Smart Archie 雲端架構生成助手

顏勝豪 Otto

現在

- 國泰金控 數數發中心 雲端策略發展部 協理
- 負責執行集團雲端轉型計劃，擬定集團上雲策略

過去

- 10年以上的軟體專案開發經驗
- 歷經紮實的養成訓練，由程式設計師→系統設計師→系統分析師→技術經理→協理
- 帶領專案開發部門承接多種領域之
  - ✓ 外商銀行交易系統、商業智慧分析
  - ✓ 警政情資整合分析平台、大數據分析平台架構設計與建置

國泰雲端策略發展史：Cloud Ready -> 大規模上雲

2021 H1

- •Cloud Ready for 銀行 啟動
- •Cathay 6R 方法論誕生

2021 H2

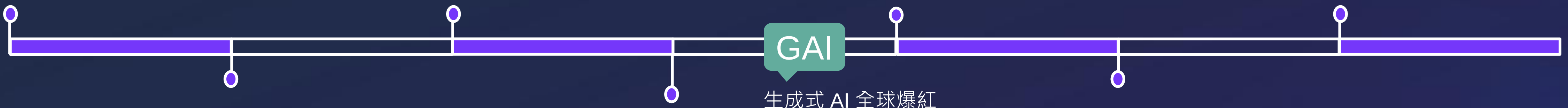
- •台灣雲端大會：上雲首部曲分享
- •國泰技術年會：跨界雲端新常態

2022 H1

- •Cloud Ready for 人壽 啟動
- •Cloud Ready for 產險 啟動

2022 H2

- •Cloud Ready Phase 2 啟動
- •國泰技術年會：雲的多重宇宙
- •台灣雲端大會：上雲二部曲分享
- •雲端策略發展部正式成立
- •銀行上雲戰情室



生成式 AI 全球爆紅

2023 H1

- •人壽上雲戰情室
- •雲端遷移評估 CCMA 2.0 啟動
- •台灣資安大會：上雲三部曲分享

2023 H2

- •金融業委外法規放寬，正式進入大規模上雲時期
- •Cloud Ready Platform 啟動
- •國泰技術年會：與生成式 AI 共存

2024 H1

- •產險戰情室
- •Cloud Ready for 證券 啟動
- •Cloud Ready Platform 正式上線

國泰從 2021 開始雲端轉型，隨著大規模上雲需求，雲端策略發展部正式成立，便開始自研自建許多工具。因應生成式 AI (GAI) 在全球爆紅，也開始思考如何結合 GAI 提供自動化服務，於是誕生了「雲端智能助手」 點子

Smart Archie 雲端架構生成助手可以解決三大問題

問題

大規模上雲的架構圖需求	架構圖規格不一致	過程缺乏完整記錄
國泰面臨大量上雲需求，需繪製大量遵循技術、政策規範的雲端架構圖，但架構師資源相對稀缺。	不同架構師繪製的架構圖格式不一致專案溝通成本增加。	專案跨組織溝通時，架構圖多僅保留最終版本，更新過程缺乏完整記錄，難以追溯需求變更的過程。

解方

自動生成架構圖內建技術政策	統一架構圖格式與模板	版本控制與需求變更管理
Smart Archie 能自動生成雲端架構圖，同時內建國泰集團的技術政策和標準，確保架構符合公司要求。解決架構師資源稀缺的問題，亦確保每個架構圖都符合規範，減少人工檢查的負擔。	Smart Archie 提供統一的架構圖模板與繪製工具，無論哪位架構師繪製，都能保持相同的格式和標準，降低格式不一致造成的溝通成本，提升跨部門專案協作效率。	Smart Archie 提供完整的版本控制功能，架構圖每次更新都會記錄變更歷程，含變動細節。跨組織溝通時，不僅可以查看最終版本，還可以追溯需求變更具體過程。

AWS 根據國泰金控需求提供企業級的生成式 AI 雲端服務

- AWS GENERATIVE AI STACK

APPLICATIONS THAT LEVERAGE LLMs AND OTHER FMs

Amazon Q Business | Amazon Q Developer | Amazon Q in QuickSight | Amazon Q in Connect

TOOLS TO BUILD WITH LLMs AND OTHER FMs

Amazon Bedrock

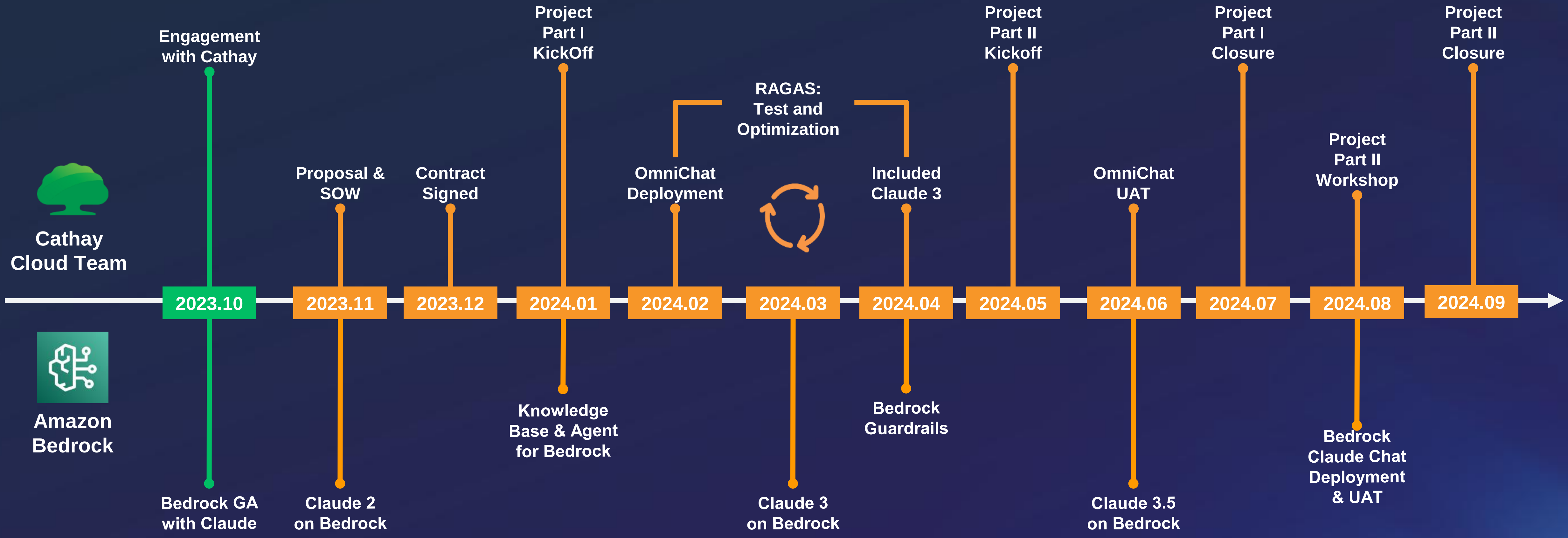
Guardrails | Agents | Studio | Customization Capabilities | Custom Model Import

