Quality assurance and testing have become essential aspects of software systems execution.

1. With the rise of DevOps practices, how do you see testing's role evolving within this context?
2. Share your thoughts on how continuous integration, continuous delivery, and automated testing impact the execution phase.
3. Furthermore, consider the balance between speed and quality in DevOps-driven software systems execution.
4. How can development and testing teams collaborate effectively to ensure both rapid deployment and high software quality?

DevOps, and more specifically DevOps pipelines, are meant to streamline the continuous and automated deployment of software. In order to deploy robust code, it will need to be tested before deployment. This means that the testing will need to be done automatically since it would be incredibly expensive to manually test before every deployment. Testing will have to be increasingly more automated in order to take full advantage of DevOps practices.

It is interesting to see continuous integration is becoming more standard, especially in environments where the traditional methodology has been waterfall, like in the DoD. In my experience, automated testing has not caught up with automated deployments and is causing issues with software deliveries. For some software this isn’t an issue as it is not mission critical software and is more of a QoL product for the users. It is more important that the users get the software than it is working 100% of the time. However, sometimes speed of delivery is not the most important part. Mission critical software must work 100% of the time, therefore the testing (whether it is happening or not), will slow down deliveries, but quality is more important.

To get the best efficiency out of development and testing teams, the responsibilities of each team should be clearly defined. The development should focus solely on development while the testing team is focused on testing. Immediate and frequent communication between the teams will also foster a collaborative environment. Both teams need to be on the same page in regards to how the software needs to function. The developers need to make sure they are developing the right functionality and the testers need to know which tests should succeed and which should “fail”.