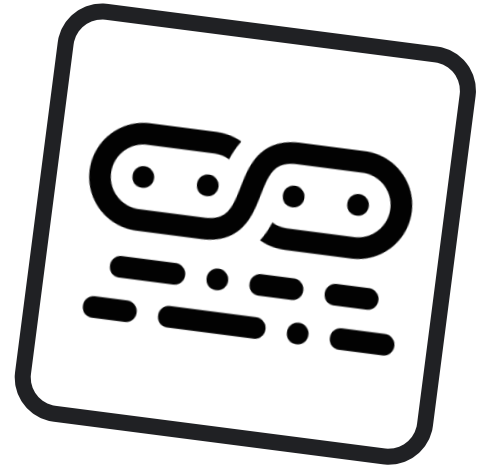


A2A Overview

Building collaborative agentic systems





Holt Skinner

Developer Advocate

Google Cloud AI ☁️

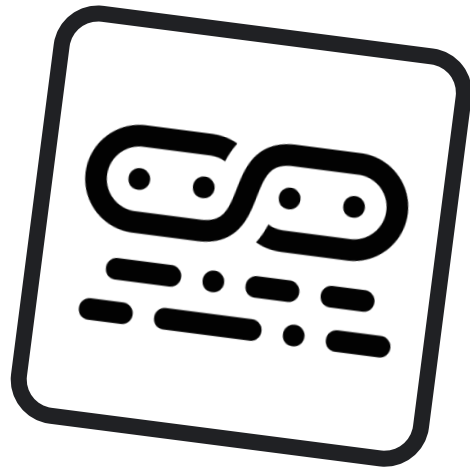
Austin, TX 🤠

Also a classical singer 🎵



Agenda

- Speaker Introductions
- Google - Intro to A2A Protocol
- MindsDB Agents Stack and Demo
- Q&A



How do you work?

How prompts work	How humans work
In one huge step?	Step-by-step
Perfectly on the first attempt?	Iteratively, diverge then converge
In a vacuum?	With others
With no eraser?	With a scratchpad
With no memory?	With past context
With no Google?	With all of the external knowledge

What is an Agent?



Agent

[ˈā-jənt]

A person legally empowered to act on behalf of another person or an entity.

The term *agent* is derived from the Latin *agere* (to do): an agreement to act on one's behalf.

What is an Agent?



Agent

['ā-jənt]

A person legally empowered to act on behalf of another person or an entity.



What is an AI Agent?

An AI agent is an **application** that uses a combination of a generative **model**'s capabilities, **tools** to connect to the external world, and high-level **reasoning** to achieve a desired end goal or state