INFRASTRUCTURE FOR FM TRAINING AND INFERENCE

GPUs | Trainium | Inferentia | SageMaker

UltraClusters | EFA | EC2 Capacity Blocks | Nitro | Neuron

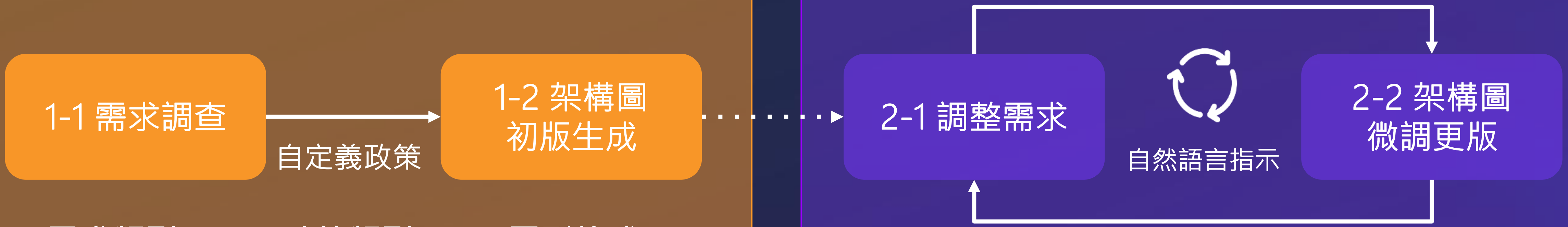
GenAI 技術快速躍進，我們專案跟隨 Bedrock 更新，適時導入最新資源



生成架構圖主要任務：架構圖初版生成、自然語言微調

Part I 架構圖初版生成

Part II 自然語言微調



需求類型：

1.技術需求

2.商業需求

政策類型：

1.基本原則

2.組織政策

圖形格式：

1.Diagrams

2.PlantUML

調整需求類型：

1.雲端服務更換、連線關係

2.問卷問題更新

3.其他抽象需求

輸入：用戶需求與政策規範

Prompt Input: User Requirements & Policy

系統技術需求調查：參考上雲評估問卷與雲端架構師的知識經驗

用戶需求調查範例

模組	問題 (Tech Requirement)
Networking 網路	系統是否需要共用VPC或自建VPC？
	系統是否公開服務給外部網路(Internet)？
	系統是否需要域名服務(DNS)？
	系統是否本身無公開位址(Public IP)，但需要存取到外部網路(Internet)？
	系統是否需要靜態網頁快取(Cache)？
	系統是否須與其他系統串接？
Computing 運算	系統現行的應用系統架構為何？
	系統涵蓋的服務數量？
	Stateful or Stateless?
	是否需要用到GPU?
Database 資料庫	系統需要使用哪種資料庫(DB)？
	資料是否需要快取記憶？
	資料是否需要高可用配置?
Storage 儲存	系統是否需要使用共享儲存裝置？(如NFS)
	系統是否有大於1GB的檔案存放(如文件、圖片、影片、音樂)？
	儲存是否需高可用配置？
CI/CD	是否需要使用雲端的CI/CD?
	是否需要使用雲端的映象檔？(Image)
	是否需要使用雲端的源碼管理？(Source code)
IaC	是否使用 IaC 工具？

輸入需求範例

Tag,Questions (Tech Requirement),Options and Answers
Networking網路,Does the system require a shared VPC or a self-built VPC?,Self-built VPC
Networking網路,Does the system need to serve publicly over the Internet?,Yes
Networking網路,Does the system require domain name services (DNS)?,Yes
Networking網路,Does the system lack a public IP but need access to the Internet?,No
Networking網路,Does the system need static web page caching (Cache)?,Yes
Networking網路,Does the system need to integrate with other systems?,Yes
Computing運算,How many services are covered by the system?,2 (Web, API)
Computing運算,Stateful or Stateless?,Stateless
Computing運算,Does the system require GPU usage?,No
Database資料庫,What type of database does the system need to use?,PostgreSQL
Database資料庫,Does the data need to be cached?,No
Database資料庫,Does the data require high availability configuration?,Yes
Storage儲存,Does the system need to use shared storage devices (e.g., NFS)?,No
Storage儲存,Does the system have files larger than 1GB (e.g., documents, images, videos, music)?,No
Storage儲存,Does the storage require high availability configuration?,No
CI/CD,Does the system need to use cloud CI/CD?,Yes
CI/CD,Does the system need to use cloud images?,Yes
CI/CD,Does the system need to use cloud source code management?,Yes
IaC,Does the system use Infrastructure as Code (IaC) tools?,Yes

自定義政策：參考金融業上雲自律規範、實務手冊與國泰雲端管理政策

Naïve Policy

- There is no unreasonable design; any combination in this system works in the real world.
- There is no redundant design; every component in this system is necessary for meeting requirements.

Organization Policy

Module	Policy
Operational Policy	• Ensure to use CloudWatch for log collection and tracing. No need to connect to other icons; use as a standalone icon. • Ensure to use AWS X-Ray for app monitoring and performance tracing. No need to connect to other icons; use as a standalone icon
Networking Policy	• Ensure to use AWS Site-to-Site VPN for secure cloud-to-on-premises connectivity.
Computing Policy	• Use Amazon EKS (Elastic Kubernetes Service) as the primary managed Kubernetes orchestration platform. • Implement AWS Fargate as the serverless compute engine for EKS. • Do not use EC2 instances for container hosting when dealing with Web, API, or Batch job workloads • Always containerize these workloads: Deploy them in EKS clusters running on Fargate. • Use AWS Lambda if and only if all of the following conditions are met: 1. event-driven. 2. short-lived (typically within minutes). 3. stateless 4. Use case is not suitable for containerization. • In all other cases: Default to using EKS with Fargate for compute resources.
Database Policy	• If not required by requirements expecially for relational database secections, use Amazon RDS for PostgreSQL for managed relational database services. • If not required by requirements expecially for relational database secections, use Amazon RDS for PostgreSQL for managed relational database services. • If not required by requirements expecially for relational database secections, use Amazon RDS for PostgreSQL for managed relational database services.
Security Policy	• Ensure to use AWS IAM for identity and access management. No need to connect to other icons; use as a standalone icon. • Ensure to use AWS KMS for encryption key management. No need to connect to other icons; use as a standalone icon. • Ensure to use AWS CloudHSM for hardware-based key storage. No need to connect to other icons; use as a standalone icon. • Ensure to use AWS Secrets Manager for secure storage of secrets. No need to connect to other icons; use as a standalone icon. • Ensure to use AWS Security Hub for security posture management. No need to connect to other icons; use as a standalone icon. • Ensure to use Amazon Macie for sensitive data discovery and protection. No need to connect to other icons; use as a standalone icon.

