CSCI 5902 Advanced Cloud Architecting

1. Instance Selection and Auto Scaling: Evaluate the requirements of the application and determine the type and size of EC2 instances that would be most appropriate for your needs. Consider factors like the application's CPU, memory, storage, and network performance requirements.

Instance Selection:

Based on the given scenario, a mix of network optimisation and compute-optimised instances (C-Series) would likely be the most suitable for achieving high overall performance.

The C5 series would enable low-latency connections across instances, which are crucial for providing smooth and engaging user experiences, particularly when users engage and go on adventures in multiplayer settings.
Improved networking capabilities in these instances allow for high throughput and low latency communication between instances spread across different geographic locations. Furthermore, the C5 series may provide network capacity of up to 25 Gbps. It is also based on the AWS Nitro system, a vast array of building blocks that transfers many of the conventional virtualization tasks to specialised hardware and software in order to minimise virtualization overhead and provide high performance, high availability, and high security.

In order to dynamically change the number of instances based on demand, implement auto scaling for both the database and web application tiers. Set up policies for scaling according to metrics like CPU, RAM, or application-specific metrics.
Use AWS Auto Scaling groups to automatically alter capacity based on specified scaling parameters and manage the appropriate number of instances. This aids in managing the fluctuating volume of users, particularly during peak hours.
Elastic Load Balancers (ELB) can be used to split traffic among several instances in the multiplayer mode, guaranteeing high availability and effectively allocating load.
It is possible to set threshold-based scaling actions to be triggered by Cloudwatch alarms. Establish cooldown times to avoid hurried and pointless scaling operations.
To guarantee appropriate load distribution, integrate the Auto Scaling group with a load balancer to split traffic equally among instances.

2. Multi-AZ and Multi-Region Deployment: Implement a Multi-AZ and Multi-Region deployment to ensure high availability and fault tolerance.

In the realm of Multi-AZ deployment, the Holodeck application's web hosting is fortified by deploying EC2 instances across multiple Availability Zones within a single region. This strategic distribution ensures that any issues encountered by one Availability Zone won't disrupt the application's flow, as traffic seamlessly redirects to instances in another zone. This not only enhances availability but also safeguards against localized failures, providing users with a consistent and reliable experience.

For database hosting, the utilization of Amazon RDS in a Multi-AZ configuration elevates the system's robustness. Configuring RDS to operate in multiple Availability Zones automatically creates a standby instance in a different zone, allowing for high availability and automatic failover. This redundancy is crucial for safeguarding the integrity of data and ensuring uninterrupted service in the face of potential database issues.

To efficiently manage incoming traffic, Elastic Load Balancers (ELB) are deployed, intelligently distributing requests across multiple instances situated in different Availability Zones. ELB's ability to detect unhealthy instances and reroute traffic to healthy counterparts further solidifies the system's resilience, contributing to a seamless user experience even in the event of instance failures.

Auto Scaling groups are configured to transcend multiple Availability Zones, ensuring that the system's capacity is dynamically maintained across different zones. This dynamic scaling not only optimizes resource utilization based on demand but also enhances fault tolerance by distributing the load effectively.

In the expansive strategy of Multi-Region Deployment, a dedicated Disaster Recovery (DR) Region is established in a distinct geographical location. This secondary region mirrors critical resources, encompassing EC2 instances, databases, and storage, thereby creating a failover site ready to take over in the event of a catastrophic failure in the primary region.

Data replication between the primary and disaster recovery regions is achieved through the implementation of AWS services like the Database Migration Service (DMS) or Amazon S3 Cross-Region Replication. This replication ensures consistent and available data in the face of a region-wide failure, maintaining the integrity of the Holodeck application's vast data landscape.

Global Traffic Management is orchestrated through Amazon Route 53, employing the multi-region failover routing policy. This strategic use of DNS allows queries to be directed to healthy resources in the designated primary region. In the event of a region failure, Route 53 intelligently redirects traffic to the disaster recovery region, providing a seamless and uninterrupted experience for users.

The architecture further fortifies itself with Cross-Region Load Balancing, employing tools such as Global Accelerator or Load Balancers that distribute traffic across instances in different regions. This not only enhances availability but also ensures responsiveness for users, irrespective of their geographical location.