Model



Tools



Reasoning



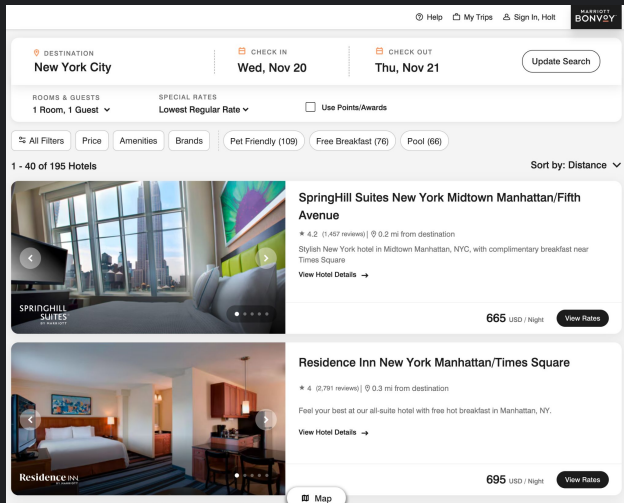
Deployment



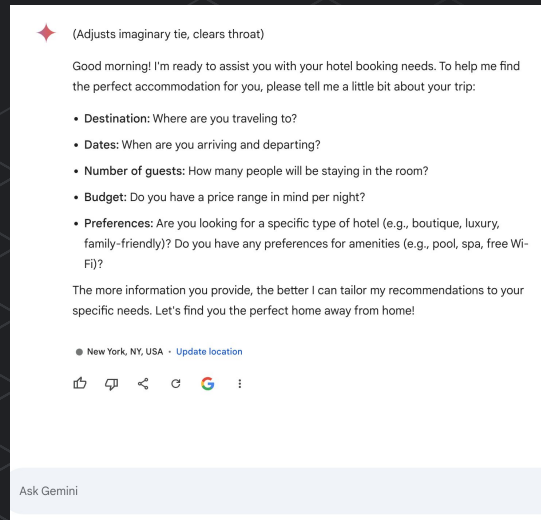
Types of Agents



Human Agent

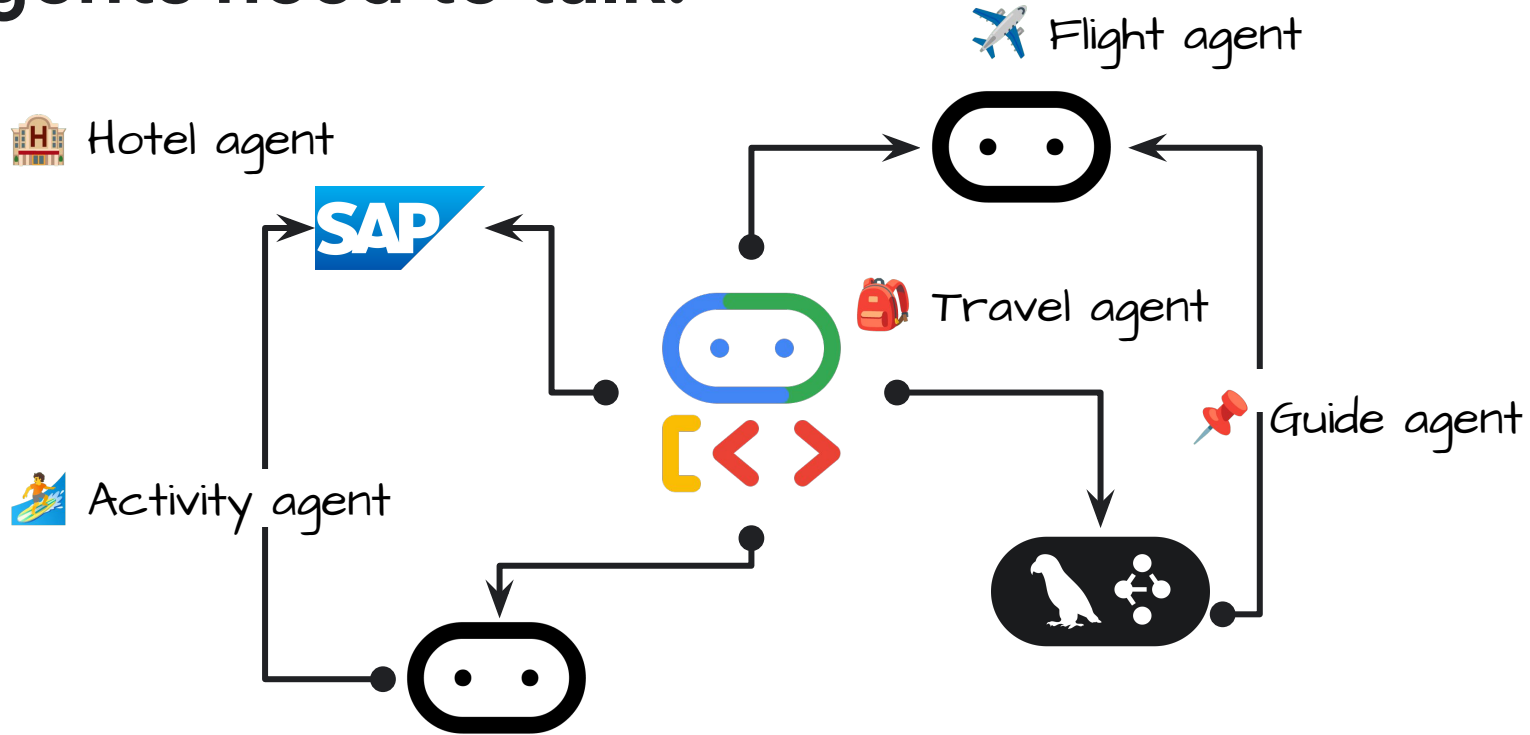


Software Agent

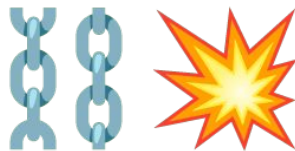


AI Agent
(Software Agent + LLM for communication)

Agents need to talk!



Agent challenges



Fragmented landscape

How do we achieve faster development and iteration?

Hard to integrate

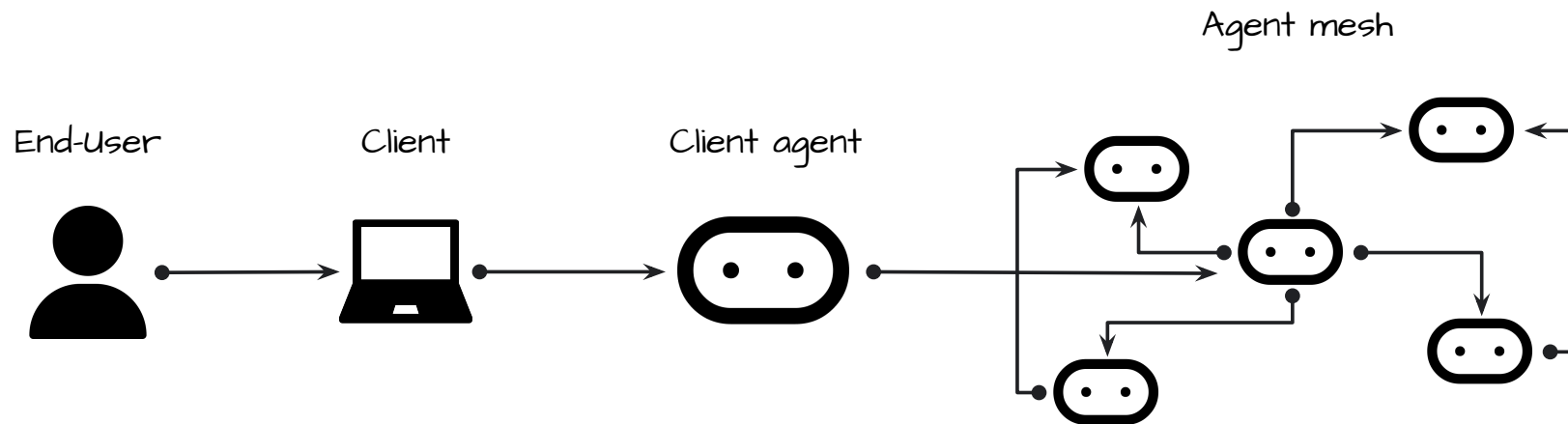
How do we manage dependencies, delegate effectively, and avoid creating hard-to-debug workflows between agents?

Lack of ops & governance

Beyond development, how do we reliably deploy, scale, monitor, and govern an ecosystem of agents?

Agent2Agent (A2A) protocol

Open protocol to handle agent collaboration



Partners contributing to the Agent2Agent protocol



A2A capabilities



Discovery

Agents must advertise their capabilities so clients know when and how to utilize them for specific tasks.



Negotiation

Clients and agents need to agree on communication methods like text, forms, iframe, or audio/video to ensure proper user interaction.



Task and State Management

Clients and agents need mechanisms to communicate task status, changes, and dependencies throughout task execution.



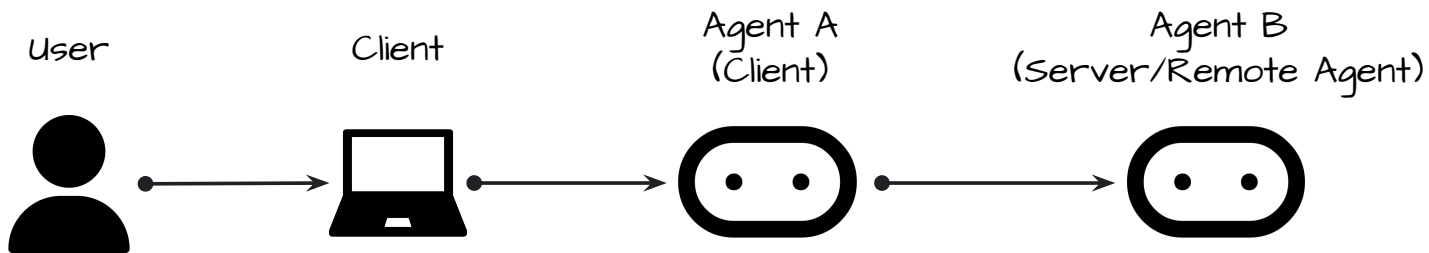
Collaboration

Clients and agents must support dynamic interaction, enabling agents to request clarifications, information, or sub-actions from client, other agents, or users.



