AgentCore 超基礎講座

第2回 AgentCore サービス解説

Gateway / Observability / Evaluations / Code Interpreter / Browser

2026年3月5日
産業システム第二事業本部産業システム第三部
野口 碧生

アジェンダ

01 前回のおさらい
- ・Amazon Bedrock Agent Coreとは
- ・Runtime / Memory / Identity / Policy

02 サービス説明
- ・Gateway / Observability / Evaluations
  Code Interpreter / Browser

03 まとめ

本解説について

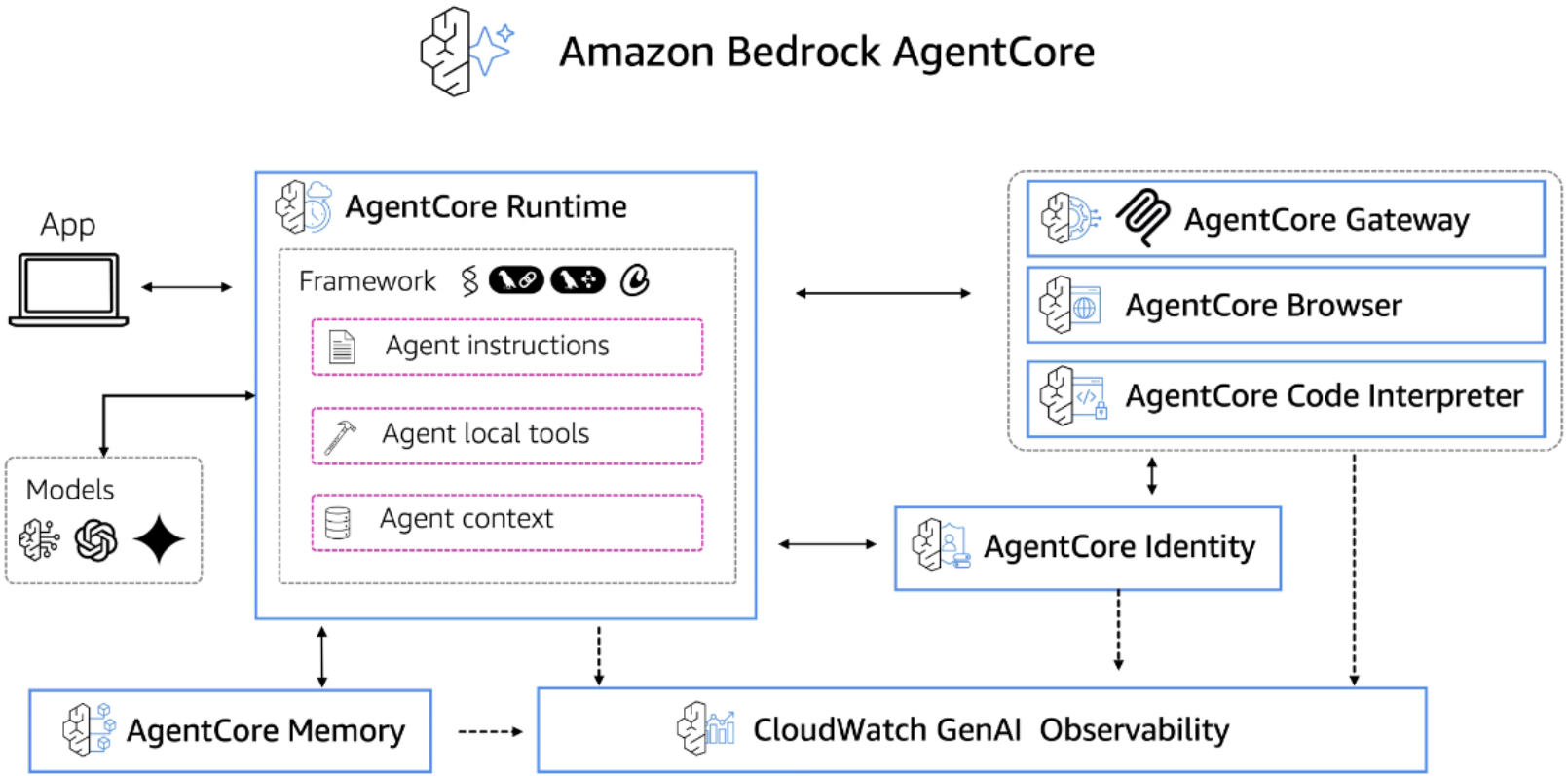
- 目的
  - AgentCoreの各サービスの役割についてイメージを持っていただくこと
  - AgentCoreの各サービスの使い方の流れを知っていただくこと

- 扱わないこと
  - エージェントの実装
    - ローカルで動くエージェントができたという段階から説明します
    - 一部エージェントの実装が登場しますがAgentCoreサービスに関わりがある部分のみを扱います
  - 詳細なコードの解説
    - サービスの利用の流れの説明に用いますが技術的な詳細には踏み込みません

前回のおさらい

Amazon Bedrock AgentCore

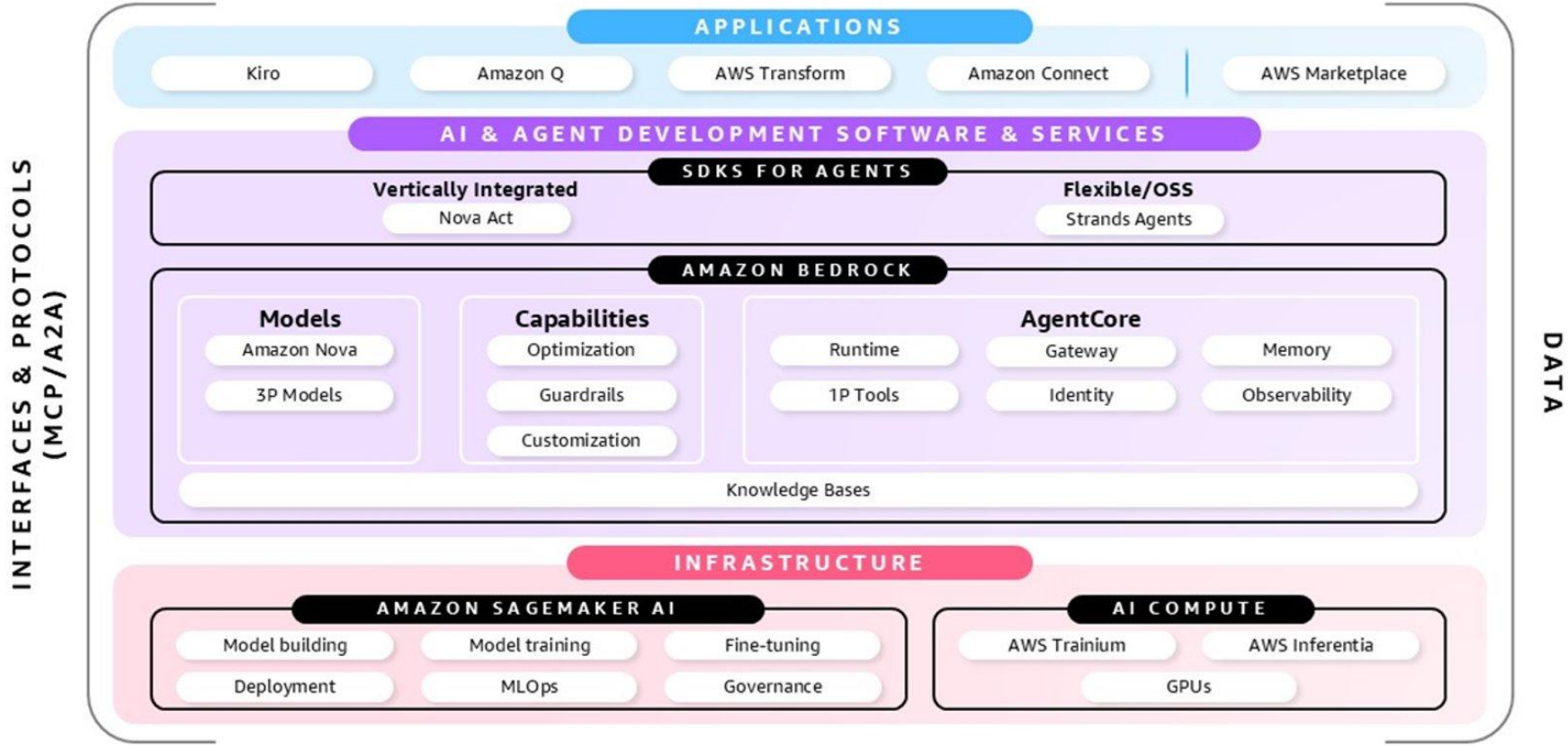
- AIエージェントを構築・展開・運用するためのフルマネージド型プラットフォーム
- インフラ管理不要（サーバーレス）
- フレームワーク非依存（Strands Agents、LangGragh、CrewAIなど）



AgentCore と Strands Agentsとの関係

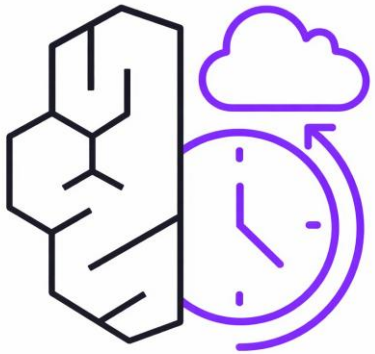
AgentCore : エージェントが稼働する基盤

Strands Agents : エージェントのSDK



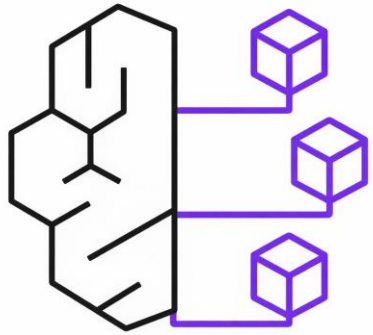
Runtime / Memory

Runtime（実行環境）



- エージェントの実行環境を提供するサーバーレス基盤
- コンポーネントを柔軟に変更可能
- セッションごとの実行環境分離によるリスク軽減

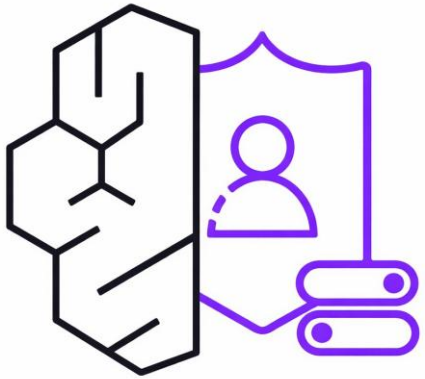
Memory（記憶）



- エージェントに「記憶」を持たせる機能
- 短期記憶（会話履歴）と長期記憶（事実の蓄積）を管理

Identity / Policy

Identity(認証・認可)

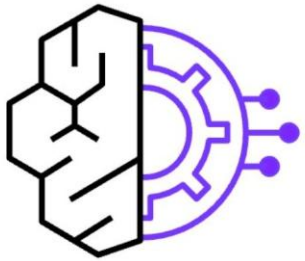


- エージェントのアイデンティティ管理
- ユーザーまたはエージェント自身の代理として権限を管理

Policy(ガードレール)

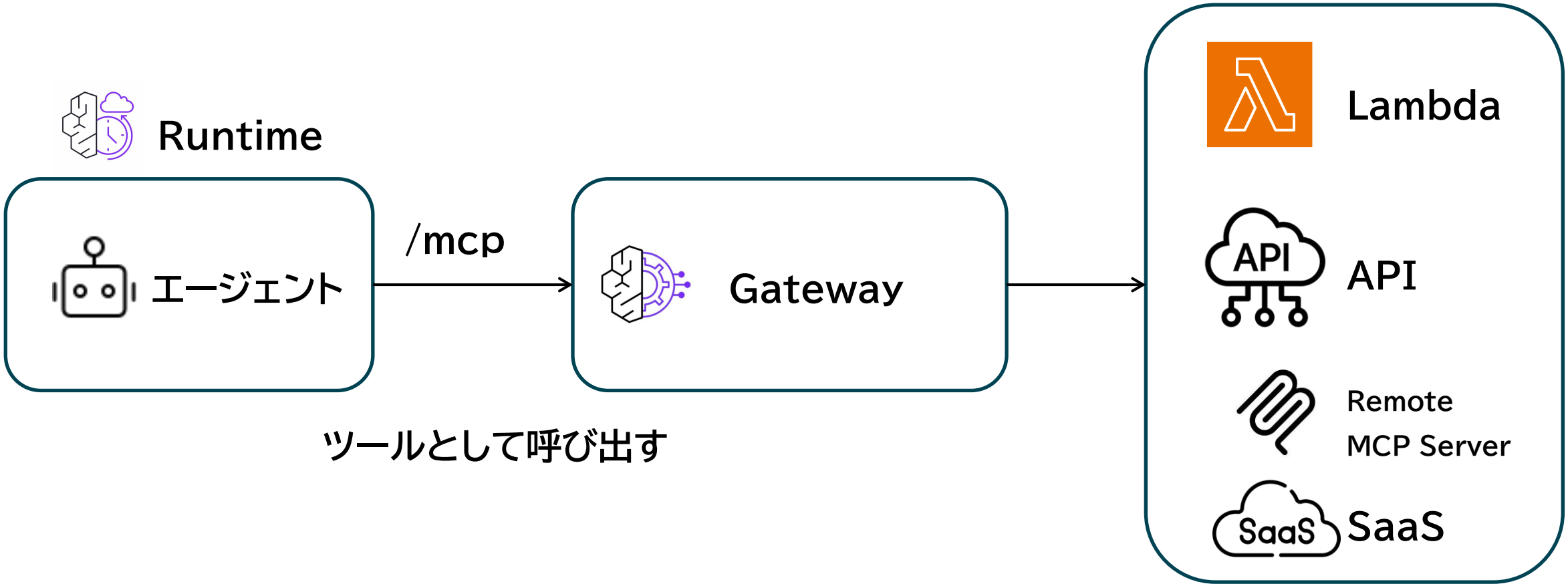
- エージェントのアクション境界線を定義し、情報漏洩を防止
- ルールベースで早く、正確に防御

