Combine different techniques and tools that can help in monitoring services in Kubernetes and ensure high availability. It is suggested to take the following steps:

1. **By using Health Checks:** Kubernetes has built-in health checks like probes for readiness and aliveness. These probes can be configured for application containers in order to check the health of those containers periodically. The readiness probe checks if the container can receive traffic while the liveness probe checks if it is running properly.
2. **Cluster Monitoring:** Prometheus is a popular monitoring tool in the Kubernetes environment. Prometheus is able to scrape metrics directly from Kubernetes and your applications. This allows you monitor metrics such as resource usage, response time, error rate, etc.
3. **Notifications & Alerts:** Configure alerts using your monitoring software based on thresholds and conditions. One can, for example, set alerts based on high CPU and memory usage, latency in the network, or error late. After that they can then respond more quickly to problems. Then they integrate the monitoring tool with tools such as PagerDuty and Slack in order to be notified when issues occur.
4. **Horizontal Pod Autoscaling:** Kubernetes provides Horizontal Pod Autoscaling to automatically scale your number of replicas based on custom metrics or metrics such as CPU usage. HPA can be configured based on desired resource and performance thresholds, to scale up or down the application automatically. This ensures high availability in periods of increased traffic.
5. **Distributed Tracing:** Use tools such as Jaeger and OpenTelemetry to implement distributed tracing into your application. One can use distributed tracing to understand how requests flow across the microservices, and pinpoint performance problems or bottlenecks. It can improve your service availability and optimize it by analyzing the traces.
6. **Centralized Logging:** Set up centralized logging in your cluster or applications with tools such as Elasticsearch (EFK stack), Fluentd and Kibana, Prometheus, and Grafana. It can aggregate logs and analyze them from different sources to help with troubleshooting.
7. **Load Testing:** Regularly load test the application to determine its performance in different traffic situations. Stress-testing your system can be done using tools like Apache JMeter and Kubernetes specific solutions such as kube-burner. Load tests can also help to identify bottlenecks and in measuring the response time. They also validate that autoscaling, monitoring and other setups are working.
8. **Backup and Disaster Recovery:** In the event of a failure, implement a disaster-recovery strategy. Backup critical data regularly and configure replication to keep copies across different availability zones or areas. Consider disaster recovery solutions and tools that meet customer needs.

It is important to remember that the specific configurations and tools may differ depending on the application or infrastructure customers are using. To adapt to new requirements and increase the resilience of Kubernetes services, it's essential to continually monitor and refine the monitoring and reliability practices.