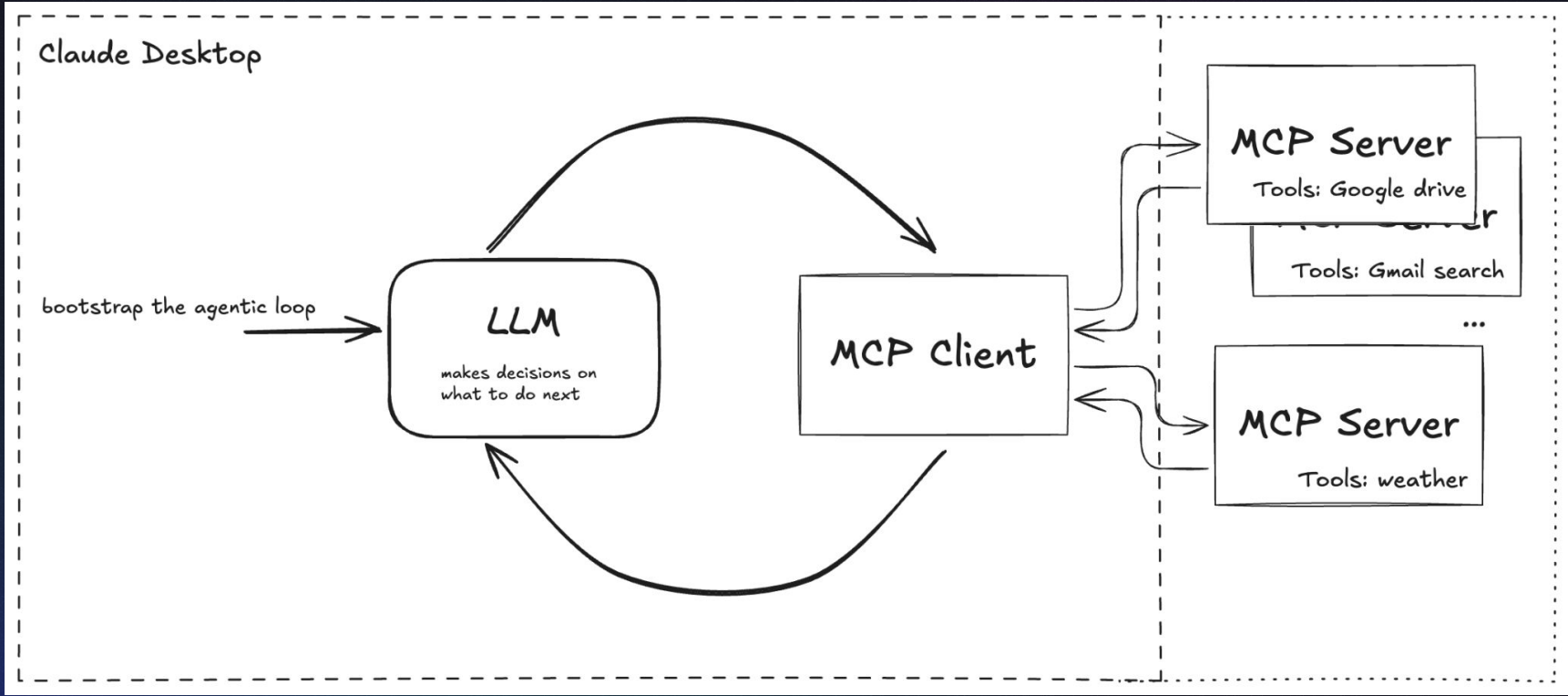
MCP CLIENTS ARE EMBEDDED IN THE AGENT



MANY APPLICATIONS EMBED MCP CLIENTS

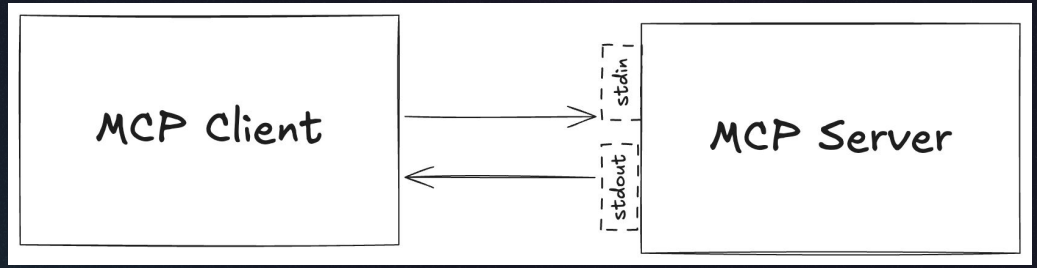


MANY APPLICATIONS EMBED MCP CLIENTS

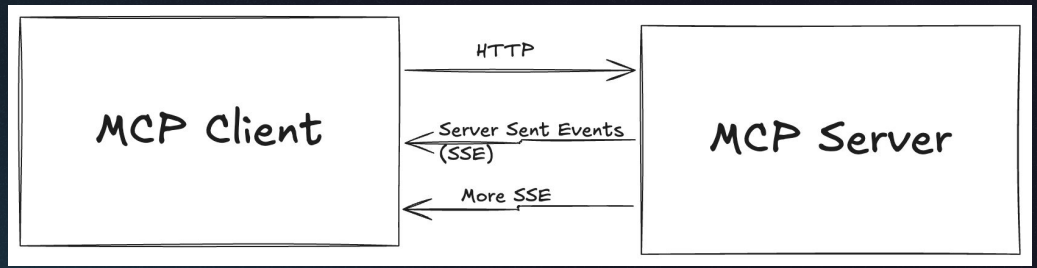


SUPPORTS MULTIPLE TRANSPORT PROTOCOLS

Transport: stdio



Transport:
streamable-http



MCP HAS TWO PARTS

Primitives

- Prompts
- Resources (i.e. files)
- Tools (i.e. Agent-ready APIs)

Client Server Architecture

- MCP Clients
- MCP Servers
- A transport protocol between them

* Note: there are a few other concepts, but these are the core ones

OKAY, THEN WHAT MAKES A GOOD MCP SERVER

“An API built for a human will poison your AI agent.”

- Jeremiah Lowin,
  Creator of FastMCP

https://www.jlowin.dev/blog/stop-converting-rest-apis-to-mcp

Commerce

Process orders, manage catalogs, track inventory, book reservations, and manage vendors.

Orders
Get sales data for a Square seller, itemize payments, push orders to POS, and more.

Order custom attributes
Create and manage order-related custom attribute definitions and custom attributes.

Catalog
Programmatically catalogs a Square seller's products for sale and services for hire.

Inventory
Programmatically manages a Square seller's inventory of catalog items.

Bookings
Create and manage bookings for Square sellers with the Bookings API.

Booking custom attributes
Create and manage booking-related custom attribute definitions and custom attributes.

Vendors
Manages a seller's suppliers.

Sites
Get details about Square Online sites that belong to Square sellers.

Snippets
Manage snippets for Square Online sites.

Cash drawers
Get details about cash drawer shifts.

Payments

Take payments, process refunds, manage disputes, enable subscriptions, and get paid for your sales.

Payments
The Payments API lets developers take and manage payments.

Refunds
Manage and issue refunds for payments made to Square sellers.

Checkout
Accept payments through a pre-built, Square-hosted checkout page. No frontend required.

Terminal
Requests a checkout from a paired Square Terminal.

Disputes
Use the Disputes API to manage disputes (chargebacks).

Invoices
Create, configure, and publish invoices for orders that were created using the Orders API.

Cards
Use the Cards API to save a credit or debit card on file.

Subscriptions
Create and manage subscriptions.

Bank accounts
Get a list of a seller's bank accounts.

Payouts
Get a list of deposits and withdrawals from a seller's bank accounts.

Devices
Create device codes used to connect Square Terminal with a 3rd-party point of sale system, and get details about all connected Terminals.

Apple pay
Apple Pay support APIs

Customers

Securely manage customer data and integrate engagement features to increase repeat business and attract new customers.

Customers
Create and manage customer profiles and sync CRM systems with Square.

Customer custom attributes
Create and manage customer-related custom attribute definitions and custom attributes.

Customer groups
Create and manage customer groups to streamline and automate workflows and help personalize customer interactions.

Customer segments
Retrieve customer segments (also called smart groups) in a business account.

Loyalty
Enroll buyers in a Square loyalty program, view program settings, manage and track loyalty activity, and create and manage promotions.

Gift cards
Create and retrieve gift cards and manage gift cards on file.

Gift card activities
Create and retrieve gift card activities.

Staff

Manage team members and jobs, track hours worked, and manage team schedules.

Labor
Manage timecards and schedules for team members.

Team
Manage a roster of team members and pull employee data into accounting and payroll systems.

Merchants

Manage a seller's core business information used for business, and suppliers.

Merchants
Retrieve information about an organization that sells with Square.

Merchant custom attributes
Create and manage merchant-related custom attribute definitions and custom attributes.

Locations
Create and manage the locations of a seller's business.