To maintain a vigilant watch over the system, AWS CloudWatch and AWS CloudTrail are implemented for monitoring and logging across both regions. Automated alerts and responses are set up to swiftly address issues, minimizing downtime and ensuring a proactive approach to system health.

Regular disaster recovery drills are conducted to validate the efficacy of failover mechanisms. These simulations of region failures serve as a litmus test for the system's resilience and recovery capabilities, allowing for continuous improvement and optimization of the disaster recovery strategy.

Lastly, in navigating the cosmos of cost considerations, prudent strategies are employed. This involves being mindful of costs associated with resources in multiple regions, employing cost-effective measures such as reserved instances, and leveraging tools like AWS Cost Explorer to diligently monitor and manage expenses. This ensures that Spacetech Galac5c's journey through the virtual cosmos remains not only technically robust but also economically sustainable.

3. EBS & EFS: Plan a strategy for data storage using Amazon EBS and/or EFS.

Amazon Elastic File System (EFS) is the best choice to run Spacetech Galac5c's Holodeck application's real-time physics engine. This is the reason why:

Suitability for Changing Tasks:
EFS can withstand heavy workloads by dynamically scaling to provide the required throughput in demanding settings. With the ability to handle sudden increases in workload with ease, it guarantees peak performance even in times of rapid file system expansion and can handle workloads up to 500,000 IOPS, or 10 GB per second.

Adaptive Storage Administration:
Because EFS automatically scales storage up or down in response to demand, it offers energetic flexibility. Storage provisioning is no longer an issue because files can be added or removed without interfering with running programmes. Without human input, the file system adjusts to changing requirements.

Versatile Accessibility:
EC2 instances can easily access EFS file systems not only within the same region but also across different AWS regions through VPC peering. This ensures seamless collaboration and data access, facilitating distributed architectures and enhancing flexibility.

Support for Serverless Architectures:
In contrast to EBS, EFS integrates seamlessly with AWS Lambda serverless functions. This enables efficient data sharing between serverless functions without the need for managing storage provisioning. Lambda functions can seamlessly read and write large files to EFS, enhancing the agility of serverless workflows.

Fully Managed Service:
EFS is a comprehensive managed service, eliminating the need for manual tasks such as patching, deployment, or maintenance. This aligns with the requirement of avoiding downtime caused by regular system updates. The burden of file system management is lifted, allowing the team to focus on innovation rather than infrastructure maintenance.

Cost-Effective Usage:
One of the key advantages of EFS is its cost model. The firm only pays for the storage used, without the need for advance provisioning, up-front fees, or commitments. This cost-effective approach aligns with Spacetech Galac5c's mission to balance high computational demands with economic sustainability.

In conclusion, leveraging EFS for the real-time physics engine not only ensures top-notch performance and scalability but also aligns with a serverless and cost-effective paradigm, allowing Spacetech Galac5c to continue its journey through the virtual cosmos seamlessly.

4. Spot and Reserved Instances: Incorporate spot instances and reserved instances in your architecture

The Holodeck application by Spacetech Galac5c uses both Reserved Instances and Spot Instances in its architecture, which offers a calculated method of striking a balance between cost and performance. Spot instances are a cost-effective solution that may be used for batch processing, background operations, and non-critical workloads. The application can adapt its resource usage to fluctuating user traffic by setting up Auto Scaling groups to take advantage of Spot Instances when demand is low. By responding to shifting capacity requirements, a varied Spot Fleet spanning several instance types and Availability Zones assures resilience and reduces the chance of disruptions.

Reserved Instances, on the other hand, offer a discounted pricing model suitable for applications with steady and predictable workloads. Allocating Reserved Instances for baseline capacity ensures cost predictability and stability for essential components of the Holodeck application, such as databases. The choice between Standard and Convertible Reserved Instances provides flexibility in adapting to evolving requirements, allowing the architecture to scale with changing needs.

To maximize the benefits of both Spot and Reserved Instances, Auto Scaling policies are implemented to dynamically adjust the mix of instance types based on demand. This strategy ensures that the application optimally utilizes the cost savings from Spot Instances while maintaining reliability and consistency with Reserved Instances. Monitoring solutions like AWS CloudWatch are integrated to track Spot Instance interruptions, enabling continuous optimization of bidding strategies to align with the application's sensitivity to interruptions.

