cloud poe part 1A

ST10053561

Varsity College

BCAD

**Table of Contents**

[Traditional On-Premises 2](#_Toc144479741)

[Modern Cloud 2](#_Toc144479742)

[References 10](#_Toc144479743)

Part 1 A:

|  |  |  |  |
| --- | --- | --- | --- |
| Traditional On-Premises | | Modern Cloud | |
| On-Premises Definition | On-Premises example | Cloud definition | Cloud example |
| **Monolithic**  Monolithic architecture represents a conventional software development approach where all elements of an application are complicatedly intwined within a singular executable codebase (Gallardo, 2022). In this design, the whole application, including its UI, business rationale, and data access layer, lives inside a single codebase. Different to programming, monolithic applications brag firmly coupled parts, working with simple organization, troubleshooting, and fast advancement because of their combined nature. Nonetheless, this approach has likely disadvantages, for example, overseeing complexity as the application extends and requiring full application redeployment for even minor changes.  An illustrative occurrence of monolithic architecture's importance inside cloud computing is apparent in application arrangement on cloud stages. Inside this model, applications are packaged and sent as brought together elements on servers. For instance, AWS Versatile Beanstalk streamlines cloud-based sending utilizing a monolithic architecture. | **Monolithic Example:**  Microsoft Word | **Decomposed:**  Decomposition serves as a strategic method within cloud computing, aimed at separating involved applications into discrete, self-reliant components or services, as highlighted by (OneNeck IT Solutions, 2023).  The essential goal spins around upgrading the application's intelligibility, reasonability, and practicality. These modularized parts are mindfully intended for ideal arrangement inside cloud conditions, conceding the upside of independent turn of events and sending cycles.  This approach streamlines the complexities of cloud relocation as well as adjusts advancement endeavours to centre business goals, preparing for the making of deft and expandable cloud-local applications. Embracing decay eventually enables associations to outfit the maximum capacity of cloud computing, working with consistent flexibility and adaptability. | **Decomposed Example:**  Spotify |

|  |  |  |  |
| --- | --- | --- | --- |
| **Designed for predictable scalability Definition:**  Predictable scalability, as outlined by (Bridgwater, 2022), entails an organized and controlled capacity expansion within a system to accommodate growing workloads or higher levels of traffic. The underlying aim is to enable seamless growth without causing abrupt interruptions or demanding extensive alterations to the system's architecture.  This strategy includes precise system plan to alleviate mistakes and to guarantee ideal effectiveness lined up with existing or projected requests. Customarily, this versatility is accomplished through equipment redesigns, yet it's vital to perceive its restrictions because of natural asset imperatives. This approach features the significance of establishing a groundwork that can richly adjust to developing requests while supporting dependability and execution. | **Designed for predictable scalability Example:**  Reddit | **Designed for elastic scale Definition:**  Elastic scale, within the framework of contemporary cloud methodologies, signifies a system or infrastructure's inherent capability to autonomously fine-tune its resources in response to shifts in application traffic dynamics (AVI Networks, 2023).  This basically engages data focuses to flawlessly accommodate floods or decreases in application traffic by quickly presenting or resigning figure and systems administration foundation. In functional terms, like a self-changing component guarantees our system maintains ideal execution and proficiency paying little mind to differing responsibilities, improving the general dexterity and unwavering quality of our tasks. | **Designed for elastic scale Example:**  Uber |

|  |  |  |  |
| --- | --- | --- | --- |
| **Relational Database Definition**:  A relational database is a type of database that organizes and stores data in a structured manner. Imagine it as a collection of tables, like spreadsheets, where each table represents a specific thing – it could be users, products, or orders, for instance. Every row in a table is like a record, containing details about that specific thing, while each column holds a specific piece of information, like a characteristic of that record (Abba, 2022).  This concept is based on an idea put forward by E. F. Codd in 1970, suggesting that data should be organized into these neat tables, or relations. The cool part is how these tables can be connected using unique IDs, almost like secret passcodes, that create links between data in different tables. This way, you can piece together information from various tables and get a complete picture. It's kind of like connecting the dots, where the dots are your data points. | **Relational Database Example**:  Quick Schools  Oracle | **Polyglot persistence Definition**:  Polyglot persistence is an idea that addresses the down to earth insight of utilizing different information storing innovations inside a bound together framework to take special care of a range of information stockpiling prerequisites (Brunskill, 2023). This approach is a sign of approval for the truth that there's no widespread solution for all information stockpiling difficulties. Whether we're managing a multifaceted organization of numerous applications or a lone application lodging more modest module, polyglot persistence expands its utility.  In the domain of cloud computing, polyglot persistence tracks down more than adequate application, giving an application the ability to talk fluidly with a bunch of data sets, utilizing each unique skill. This adaptable technique recognizes that various information bases have changing characteristics, and by taking advantage of these singular qualities, an application can orchestrate its capacity systems more skilfully. | **Polyglot persistence Example:**  Shopify |

