Item 353
git_comments:

1. Configuration for unit/integration tests section 3 of 3 (profiles) ENDS HERE.
2. Configuration for unit/integration tests section 3 of 3 (profiles) STARTS HERE. Use the following profile to run Integration tests. From the command line: mvn clean install -P integration-test or: mvn integration-test -P integration-test * Note that if you do: 'mvn test -P integration-test' integration tests will not run, because the test phase is before the integration phase in the default maven lifecycle. * Also note that unit tests will also be run when integration tests are run, because the integration-test phase is always after the test phase in the default Maven lifecycle. See also: surefire plugin section and properties section.
3. Configuration for unit/integration tests section 1 of 3 (properties) STARTS HERE. Property for running integration tests with profiles at the command line, where you do: mvn integration-test -P integration-test See also: the surefire plugin section and the profiles section.
4. Configuration for unit/integration tests section 2 of 3 (plugins) ENDS HERE.
5. Configuration for unit/integration tests section 2 of 3 (plugins) STARTS HERE. This is the core piece for being able to manage unit tests and integration (='system') tests within Maven by using surefire. See also: properties section and profiles section.
6. <maven.test.skip.exec>true</maven.test.skip.exec>
7. Configuration for unit/integration tests section 1 of 3 (properties) ENDS HERE.

git_commits:

1. **summary:** [HELIX-77] Applying patch from Max to clean up pom and separate unit tests from integration test
   message: [HELIX-77] Applying patch from Max to clean up pom and separate unit tests from integration test
   label: test

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Skip tests in default build or separate unit/integration tests
   description: A couple of miscellaneous questions about how you require building helix. Would it be possible to skip tests by default in your pom? Normally, after cloning sources (for experimenting/browsing), you would do: mvn clean install which for helix should take maybe 10-30 seconds. Because you have tests turned on by default the build takes much much longer (it looks like there are integration tests in the mix, not only fast unit tests). I think it would be more usable for people doing 'mvn clean install' rather than the more verbose: mvn clean install -Dmaven.test.skip.exec=true Attaching a patch that shows what i mean in the pom. The other related question is about integration tests. The reason why takes so long to build is maybe because not all tests being run are unit tests. Sometimes there is no clear distinction between integration and unit tests, but the point is that the build should be fast, so one way to distinguish, arguably, is that if a test is slow is not a unit test. I can show one way of separating unit and integration tests in Maven (Maven does not have very good support for this, there are different ways to work around it) in a later post to this. Please let me know if there is any interest in correcting this.

jira_issues_comments:

1. **body:** Defaults build to skip tests. Added also a default goal with <defaultGoal>. This is unnecessary, but for those who know it, it is useful, as it just takes command 'mvn' (shorter) to do the most common operation of building.