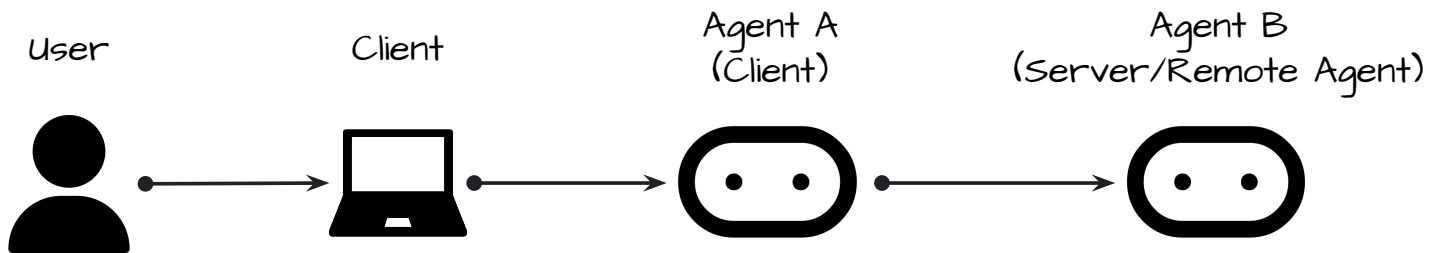
How A2A works

Building a simple agent system



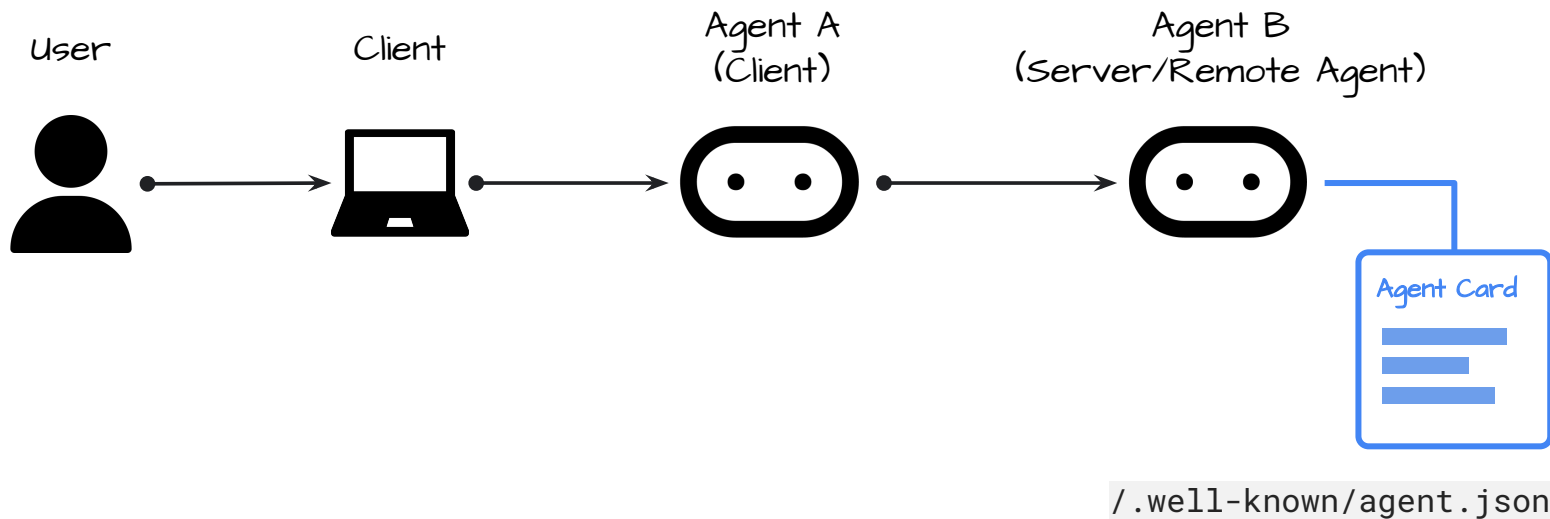
Step 1: Agent Discovery

Who are you & what can you do?



Step 1: Agent Discovery

The agent card



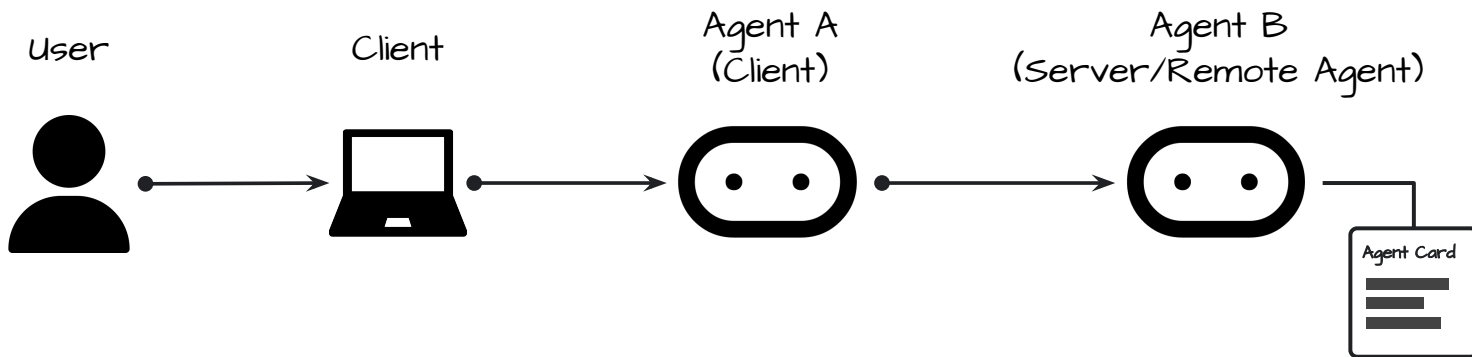
Agent Card

Example

```
agent_card = AgentCard(  
    name='Currency Agent',  
    description='Helps with exchange rates for currencies',  
    url=f'http://{host}:{port}/', # e.g., http://localhost:10000/  
    version='1.0.0',  
    default_input_modes=CurrencyAgent.SUPPORTED_CONTENT_TYPES, # Usually ['text/plain']  
    default_output_modes=CurrencyAgent.SUPPORTED_CONTENT_TYPES,  
    skills=[skill],  
)  
  
# ... (Server setup using this agent_card) ...
```