Amazon Bedrock AgentCore Gateway



Amazon Bedrock AgentCore Gateway

- 外部ツールへの接続ハブ
- 既存のLambda、API、SaasなどをMCP互換のツールに変換するサービス



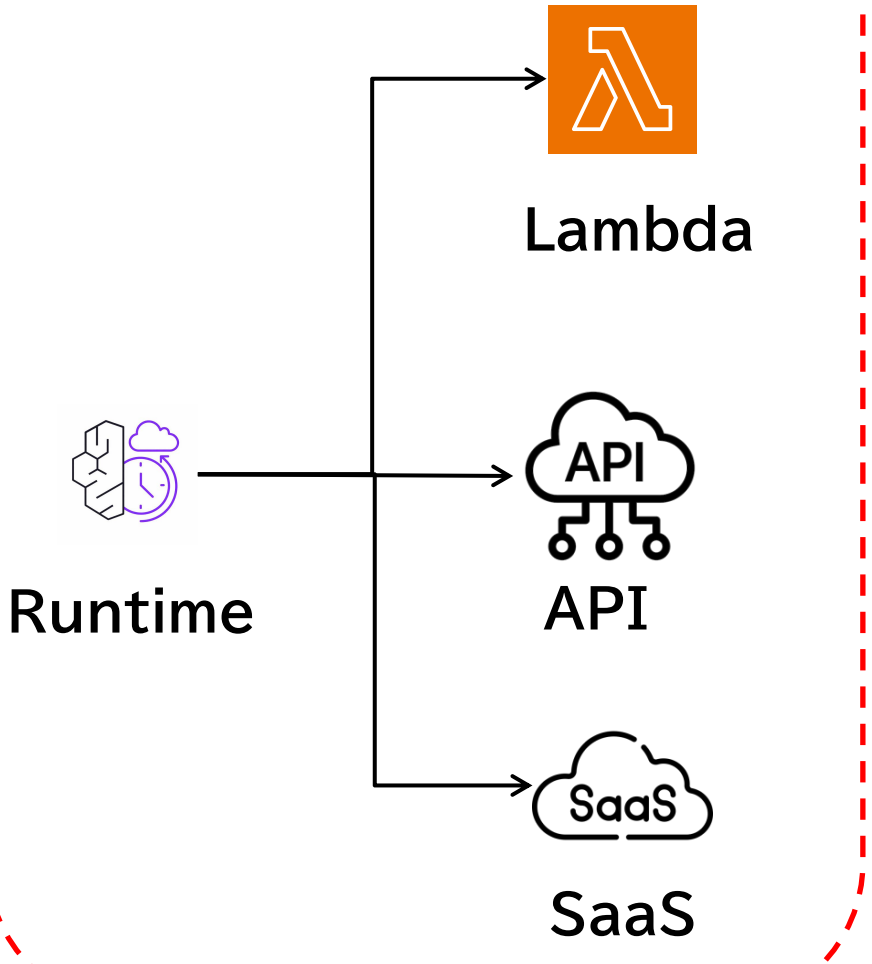
Gatewayの役割

一言で言うと

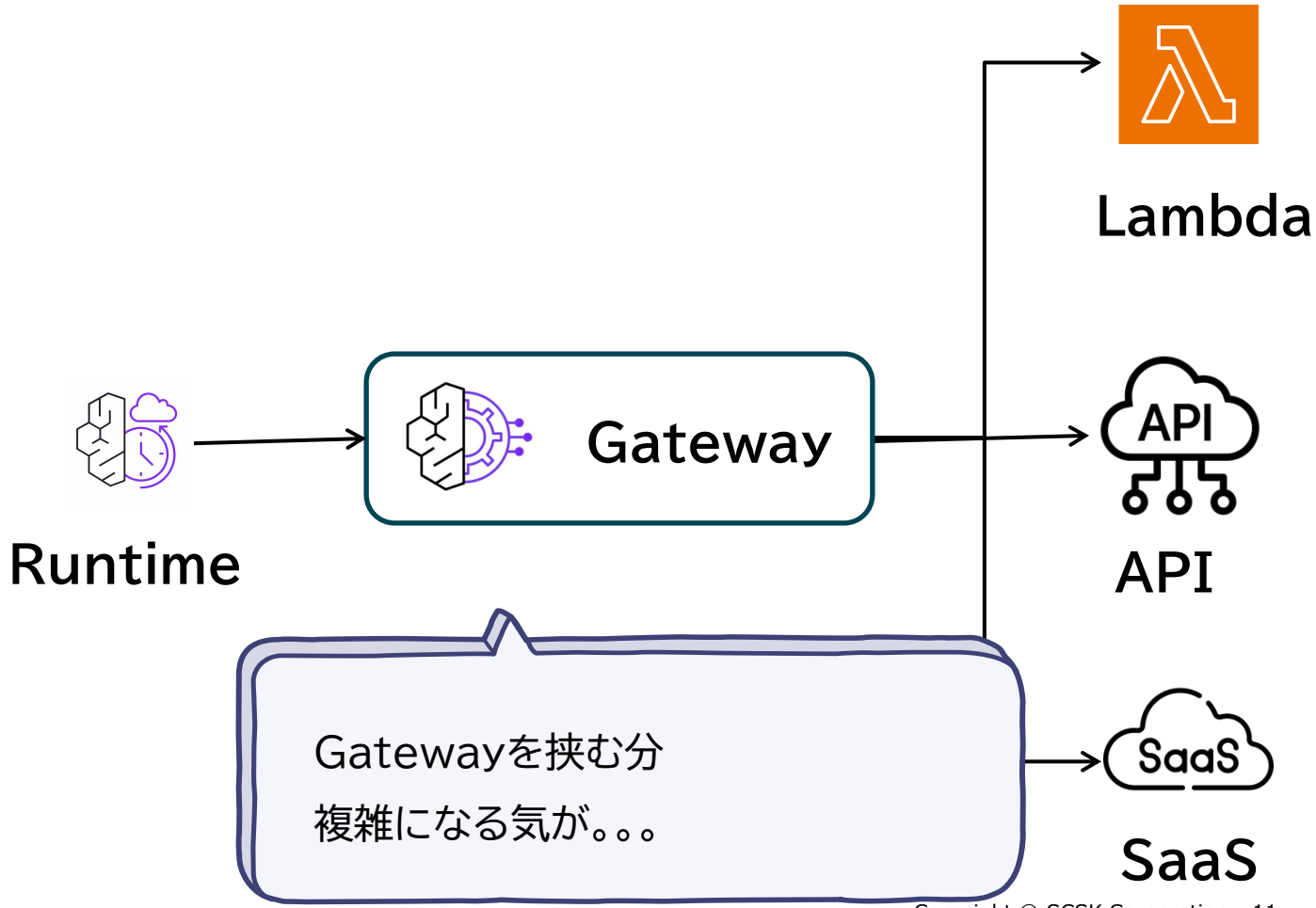
エージェントと外部システムを一元的に管理すること

Gateway有無の違い

Gatewayを使わない

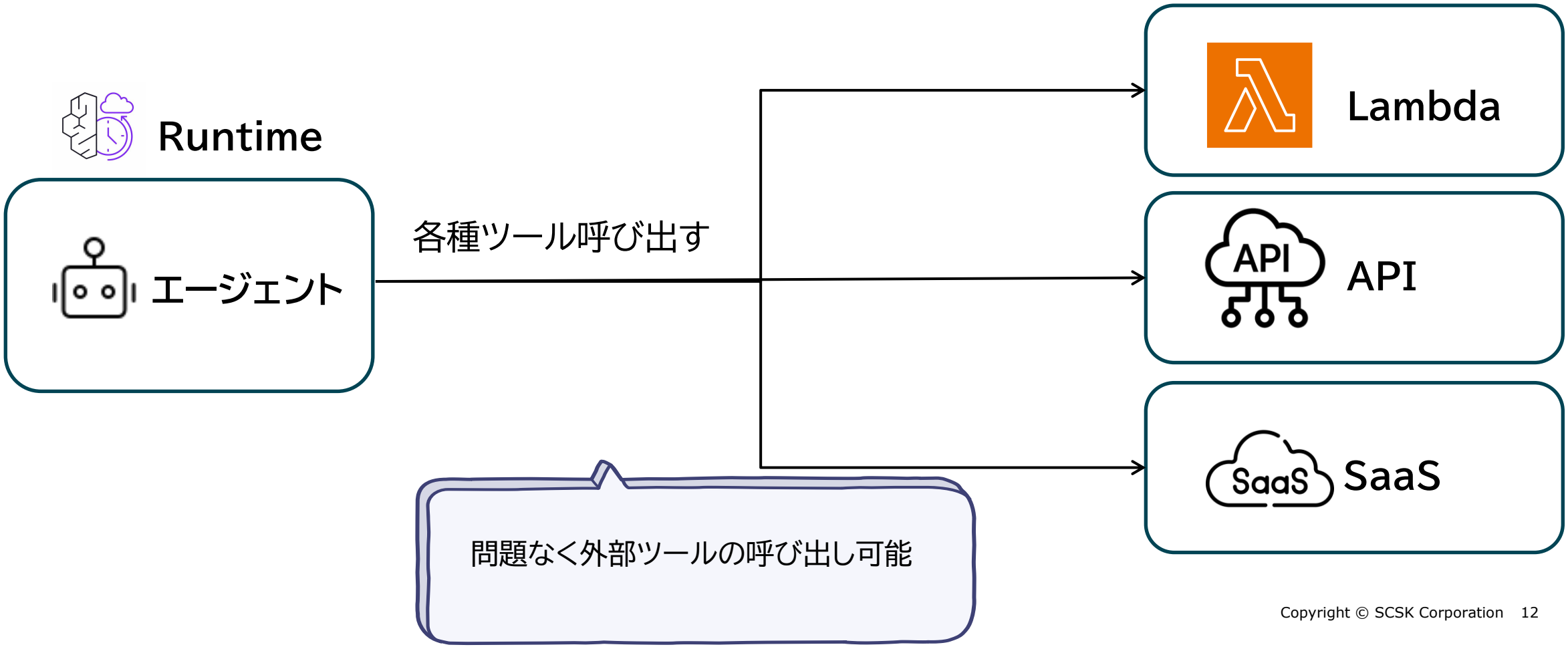


Gatewayを使う



Gatewayを使わない場合

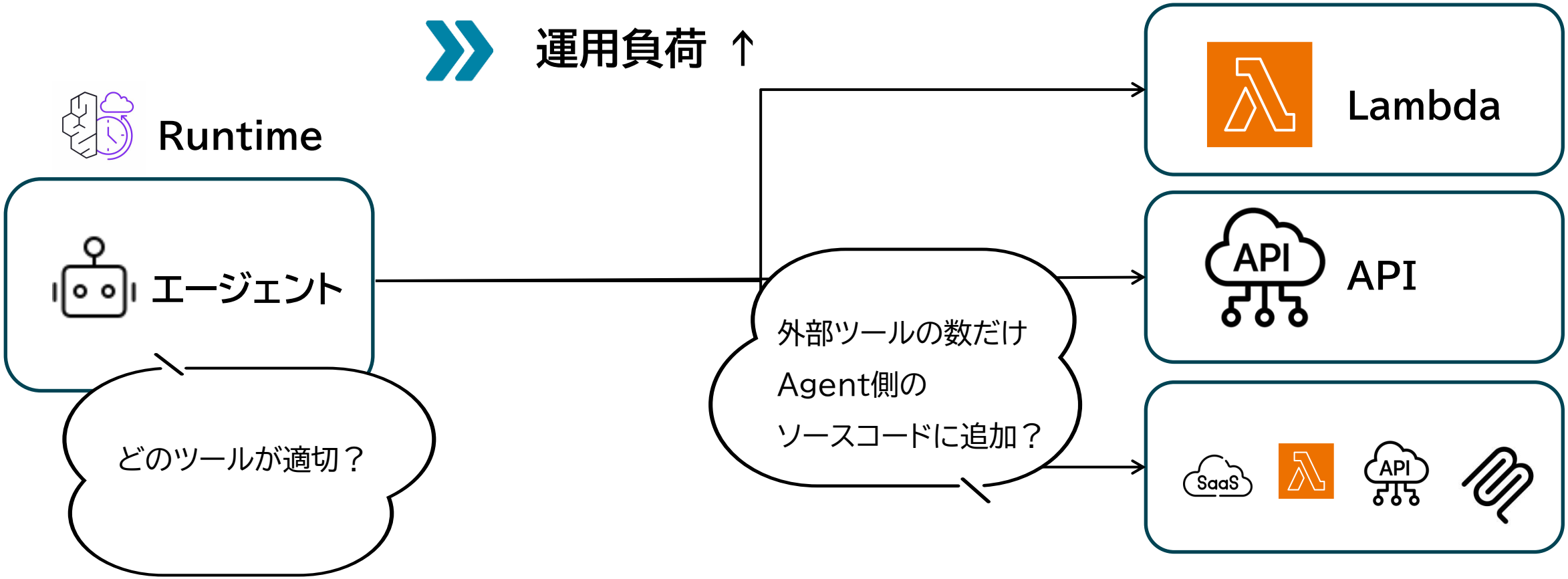
GatewayがなくてもLambdaやSaaSなどの呼び出しは可能



Gatewayを使わない場合

呼び出したい外部ツールの数がどんどん増えていった場合は？

（100個のツールを利用したい）



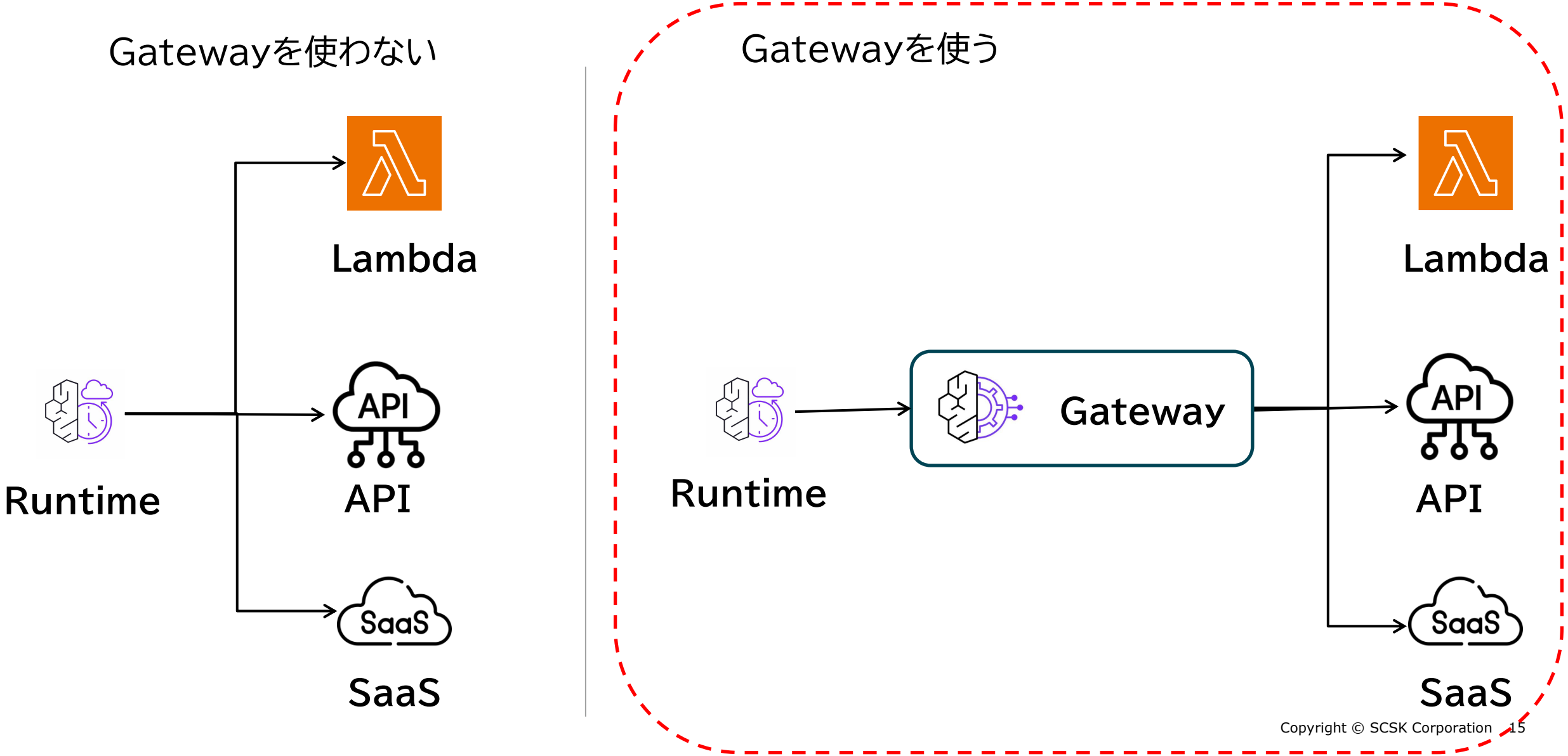
どうやって運用負荷を軽減する？

ツールを増やすたびにAIエージェント側のソースコード修正したくない。。。

エージェントが適切なツール選択できるの？

Gatewayが解決

Gateway有無の違い



Gatewayを使う場合

外部ツールへの接続は必ずGatewayを通す

中央集権的に管理

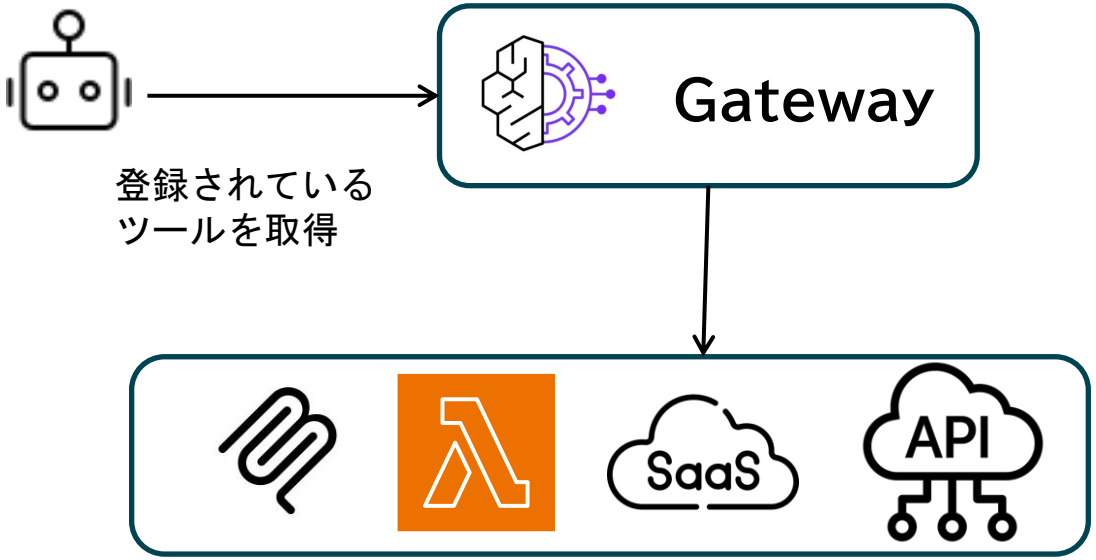
Gatewayに登録されているツールの同期方法

例）GatewayのMCPツールを取り込む

```python
# Gateway を MCP サーバとして接続し、ツール一覧を Agent に渡す

from strands import Agent
from strands.models import BedrockModel
from strands.tools.mcp.mcp_client import MCPClient
from mcp.client.streamable_http import streamablehttp_client

GATEWAY_URL = "<gatewayUrl>"
ACCESS_TOKEN = "<AccessToken>"  # Identity等で取得したBearerを入れる


def transport():
    # Authorization ヘッダ付きで Gateway に接続
    return streamablehttp_client(
        GATEWAY_URL, headers={"Authorization": f"Bearer {ACCESS_TOKEN}"}
    )


model = BedrockModel(
    inference_profile_id="<inference_profile_or_model_id>", temperature=0
)

with MCPClient(transport) as mcp:
