The actual question asked should be when to automate and when to test manually. A conception exists automated testing is vastly superior to manual testing and the perfect state end goal for testing in a Dev Ops environment is 100% automation. In reality, both methodologies have their advantages and drawbacks. Discounting one approach over the other may result in testing process gaps and a decrease in the product quality. By utilizing a blended approach the strengths of both methodologies will lead to the most comprehensive understanding of an application and raise the quality of the testing process and product.

This goal of this page is to provide a set of guidelines to determine which method can be used to more effectively test an item based on a combination of speed, quality, effectiveness, and long term objective.

To determine which testing methodology to apply to a ticket consider the following guidelines:

**Use manual testing when:**

* Subjective functionality such as UI visual changes/updates or usability/user experience
* Hot Fixes (depending on severity and need to promote to production). These can subsequently be automated if they meet the guidelines below.
* New or rapidly evolving functionality less than 80% stable/consistent
* Items not having a consistent predictable outcome
* Tests requiring ad hoc/random testing based on domain knowledge/expertise. Usually associated to exploratory testing.
* Extreme Edge Cases
* Bugs that are non-critical functionality and/or unlikely to occur again (such as fixing a typo)

**Use automation when:**

* Business critical processes which, when not working as expected, increase risk or cause damage to the business
* Items require no manual intervention
* Low maintenance cost (that is, the upfront cost of automation is less than repeated manual testing or maintenance of automated scripts)
* Repetitive tasks
* Scenarios prone to human error
* Tests executed with multiple or large data sets
* Smoke Testing
* Regression Testing

Possible next steps:

1. For each application create a smoke test plan that can be automated and run after each build
   1. Identify minimal critical functionality that would be smoke tested after a build
   2. Document the scenarios
   3. Create a test plan with the steps for each scenario
   4. Start automation scripts for proof of concept

***Smoke Testing definition*:** A non-exhaustive shallow set of test cases used to validate the most critical functions in an application work as expected after a build. By verifying base functionality works as expected testers can decide whether or not to continue to the next level of testing. For more information see: <http://softwaretestingfundamentals.com/smoke-testing/>