大規模ゲーム開発を支えるデータ分析・可視化基盤の構築

Google Cloud Next Tokyo

高本 真宏

株式会社 カプコン
CS 第二開発統括
システム基盤部
ソリューション開発室

2018 年サーバー エンジニアとして、株式会社カプコン入社。

各種タイトルのサーバー エンジニアやデータ分析基盤構築の担当を経て、現在はデータや AI/ML の活用を軸とした、開発や 品質保証 (QA) の効率化業務にエンジニア、PM として従事。

所属プロジェクトについて

アジェンダ

01 課題とその背景

02 解決策

03 運用のポイント

04 今後の展望

05 まとめ

01. 課題とその背景

ゲーム開発は大規模化している

近年のゲーム開発は非常に大規模なプロジェクトです。

そのため、開発の現場において状況を正しく、客観的に把握するために、データによる可視化が必要になってきます。

開発データ収集における課題

オンライン要素がないタイトル

ゲームサーバーを持たないため、サーバーとの通信は基本的に行っていない。

大量のテストプレイ機材

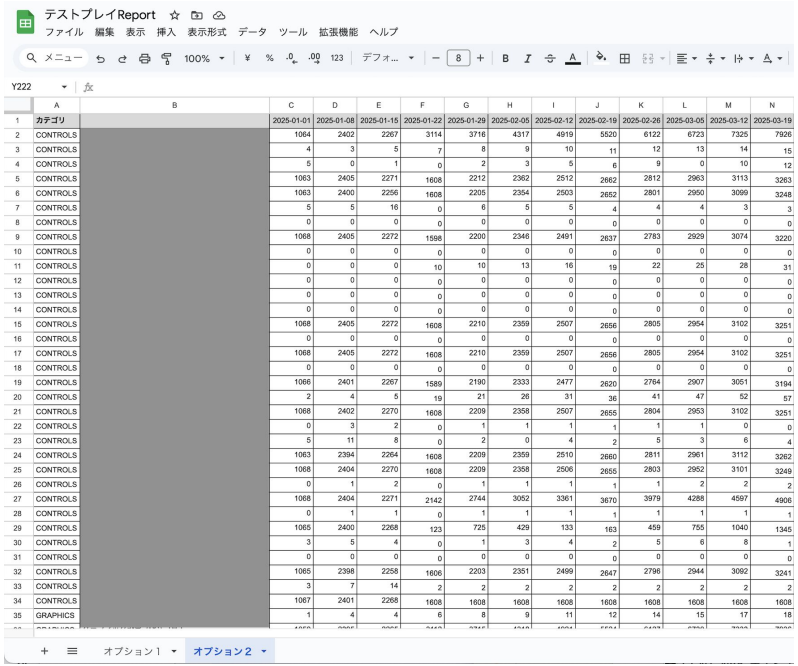
テストプレイ用の機材は、QA 繁忙期には 1 タイトル数十台規模の台数に。

サポート用途のログは...?

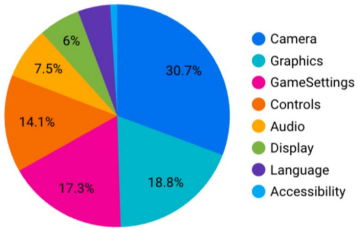
ユーザーサポート目的のログであれば多くの場合で収集の仕組みがあるが、開発状況の把握には適さない。

集計や可視化における課題

表計算ソフトによる集計
※画面はイメージです



BI ツールによる集計
※画面はイメージです



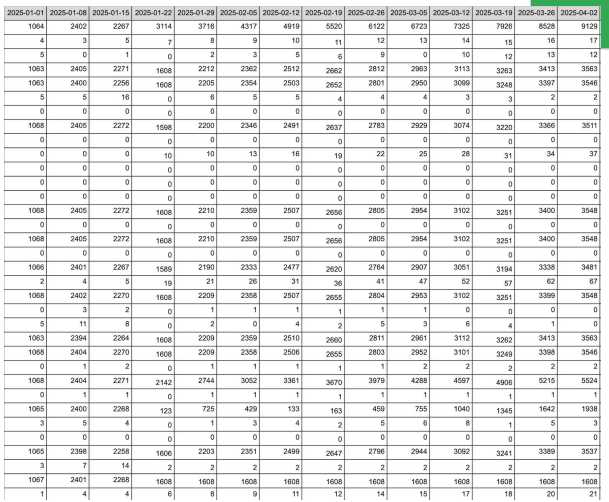
	CategoryName	MOptionValue	Record Count
6.	GameSettings	大	
7.	GameSettings	標準	
8.	GameSettings	大	
9.	GameSettings	標準	
10.	GameSettings	標準	
11.	GameSettings	大	