    tools = mcp.list_tools_sync()  # Gateway に登録済みツールを取得
    agent = Agent(model=model, tools=tools)
    print(agent("注文状況を調べて").message)
```

Strands Agents

- list_tools_syncメソッド
- Gateway に登録済みツールを取得



Gatewayの作成方法

AgentCore CLI

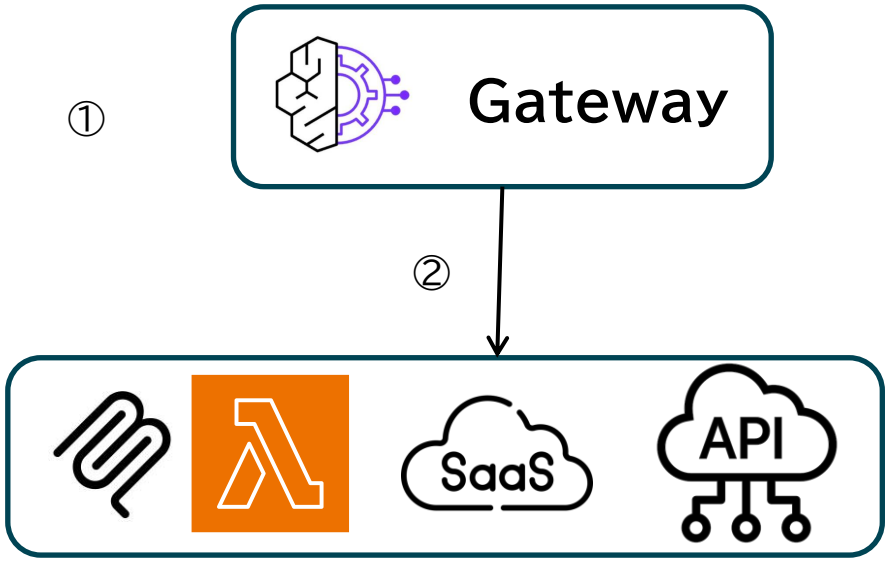
① Gatewayの作成　　`create-mcp-gateway`

② Gateway ターゲットの作成　　`create-mcp-gateway-target`

（Lambda、openApiSchema、mcpServer、smithyModel）

例）Gateway%20/%20Gatewayターゲットの作成

```
# 検索ツールを有効化したGatewayの作成
agentcore gateway create-mcp-gateway \
  --name TestGateway --region ap-northeast-1 --enable_semantic_search

# Lambdaターゲットを作成
agentcore gateway create-mcp-gateway-target \
  --gateway-arn <GATEWAY_ARN> --gateway-url <GATEWAY_URL> \
  --role-arn <ROLE_ARN> --name ToolTarget --target-type lambda --region ap-northeast-1
```



Gatewayまとめ

Gatewayとは

- 外部ツールへの接続ハブ
- 既存のLambda、API、SaasなどをMCP互換のツールに変換するサービス

メリット

- Gatewayに登録するだけで、エージェントからツールを利用可能
- セマンティック検索で効率的にツールを選択可能

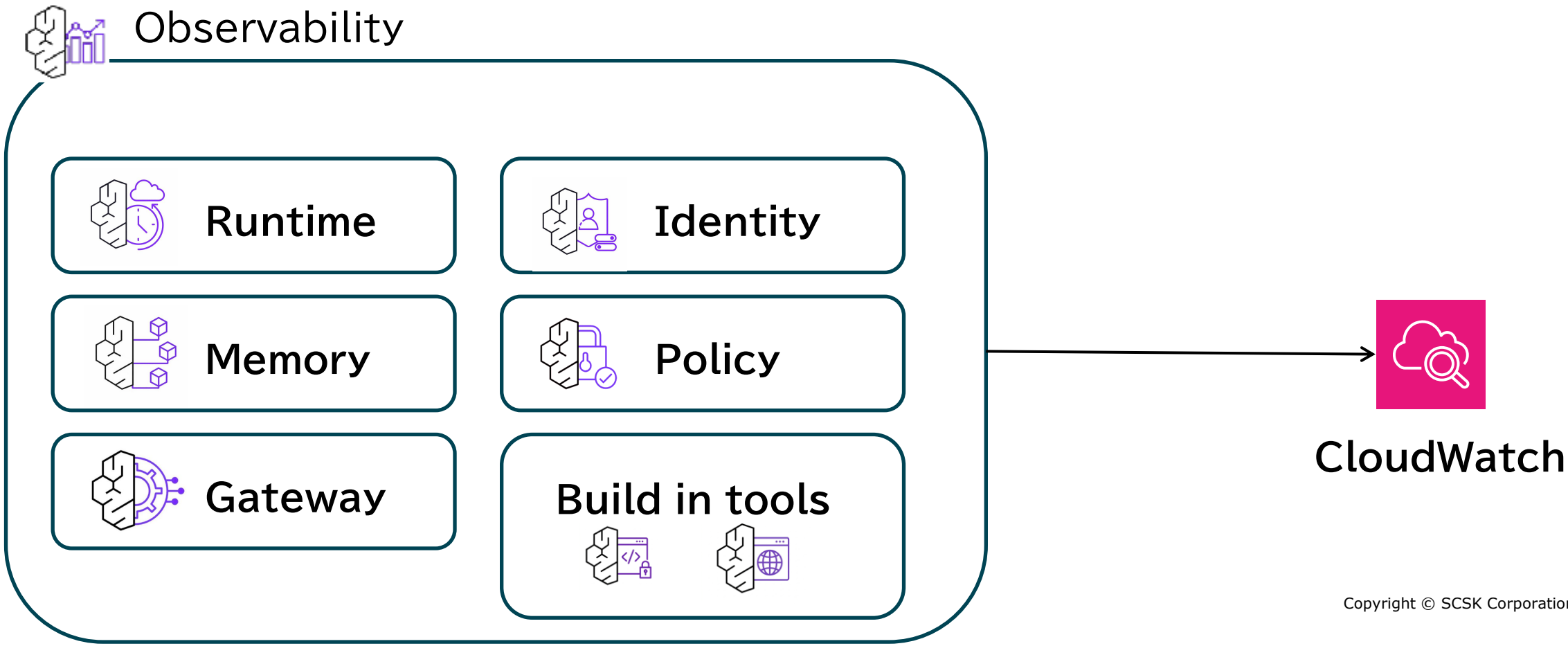
※利用するツールの数が少なければ無理に利用する必要はない

あくまでオプション的な立ち位置ではあるが、ツール数が増えることを見込まれるのであれば導入を検討すべき

Amazon Bedrock AgentCore Observability

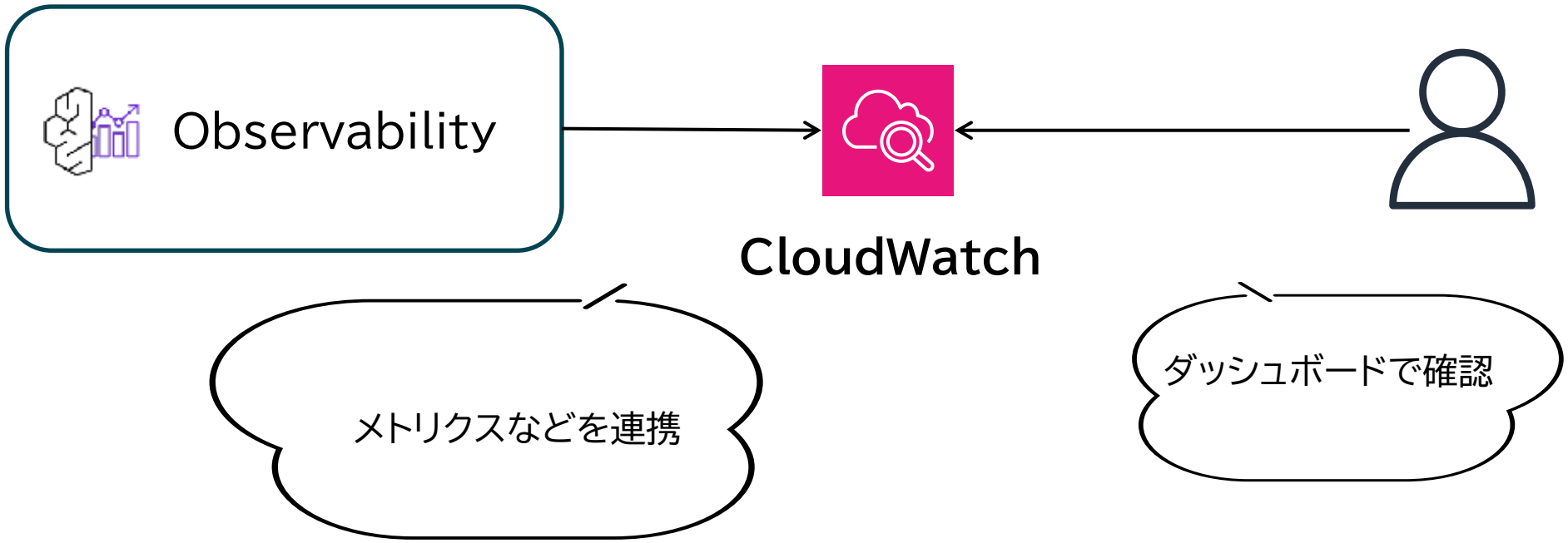
Amazon Bedrock AgentCore Observability

- AIエージェントのパフォーマンスのトレース、デバッグ、監視を支援するサービス
- CloudWatchにメトリクス / スパン / ログを蓄積



Amazon Bedrock AgentCore Observability

- OpenTelemetry（OTEL）互換形式でテレメトリデータを出力
  既存のモニタリングおよびオブザーバビリティスタックと簡単に統合
- CloudWatchに連携されたデータは、ダッシュボード上で確認可能



