Litcoder
Powered by Litwork

Code Availability

Case Reference: Amazon Web Services

Litcoder Parameter

Code Flexibility denotes the adaptability and extensibility of the codebase to accommodate future requirements and modifications. It emphasizes writing modular and reusable code. It leverages interfaces and abstraction to decouple components. Finally, it incorporates, and embracing principles such as dependency injection and inversion of control.

Code Availability, an important constituent of code flexibility, refers to the accessibility and availability of source code for software applications or projects. It is a fundamental aspect of open-source software, where the source code is made freely available for anyone to use, modify, and distribute

About AWS

Amazon Web Services (AWS) is a comprehensive, evolving cloud computing platform provided by Amazon. Launched in 2006, AWS has pioneered on-demand cloud infrastructure and platform services. AWS enables organizations to rapidly innovate without upfront infrastructure costs. The breadth and depth of services that AWS provides empowers developers to build and scale applications rapidly.

AWS's major outage issue

In December 2021, AWS suffered a major outage impacting services across multiple regions. Over 100 services for approximately four hours late on June 13 like EC2, Lambda, CloudFront, etc. leading to downstream application failures. Many big companies like FIFA, Fox News, and even the McDonald's app was affected. The main reason for this was that some network devices in the US-EAST-1 area, which are supposed to help manage how much capacity AWS Lambda needs, weren't working correctly. The root causes behind the AWS outage highlights vulnerabilities exacerbated by breakneck growth.

Solutions that can be implemented

To enhance failure resiliency, AWS focused on expanding chaos testing across availability zones. For uncovering weaknesses, it implemented unified logging, metrics, and tracing for holistic observability of inter-service dependencies. Further, resilient design principles like redundancy and compartmentalization were applied and automated remediation mechanisms were deployed. These included self-healing and broader recovery script triggers. Customer communications were enhanced with faster event updates and guidance for allocation of resources after refactoring risky legacy architectures. In order to manage risk effectively amidst growing complexity, they standardized and centralized data schemas across systems. Finally, the improvements included expanding the skills matrix between engineering teams, and combining incremental improvements with long-term modernization initiatives.

AWS rapidly built new services. Its complex and evolving architectures had short term focus and therefore incurred technical debt. AWS reduced technical debt such as shortcuts they took to optimize the expedited delivery over architectural purity, to enhance code quality and availability. This was done through incremental refactoring, chaos testing, observability investments, dedicated architectural efforts, documentation & cross-training.

References: [Click Here]

Contact us: Support@litwork.in