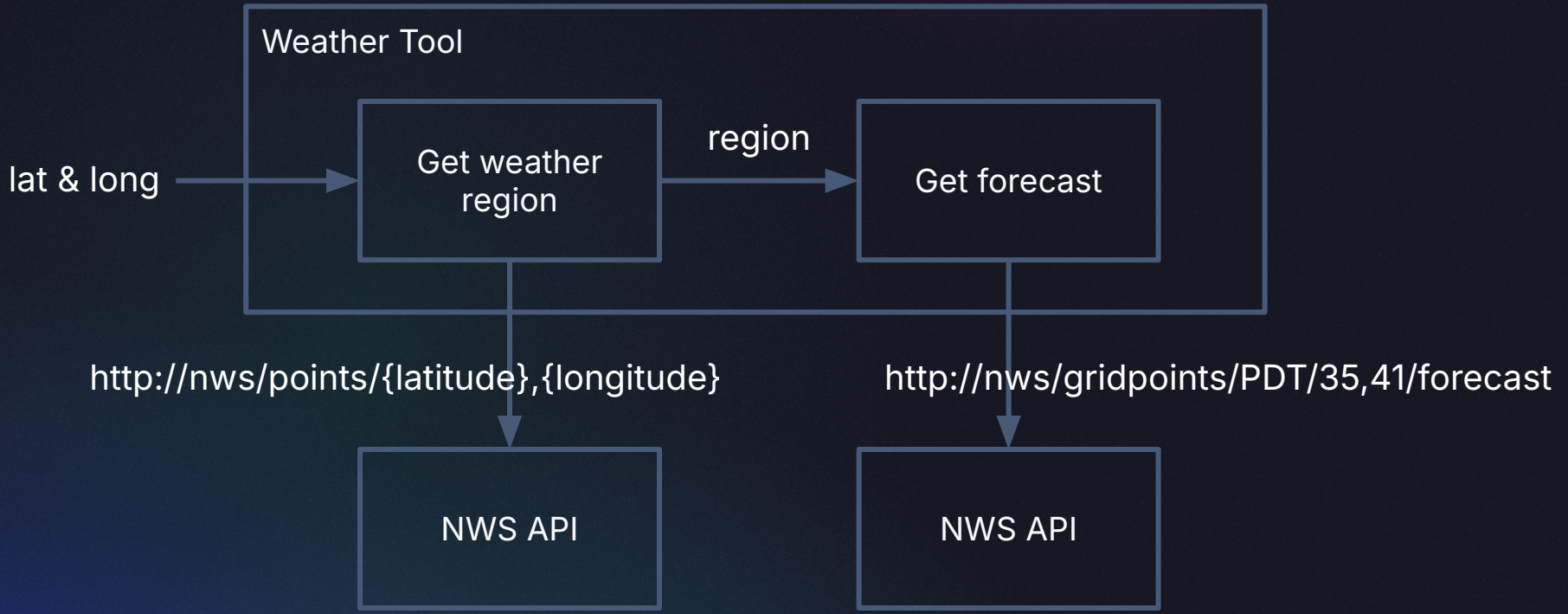
Location custom attributes
Create and manage location-related custom attribute definitions and custom attributes.

SQUARE APIS

WE NEED COARSER GRAINED APIS FOR LLMS

AND WE BUILD THOSE BY ORCHESTRATING
LOWER LEVEL APIS
INTO MCP SERVERS

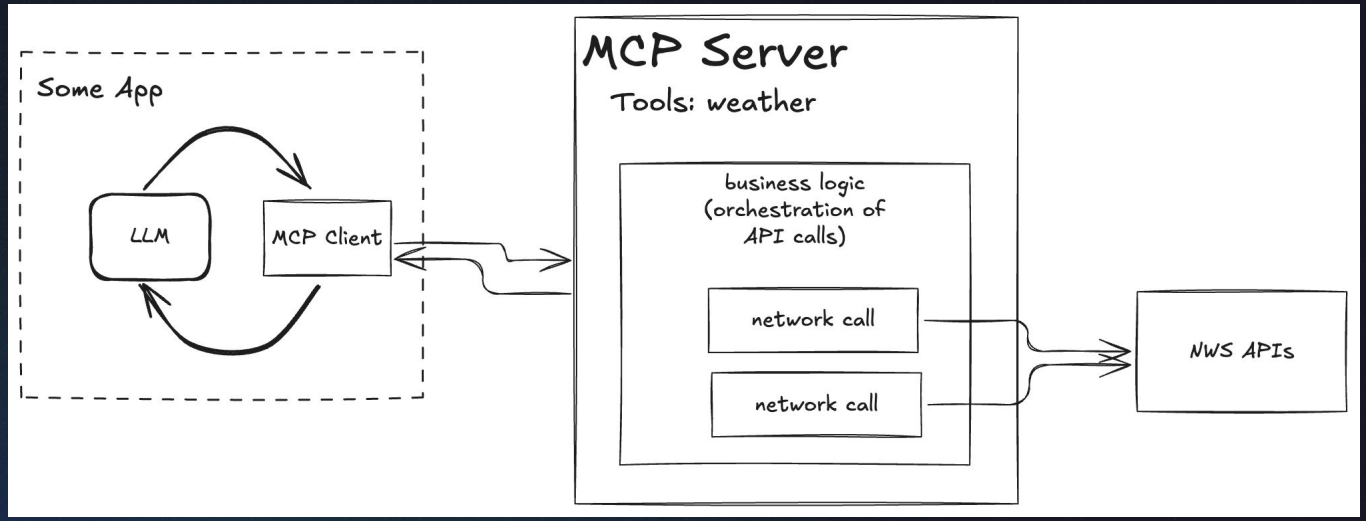
LET'S LOOK AT AN EXAMPLE



This is the example in the MCP spec

DEMO TIME!

