Q1.What is the goal of building A cloud native Application?

A)
The goal of building a cloud - native application is to leverage the capabilities and advantages offered by cloud computing environments to create scalable ,resilient,and flexible software solutions.Cloud - native applications are designed and optimized to run in cloud environments,taking full advantage of cloud services and architectures.
Sometimes the ISV might just need to modernize existing systems,workflows,and processes to enhance agility and scalability.

Key goals of building a cloud-native application include:

1.Scalability:
Cloud-native applications should be able to scale easily to handle varying workloads and demand.This scalability is often achieved by leveraging cloud resources such as auto-scaling ,load balancing ,and container orchestration.

2.Resilience and Fault Tolerance:
Cloud-native applications are built with a focus on resilience and fault tolerance.They are designed to handle failures gracefully,recover from faults,and minimize downtime.This is often achieved through distributed architectures,redundancy and failover mechanisms.

3.Flexibility and Agility:
Cloud native applications embrace agile development practices and Devops methodologies.They enable rapid development,continuous integration , and continuous deployment (CI/CD).This allows for quicker release cycles,faster time-to-market,and the ability to respond promptly to changing business requirements.

4.Microservices Architecture:
Cloud-native applications are often built using a microservices architecture,where the application is composed of small ,independently deployable services.This modular approach facilitates development,deployment,and maintenance , as well as enables scalability and fault isolation.

5.Containerization:
Containers,such as Docker,play a significant role in cloud-native applications.Containerization provides consistency across different environments,simplifies deployment,and supports the efficient scaling of services.

6.Elasticity:
Cloud-native applications can dynamically scale up or down based on demand.This elasticity allows organizations to optimize resource utilization ,reduce costs,and provide a better user experience during peak usage periods.

7.Infrastructure as Code(IaC):
Cloud-native development often involves treating infrastructure as code,automating the provisioning and management of resources using tools like Terraform or AWS CloudFormation.This enables reproducibility,versioning and efficient infrastructure management.

8.Utilization of Cloud services:
Cloud-native applications leverage various cloud services ,such as storage , databases,messaging queues , and machine learning services,to offload responsibilities and take advantage of specialized offerings provided by cloud providers.

9.Continuous monitoring and Feedback:
Cloud-native applications incorporate continuous monitoring and feedback loops to ensure performance ,detect issues in real time , and provide insights for further optimization.

In summary,the goal of building a cloud-native application is to harness the benefits of cloud computing ,including scalability,resilience,flexibility and efficiency,to deliver applications that meet the demands of modern , dynamic business environments.

Q2. What Are The Key Criteria Influencing The Choice Of A Cloud Platform?

A)
The choice of cloud platform is a crucial decision for organizations , and it is influenced by various criteria .Different cloud providers offer a range of services ,pricing models, and features ,So organizations need to consider their specific needs and priorities.

Here are key criteria influencing the choice of cloud platform:

1.Service offerings:
Assess the breadth and depth of services offered by the cloud provider.Consider Infrastructure as a Service(IaaS),platform as a Service(PaaS),and software as a Service(SaaS) offerings,as well as specialized services like machine learning,analytics and databases.

2.Scalability and performance:
Evaluate the scalability and performance capabilities of the cloud platform.
Consider the ability to scale resources horizontally and vertically to meet the demands of your application.

3.Geographic presence:
Consider the global footprint of the cloud provider.Choose a provider with data centers in regions that align with your business requirements for data residency ,low -latency access ,and disaster recovery.

4.Compliance and security:
Assess the cloud provider’s compliance certifications ,security features ,and data protection capabilities.This is crucial ,especially for industries with regulatory compliance requirements. E.g healthcare , finances

5.Cost and pricing Models:
Understand the pricing structure and models of the cloud provider.consider factors such as on-demand pricing,reserved instances,spot instances , and any hidden costs.Choose a provider whose pricing aligns with your usage patterns and budget.

6.Ease of Integration:
Evaluate the ease of integrating the cloud services with your existing infrastructure,applications and tools.Consider compatibility with popular development frameworks and programming languages.

7.Reliability and Uptime SLA:
Review the cloud provider’s track record for reliability and the service level agreement (SLA) for uptime.Look for guarantees on availability and performance.

8.Management and Monitoring tools:
Consider the availability and usability of management and Monitoring tools provided by the cloud platform.These tools help in resource management,monitoring and troubleshooting.

9.Community and Ecosystem:
Assess the size and vibrancy of the community and ecosystem around the cloud platform.A strong ecosystem can provide access to third-party tools,libraries and community for knowledge sharing.

10.Support and Service level Agreements(SLAs):
Evaluate the level of customer support offered by the cloud provider.Review SLAs for support response times and issue resolution.understand the support plans available.

11.Innovation and roadmap:
Consider the provider’s commitment to innovation and staying at the forefront of technology.Evaluate their roadmap for introducing new features and services that align with your future needs.

12.Exit Strategy:
Consider the ease of migrating away from the cloud platform if needed.Evaluate the portability of your applications and data and ensure you have exit strategy case you decide to switch providers.

By carefully considering these criteria , organizations can make informed decisions when choosing a cloud platform that best fits their business requirements,technical needs and long term strategic goals.