1 - 100 / 602

表計算ソフトでの集計

- データの入力やまとめに時間がかかる
- セル上に埋め込まれた計算式のメンテナンスが大変
- 関係者への共有の仕方やバージョン管理など、運用上の取り回し

※画面はイメージです

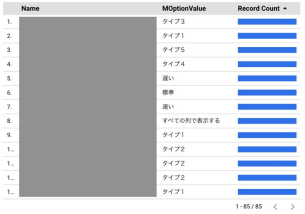


2025-01-01	2025-01-08	2025-01-15	2025-01-22	2025-01-29	2025-02-05	2025-02-12	2025-02-19	2025-02-26	2025-03-05	2025-03-12	2025-03-19	2025-03-26	2025-04-02
1064	2402	2267	3114	3716	4317	4919	5520	6122	6723	7325	7926	8528	9129
4	3	5	7	8	9	10	11	12	13	14	15	16	17
5	0	1	0	2	3	5	6	9	0	10	12	13	12
1063	2405	2271	1608	2212	2362	2512	2662	2812	2963	3113	3263	3413	3563
1063	2400	2256	1608	2205	2354	2503	2652	2801	2950	3099	3248	3397	3546
5	5	16	0	6	5	5	4	4	4	3	3	2	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1068	2405	2272	1598	2200	2346	2491	2637	2783	2929	3074	3220	3366	3511
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	10	10	13	16	19	22	25	28	31	34	37
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1068	2405	2272	1608	2210	2359	2507	2656	2805	2954	3102	3251	3400	3548
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1068	2405	2272	1608	2210	2359	2507	2656	2805	2954	3102	3251	3400	3548
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1066	2401	2267	1589	2190	2333	2477	2620	2764	2907	3051	3194	3338	3481
2	4	5	19	21	26	31	36	41	47	52	57	62	67
1068	2402	2270	1608	2209	2358	2507	2655	2804	2953	3102	3251	3399	3548
0	3	2	0	1	1	1	1	1	1	0	0	0	0
5	11	8	0	2	0	4	2	5	3	6	4	1	0
1063	2394	2264	1608	2209	2359	2510	2660	2811	2961	3112	3262	3413	3563
1068	2404	2270	1608	2209	2358	2506	2655	2803	2952	3101	3249	3398	3546
0	1	2	0	1	1	1	1	1	2	2	2	2	2
1068	2404	2271	2142	2744	3052	3361	3670	3979	4288	4597	4906	5215	5524
0	1	1	0	1	1	1	1	1	1	1	1	1	1
1065	2400	2268	123	725	429	133	163	459	755	1040	1345	1642	1938
3	5	4	0	1	3	4	2	5	6	8	1	5	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1065	2398	2258	1606	2203	2351	2499	2647	2796	2944	3092	3241	3389	3537
3	7	14	2	2	2	2	2	2	2	2	2	2	2
1067	2401	2268	1608	1608	1608	1608	1608	1608	1608	1608	1608	1608	1608
1	4	4	6	8	9	11	12	14	15	17	18	20	21

BI ツールでの集計

- SQL 等、問い合わせ言語の知識が必要
- 高機能なゆえに使いこなすのに一定のスキルが必要
- 汎用的なシステムなので痒いところに手が届きにくい

※画面はイメージです



データの活用が限定的な状態

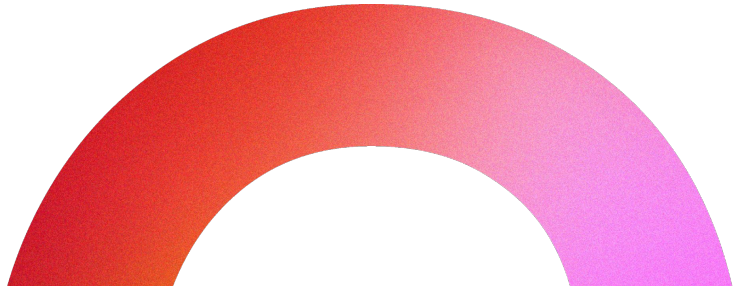
従来の方法では、そういった課題があり、特に導入や運用のハードルやコストが高かった。

