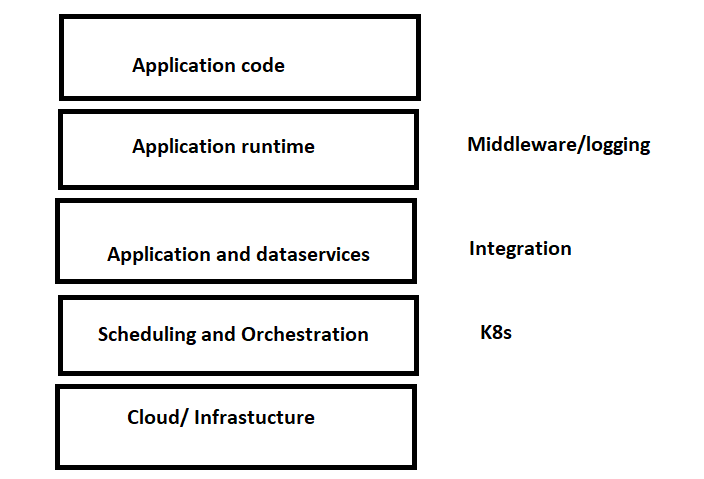
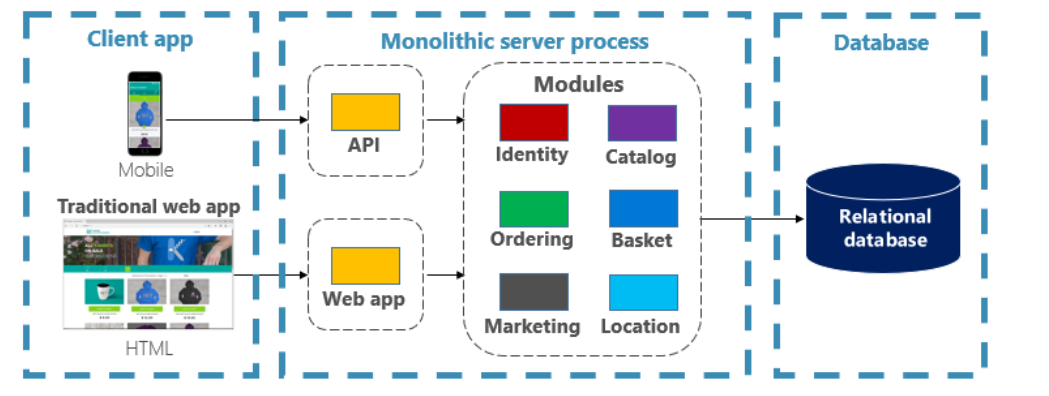
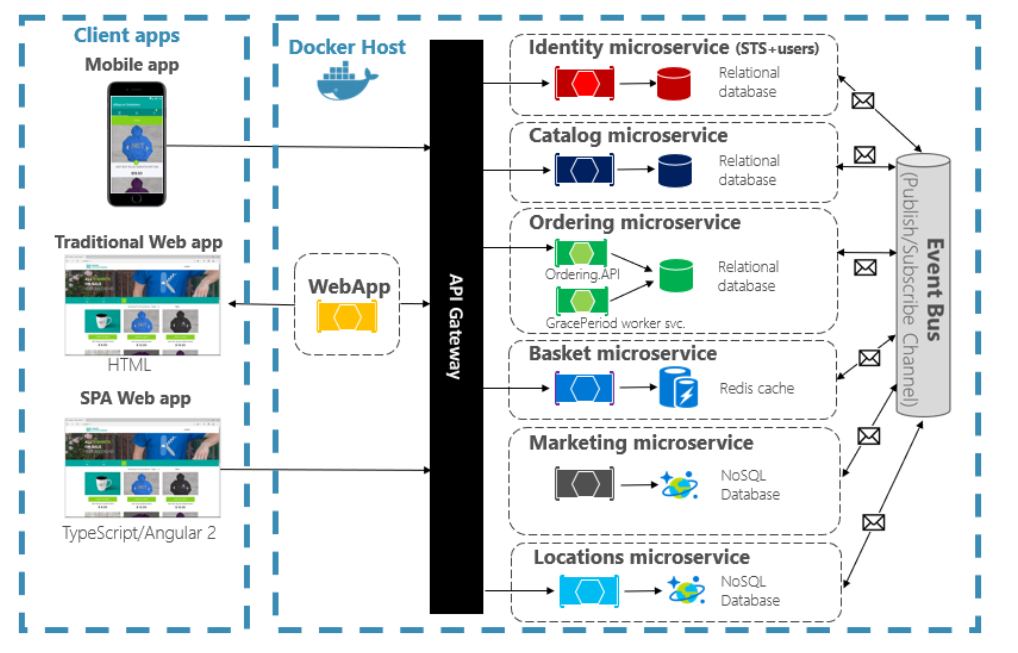
Cloud native applications are composed of reusable components known as microservices and are designed to integrate into any cloud environment. This flexible and scalable approach is ideal for enterprises who are tasks with improving existing applications, building new applications, and enhancing user experience.

