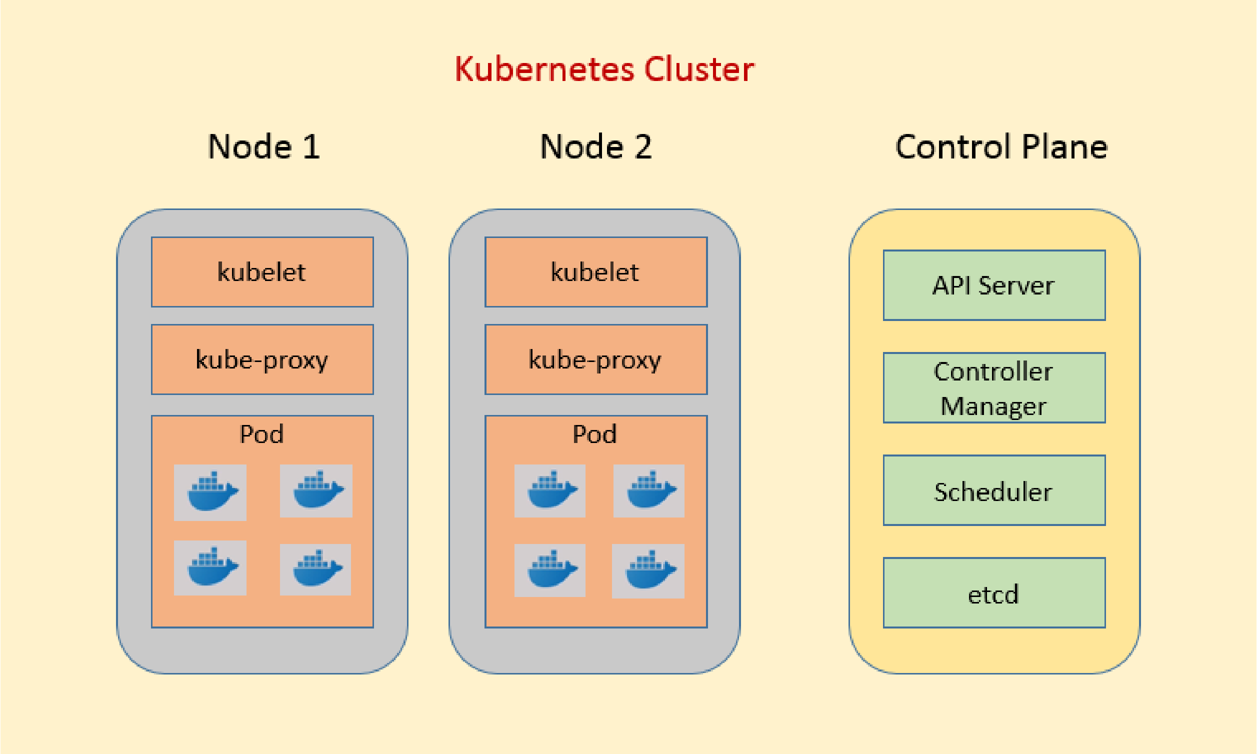
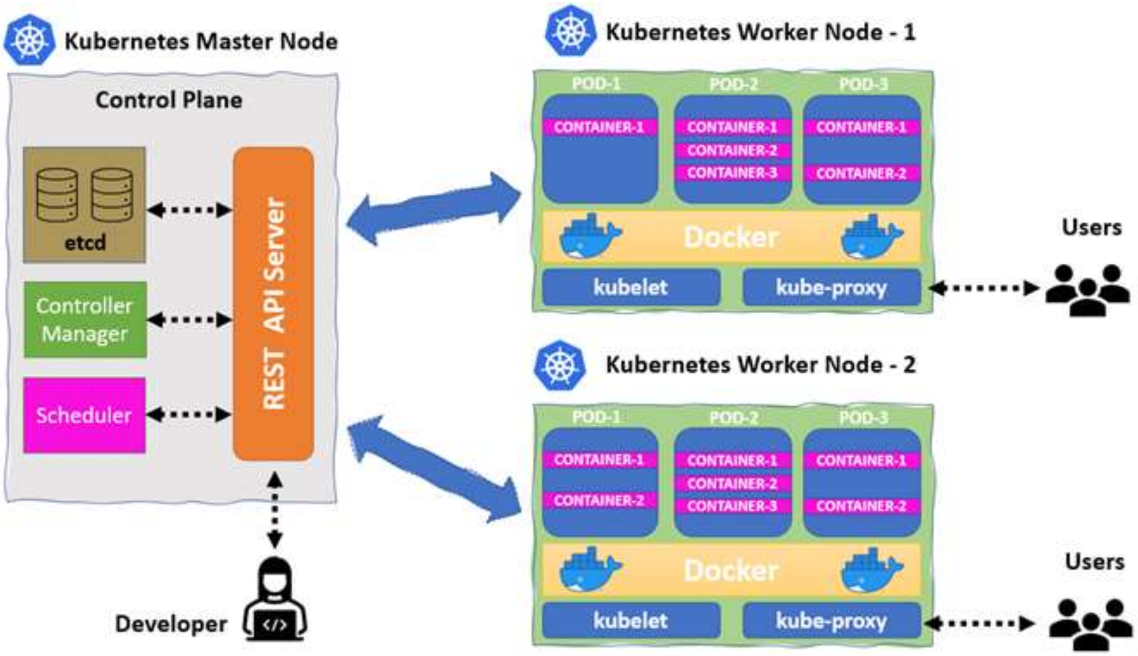
AmazingDevOps