そのため、データ活用自体は行っていても、一過性のものだったり限定的なものだった。

02. 解決策

開発や QA の現場でもデータの活用ニーズはある

大規模化しているゲーム開発・ QA において、客観的かつ俯瞰的に
状況把握するため。
そして、それは一過性の取り組みではなく、継続的なものにしたい。



行った取り組み

データを集める

タイトル問わず、共通した方法で、データを収集するための基盤を構築。

データを可視化する

収集したデータを手軽に可視化できる基盤。

そして、共通利用だけでなく、様々な現場のニーズに答えられる用途別の専用アプリケーション。

データを活用する

データ分析の専門家でなくても、容易にデータから意味を読み解くことができる仕組み。

データを集める

データを集めるにもログの送信を仕込んだり、一定のコストがかかる。

極力シンプルかつ統一的な方法で実現。

データを可視化する

大きく分けて 2 つ。

汎用的で誰でも GUI で可視化できる仕組み。

特定の用途に応じた専用の仕組み。

データを活用する

「データ分析の専門家でなくても、データ活用できる」ように、状況判断や意思決定ができる情報に落とし込む。

全体イメージと主要プロダクト

Overall view

Test Play Data

Development Data

Developers and QAs

Google Cloud

データを集める

BigQuery

データを可視化する

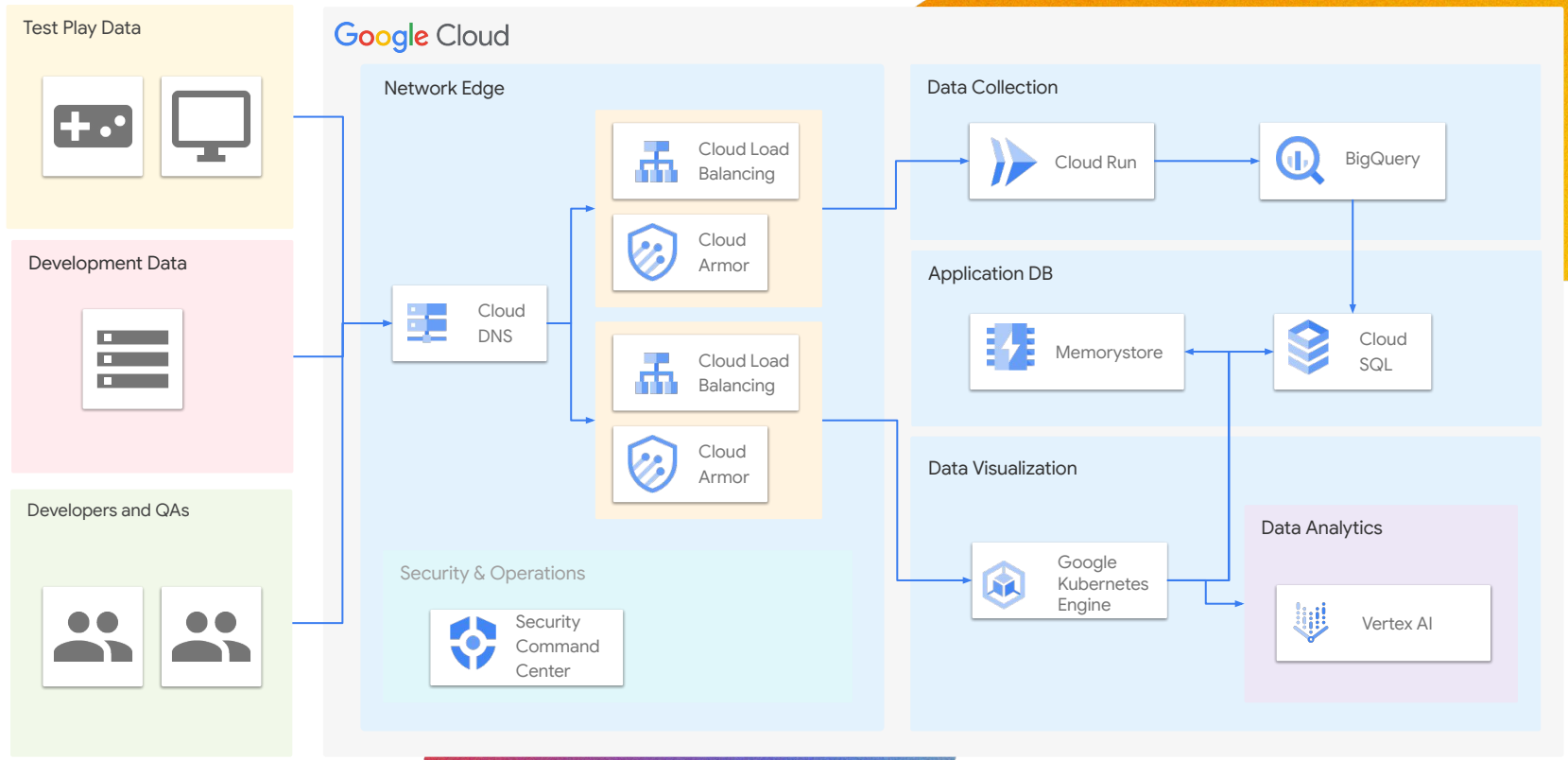
Google Kubernetes Engine

データを活用する

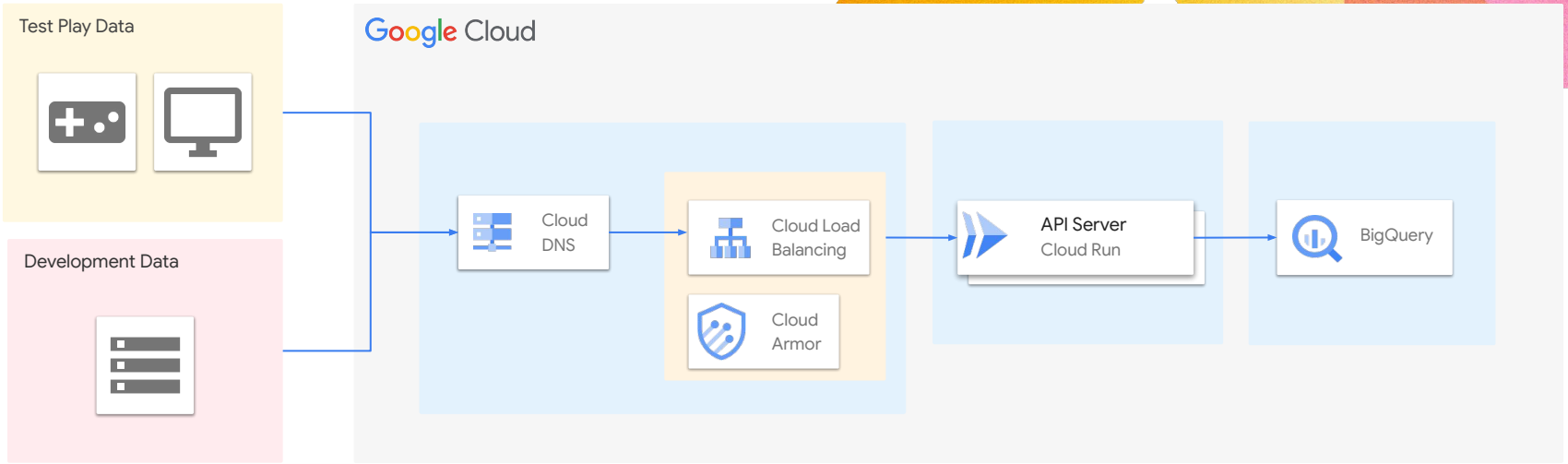
Vertex AI

アーキテクチャ

Architecture: Capcom Development Data Platform



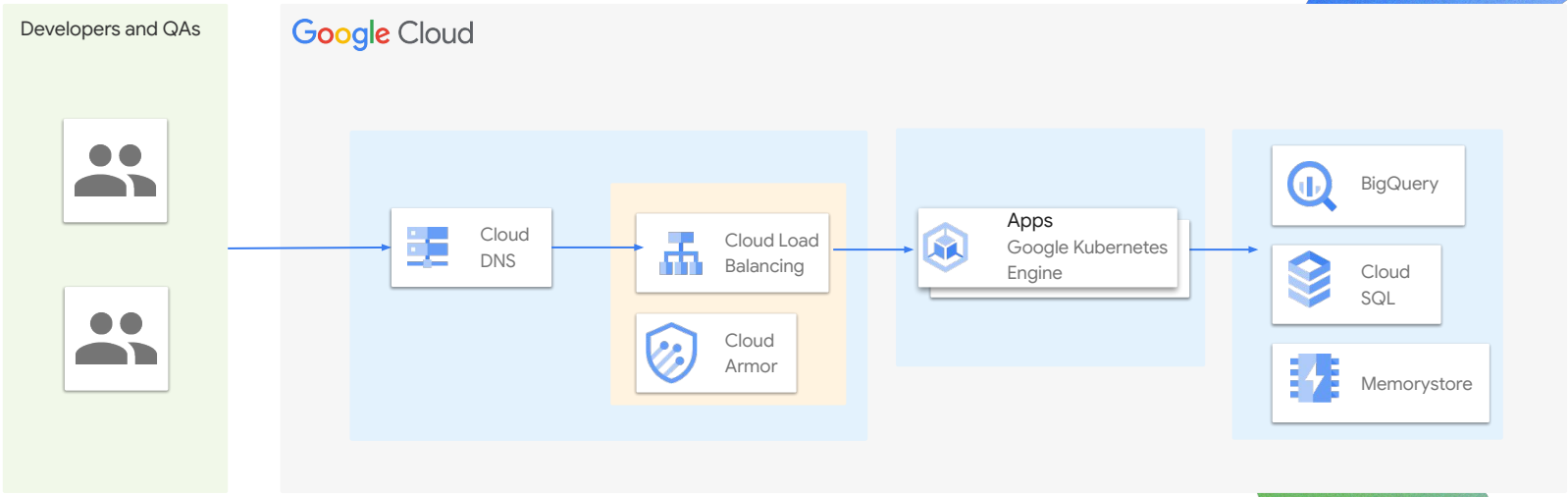
データ収集部分



社内のみで発生するデータとはいえ膨大なデータ量となるため、BigQuery を選択。

そして、API サーバーは構築や運用の手間を少なくするため、Cloud Run を選択。