取得できるテレメトリデータ

1. トレース

   エージェントの実行経路全体

2. スパン

   各処理ステップの詳細情報

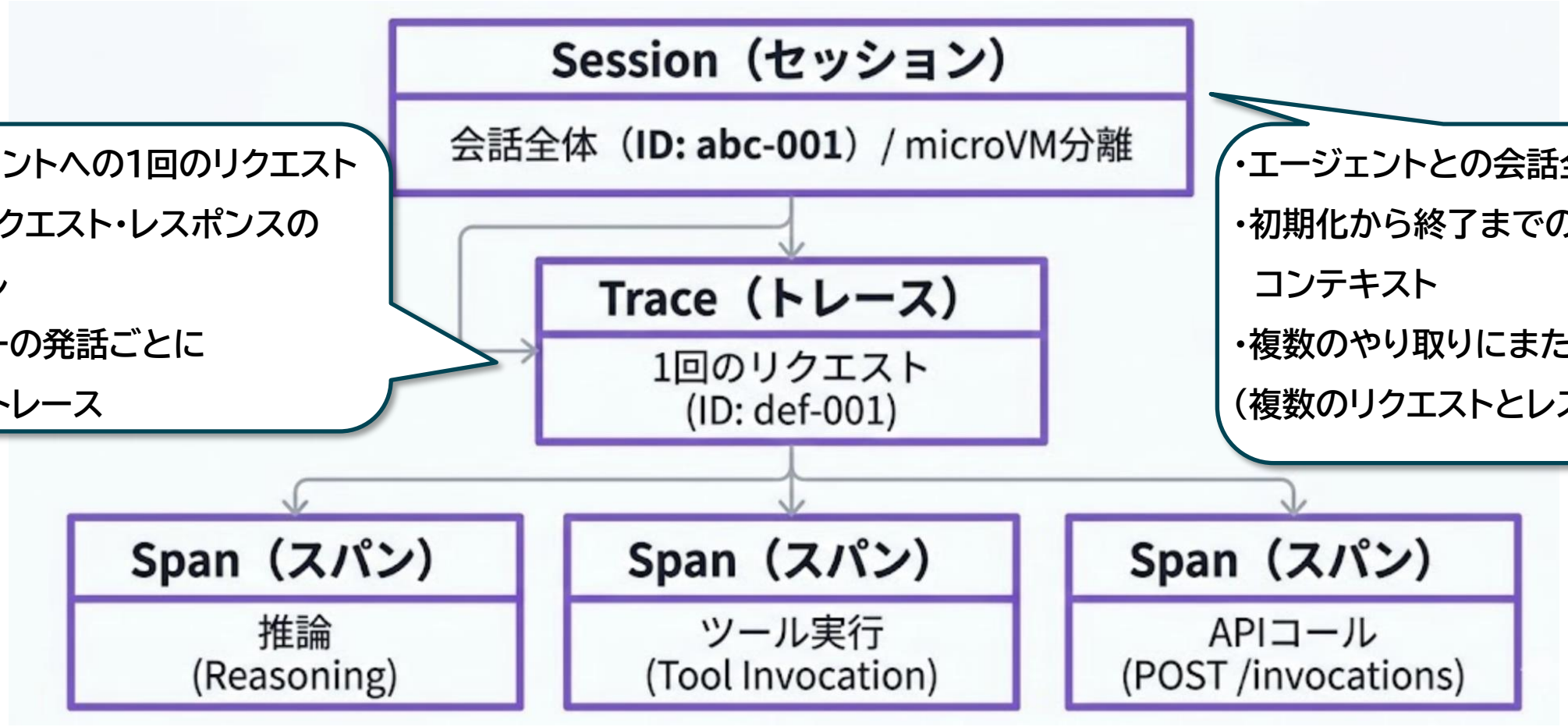
3. メトリクス

   トークン使用量、レイテンシー、エラー率など

4. ログ

   アプリケーションログ、標準出力、エラーログなど

セッション・トレース・スパンの関係



・エージェントへの1回のリクエスト
・1回のリクエスト・レスポンスの
サイクル
・ユーザーの発話ごとに
複数のトレース

・エージェントとの会話全体
・初期化から終了までの完全な
コンテキスト
・複数のやり取りにまたがる
（複数のリクエストとレスポンス）

・トレース内の個別の処理
例）「ユーザー入力の解析」「プロンプトの処理」「LLMの呼び出し」「ツールの実行」な

CloudWatchコンソール(機能)

CloudWatchコンソール(セッション)

CloudWatchコンソール(トレース)

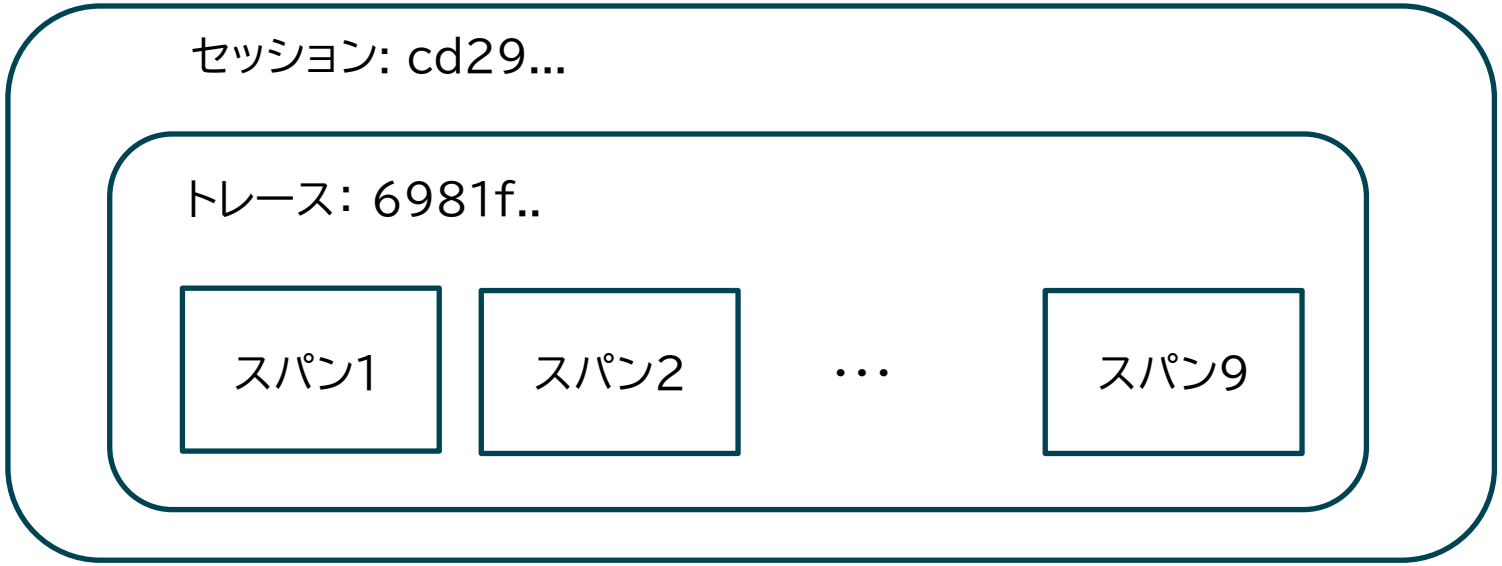
・traceは2つ

・各トレースに9つのスパン

ここまでのセッション・トレース・スパンの図解

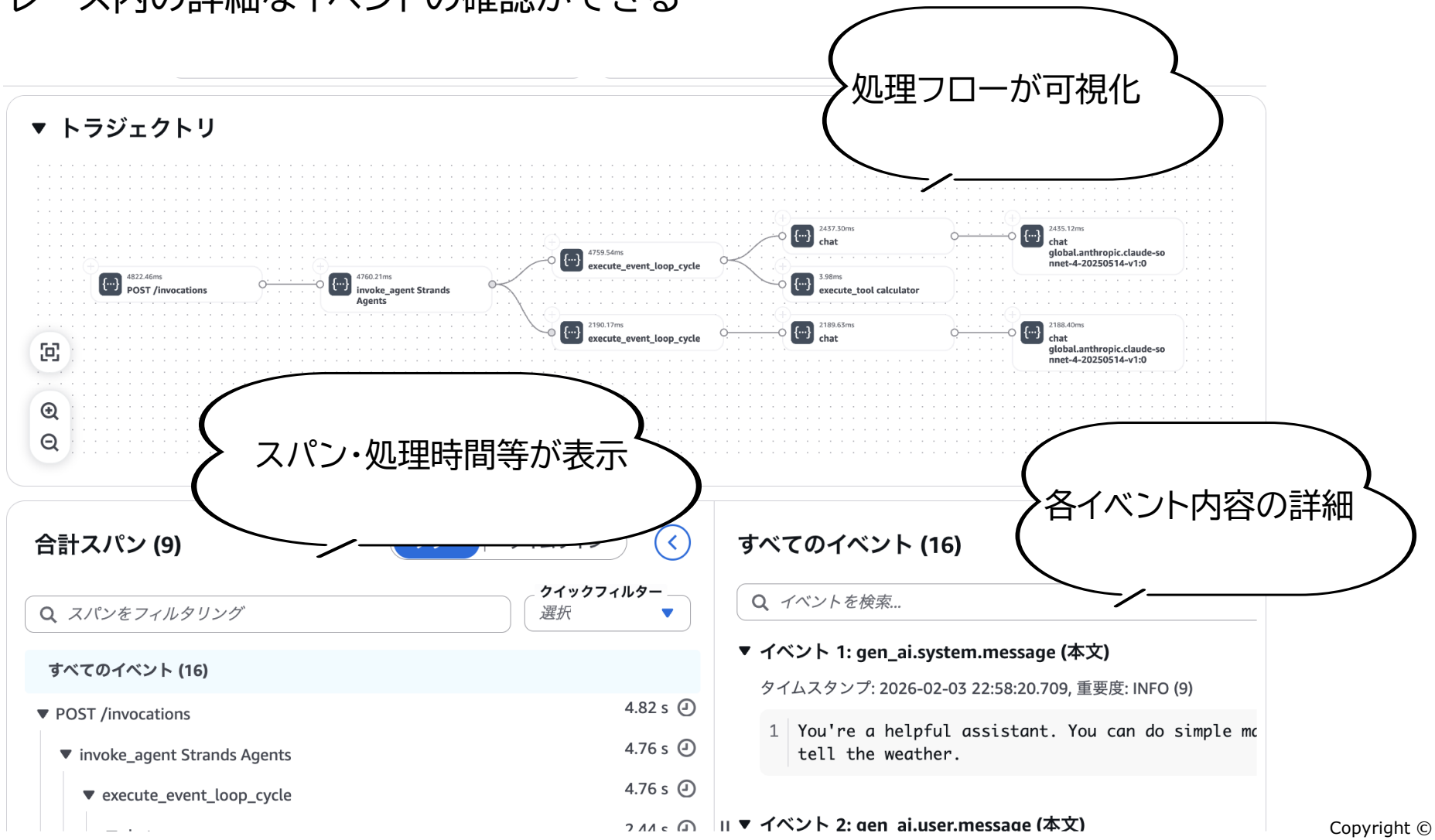
セッション：トレース：スパン

1　　1　　9



CloudWatchコンソール(トレース)

トレース内の詳細なイベントの確認ができる



CloudWatchコンソール(トレース)

トレースの表示方法は2種類

ツリー表示

タイムライン表示

Observabilityの設定

- opentelemetry-instrumentによりエージェントを起動
- 自動で計測される

ローカルでの確認

OpenTelemetry設定

```
# OTel自動計測を差し込んで起動（CloudWatch側にトレース/メトリクスを出す）

export AGENT_OBSERVABILITY_ENABLED=true
export OTEL_PYTHON_DISTRO=aws_distro
export OTEL_PYTHON_CONFIGURATOR=aws_configurator
export OTEL_RESOURCE_ATTRIBUTES=service.name=<YOUR-AGENT-NAME>

opentelemetry-instrument python agent.py
```

DockerFile記載

Dockerfileでの記載

```
CMD ["opentelemetry-instrument", "python" "-m", "strands"]
```


Observabilityまとめ

Observabilityとは

- AIエージェントのパフォーマンスのトレース、デバッグ、監視を支援するサービス
- CloudWatchにメトリクス / スパン / ログを蓄積

メリット

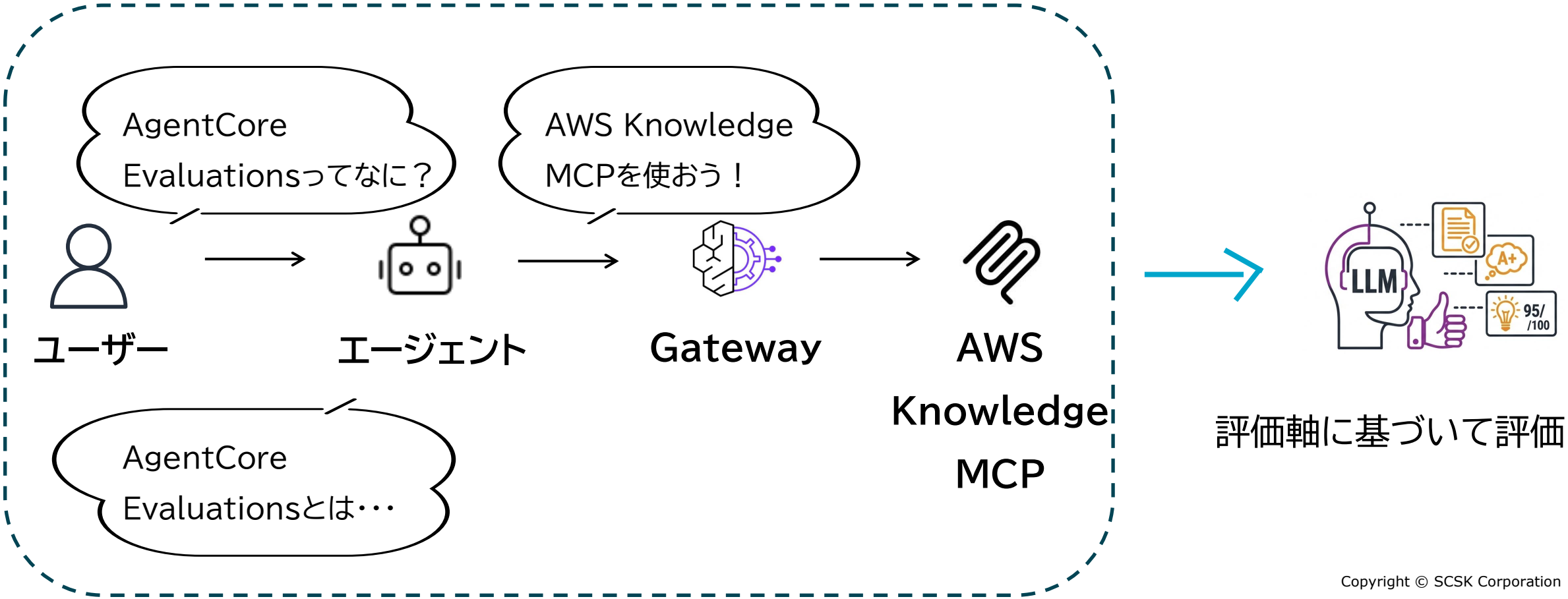
- 処理時間や各トレース・スパンの詳細を確認することができ、
  どこがボトルネックになっているかを確認できる

