**Interview Questions for Senior Architect – unfinished\_architecture\_2**

Use these questions during an interview to explore the candidate's approach to designing and evolving this system.

**🔐 Authentication and Authorization**

1. **What authentication pattern would you recommend here—OAuth 2.0, API keys, JWT, mTLS? Why?**
2. **How would you propagate the identity of the user or calling system through the layers?**
3. **How would you enforce different access levels (e.g., pharmacy staff vs. broker tools)?**

**☁️ Compute Strategy**

1. **Would you implement the application logic using EC2, containers, or serverless? Why?**
2. **Given the traffic spike near year-end (from 1,000 to 400,000 requests/day), how would you scale the compute layer?**
3. **Would you split the formulary logic into microservices or keep it monolithic for now? Explain your tradeoffs.**

**📦 Data Storage**

1. **What type of database would you choose for formulary data—relational, NoSQL, or in-memory? Justify your selection.**
2. **How would you keep the formulary data fresh and resilient to corruption or sync lag?**
3. **What’s your approach to logging and storing audit data for 365 days with compliance constraints (HIPAA, etc.)?**

**🧰 Elasticity and Fault Tolerance**

1. **What would you do to ensure the system handles 400x traffic surges with high availability?**
2. **How would you monitor performance and detect degraded service during these spikes?**
3. **Would you use a queue or buffer anywhere in this design to offload peak traffic? Why or why not?**

**💸 Cost Optimization**

1. **How would you optimize cost for a system that is idle most of the year but bursts heavily for a few days?**
2. **What components would benefit most from a serverless or spot instance approach?**

**🔁 Future Enhancements**

1. **If we later introduced personalized responses based on the member’s plan, how would you modify the architecture?**
2. **What design decisions now would make it easier to introduce file-based input or ML classification in the future?**

Unfinished\_architecture –

Okay, understood! No AWS extensions, just plain PlantUML for a conceptual architecture diagram. This will give you a very clean, generic representation that focuses purely on the architectural components and their relationships, without any vendor-specific icons or styling.

Here's the PlantUML code:

Code snippet

@startuml

package "Larger Network" {

rectangle "Subnet" {

component "API Gateway" as API

package "Compute Layer" {

component "EC2 Instance 1" as EC2\_1

component "EC2 Instance 2" as EC2\_2

component "EC2 Instance 3" as EC2\_3

API -[dashed]-> EC2\_1

API -[dashed]-> EC2\_2

API -[dashed]-> EC2\_3

}

component "Authentication Service" as Auth

API -[dashed]-> Auth

component "Data Storage" as Storage

"Compute Layer" -[dashed]-> Storage

}

}

@enduml

**Explanation of the Diagram (Plain PlantUML):**

* **package "Larger Network"**: Represents the broader network environment, similar to a VPC.
* **rectangle "Subnet"**: A nested box indicating a specific subnet within the larger network.
* **component "API Gateway" as API**: A generic component labeled "API Gateway." as API provides a short alias for drawing connections.
* **package "Compute Layer"**: A logical grouping for your application's compute resources.
* **component "EC2 Instance 1" as EC2\_1**: Generic components representing your application instances.
* **component "Authentication Service" as Auth**: A generic component for the authentication mechanism.
* **component "Data Storage" as Storage**: A generic component for the persistent storage.
* **-[dashed]->**: A dashed arrow indicating a dependency or interaction flow.

**Business Case (Same as before):**

A health insurance company needs to provide a service for querying their drug formulary. On a typical day, they expect around 1,000 requests. However, during the last three business days of the year, due to policy renewals and open enrollment deadlines, the request volume is expected to spike dramatically, potentially reaching up to 400,000 requests per day. The service needs to be reliable, secure, and cost-effective, handling the significant fluctuations in demand.

**Questions for a Senior Solutions Architect Interview (Same as before):**

**Elasticity & Scalability:**

1. Given the significant difference in request volume between typical days and peak periods, how would you design the "Compute Layer" of this architecture to handle this elasticity? What are the pros and cons of your chosen approach?
2. How would you ensure the "API Gateway" can handle the peak load of 400,000 requests per day? What scalability considerations are important for this component?
3. How would you monitor the system's performance and automatically scale resources up or down based on the observed load? What metrics would you focus on?

**Compute Options (Serverless vs. EC2 & Containerization):**

1. We've represented the compute layer with generic instances. What are the potential benefits and drawbacks of using a serverless compute option (like AWS Lambda or Fargate) for this workload, either for the entire service or specific parts of it, compared to traditional virtual machines?
2. How could containerization (using Docker and potentially an orchestration service) be incorporated into this architecture? What advantages would it offer?
3. If we were to use containers, how would you manage the deployment, scaling, and updates of these containers?

**Authentication & Security:**

1. This architecture includes a generic "Authentication Service." What specific technologies or approaches would you consider implementing for authentication in this scenario? Discuss the trade-offs between different options.
2. How would you secure the communication between the "API Gateway" and the backend "Compute Layer" instances?
3. What security considerations would be paramount when designing the authentication and authorization mechanisms for accessing sensitive health insurance formulary data?

**Storage & Data Management:**

1. We have a generic "Data Storage" block. What type of database or storage service would you recommend for storing the formulary data, considering the potential for high read volumes during peak periods? Justify your choice.
2. How would you handle data backups and disaster recovery for the formulary data?
3. Are there any caching strategies you would consider implementing to improve the performance and reduce the load on the storage layer, especially during peak times?

**Cost Optimization:**

1. How would you optimize the cost of this architecture, especially considering the vast difference in demand between typical and peak days?
2. What are some strategies for ensuring that we are not over-provisioning resources during normal operation while still being able to handle the peak load?
3. How would you monitor and analyze the costs associated with this architecture on an ongoing basis?