データ可視化基盤



様々なアプリケーションを柔軟に運用したい。

運用の手間を少なくするため、Google Kubernetes Engine (GKE) を採用。

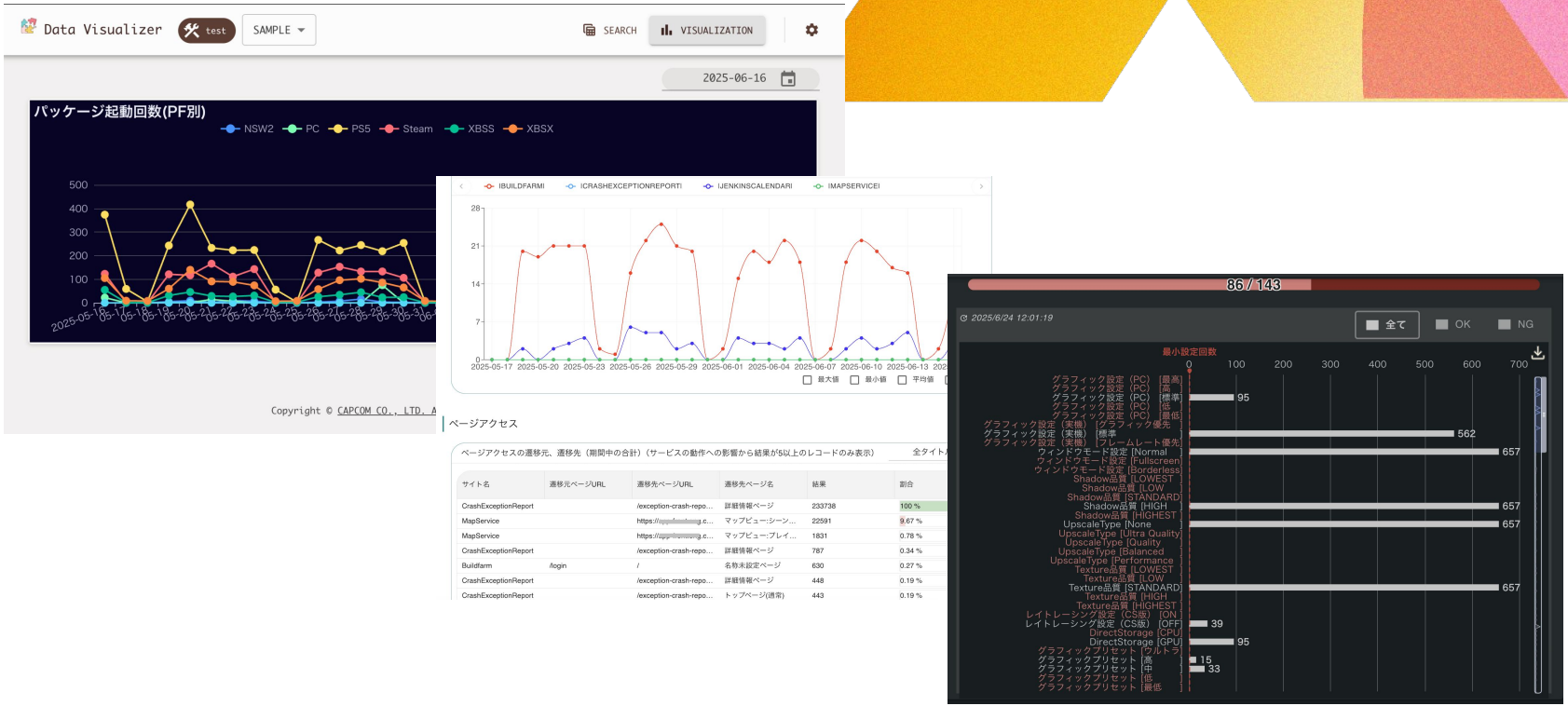
データベースについての補足

アプリケーションの要件に応じて
CloudSQL や Memorystore も使用し
ています。

Cloud SQL

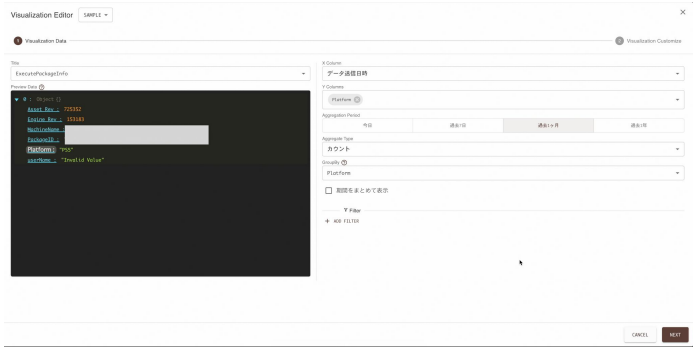
Memorystore

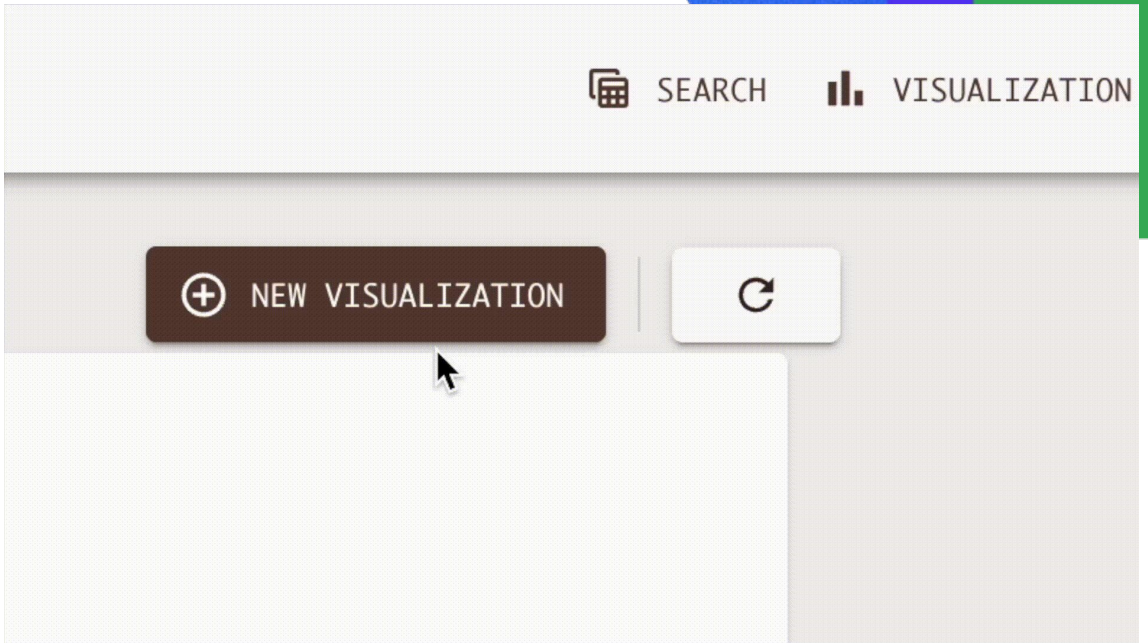
アプリケーションの例


Data Visualizer
test
SAMPLE
SEARCH
VISUALIZATION
2025-06-16
パッケージ起動回数(PF別)
NSW2
PC
PS5
Steam
XBSS
XBSX
Copyright © CAPCOM CO., LTD. A
ページアクセス
86 / 143
2025/6/24 12:01:19
全て
OK
NG
最小設定回数