The architecture also emphasizes a holistic approach, combining Spot Instances for burstable workloads and Reserved Instances for baseline capacity. This allows the system to seamlessly adapt to variable computational demands, ensuring a cost-optimized yet resilient environment. The inclusion of On-Demand Instances within Auto Scaling groups enhances the overall system resilience, acting as a fallback in case of Spot Instance interruptions.

In conclusion, the integration of Spot Instances and Reserved Instances in the architecture of Spacetech Galac5c's Holodeck application is a strategic move that aligns with the mission of exploring new frontiers while maintaining economic sustainability. The combination of flexibility, cost predictability, and resilience positions the architecture to handle the diverse and dynamic demands of virtual space travel experiences efficiently.

5. AMI: Explain how you would use Amazon Machine Images (AMIs) to quickly deploy and replicate your application.

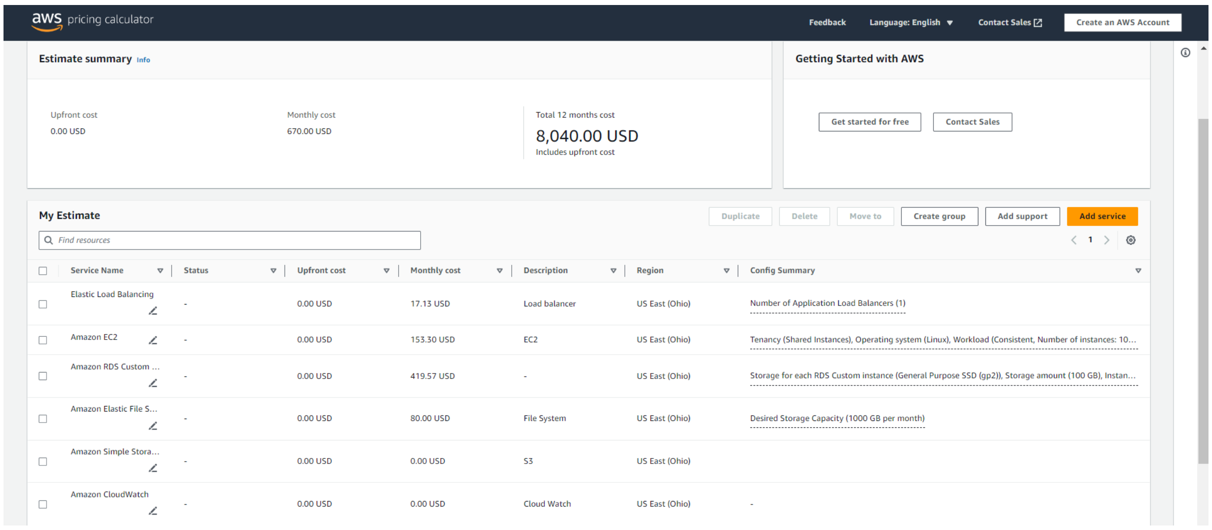
The Amazon Machine Image (AMI) life cycle is a critical element in expediting the deployment and replication of Spacetech Galac5c's Holodeck application. AMIs serve as snapshots of instances, facilitating rapid provisioning and replication across various regions or availability zones. To begin, a base AMI is crafted with the necessary operating system, software dependencies, and configurations. This base is then customized by adding application code, configuration files, and specific dependencies, tailoring it to the application's requirements. The AMI is subsequently replicated across multiple regions or availability zones using AWS tools, ensuring high availability and fault tolerance. Instances are launched directly from these replicated AMIs during application deployment, leveraging features like AWS Auto Scaling for dynamic management based on demand.

Maintaining agility and evolution, the AMI undergoes updates to include the latest application code, configurations, and dependencies, while version control ensures consistency. Automation becomes a key player in streamlining the deployment process, allowing for the definition of infrastructure, including AMIs, networking, security settings, and other resources. This automated approach expedites deployment while ensuring reliability and consistency, ultimately saving time and effort.

