Testing is a cornerstone in the Continuous Integration/Continuous Deployment (CI/CD) process, yet, it's often overlooked or under-implemented in the tech industry.  
  
But why is it so critical?  
  
In CI/CD, testing is not just a phase; it's an ongoing practice that ensures each integration is verified, minimizing bugs in production.  
  
Automated tests in CI/CD pipelines help detect issues early, reduce manual work, and enhance code quality.  
  
Despite its benefits, testing can be missed due to tight deadlines or the misconception that extensive testing slows down delivery.  
  
The truth is, without rigorous testing, we risk more than just delayed deployments.  
  
We risk the integrity of our product, user trust, and ultimately, our brand's reputation.  
  
Let’s break down the steps of CI/CD testing that should never be missed:  
  
1. 𝗗𝗲𝘃𝗲𝗹𝗼𝗽𝗺𝗲𝗻𝘁 𝗧𝗲𝘀𝘁𝗶𝗻𝗴: Starts with unit tests to verify code at the smallest level, followed by local UI tests to ensure functionality.  
  
2. 𝗤𝘂𝗮𝗹𝗶𝘁𝘆 𝗔𝘀𝘀𝘂𝗿𝗮𝗻𝗰𝗲: Involves functional tests to validate software functions, integration tests for seamless collaboration, and independent UI tests to isolate components.  
  
3. 𝗦𝘁𝗮𝗴𝗶𝗻𝗴: At this stage, performance and load testing assess system efficiency under stress, followed by system testing to validate overall functionality.  
  
4. 𝗨𝘀𝗲𝗿 𝗔𝗰𝗰𝗲𝗽𝘁𝗮𝗻𝗰𝗲: Conduct ad hoc smoke tests for a rapid check of the basic functionality before the product reaches the end-users.  
  
5. 𝗣𝗿𝗼𝗱𝘂𝗰𝘁 𝗔𝗰𝗰𝗲𝗽𝘁𝗮𝗻𝗰𝗲: Disaster recovery tools simulate potential disasters to ensure rapid recovery strategies are in place.  
  
6. 𝗥𝗲𝗹𝗲𝗮𝘀𝗲: A/B tests optimize user experiences, penetration tests identify and fortify system vulnerabilities, and system & performance monitoring tracks health for optimal performance.  
  
Incorporating these steps into your CI/CD pipelines can make the difference between a product that excels and one that fails.  
  
Remember, skipping testing is not a shortcut; it's a detour that leads to longer and costlier development cycles.  
  
Let's discuss how we can prioritize testing to build better, more reliable software.

Activate to view larger image,