汎用アプリケーション

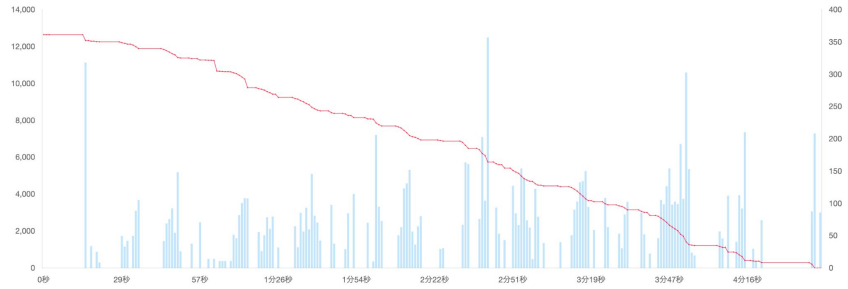
GUI によりセルフサービスで簡単なグラフが作成できる。

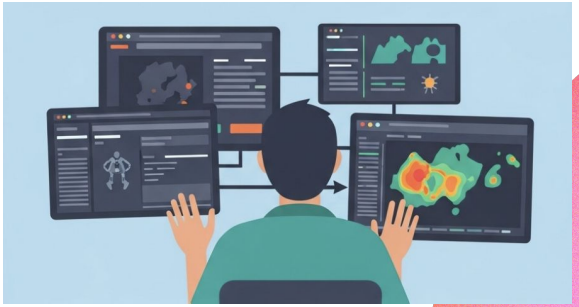




専用アプリケーション

目的に特化した専用アプリケーション群



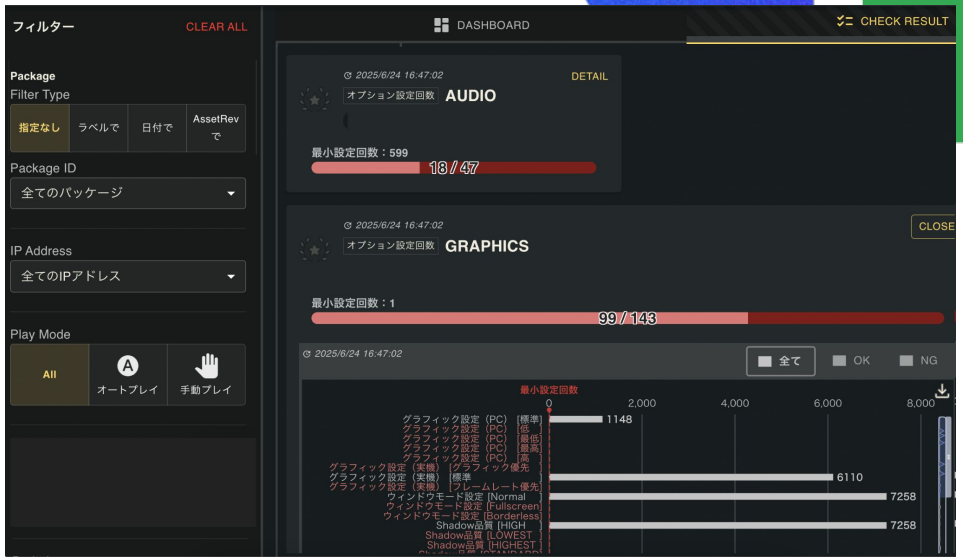


専用アプリケーションの一例

全てのゲーム中の要素が偏りなくチェックを通っていることを確認する目的のアプリケーション。

実績を集計し、既定値以下の項目を分かりやすく表示。

テストが不十分な箇所を特定、計画の修正などに活かす。



集計結果のレポート生成

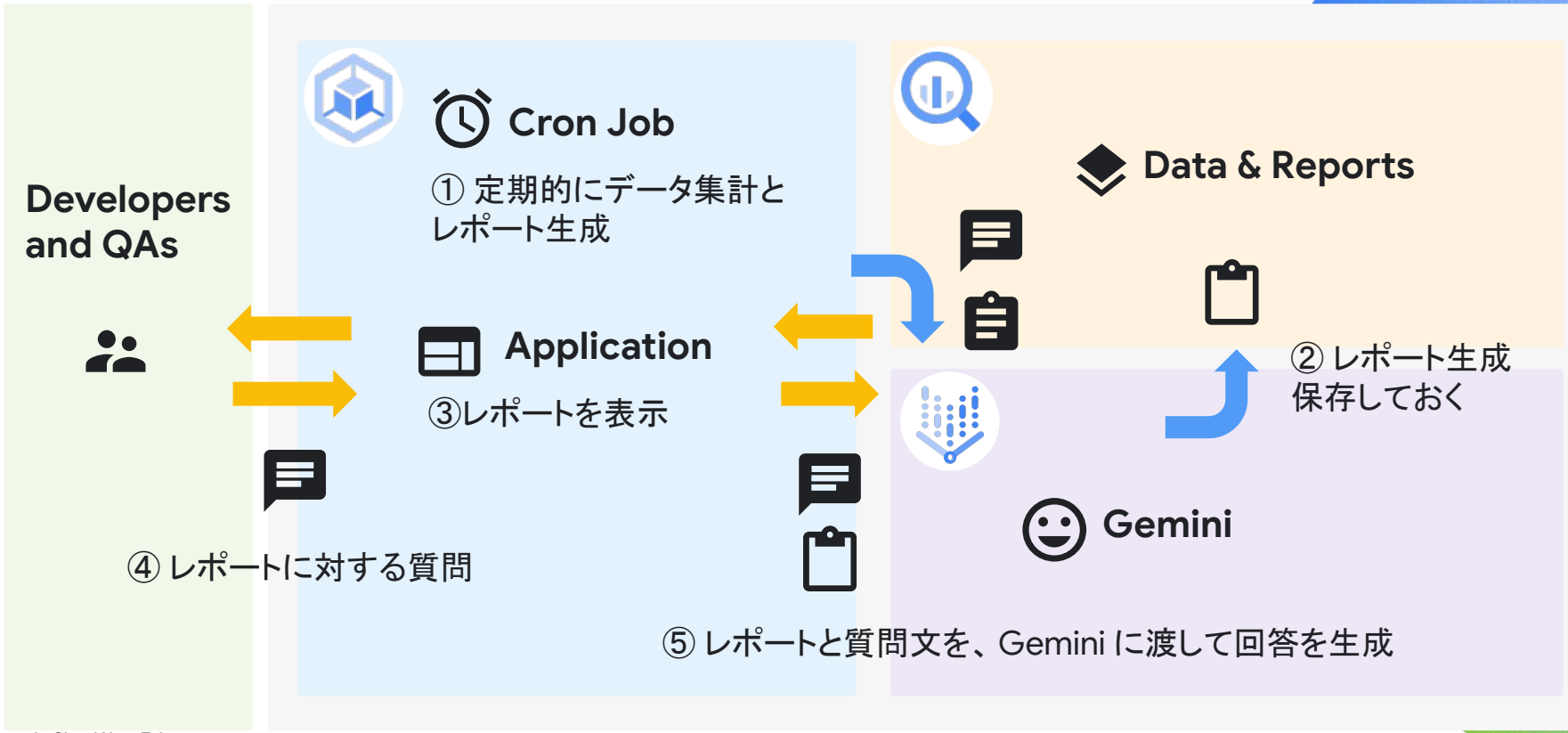
集計結果をレポートから大まかな傾向を掴む

LLM のモデルは

Gemini 2.0 Flash を使用。

100 万トークンの入力が可能、数学にも強い、ということで、データの集計結果からレポートを生成する、ということも容易に実現。

レポート生成と質疑応答の内部構成



プロンプトで工夫した点

マークダウン形式

マークダウン形式で、構造化して指示する。

フォーマット指定

入出力データのフォーマットを指定。具体例も記載。

ペルソナ設定

レポートを誰に対して提出する立場なのか、ペルソナを設定する。

03. 運用のポイント

活用に至ったポイント

- フルマネージド サービスを活用し、運用構築に割く時間を最低限にできたこと
- その分現場のニーズをヒアリングし、目的に応じたサービスを開発することに注力てきたこと
- アプリ部分は内製化し、小回りの効く運用ができたこと

