aws
COMMUNITY DAY
MIDWEST | INDY

Skip the Middleman

Direct, Secure UI-to-Agent Communication with Amazon Bedrock **AgentCore**

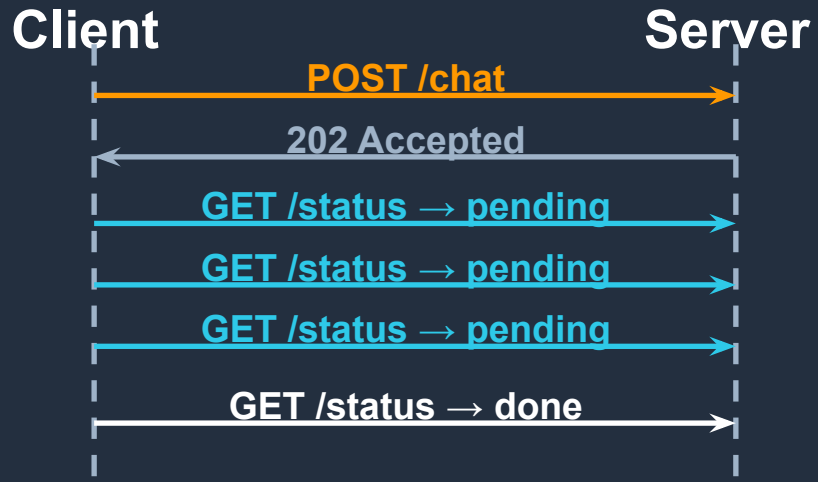
Andres Moreno
Principal Software Architect @ Caylent
AWS Community Builder - Serverless
AWS User Group Leader - AWS UG Cloud Del Norte

aws
COMMUNITY DAY
MIDWEST | INDY

Why Do We Always Start Here?

