John Elbogen

8/18/2021

CS 470 Final Reflection

https://youtu.be/LSMIoXTyt-Y

- **Experiences and Strengths:** Explain how this course will help you in reaching your professional goals.
    - What skills have you learned, developed, or mastered in this course to help you become a more marketable candidate in your career field?

      I learned how to setup a serverless application and environment utilizing Amazon S3 and Amazon services. Specifically, I have worked with Lambda functions, Identity and Access Management, DynamoDB, S3 buckets and permissions. Outside of Amazon related services I have built skills in docker and containerizing applications.
    - Describe your strengths as a software developer.
      I would say my strengths as a software developer are the patience and focus to solve problems. Personally, it is satisfying to overcome an issue. The debugging process is something I can spend plenty of time doing. In addition, I learn new concepts quickly and understand how to learn about different applications and correlate new and old information.
    - Identify the types of roles you are prepared to assume in a new job.
      I have prepared to assume roles within Cloud Application developer, Cloud automation engineer, and Cloud architect.
- **Planning for Growth:** Synthesize the knowledge you have gathered about cloud services.
    - Identify various ways that microservices or serverless may be used to produce efficiencies of management and scale in your web application in the future. Consider the following:
        - How would you handle scale and error handling?
          Handling scale is straightforward, as the whole purpose of cloud architecture is to provide a way to scale horizontally and vertically by creating servers as needed. As for error handling, I would utilize AWS Step Functions or related service which allows a workflow to support error handling. Essentially it helps respond to errors with conditional logic. In a more general sense, would handle the logic in my Lambda functions to handle errors.
        - How would you predict the cost?
          I would predict the cost of servers by considering the amount of requests that my application will need to make, and then estimating a typical session per user. Then I would estimate the expected number of users and find the total cost

based on those numbers. For example, if my application from open to close calls for 10 requests (like loading web pages on a survey), then I would know each user is expected to send 10 requests. Since serverless is a pay-per-request model, then we can use that to calculate an estimated cost.

  - What is more cost predictable, containers or serverless?

    Serverless is more predictable. Many of the costs of management of services and maintenances are baked into the pay-per-request cost. Containers have costs that are more difficult to calculate as all underlying extra services must be paid for as you utilize them. For example, there is a cost that varies for different maintenances that are manually managed in a container. The price of this is built into your serverless price tag.

- Explain several pros and cons that would be deciding factors in plans for expansion.
  Pros:

  - *More services attract more customers*
  - *Spreading business risks into multiple venues*
  - *Increase diversity in ventures*
  - *Room for larger workforce and talent*

  Cons:

  - *Increased cost for more requests and application management*
  - *Increased complexity of model or applications and its management*
  - *More employees required*
  - *More organization to manage and layers of communication to stay organized*

- What roles do elasticity and pay-for-service play in decision making for planned future growth?
  Elasticity and pay-for-service become more important as resource allocation becomes unknown, or variable. For example, business and applications which have high uptimes at different intervals have a greater chance of wasting server resources. This is because server costs remain similar (and management of these services) regardless of use. However, the server costs cannot be down sized because then customer dissatisfaction will increase during peak times (slow servers, errors, etc.). Also, these concepts become key for business that want to expand quickly, or massively. To elaborate, a product that requires expansion quickly can use elasticity and pay-for-service as the price will become more predictable and stable matching the increase. The speed of server integration is much faster through serverless. As for products looking for large expansion, a local server has a time component to set-up. The larger the server the more time for this component.