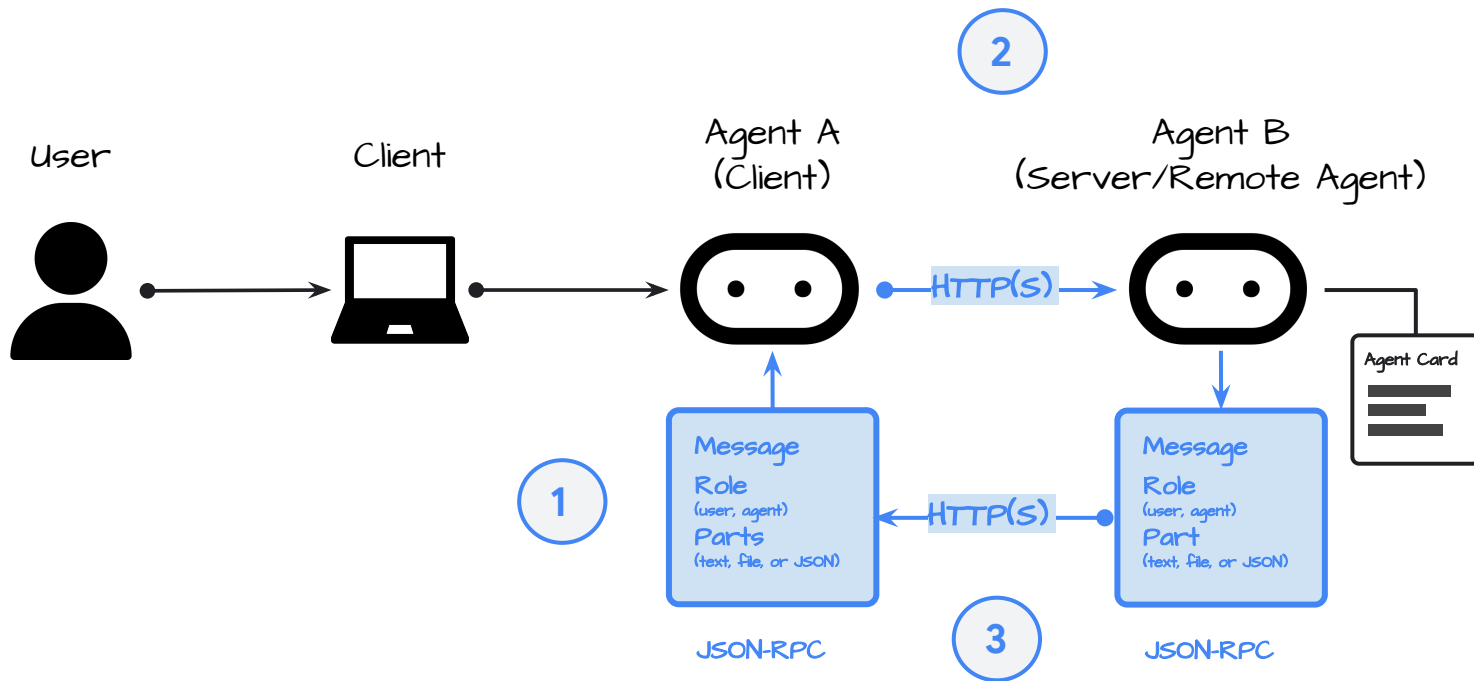
Step 2: Basic Interaction

How do we actually talk?