輸出：圖形即程式碼

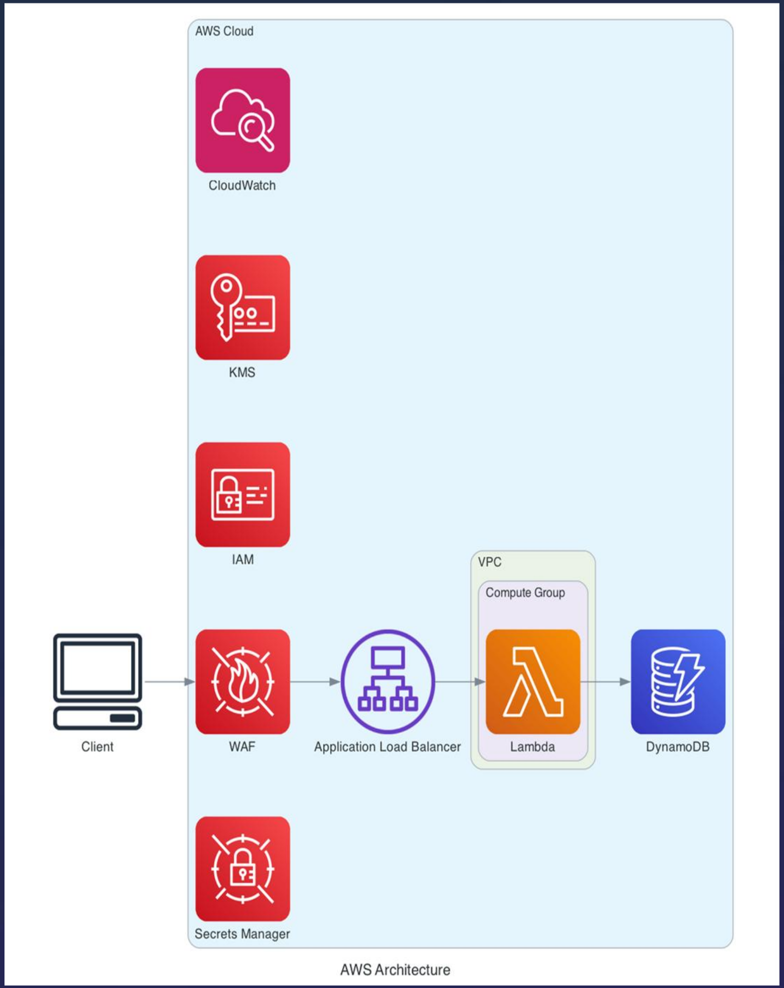
Output: Diagram as Code

圖形生成工具比較：使用 Diagram as code 可以統一架構圖格式標準

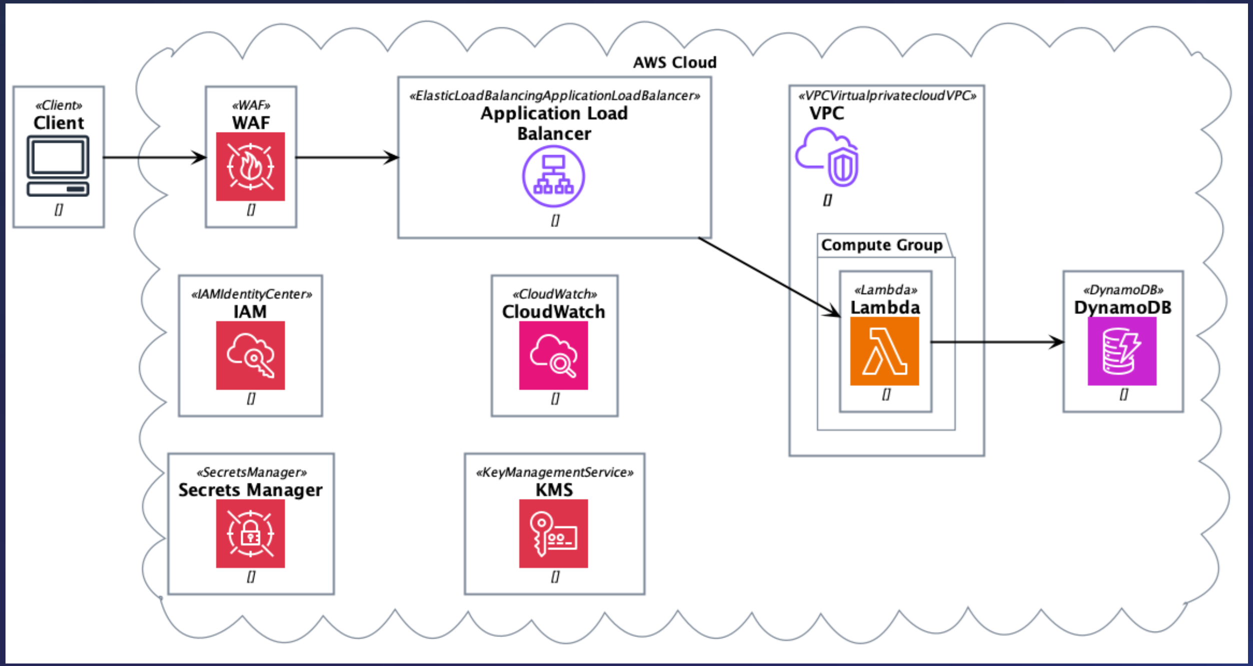
Dimension/Tool	**Draw.io**	**Diagrams**	**PlantUML**
Conducive to automatic generation by LLM	No	Yes	Yes
Easy version control	No	Yes	Yes
Integration with other editors	General (Output Image)	General (Output Image)	High (supported by most editors)
Scope of application	Extensive	Mainly used in architecture diagramming	Extensive
Learning difficulties	Graphic interface	Basic Python program syntax	Need to adapt to the annotation language
AWS component diagrams	Available	Available	Available (via extension package)
How the chart is generated	Drag-and-drop	Python Execution	Annotation Execution
Performance	Performance degradation when large charts	Stable	Stable

Draw.io 資料格式很難透過GenAI技術來生成，因為它們的格式在標記語言中定義結構變得很困難。

Diagrams vs PlantUML：因兩種 DaC 各有優劣，AWS ProServe 顧問案並行研究



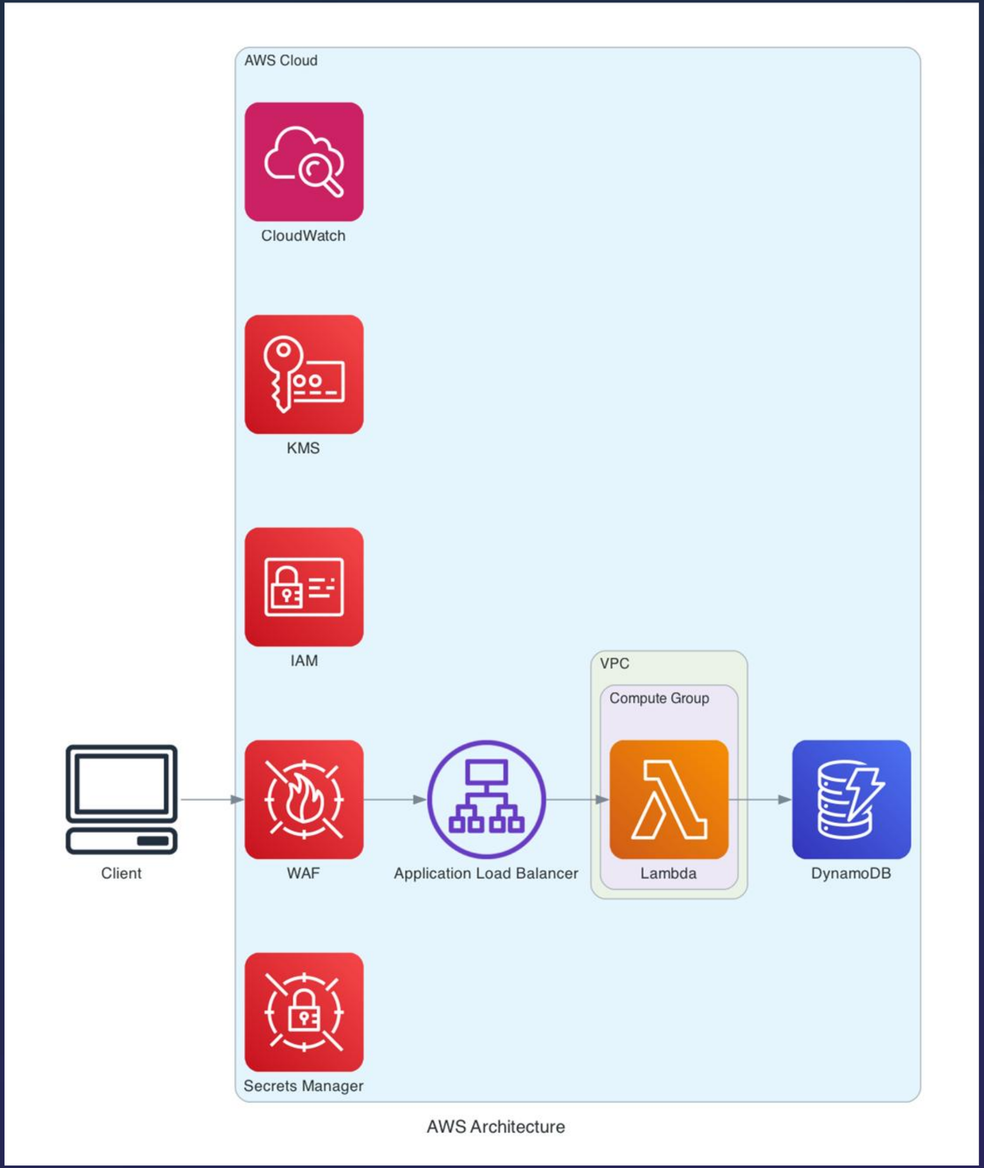
DaC version



AWS Cloud
«Client» Client
«WAF» WAF
«ElasticLoadBalancingApplicationLoadBalancer» Application Load Balancer
«VPCVirtualprivatecloudVPC» VPC
Compute Group
«IAMIdentityCenter» IAM
«CloudWatch» CloudWatch
«Lambda» Lambda
«DynamoDB» DynamoDB
«SecretsManager» Secrets Manager
«KeyManagementService» KMS

