AWS Certified Solutions Architect: Associate - 14.0 Architecture Best Practices

File Name: amazon-acsaa-14-1-architecture_best_practices

Title: Architecture Best Practices

Subtitle: AWS Certified Solutions Architect: Associate

14.1 Architecture Best Practices

- Design for Failure
  - Assume each component will fail
  - *"Everything fails, all the time"* - AWS CTO Werner Vogels
  - Eliminate single points of failure
    - **Diagram 1**: Simple Web Application
    - **Diagram 2**: Introduce redundancy
      - Multi-AZ Database
      - Multi-AZ Web App
      - ELB
- Implement Elasticity
  - Assume you will need to grow/shrink the environment
  - **Diagram 3**
  - Scaling Vertically
    - Increasing CPU/RAM
    - Usually requires restarting the instance
    - Has a limit
  - Scaling Horizontally
    - Increases the number of instances
    - Not all applications can be distributed
    - Stateless Applications
      - Does not need previous session information
      - Does not require client to reconnect to the same node
    - Stateless Components
      - The entire application is not stateless, but some components are
      - Cookies allow web front ends to become stateless
      - Introduces risk of tampering and connection latency
      - Ideally the cookie should only contain a session ID number
      - Optionally, a DynamoDB table can be used to track sessions
    - Automation
      - Use an automated deploymenmt process like Auto Scaling
      - Bootstrap the instances with custom AMIs
      - Eliminates launch latency
      - Minimizes human error
- Leverage Different Storage Options
  - **Diagram 4 and 5**
  - Choose the right solution
    - Amazon S3
      - Used for bulk data with high performance
      - Also used for backups and disaster recovery
    - Amazon Glacier
      - Used for disaster recovery and data retention
    - Amazon CloudFront
      - Used for high performance content delivery
    - Amazon DynamoDB
      - Used for high performance NoSQL data
    - Amazon EBS
      - Used when reliable storage is required
      - Typically database and mission critical applications
    - Amazon RDS
      - Used for highly available and scalable database solutions
    - Amazon Redshift
      - Used when a fast, fully managed, petabyte scale database is required
    - Amazon ElastiCache
      - Used when a Redis cluster is required for a web application
    - Amazon Elastic File System
      - Used when storage needs to be shared among multiple instances
- Build Security in Every Layer
  - Basic steps

        1. Inventory your data
        2. Prioritize it by value
        3. Apply encryption for data in motion and at rest
    - Defense in depth
        - VPC
        - Security Groups
        - Network ACLs
        - AWS Web Application Firewall
        - IAM
    - Principle of least privilege
    - Perform real-time auditing
        - AWS Config Rules
        - Amazon Inspector
        - AWS Trusted Advisor
        - AWS CloudTrail
- Think Parallel
    - Perform multiple actions simultaneously when possible
    - Hadoop clusters (EMR) allow performing OLTP across multiple nodes
- Loose Coupling Sets You Free
    - Reduce dependancy of one component from another
    - Use only the API to interract with services
    - SQS can also help (**Diagram 6**)
    - Move slow operations off of the critical path
        - For example, transcoding