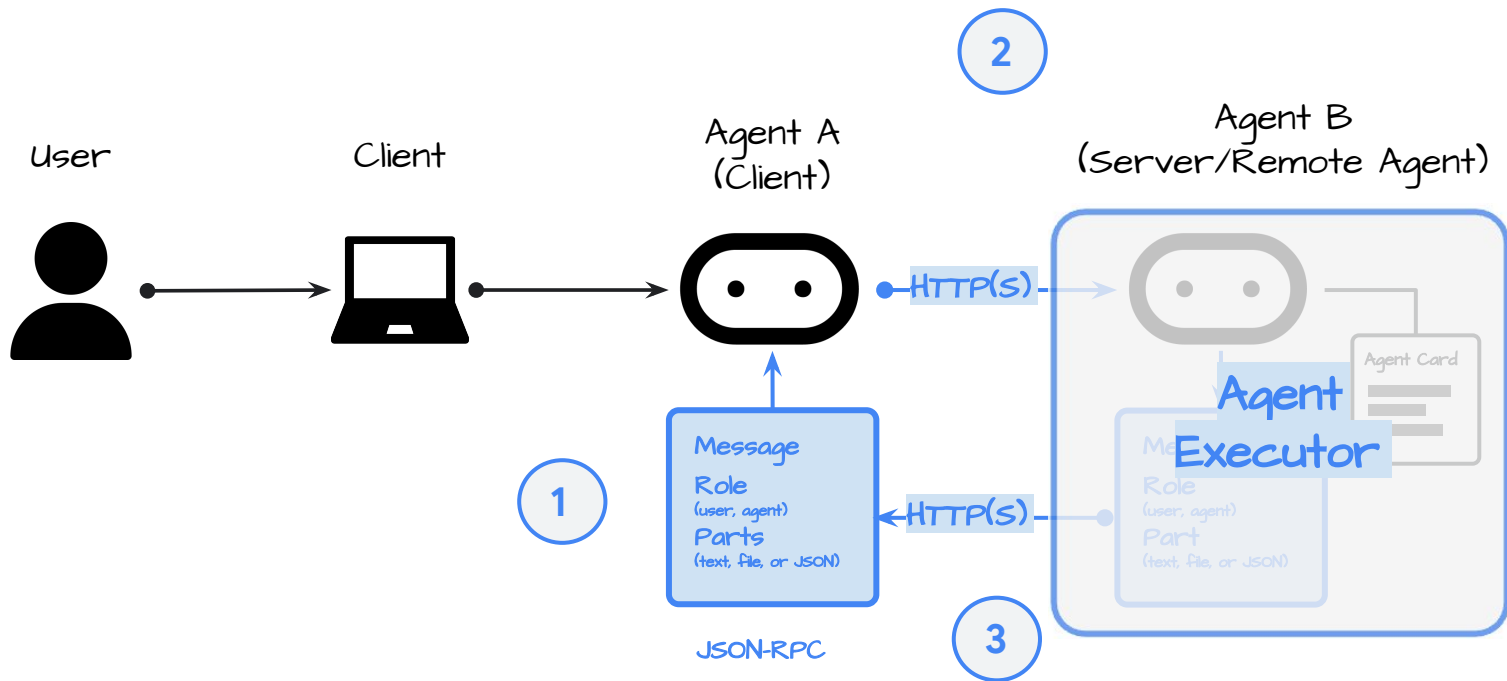
Step 2: Basic Interaction

Messages, Tasks & Parts



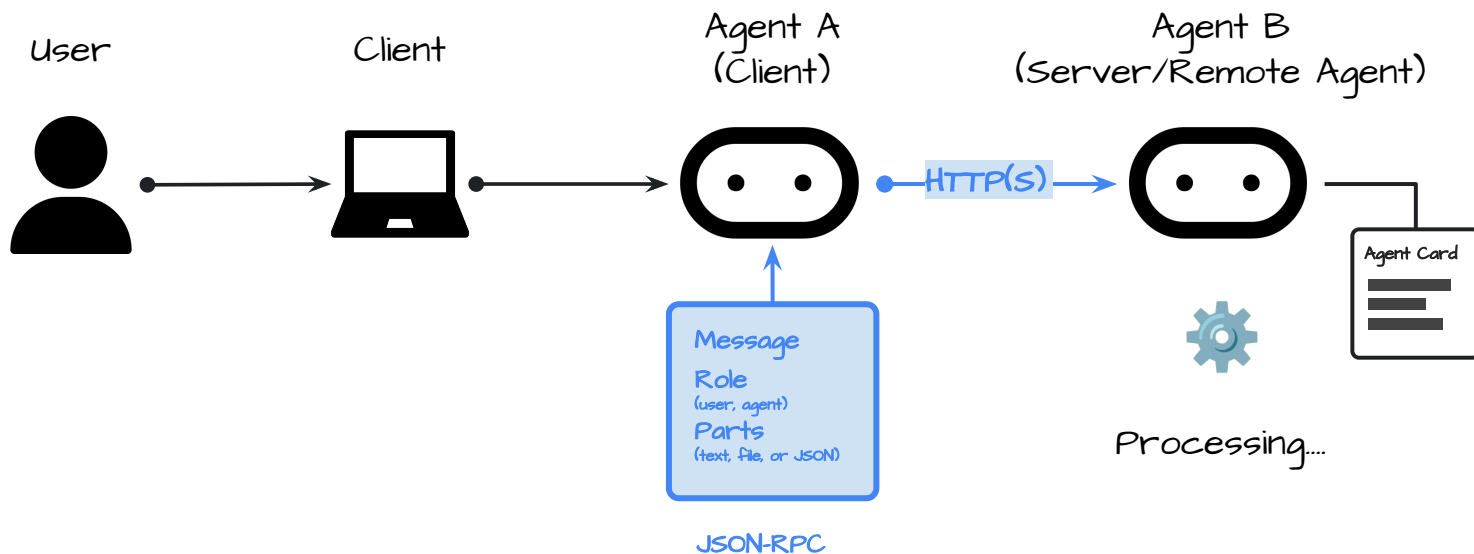
Step 2: Basic Interaction

The Agent Executor



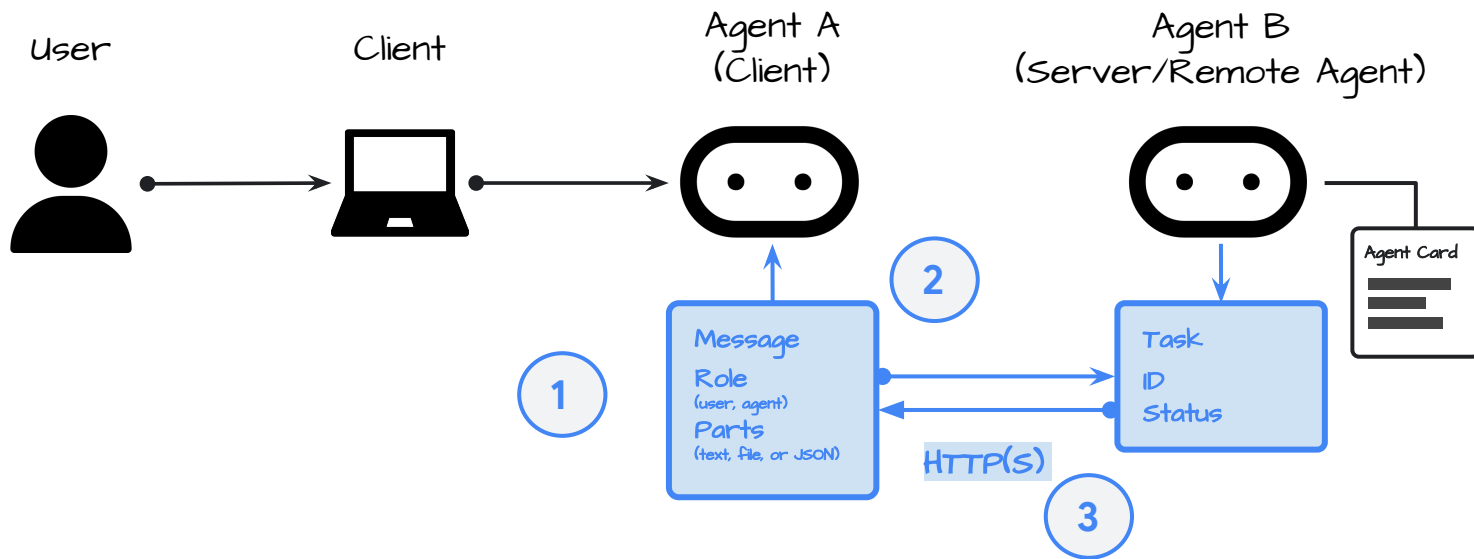
Step 3: Handling Real Work

Are we there yet?