Why are most companies starting to move away from tomcat server to Kubernetes?



Tomcat is a popular web server and servlet container that is widely used in enterprise applications for hosting Java web applications. However, with the rise of containerization and the need for more flexible and scalable application deployment, many companies are moving towards Kubernetes as their preferred platform.

Kubernetes is an open-source container orchestration platform that provides a way to automate deployment, scaling, and management of containerized applications. It allows companies to manage containerized applications across multiple nodes in a cluster, ensuring high availability, fault tolerance, and scalability.



There are several reasons why companies are moving away from Tomcat and towards Kubernetes:

1. Scalability: Kubernetes provides a more scalable architecture than Tomcat. With Kubernetes, you can easily scale your application up or down based on demand, while Tomcat requires manual configuration for scaling.

2. Resilience: Kubernetes offers built-in fault tolerance, enabling applications to remain available even if individual nodes or containers fail. Tomcat does not offer this level of resilience.

3. Portability: Kubernetes allows for the deployment of applications across different cloud providers and on-premises data centers. This means that companies can easily move their applications between different environments without having to make significant changes to their infrastructure.

4. Management: Kubernetes offers a more streamlined approach to application management, with features such as automatic load balancing and self-healing. This can significantly reduce the time and effort required to manage applications.

Overall, the move towards Kubernetes is driven by the need for more flexible and scalable application deployment, as well as the desire for more streamlined application management. While Tomcat remains a popular choice for Java web applications, Kubernetes offers a more modern and versatile approach to application deployment and management.

What does self-healing mean in Kubernetes?

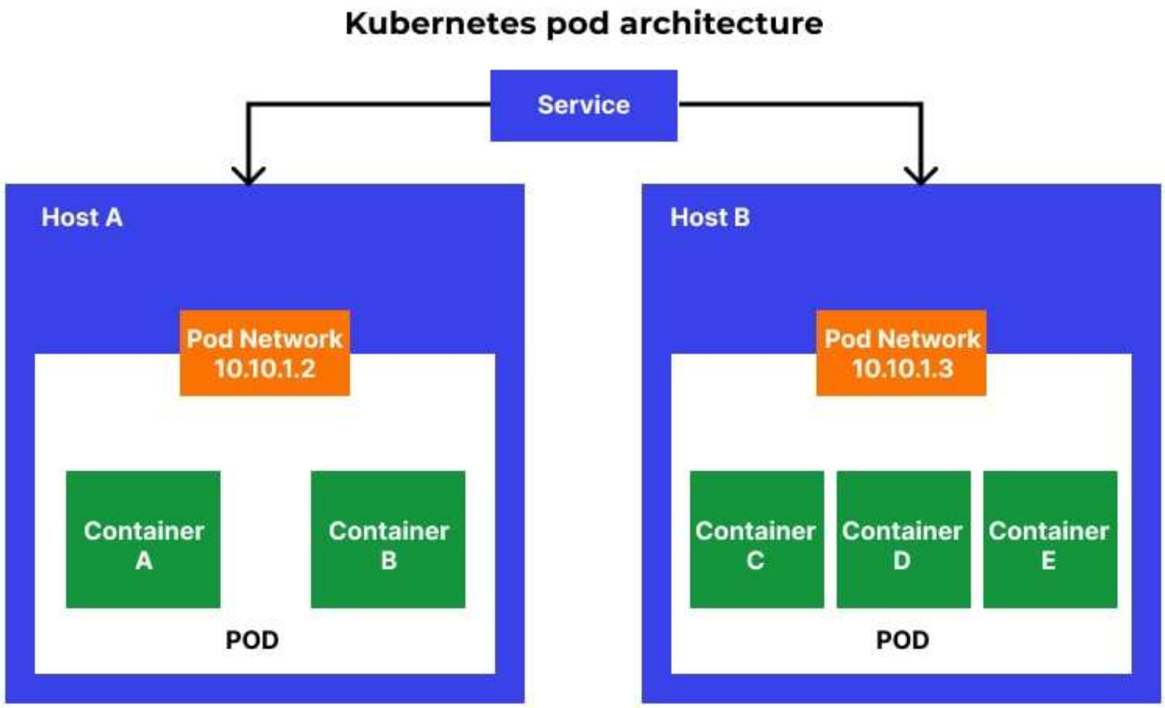
Self-healing is one of the core features of Kubernetes. It refers to the ability of Kubernetes to automatically detect and recover from failures in the system without any manual intervention.

