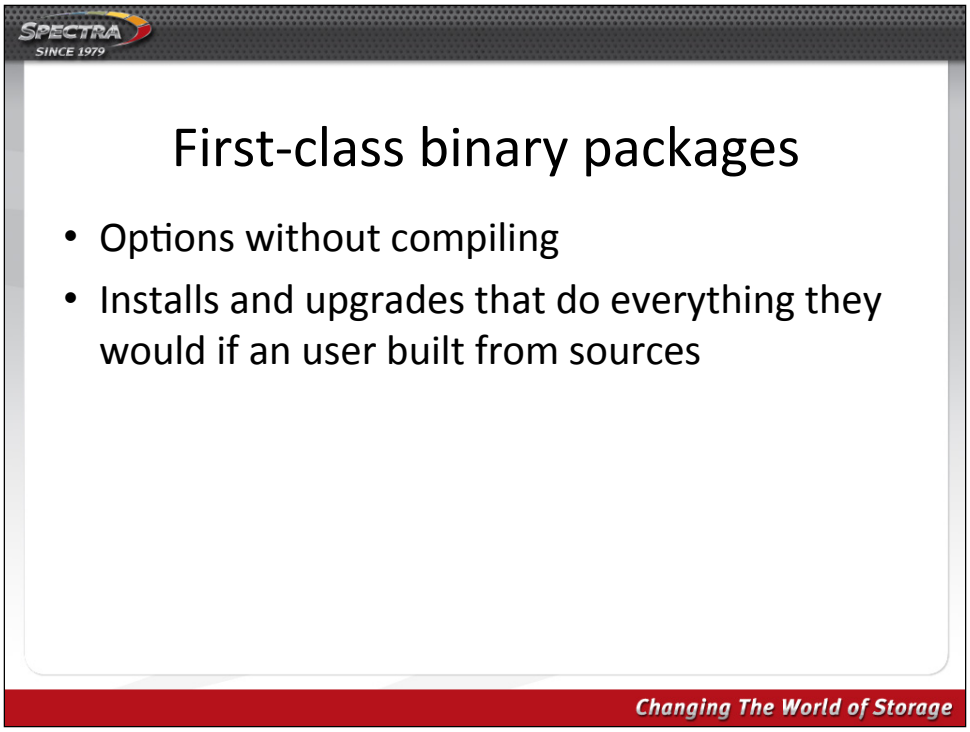
SPECTRA
SINCE 1979

Packaging: A Vision

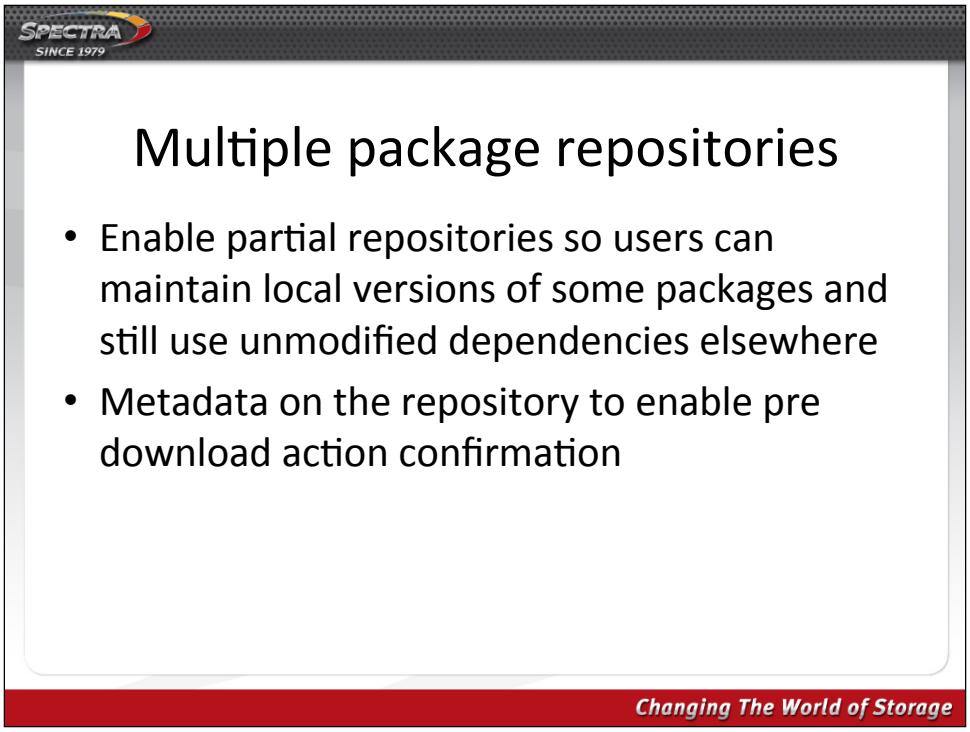
Will Andrews

Changing The World of Storage



-It's long been a tradition in FreeBSD ports that to get the proper set of features or behavior, one must compile.