|  |  |  |  |
| --- | --- | --- | --- |
| **Synchronized processing.**  in the cloud is a method of correspondence where you, as the client, pause for a minute to respite and stand by in the wake of sending a request to the server. You keep down and do not continue with different activities until the server sends you a reaction. This holding up stage is frequently referring to as "blocking," and it resembles how you'd remain in line at a counter, persistently trusting that your turn will be served. It resembles requiring all the other things to be postponed until you find the solution you're sitting tight for.  This approach is a piece like approaching slowly and carefully and ensuring you get an unmistakable reaction before you push ahead with your next activity (Orelouwa, 2021 ). | **Synchronized Processing Example:**  Multiplayer online games:  Fortnite  Virtual reality (VR) and augmented reality (AR) applications:  Pokémon Go | **Asynchronous processing**  Asynchronous in the cloud is a strategy for correspondence that doesn't need the client to sit tight for a prompt reaction just after sending a request. It presents a more adaptable method of cooperation where the client isn't slowed down while hanging tight for an answer. This approach is frequently alluded to as non-obstructing, primarily because the client's work process isn't restricted while sitting tight for a reaction from the server based on (Orelouwa, 2021 ). This implies that the client can continue with different tasks without being thwarted by the correspondence cycle.  This approach is especially helpful while managing undertakings that could require some investment to finish, for example, information getting or complex calculations. By permitting the client to proceed with its tasks while hanging tight for a reaction, nonconcurrent handling improves proficiency and responsiveness, adding to a smoother client experience. | **Asynchronous processing Example:**  Google Cloud Pub/Sub |

|  |  |  |  |
| --- | --- | --- | --- |
| **Design to avoid failures (MTBF).**  MTBF, or Mean Time Between Failures, serves as a vital device of the frequency of mishaps within a feature or component, aiding teams in predicting the reliability and availability of our IT environments (Wessel, 2021). This metric, often associated with traditional application development, has been focused on extending the gap between failures to enhance performance. In the realm of cloud computing, MTBF finds its place as a measure for evaluating the dependability and consistency of our IT setups. Its significance expands across various technology domains, such as cybersecurity, incident response, and DevOps.  For instance, in cybersecurity, MTBF can signal the decline of a system's value and heightened risks of a significant disruption. Moreover, in DevOps and SRE practices, MTBF emerges as a tool for assessing the quality of workloads or platform services. An appropriate instance of MTBF's relevance within cloud computing emerges through Microsoft Azure's Well-Architected Framework, where it underpins the reliability pillar. Transitioning to cloud computing demands a different mindset, acknowledging the complexity of distributed systems where failures can occur.  Thus, implementing resiliency strategies at all architectural levels becomes critical to inserting failure recovery into the system. In essence, MTBF equips us to predict and enhance reliability and availability in our IT landscapes, while cloud computing introduces resilience strategies to navigate the intricacies of modern distributed systems. | **Design to avoid failures (MTBF) Example**.  Uptime Robot | **Design for failure (MTTR)**  MTTR, which stands for Mean Time to Repair, serves as a vital metric in the digital landscape, representing the average time needed to restore functionality after an incident from failures (Edward, 2022) and (Hertvik, 2017).  Especially in distributed computing, MTTR acquires significance inside the setting of clients getting to cloud-based applications through their own gadgets, taking the estimation of recuperation effectiveness in such powerful conditions. This measurement assumes a basic part in surveying DevOps and IT Operations execution, assessing security processes, estimating framework practicality, and checking the viability of safety arrangements. Distributed computing's development has remarkably moved support and execution obligations onto Framework as Infrastructure as a Service (IaaS) or Platform as a Service (PaaS) providers, with acceptable MTTR rates often outlined in service level agreements (SLAs).  Immediately, MTTR's translation depends on factors like foundation scale, type, group capability, and the complexity of events. Although maintenance times can't be ensured because of the fluctuating multifaceted nature of issues, outsider arrangements, and SLAs set assumptions for MTTR. | **Design for failure**  **(MTTR) Example.**  Opsgenie |