In summary,To develop a cloud native application , ISVs will have to evaluate the choice of platform based on the following criteria:

Cloud platform:ISVs need to check if the available cloud platforms can support their customer's business needs.For instance , does a platform offer capabilities such as business intelligence and data analytics that may be necessary to draw the most benefit out of an upgraded application?.Is it flexible enough to be re configured based on the needs and challenges of the enterprise customer base?

Development Model:The ISV should focus on picking the development and deployment model that can align with the business scenarios and requirements of their target enterprises.They must ensure that the model they choose supports all the current and future strategies of the enterprises they cater to.

Containerization:The chosen approach should support containerization whether it is the Docker container with Kubernetes orchestration or Docker Swarm clusters with bare metal capability.This helps the ISV build more flexibility and speed.
Third - party Integration:ISVs must ensure that the chosen cloud platform interoperates and integrates seamlessly with the third-party service providers likely to be a part of the enterprise tech ecosystem.

Culture Shift:The move to cloud-native services could change the existing business workflows and processes within the ISV.For instance , it might require a drastic cultural shift by making DevOps mainstream.ISVs need to assess if they are ready for such radical shifts.

Q3. How To Move From Monolithic App Development To Cloud-Native App Development?

A)
Moving from monolithic app development to cloud-native development involves a shift in architecture development practices ,and deployment strategies.

Before ISVs decide to move their environment and applications to the cloud,they need to assess products for cloud readiness , formulate cloud migration strategy and choose the right cloud migration path for product transformation.ISVs could move from monolithic applications to a cloud native environment in the following ways:

Rehosting:In rehosting,also known as lift and shift,ISVs can shift the existing monolithic platform to the cloud environment with minimal or no code changes . This can be implemented for applications that do not involve re-writing or had been previously architected with the cloud in mind.This is the easiest and the most cost effective way to migrate the applications.However its not always possible.

Re-factoring: To achieve scalability of your applications,ISVs need minimal transformation to connect to cloud services.Re-factoring can be done partially or fully depending on the objective.This includes modifying the application code to ensure that it works well in the cloud environment.

Re-architecting: For any ISV , this can be a resource - intensive and time-consuming process since it involves rearchitecting and rewriting the applications from scratch in modern language and framework.ISVs must consider the time cost goal triage before investing in this process.One best practices can be incorporating cloud native approaches auto scaling,elastic computing , and orchestration between services to improve the sustainability and scalability of your applications.
ISVs must evaluate their existing technical capabilities and cloud readiness and how these would need to evolve to support their future growth.The goal should be to maximize the cloud benefits and minimize for the enterprise customer.

Key steps to transition from a monolithic application to a cloud native architecture:

1.Understand cloud Native Principles:

Understand cloud native principles such as micro services architecture,containerization ,automation and scalability.Understand the benefits of cloud native development ,including improved agility, resilience and scalability.

2.Assess Your Monolith:

Conduct a thorough assessment of your existing monolithic application.Identify components that can be decoupled and modularized.Evaluate dependencies and data flows within the monolith.

3.Breakdown the Monolith:

Identify and decouple functionalities into independent services.This can be achieved through a process known as “strangling the monolith”,where we gradually extract and migrate components to microservices.

4.Containerize the Application:

Containerization enables us to package the application and its dependencies into lightweight,portable containers.Tools like Docker are commonly used for containerization.Containerization facilitates consistency across different environments and simplifies deployment.

5.Adopt Microservices Architecture:

Break down your application into small,independently deployable microservices.Each microservice should handle a specific business capability.This modular approach allows for easier development,deployment and scalability.

6.Implement API Gateways:
Use an API gateway to manage and expose your microservices through a unified API.This helps abstract the complexity of multiple services and provides a single entry point for external clients.

7.Choose Cloud-Native Databases:
Consider adopting cloud-native databases that are designed to scale horizontally and take advantage of cloud storage options.noSQL databases and cloud managed databases can be suitable choices.

8.Implement DevOps Practices:
Embrace DevOps practices to automate and streamline development,testing,and deployment processes.Implement continuous integration and continuous deployment(CI/CD) pipelines to enable frequent releases.

9.Container Orchestration:
Use container orchestration tools such as Kubernetes to automate the deployment , scaling,and management of containerized applications.Kubernetes simplifies the management of microservices at scale.

10.Implement Infrastructure as Code(IaC):
Manage your infrastructure programmatically using Infrastructure as Code(IaC) tools like Terraform or AWS CloudFormation.This allows you to version and automate the provisioning of infrastructure resources.

11.Monitor and Analyze:
Implement monitoring and logging to gain insights into the performance and behavior of your cloud native application.Leverage tools for log aggregation,metrics, and tracing to facilitate debugging and performance optimization.

12.Embrace Cloud Services:
Take advantage of cloud services for various capabilities such as storage,messaging,authentication , and machine learning.Cloud services can replace or augment components of your application,allowing you to focus on business logic.

13.Train your Team:
Provide training and support for your development and operations teams to familiarize them with cloud native principles,tools, and practices.Encourage a culture of learning and experimentation.

<u>**14.Iterate and Optimize:**</u>

The transition to cloud native is an iterative process.Continuously gather feedback,measure performance,and optimize your application architecture and deployment strategies based on lessons learned.

By following these steps ,you can successfully transition from a monolithic application to a cloud native architecture ,unlocking the benefits of scalability, resilience and agility in the cloud environment.