» ブラックボックス化しやすいエージェントを可視化できる

Amazon Bedrock AgentCore Evaluations

Amazon Bedrock AgentCore Evaluations

- AIエージェントのパフォーマンスを自動測定し
  品質基準への適合を確認するサービス



Amazon Bedrock AgentCore Evaluations

- Strands、LangGraph と OpenTelemetry 経由で統合
- LLM as a Judgeによる エージェントの自動評価
- 評価指標
  - 組み込み（公開）
  - カスタム（非公開）
- 評価モード
  - オンライン評価
  - オンデマンド評価

評価指標

組み込み

- LLMを判定基準として
  エージェントのパフォーマンスを評価
- 事前定義されており
  全ユーザー利用可能
- ユーザー側で設定の変更はできない

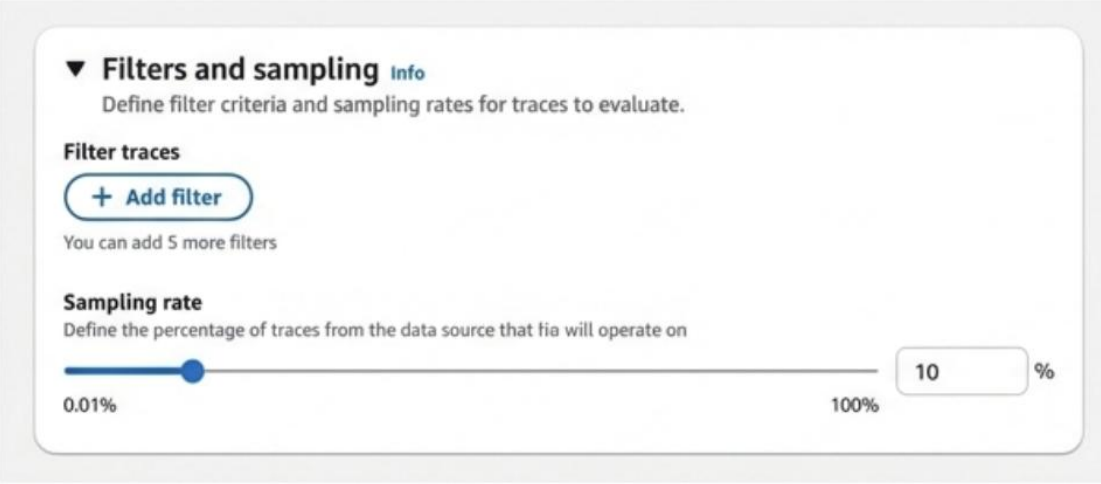
カスタム

- LLMを判定基準と活用しつつ
  評価プロセスを設定可能
- より柔軟な評価を実現
- 下記の場合に役立つ
  ・ドメイン固有のエージェント評価
  ・独自の品質基準を適用
  ・特殊なスコアリングシステム

評価モード

Online

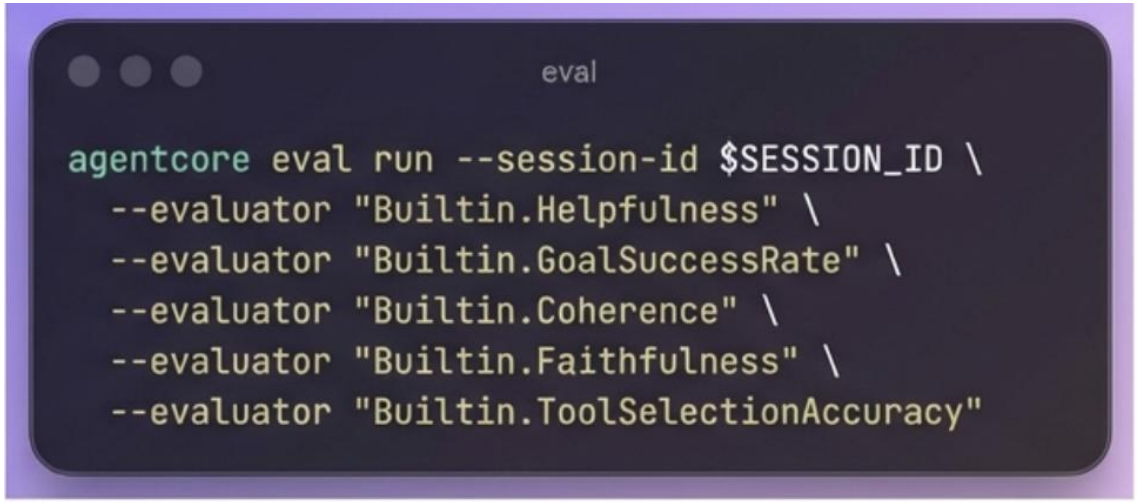
- リアルタイムでエージェント品質を継続的にモニタリング
- サンプリング率を設定
- 結果はCloudWatchに自動連携



例)サンプリング率を10%に設定

On-demand

- 特定のセッションやシナリオを即座に評価
- CLI / SDK(Starter Toolkit)から実行可能
- 開発ループの高速化(修正 → 即評価)



例)CLIから実行

オンライン評価のセットアップ

1 データソースの設定

・エージェントのエンドポイント

・CloudWatch

2 評価指標の選択

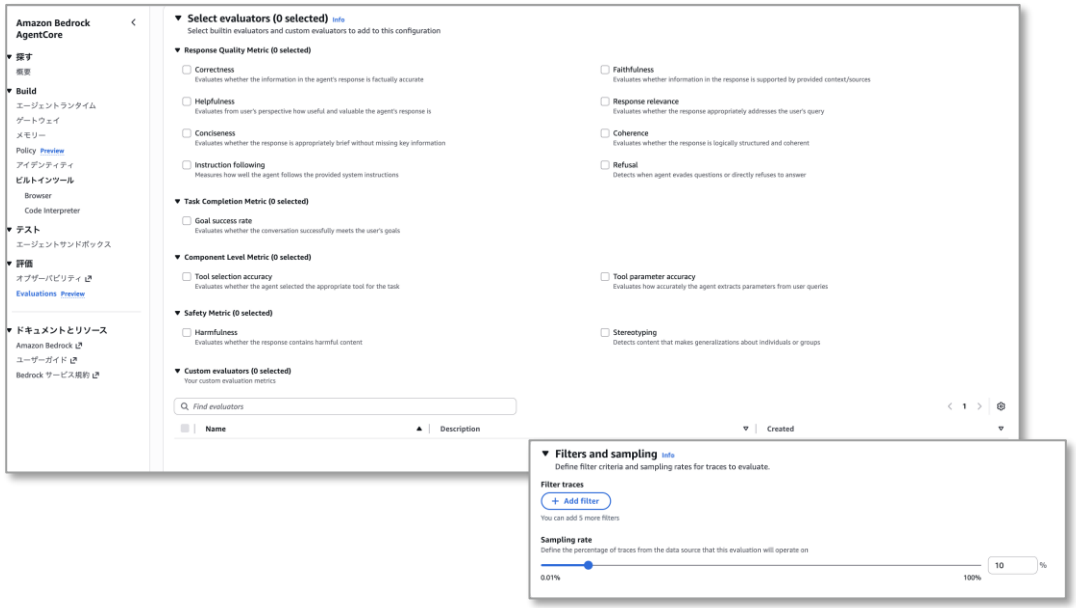
・最大10個までの指標を選択可能

3 サンプリング・フィルターの設定

・評価対象のフィルタリング

・全トラフィックを評価するか、サンプリング率（例: 10%）を指定

※2026年3月5日時点では、ap-northeast-1ではまだ利用できない点に注意



オンデマンド評価の実行

Starter Toolkit の`eval`コマンドでオンデマンド評価

--sessionid：評価対象のセッションID
--evaluator：評価指標

オンデマンド評価

```
agentcore eval run --sessionid "<セッションID>" ¥
    --evaluator "Buildin.Helpfulness" ¥
    --evaluator "Builin.GoalSuccessRate" ¥
    --evaluator "Buildin.Coherence" ¥
    --evaluator "Builtin.Faithfulness" ¥
    --evaluator "Builtin.ToolSelectionAccuracy"
```

コマンド

評価実行結果の一部

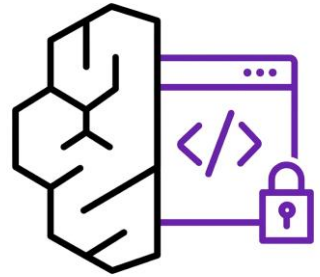
Evaluationsまとめ

Evaluationsとは

- AIエージェントのパフォーマンスを自動測定し
  品質基準への適合を確認するサービス
- 評価指標
  組み込み / カスタム
- 評価モード
  オンライン評価 / オンデマンド評価

「なんとなく動いている」状態から「数値で保証できる」状態へ

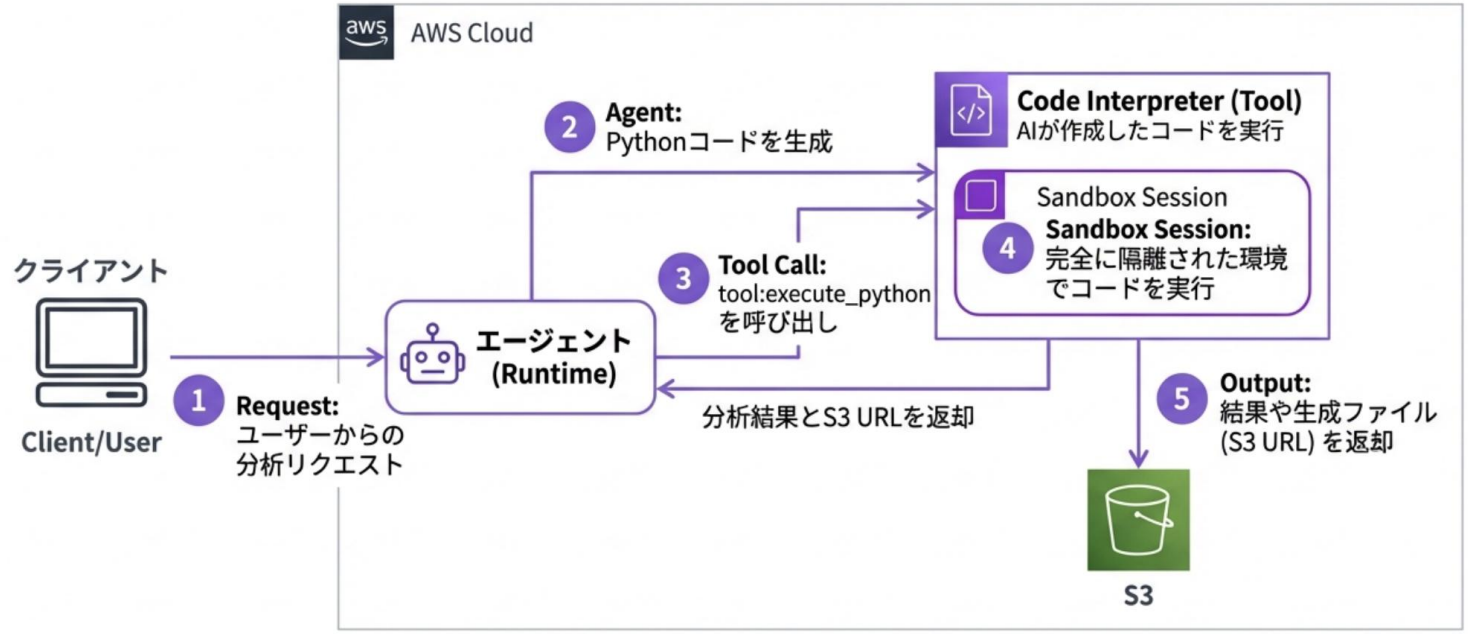
Amazon Bedrock AgentCore Code Interpreter



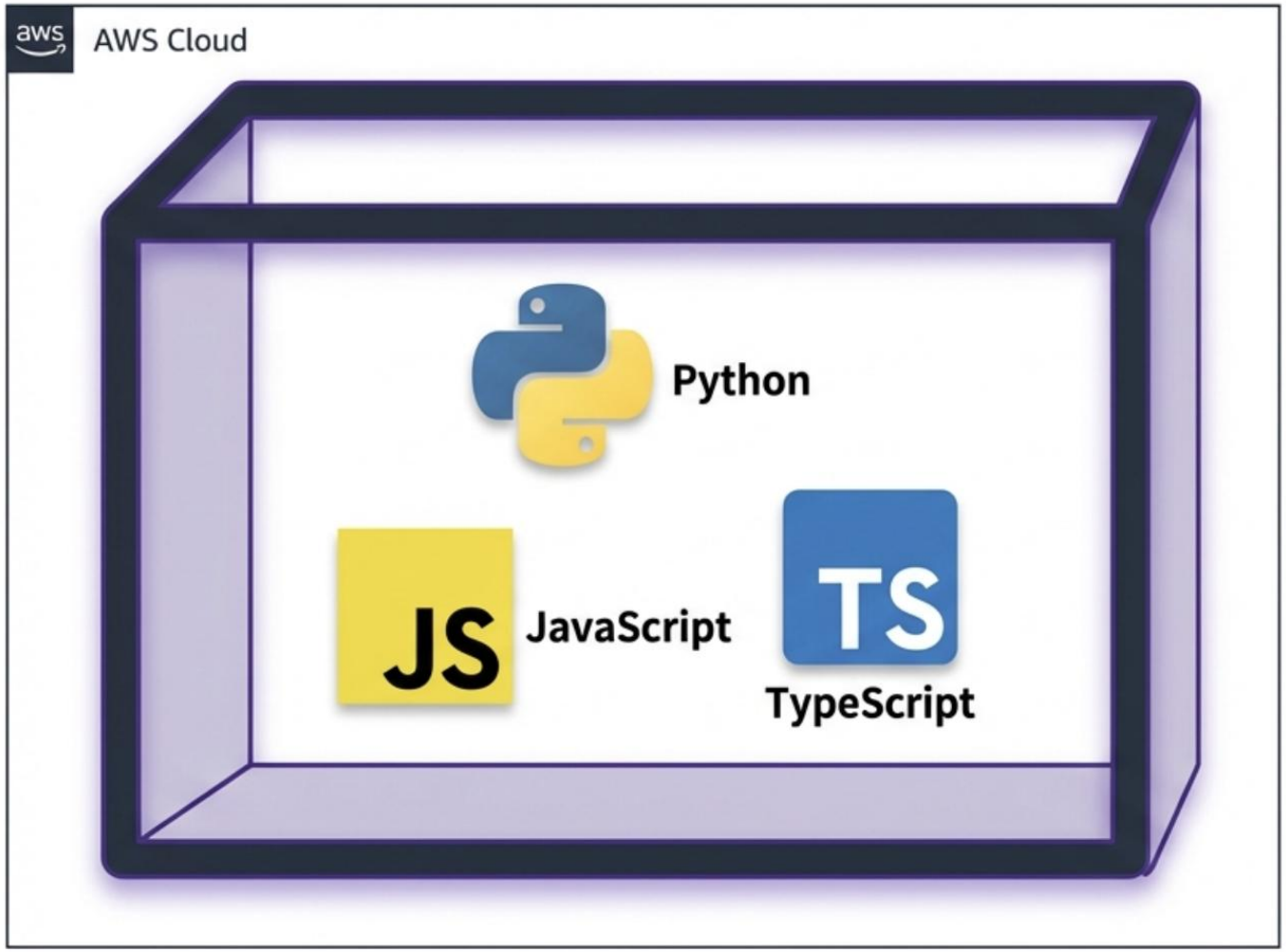
Amazon Bedrock AgentCore Code Interpreter