aws
COMMUNITY DAY
MIDWEST | INDY

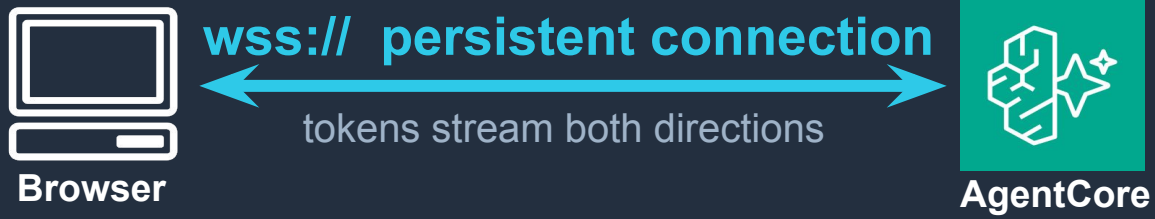
REST Is Killing The UX



Polling = wasted requests, wasted latency, wasted compute

aws
COMMUNITY DAY
MIDWEST | INDY

Direct UI-to-Agent Communication

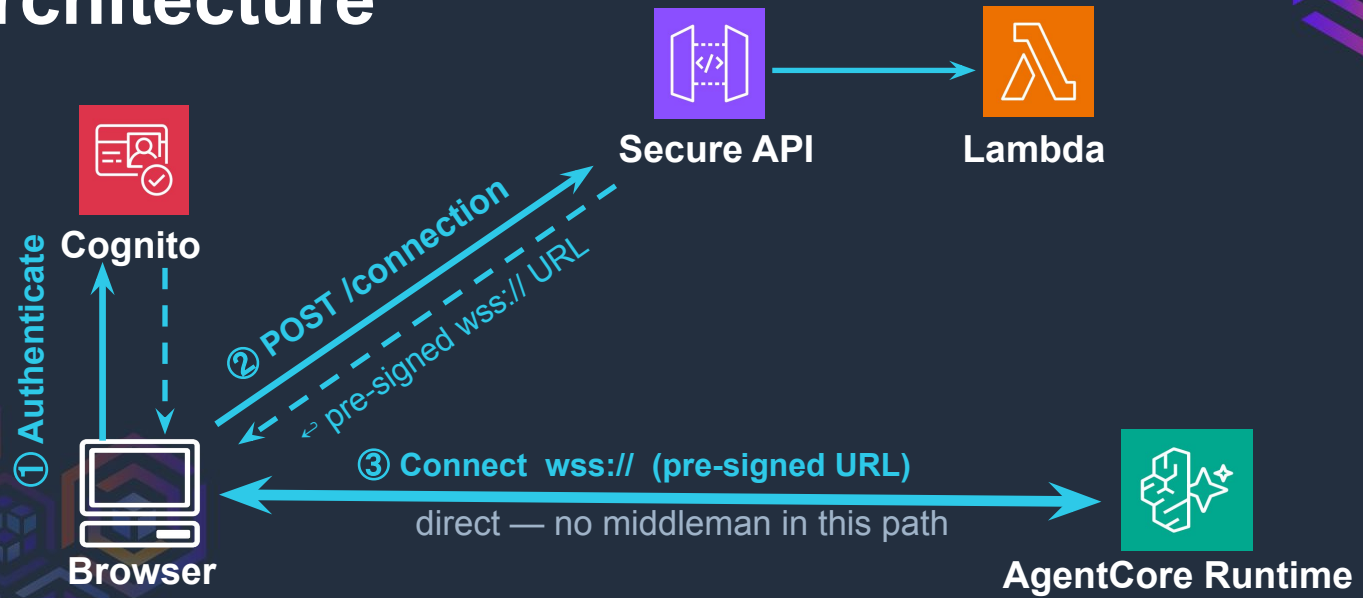


aws
COMMUNITY DAY
MIDWEST | INDY

How It Actually Works

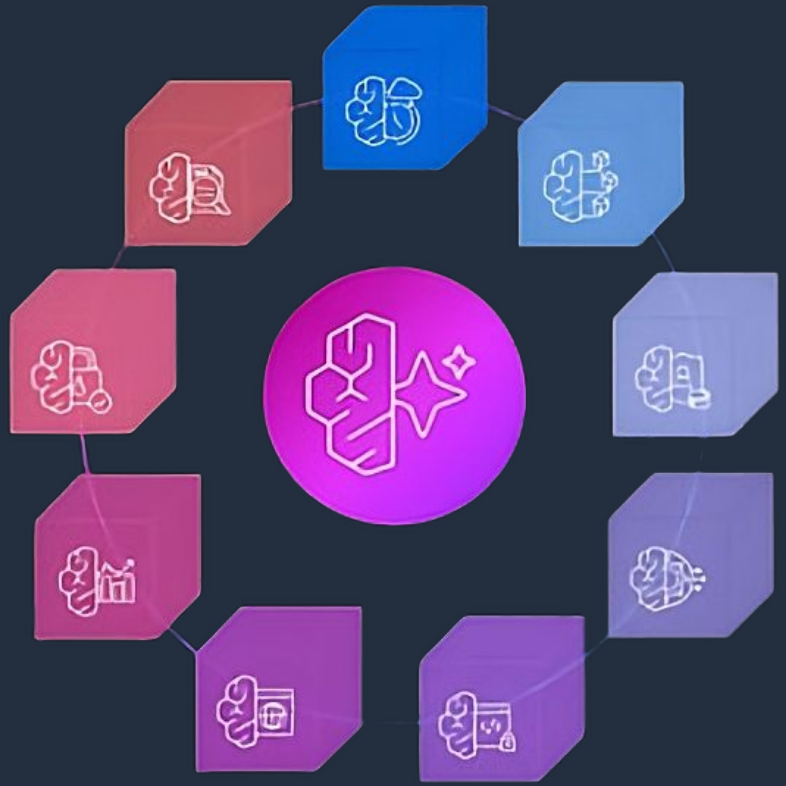
aws
COMMUNITY DAY
MIDWEST | INDY

Architecture



aws
COMMUNITY DAY
MIDWEST | INDY

- Runtime
- Memory
- Gateway
- Observability
- Eval
- Policy
- Identity
- Etc



aws
COMMUNITY DAY
MIDWEST | INDY

Live Demo

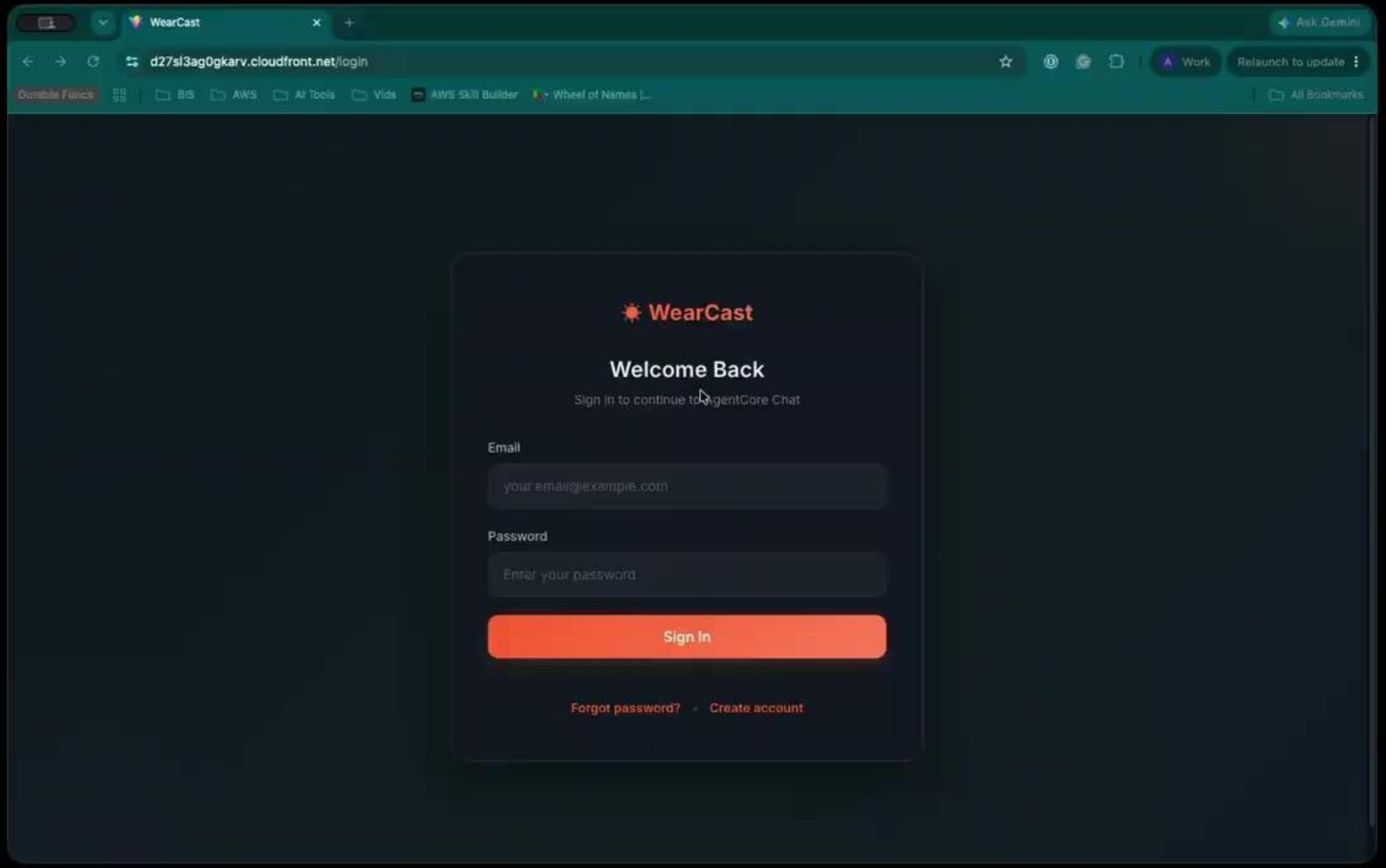