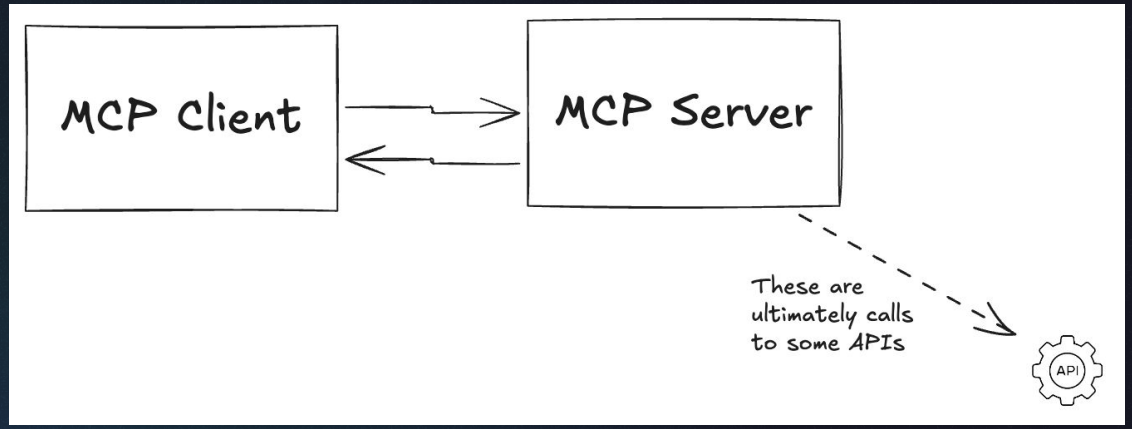
NON-DURABLE MCP ARCHITECTURE



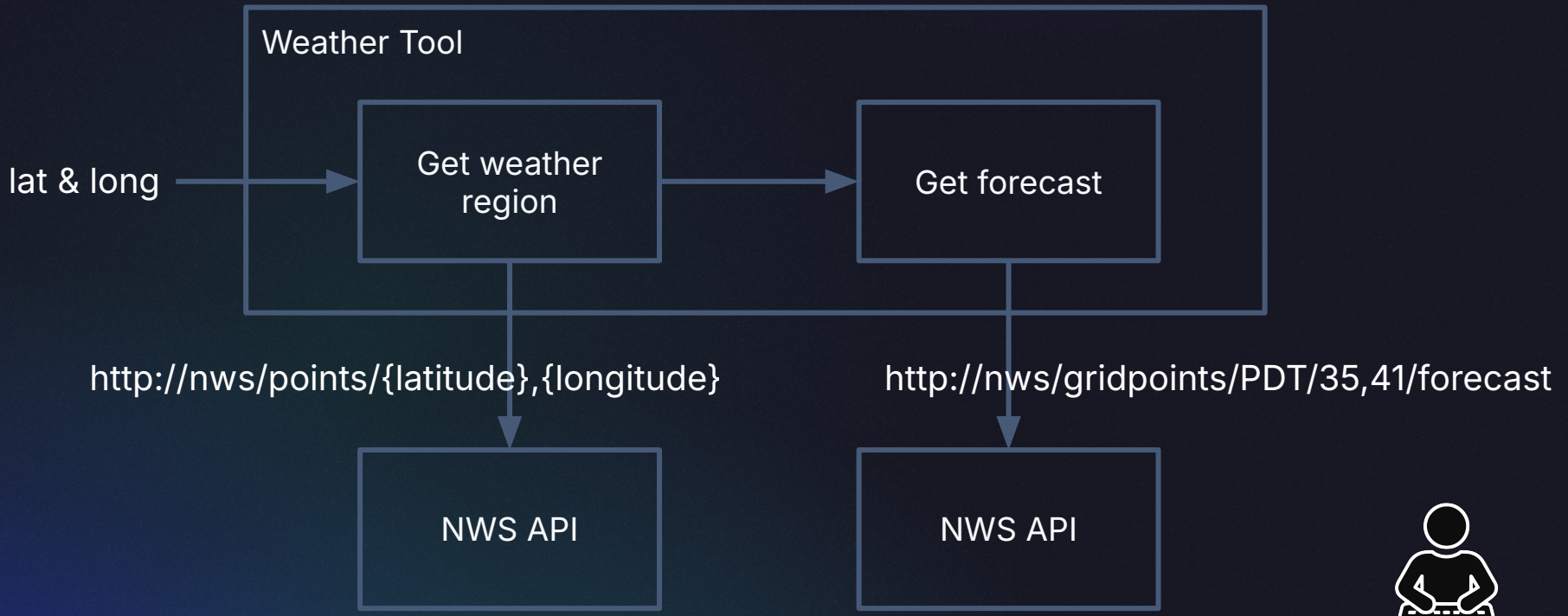
NOW, TEMPORAL

MCP SERVERS

Temporal is the ideal technology for building MCP Servers!



TEMPORAL IS THE BEST ORCHESTRATOR



NOTEBOOK SETUP

Follow along and play with our samples by accessing our Jupyter Notebooks here:

t.mp/temporal-mcp-workshop

Activity

```python
@activity.defn
async def make_nws_request(url: str) -> dict[str, Any] | None:
    """Make a request to the NWS API with proper error handling."""
    headers = {
        "User-Agent": USER_AGENT,
        "Accept": "application/geo+json"
    }
    async with httpx.AsyncClient() as client:
        try:
            response = await client.get(url, headers=headers, timeout=30.0)
            response.raise_for_status()
            return response.json()
        except Exception:
            return None
```

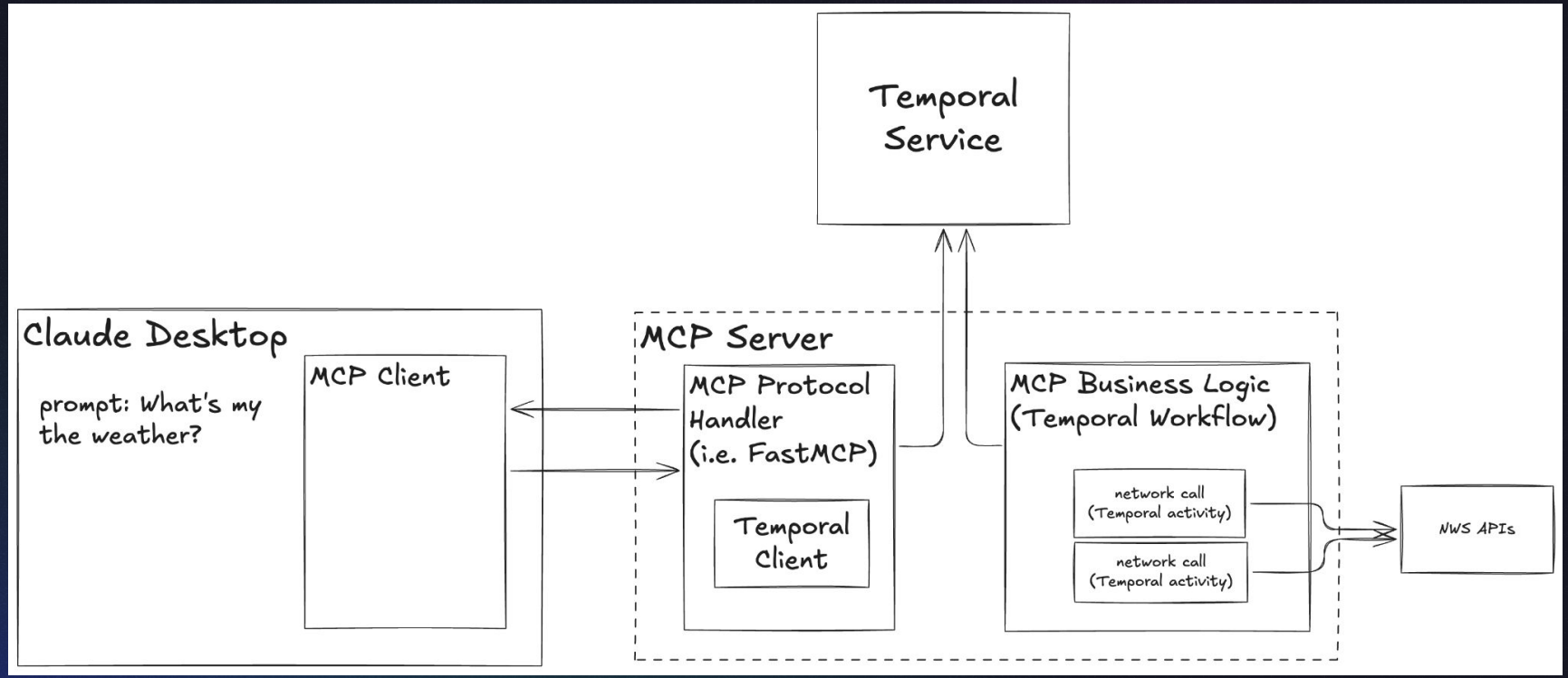
Workflow

```python
@workflow.defn
class GetForecast:
    @workflow.run
    async def get_forecast(self, latitude: float, longitude: float) -> str:

        # First get the forecast grid endpoint
        points_url = f"{NWS_API_BASE}/points/{latitude},{longitude}"
        points_data = await workflow.execute_activity(
            make_nws_request,
            points_url,
            schedule_to_close_timeout=timedelta(seconds=40),
            retry_policy=retry_policy,
        )

        await workflow.sleep(10)

        # Get the forecast URL from the points response
        forecast_url = points_data["properties"]["forecast"]
        forecast_data = await workflow.execute_activity(
            make_nws_request,
            forecast_url,
            schedule_to_close_timeout=timedelta(seconds=40),
            retry_policy=retry_policy,
        )
```

MCP Protocol Handler and Workflow Client

```python
@mcp.tool
async def get_forecast(latitude: float, longitude: float) -> str:
    """Get weather forecast for a location.

    Args:
        latitude: Latitude of the location
        longitude: Longitude of the location
    """
    # The business logic has been moved into the temporal workflow, the mcp tool kicks off the workflow
    client = await get_temporal_client()
    handle = await client.start_workflow(
        workflow="GetForecast",
        args=[latitude, longitude],
        id=f"forecast-{latitude}-{longitude}",
        task_queue="weather-task-queue",
    )
    return await handle.result()
```

The Durable Architecture



DEMO TIME!

WHEN ERROR HANDLING IS A BUG

Activity (corrected)

```python
@activity.defn
async def make_nws_request(url: str) -> dict[str, Any] | None:
    """Make a request to the NWS API with proper error handling."""
    headers = {
        "User-Agent": USER_AGENT,
        "Accept": "application/geo+json"
    }
    async with httpx.AsyncClient() as client:

        response = await client.get(url, headers=headers, timeout=20.0)
        response.raise_for_status()
        return response.json()
```

Summarizing

- MCP is the defacto standard
- What we are seeing in our customer base → you are building MCP servers.
  - Platform Engineering orchestration from slack
  - Extensions to your coding agents
  - ...
- Still early days - an abundance of trickiness (security, scaling, observability, ..)

But some things are clear...

- MCP servers are distributed systems
- They need durability
- The durability needs compound because... Agents!
- Temporal is committed!

Challenge	Without Temporal	With Temporal
Failures	Tool execution lost on crash	Automatic recovery and retry
Long operations	Timeout limitations	Unlimited execution time
State Management	Complex manual handling	Built-in durable state
Network Issues	Manual retry logic needed	Automatic retry with backoff
Multi-step Processes	Difficult coordination	Natural workflow orchestration

LET'S CODE AND BUILD SOME DURABLE MCP TOOLS.

Commerce

Process orders, manage catalogs, track inventory, book reservations, and manage vendors.

- **Orders** — Get sales data for a Square seller, itemize payments, push orders to POS, and more.
- **Order custom attributes** — Create and manage order-related custom attribute definitions and custom attributes.
- **Catalog** — Programmatically catalogs a Square seller's products for sale and services for hire.
- **Inventory** — Programmatically manages a Square seller's inventory of catalog items.
- **Bookings** — Create and manage bookings for Square sellers with the Bookings API.
- **Booking custom attributes** — Create and manage booking-related custom attribute definitions and custom attributes.
- **Vendors** — Manages a seller's suppliers.
- **Sites** — Get details about Square Online sites that belong to Square sellers.
- **Snippets** — Manage snippets for Square Online sites.
- **Cash drawers** — Get details about cash drawer shifts.