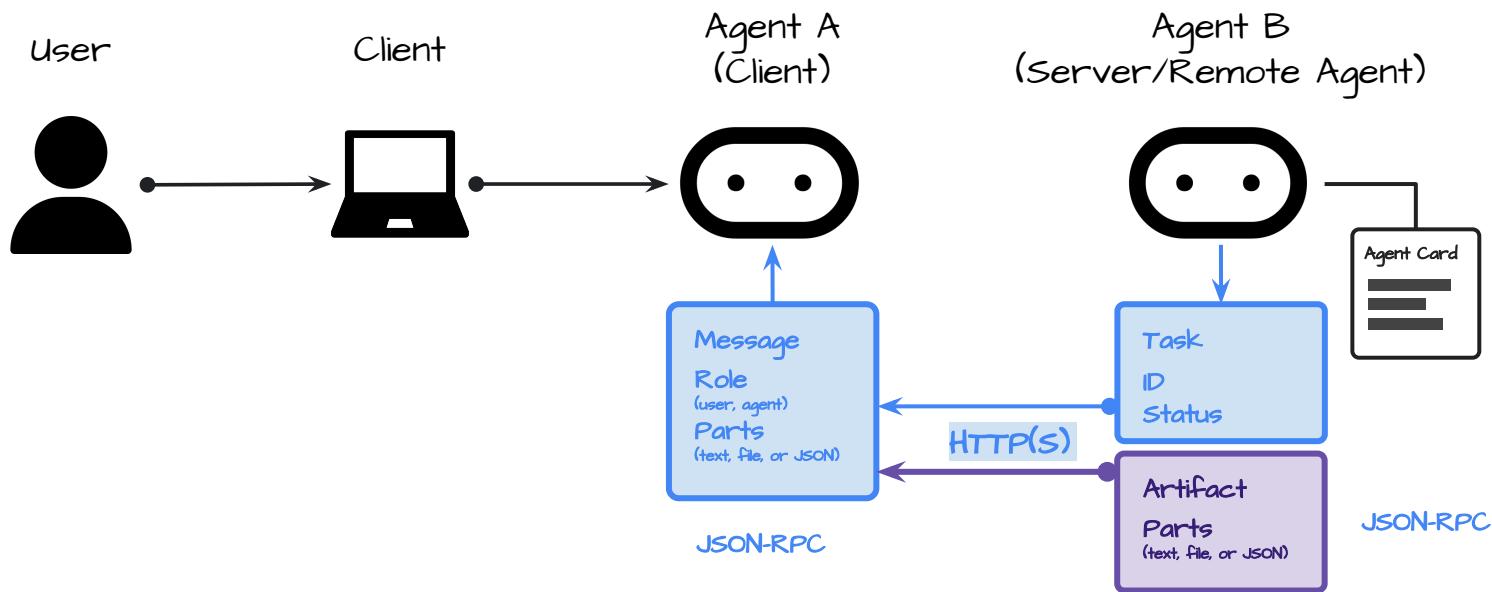
Step 3: Handling Real Work

Task Lifecycle & Polling



Step 3: Handling Real Work

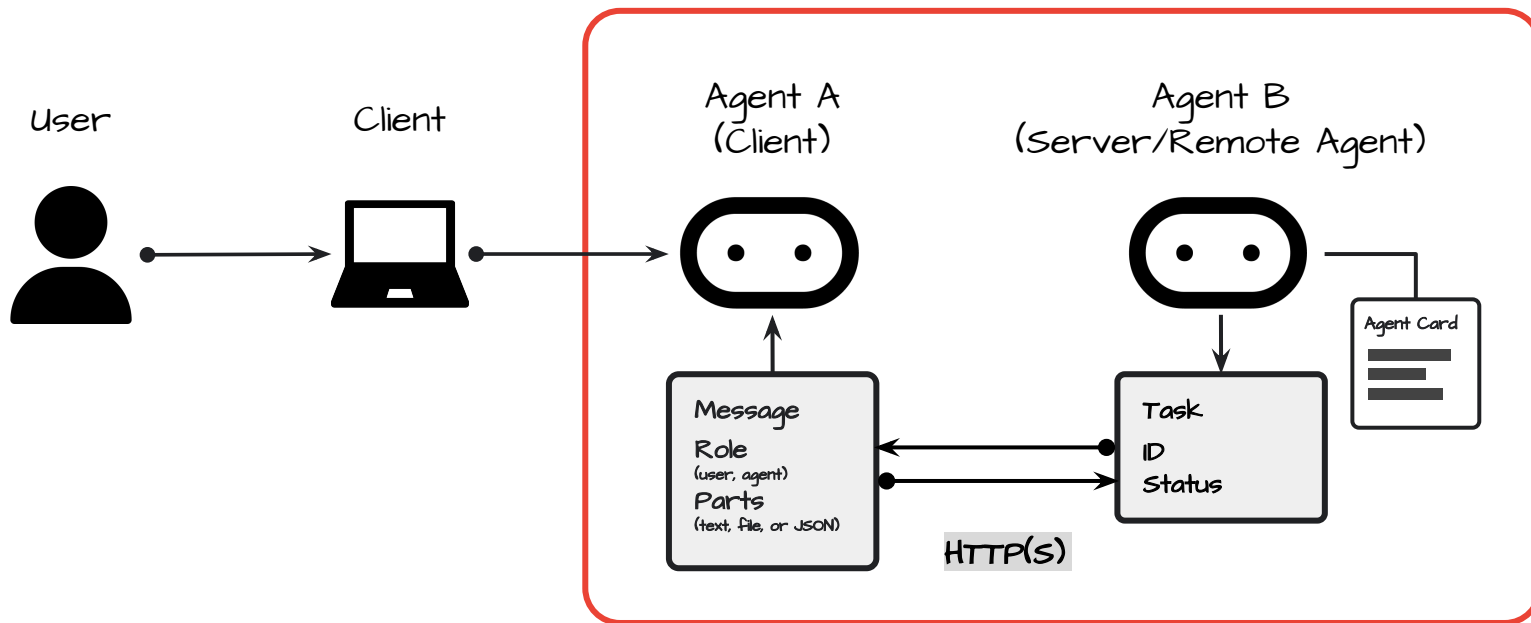
Task Lifecycle & Polling



Task Lifecycle & Polling

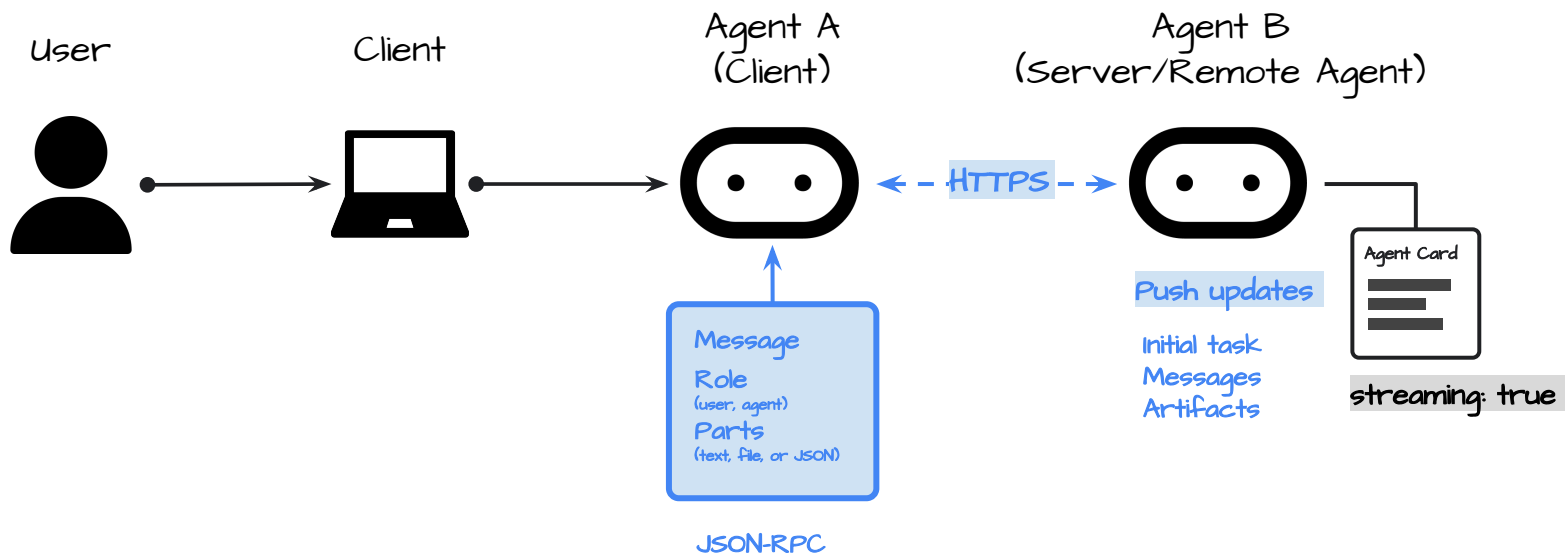
Challenge

✗ Polling is inefficient!



Step 4: Real-time Updates

Streaming with SSE



Tutorial

Get code 🖱️

goo.gle/a2a-tutorial



Agent Card with capabilities

Example

```
capabilities = AgentCapabilities(streaming=True, push_notifications=True)

agent_card = AgentCard(
    name='Currency Agent',
    description='Helps with exchange rates for currencies',
    url=f'http://{host}:{port}/',
    version='1.0.0',
    default_input_modes=CurrencyAgent.SUPPORTED_CONTENT_TYPES,
    default_output_modes=CurrencyAgent.SUPPORTED_CONTENT_TYPES,
    capabilities=capabilities,
    skills=[skill],
)
```

Agent Executor

Example

```
class CurrencyAgentExecutor(AgentExecutor):  
    """Currency Conversion AgentExecutor Example."""  
  
    def __init__(self):  
        self.agent = CurrencyAgent()
```

Agent Executor

Example

```
async def execute(self, context: RequestContext, event_queue: EventQueue) -> None:
    ...
    query = context.get_user_input()
    task = context.current_task
    if not task:
        task = new_task(context.message)
        event_queue.enqueue_event(task)
    updater = TaskUpdater(event_queue, task.id, task.context_id)
```

Agent Executor

Example

```
async def execute(self, context: RequestContext, event_queue: EventQueue) -> None:
    ...
    query = context.get_user_input()
    task = context.current_task
    if not task:
        task = new_task(context.message)
        event_queue.enqueue_event(task)
    updater = TaskUpdater(event_queue, task.id, task.context_id)
```

Agent Executor

Example

```
async def execute(self, context: RequestContext, event_queue: EventQueue) -> None:
    ...
    query = context.get_user_input()
    task = context.current_task
    if not task:
        task = new_task(context.message)
        event_queue.enqueue_event(task)
    updater = TaskUpdater(event_queue, task.id, task.context_id)
```


Agent Executor

Example

```
async def execute(self, context: RequestContext, event_queue: EventQueue) -> None:
    ...
    query = context.get_user_input()
    task = context.current_task
    if not task:
        task = new_task(context.message)
        event_queue.enqueue_event(task)
    updater = TaskUpdater(event_queue, task.id, task.contextId)
