I am going to talk about the DevOps practice known as monitoring.

Monitoring is the collection and presentation of data about the performance and stability of services and infrastructure. Within a DevOps culture, monitoring is vital. It supports frequent deployments and fast time to market by allowing you to respond to any problems caused by those frequent deployments quickly. Of course, it promotes stability for the same reason by enabling you to address any issues that could be affecting stability quickly.

Monitoring tools collect data about memory usage, CPU, disk i/o in the usage of other resources over time. You can also collect application logs, information about network traffic, and a lot of other data that you can use to gain insight into the health and performance of your infrastructure.

The collected data can be presented in various forms. You could have charts and graphs that somebody can look at to see how things were performing. You can also use this data in the form of real-time notifications about issues if the monitoring tools notice a problem in production. This immediate notification to developers can get them to start fixing the error in the hopes to fix that problem before customers even notice it.

Let us look into what monitoring looks like. The most identifiable way is real-time notifications. We will dive into a scenario where the performance on a website is starting to slow down. Response times are getting longer and longer. The monitoring tool is watching the response times and notices that they're growing and promptly notifies an administrator so they can intervene before that performance degradation leads to actual downtime.

Another use case of monitoring involves postmortem analysis. That means looking at something after the event. This scenario will be on something that went wrong in production last night. It is, however, working now. Production is back up, and the customers are happy. But we still don't know what caused the problem that occurred last night. Thankfully, our monitoring tools have collected a lot of data during that outage, which is now available to us.

Let us join the team and have a look. We would now look at the data, and the developers and operations engineers can determine the root cause of last night's outage. It just happened to be a poorly performing SQL query, but now that we know what the cause was, we're able to fix it.

We will now cover why we should use monitoring. The first reason is fast recovery. The sooner we detect a problem, the sooner we can fix it. We want to know about a problem before our customer does. When something goes wrong in production, the worst-case scenario is for the team to find out about it by getting a call from a customer. We want to know about that problem before the customer calls me and already be working to resolve it and potentially even resolve it before the customer experiences the problem.

Monitoring also offers better root cause analysis. The more data we have about our production systems, the easier it is to determine the root cause of any particular problem. Without good monitoring, we would wind up making assumptions about what caused the problem in production. These assumptions could lead to more significant lag times to get the product back up and running while also not have the tools or information that we need to fix those problems for the future effectively.

Monitoring is critical within a DevOps context because it's one of the primary ways to offer visibility into production systems across both operation and development teams. Good monitoring tools give us valuable data that both developers and operations could use to understand code performance in production and potentially do optimizations to improve performance.

Another great benefit of monitoring is automated response. Data that comes from monitoring tools can be used alongside orchestration tools and other types of automation tools to provide automated answers to events. An example would be automated recovery from failures in the production system.

To sum it up, monitoring tools can be used to detect problems, and then an orchestration tool could be used to respond to that data by isolating the broken node and replacing it with healthy ones. Monitoring is essential to bringing together fast delivery and stability, especially in complex systems and at large scales.