CloudArchitectBuddy
構造的状態制御に基づくシステム主導型設計支援

小比田 涼介 * 春日 瑛 *
サイバーエージェント
{kohita_ryosuke, kasuga_akira}@cyberagent.co.jp

概要

本研究では、クラウドアーキテクチャ設計の複雑さを取り上げ、その設計支援システムとしてCloudArchitectBuddy（CA-Buddy）を提案する．CA-Buddy は要件と設計を状態として構造的に表現・制御することで設計の理解と一貫性を支援し、システムが主導的に検証と具体化を進めることでスムーズな設計作業を実現する．実務者による CA-Buddy と ChatGPT の比較実験の結果、設計品質は同等ながらも CA-Buddy は高いユーザー体験を示した．またフィードバックから、CA-Buddy の構造化された設計情報とシステム主導型の設計支援はチャット UI の柔軟な対話を統合することによってより効果的なクラウド設計支援ができる可能性が示唆された．

1 導入

クラウドアーキテクチャ設計は、多様なサービスの組み合わせとその構成管理を必要とするシステム開発工程である [1]．設計者は機能要件に加え、スケーラビリティやセキュリティといった非機能要件、コストと運用性のバランス、将来的な拡張性も考慮しなければならない [2]．パターンカタログやベストプラクティス集 [3] は存在するものの、個別の要件に適したアーキテクチャの設計にはクラウド環境に関する深い理解と経験が求められる．

このような設計支援に向けて、近年では大規模言語モデル（LLM）を活用したシステム開発支援の研究が進展している [4]．LLM はコード生成やデバッグ支援などソフトウェア開発領域で幅広く活用でき [5]，ChatGPT などの対話型インターフェースは、アイデア発想促進や要件定義の効率化に有効である [6]．継続的な対話を通じた要件の具体化や、一貫性のある文書化による人的エラー低減などの利点が報告されている [7, 8, 9]．しかし、要件の曖昧性や推論エラーによる誤解釈の可能性など、効果的な活用のためには専門家による適切な対話の誘導と要件の精緻化が必要とされている [10, 11]．

これらの課題に対し、我々はクラウド設計支援システム CloudArchitectBuddy（CA-Buddy）を提案する．図 1 がシステム概要（上段）とシステム UI（下段）である．本システムは設計プロセスを二つの状態 (User State と Architecture State) として構造的に表現し制御する．User State は要件を、Architecture State は設計を構造的に保持する．対話フローは初期要件に基づくアーキテクチャの提案から始まり、システムが主導的に検証と質問生成を行い、設計者の回答を通じて段階的に状態を更新することで要件の精緻化と設計の洗練を実現する．構造的な状態制御により設計の理解と一貫性を向上させ、システム主導型のアプローチによりスムーズな設計プロセスを実現する．

本研究では、16 名の実務者による評価実験を実施した．実験を通じて、設計品質としては従来の対話型 UI と同程度であったものの、構造的な状態制御とシステム主導型アプローチにより設計の理解を促進し、作業負荷を軽減できることを確認した．また、システム主導型の設計支援は設計者の負荷を軽減する一方で、柔軟な設計指示や詳細把握に制約があることも明らかになった．これらの知見は、CA-Buddy の構造的な状態制御とチャット UI の柔軟な対話を組み合わせることで、両者の利点を活かしたより効果的な支援の実現可能性を示唆している．

2 手法

CA-Buddy は、**要件と設計を状態モデルとして構造的に表現・制御**しつつ、**システムが主導的に設計プロセスを進める**ことで、**要件の精緻化と設計の洗練を実現する**システムである．

