Architecting GenAI

**Difficulty:** Level 100

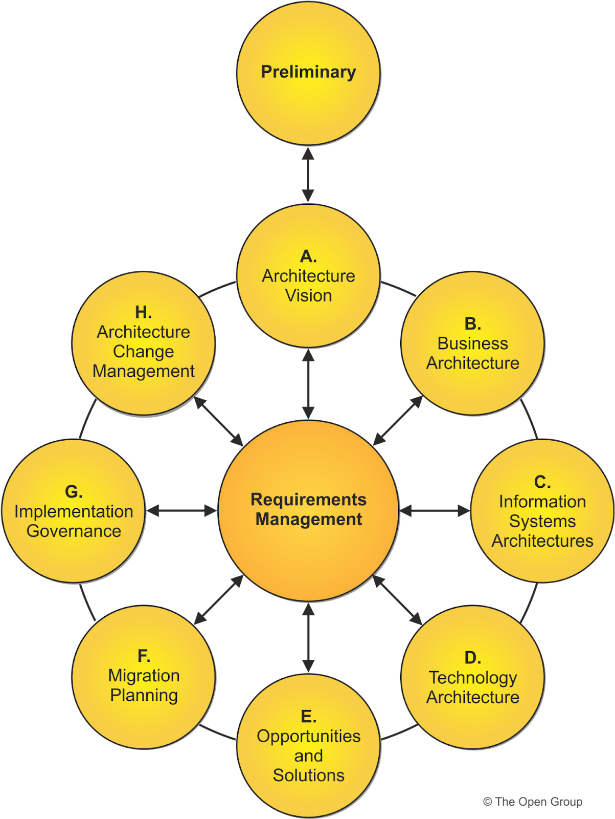
**Architecting Link:** [**Lucid Chart**](https://lucid.app/lucidchart/c7945e5c-a177-410c-8a24-2481a29563f1/edit?viewport_loc=-1795%2C-201%2C2217%2C1076%2CwfKOB.kyPhrS&invitationId=inv_aef7b30b-bf51-4cc6-a320-4e5f30574496)

**Business** **Goal:**

As a Solution Architect after consulting with real-world AI Engineers, you have been tasked to create architectural diagram(s) that serve as a teaching aid to help stakeholders understand their key components of GenAI workloads. The outcome is to help let stakeholders visualize possible technical paths, technical uncertainty when adopting GenAI.

We are guiding key stakeholders through the technical landscape without directly prescribing solutions, while fostering informed discussions about infrastructure choices, integration patterns, and system dependencies across the organization.

We can use all levels of technical diagramming to achieve our goal.



* <https://www.opengroup.org/togaf>
* <https://c4model.com/>
* <https://medium.com/@nolomokgosi/conceptual-logical-and-physical-design-c24100846931>

**Technical Considerations**

Let’s assume we are following the three levels of diagramming:

* **Conceptual** — a high level diagram that is used to communicate to key stakeholders the business solution we are implementing
* **Logical** — a mid level diagram that describes the key technical components but not requiring detailed parameters so we can quickly rearchitect and communicate to our technical team the current workload
* **Physical** — a low level diagram that details all possible parameters and connections used by engineers/developers to accurately implement a solution (e.g. ARNs for resources, IP addresses, etc)

**Architectural/Design Considerations**

* **Requirements, Risks, Assumptions, & Constraints:**
  + **Requirements** are the specific needs or capabilities that the architecture must meet or support.
    - Categories:
      * Business Requirements: Business goals and objectives
      * Functional Requirements: Specific capabilities the system must have
      * Non-functional Requirements: Performance, scalability, security & useability
      * Tooling: GenAI vs ML
  + **Risks** are potential events or conditions that could negatively affect the success of the architecture or its implementation. Identifying and mitigating risks ensures smoother project delivery.
  + **Assumptions** are things considered to be true without proof at the time of planning and development. These are necessary for decision-making but can introduce risks if proven false.
  + **Constraints** are limitations or restrictions that the architecture must operate within. These are non-negotiable and must be adhered to during design and implementation.
* **Data Strategy**
  + Develop a comprehensive data strategy that addresses:
    - Data collection and preparation
    - Data quality and diversity
    - Privacy and security concerns
    - Integration with existing data systems
* **Model Selection and Development**
  + Choose appropriate models based on your use cases. Consider factors such as:
    - Self Hosted vs SaaS
    - Open weight vs Open Source
    - Input-Output: text-to-text?
    - Number of models needed
    - Number of calls/model
    - Size
    - Evaluation
    - Context window: input, output
    - Fine-tuning requirements
    - Model performance and efficiency
* **Infrastructure Design**
  + Design a scalable and flexible infrastructure that can support GenAI workloads:
    - Leverage cloud platforms for scalability and access to specialized hardware
    - Implement a modular architecture to allow for easy updates and replacements of components
    - Consider hybrid or multi-cloud approaches for optimal performance and cost-efficiency
* **Integration and Deployment**
  + Plan for seamless integration with existing systems and workflows:
    - Develop APIs and interfaces for easy access to GenAI capabilities
    - Implement CI/CD pipelines for model deployment and updates
    - Ensure compatibility with legacy systems
* **Monitoring and Optimization**
  + Establish robust monitoring and optimization processes:
    - Implement logging and telemetry for model performance
    - Set up feedback loops for continuous improvement
    - Develop KPIs to measure the business impact of GenAI solutions
    - Depending on the location, set up billing alerts to monitoring usage over time
* **Governance and Security**
  + Implement strong governance and security measures:
    - Develop policies for responsible AI use
    - Implement access controls and data protection measures
    - Ensure compliance with relevant regulations and industry standards
* **Scalability and Future-Proofing**
  + Design the architecture with scalability and future advancements in mind:
    - Use containerization and microservices for flexibility
    - Implement version control for models and data
    - Plan for potential increases in computational requirements