生成AIが生成したコードを外部の安全な環境で実行するための機能

- 組み込みツール（Built-in Tools）の一つ
- AIが生成したコードを**完全に隔離されたサンドボックスで安全に実行**



対応言語とプリインストールライブラリ



対応言語

- Python
- JavaScript
- TypeScript

ライブラリ

- データサイエンス系

  numpy, pandasなど

ファイル処理能力と実行時間

ファイル 入出力

インラインアップロード: 最大 **100MB**

S3経由: 最大 **5GB**

ギガバイトクラスのデータ処理が可能

実行時間

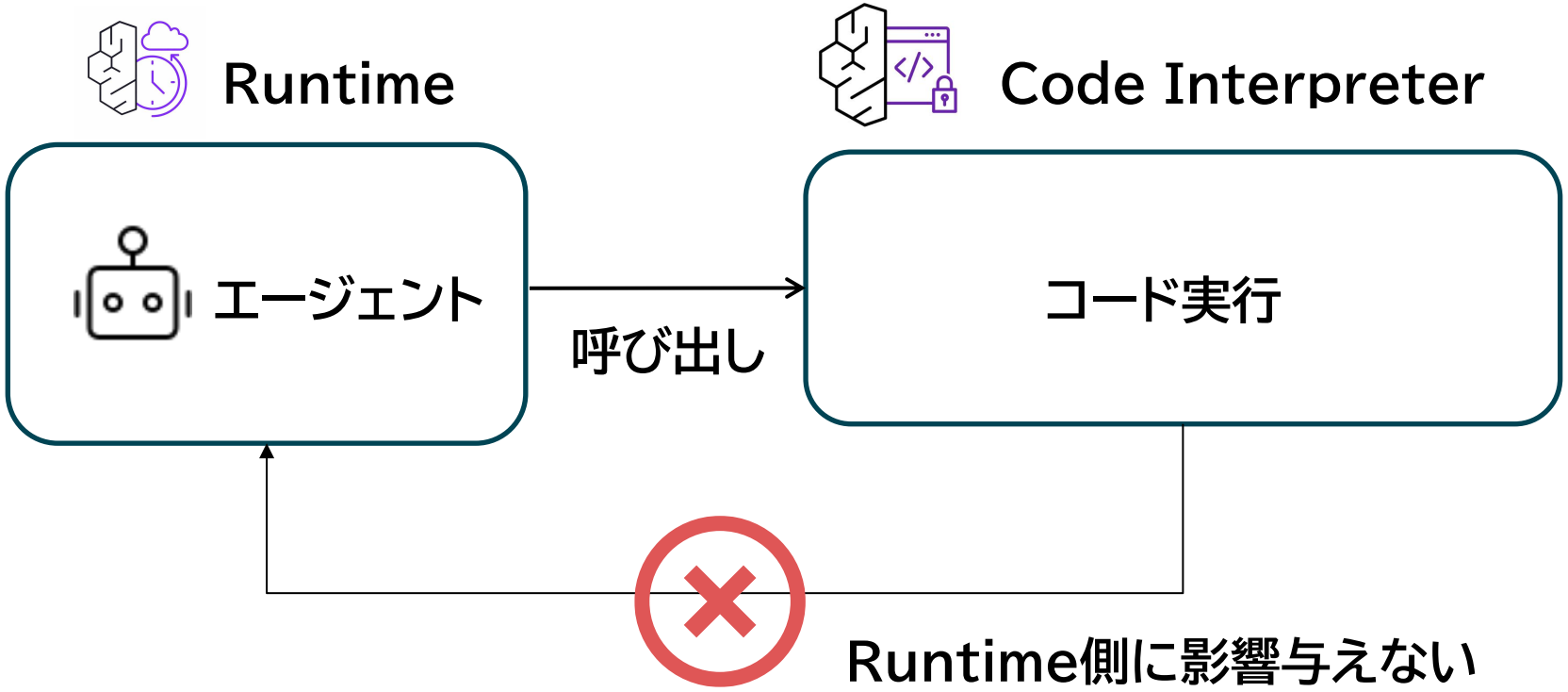
デフォルト: **15分**

最大: **8時間**

非同期の長時間実行に対応

Session State: セッション内でファイルや変数を保持し、対話的な分析が可能

サンドボックスによるセキュリティと分離



- 完全に独立した環境でコード実行
- 実行環境のクラッシュやエラーは、Runtime側に一切影響を与えない

実装例

Import:
AgentCoreCodeInterpreter
をインポートするだけで
利用可能

System Prompt:
自律的なツール利用を
促す指示

Code Interpreter Tool

```python
from strands import Agent
from strands_tools.code_interpreter import AgentCoreCodeInterpreter

# Code Interpreterツールを初期化
code_interpreter_tool = AgentCoreCodeInterpreter(region="us-west-2")

# エージェントのシステムプロンプトを定義
SYSTEM_PROMPT = """あなたはコード実行を通じて回答を検証するAIアシスタントです。
コード、アルゴリズム、計算について質問されたら、Pythonコードを書いて回答を検証してください。"""

# Code Interpreterツールを持つエージェントを作成
agent = Agent(
    tools=[code_interpreter_tool.code_interpreter],
    system_prompt=SYSTEM_PROMPT
)

# サンプルプロンプトでエージェントをテスト
prompt = "フィボナッチ数列の最初の10個を計算してください。"
print(f"\n\nプロンプト: {prompt}\n\n")

response = agent(prompt)
print(response.message["content"][0]["text"])
```

Ease of Use:
数行のコードで実装完了

https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/code-interpreter-using-strands.html

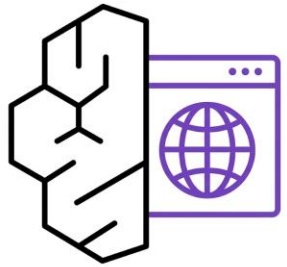
Code Interpreterまとめ

Code Interpreterとは

- 生成AIが生成したコードを外部の安全な環境で実行するための機能
- 対応言語

  Python / JavaScript / TypeScript
- 最大 5GBのファイル処理能力
- 最大 8時間の実行時間

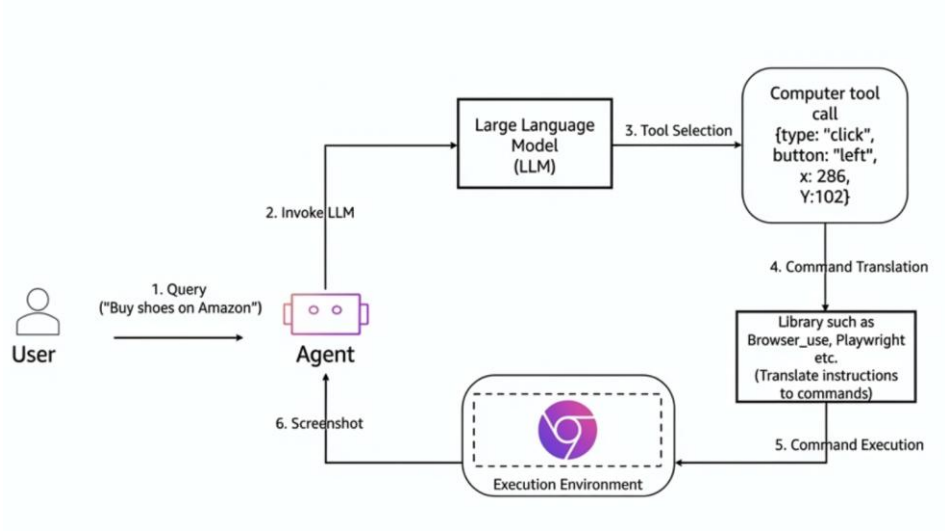
Amazon Bedrock AgentCore Browser



Amazon Bedrock AgentCore Browser

生成AIがWebアプリケーションと対話し、フォーム入力、サイトナビゲーション、
情報抽出を行うための高速でセキュアなクラウドベースの実行環境

- 組み込みツール（Built-in Tools）の一つ
- Playwright や Browser Use などの人気フレームワークに対応
- フォーム入力、ボタンクリック、データ抽出が可能