```

Agent Executor

Example

```
async def execute(self, context: RequestContext, event_queue: EventQueue) -> None:
    ...
    async for item in self.agent.stream(query, task.contextId):
        is_task_complete = item['is_task_complete']
        require_user_input = item['require_user_input']
        if not is_task_complete and not require_user_input:
            updater.update_status(...)
        elif require_user_input:
            updater.update_status(..., final=True)
            break
        else:
            updater.add_artifact(...)
            updater.complete()
            break
```

Agent Executor

Example

```
async def execute(self, context: RequestContext, event_queue: EventQueue) -> None:
    ...
    async for item in self.agent.stream(query, task.contextId):
        is_task_complete = item['is_task_complete']
        require_user_input = item['require_user_input']
        if not is_task_complete and not require_user_input:
            updater.update_status(...)
        elif require_user_input:
            updater.update_status(..., final=True)
            break
        else:
            updater.add_artifact(...)
            updater.complete()
            break
```

Agent Executor

Example

```
async def execute(self, context: RequestContext, event_queue: EventQueue) -> None:
    ...
    async for item in self.agent.stream(query, task.contextId):
        is_task_complete = item['is_task_complete']
        require_user_input = item['require_user_input']
        if not is_task_complete and not require_user_input:
            updater.update_status(...)
        elif require_user_input:
            updater.update_status(..., final=True)
            break
        else:
            updater.add_artifact(...)
            updater.complete()
            break
```

Agent Executor

Example

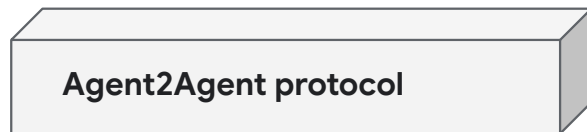
```
async def execute(self, context: RequestContext, event_queue: EventQueue) -> None:
    ...
    async for item in self.agent.stream(query, task.contextId):
        is_task_complete = item['is_task_complete']
        require_user_input = item['require_user_input']
        if not is_task_complete and not require_user_input:
            updater.update_status(...)
        elif require_user_input:
            updater.update_status(..., final=True)
            break
        else:
            updater.add_artifact(...)
            updater.complete()
            break
```