In essence, harnessing the potential of AMIs is instrumental in not only expediting deployment and replication but also in guaranteeing reliability and scalability for the Holodeck application. This strategic utilization of AMIs aligns with the dynamic and evolving nature of the application, offering efficiency, consistency, and resilience in the ever-expanding virtual cosmos.

6. Cost Management: Provide a rough estimate of the costs of running this infrastructure and discuss the strategies you would use to manage these costs.

Using the AWS pricing calculator, the anticipated costs were computed while accounting for the particular region and the specifications listed for the architecture diagram that we created. The estimate is based on assumptions and could change dependent on things like instance kinds, storage requirements.



Estimate summary Info

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	670.00 USD	8,040.00 USD Includes upfront cost

My Estimate

Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary
Elastic Load Balancing	-	0.00 USD	17.13 USD	Load balancer	US East (Ohio)	Number of Application Load Balancers (1)
Amazon EC2	-	0.00 USD	153.30 USD	EC2	US East (Ohio)	Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 10...
Amazon RDS Custom ...	-	0.00 USD	419.57 USD	-	US East (Ohio)	Storage for each RDS Custom instance (General Purpose SSD (gp2)), Storage amount (100 GB), Instan...
Amazon Elastic File S...	-	0.00 USD	80.00 USD	File System	US East (Ohio)	Desired Storage Capacity (1000 GB per month)
Amazon Simple Stora...	-	0.00 USD	0.00 USD	S3	US East (Ohio)	
Amazon CloudWatch	-	0.00 USD	0.00 USD	Cloud Watch	US East (Ohio)	-

Cost Estimation Components:

Amazon EC2 Instances:
The costs for EC2 instances will depend on the type, size, and number of instances running. Considering a mix of On-Demand, Reserved, and Spot Instances for different workloads, costs may range from a few hundred to several thousand dollars per month.

Amazon EBS Volumes:
EBS volume costs will depend on the type and size of volumes used for storage. This cost can vary from a few cents to several dollars per GB per month.

Amazon EFS Storage:
EFS costs are based on the amount of data stored and accessed. Costs can range from a few cents to dollars per GB per month.

Data Transfer:
Data transfer costs will depend on the amount of data transferred between regions, availability zones, and over the internet. Costs can vary from a few cents to dollars per GB.

Additional Services:
Costs for additional services such as Amazon RDS for databases, AWS Lambda for serverless functions, and other services will contribute to the overall expenses.

Cost Management Strategies:

1. Reserved Instances and Savings Plans:
Utilize Reserved Instances for baseline capacity where usage is consistent. This provides significant cost savings compared to On-Demand Instances.
Consider AWS Savings Plans, which offer flexibility across various services, allowing for significant cost reductions.

2. Spot Instances for Non-Critical Workloads:
Leverage Spot Instances for non-critical workloads, background tasks, and batch processing. Spot Instances can provide substantial cost savings, especially during periods of low demand.

3. Auto Scaling and Efficient Resource Utilization:
Implement Auto Scaling to dynamically adjust the number of instances based on demand. This ensures efficient resource utilization and cost savings during off-peak hours.

4. Data Storage Optimization:
Regularly review and optimize the storage needs for Amazon EBS and EFS. Use lifecycle policies to move infrequently accessed data to lower-cost storage classes.

5. Monitor and Analyze Costs:
Utilize AWS Cost Explorer and AWS Budgets to monitor and analyze costs. Set up alerts for cost thresholds to receive notifications when spending exceeds predefined limits.

6. Scheduled Scaling and Resources:
Implement scheduled scaling to adjust resources based on expected usage patterns. Scale down during off-peak hours and scale up to meet peak demand.

7. Efficient Data Transfer:
Optimize data transfer costs by using Content Delivery Networks (CDN) like Amazon CloudFront for content delivery. Minimize cross-region data transfers where possible.

8. Regularly Review Architecture:
Periodically review the architecture to ensure it aligns with the evolving needs of the application. Consider adjustments based on the latest AWS pricing models and services.

7. EC2 Placement Groups: Design an architecture where the application has a need for low-latency, high throughput communication between instances.