|  |  |  |  |
| --- | --- | --- | --- |
| **Occasional Larger updates**  Larger updates in traditional computing, as described by (Microsoft Corporation, 2017), refer to making significant and comprehensive changes to a system or application to enhance how well it performs, how secure it is, and how functional it remains. These updates involve making big alterations that focus on addressing the core problems while also adding improvements.  The main goal is to make sure that the application continues to stay strong and flexible as the needs and demands around it evolve. This could involve reworking major parts of the software, improving its security measures, and adding new features to make it even better than before. The idea is to ensure that the application remains relevant and effective over time, adapting to changes and staying dependable. | **Occasional Larger**  **Updates Example:**  Microsoft Azure Cloud Services Update | **Frequent Small updates**  In the world of distributed computing, continuous little updates, as called attention to by (Freeman, 2019), rotates around consistently making little, slow changes and enhancements to our cloud-based frameworks or applications. Not at all like those infrequent large updates that include significant makeovers, the focal point of incessant little updates is tied in with making minor improvements, fixing those troublesome bugs, and adding little yet important highlights.  This approach to doing things includes a bit-by-bit approach that assists us with working quicker and even more deftly. It likewise lessens the possibilities of things turning out badly when we send off refreshes, speeds up the most common way of fixing any issues that truly do spring up, and enables us to concoct previously unheard-of things for our clients in a snap. It be similar to continually adjusting our manifestations to remain on top of things and keep our clients blissful. | **Frequent Small updates Example:**  Google Workspace |

|  |  |  |  |
| --- | --- | --- | --- |
| **Manual management**  The manual management in the traditional approach to cloud computing (Semilof, et al., 2021), refers to the active act of administering IT frameworks and applications, enveloping equipment, programming, and systems administration components. This technique requires direct contribution from IT staff, for example, me, who perform tasks like updates establishment, framework arrangement, and issue investigating. Inside the setting of distributed computing, manual administration involves tasks, for example, designing virtual machines, overseeing capacity, and systems administration assets, and sending and scaling applications physically.  This approach could be beneficial in some cases, such as when there is a need for a high degree of control over the IT environment. However, it can also be time-consuming and error-prone, especially as IT environments become more complex. | **Manual Management Example:**  Azure Virtual Machines | **Automated self-management**  Automated self-management in the modern cloud approach (McKinsey & Company, 2021),refers to the utilization of robotization and arrangement advancements to oversee cloud-based frameworks and applications. This approach automates tasks, for example, asset provisioning, scaling, and checking, bringing about smoothed out and effective cloud climate the board. As an IT proficient, I'm keen on more deeply studying how computerized self-administration can be applied to my association's cloud climate. I accept that this innovation can possibly essentially further develop our cloud the board capacities and assist us with accomplishing our business objectives.  I am aware that automated self-management can decrease reliance on manual intervention, enhance system reliability and availability, and achieve heightened agility and scalability | **Automated self-management Example:**  Pinterest |

|  |  |  |  |
| --- | --- | --- | --- |
| **Snowflake servers**  A Snowflake server summarises the traditional technique for overseeing servers, where everyone is particular and physically designed. This stands in sharp differentiation to the contemporary practise of cloud-based foundation, where servers are produced from layouts and dealt with automatically based on (CloverDX Group, 2021). Quiet, the expression "Snowflake" likewise compares to a cloud-based information stage that presents a variety of answers for information capacity, handling, and investigation.  This stage stands apart because of its elevated speed, ease of use, and flexibility in contrast with conventional other options from (Snowflake, Inc. All Rights Reserved., 2023).  Furthermore, the Snowflake platform functions as a fully managed service, operating on cloud infrastructure. It effectively harnesses the capabilities of major platforms like Amazon Web Services, Microsoft Azure, and Google (Reckers, 2023). By embracing Snowflake's imaginative methodology, we can avoid the limits of the Snowflake server model and impel our activities into the cutting-edge domain of distributed computing. This assists server the chiefs as well as heightens our capacity for data limit, dealing with, and assessment, finally uplifting overhauled efficiency and flexibility. | **Snowflake servers Example:**  Trade Me | **Immutable infrastructure**  Immutable infrastructure is a contemporary way to deal with overseeing programming arrangements and administrations on IT assets, wherein servers or virtual machines stay unaltered following sending. As opposed to adjusting existing parts, this approach includes supplanting them, prompting the redeployment of an application or administration at whatever point any change happens (Bigelow, 2022).  This technique offers benefits like reduced IT difficulty and failures, upgraded security, and streamlined investigating contrasted with impermanent framework. Server fixing and design changes are killed, as each update starts the making of a new, completely tried, and current example. In situations where the new case misses the mark concerning assumptions, returning to the past known-great occurrence is direct. | **Immutable infrastructure**  **Example:**  Netflix |