* 共同筆頭著者

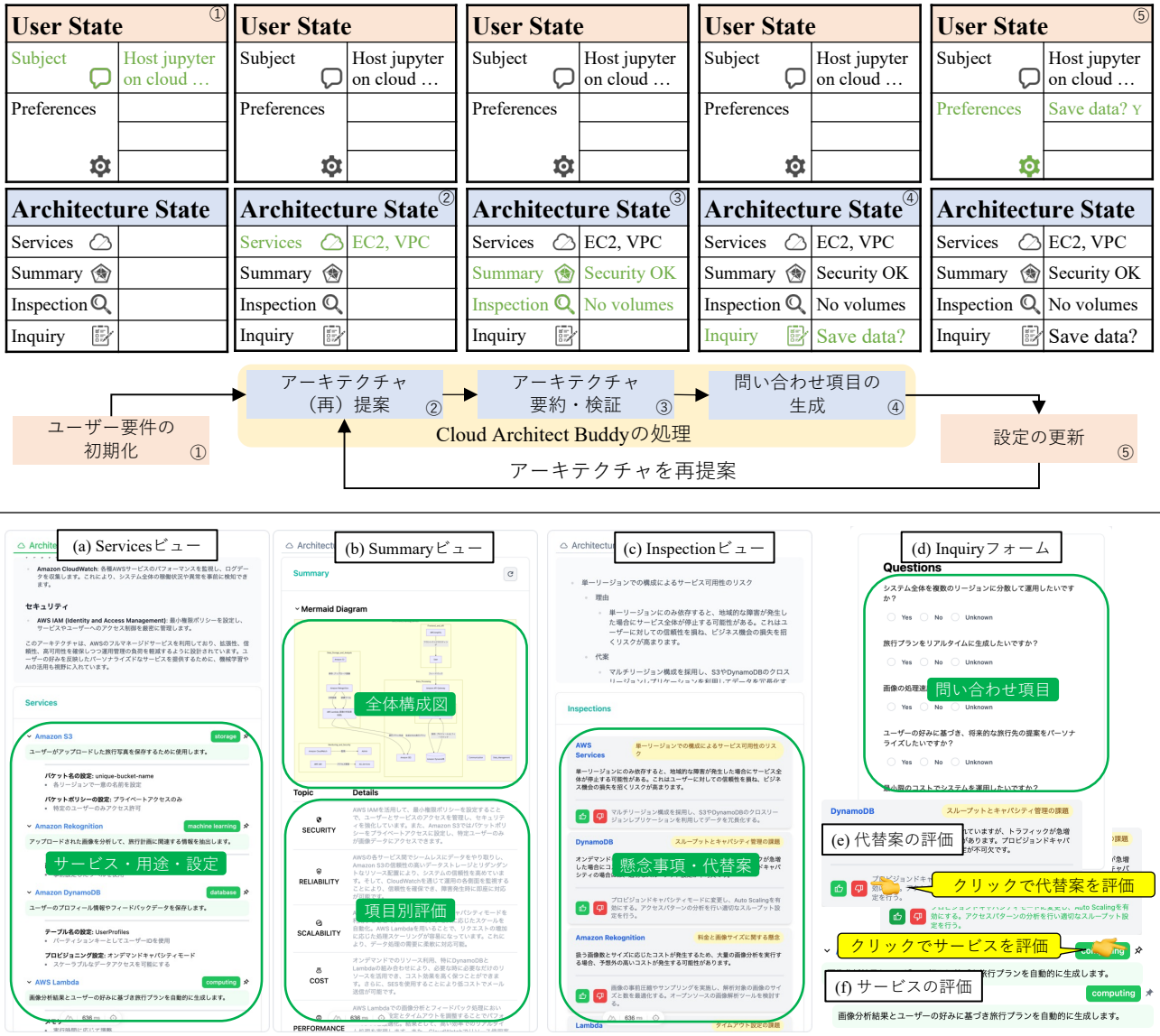

図 1　【上段】状態モデルおよびその遷移（上）・対話フロー（下）【下段】アプリケーション UI

構造的状態モデル　図 1（上段上部）に状態の構造を示す．User State は設計者の要件を表現し、初期のビジネスニーズ（Subject）と対話を通じて具体化される詳細要件（Preferences）で構成される．Architecture State はアーキテクチャ設計情報を管理し、クラウドサービスの構成（Services）、アーキテクチャの要約（Summary）、検証結果（Inspection）、および要件確認用の問い合わせ（Inquiry）を含む．この構造的な状態表現により、設計情報の体系的な理解と一貫性の維持を実現する．

システム主導型の対話フロー　図 1（上段下部）に対話フローを示す．設計プロセスは初期要件の入力（①）から始まり、その後はシステムが自律的に以下の処理を実行する：まず User State に基づきアーキテクチャを提案し（②），続いて提案内容の要約（項目別評価と構成図の生成）と検証（懸念事項と代替案の洗い出し）を行う（③）．さらに設計判断に必要な質問を自動生成し（④），得られた回答を詳細要件として登録する（⑤）．このシステム主導のループを通じて、要件とアーキテクチャを段階的に洗練する