EC2 Placement Groups are essential to designing an architecture that satisfies the need for high-throughput, low-latency communication between instances in Spacetech Galac5c's Holodeck application. By guaranteeing that instances are physically situated adjacent to one another, EC2 Placement Groups give administrators control over where they are placed inside the underlying infrastructure, reducing network latency and boosting communication speed. This is a recommended architecture:

1. EC2 Placement Group Configuration:
Create an EC2 Placement Group specifically tailored for the application's low-latency, high-throughput needs. Choose the appropriate type of Placement Group based on the requirements. For this scenario, a Cluster Placement Group, which places instances in close proximity to each other, is suitable.

2. Application Component Instances:
Identify the application components that require low-latency, high-throughput communication. This could include instances running real-time physics simulations, multiplayer game servers, or any other components that heavily rely on fast and efficient communication.

3. Use of Compute-Optimized Instances:
Select compute-optimized EC2 instance types that offer high CPU performance, ideal for applications with compute-intensive workloads and high communication requirements. Instances like the C6g or C5n families might be suitable
.
4. Deployment Across Availability Zones:
Distribute instances across multiple Availability Zones within the same region to enhance fault tolerance. This helps ensure that even if there are issues in one Availability Zone, the application can continue to function with minimal disruption.

5. Load Balancing and Auto Scaling:
Implement Elastic Load Balancers (ELB) to evenly distribute incoming traffic across instances within the Placement Group. This not only enhances availability but also facilitates load balancing for a scalable and fault-tolerant architecture.
Utilize Auto Scaling groups to dynamically adjust the number of instances based on demand, ensuring that the application can scale seamlessly to handle variable workloads.

6. Secure Network Configuration:
Implement Virtual Private Cloud (VPC) to isolate the application's network environment. Configure security groups and network ACLs to control inbound and outbound traffic between instances while maintaining the required low-latency communication.

7. Monitoring and Optimization:
Implement AWS CloudWatch for monitoring the performance of instances, network utilization, and latency. Set up alarms to receive notifications in case latency thresholds are breached.

8. Instance Store: Design a scenario where temporary, high-IOPS storage is required and an instance store would be used.

Scenario: Real-Time Video Processing Cluster

Use Case: Spacetech Galac5c is developing a real-time video processing application for the Holodeck experience. Users can upload and stream high-definition 3D videos that require on-the-fly processing for rendering and virtual reality effects.

Requirements:

High Performance: The application demands real-time video processing with low latency to provide users with an immersive experience.
Temporary Storage: Video processing involves heavy read/write operations during the processing phase, but the processed data does not need to be stored persistently after the session ends.

Design:

EC2 Instance Selection:
Choose EC2 instances optimized for compute-intensive workloads, such as the "c" (compute-optimized) or "i" (storage-optimized) instance families. Instances like "i3" or "c5" could be suitable for this scenario.

Instance Store Configuration:
Configure instances to use the local instance store volumes for temporary storage during video processing. Instance store provides high IOPS and low-latency access to data, making it ideal for the intensive read/write operations involved in real-time video processing.

Clustered Architecture:
Deploy multiple EC2 instances in a clustered architecture to create a processing cluster. Each instance in the cluster contributes to the overall processing power.

Load Balancing:
Implement an Elastic Load Balancer (ELB) to distribute incoming video processing requests across instances in the cluster. This enhances performance and ensures efficient utilization of resources.

Parallel Processing:
Design the application to leverage parallel processing capabilities. Split the video processing workload into smaller tasks and distribute them across instances for concurrent execution.

Auto Scaling:
Utilize Auto Scaling groups to dynamically adjust the number of instances based on the workload. This ensures that the cluster can efficiently scale up or down to handle varying numbers of video processing requests.

Temporary Storage Cleanup:
Implement a cleanup mechanism to remove temporary storage data once the video processing is completed. Since instance store data is ephemeral and doesn't persist after instance termination, this step ensures efficient resource management.

9. Dedicated Hosts/Instances: Plan for scenarios where the application has to comply with strict licensing terms (BYOL) or meet dedicated hardware requirements.

10. EC2 Metadata and User Data: Describe how you would use EC2 metadata and user data to handle configuration tasks and pass information to instances at launch time.

11. Optimization and Performance: Describe how to use tools like AWS Compute Optimized and Trusted Advisor for identifying optimal EC2 instance types and for maintaining cost efficiency