**Business Considerations**

* **Use Cases:**
  + Start by clearly defining the specific use cases for GenAI within your organization:
  + Identify the business problems you're trying to solve and the desired outcomes
* **Complexity:** As a stakeholder how do I understand the level of complexity integrating GenAI (specifically) LLMs into our workload?
  + eg. How many moving parts will it add to our workload?
  + eg. Is this set and forget, or do we need people to monitor and maintain these components regularly?
* **Key levers of cost:** As a stakeholder how can I understand the key costs to running GenAI at a glance?
  + eg. Size of servers
  + eg. Size of models
* **Lock-in:** What is a technical path we should consider so we are not locked-in to a vendor solution.
  + eg. How do we avoid rug pulls? (The cost going up being locked into a solution)
  + Eg. How do we position our technical stack so we can transition to better models or solutions?
* What essential components should be conveyed as necessary when deploying a GenAI workload for production
  + Guardrails
  + Evaluations
  + Sandboxing via Containers

**LLM specific thoughts:**

1- Choosing a Model:

* input-output modalities
* open source vs proprietary
* SaaS or self hosted
* context window
* cost

2- Enhance Context:

* Some options: Direct context injection or setting up a knowledge base?
* Some criteria to evaluate:
  + Size of input (one document or chunks of several docs)
  + Model context window
  + One time use or repeated use of information
  + Prototyping or scalable system?

3- Guardrails:

* Input guardrails
* Output guardrails
* Implementation

4- Abstract Model access

* Models & patterns to support
* Modalities to support

5- Caches

* Caching Strategy
* Cache levels
* Invalidation rules
* Storage options
* Hit rate optimization

6- Agents

* Actions to be executed
* System integration

**Understanding the Document: Architecting GenAI for Beginners**

This document is a **high-level guide** for designing **Generative AI (GenAI) architectures**. It is meant for **Solution Architects** to create diagrams and frameworks that help businesses understand how to deploy and manage GenAI workloads.

I'll break it down in **simple terms** so you can grasp the **core ideas** as a beginner in software development.

**1️⃣ What is the Business Goal?**

The main goal is to create **architectural diagrams** that help stakeholders (business & technical teams) understand:  
✅ What components are needed to run **GenAI** (like ChatGPT or AI-powered tools)?  
✅ What are the **risks, challenges, and trade-offs** in setting up GenAI?  
✅ How does GenAI fit into the company’s **existing infrastructure**?  
✅ How can we make the system **scalable, cost-efficient, and future-proof**?

This is **not** about prescribing one fixed solution. Instead, it’s about **helping businesses explore options** for integrating GenAI.

**2️⃣ Three Levels of Architecture Diagrams**

When designing **GenAI systems**, architects use three **types of diagrams**:

**📌 Conceptual Diagram (High-Level)**

* **Audience**: **Non-technical stakeholders** (business leaders, product managers)
* **Purpose**: Shows the overall **business solution**
* **Example**: “We want to build a chatbot that helps customers with support questions.”

**📌 Logical Diagram (Mid-Level)**

* **Audience**: **Technical managers, solution architects**
* **Purpose**: Shows the **key components** without deep details
* **Example**: “The chatbot needs an AI model, a data pipeline, and a cloud service.”

**📌 Physical Diagram (Low-Level)**

* **Audience**: **Developers, engineers, DevOps**
* **Purpose**: Shows exact **infrastructure details** (e.g., server specs, IPs, APIs, databases)
* **Example**: “The chatbot runs on AWS Lambda, connects to OpenAI’s API, and stores logs in MongoDB.”