Plant UML version

Diagrams: 用 Python 語法定義的雲端架構圖



Diagrams 是使用 Python 程式碼來繪製雲端系統的架構圖。它的誕生是為了在不使用任何設計工具的情況下快速原型設計新的系統架構。作為「程式碼化的架構圖」(Diagrams as Code) ，它允許你在版本控制系統中追蹤架構變更。

優點：

- 大量的 **AWS** 元件圖示可以**原生提供**，涵蓋範圍廣泛
- 以程式碼定義，便於自動化生成
- 以程式碼定義，生成的圖表**穩定性高**，且可以在版本管理系統中有效追蹤

缺點：

- 透過程式碼生成圖表，手動撰寫的學習曲線較陡峭
- 它基於 Python，主要用於架構圖，通用性較低
- 對於更複雜的架構圖，渲染可能會有錯位，但不會影響語義

Diagrams: 用 Python 語法定義的雲端架構圖

```
from diagrams import Cluster, Diagram, Edge
from diagrams.aws.compute import Lambda
from diagrams.aws.database import Dynamodb
from diagrams.aws.network import VPC
from diagrams.aws.security import SecretsManager, WAF, IAM, KMS
from diagrams.aws.network import ALB
from diagrams.aws.management import Cloudwatch
from diagrams.onprem.client import Client
```

定義匯入資源
Define Import Resources

```
with Diagram("output_diagrams", show=False, direction="LR", outformat=["png"]) as diagram:
    client = Client("Client")

    with Cluster("AWS Cloud"):
        with Cluster("VPC"):
            with Cluster("Compute Group"):
                lambda_func = Lambda("Lambda")

            dynamodb = Dynamodb("DynamoDB")
            secrets_manager = SecretsManager("Secrets Manager")
            waf = WAF("WAF")
            alb = ALB("Application Load Balancer")
            iam = IAM("IAM")
            kms = KMS("KMS")
            cloudwatch = Cloudwatch("CloudWatch")
```

定義分群與雲服務
Define Cluster and Nodes

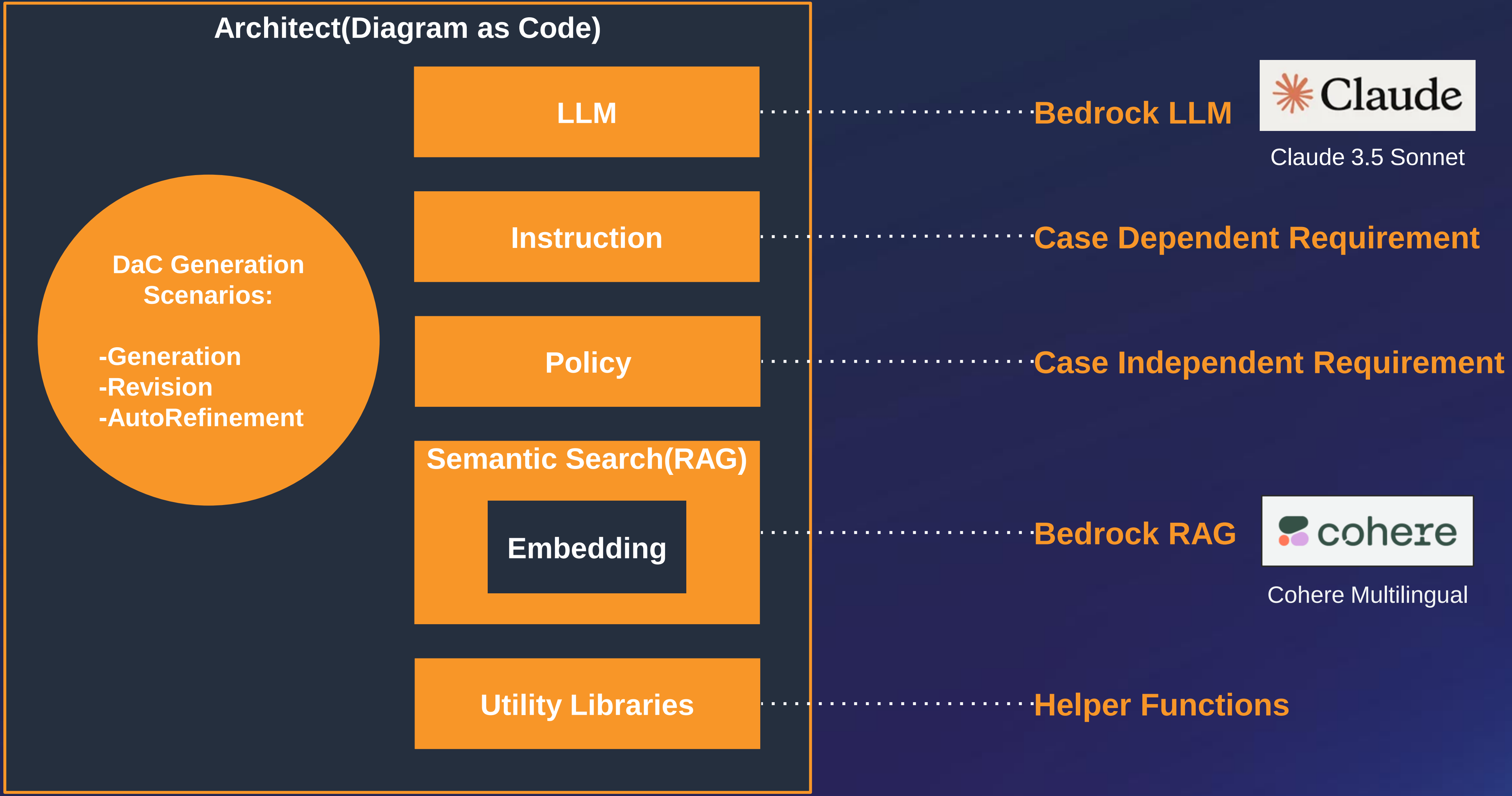
```
    client >> Edge(label="") >> waf
    waf >> alb
    alb >> lambda_func
    lambda_func >> dynamodb
```