開発スプリント

ヒアリング

利用者の要望を、定例ミーティングなどでヒアリング。

要望の実装

追加要望があった機能の検討、実装。

機能リリース

追加機能のリリース。

ヒアリング

追加機能の使用感、新たな要望などのヒアリング。

繰り返し...

04. 今後の展望

レポート生成やチャット機能の強化

Detailed report creation and Q&A by AI

Make Gemini an MCP client.

By doing so, questions could be answered by chat only from pre-collected results in the future, we think it will bs possible posible fo Gemini informatior from APIs and generate answere specifying to parameters according to questions.

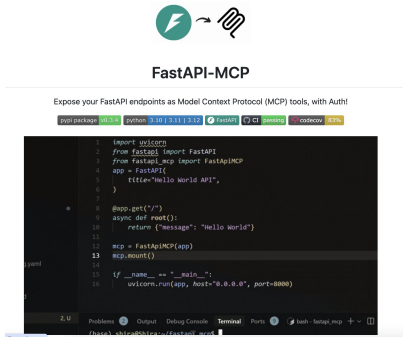
In the development field, there are times when waiw a rough overeiw, and times you want you know the pincilleitos status status of a packaged created yesterday, and there also needs to know the situation various perspectives on a case-by-case basis.

MCP サーバー対応

下記の OSS を利用する場合の例

- API サーバーに、FastAPI
- MCP サーバー化に、FastAPI-MCP

出典：https://fastapi.tiangolo.com/ja/



出典：https://github.com/tadata-org/fastapi_mcp

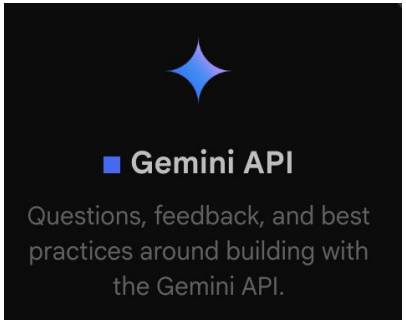
MCP サーバー対応

最低限、追加するコードはこれだけ

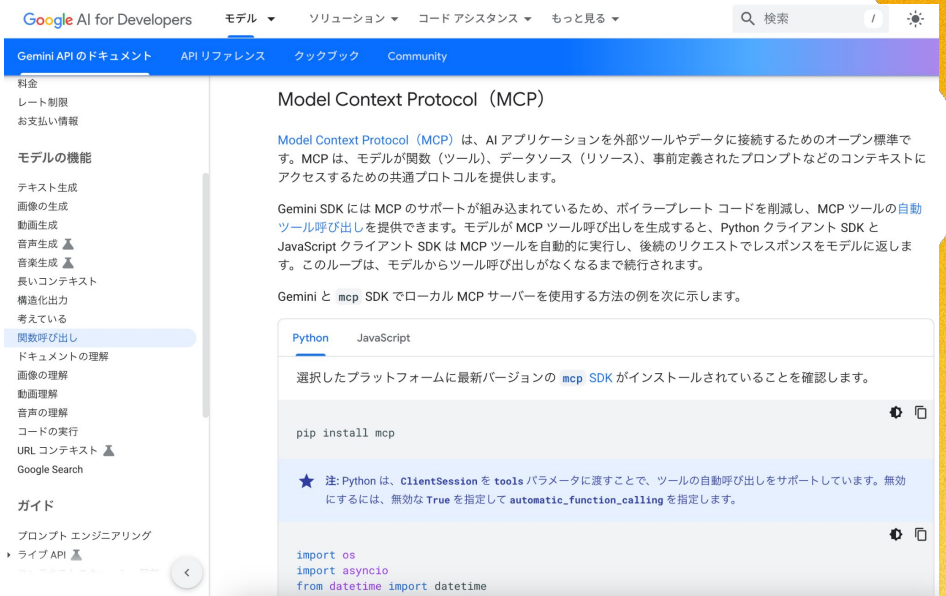
```python
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from fastapi_mcp import FastApiMCP

from app.router import router as api_router


def get_application():
    app = FastAPI(title="AutoPlayVisualizer API", version="1.0.0", docs_url="/autoplay-visualizer/api/docs")

    app.add_middleware(
        CORSMiddleware,
        allow_origins=["*"],
        allow_credentials=True,
        allow_methods=["*"],
        allow_headers=["*"],
    )

    app.include_router(api_router, prefix="/testplay-visualizer/api")

    return app

app = get_application()

# MCPサーバーを作成
mcp = FastApiMCP(
    app,
    name="TestplayVisualizer MCP Server",
    include_tags=["mcp"],
)
mcp.mount()
```

MCP クライアント

Gemini API SDK を使用



出典：https://discuss.ai.google.dev/



出典
：https://ai.google.dev/gemini-api/docs/function-calling?hl=ja&example=weather#model_context_protocol_mcp