**3️⃣ Key Considerations in GenAI Architecture**

**📌 1. Requirements, Risks, Assumptions & Constraints**

* **Business Requirements** – What is the goal? (e.g., AI-powered customer service)
* **Functional Requirements** – What should the system do? (e.g., answer FAQs in real-time)
* **Non-functional Requirements** – Performance, security, cost, usability

⚠️ **Risks**: What could go wrong? (e.g., data leaks, high costs, model inaccuracy)  
⚙️ **Assumptions**: What do we assume is true? (e.g., AI models will be accurate)  
⛔ **Constraints**: What are the limits? (e.g., budget, compliance regulations)

**📌 2. Data Strategy**

Since **GenAI models rely on data**, we must **handle data properly**: ✔ **How do we collect & prepare data?**  
✔ **Is the data diverse & high-quality?**  
✔ **How do we protect sensitive data?** (security & privacy)  
✔ **Can the AI integrate with existing company data sources?**

**📌 3. Model Selection & Development**

When choosing a **GenAI model**, consider:  
🤔 **Self-Hosted vs SaaS** – Do we **run our own AI** or **use an external service** (like OpenAI)?  
🔓 **Open-Source vs Proprietary** – Do we use a **free** AI model (like LLaMA) or a **paid** model?  
🔢 **Input & Output** – Is the AI handling **text, images, or both**?  
📏 **Model Size & Performance** – How big should the model be?

**📌 4. Infrastructure Design**

GenAI needs **powerful computing resources**. We must decide:  
✔ Should we use **cloud computing** (AWS, Google Cloud, Azure)?  
✔ Should we design a **flexible system** so we can switch AI models?  
✔ Should we use a **hybrid approach** (mix of cloud & local servers)?

**📌 5. Integration & Deployment**

AI models need to be **connected** to existing company systems. To do this:  
✔ Build **APIs** to let different systems talk to the AI  
✔ Use **CI/CD pipelines** to automatically update models  
✔ Ensure **compatibility with old systems**

**📌 6. Monitoring & Optimization**

After deployment, we must **track AI performance**:  
📊 **Set up logging & monitoring** – How well is the AI working?  
📉 **Measure Key Performance Indicators (KPIs)** – Is AI helping the business?  
💰 **Track costs** – How much are we spending on AI?

**📌 7. Governance & Security**

AI must follow **rules & security policies**:  
✅ Who **controls access** to the AI system?  
✅ How do we **protect customer data**?  
✅ Does the AI follow **legal regulations** (like GDPR)?

**4️⃣ Business Considerations**

**📌 1. Use Cases**

* What **business problems** are we solving with GenAI?
* Example: “We want AI to help **automate email responses** to customers.”

**📌 2. Complexity**

* How difficult is it to integrate GenAI into our company’s tech stack?
* Example: “Will we need to hire **AI engineers** to maintain the system?”

**📌 3. Cost**

GenAI can be expensive!

* Example:
  + Running **big AI models** requires **powerful (expensive) servers**
  + More **API calls** = Higher **operating costs**

**📌 4. Avoiding Lock-In**

* If we **depend too much on one AI vendor**, what happens if they **increase prices**?
* Should we design a **flexible system** so we can switch to a different AI provider later?

**5️⃣ LLM-Specific Considerations (Large Language Models like GPT)**

**📌 1. Choosing a Model**

* **Open-source vs Proprietary** – Do we need full control or an easy-to-use service?
* **Cost & Performance** – Does the model fit our budget?

**📌 2. Enhancing Context**

* Should we **store extra knowledge** for the AI? (like FAQs or company documents)

**📌 3. Guardrails**

* How do we **prevent AI from making bad responses**? (e.g., **filtering harmful content**)

**📌 4. Caching**

* How do we **store AI-generated responses** to reduce costs?

**📌 5. Agents**

* Can AI perform **automated tasks** beyond text generation?

**🚀 Summary (Key Takeaways)**

📌 **GenAI architecture helps businesses design scalable AI solutions.**  
📌 **Three levels of architecture diagrams** help different teams understand AI deployment.  
📌 **Data, security, costs, and monitoring** are critical for success.  
📌 **AI models must be chosen carefully** based on **business needs, flexibility, and cost.**  
📌 **Avoiding vendor lock-in** ensures we don’t get trapped in an expensive AI provider.

**💡 Next Steps for You (as a Beginner)**

✔ **Try drawing a "Conceptual Diagram"** – What AI system would you design?  
✔ **Explore AI models** – Try OpenAI, Google Gemini, or open-source LLaMA models.  
✔ **Experiment with cloud AI tools** like AWS SageMaker or Google Vertex AI.

Would you like **help creating a basic AI architecture diagram** to practice? 😊