定義雲服務間的連線關係
Define Edges

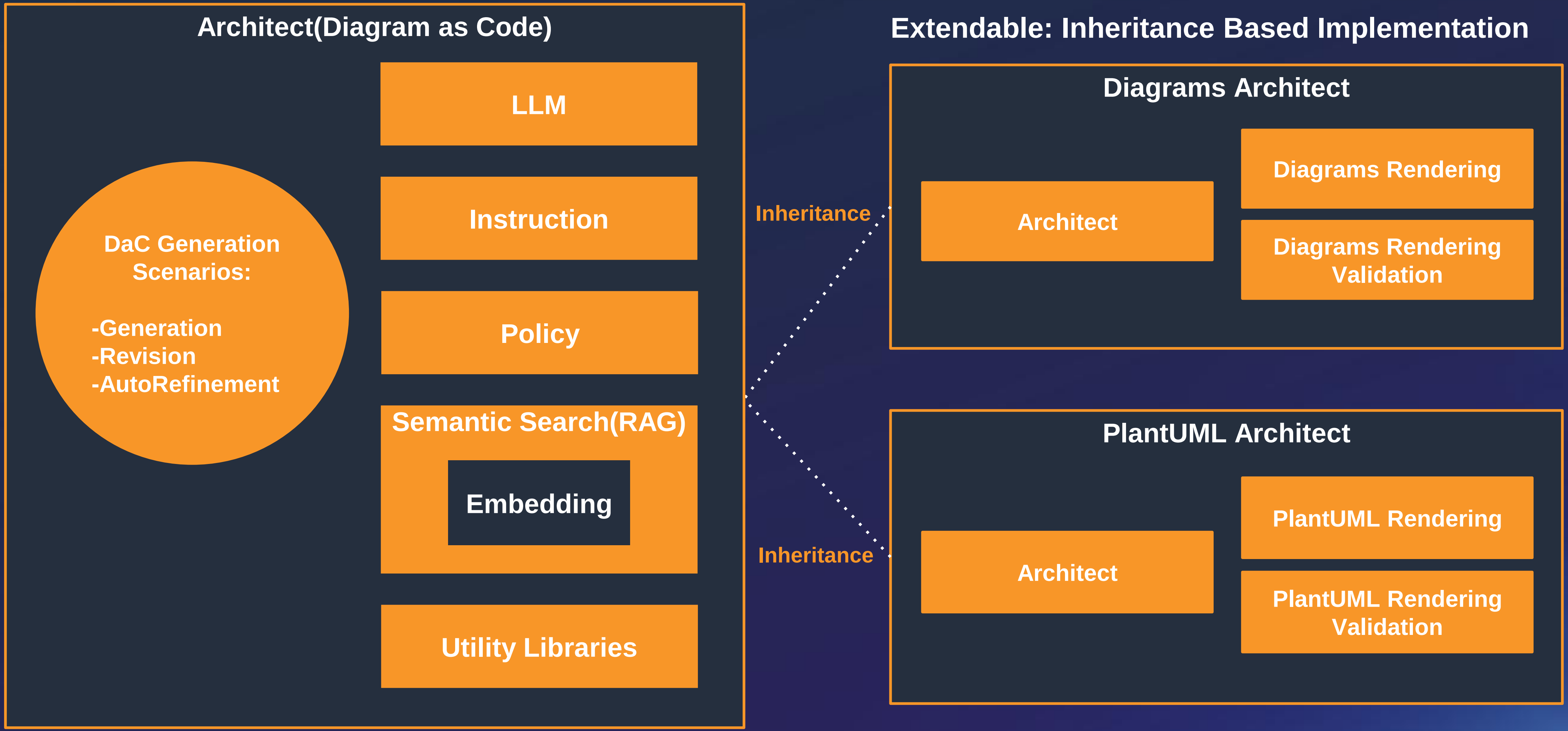
功能模組化

Function Modularization

功能模組：使用 Python 撰寫，並模組化功能，方便後續發展與拆換



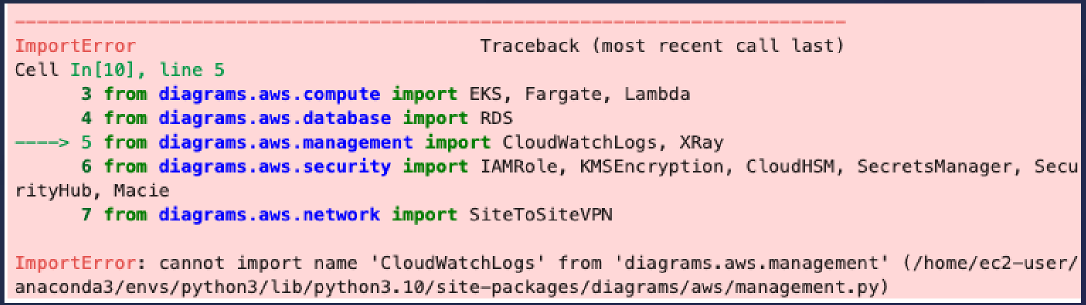
功能模組：使用 Python 撰寫，並模組化功能，方便後續發展與拆換



實驗結果：Prompt & RAG

Experiment Result : Prompt & RAG

我們先用最簡單的指示試試看...



```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[10], line 5
      3 from diagrams.aws.compute import EKS, Fargate, Lambda
      4 from diagrams.aws.database import RDS
----> 5 from diagrams.aws.management import CloudWatchLogs, XRay
      6 from diagrams.aws.security import IAMRole, KMSEncryption, CloudHSM, SecretsManager, Secu
rityHub, Macie
      7 from diagrams.aws.network import SiteToSiteVPN

ImportError: cannot import name 'CloudWatchLogs' from 'diagrams.aws.management' (/home/ec2-user/
anaconda3/envs/python3/lib/python3.10/site-packages/diagrams/aws/management.py)
```

Instructions:

Naïve

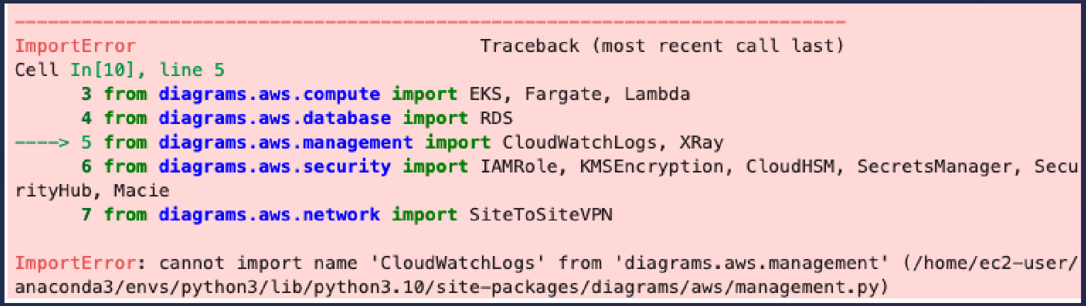
You are an expert in designing architecture diagrams using the Diagrams library.

Architecture requirements:
<requirement>
{requirement}
</requirement>

Output format:
<output>
[Your diagram code here]
</output>

% Compliable	% Avg. Requirement Satisfy	Gen. Time
0.0%	86.67%	8.25 – 19.88 sec

再加入 CoT 思維鏈試試看...



```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[10], line 5
      3 from diagrams.aws.compute import EKS, Fargate, Lambda
      4 from diagrams.aws.database import RDS
----> 5 from diagrams.aws.management import CloudWatchLogs, XRay
      6 from diagrams.aws.security import IAMRole, KMSEncryption, CloudHSM, SecretsManager, Secu
rityHub, Macie
      7 from diagrams.aws.network import SiteToSiteVPN

ImportError: cannot import name 'CloudWatchLogs' from 'diagrams.aws.management' (/home/ec2-user/
anaconda3/envs/python3/lib/python3.10/site-packages/diagrams/aws/management.py)
```

Instructions:

Naïve

+ Chain of Thought (CoT)

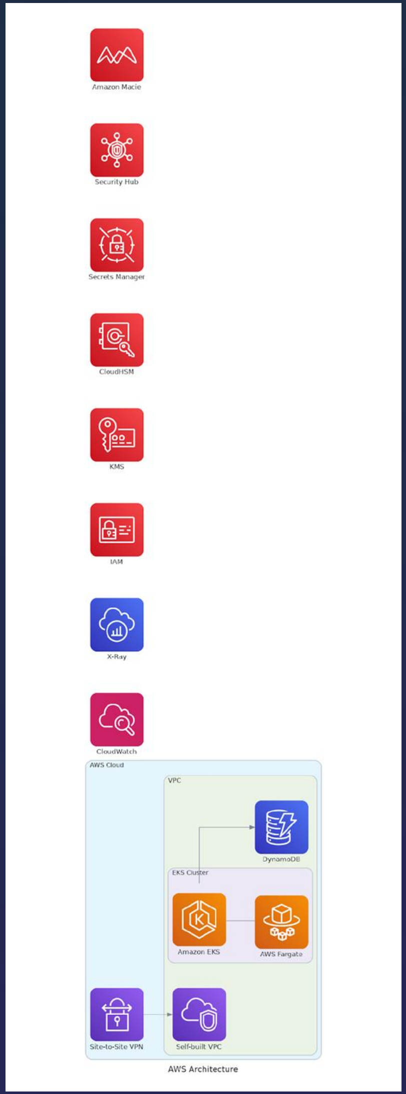
Follow these steps to generate diagram code:
1. Import the Cluster, Diagram class, and relevant node classes from the required resource types and providers.
2. Define the diagram name and attributes.
3. Create source nodes that are not in any cluster.
4. Create clusters, sub-clusters, and required nodes within them.
5. Add edges to indicate data flow directions.