-But for many users, this is a significant time sink that's simply not necessary or desirable. And FreeBSD can do better.



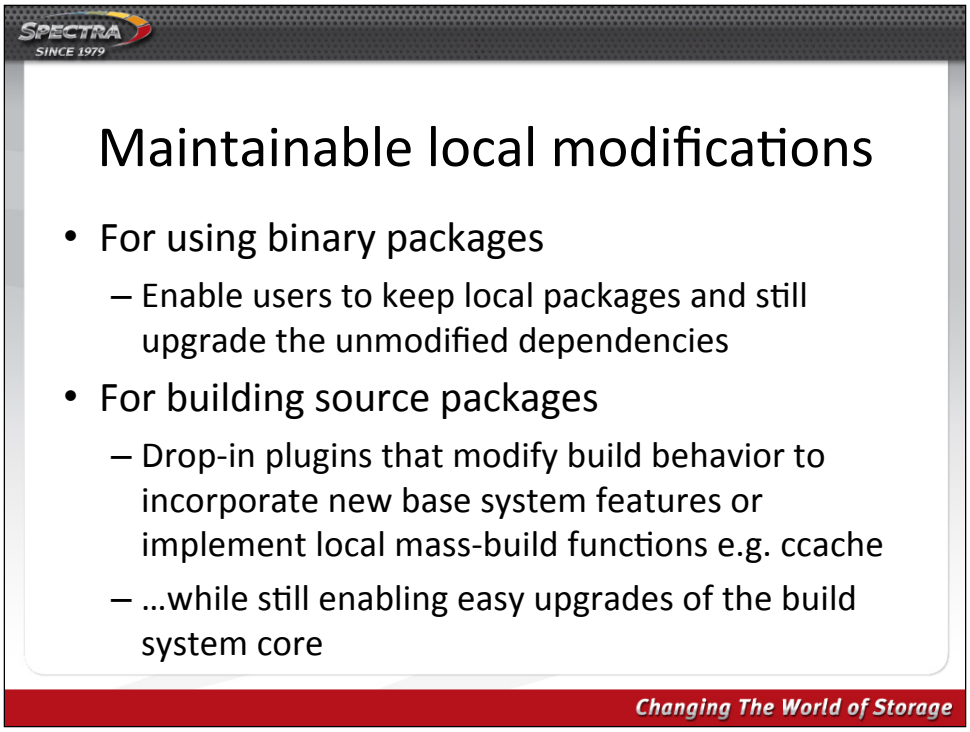
-Multiple package repositories sounds simple, but it implies a number of features important to many users

-First, the dependency resolution mechanism must be able to reference other packages that are sufficient enough to meet their requirements, and not necessarily exactly the same version it was built against.

-Second, since the dependency resolution may not be able to satisfy all of a package's requirements on a given repository, metadata files must exist on the repository to bridge the gap. This also allows pre-download confirmation for users.

-Third, the version comparison dependency resolution mechanism would also enable incremental updates of package repositories. Packages would not need to be built as part of one giant set; instead, package builds can be done incrementally. This would save a lot of time building them and transporting them around mirror sites etc.

-Fourth, once all of that is done, packages not part of the main build can be supported properly. This enables commercial vendors that ship their software as packages like everything else, either directly to users or as part of appliances. It also enables larger sub-groups of packagers that need to test their packages, to provide only the packages they modified.