Agent Executor

Example

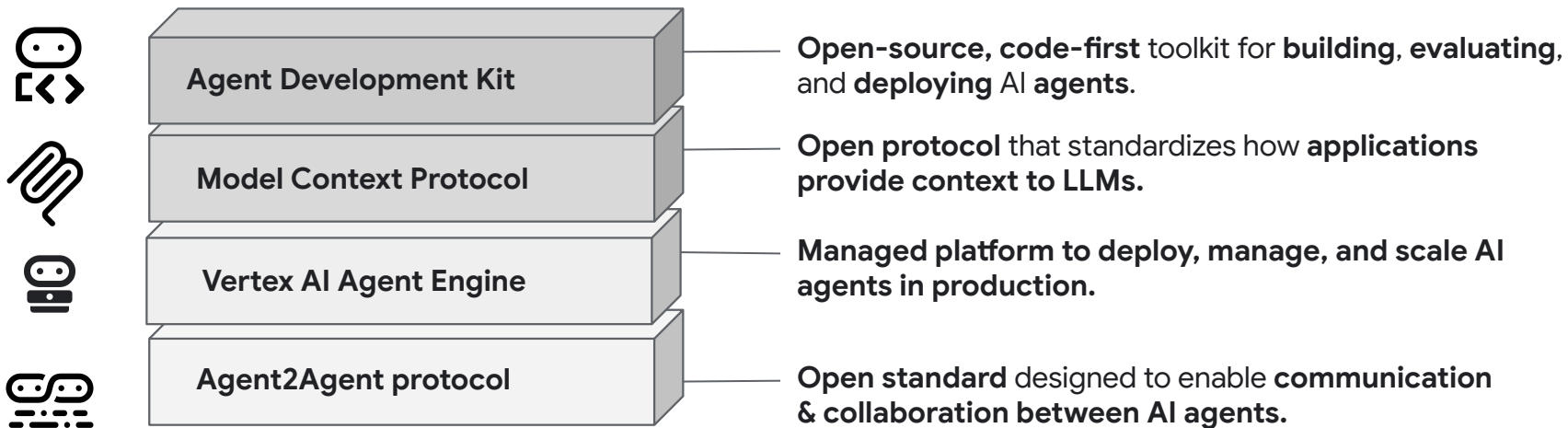
```
async def execute(self, context: RequestContext, event_queue: EventQueue) -> None:
    ...
    async for item in self.agent.stream(query, task.contextId):
        is_task_complete = item['is_task_complete']
        require_user_input = item['require_user_input']
        if not is_task_complete and not require_user_input:
            updater.update_status(...)
        elif require_user_input:
            updater.update_status(..., final=True)
            break
        else:
            updater.add_artifact(...)
            updater.complete()
            break
```

This is it?



Open standard designed to enable communication & collaboration between AI agents.

A possible agentic stack



A2A vs MCP???

A2A ❤️ MCP

Complementary, Not Competing



Model Context Protocol (MCP)

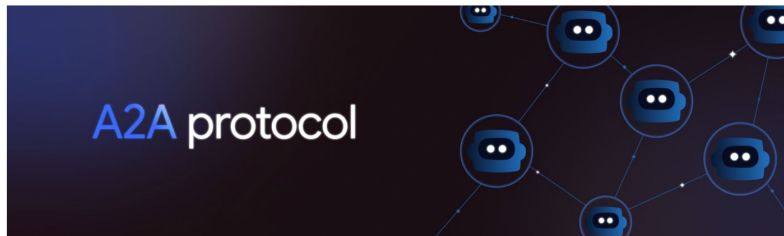
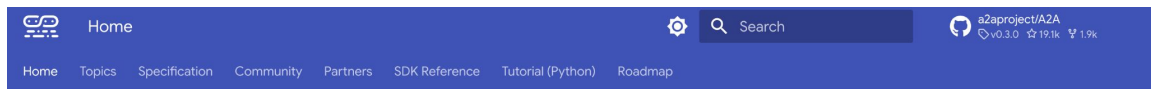
- Connects agents to **tools, APIs, and resources**.
- Think: *How an agent uses its capabilities (function calling).*
- Example: Agent uses MCP to call a weather API tool.



Agent2Agent Protocol (A2A)

- Facilitates dynamic communication **between different agents** as peers.
- Think: *How agents collaborate, delegate, and manage shared tasks.*
- Example: A Travel Agent (A2A) asks a Flight Booking Agent (A2A) to find flights.

A2A Documentation



The **Agent2Agent (A2A) Protocol** is an open standard designed to enable seamless communication and collaboration between AI agents. In a world where agents are built using diverse frameworks and by different vendors, A2A provides a common language, breaking down silos and fostering interoperability.

Build with **ADK** (or any framework), equip with **MCP** (or any tool), and communicate with **A2A**, to remote agents, local agents, and humans.

A2A Announcements

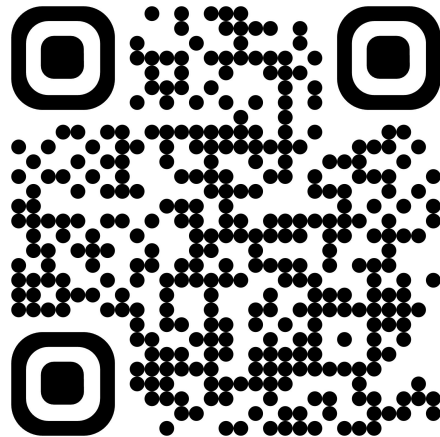
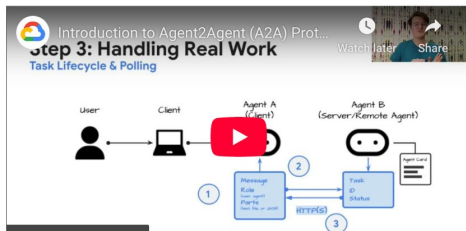
Ramp up quickly

- [Announcing the A2A Protocol \(Apr\)](#) our initial blog
- [Agents are not tools \(Jun\)](#) our TL thought piece
- [Google Cloud donates A2A to Linux Foundation \(Jun\)](#) covered by Forbes

Dive deep with end to end examples

- [Designing with A2A \(O'Reilly\)](#)
- [Start the Python Tutorial](#)

Video Intro in <8 min



a2a-protocol.org

A2A SDK

```
pip install a2a-sdk
```

```
from a2a.types import AgentCard, AgentSkill

agent_card = AgentCard(
    name='Hello World Agent',
    description='Just a hello world agent',
    url='http://localhost:9999/',
    version='1.0.0',
    default_input_modes=['text'],
    default_output_modes=['text'],
    capabilities=AgentCapabilities(streaming=True),
    skills=[skill],
)
```



goo.gle/a2a-python-sdk

How about samples?

Get code 🙌

goo.gle/a2a-samples



A2A GitHub

- github.com/a2aproject - GitHub Organization
 - github.com/a2aproject/A2A - Protocol and Documentation
 - github.com/a2aproject/a2a-python - Python SDK
 - github.com/a2aproject/a2a-js - TypeScript/Node.js SDK
 - github.com/a2aproject/a2a-java - Java SDK
 - github.com/a2aproject/a2a-go - Go SDK
 - github.com/a2aproject/a2a-dotnet - C#/.NET SDK
 - github.com/a2aproject/a2a-samples - Code Samples/Demos
 - github.com/a2aproject/a2a-inspector - Test A2A Agents