Your task:
- Create a diagram code that represents the architecture described in the <requirement> section.
… 3 Lines

Important notes:
- Do not use import statements that are not in the icon libs (e.g., don't use "from diagrams.aws.management impor CloudWatchLogs" or "from diagrams.aws.network import Subnet").

% Compliable	% Avg. Requirement Satisfy	Gen. Time
0.0%	89.83%	8.05 – 21.92 sec

再加入 RAG 機制引入 Icon 語法...



Instructions:

Naïve

+ Chain of Thought (CoT)

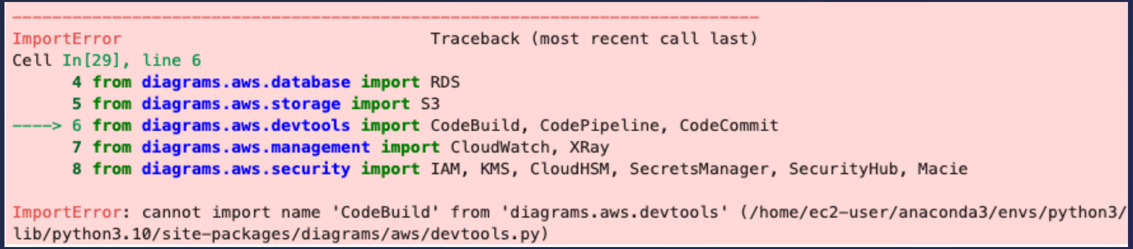
RAG:

+ Icon

Below listed some library import patten seems useful:
from diagrams.aws.compute import ElasticKubernetesService
from diagrams.aws.storage import FsxForWindowsFileServer
from diagrams.aws.iot import IotServo
from diagrams.aws.compute import ElasticContainerServiceService
from diagrams.aws.compute import ElasticContainerServiceContainer

% Compliable	% Avg. Requirement Satisfy	Gen. Time
8.3%	81.67%	8.33 – 19.96 sec

RAG 有效，那我們拿掉 CoT 思維鏈看看...



```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[29], line 6
      4 from diagrams.aws.database import RDS
      5 from diagrams.aws.storage import S3
----> 6 from diagrams.aws.devtools import CodeBuild, CodePipeline, CodeCommit
      7 from diagrams.aws.management import CloudWatch, XRay
      8 from diagrams.aws.security import IAM, KMS, CloudHSM, SecretsManager, SecurityHub, Macie

ImportError: cannot import name 'CodeBuild' from 'diagrams.aws.devtools' (/home/ec2-user/anaconda3/envs/python3/
lib/python3.10/site-packages/diagrams/aws/devtools.py)
```

Instructions:

Naïve

~~**+ Chain of Thought (CoT)**~~

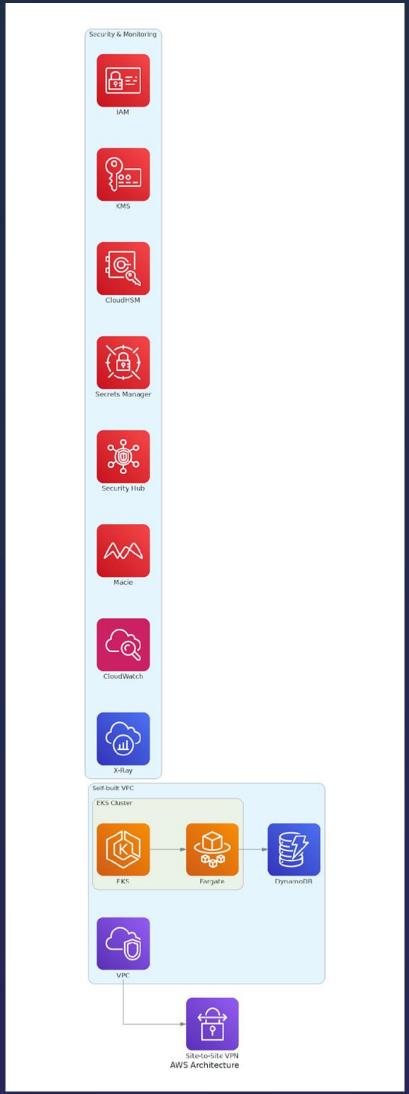
RAG:

+ Icon

Below listed some library import patten seems useful:
from diagrams.aws.compute import ElasticKubernetesService
from diagrams.aws.storage import FsxForWindowsFileServer
from diagrams.aws.iot import IotServo
from diagrams.aws.compute import ElasticContainerServiceService
from diagrams.aws.compute import ElasticContainerServiceContainer

% Compliable	% Avg. Requirement Satisfy	Gen. Time
0.0%	92.11%	9.13 – 21.92 sec

我們直接把 Icon 語法放進 Prompt，然後拿掉 RAG 看看...



Instructions:

Naïve

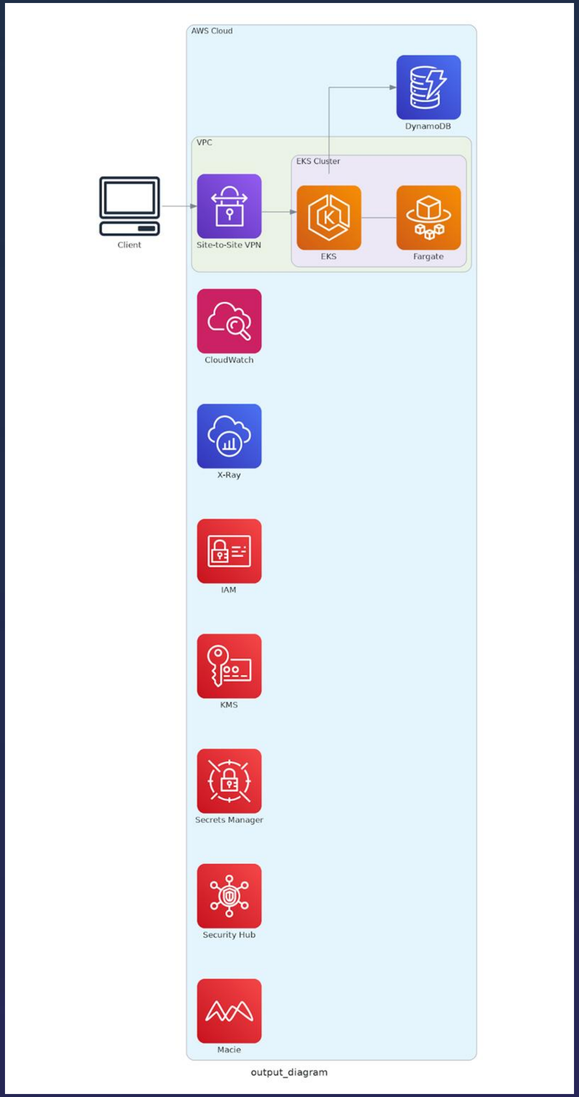
+ Chain of Thought (CoT)

+ Icon

Avaliable library / import statements:
<library>
from diagrams.aws.analytics import Athena
...531 Lines
</library>

% Compliable	% Avg. Requirement Satisfy	Gen. Time
25.00%	90.72%	9.13 – 21.67 sec

我們加幾個範例 DaC 看看...