label: code-design

2. **body:** Thanks Max, I agree we should separate out the tests. We also run the tests sequentially because of integration tests. We have some separation in terms of package layout. It would be great if you can add a way to separate unit and integration tests. But we definitely want integration tests to be run in jenkins. Thanks for the patch will apply it today
   label: test
3. Applied the patch, thanks Max
4. Integrated in helix #664 (See [https://builds.apache.org/job/helix/664/]) [Helix-77] Setting build to skip tests by default (Revision 32a4e37d9397cefc8202cd489ec05ec504ddd066) Result = SUCCESS
5. **body:** Thought i reopened this one so that i add the patches i was talking about. If i should instead open different Jira issues, please let me know and i will. Going to include two patches: 1) the first patch are just cleanups i ran into. Simply made plugin management and dependency management consistent (did not introduce it, was already there, just made it consistent) 2) This is the critical one: added support for separating integration and unit tests in the main pom (see comment lines in pom). So now, when users issue 'mvn' (defaults to 'mvn clean install'), they would compile and run only designated unit tests. When users (and Jenkins i guess, from previous comments) issue a 'mvn -P integration-test' then not only the unit tests will be run, but also the designated integration tests. The pattern i picked for separating test types is "**/integration/**". That pattern means that any tests under a folder named 'integration' is treated as integration tests, but any other pattern can be used. For example, if you wanted to mark as integration tests all tests ending with "IntegTest.java", no matter what folder they are into, then the pattern would be: "**/*IntegTest.java". Note that, for this change to make sense, i had to remove my previous change in the original patch i had submitted, where i was setting maven.test.skip.exec=true by default (I commented out that line here) Before this change can be submitted, two things must be done: 1) the preferred pattern to designate integration tests needs to be chosen ("**/integration/**", or "**/*IntegTest.java", etc) 2) tests need to be marked according to the pattern above (that normally implies moves or renames). If all or almost all tests now in Helix are slow integration tests, then maybe the fastest thing to do is reverse the logic i did in the main pom and designate "unit" tests instead, with a pattern like ("**/*UnitTest.java"). Maybe the renames that way will be less.
   label: code-design
6. Made plugin management and dependency management consistent with other pom files in Helix.
7. Added support for separating integration and unit tests in the main pom.
8. Hi Max, I am not able to apply the patch, looks like some changes were made the pom.xml. Can you please merge the changes and re-submit the patch ?
9. Just verified, not sure why, going to check and resubmit.
10. cleanupsNew.diff: this is patch 1 of 2 (must be applied first)
11. attachment separateintegtests.diff: this is the patch that adds support for integration tests as separate from unit tests. It is patch 2 of 2.
12. patchBoth.diff: this is a squash of both patches cleanupsNew.diff and separateintegtestsNew.diff. I provided separate patches, because i am not sure the second patch can be applied before moving/renaming integ tests around or finding the correct integ tests pattern (see comments about this above)
13. Integrated in helix #671 (See [https://builds.apache.org/job/helix/671/]) [HELIX-77] Applying patch from Max to clean up pom and separate unit tests from integration test (Revision 6ffeb30b69203827f8333ad8a625b80b46d6facd) Result = FAILURE g.kishore : Files : * recipes/distributed-lock-manager/pom.xml * helix-admin-webapp/pom.xml * pom.xml * recipes/rsync-replicated-file-system/pom.xml
14. **body:** Hi Kishore, i checked the failure, which is test TestClusterAlertItemMBeanCollection, but at first does not seem to be related to these changes (no code changes, pom changes). Unfortunately, it is difficult for me to test what should work and what should not, because i get all sorts of exceptions when i run the full build on master as well. So, i guess my question is: how do i go about ensuring that a patch does not make the build fail? Is there a map of all tests that are integ as opposed to unit? This is an important question BTW, because, again, my second patch cannot really help without first knowing how to map/distinguish them. Maybe should do this: how about applying just the first patch for the moment (cleanupsNew.diff), and then deal with the second one later (i can open a separate child issue)?
    label: code-design
15. **body:** Hi Max, You are right, the failure was not because of the patch. The Jenkins box provided by Apache is not the best and the time taken for tests to run vary quite a lot for every run. We never have this issue within our company. There are couple of things we need to re-design in Helix to make the tests run in a reliable way. For the integration test, we bring up zookeeper and tear it down after the test is

complicated. This has resulted in many tests getting stuck at shutting down zookeeper and some times the port is not freed by the previous zookeeper so the new zookeeper never starts. Most of the integration tests should be under integration package. If there is a test that starts/stops zookeeper but not under integration test we should move it under integration package. We rely heavily of integration tests, they catch most of the issues. Other Apache projects have this automated QA job setup that applies the patch and does some checks like did coverage come down? find bugs increased? compiles, tests pass etc. I dont know how to set it up but I think we need to do some clean up in the existing test framework so that we get a reliable mechanism to say if a patch broke the build. We should create a separate jira for sanitizing test cases. Thanks again for driving this, feel free to suggest improvements.
label: code-design

16. Sorry, just realized patch is already applied - message was from Hudson. Ok, please let me know how to go about this based on my prev comments/questions. Please note that with my change in place, in order to run the build on Hudson as before, Hudson *needs to run*: mvn clean install -P integration-test The previous command alone: mvn clean install will *not* run any tests under a directory containing a folder named "integration".