# References

Abba, I. V., 2022. *What is a relational database? RDBMS definition.* [Online]   
Available at: https://www.freecodecamp.org/news/what-is-a-relational-database-rdbms-definition/  
[Accessed 8 August 2023].

AVI Networks, 2023. *What is elastic scale? Definition and related FAQs.* [Online]   
Available at: https://avinetworks.com/glossary/elastic-scale/  
[Accessed 8 August 2023].

Bigelow, S. j., 2022. *What is immutable infrastructure?: Definition from TechTarget.* [Online]   
Available at: https://www.techtarget.com/searchitoperations/definition/immutable-infrastructure  
[Accessed 17 August 2023].

Bridgwater, A., 2022. *What does it scalability actually mean?.* [Online]   
Available at: https://www.forbes.com/sites/adrianbridgwater/2022/07/04/what-does-it-scalability-actually-mean/?sh=5c38dc293429  
[Accessed 8 August 2023].

Brunskill, V. L., 2023. *What is polyglot persistence? definition from whatis.com..* [Online]   
Available at: https://www.techtarget.com/searchapparchitecture/definition/polyglot-persistence  
[Accessed 8 August 2023].

CloverDX Group, 2021. *What is Snowflake?.* [Online]   
Available at: https://www.cloverdx.com/blog/what-is-snowflake  
[Accessed 17 August 2023].

Edward, E., 2022. *Global AI and Data Science.* [Online]   
Available at: https://community.ibm.com/community/user/ai-datascience/blogs/emiley-edward/2022/04/05/what-is-mttr-in-cloud-computing  
[Accessed 19 August 2023].

Freeman, E., 2019. *Amazon.* [Online]   
Available at: https://aws.amazon.com/devops/what-is-devops/  
[Accessed 15 August 2023].

Gallardo, E. G., 2022. *What is a monolithic application?.* [Online]   
Available at: https://builtin.com/software-engineering-perspectives/monolithic-application  
[Accessed 17 August 2023].

Hertvik, J., 2017. *MTTR explained: Repair vs recovery in a digitized environment.* [Online]   
Available at: https://www.bmc.com/blogs/mttr-explained-repair-vs-recovery-in-a-digitized-environment/  
[Accessed 19 August 2023].

McKinsey & Company, 2021. *Building a cloud-ready operating model for agility and resiliency.* [Online]   
Available at: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/building-a-cloud-ready-operating-model-for-agility-and-resiliency  
[Accessed 16 August 2023].

Microsoft Corporation, 2017. *Cloud Application Architecture Style.* Redmond, Washigton: Microsoft Press.

OneNeck IT Solutions, 2023. *How decomposition and microservices make cloud migration easier.* [Online]   
Available at: https://www.oneneck.com/blog/decomposition-and-microservices-make-cloud-migration-easier/  
[Accessed 5 August 2023].

Orelouwa, F. A., 2021 . *Introduction to Asynchronous Processing.* [Online]   
Available at: https://hookdeck.com/blog/post/introduction-asynchronous-processing#what-is-synchronous-processing  
[Accessed 9 August 2023].

Reckers, E., 2023. *What is the snowflake data platform?  .* [Online]   
Available at: https://www.snaplogic.com/blog/snowflake-data-platform  
[Accessed 17 August 2023].

Semilof, M., Bigelow, S. J. & Casey, K., 2021. *What is cloud management? everything you need to know.* [Online]   
Available at: https://www.techtarget.com/searchcloudcomputing/definition/cloud-management  
[Accessed 16 August 2023].

Snowflake, Inc. All Rights Reserved., 2023. *Key Concepts & Architecture | Snowflake Documentation.* [Online]   
Available at: https://docs.snowflake.com/en/user-guide/intro-key-concepts  
[Accessed 17 August 2023].

Wessel, k., 2021. *MTBF is an integral part of business operations - here's why.* [Online]   
Available at: https://www.xmatters.com/blog/mtbf-is-an-integral-part-of-business-operations-heres-why/  
[Accessed 19 August 2023].