MCP クライアント

最小限のサンプルコード

```python
import asyncio
import argparse
from mcp import ClientSession
from mcp.client.sse import sse_client
from google import genai
from google.oauth2 import service_account

credentials = service_account.Credentials.from_service_account_file(
    scopes=["https://www.googleapis.com/auth/cloud-platform"]
)
client = genai.Client(
    vertexai=True,
    project="developer-datapf-prd",
    location="us-central1",
    credentials=credentials
)

server_url = "http://0.0.0.0:8001/mcp"

async def run(prompt: str):
    async with sse_client(server_url) as (read, write):
        async with ClientSession(read, write) as session:

            await session.initialize()

            response = await client.aio.models.generate_content(
                model="gemini-2.0-flash-001",
                contents=prompt,
                config=genai.types.GenerateContentConfig(
                    temperature=0,
                    tools=[session],
                ),
            )

            print("----- Response -----")
            print(response.text)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--prompt", type=str)
    args = parser.parse_args()

    asyncio.run(run(args.prompt))

```

docstring を書く

LLM にとって API の
仕様理解に必要

```python
@router.get('/{title}/results', operation_id="get_result_list", tags=["mcp"])
async def get_result_list(
    title: str,
    phase_id: int = Query(default=1),
    qa_group: int = Query(default=0),
    category_id: int = Query(default=10),
    platform: str = Query(default=const.PARAM_PLATFORM_ALL),
    package_id: str = Query(default=const.PARAM_PACKAGE_ALL),
    date_range: str = Query(default=const.PARAM_DATE_RANGE_ALL),
    from_date: str = Query(default=None),
    to_date: str = Query(default=None),
    play_mode: str = Query(default=const.PARAM_PLAY_MODE_ALL),
    ip_address: str = Query(default=None),
):
    """
    # 関数の説明
    指定したカテゴリの集計結果を取得します。
    titleの指定は必須です。
    パラメータを指定しなくてもデフォルト値が適用されますので、必要に応じて変更してください。
    # パラメータの説明
    * from_date
    * 型
        str
    * 説明
        集計期間の開始日を YYYY-MM-DD 形式で指定します。
        date_range が CUSTOM_RANGE の場合に有効です。

    * to_date
    * 型
        str
    * 説明
        集計期間の終了日を YYYY-MM-DD 形式で指定します。
        date_range が CUSTOM_RANGE の場合に有効です。

    * play_mode
    * 型
```

補足のプロンプト

API 使用の精度を上げるため
補足の情報なども与える

```
    credentials=credentials
)

server_url = "http://0.0.0.0:8001/mcp"

api_param_rules = """
以下は、汎用的なMCPサーバーのAPIパラメータルールです。
全てのAPIで必要なものではありませんが、パラメータとそれが意味するところは共通です。
適宜解釈し、ツール呼び出しのパラメータ生成の際に参照してください。
指定されているかどうか、判断ができないパラメータは未指定としてください。
API側でデフォルト値が適用されます。

### 判例
* title
  * 型
    str
  * 説明
    データの対象となるプロジェクトのタイトル名。
    例: "XXXXXX"
"""

async def run(prompt: str):
    async with sse_client(server_url) as (read, write):
        async with ClientSession(read, write) as session:

            await session.initialize()

            response = await client.aio.models.generate_content(
                model="gemini-2.0-flash-001",
                contents=prompt + api_param_rules,
                config=genai.types.GenerateContentConfig(
                    temperature=0,
                    tools=[session],
                ),
            )

            print("----- Response -----")
            print(response.text)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--prompt", type=str)
    args = parser.parse_args()
```

実行結果

```
$ python sse_app.py --prompt 開発チームの手動プレイによる、オプションチェックの集計結果をまとめてください。対象日は2025/5/26から2/25/5/30までとしてください。
----- Response -----
開発チームの手動プレイによる、オプションチェックの集計結果です。対象日は2025/5/26から2025/5/30までです。

*   オプション設定回数：GAME\_SETTING

    *   最小設定回数：20
    *   パスした回数：96
    *   チェック項目の総回数：114
*   オプション設定回数：CONTROLS

    *   最小設定回数：20
    *   パスした回数：60
    *   チェック項目の総回数：85
*   オプション設定回数：CAMERA

    *   最小設定回数：20
    *   パスした回数：114
    *   チェック項目の総回数：187
*   オプション設定回数：DISPLAY

    *   最小設定回数：20
    *   パスした回数：89
    *   チェック項目の総回数：132
*   オプション設定回数：AUDIO

    *   最小設定回数：20
    *   パスした回数：40
    *   チェック項目の総回数：47
```

LLM が自分で判断して、API をコール。
さらにプロンプトの情報から、API のパラメータも生成。
そして得られた結果を要約している。

パラメータの確認

【プロンプト】
開発チームの手動プレイによる、オプションチェックの集計結果をまとめてください。対象日は 2025 / 5 / 26から 2025 / 5 / 30 までとしてください。

API 側で受け取ったパラメータのログ

拡大

```
TTP/1.1" 202 Accepted
 date_range=CUSTOM_RANGE, from_date=2025-05-26, to_date=2025-05-30
```

より柔軟なやり取りが可能に

- 「昨日作成した」パッケージの動作は安定してる？
- 過去一週間で不具合の件数は「減少傾向」なのか？
- 「バージョン ？？？」の「〇〇ステージ」の状況を知りたい

等々...

そして、そこから続けて質問をしていくことで

より深堀りした詳細な分析が可能に...

全ての情報へ統合的なアクセスが可能に

05. まとめ

まとめ

1. マネージド サービスを活用し、迅速にアプリを構築。

2. LLM の活用により、専門家でなくてもデータから状況を読み取れるように。

3. 今後、さらなる AI の発展に期待。そして、さらにクリエイティブ領域の時間を確保し、より面白いゲームを。