WearCast

weather agent / what to wear

The Code

The App

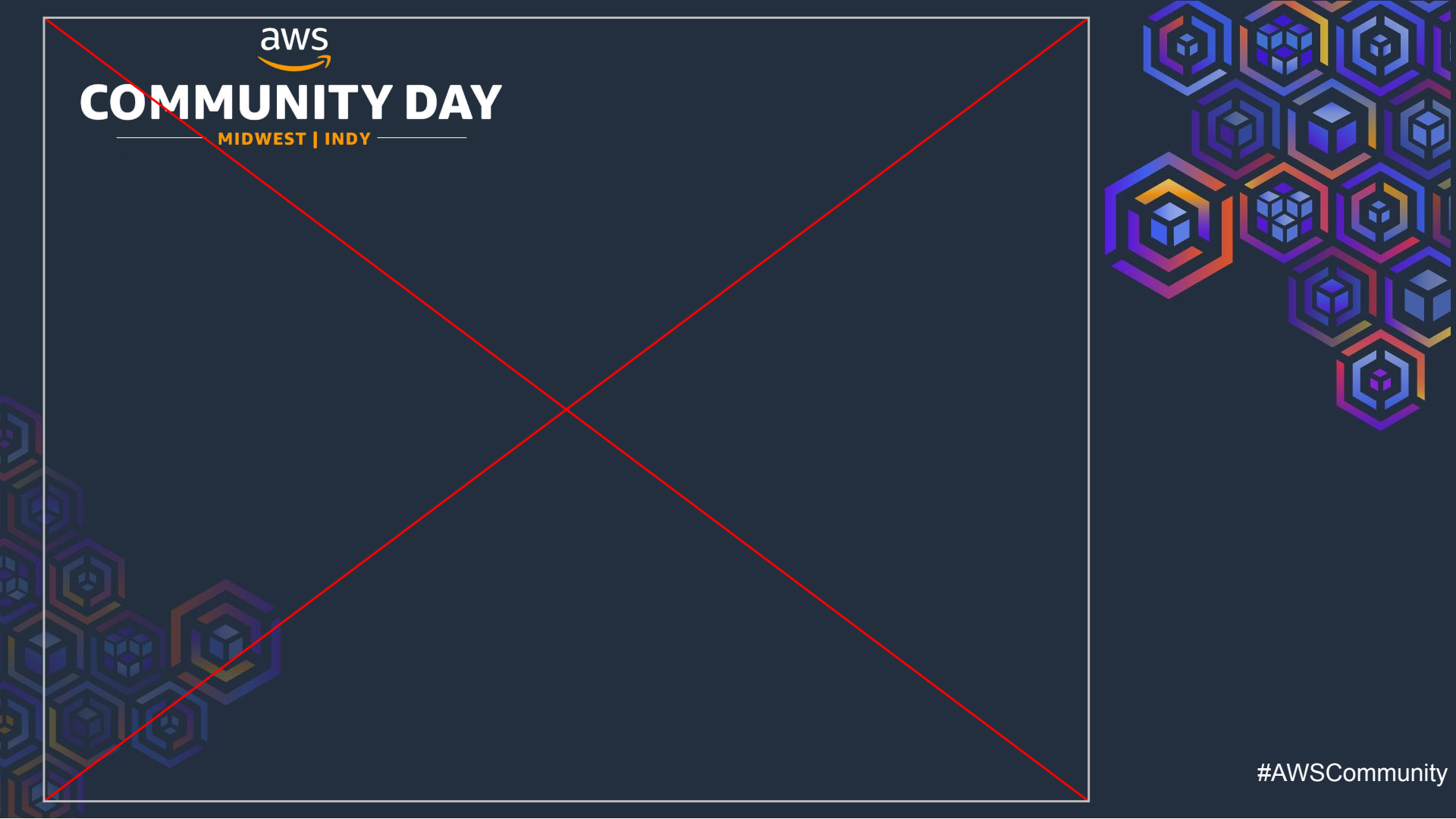
#AWSCommunity


WearCast
d27sl3ag0gkarv.cloudfront.net/login
Ask Gemini
Work
Relaunch to update
Dumbo Funcs
BIS
AWS
AI Tools
Vids
AWS Skill Builder
Wheel of Names |...
All Bookmarks
WearCast
Welcome Back
Sign in to continue to AgentCore Chat
Email
your.email@example.com
Password
Enter your password
Sign In
Forgot password?
Create account

aws

COMMUNITY DAY

MIDWEST | INDY



#AWSCommunity

aws
COMMUNITY DAY
MIDWEST | INDY

Demo Architecture

Browser

wss:// persistent connection

tokens stream both directions

Runtime
serverless hosting

agent.py

```python
from bedrock_agentcore.runtime import BedrockAgentCoreApp
from strands import Agent

app = BedrockAgentCoreApp()
@app.websocket
async def handler(websocket, context):
    agent = Agent(
        system_prompt=SYSTEM_PROMPT,
    )

    async for event in agent.stream_async(request):
        client_event = None

        client_event = handle_event(event)

        if client_event is not None:
            await websocket.send_json({
                "type": "stream_event",
                "event": client_event
            })

if __name__ == "__main__":
    app.run()
```

