Assignment 3.8

SCTP CLOUD INFRASTRUCTURE ENGINEERING

Chin Jinn Liong

Ng Hanrong

Lim Wei Yang

Richie Chia

Overcoming Design Complexity in Microservices with AWS

Challenge: Microservices can introduce complex distributed systems, making them difficult to design, maintain, and understand.

Solutions:

- Start Small and Scale: Begin with a few well-defined services and gradually expand based on needs.
- Leverage Serverless Architectures: Utilize AWS Lambda and other serverless services to avoid infrastructure management complexity.
- Utilize Service Discovery: Implement Service Catalog or Route 53 to manage service discovery and routing dynamically.
- Standardize Development Practices: Enforce consistent coding styles, documentation formats, and API design across teams.
- Adopt Domain-Driven Design: Organize services based on business domains to improve modularity and maintainability.

Achieving Data Consistency in Microservices with AWS

Challenge: Ensuring data consistency across multiple microservices can be challenging due to distributed nature.

Solutions:

- Event-Driven Architecture: Use SQS or SNS for asynchronous communication and data updates between services.
- API Gateway with Data Validation: Implement API Gateway with data validation and transformation to ensure data consistency across services.
- DynamoDB with Global Tables: Leverage DynamoDB Global Tables for globally consistent data access across regions.
- Event Sourcing: Use EventBridge for event sourcing, allowing services to react to changes and maintain consistency.
- CQRS (Command Query Responsibility Segregation): Separate read and write operations using different services for better performance and scalability.

Testing and Monitoring Microservices with AWS

Challenge: Effectively testing and monitoring distributed microservices can be complex and time-consuming.

Solutions:

- Utilize CloudWatch: Implement CloudWatch for monitoring service health, performance, and errors.
- Utilize X-Ray: Leverage X-Ray for distributed tracing across services to identify bottlenecks and diagnose issues.
- Utilize CodeBuild and CodePipeline: Automate testing pipelines with CodeBuild and CodePipeline for continuous integration and delivery.
- Utilize CodeCommit with Pull Requests: Use CodeCommit with pull requests for code reviews and collaborative development.
- Utilize Security Hub: Leverage Security Hub for centralized security monitoring and vulnerability assessments across services.

Debugging Issues in Microservices with AWS

Challenge: Debugging issues in distributed systems can be challenging due to complexity and lack of visibility.

Solutions:

- Utilize X-Ray: Use X-Ray for distributed tracing and detailed logs to pinpoint the origin of issues.
- Utilize CloudWatch Logs: Analyze detailed logs from CloudWatch for errors and service interactions.
- Utilize Step Functions: Implement Step Functions for orchestrating complex workflows and easier debugging.
- Implement Canary Deployments: Use canary deployments to gradually roll out changes and monitor for issues in production.
- Utilize Chaos Engineering: Utilize tools like Chaos Monkey to proactively identify potential failures and improve resilience.

Compromised Security in Microservices with AWS

Challenge: Microservices can introduce new attack surfaces, increasing the risk of security breaches.

Solutions:

- Implement IAM Roles for Services: Use IAM roles to grant least-privilege access to services based on their needs.
- Enable CloudTrail and CloudWatch Logs: Monitor all API calls and service interactions for suspicious activity.
- Utilize Amazon GuardDuty: Leverage GuardDuty for threat detection and automated incident response.
- Utilize WAF and Shield: Implement WAF for web application firewall protection and Shield for DDoS mitigation.
- Implement Secrets Manager: Use Secrets Manager to securely store and manage sensitive data.