Instructions:

Naïve

+ Chain of Thought (CoT)

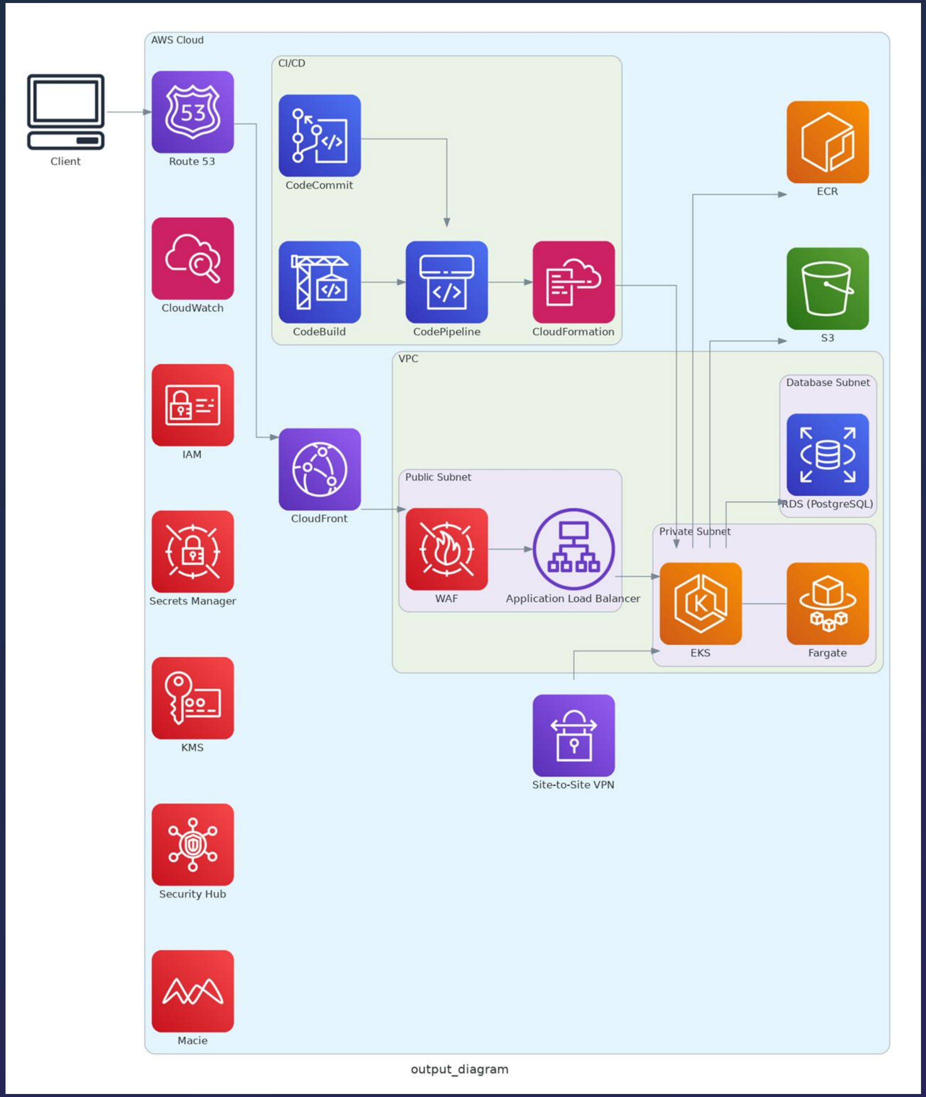
+ Icon

+ Few Shot

Example diagram code:
<examples>
<example0>
…63 Lines Example Diagram as Code
</example0>
… 227 Lines DaC example
</examples>

% Compliable	% Avg. Requirement Satisfy	Gen. Time
83.33%	89.33%	8.81 – 21.54 sec

範例有效，那我們連 Input requirement 都給範例看看 ..



Instructions:

Naïve

+ Chain of Thought (CoT)

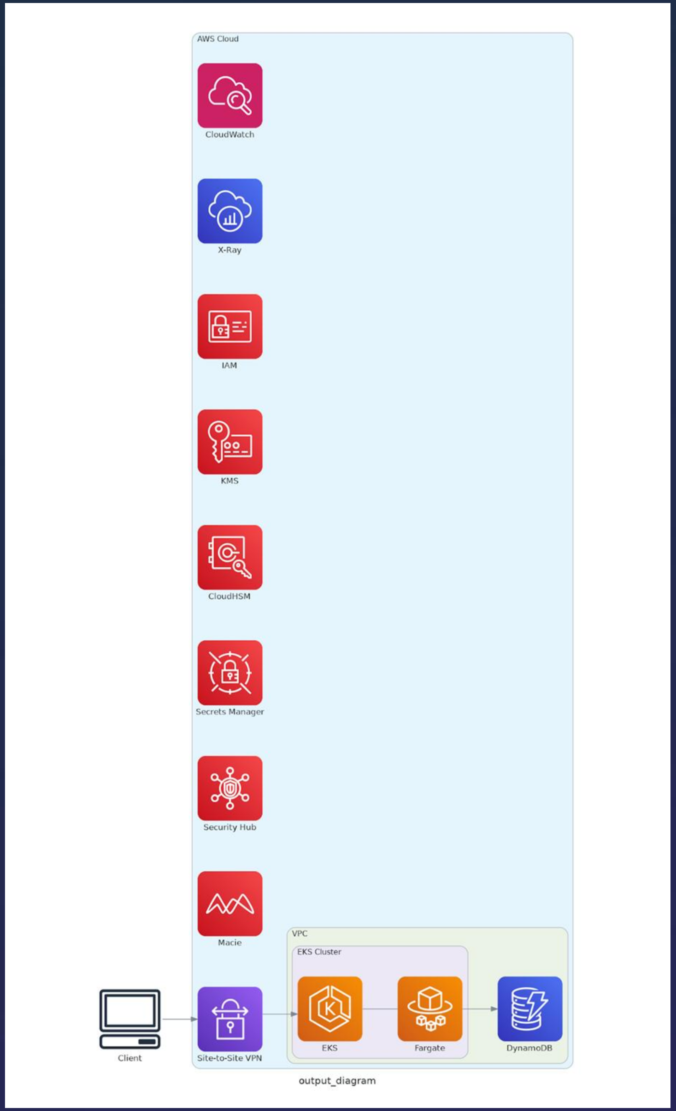
+ Icon

+ Few Shot with Input/Output

Example diagram code:
<examples>
<input0>
…30 Lines Input Requirement
</input0>
<output0>
… 63 Lines DaC example
</output0>
… 390 Lines
</examples>

% Compliable	% Avg. Requirement Satisfy	Gen. Time
75.00%	84.72%	9.17 – 22.21 sec

我們再加回 RAG Icon，編譯率可以到 100% 了！



Instructions:

Naïve

+ Chain of Thought (CoT)