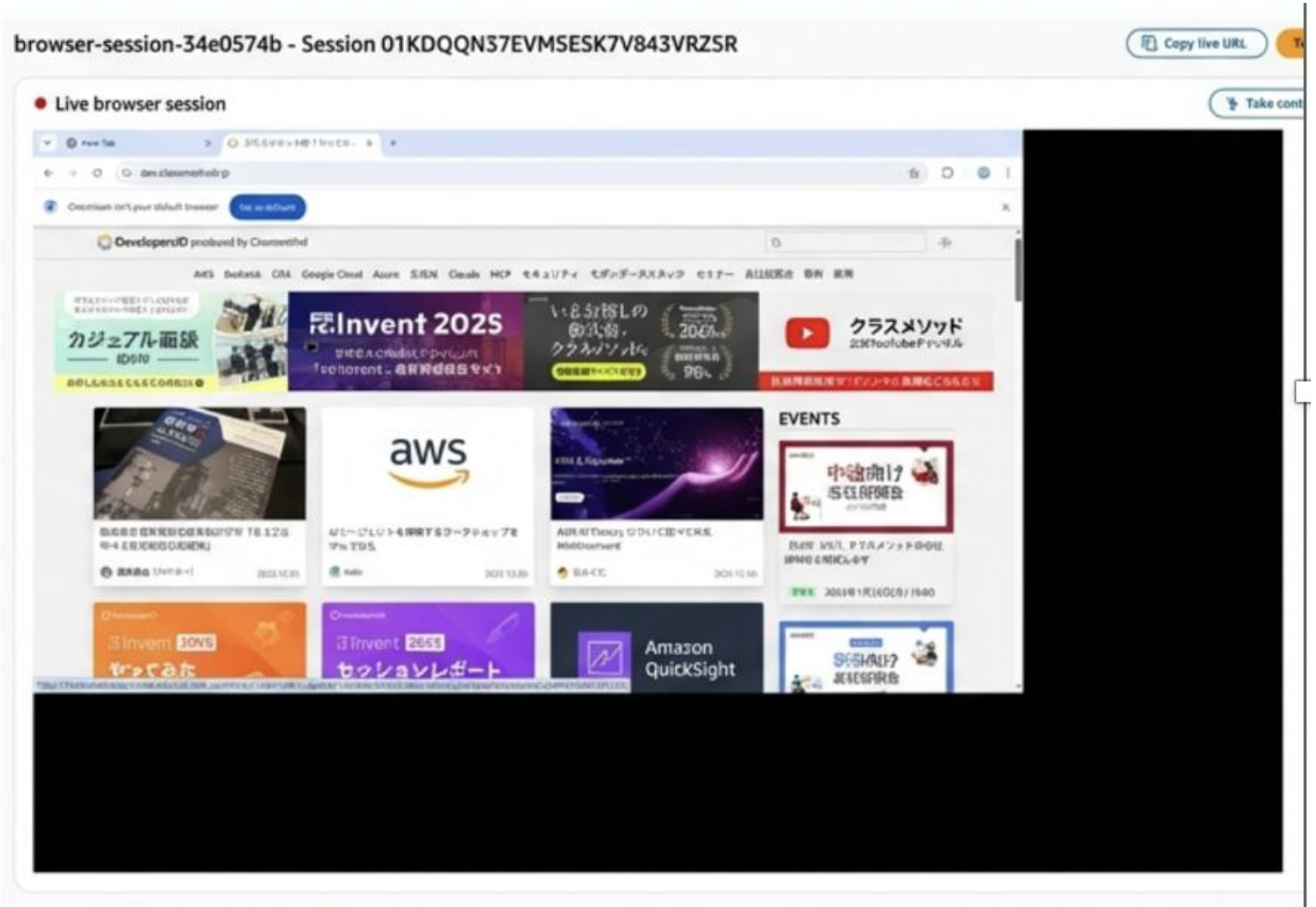
オブサーバビリティ: エージェントの視界を共有

ブラックボックス化しがちなAIの挙動を、リアルタイムの映像やログで可視化

Live View:
ブラウザセッションを
リアルタイムで動画として
視聴可能

Session Recording:
操作ログと映像をS3に自動保存

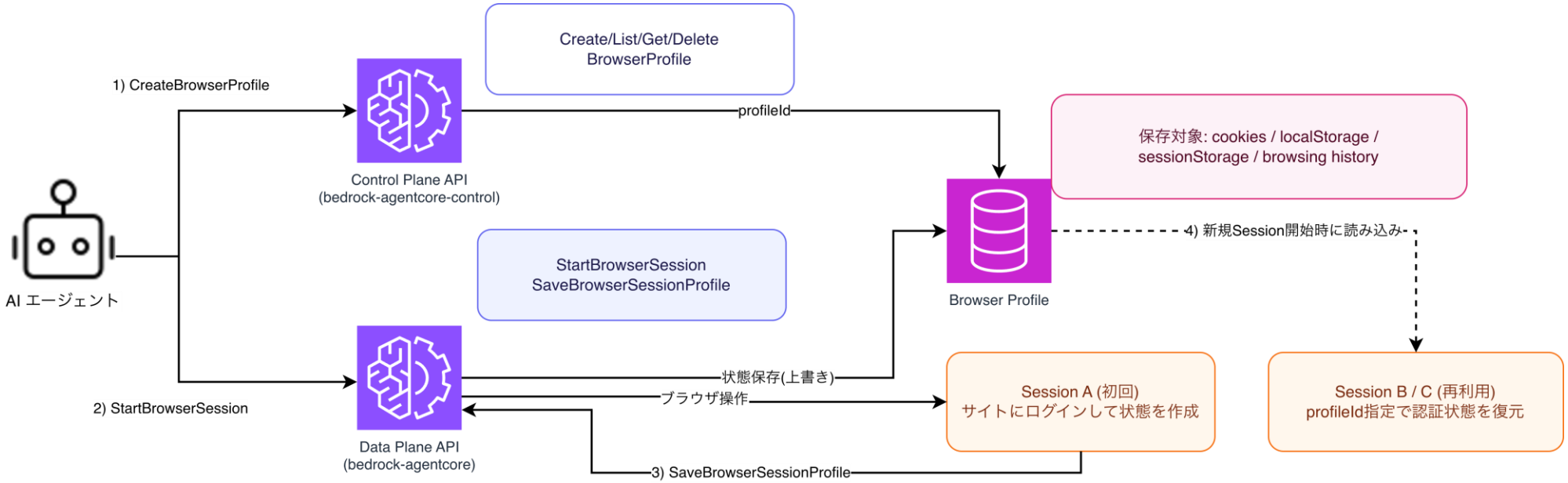
CloudTrail Integration:
API呼び出しの完全な追跡



ブラウザプロファイルによる認証情報の共有

複数のブラウザセッション間で認証状態（ログイン情報やCookieなど）を保存・再利用するための機能

- セッション設定時間の短縮
- 認証の自動化
- 状態の継続性



※ 並列実行は可能だが各セッションは独立 / 保存は明示実行 / Cookie期限切れは引き継げない

実装例

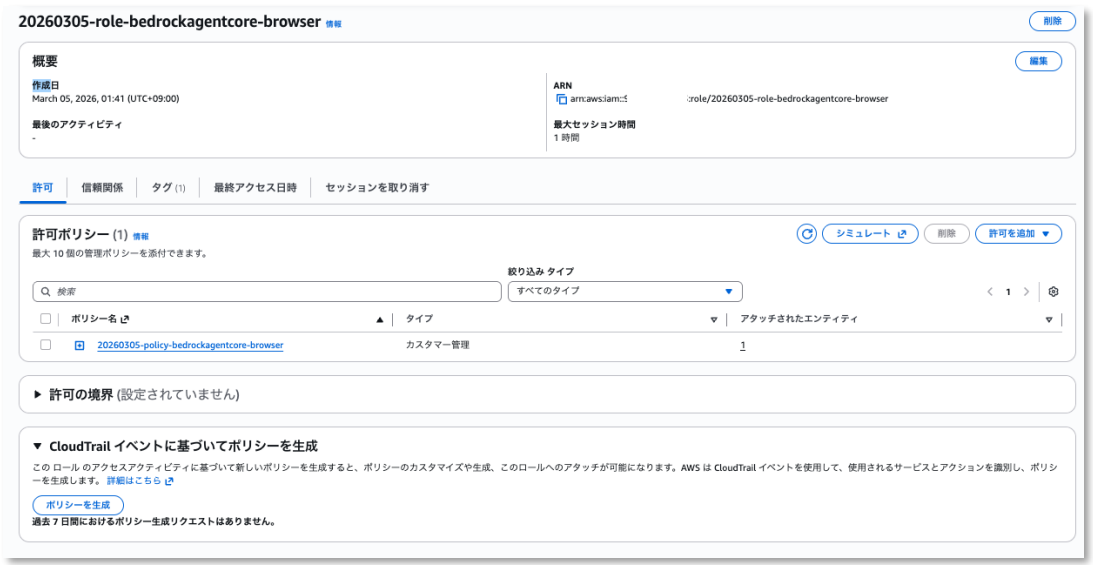
レコーディング保存用のS3バケットの作成

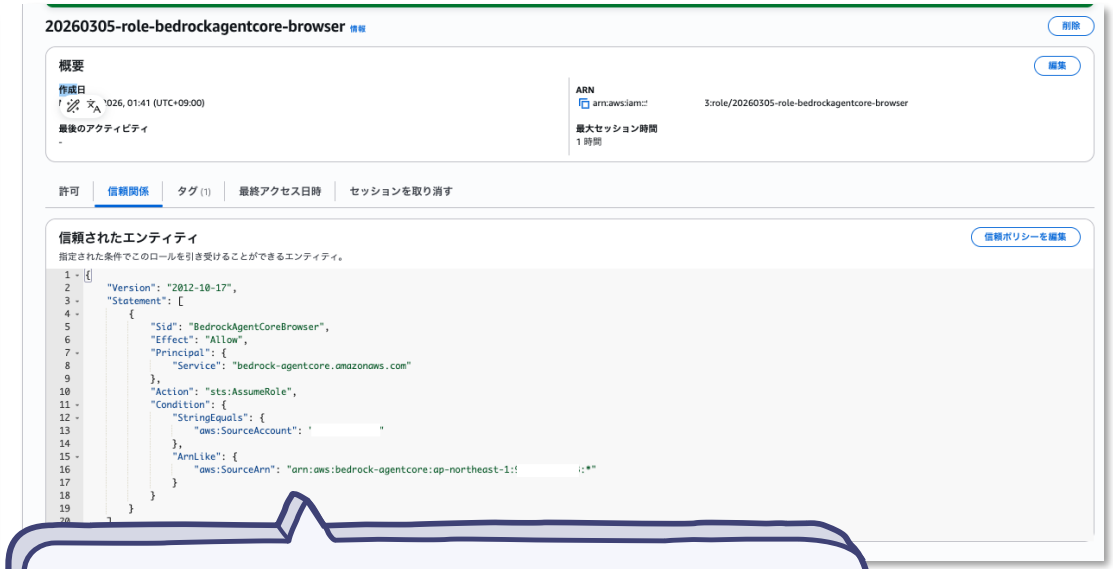
レコーディング保存用のフォルダ作成

実装例

Browserが利用するIAMロールの作成

- レコーディング保存用S3への権限
- CloudWatchの権限

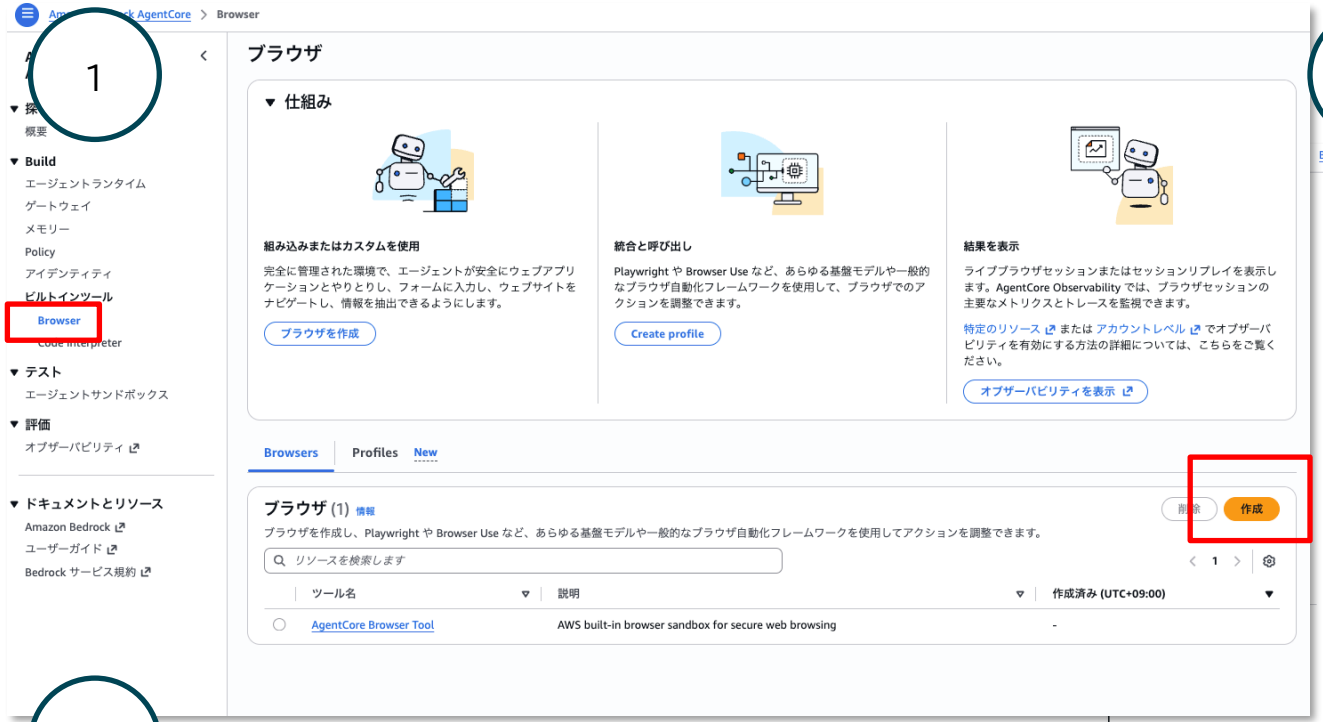


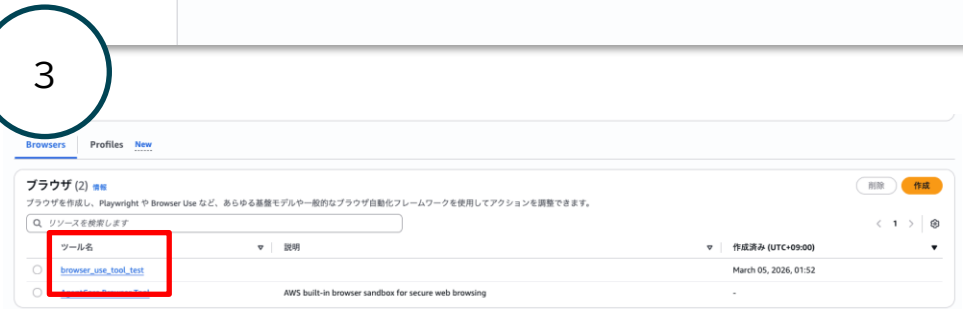


Bedrock AgentCoreに対して
AssumeRole権限

実装例

Browserツールの作成





「記録を有効にする」を選択することで
s3にレコードが連携

実装例

コード例

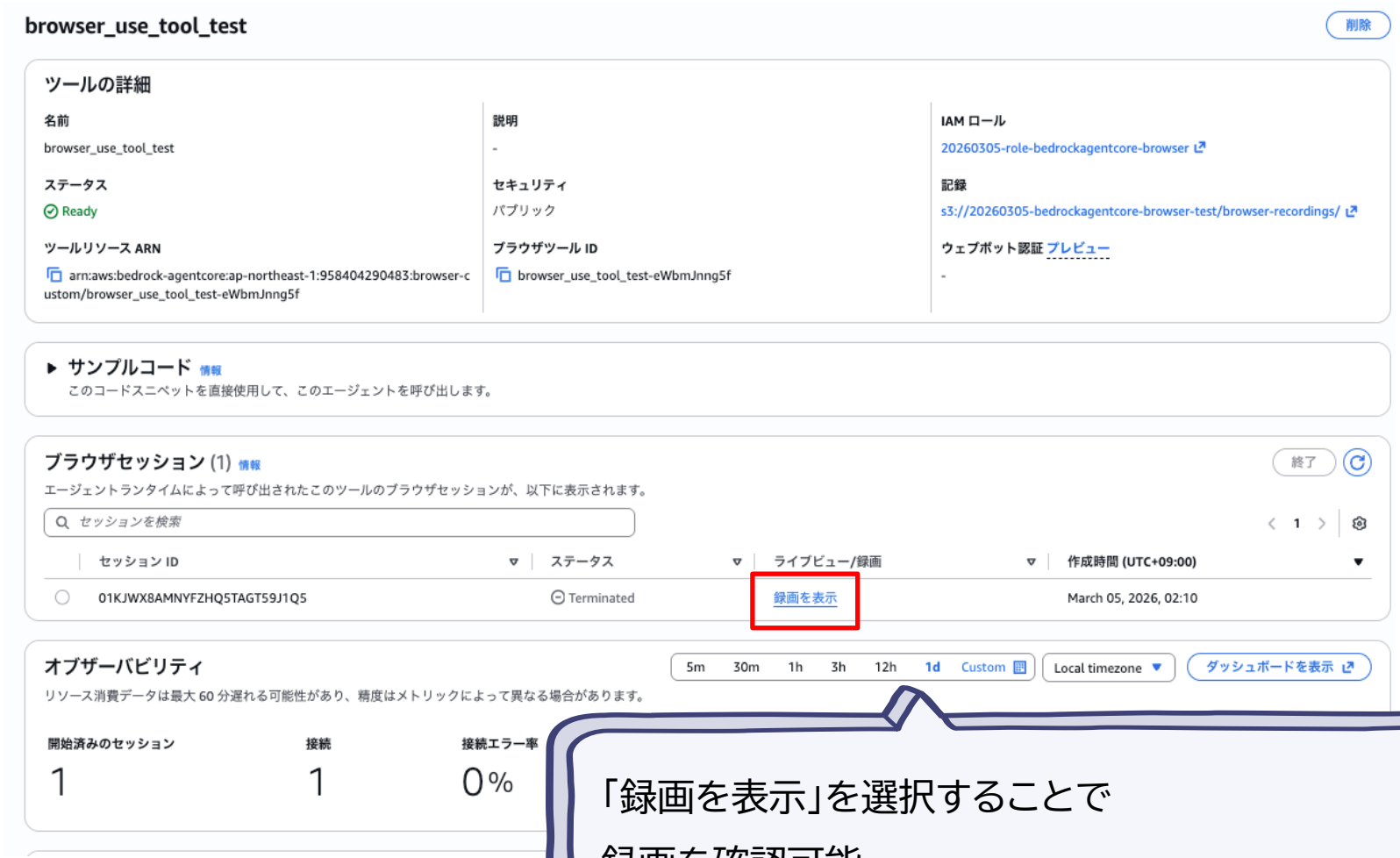
SCSKが取り組んでいる生成AI活用についてホームページから情報を取得してもらう