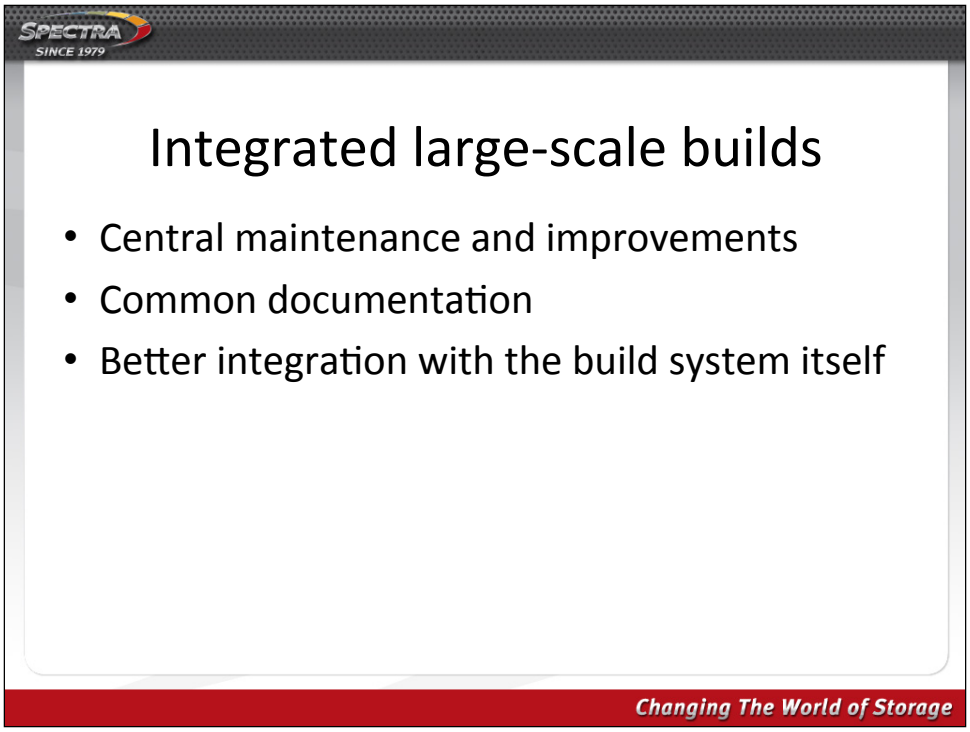


This is a continuation of the previous slide. FreeBSD can do better not just for binary packages, but also for local source build modifications.

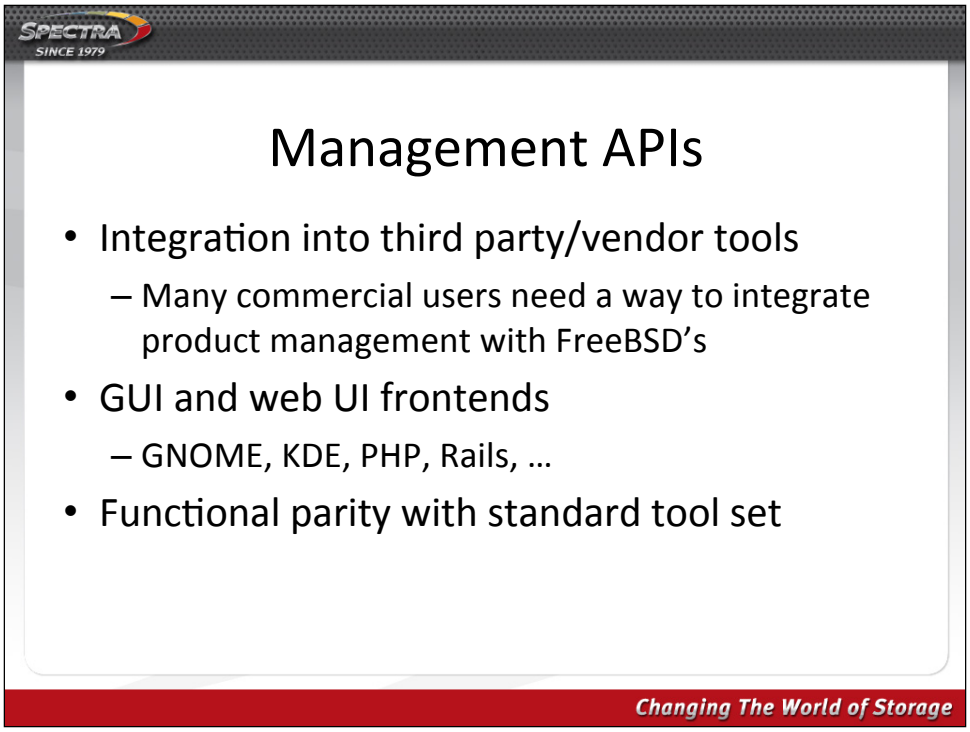
-Supporting plugins for the build system would enable custom behavior only pertinent on certain versions of FreeBSD, without needing to include it in the core.

-Plugins also enable custom behavior only available on systems with certain other infrastructure, e.g. cluster compiling with ccache. This can be true of commercial vendors using FreeBSD.