アシスタント　システム主導型の各処理（アーキテクチャ提案、要約、検証、問い合わせ生成）は GPT-4o の Structured Output 機能で実装されている[1)]．各処理では、現在の状態をプロンプトとして与え、定められた構造で出力を得る．これにより、柔軟なコンテンツ生成と確実な状態モデルの更新を実現し

1）https://platform.openai.com/docs/guides/structured-outputs

表 1　クラウド資格認定試験問題の正答率

モデル	PCA（GCP）	SPA（AWS）
GPT-4o-mini	41/50 (82%)	32/50 (64%)
GPT-4o	44/50 (88%)	41/50 (82%)

ている．

UI　図 1（下段）に CA-Buddy の UI を示す．状態モデルを構造的に可視化し、使用サービス一覧（a）、全体構成図・項目別評価（b）、懸念点と代替案（c）を分かりやすく表示する．対話フローにおける質問への回答（d）に加え、提案された代替案（f）や使用サービス（e）に対する直接的な評価を通じて要件を伝えることができる．また設計履歴の追跡や過去バージョンへの復帰機能も備える．これらの UI により、状態モデルの構造的な表現とシステム主導の設計支援を効果的に実現する．

3　実験

設定　本研究では CA-Buddy の有効性を 2 段階で評価した．第 1 段階では、基盤となる GPT-4o のクラウド設計知識を AWS と GCP の資格認定試験問題を用いて検証した．第 2 段階では、実務者 16 名を対象に CA-Buddy と ChatGPT の比較実験を実施した．比較実験では、データ分析基盤、EC サイト、AI 旅行サービス、SNS アプリの 4 つの設計シナリオについて、各参加者がいずれかのツールを用いてアーキテクチャ設計を行った．評価は 2 つの観点から実施した：設計品質についてはエキスパートが設定した評価基準に基づく 3 段階スコアリング、ユーザー体験については（1）使いたいか（2）使いやすいか（3）薦めたいかの 10 段階のリッカート尺度による評価と自由記述を収集した．実験設定の詳細は付録 A に記載する．

結果　まず第 1 段階の GPT-4o のクラウド知識レベルの検証結果について報告する．表 1 は Google Professional Cloud Architect（PCA）および AWS Solutions Architect Professional（SAP）の資格認定試験問題に対する解答結果である．GPT-4o は PCA で 88%、SAP で 82%の高い正答率を達成した．より小規模な GPT-4o-mini でも PCA で 82%で、SAP で 64%の正答率を記録しており、LLM のクラウド設計に関する基本的な理解が確認できた．問題文はアーキテクチャ設計を決める上で必要となる要件が詳細に記述されており、この結果から、詳細要件が与えられれば適切なアーキテクチャを提案できる能力が GPT-4o に備わっていることがわかった．

表 2　シナリオ別・項目別設計スコア（3 点満点）

	評価項目	ChatGPT	CA-Buddy
GCP-1	データ取得	**2.875**	2.750
	データ保管	**2.000**	1.875
	データ分析	**2.500**	2.375
	データ統合	**1.750**	1.625
	運用チームとの連携	1.625	**2.000**
	モニタリング	**2.750**	2.500
GCP-2	EC サイト構築	**2.375**	2.125
	データ保管	1.250	**1.375**
	チャット機能	2.375	2.375
	データ処理	1.750	**1.875**
	データ分析	**1.750**	1.625
	データ機密	1.125	**1.625**
	モニタリング	1.500	**2.000**
AWS-1	写真の保管	2.875	**3.000**
	通知メールの送信	**3.000**	2.875
	ユーザー情報の保管	**2.625**	2.250
	分析の実行フロー	2.000	**2.125**
	アプリ実行環境	2.375	2.375
	UI ホスティング	1.625	**2.250**
AWS-2	構造化データの保管	2.250	**2.500**
	高トラフィック対応	**3.000**	2.500
	通知機能	2.250	**2.375**
	アプリ実行環境	**2.750**	2.125
	チャット機能 *	**2.750**	2.000
	UI ホスティング	**2.875**	2.375