デモ

```python
from strands.models.gemini import GeminiModel
from strands_tools.browser import AgentCoreBrowser
from bedrock_agentcore.tools.browser_client import Browser Client as AgentCoreBrowserClient
from playwright.async_api import Browser as PlaywrightBrowser
from dotenv import load_dotenv
import os

# .envファイルの読み込み
load_dotenv()

# 環境変数から各種読み込み
# 省略

# Browserラッパークラスの定義(LiveViewAgentBrowser)
# 省略

# AgentCore Browserツールを初期化
browser_tool = LiveViewAgentCoreBrowser(region=BROWSER_REGION, identifier=BROWSER_IDENTIFIER)
gemini_model = GeminiModel(
    model_id=GOOGLE_API_MODEL
)
# Browser ツールとモデルを指定してエージェントを作成
agent = Agent(
    model=gemini_model,
    tools=[browser_tool.browser]
)

# シナリオに沿ったプロンプト
prompt = f"""
https://www.scsk.jp/sp/technology_strategy/ai.html  を読んで、SCSKが取り組んでいる生成AIの活用について
まとめてください"
"""

# エージェントを実行
response = agent(prompt)
```


実装例

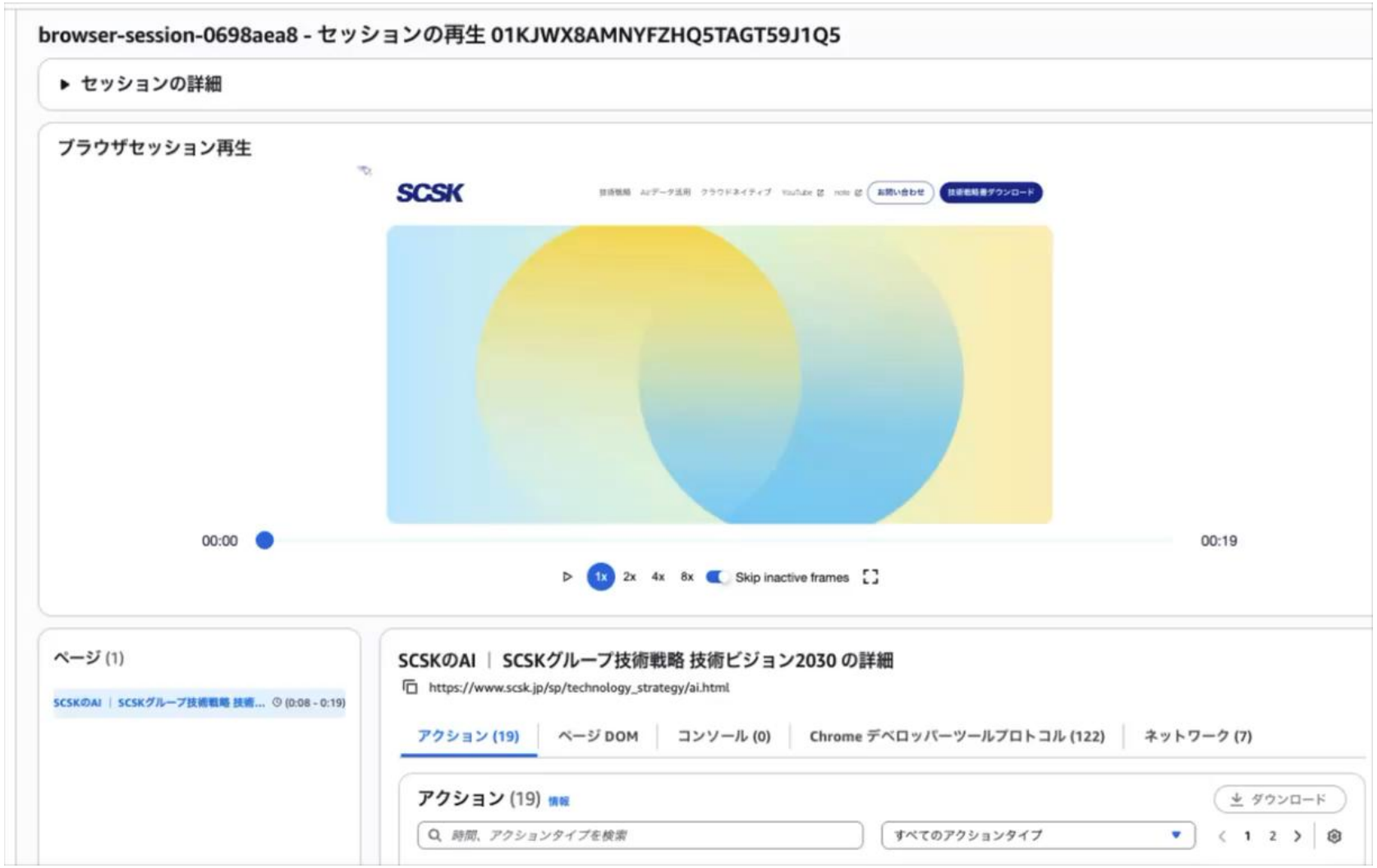
録画が作成されている



「録画を表示」を選択することで
録画を確認可能
※リアルタイムでの確認の場合は「Live View」と表示

実装例

録画の確認



Browserまとめ

Browserとは

- 生成AIがWebアプリケーションと対話し、フォーム入力、サイトナビゲーション、情報抽出を行うための高速でセキュアなクラウドベースの実行環境
- ブラウザプロファイルによって認証情報の共有が可能
- Live View / 録画機能により、エージェントのブラウザ操作を確認できる
- Strands SDKで簡単に実装できる

まとめ

各種サービスのまとめ（１／２）

Gatewayとは

- 外部ツールへの接続ハブ
- 既存のLambda、API、SaasなどをMCP互換のツールに変換するサービス

Observabilityとは

- AIエージェントのパフォーマンスのトレース、デバッグ、監視を支援するサービス
- CloudWatchにメトリクス / スパン / ログを蓄積

各種サービスのまとめ（２／２）

Evaluationsとは

- AIエージェントのパフォーマンスを自動測定し、品質基準への適合を確認するサービス

Code Interpreterとは

- 生成AIが生成したコードを外部の安全な環境で実行するための機能

Browserとは

- 生成AIがWebアプリケーションと対話し、フォーム入力、サイトナビゲーション、情報抽出を行うための高速でセキュアなクラウドベースの実行環境

SCSK

夢ある未来を、共に創る。

オンデマンド評価の実行

```
(strands-agentcore-demo) ao.noguchi@noguchimidoriseinoMacBook-Air strands_agentcore_demo % uv run agentcore eval run --session-id $SESS
ION_ID \
  --evaluator "Builtin.Helpfulness" \
  --evaluator "Builtin.GoalSuccessRate" \
  --evaluator "Builtin.Coherence" \
  --evaluator "Builtin.Faithfulness" \
  --evaluator "Builtin.ToolSelectionAccuracy"

/Users/ao.noguchi/Developer/strands_agentcore_demo/.venv/lib/python3.13/site-packages/requests/__init__.py:113: RequestsDependencyWarni
ng: urllib3 (2.6.3) or chardet (6.0.0.post1)/charset_normalizer (3.4.4) doesn't match a supported version!
  warnings.warn(

Evaluating session: 67E4D4F5-621F-4CC2-9C8E-F60F68482DE1
Mode: All traces (most recent 1000 spans)
Evaluators: Builtin.Helpfulness, Builtin.GoalSuccessRate, Builtin.Coherence, Builtin.Faithfulness, Builtin.ToolSelectionAccuracy

  Evaluation Results
  Session: 67E4D4F5-621F-4CC2-9C8E-F60F68482DE1
```


オンデマンド評価の実行

✓ Successful Evaluations

Evaluator: Builtin.GoalSuccessRate

Score: 0.00
Label: No

Explanation:
The user's goal was to get weather information for the Imperial Palace (皇居), which is located in Tokyo. To achieve this goal, the AI assistant should: 1) Use the get_weather tool with Tokyo as the location, 2) Retrieve the weather information, and 3) Provide the weather details to the user.

Analyzing the conversation:
- The assistant correctly identified that 皇居 (Imperial Palace) is in Tokyo
- The assistant appropriately called the get_weather tool with '東京' (Tokyo) as the parameter
- However, the tool returned: 'Weather information for 東京 is not available'
- The assistant acknowledged the failure and explained that weather information could not be retrieved
- The assistant provided alternative suggestions for obtaining weather information

While the assistant took the correct approach and used the appropriate tool, the tool failed to return the requested weather information. The user's goal was to know the weather at the Imperial Palace, but this information was not provided. Despite the assistant's helpful response with alternative suggestions, the core objective - obtaining and providing the weather information - was not achieved. The failure was due to the tool's unavailability rather than the assistant's approach, but the end result is that the user did not receive the weather information they requested.

Token Usage:
- Input: 1,099
- Output: 307
- Total: 1,406

Evaluated:
- Session: 67E4D4F5-621F-4CC2-9C8E-F60F68482DE1

Evaluator: Builtin.Helpfulness

Score: 0.33
Label: Somewhat Unhelpful

Explanation:
The user's goal is clear: they want to know the weather at the Imperial Palace (皇居) in Tokyo. The assistant attempted to retrieve this information using a weather tool, but the tool failed to return weather data.

The assistant's response appropriately acknowledges the failure and apologizes for not being able to provide the requested information. It explains that the weather service is unavailable, which is transparent and honest.

However, from the user's perspective, this response does not move them closer to their goal at all. The user still doesn't know the weather at 皇居. While the assistant provides alternative suggestions (Japan Meteorological Agency website, weather apps, news sites), these are generic recommendations that require the user to take additional steps outside this conversation to achieve their goal.

The response is polite and professional, and the suggestions are reasonable alternatives. However, it fundamentally fails to deliver what the user asked for. The assistant doesn't provide any actual weather information, doesn't offer to try alternative methods within the conversation, and essentially tells the user to solve the problem themselves.

This is a case where the response is courteous and explains the situation well, but provides no progress toward the user's actual goal of learning the weather.

Token Usage:
- Input: 1,098
- Output: 297
- Total: 1,395

Evaluated:
- Session: 67E4D4F5-621F-4CC2-9C8E-F60F68482DE1
- Trace: 69a59ae81e72fb5254b653e571751427

オンデマンド評価の実行

```
Evaluator: Builtin.Coherence

Score: 1.00
Label: Completely Yes

Explanation:
The assistant's response demonstrates sound logical cohesion. The user asked about the weather at the Imperial Palace (皇居). The
assistant correctl
Tokyo using the ge
logically conclude
('申し訳ございませ
and provides alter
assistant consiste
failure to acknowl
clear reasoning. T

Token Usage:
  - Input: 956
  - Output: 243
  - Total: 1,199

Evaluated:
  - Session: 67E4D
  - Trace: 69a59ae
```

```
Evaluator: Builtin.Faithfulness

Score: 1.00
Label: Completely Yes

Explanation:
Let me analyze the assistant's response against the conversation history:

1. User asked: "皇居の天気をおしえて" (Tell me the weather at the Imperial Palace)

2. The assistant made a tool call to get_weather with parameter city='東京' (Tokyo), which is appropriate since the Imperial
Palace is located in Tokyo.

3. The tool returned: "Weather information for 東京 is not available" - indicating the weather service failed to provide
information.

4. The assistant's response states:
   - Apologizes that Tokyo weather information could not be retrieved
   - Explains the weather information service appears unavailable
   - Suggests alternative methods to check weather (Japan Meteorological Agency website, weather apps, news sites)
   - Recommends trying again later

The assistant's response accurately reflects what happened in the conversation:
- It correctly acknowledges the tool call failure
- It accurately reports that Tokyo weather information was unavailable
- It provides helpful alternatives
- It does not fabricate any weather information

There are no conflicts between the assistant's response and the conversation history. The assistant faithfully represents the
tool's output and provides appropriate guidance.

Token Usage:
  - Input: 953
  - Output: 300
  - Total: 1,253

Evaluated:
  - Session: 67E4D4F5-621F-4CC2-9C8E-F60F68482DE1
  - Trace: 69a59ae81e72fb5254b653e571751427
```

オンデマンド評価の実行

```
Evaluator: Builtin.ToolSelectionAccuracy

Score: 1.00
Label: Yes

Explanation:
The user asked '皇居の天気をおしえて' which translates to 'Tell me the weather at the Imperial Palace.' The Imperial Palace
(皇居) is located in Tokyo, Japan. The assistant called the get_weather tool with the parameter 'city': '東京' (Tokyo), which is
a reasonable interpretation of the user's request.

Analyzing the justification:
1. The action directly addresses the user's request for weather information about a specific location.
2. The action is well-aligned with the user's intent - they want weather information, and the assistant is attempting to retrieve
it.
3. The assistant correctly identified that the Imperial Palace is in Tokyo and used '東京' as the city parameter, which is the
minimum necessary parameter to make a useful weather query.
4. A helpful assistant would reasonably take this action to serve the user's needs.

The fact that the tool result returned 'Weather information for 東京 is not available' is a limitation of the tool itself, not a
problem with the decision to call it. The assistant made the right choice in attempting to retrieve weather information for Tokyo
when asked about the Imperial Palace's weather. The action was justified and appropriate given the user's request.

Token Usage:
  - Input: 821
  - Output: 295
  - Total: 1,116

Evaluated:
  - Session: 67E4D4F5-621F-4CC2-9C8E-F60F68482DE1
  - Trace: 69a59ae81e72fb5254b653e571751427
  - Span: abed480dac4f191b
```


Browser IAM Policy

IAMポリシー_for_Browser

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ListBucketForRecordingPrefix",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::20260305-bedrockagentcore-browser-test",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "browser-recordings",
                        "browser-recordings/",
                        "browser-recordings/*"
                    ]
                }
            }
        },
        {
            "Sid": "S3ObjectAccessForRecordingPrefix",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "arn:aws:s3:::20260305-bedrockagentcore-browser-test/browser-
recordings/*"
        },
        {
            "Sid": "CloudWatchCreateLogGroupForAgentCoreBrowser",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup"
            ],
            "Resource": "arn:aws:logs:ap-northeast-1:<ACCOUNT_ID>:log-group:/aws/bedrock-
agentcore/browser/*"
        },
        {
            "Sid": "CloudWatchWriteLogsForAgentCoreBrowser",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:ap-northeast-1:<ACCOUNT_ID>:log-group:/aws/bedrock-
agentcore/browser/*",
                "arn:aws:logs:ap-northeast-1:<ACCOUNT_ID>:log-group:/aws/bedrock-
agentcore/browser/*:log-stream:*"
            ]
        }
    ]
}
```


Observabilityのテレメトリデータをサードパーティに連携する方法

- ObservabilityはOTEL互換形式で出力される
- OTELフォーマットのデータ取り込みに対応している任意の監視・オブザーバビリティスタックと簡単に統合可能

  Datadog、Elasticなど

サードパーティ製品へ連携する際の設定方法

エージェントの実行環境で以下の環境変数を設定