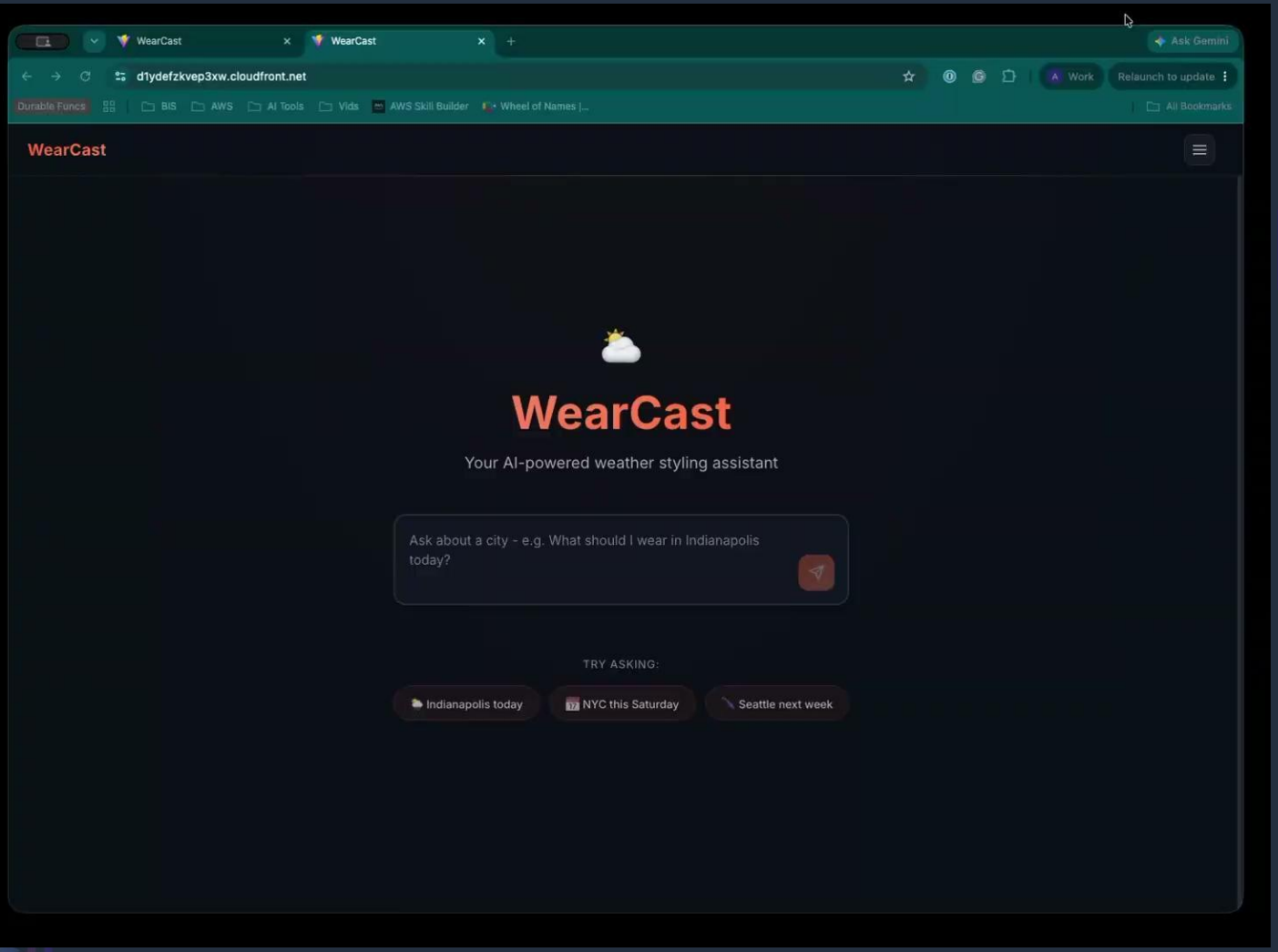
#AWSCommunity

agent.py

```
SYSTEM_PROMPT = """You are WearCast, a friendly weather-based clothing advisor. \
When the user asks about a city, call the get_weather tool once to fetch current \
conditions, then give practical outfit recommendations.

Reasoning guidelines:
- Base advice on feels_like (apparent temperature), not raw temperature.
- Recommend an umbrella or rain jacket if precipitation > 0 mm.
- Suggest a windbreaker or extra layer if wind_speed > 20 km/h.
- Layer advice: heavy coat < 20 °F, winter coat 20–35 °F, jacket 35–55 °F, \
light layer 55–70 °F, light clothing > 70 °F.
- Combine all factors into one coherent recommendation (e.g. "light jacket + \
umbrella" rather than listing rules).

Response style:
- Write 3–5 sentences so the token stream is visibly satisfying.
- Start with the city name and the plain-English condition (from the tool result).
- End with a concrete outfit recommendation.
- Use your memory tool to remember the conversation so follow-up questions like \
"What about Chicago?" are understood in context.
- Format responses in Markdown."""
```

aws
COMMUNITY DAY
MIDWEST | INDY

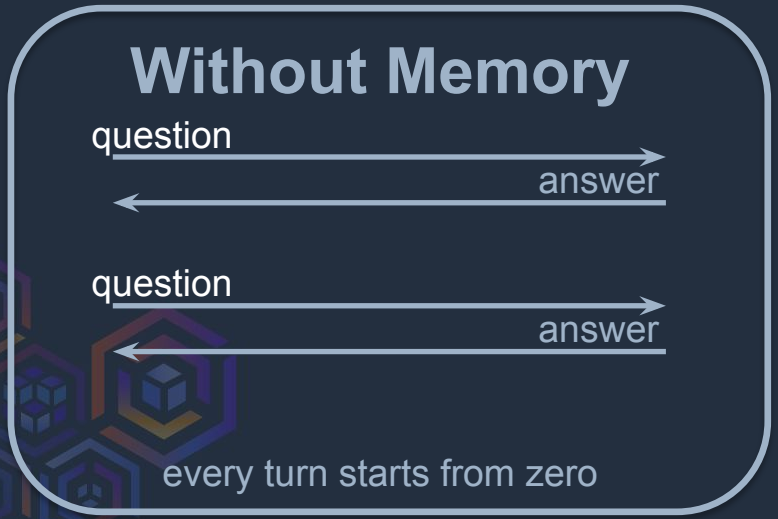
Let’s add memory!