Payments

Take payments, process refunds, manage disputes, enable subscriptions, and get paid for your sales.

- **Payments** — The Payments API lets developers take and manage payments.
- **Refunds** — Manage and issue refunds for payments made to Square sellers.
- **Checkout** — Accept payments through a pre-built, Square-hosted checkout page. No frontend required.
- **Terminal** — Requests a checkout from a paired Square Terminal.
- **Disputes** — Use the Disputes API to manage disputes (chargebacks).
- **Invoices** — Create, configure, and publish invoices for orders that were created using the Orders API.
- **Cards** — Use the Cards API to save a credit or debit card on file.
- **Subscriptions** — Create and manage subscriptions.
- **Bank accounts** — Get a list of a seller's bank accounts.
- **Payouts** — Get a list of deposits and withdrawals from a seller's bank accounts.
- **Devices** — Create device codes used to connect Square Terminal with a 3rd-party point of sale system, and get details about all connected Terminals.
- **Apple pay** — Apple Pay support APIs

Customers

Securely manage customer data and integrate engagement features to increase repeat business and attract new customers.

- **Customers** — Create and manage customer profiles and sync CRM systems with Square.
- **Customer custom attributes** — Create and manage customer-related custom attribute definitions and custom attributes.
- **Customer groups** — Create and manage customer groups to streamline and automate workflows and help personalize customer interactions.
- **Customer segments** — Retrieve customer segments (also called smart groups) in a business account.
- **Loyalty** — Enroll buyers in a Square loyalty program, view program settings, manage and track loyalty activity, and create and manage promotions.
- **Gift cards** — Create and retrieve gift cards and manage gift cards on file.
- **Gift card activities** — Create and retrieve gift card activities.

Staff

Manage team members and jobs, track hours worked, and manage team schedules.

- **Labor** — Manage timecards and schedules for team members.
- **Team** — Manage a roster of team members and pull employee data into accounting and payroll systems.

Merchants

Manage a seller's core business information used for business, and suppliers.

- **Merchants** — Retrieve information about an organization that sells with Square.
- **Merchant custom attributes** — Create and manage merchant-related custom attribute definitions and custom attributes.
- **Locations** — Create and manage the locations of a seller's business.
- **Location custom attributes** — Create and manage location-related custom attribute definitions and custom attributes.

SQUARE APIS

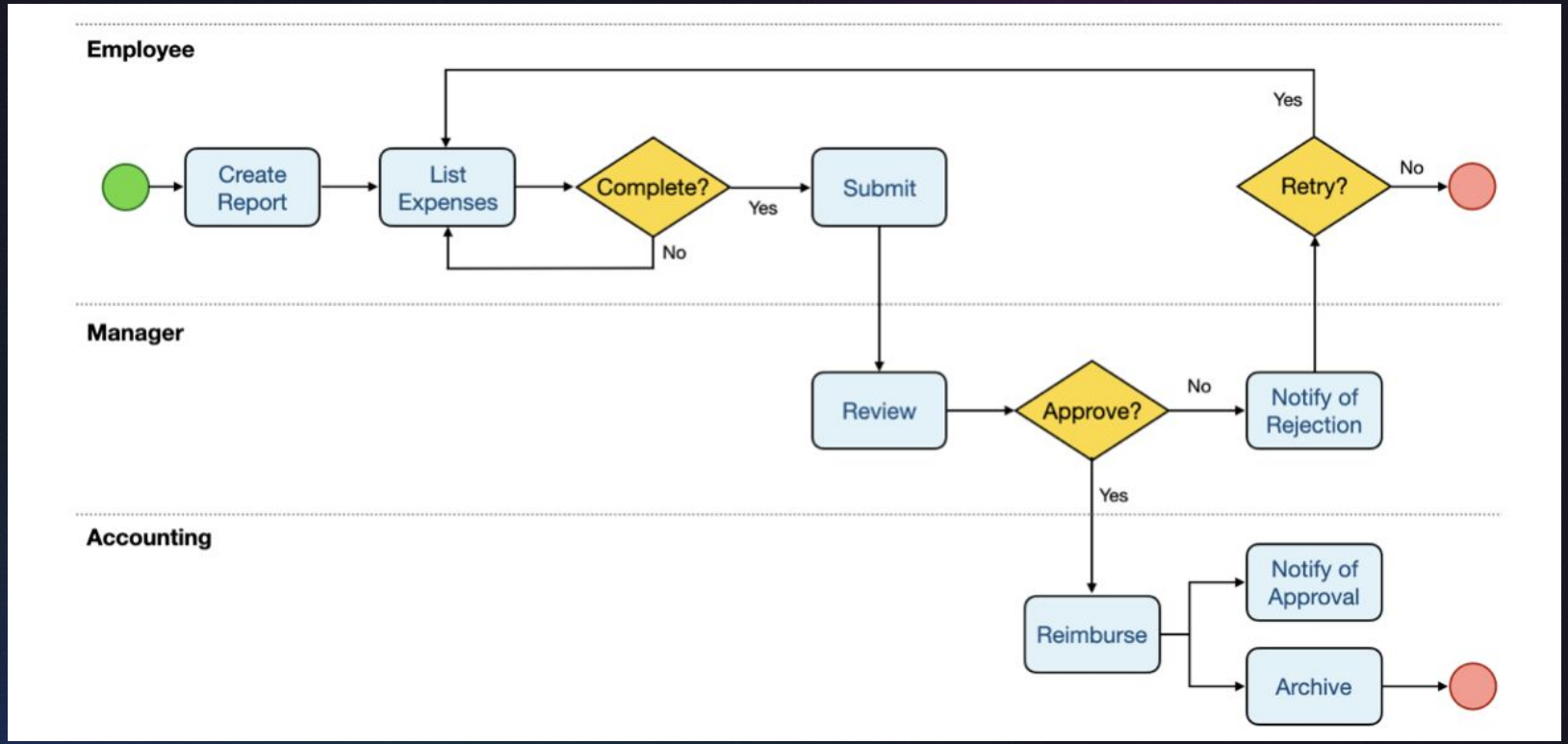
Temporal

Long-running MCP tools and HITL

Temporal

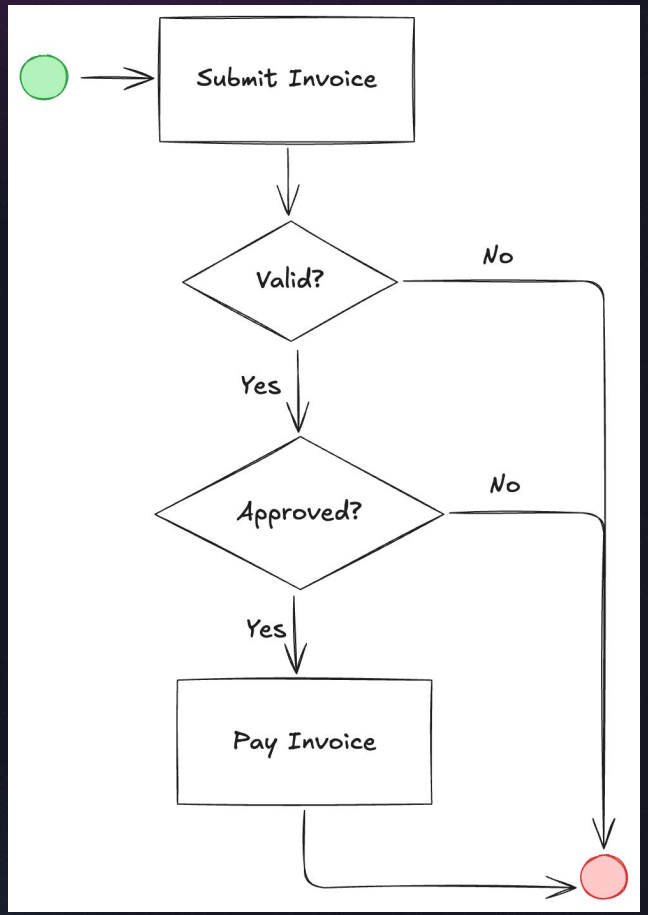
Processes

Long-running ~~MCP tools~~ and HITL


Employee
Create Report
List Expenses
Complete?
Yes
No
Submit
Retry?
Yes
No
Manager
Review
Approve?
No
Notify of Rejection
Yes
Accounting
Reimburse
Notify of Approval
Archive

