Kubernetes Agenda for Day 1:

- Microservices vs Monolithic

- Why Kubernetes

- History of K8s

- Knowledge Pre-requisites for K8s

- K8s Architecture

- K8s Objects

Monolithic vs Microservice:

---------------------------

Every enterprise application has a similar kind of layered architecture:

1. Presentation: The user interface.

2. Business logic: The application’s internal business logic.

3. Database access: Almost all applications need to access DB, either SQL or NoSQL.

4. Application integration: Quite often, the application needs integration with other applications. This is usually achieved via web service calls (SOAP or REST), or via messaging.

Even though applications have clear, logically modular architecture, but usually, most of the application is packaged and deployed as a monolith. There are actually some advantages to doing this.

Advantages of Monolithic Architecture:

- Simple Development

- Simple Testing

- Simple Deployment

- Simple Scaleup

Disadvantages of Monolithic Architecture:

- Flexibility

- Reliability

- Speed in Development

- Building complex apps

- Continuous deployment

Because of all the above drawbacks of monolithic applications, microservice architecture is becoming more and more popular day by day. So what is microservice-based architecture?

Microservice Architecture:

In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, usually via RESTful web services or messaging.

How Microservice Architecture Tackles the Drawbacks of Monolithic Architecture??

1. Flexibility: Microservices architecture is quite flexible. Different microservices can be developed in different technologies. Since a microservice is smaller, the code base is quite less, so it’s not that difficult to upgrade the technology stack versions. Also, we can incrementally adopt a newer technology without much difficulty.

2. Reliability: Microservices architecture can be very reliable. If one feature goes down, the entire application doesn’t go down. We can fix the issue in the corresponding microservice and immediately deploy it.

3. Development speed: Development is pretty fast in microservices architecture. Since the volume of code is much less for a microservice, it’s not difficult for new team members to understand and modify the code. They become productive right from the start. Code quality is maintained well. A microservice takes much less time to start up. All these factors considerably increase developers' productivity.

4. Building complex applications: With microservice architecture, it's easy to build complex applications. If the features of the application are analyzed properly, we can break it down into independent components which can be deployed independently. Then, even the independent components can be further broken down into small independent tasks which can be deployed independently as a microservice. Deciding the boundaries of a microservice can be quite challenging. It’s actually an evolutionary process, but once we decide on a microservice, it’s easy to develop, as there are no limitation in technologies.

5. Scalability: Scalability is a major advantage in microservice architecture. Each microservice can be scaled individually. Since individual microservices are much smaller in size, caching becomes very effective.

6. Continuous deployment: Continuous deployment becomes easier. In order to update one component, we have to redeploy only that particular microservice.