Tips for organizing tests

You want to start organizing tests from day one, right? Or, maybe you already have your tests created and it's time to organize them.

Organization should start by classifying your tests individually, so you can easily find and filter them out.

At Test level

Classify your Tests

- use *labels* to tag your Tests (or your Test Sets); if you start having many testers and your project grows, having a well-defined list of labels may help
- use the *priority* field, so you can distinguish between tests
- assign the Test to the proper *component*

Using Test Sets

Organize Tests in Test Sets

- group tests in as many Test Sets as it makes sense to you, either because they're testing the same feature or the same UI component, or if they exercise the most critical use cases
- classify your Test Sets, similarly to what you did for Tests

Organize Tests for regression testing

- you can create a Test Set for this purpose and add a specific label to it (e.g., "regression"), and then you test for requirements of previous versions
- you can create a "regression Test Set" for each version by cloning the previous version regression test set and add more tests

> Example: You're working on product v3.0 and want to do regression testing to make sure you have not broken anything. If the previous version, v2.0, had already some regression tests in a specific test set, then you can clone it and add some more tests for validating v2.0.

Using the Test Repository

The Test Repository organization follows an hierarchical, tree-like way of organizing Tests within folders. Since a Test can only be part of one folder within the Test Repository, you have to think in an hierarchical way "What is most important? How do I look for the Tests to schedule executions?"

- create top-level folders for the things that are most important (e.g., "performance", "security", "smoke tests")
- create the hierarchy from the things that are most relevant and then decompose each one; for example, if your tests are the "components" of the system, you can create a top-level "components" folder and sub-folders for each sub-component
- avoid mixing execution- or planning-related aspects in the Test Repository or you'll get messed up
- don't organize the Tests based on things that are volatile or can rapidly change