In Kubernetes, self-healing is achieved through a combination of features such as health checks, replication, and automatic scaling. Here's how it works:

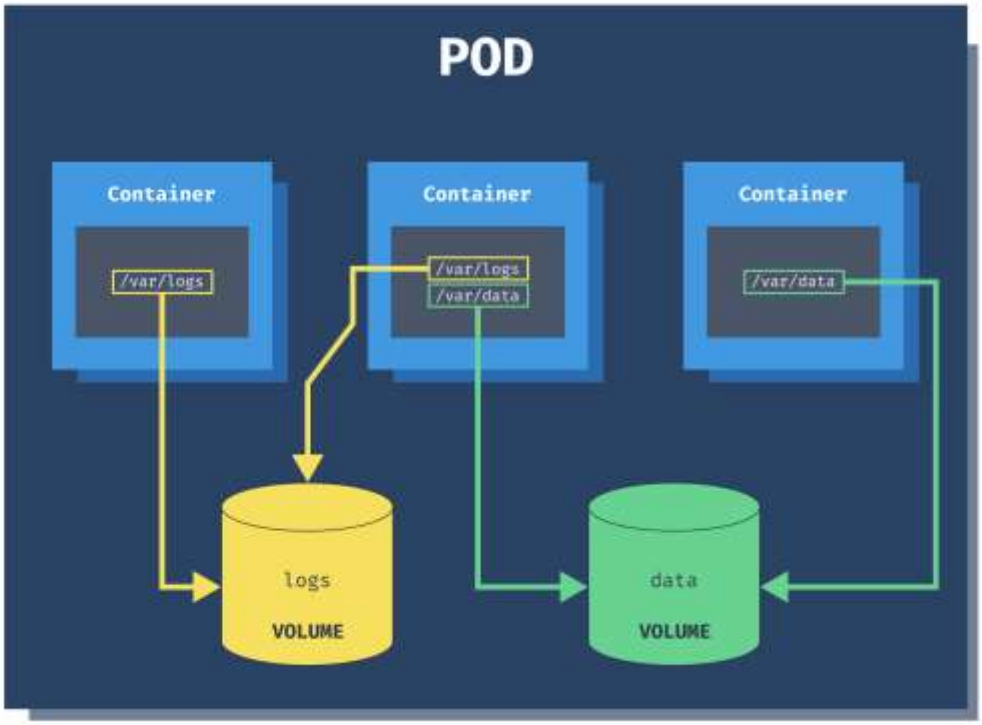
1. Health checks: Kubernetes periodically checks the health of the running pods and containers to ensure that they are functioning properly. If a pod or container fails a health check, Kubernetes marks it as unhealthy and stops sending traffic to it.
2. Replication: Kubernetes maintains a desired number of replicas of each pod, so if one pod fails, another pod can take over its work. Kubernetes can automatically create new replicas to replace failed ones.
3. Automatic scaling: Kubernetes can automatically adjust the number of replicas based on the current load and demand. For example, if the traffic to a service increase, Kubernetes can automatically create more replicas to handle the additional load.

With these features, Kubernetes can detect failures in the system and recover from them automatically. This makes Kubernetes highly resilient and helps to ensure that applications running on Kubernetes are always available and responsive to user requests.

What is Kubernetes pod



- In Kubernetes, a pod is the smallest deployable unit that can be created and managed. A pod is a logical host for one or more containers, and it provides an isolated environment for the containers to run in.
- A pod is created when you define a pod specification that includes the container image, environment variables, ports, and other configuration parameters needed to run the container. A pod can contain one or more containers that share the same network namespace and storage volumes.



www.startkubernetes.com

- When you create a pod, Kubernetes schedules it to run on a node in the cluster, and the containers in the pod are co-located on the same node. This means that the containers can communicate with each other using localhost, and they can share the same storage volumes.
- Pods are designed to be ephemeral and disposable. This means that they can be easily created, deleted, and replaced as needed. If a pod fails or becomes unhealthy, Kubernetes can automatically replace it with a new pod to maintain the desired state of the application.
- Pods are often used as the building blocks for Kubernetes applications. By grouping containers together in pods, you can create more complex applications that are easier to manage and scale.