FOR TODAY: INVOICE PROCESSING

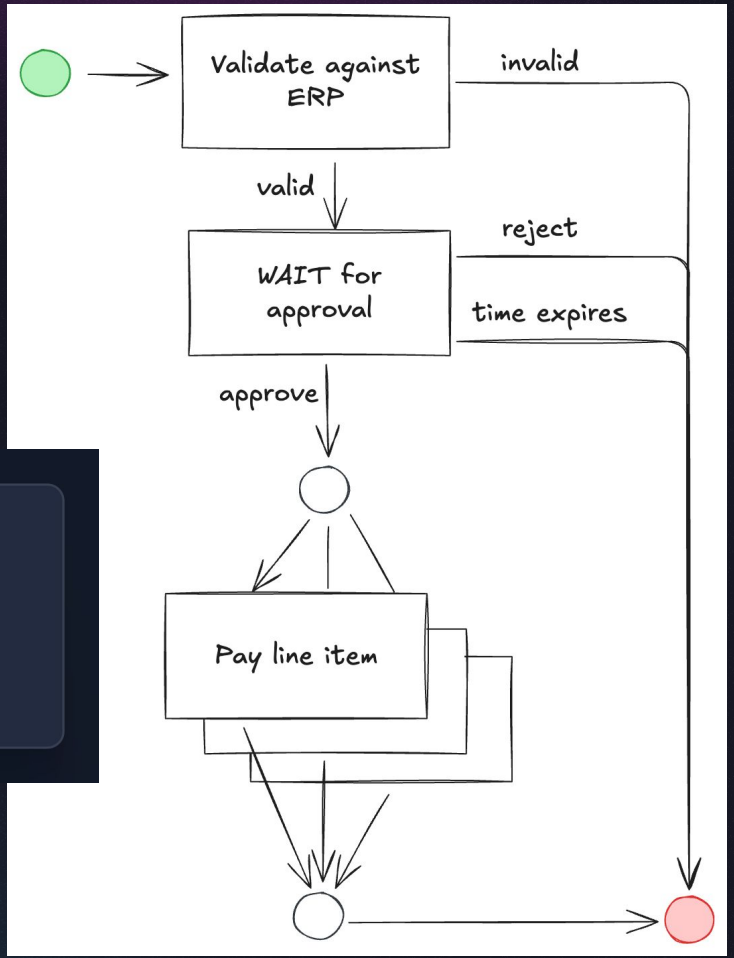
- Submit invoice
- Apply policies to determine whether an invoice can be auto approved (i.e. <$100)
- Allow for approval (HITL)
- When approved, process the invoice (i.e. issue a payment)
- Implement policies for invoices that go unapproved



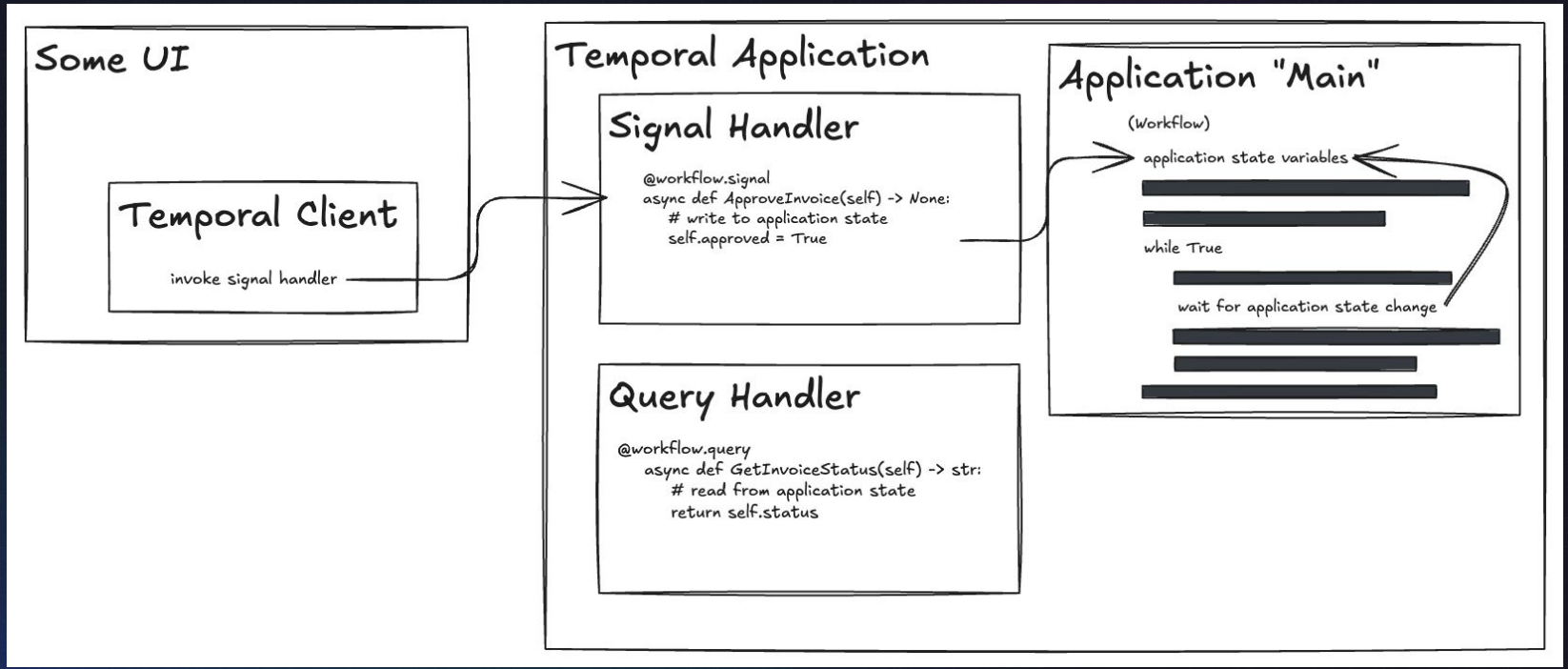
THE WORKFLOW

Durable Timers!!

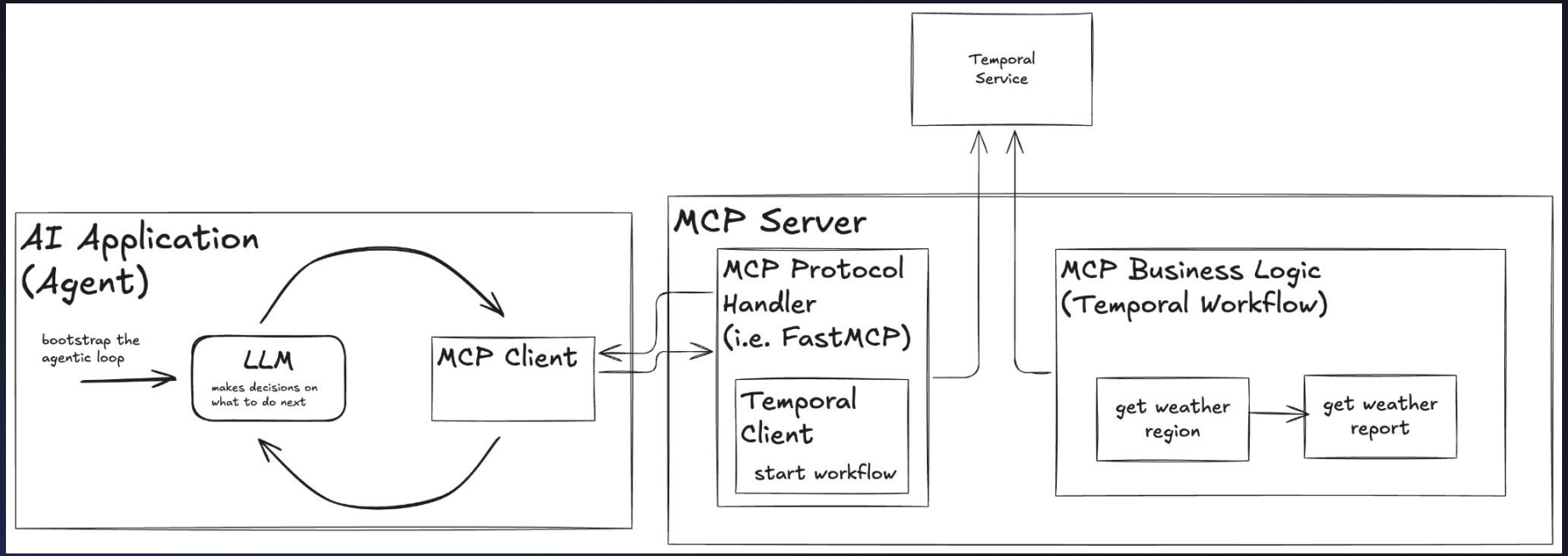
```python
await workflow.wait_condition(
    lambda: self.approved is not None,
    timeout=timedelta(days=5),
)
```