表 3　ユーザー体験評価結果（10 点満点）

システム	使いたいか	使いやすいか	薦めたいか
ChatGPT	7.06	6.75	7.12
CA-Buddy	7.00	7.93	7.62

次に第 2 段階の設計実験の結果について報告する．表 2 に項目別設計評価スコアを示す．全 25 評価項目において、ChatGPT が 13 項目、CA-Buddy が 10 項目で高スコアを示し、2 項目が同点となった．ただし多くの項目で差は僅かであり、Mann-Whitney U 検定（有意水準 $p < 0.05$）では全 25 項目中 1 項目（AWS-2 のチャット機能）でのみ有意差が検出された．異なるシナリオや評価側面においてスコアのばらつきは見られたものの、いずれかのツールが特定の面で一貫して優位である傾向は観察されなかった．

表 3 に示すアンケート調査結果では、CA-Buddy は「使いやすいか」（7.94 対 6.82）および「薦めたいか」（7.76 対 7.18）において高い評価を得た．「使いたいか」については同程度の評価（7.18 対 7.12）となり、実務者が両ツールに異なる側面での価値を見

表 4　各ツールに対するユーザーフィードバック（括弧：指摘人数、箇条書き：フィードバック例）

	ChatGPT	CA-Buddy
入力	✓ **自由な要件とディスカッション** [8] • 意図を自由に反映できる • 柔軟なテキスト入力で設計を改善できて良い • シンプルな形式で制限のない質問が可能 • 細かい詳細まで議論可能	✓ **直感的な設計意図の反映** [4] • 回答に基づくアーキテクチャの更新が便利 • サービスのピン留め機能が有用 ✓ **システム主導による容易な設計プロセス** [4] • 初期アイデアでの技術的調査時に非常に有用
	× **効果的な質問の作成に専門知識が必要** [7] • 適切な質問にはクラウドの専門知識が必要 • 大雑把に質問すると回答が的外れになる • 次に何を質問すべきか判断が難しい	× **要件指定の柔軟性の制限** [7] • テキストで具体的な修正を指示したい × **特定トピックの深掘りが困難** [6] • チャット対話でより深く掘り下げたかった
出力	✓ **柔軟な出力形式とコンテンツのカスタマイズ** [3] • 出力形式を自由にコントロール可能 • *Web* 検索不要（チャット対話で済む） • 出力順序と形式をカスタマイズ可能	✓ **構造化された情報が理解を促進** [4] • サービスが機能と共に把握しやすい ✓ **視覚的な図表が全体理解を助ける** [9] • 図表により一目でアーキテクチャが明確に
	× **非構造化テキストで理解が困難** [6] • 大規模なサービス提案が把握しづらい	× **詳細情報へのアクセスが制限的** [5] • 提案内容について深掘りしたかった
操作	✓ **特定の側面の詳細な調査が容易** [5] • チャット形式で直接質問可能 ✓ **馴染みやすく単純な対話モデル** [3] • シンプルなチャット形式が親しみやすく快適	✓ **検査機能が設計の見落としを特定** [5] • 検査により考慮していなかった要件が明らかに ✓ **意図から設計提案までのスムーズな流れ** [5] • 質問に答えるだけで設計が進んで楽
	× **会話の方向性維持が困難** [4] • 漠然と質問すると方向性がずれる	× **プロセスフローの様々な制限** [2] • ループごとの提案の改善点を理解したかった

出していることが示唆された．

表 4 は両ツールへのユーザーフィードバックをカテゴリ別に整理したものである．

入力形式について、CA-Buddy は質問回答や評価機能を通じた直感的な設計意図の反映が評価され（4 名）、「回答に基づくアーキテクチャの更新が便利」との声があった．一方、要件指定の柔軟性（7 名）や特定事項の深掘り（6 名）に制約があった．ChatGPT は自由な要件反映と質問が可能で（8 名）、「詳細まで議論可能」と評価されたが、効果的な利用には専門知識が必要と指摘された（7 名）．

出力形式では、CA-Buddy の構造化された表現、特に視覚的な図表による理解促進が支持され（9 名）、「図表により一目でアーキテクチャが明確に」との評価を得た．ただし、自由入力の制限に関連して詳細情報へのアクセスに制約があると指摘された（5 名）．ChatGPT は出力の柔軟性が評価された（3 名）一方で、非構造化テキストからの複雑なアーキテクチャ把握に負担を感じる意見が多かった（6 名）．