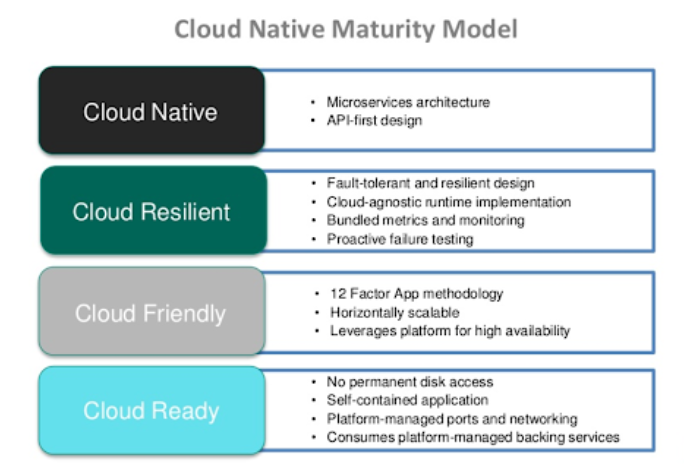


Monolithic application:



CNF application:

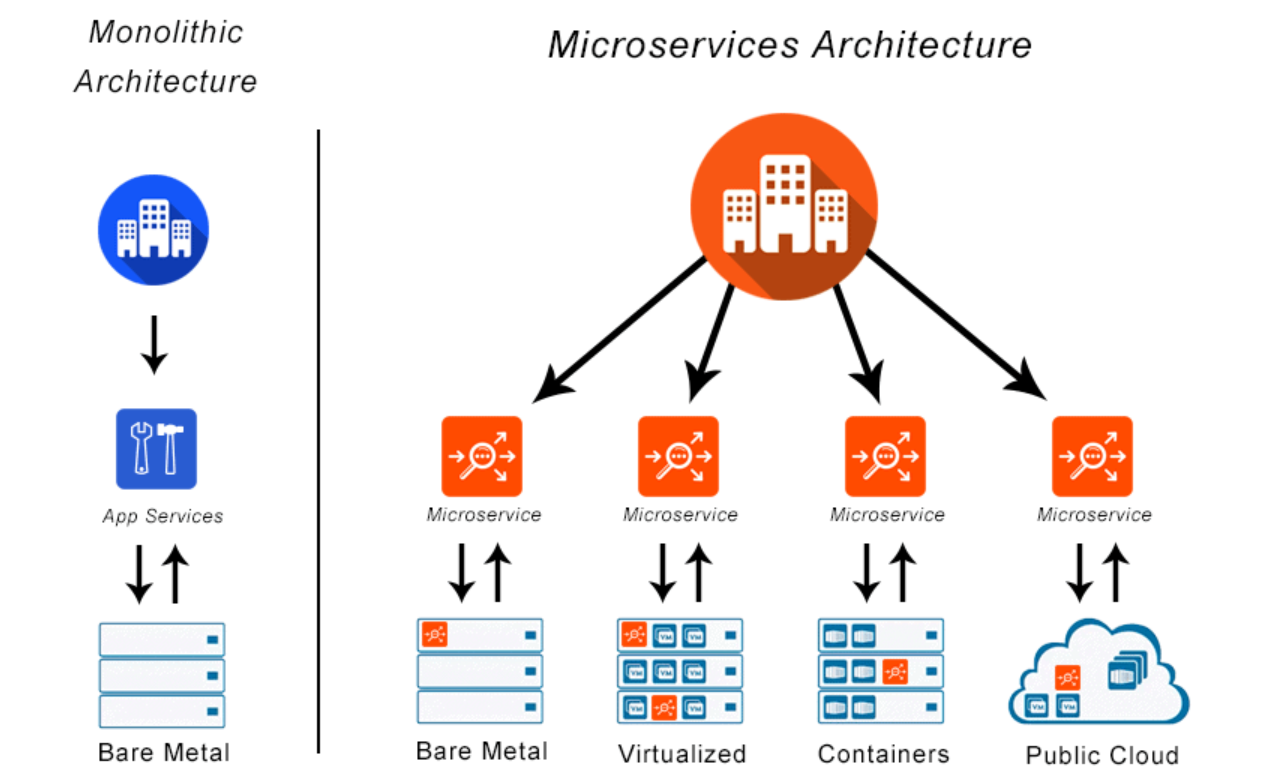




An application can be called cloud native if:

1. It has autoscaling feature
2. If it can run on any cloud

Microservices: Microservices is an architectural style that structures an application as a collection of small, independently deployable services, each focused on a specific business capability. These services communicate with each other through well-defined APIs (Application Programming Interfaces) and can be developed, deployed, and scaled independently.



Key characteristics of microservices architecture include:

1. **Modularity:** Microservices break down a monolithic application into smaller, manageable components, making it easier to understand, develop, test, deploy, and scale.
2. **Independence:** Each microservice operates independently, often with its own database. This autonomy allows teams to choose the most suitable technologies for each service and update or scale them without affecting the entire system.
3. **Scalability:** Microservices enable scaling specific services based on demand. This is in contrast to monolithic architectures where the entire application must be scaled, even if only a specific part requires additional resources.
4. **Resilience:** If one microservice fails, it doesn't necessarily bring down the entire system. The failure is contained within that service, and the rest of the application can continue to function.
5. **Technology Diversity:** Different microservices can be implemented using different programming languages, frameworks, and data storage technologies, as long as they communicate through standardized APIs.
6. **Continuous Delivery and Deployment:** Microservices facilitate continuous integration and continuous delivery (CI/CD) practices, allowing teams to deploy updates to specific services without affecting the entire application.
7. **Decentralized Data Management:** Each microservice typically manages its own database. This can lead to challenges in maintaining data consistency across services, and solutions such as event sourcing and distributed transactions are often used to address these challenges.
8. **Communication:** Services communicate with each other through well-defined APIs, often using lightweight protocols such as HTTP/REST or messaging systems like RabbitMQ or Kafka.

While microservices offer several advantages, they also introduce challenges, such as increased complexity in terms of deployment, monitoring, and managing inter-service communication. Successful implementation requires careful consideration of these challenges and the selection of appropriate tools and practices.