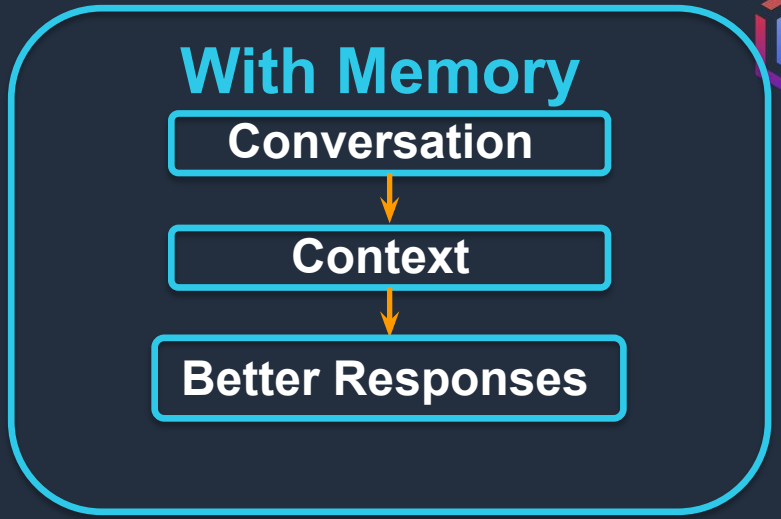


#AWSCommunity

aws
COMMUNITY DAY
MIDWEST | INDY

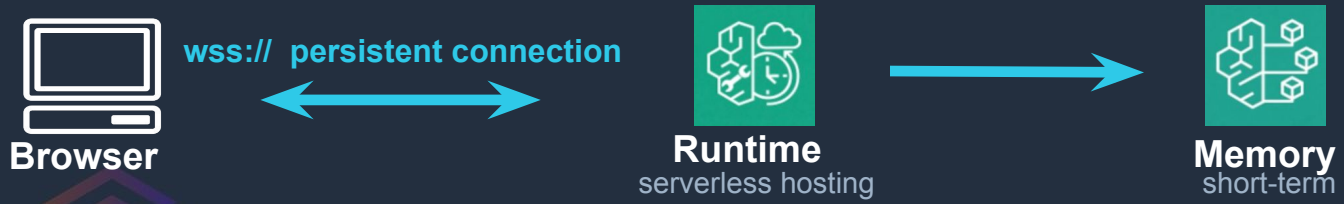
Memory Changes Everything





aws
COMMUNITY DAY
MIDWEST | INDY

Demo Architecture



aws
COMMUNITY DAY
MIDWEST | INDY

agent.py

```python
def create_session_manager(runtime_session_id: str, user_id: str = None):
    """Create AgentCore Memory session manager for conversation persistence."""
    actor_id = user_id if user_id else "user"

    config = AgentCoreMemoryConfig(
        memory_id=AGENTCORE_MEMORY_ID,
        session_id=runtime_session_id,
        actor_id=actor_id
    )

    return AgentCoreMemorySessionManager(
        agentcore_memory_config=config,
        region_name=AWS_REGION
    )
```

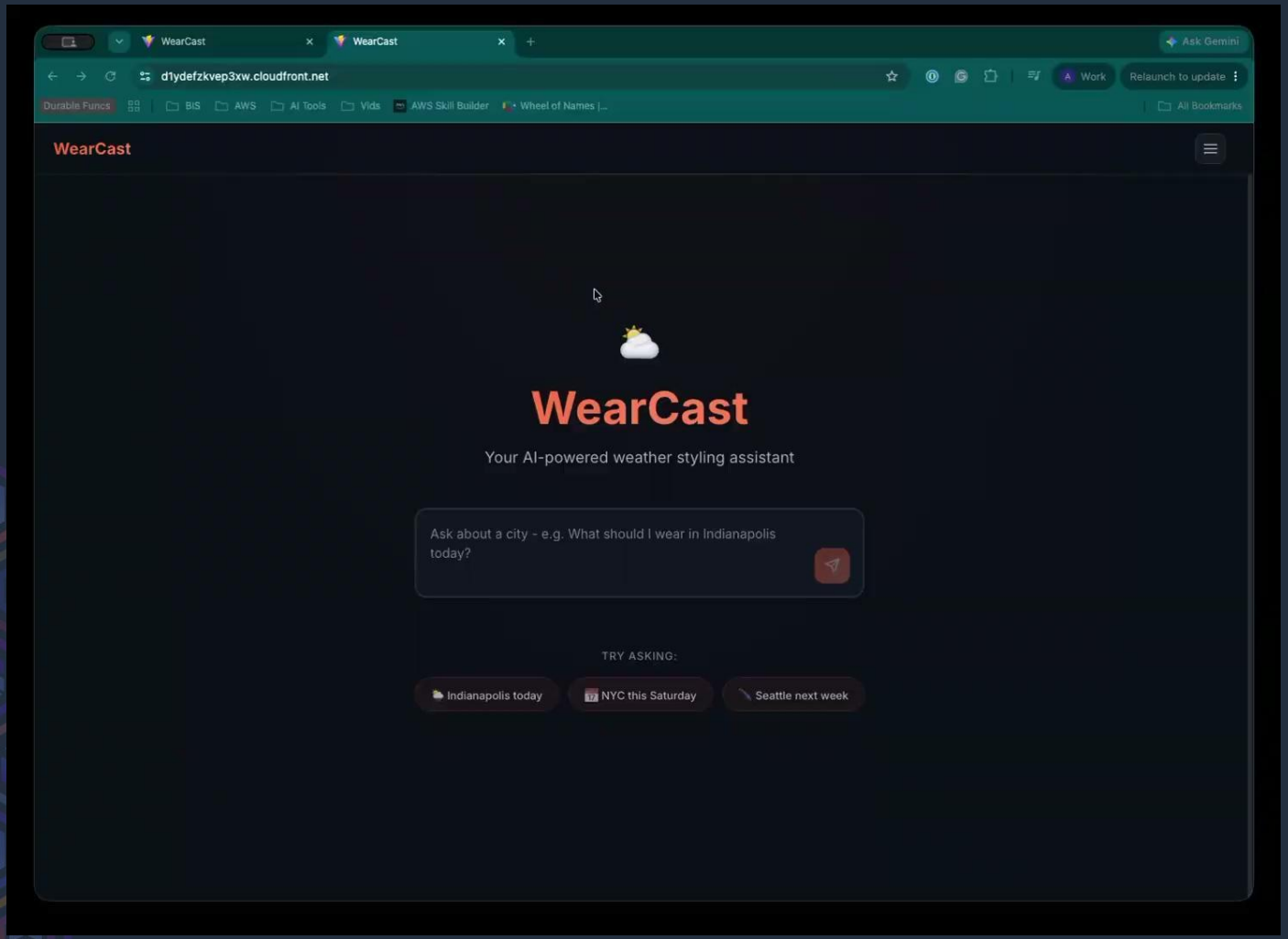
agent.py

```python
from bedrock_agentcore.runtime import BedrockAgentCoreApp
from strands import Agent

app = BedrockAgentCoreApp()
@app.websocket
async def handler(websocket, context):
    agent = Agent(
        system_prompt=SYSTEM_PROMPT,
    )

    async for event in agent.stream_async(request):
        client_event = None

        client_event = handle_event(event)

        if client_event is not None:
            await websocket.send_json({
                "type": "stream_event",
                "event": client_event
            })

if __name__ == "__main__":
    app.run()
```