操作性について、CA-Buddy の自律的な検証と質問生成は未考慮要件の発見（5 名）と設計プロセスの円滑化（5 名）に貢献し、「質問に答えるだけで設計が進んで楽」との声があった．一方、固定フローによる制約も指摘された．ChatGPT は詳細調査の容易さ（5 名）と親しみやすさ（3 名）が評価される一方、「漠然と質問すると方向性がずれる」など、会話の方向性維持の困難さが指摘された（4 名）．

4　結論

本研究ではクラウドアーキテクチャ設計支援システム CA-Buddy を提案した．要件と設計を状態モデルとして構造的に表現・制御し、システム主導で検証と具体化を進めるアプローチは、ChatGPT と比較して高いユーザー体験を実現した．設計品質は同等であったが、フィードバックから両ツールの特性が相補的であることが判明した．構造化された設計情報とシステム主導の支援に柔軟な対話を組み合わせることで、より効果的な設計支援が期待できる．

参考文献

[1] Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, and Peter Arbitter. **Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications**. Springer Vienna, 2014.

[2] Maria Salama and Rami Bahsoon. Quality-driven architectural patterns for self-aware cloud-based software. In **Proceedings of the 2015 IEEE 8th International Conference on Cloud Computing**, 2015.

[3] Alex Homer, John Sharp, Larry Brader, Masashi Narumoto, and Trent Swanson. **Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications**. Microsoft patterns & practices, 2014.

[4] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large language models for software engineering: A systematic literature review. **ACM Transactions on Software Engineering and Methodology**, Vol. 33, No. 8, 2024.

[5] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. Large Language Models for Software Engineering: Survey and Open Problems . In **Proceedings of the 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering**, 2023.

[6] Jules White, Sam Hays, Quchen Fu, Jesse Spencer-Smith, and Douglas C. Schmidt. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. In **Generative AI for Effective Software Development**, pp. 71–108. Springer, 2024.

[7] Nuno Marques, Rodrigo Rocha Silva, and Jorge Bernardino. Using chatgpt in software requirements engineering: A comprehensive review. **Future Internet**, Vol. 16, No. 6, 2024.

[8] Jenny T. Liang, Chenyang Yang, and Brad A. Myers. A large-scale survey on the usability of ai programming assistants: Successes and challenges. In **Proceedings of the IEEE/ACM 46th International Conference on Software Engineering**, 2024.

[9] Steven I. Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D. Weisz. The programmer' s assistant: Conversational interaction with a large language model for software development. In **Proceedings of the 28th International Conference on Intelligent User Interfaces**, 2023.

[10] Quanjun Zhang, Tongke Zhang, Juan Zhai, Chunrong Fang, Bowen Yu, Weisong Sun, and Zhenyu Chen. A critical review of large language model on software engineering: An example from chatgpt and automated program repair. **arXiv preprint arXiv:2310.08879**, 2024.

[11] Chetan Arora, John Grundy, and Mohamed Abdelrazek. Advancing requirements engineering through generative ai: Assessing the role of llms. In **Generative AI for Effective Software Development**, pp. 129–148. Springer, 2024.

A　実験設定

クラウド知識レベルの検証　GCP の Professional Cloud Architect[2] と AWS Solutions Architect Professional[3] の試験問題を使用した。両試験とも詳細な要件と設計に関する選択式問題で、GCP は 50 問、AWS は 50 問を出題し、正答率を評価指標とした。検証モデルは gpt-4o-2024-08-06 を使用した。

設計実験　被験者はクラウド設計の実務経験を持つエンジニアとデータサイエンティスト 16 名とした。4 つのシナリオ（データ分析基盤、EC サイト、AI 旅行サービス、SNS アプリ）について、被験者は CA-Buddy または ChatGPT を使用して設計を行った。シナリオとツールの組み合わせ及び実施順は被験者間でカウンターバランスを取った。

実験は 80 分のセッションで、説明（10 分）、設計（60 分：4 シナリオ× 15 分）、フィードバック（10 分）で構成した。設計フェイズでは、参加者には新規プロダクトのリードエンジニアという役割を与え、与えられたシナリオに対し 15 分以内に指定フォーマットでアーキテクチャ提案を作成するよう指示した。表 5 に指定フォーマットとその例を示す。全参加者が ChatGPT の使用経験を持つ一方、CA-Buddy は初めての使用のため 5 分間の練習時間を説明内で設けた。