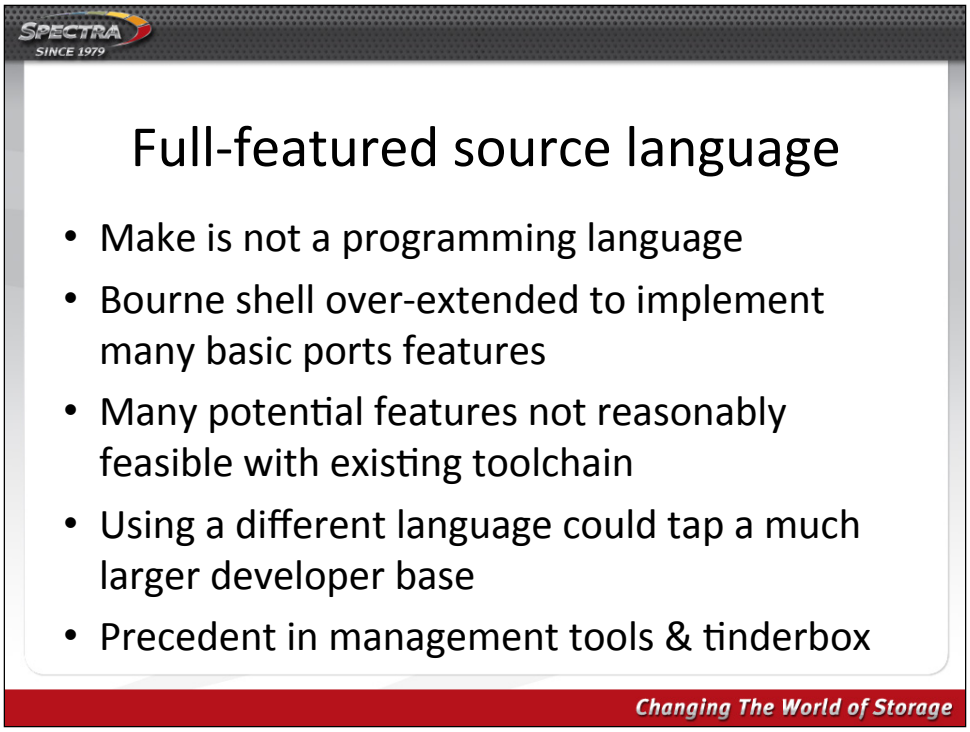
-Plugins enable these behaviors while still allowing the core to be upgraded independently. As long as the interfaces remain stable, of course.



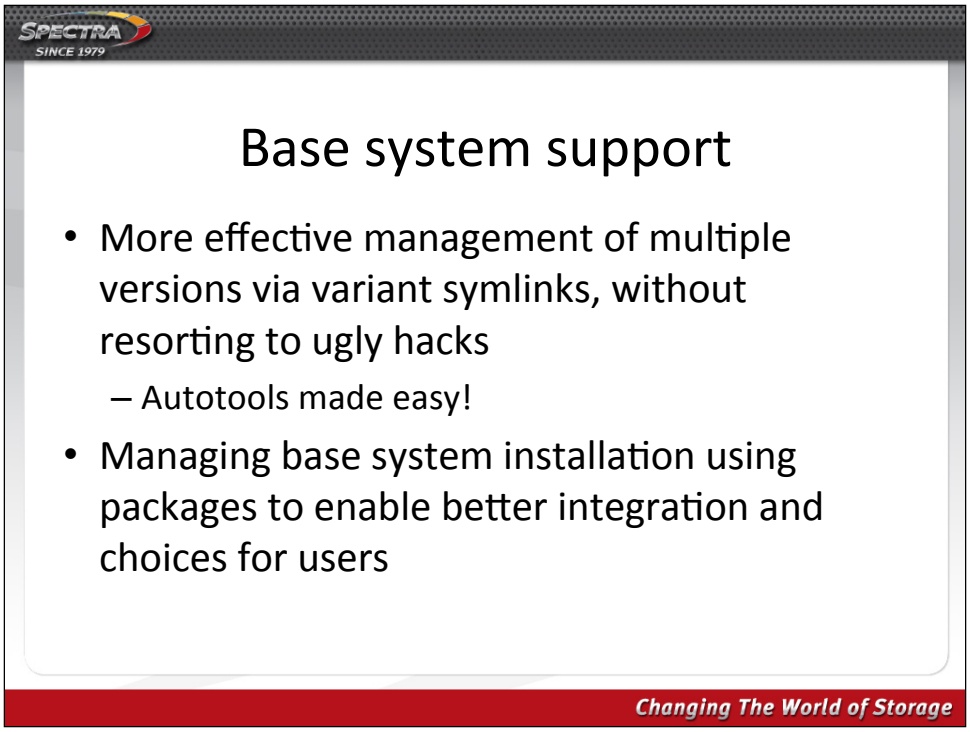
For over 10 years, there has been several different facilities for mass building of packages. Sadly, none of them fully integrates with the package system. They're built on top of it, and as separate projects. That costs users time if they need to build a large package set in a manageable fashion. By integrating these features into the package system itself, perhaps via the plugin mechanism, we could have something that has better integration, documentation, and a less balkanized developer community around it.