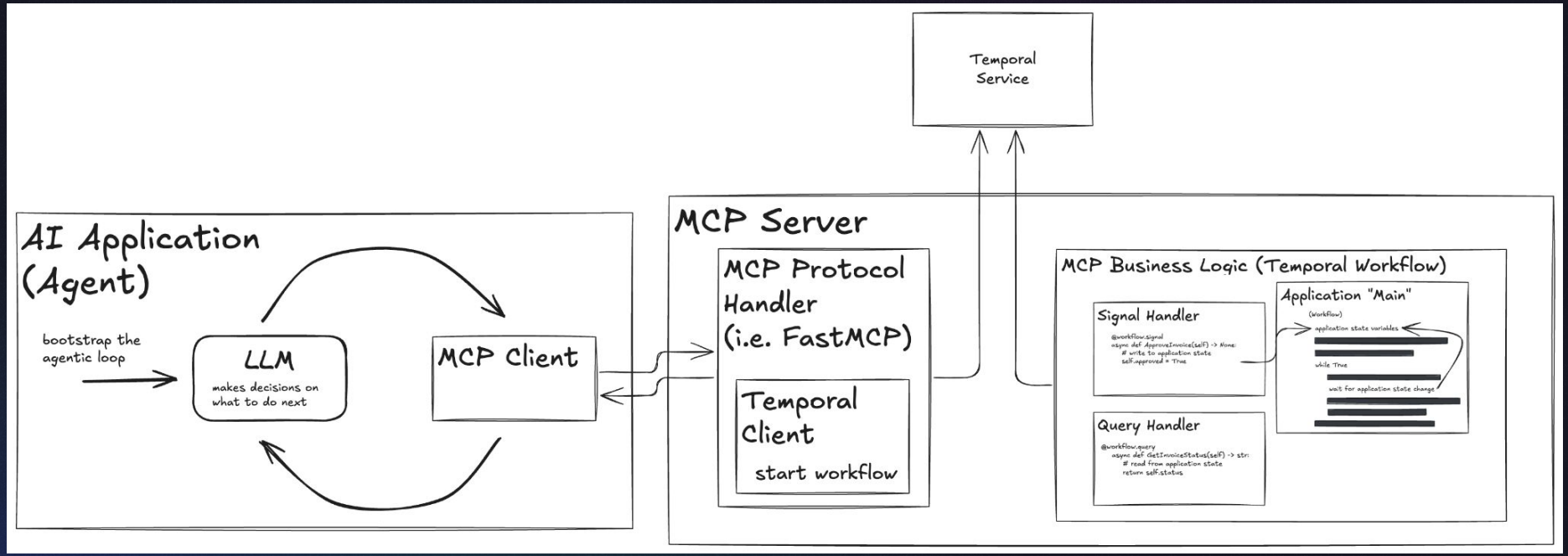
WELL SUITED TO TEMPORAL!