実験は 2024 年 12 月 3 日と 5 日に各 8 名で実施し、同一会議室で環境を統一した。CA-Buddy では GPT-4o-2024-08-06、ChatGPT では最新モデルを使用した。設計評価は、エキスパートが設定した要件とクラウドサービス（3 段階）を参照して人手でスコアリングした。表 6 にシナリオ概要と評価基準を示す。

表 5　設計案の提出フォーマットと例

サービス	用途	設定
S3	フロントエンドを静的ウェブサイトとしてデプロイ	公開アクセスの有効化 / 静的ウェブホスティング
CloudFront	S3 静的ウェブサイトをキャッシュ	キャッシュ設定の最適化 / レート制限と WAF の適用
API Gateway	RESTful API の作成	スロットリングの有効化
Lambda	API ゲートウェイのバックエンド	エンドポイントの定義（ユーザー、プロフィール、マッチング）
DynamoDB	サーバーレス NoSQL(ユーザー情報やいいね/マッチを保存)	自動スケーリングの有効化

表 6　シナリオ・評価項目・スコア別使用サービス例

シナリオ概要	評価項目	I	II	III
【GCP-1】数千の物理デバイスからユーザーデータを収集・分析する基盤を Google Cloud 上で開発します。運用チームが分析・改善しやすい環境にし、自社の WEB ログや購買データと統合してマーケティングチームでも活用したいです。	データ取得	REST API (REST)	Cloud Functions + Pub/Sub	Pub/Sub + Dataflow
	データ保管	BigQuery (BQ)	BQ + Cloud Storage	BQ + BigTable / Spanner
	データ分析	BQ	BQ + Notebook Dataproc	BQ + Looker Studio
	データ統合	BQ	BQ + Dataplex / Composer	BQ + Composer + Dataplex
	運用チームとの連携	BQ	BQ + IAM	BQ + IAM + Audit log
	モニタリング	None	Cloud Logging (CL)	CL + Cloud Monitoring (CM)
【GCP-2】GoogleCloud で自社 EC サイトを構築し、カスタマーサポート用のチャットツールを導入します。顧客情報を適切に管理した上で、継続的に分析を行い、顧客体験を改善したいです。	EC サイト構築	GCE (MIG)	GKE, Run	GKE / Run + Storage + CDN
	データ保管	Cloud SQL	Spanner	Spanner + Memory Store
	チャット機能	None	Cloud Run	Dialogflow
	データ処理	None	Pub/Sub	Pub/Sub + Dataflow / Functions
	データ分析	BQ	BQ + VertexAI	BQ + VertexAI + Looker
	データ機密	IAM / Auth	Secret Manager, KMS	DLP
	モニタリング	None	CL, CM	CL, CM + CI/CD
【AWS-1】旅行写真をアップロードするとその後の旅行プランを提案するサービスを AWS にデプロイしたいです。提案プランは登録メールアドレスに送信する予定です。また、フィードバックの仕組みを入れて、ユーザーの好みを反映したいです。	写真の保管	RDS / Aurora	Dynamo	S3
	通知メールの送信	EC2	Lambda / ECS	SES
	ユーザー情報の保管	Cognito	S3, RDS	Dynamo, Aurora, S3 + Athena
	分析の実行フロー	EventBridge	Lambda	SQS / StepFunctions
	アプリ実行環境	EC2	Lambda, ECS	II + ML services
	UI ホスティング	REST	S3	II + Cloudfront / Amplify
【AWS-2】AWS で趣味で繋がるマッチングアプリを作りたいです。パッと簡単にプロフィールを見ていいねでき、マッチしたらメッセージを送れるようにします。さらに、レストランの予約もできるようにしたいです。	構造化データの保管	S3	Dynamo	RDS, Aurora
	高トラフィック対応	S3	RDS, Aurora	Dynamo
	通知機能	EC2, ECS	AppSync	SNS
	アプリ実行環境	EC2	Lambda / ECS	Lambda / ECS + API Gateway
	チャット機能	REST	AppSync, Websocket	Lambda / ECS + Dynamo
	UI ホスティング	REST	S3	Cloudfront / Amplify

2）https://cloud.google.com/learn/certification/cloud-architect
3）https://aws.amazon.com/jp/certification/certified-solutions-architect-professional