This one is relatively straightforward. There is a need to provide APIs for third party and vendor tools that allow packages to work much the same way they would if the user was using the standard tool set. Most other OSs have GUI and web UI tools that operate by taking advantage of the APIs available to accomplish this goal.



FreeBSD Ports has its roots in the tools that were available in 1995 and it shows. In the last 15 years, despite the hard work of many smart people, the feature set of the build system core continues to improve at a glacial pace. Anyone that has worked on a major feature involving large changes to bsd.port.mk knows how difficult it often is to implement those features using the existing tools, especially compared to other, modern scripting languages. Make is excellent as a build tool, but it's been contorted beyond its intended use to do what ports needs to do. In light of that, FreeBSD should move to a different language more suited to the job. There is precedent in external tools that build on ports that point to this fact.

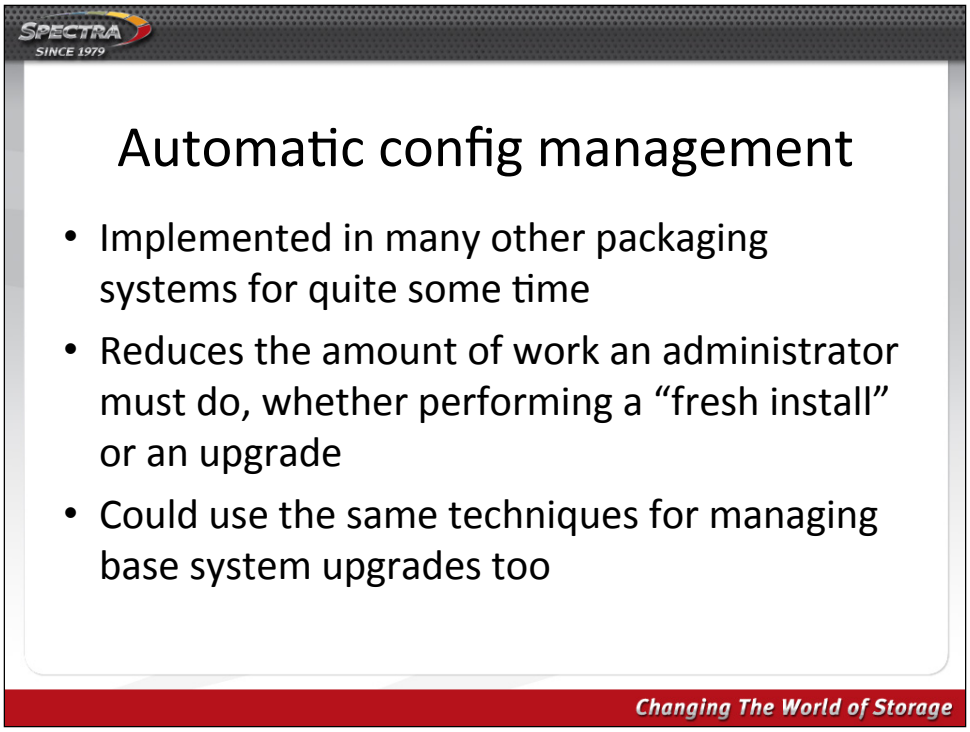


There's no reason why the "base system" and package system need to be separated. In fact, both could help each other out with certain problems.

-Kernel support that improve the usability of packages would make life easier for many users. One possibility could be to use variant symlinks to provide different versions of autotools.

-Userland support could include advanced build features like dependency tracing as proposed by jbuild.

-Incremental updates for the base system could be a lot easier as they'd no longer need to be done via an all-or-nothing updating mechanism. If binary packages supported options, then optional behavior for the base system could also be supported this way.



This is a great feature where a package can use its knowledge of installed packages and their high-level configuration to perform automatic configuration of the packages. So, for example, installing PHP could cause it to be automatically added to Apache or another web server’s configuration. Of course, such behavior could be disabled or otherwise manageable by the administrator.

The base system could leverage the same framework.

Faster package build process

- Using a modern language alone would make the package mechanism run faster
- Implement an automatic dependency resolution mechanism that gets rid of needless third party build actions, speeding up source build time

SPECTRA SINCE 1979

What's next?

- Let's build the packaging system that provides the best user experience feasible on FreeBSD!
- Work has already been done on a prototype system that implements many of these ideas