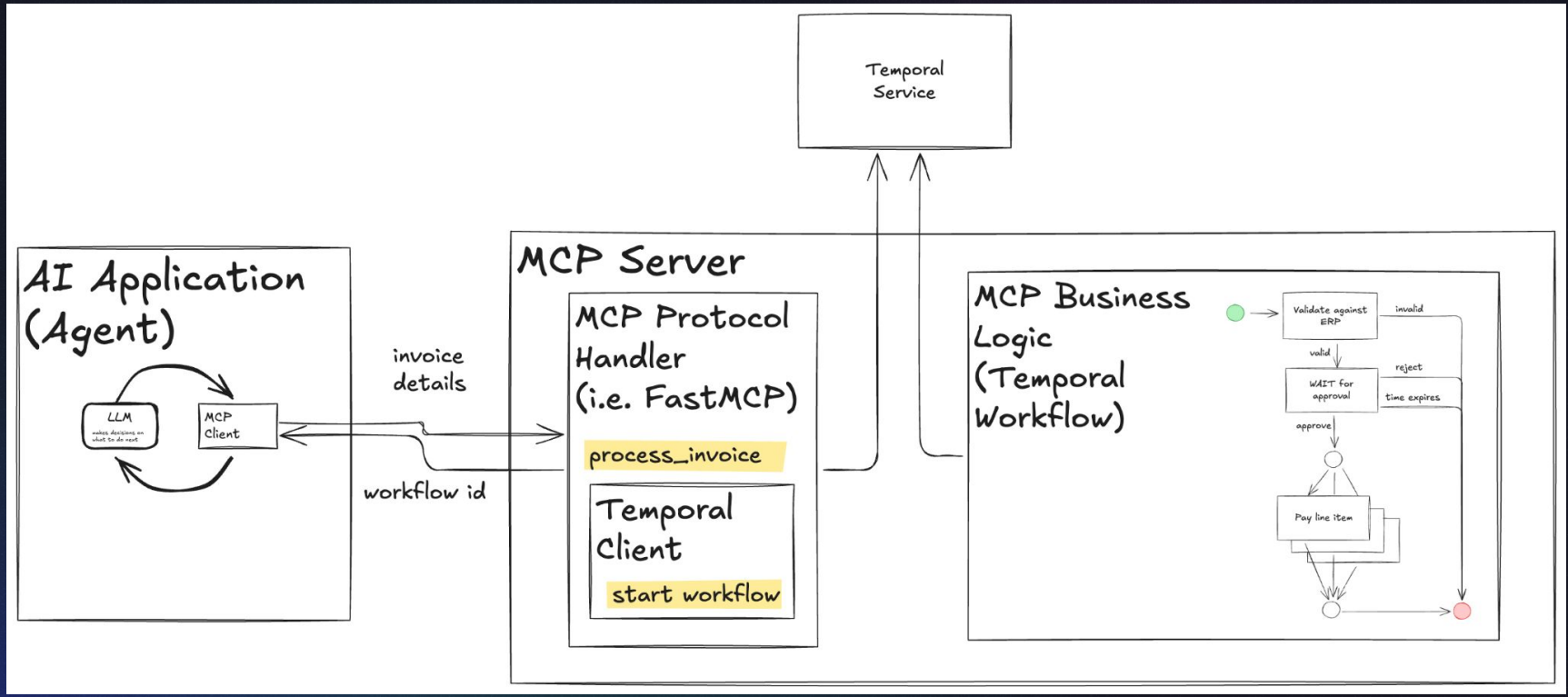
REMEMBER THIS? - FROM OUR FIRST EXAMPLE



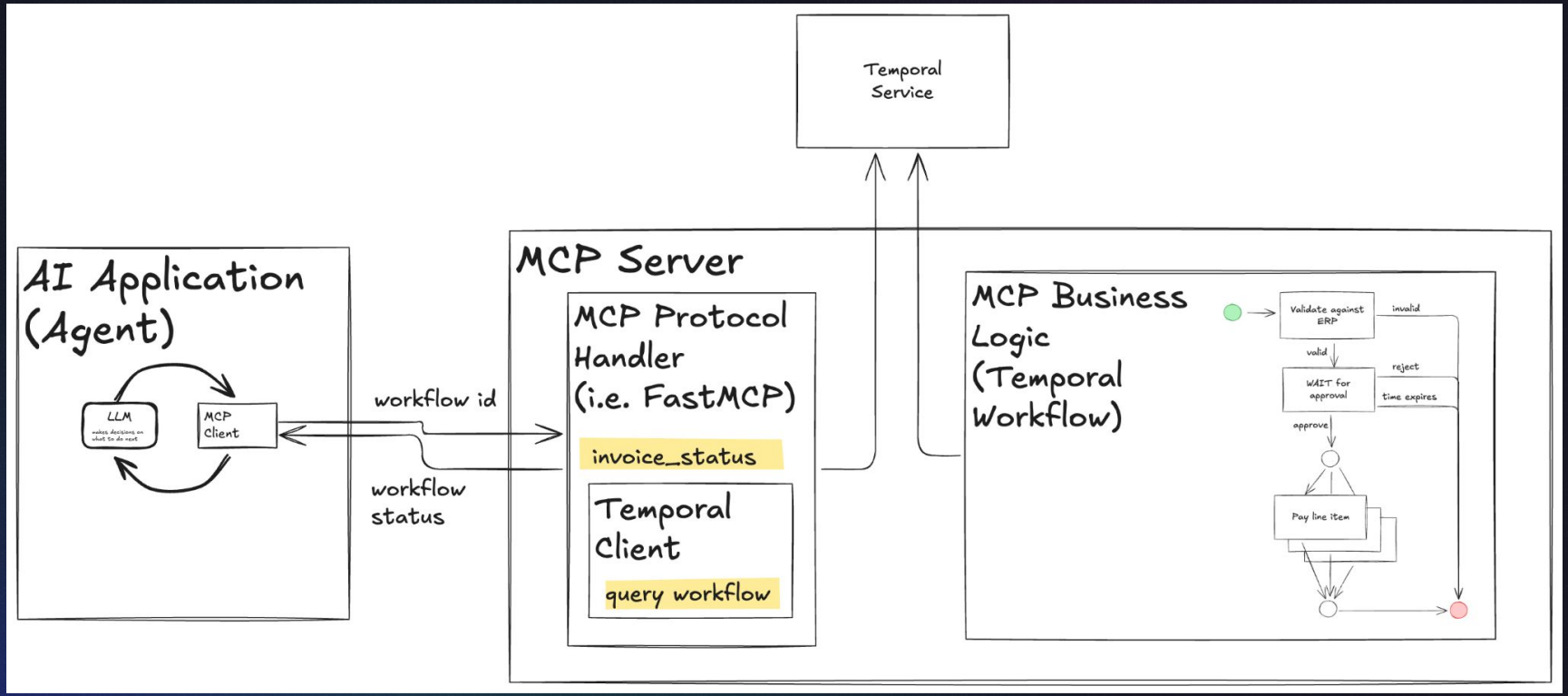
FRONT WITH THE MCP PROTOCOL



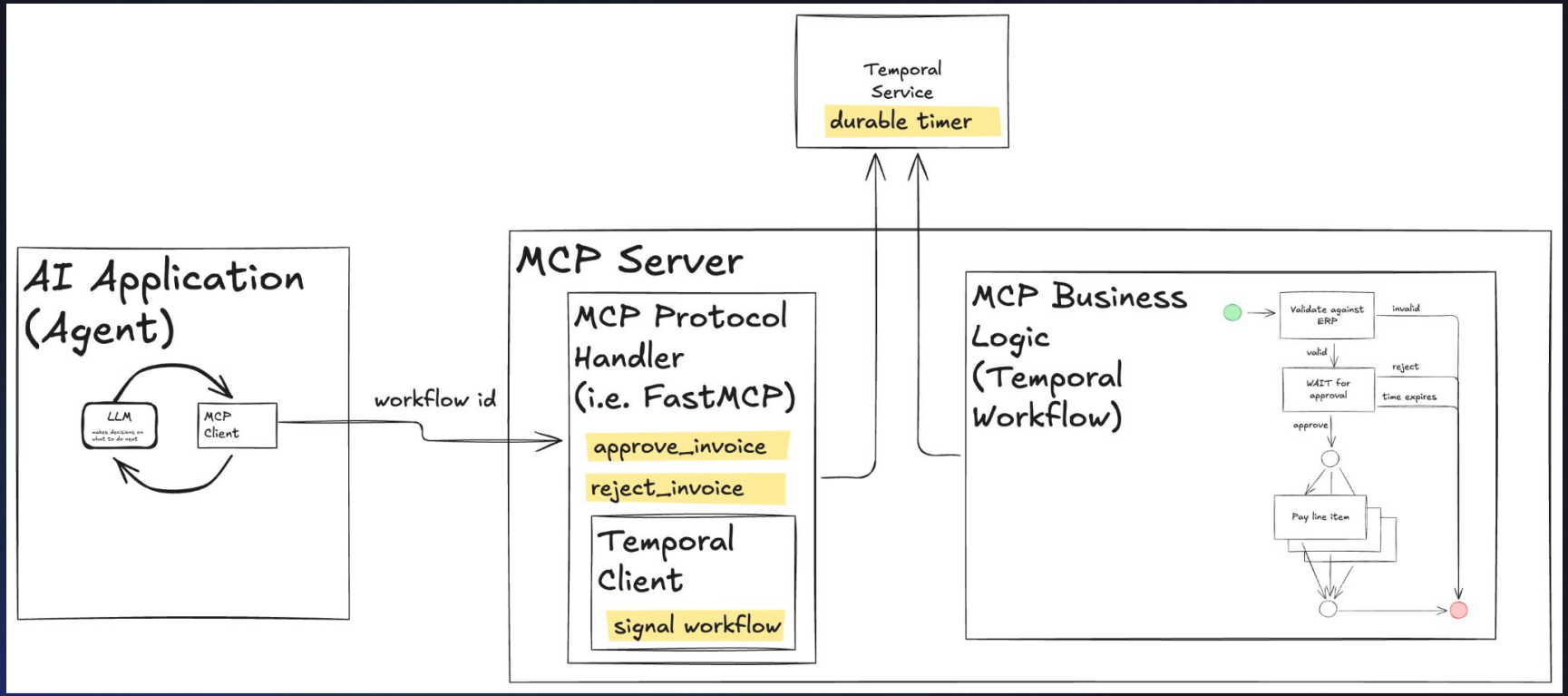
SUBMIT INVOICE



INVOICE STATUS



APPROVE/REJECT INVOICE (AND TIMERS)



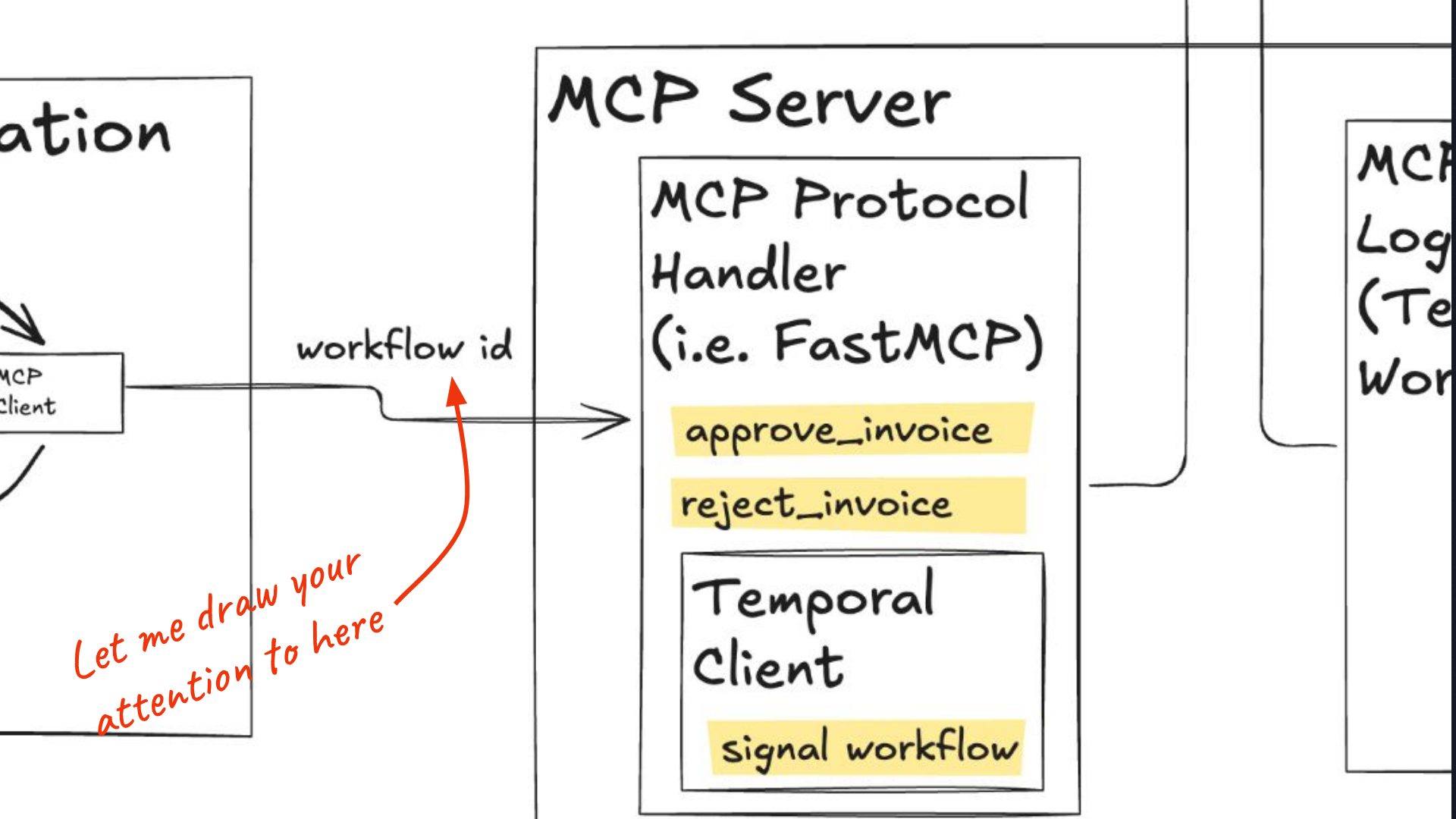
MCP Protocol Layer

```python
@mcp.tool()
async def approve_invoice(workflow_id: str, run_id: str) -> str:
    """Signal approval for the invoice workflow."""
    client = await _client()
    ...

@mcp.tool()
async def reject_invoice(workflow_id: str, run_id: str) -> str:
    """Signal rejection for the invoice workflow."""
    client = await _client()
    ...

@mcp.tool()
async def invoice_status(workflow_id: str, run_id: str) -> str:
    """Return current status of the workflow."""
    client = await _client()
    ...

@mcp.tool()
async def process_invoice(invoice: Dict) -> Dict[str, str]:
    """Start the InvoiceWorkflow with the given invoice JSON."""
    client = await _client()
    ...
```

MCP Business Logic

```python
@workflow.defn
class InvoiceWorkflow:
    def __init__(self) -> None:
        self.approved: bool | None = None
        self.status: str = "INITIALIZING"

    @workflow.signal
    async def ApproveInvoice(self) -> None:
        self.approved = True

    @workflow.signal
    async def RejectInvoice(self) -> None:
        self.approved = False

    @workflow.query
    async def GetInvoiceStatus(self) -> str:
        return self.status

    @workflow.run
    async def run(self, invoice: dict) -> str:
      ...
```

DEMO TIME!


ation
MCP
Client
workflow id
Let me draw your attention to here
MCP Server
MCP Protocol Handler (i.e. FastMCP)
approve_invoice
reject_invoice
Temporal Client
signal workflow
MCP
Log
(Te
Wor