+ Icon

+ Few Shot

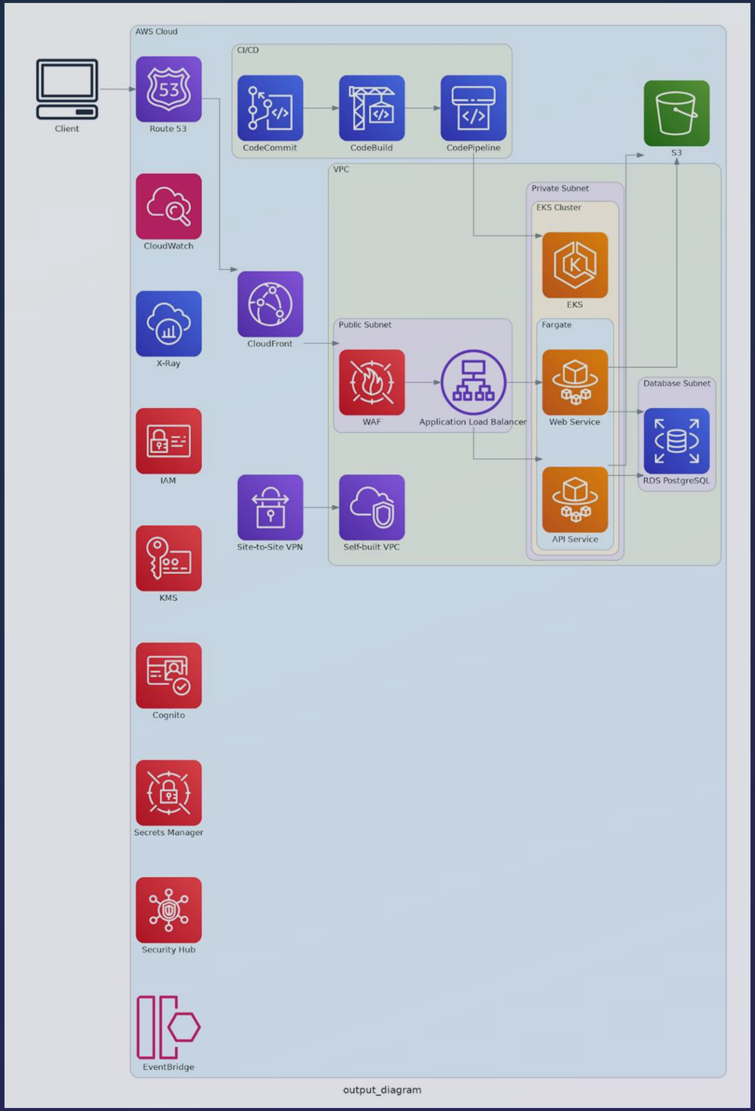
RAG:

+ Icon

Below listed some library import patten seems useful:
from diagrams.aws.compute import ElasticKubernetesService
from diagrams.aws.storage import FsxForWindowsFileServer
from diagrams.aws.iot import IotServo

% Compliable	% Avg. Requirement Satisfy	Gen. Time
100%	81.5%	10.6 – 23.25 sec

我們再加入自動修正機制，提升需求配適度！



+ Auto Refinement

Instructions:

Naïve

+ Chain of Thought (CoT)

+ Icon

+ Few Shot

RAG:

+ Icon

% Compliable	% Avg. Requirement Satisfy	Gen. Time
100.0%	98.17%	120 – 170.5 sec

經過 10 幾次 Prompt 優化與交叉比對，我們找到實驗結果最好的組合

	Naïve	CoT	Instruction(CoT) + RAG(Icon)	Naïve + RAG(Icon)	Instruction(CoT + Icon)	Instruction(CoT + Icon + Few Shot)	Instruction(CoT + Icon + Few Shot w/ IO)	Instruction(CoT + Icon + Few Shot) + RAG (Icon)	Instruction(CoT + Few Shot) + RAG (Icon)	Instruction(CoT + Icon + Few Shot) + RAG (Icon) + Auto-Refinement	Instruction(CoT + Few Shot) + RAG (Icon) + Auto-Refinement
% Compliable	0.00%	0.0%	8.3%	0.0%	25.00%	83.33%	75.00%	100%	100%	100%	100%
% Avg. Requirement Satisfy	86.67%	89.83%	81.67%	92.11%	90.72%	89.33%	84.72%	81.5%	81.02%	98.17%	97.22%
Avg. Time	8.25 - 19.88 sec	8.05 - 21.92 sec	8.33 - 19.96 sec	9.13 – 21.92 sec	9.13 – 21.67 sec	8.81 – 21.54 sec	9.17 – 22.21 sec	10.6 – 23.25 sec	7.66 – 21.09 sec	120 – 150.5 sec	73 – 130 sec

- **思維鏈(Chain of Thought, CoT)、少量範例教學(Few-Shot Instruction)、圖示檢索增強生成(icon RAG)**可以提升整體品質。
- 自動修正(Auto-Refinement)和指令中的圖示庫也能改善生成品質，但會帶來較高的計算成本。

Smart Archie Demo

結論與展望

POC 實驗成功，解構 LLM 能力與如何控制，就能生成穩定的架構圖！

目前 LLM 的能力 (Claude 3.5)

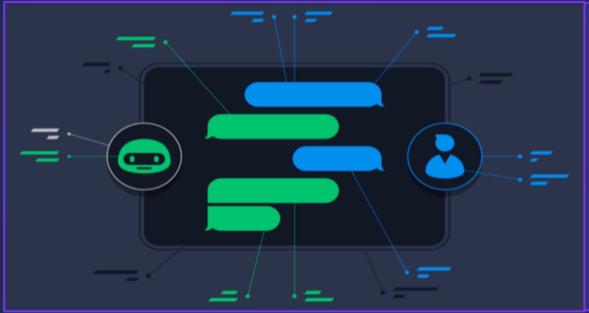
- Claude 3.5 能夠理解服務需求並有效生成雲端架構，但撰寫 **Diagram as Code 需要適當的引導**。
- **清晰、精確的需求** 對於避免模糊性至關重要，這有助於優化過程，並在提供範例時表現最佳。
- 目前圖表生成和修正的時間大約為 **20 秒到 2 分鐘**，可提高 SDLC 的效率，隨著 LLM 的發展有望進一步加快處理速度。

如何控制 LLM 輸出的不穩定性

- **Prompt:**
  - **Chain of Thought**：架構設計適合逐步思考的過程
  - **Icon Import Sample:** 提供基礎 Icon 語法
  - **Few Shot Example:** 提供基礎 DaC 範本
- **RAG:** 檢索進階 Icon 語法與 DaC 範本
- **Auto refinement:** 自動修正機制

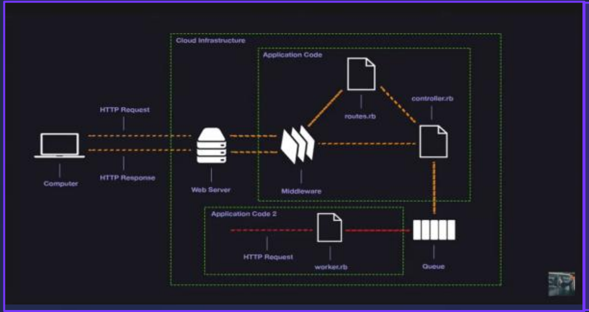
Cloud Ready Platform 未來應用

Smart Archie 已經學會了知識解惑與架構設計的雲端架構師職能...



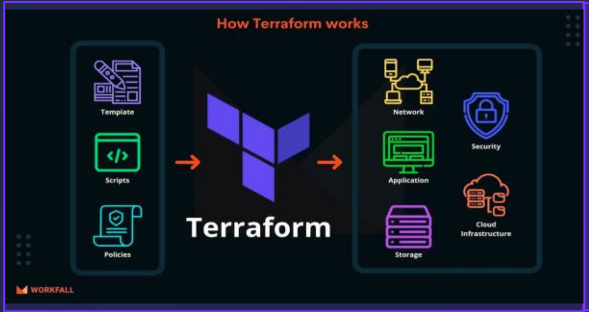
情境一：雲端知識解惑

- 雲端技術輸出文章
- 雲端管理辦法諮詢



情境二：架構評估設計

- 雲端架構範本生成
- 雲端架構設計評估



情境三：環境實作測試

- IaC 程式碼範本生成
- IaC 程式碼品質審查

所有 AI 功能皆整合在 Cloud Ready Platform，一站式加速雲端轉型！

	上雲前		上雲中		上雲後
資安面	資安評估與發展計畫		資安框架與相關檢測		治理實施
系統面	系統遷移評估 Cloud Migration Assessment		架構決策紀錄 Architecture Design Record	變更審查委員會 Change Advisory Board	系統上線
組織面	知識解惑、智能諮詢 **Smart Archie**		雲端架構圖智能生成 **Smart Archie**	IaC 智能生成 **Smart Archie**	實務培訓
任務與產出	**Questionnaire / Assessment / Report**	**Dashboard / Strategy / Plan**	**Architecture** ● IAM ● Network ● Application	**Development & Test** ● Application ● CI/CD ● IaC	**Production** • Release note • SRE • FinOps

Cloud Ready Platform
一站式雲治理平台

Thank you!