#AWSCommunity

agent.py

```python
from bedrock_agentcore.runtime import BedrockAgentCoreApp
from strands import Agent

app = BedrockAgentCoreApp()
@app.websocket
async def handler(websocket, context):
    agent = Agent(
        system_prompt=SYSTEM_PROMPT,
        session_manager=session_manager,
    )

    async for event in agent.stream_async(request):
        client_event = None

        client_event = handle_event(event)

        if client_event is not None:
            await websocket.send_json({
                "type": "stream_event",
                "event": client_event
            })

if __name__ == "__main__":
    app.run()
```



#AWSCommunity

aws
COMMUNITY DAY
MIDWEST | INDY

An agent without tools is just a chatbot.

Let’s add tools!

aws

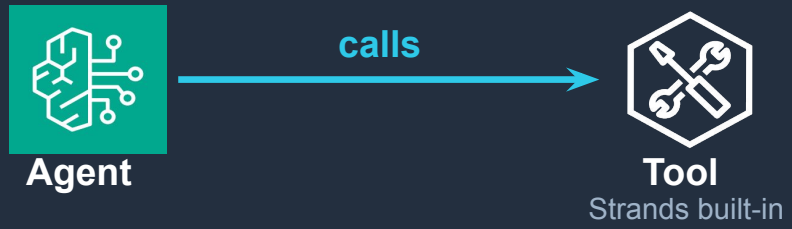
COMMUNITY DAY

MIDWEST | INDY

#AWSCommunity

aws
COMMUNITY DAY
MIDWEST | INDY

Giving The Agent Tools



agent.py

```python
@tool
def get_weather(city: str) -> dict:
    geo_url = (
        "https://geocoding-api.open-meteo.com/v1/search"
        f"?name={urllib.parse.quote(city)}&count=1"
    )
    with urllib.request.urlopen(geo_url, timeout=10) as resp:
        geo_data = json.loads(resp.read())

    results = geo_data.get("results")
    if not results:
        return {"error": f"City not found: {city}"}

    place = results[0]
    lat, lon = place["latitude"], place["longitude"]
    resolved_name = place.get("name", city)

    forecast_url = (
        "https://api.open-meteo.com/v1/forecast"
        f"?latitude={lat}&longitude={lon}"
        "&current=temperature_2m,apparent_temperature,precipitation,weather_code,wind_speed_10m"
        "&temperature_unit=fahrenheit"
    )
    with urllib.request.urlopen(forecast_url, timeout=10) as resp:
        forecast_data = json.loads(resp.read())

    current = forecast_data["current"]
    code = current["weather_code"]
    condition = _WMO_CONDITIONS.get(code, f"weather code {code}")

    return {
        "city": resolved_name,
        "temperature": current["temperature_2m"],
        "feels_like": current["apparent_temperature"],
        "precipitation": current["precipitation"],
        "wind_speed": current["wind_speed_10m"],
        "condition": condition,
    }
```

agent.py

```python
from bedrock_agentcore.runtime import BedrockAgentCoreApp
from strands import Agent, tool

app = BedrockAgentCoreApp()

@app.websocket
async def websocket_handler(websocket, context):

    agent = Agent(
        system_prompt=SYSTEM_PROMPT,
        session_manager=session_manager,
    )

    async for event in agent.stream_async(request):
        client_event = None

        client_event = handle_event(event)

        if client_event is not None:
            await websocket.send_json({
                "type": "stream_event",
                "event": client_event
            })

if __name__ == "__main__":
    app.run()
```