LET'S TALK ABOUT DIGITAL TWINS

*“A **digital twin** is a digital model of an intended or actual real-world physical product, system, or process (a physical twin) that serves as a digital counterpart of it for purposes such as simulation, integration, testing, monitoring, and maintenance.”*

https://en.wikipedia.org/wiki/Digital_twin

INVOICE = LONG RUNNING WORKFLOW

Physical twin

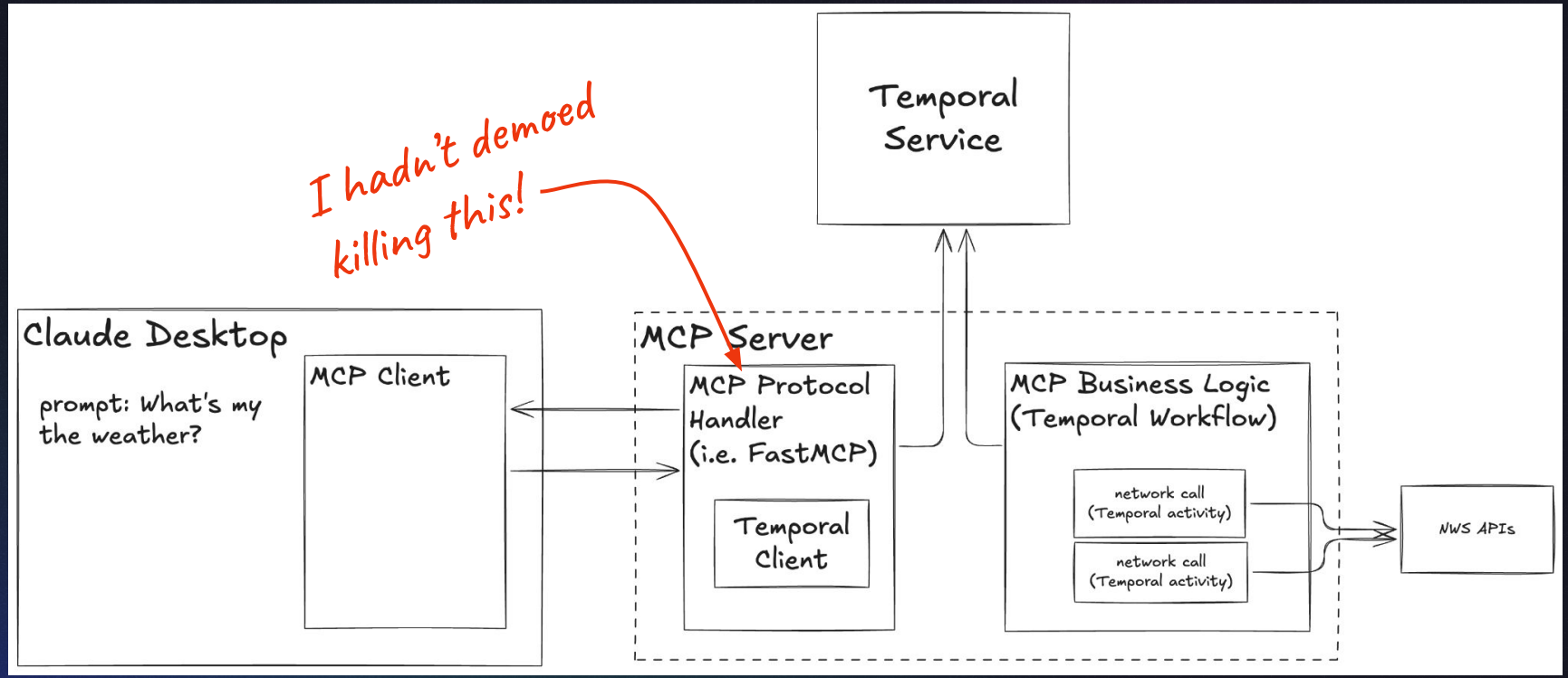
Digital twin

We also refer to these as
Entity Workflows

DEMO TIME!

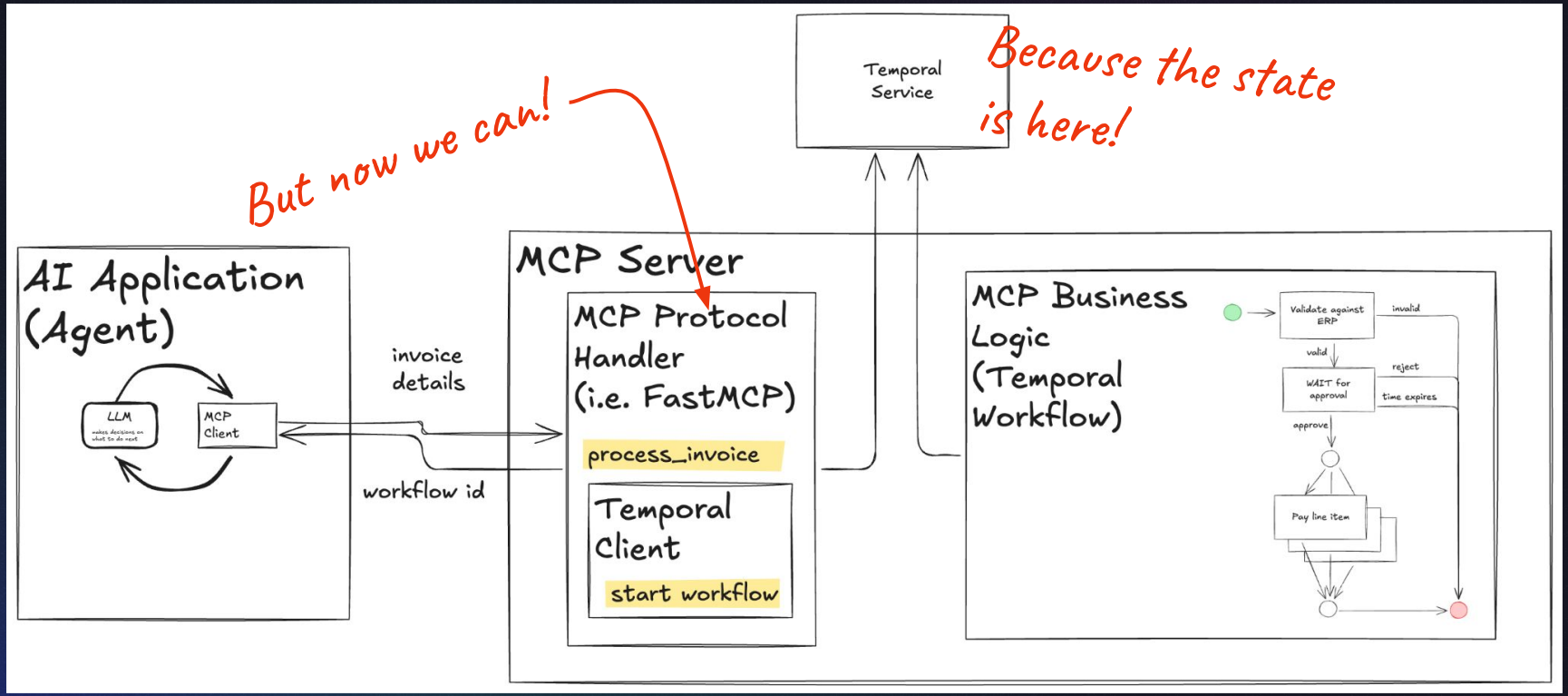
ONE MORE THING TO NOTE...

THE DURABLE ARCHITECTURE



THE CONNECTION BETWEEN THE MCP CLIENT AND THE MCP SERVER IS STATEFUL

THE DURABLE ARCHITECTURE



THIS IS A GAP IN THE MCP PROTOCOL

[notes] Long running tools/ async tools/ resumability #982

THAT IS ADDRESSED - TODAY - IF YOU IMPLEMENT YOUR MCP SERVERS USING TEMPORAL

DEMO TIME!

IF YOU DON'T...

We're working on it with some of you.

LEARNING OBJECTIVES

- ✅ Explore the role that MCP plays in AI applications
- ✅ Review MCP fundamentals
- ✅ Build durability into your MCP tools with Temporal
- ✅ Debug and monitor MCP tools with Temporal Web UI
- ✅ Implement long running MCP tools with Temporal (queries, signals)

WE WELCOME YOUR FEEDBACK!

T.MP/AI-WORKSHOP-FEEDBACK

WANT TO WATCH MORE ABOUT TEMPORAL AND MCP?

GET STARTED WITH TEMPORAL!

- Join the community!
- Read the documentation
- Follow a tutorial
- Take a free course online and more

Replay 2026

Moscone Center, SF

May 5-7, 2026

1500+ attendees

Use Code **LAUNCHANDLEARN75** for 75% off

// JUST ANNOUNCED

01 TEMPORAL

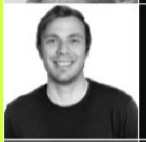
SAMAR ABBAS

02 TEMPORAL

MAXIM FATEEV

03 REPLIT

AMJAD MASAD



04 NVIDIA

JEREMY COOK

05 NVIDIA

MATT TESCHER

06 FIVETRAN

ZAINAB MERCHANT

07 PYDANTIC

SAMUEL COLVIN

REPLAY

MAY 05-07, 2026

MOSCONE CENTER, SAN FRANCISCO

Temporal