#AWSCommunity

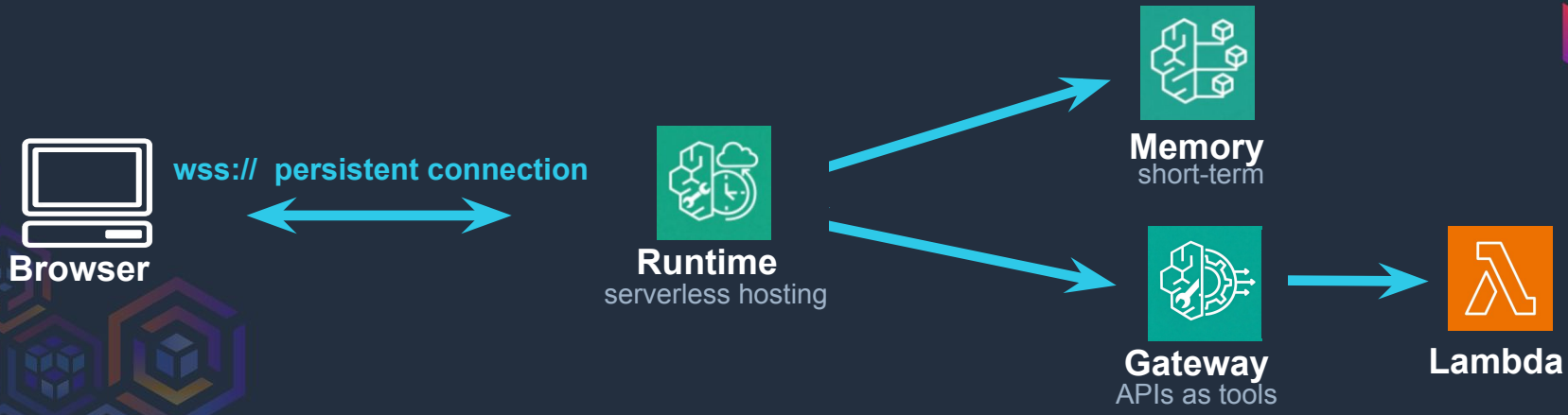
agent.py

```python
from bedrock_agentcore.runtime import BedrockAgentCoreApp
from strands import Agent, tool

app = BedrockAgentCoreApp()

@app.websocket
async def websocket_handler(websocket, context):

    agent = Agent(
        tools=[get_weather],
        system_prompt=SYSTEM_PROMPT,
        session_manager=session_manager,
    )

    async for event in agent.stream_async(request):
        client_event = None

        client_event = handle_event(event)

        if client_event is not None:
            await websocket.send_json({
                "type": "stream_event",
                "event": client_event
            })

if __name__ == "__main__":
    app.run()
```

#AWSCommunity

aws
COMMUNITY DAY
MIDWEST | INDY

Demo Architecture



aws
COMMUNITY DAY
MIDWEST | INDY

agent.py

```python
def create_gateway_mcp_client() → MCPClient:
    return MCPClient(
        lambda: streamablehttp_client(url=AGENTCORE_GATEWAY_URL),
        prefix="gateway",
    )
```

agent.py

```python
from bedrock_agentcore.runtime import BedrockAgentCoreApp
from strands import Agent, tool

app = BedrockAgentCoreApp()

@app.websocket
async def websocket_handler(websocket, context):

    agent = Agent(
        tools=[get_weather],
        system_prompt=SYSTEM_PROMPT,
        session_manager=session_manager,
    )

    async for event in agent.stream_async(request):
        client_event = None

        client_event = handle_event(event)

        if client_event is not None:
            await websocket.send_json({
                "type": "stream_event",
                "event": client_event
            })

if __name__ == "__main__":
    app.run()
```

#AWSCommunity

agent.py

```python
from bedrock_agentcore.runtime import BedrockAgentCoreApp
from strands import Agent, tool

app = BedrockAgentCoreApp()

@app.websocket
async def websocket_handler(websocket, context):

    agent = Agent(
        tools=[gateway_mcp_client],
        system_prompt=SYSTEM_PROMPT,
        session_manager=session_manager,
    )

    async for event in agent.stream_async(request):
        client_event = None

        client_event = handle_event(event)

        if client_event is not None:
            await websocket.send_json({
                "type": "stream_event",
                "event": client_event
            })

if __name__ == "__main__":
    app.run()
```

aws
COMMUNITY DAY
MIDWEST | INDY

When To Skip The Middleman

Good Fit

Chat interfaces

Streaming UX

Low latency

Interactive systems

Poor Fit

Heavy orchestration

Batch workloads

Compliance-heavy systems

Complex business logic

aws
COMMUNITY DAY
MIDWEST | INDY

Questions?

Skip the Middleman: Direct, Secure UI-to-Agent Communication with Amazon Bedrock AgentCore

Survey

Socials

GitHub