Totally renewed and ready for the next 10 years
Egg, Switzerland, 12 July 2016: *With version 2016/VII, ArchivistaVM is equipped with completely new technology. There are two pieces of good news in this context: Firstly, the new version is 100% compatible with the previous versions, both in the web view and on the console, and secondly, the technological base does not contain any dependencies on third party products. The following blog post shows why the second point is of crucial importance.*



ArchivistaVM arises from Proxmox 1.x
ArchivistaVM emerged in 2009 as a fork from Proxmox 1.x. Step by step, dependencies have been removed, with the result that ArchivistaVM is much slimmer than is the case with Proxmox products. For comparison: The current version of Proxmox is 700 MB. With ArchivistaVM, the figure is 60 MB for the free version and around 100 MB for the commercial version (with a web browser and X-server, which are not included with Proxmox).
Of course, the size of the ISO file that is delivered is not the only way of measuring things. The differences between Proxmox and ArchivistaVM have become significant in the meantime. The reason for this lies in the versions 1.x of Proxmox, as they were realised with **EmbPerl.**
EmbPerl was an attempt to implement Perl in the way PHP works. This means that the application is delivered embedded in HTML pages. In contrast to PHP, which has become widely used, this was never the case with EmbPerl. To put it in general terms, **EmbPerl** has unfortunately not been developed further since 2012.

Why ExtJS does not fit with ArchivistaVM



Proxmox solved the problem by reprogramming the whole web application, using **ExtJS.** This is a JavaScript framework. The visual part of the application is (in simplified terms) implemented with components that are very powerful, but also very extensive. **ExtJS** is distributed by the company Sencha, under a GPL licence as well as under a commercial licence. However, the conditions for the GPL version are constantly changing – more obstacles than help are experienced when trying to pursue development without the commercial version of **ExtJS.**
It was therefore already clear many years ago that ExtJS would not be introduced into ArchivistaVM. Not only because it was/is not clear to what extent the GPL version of **ExtJS** was/is usable, but also because it cannot be predicted whether or not the framework will still be in place in 10 or 20 years. In addition, the size of the framework is eye-catching – ExtJS 6.0.1 comes with 531 MB of resources, well over 44,000 files and almost 2.7 million lines of code. In short, such a 'monster' is quite simply not a good fit with ArchivistaVM.

EmbPerl is smaller, but no longer up-to-date

In comparison to that, EmbPerl is really tiny, with just 123,911 lines of code for the entire package. But, and this is the bad news, EmbPerl is only stable when used in conjunction with Apache 2.2. All attempts to migrate EmbPerl to the current version 2.4 of Apache have failed so far. Although EmbPerl is 100% open source, the maintenance work for well over 100,000 code lines is not trivial. It was therefore clear that EmbPerl had to be replaced.

Evaluation of JavaScript libraries

Various alternatives were tested and discussed over a period of 1 to 2 years. It

would have been convenient to be able to use a different modern JavaScript framework instead of ExtJS, but it was not possible to find any suitable alternatives apart from commercial 'monsters'. The most likely candidate was **W2ui,** see **http://w2ui.com.** With approximately 16,000 lines of code for a whole set of components, this library is certainly not oversized.
A problematic issue with the migration of ArchivistaVM was, however, that a lot data was transferred to an internal daemon with the previous code. In addition to a new front-end (JavaScript), a large part of the application would have had to be implemented on the server side (in Perl or another language). But, above all, it would have been impossible to maintain the existing web interface. In other words, ArchivistaVM would no longer have been ArchivistaVM.

Native Perl code as a solution for ArchivistaVM

A decision sometimes needs a bit longer in order to be taken. After numerous tests and discussions, the decision was made to transfer the 'old' fragmented code to native Perl code (without EmbPerl). This admittedly meant that all the approximately 80 files had to be adapted, but it also provided an opportunity to rework the code. It was thus possible to reduce the code base from over 40,000 lines to around 10,000 lines, and the number of files similarly decreased by three quarters.



The effort required for this migration amounted to a few hundred hours (including the entire evaluation). Any other solution would have required much more effort, not to mention the creation of new dependencies. It may be the case that such a blog entry may not seem terribly exciting at first glance. Of course, it would be easier in terms of marketing just to write that there is a completely new application, and then list a thousand new features. But, is it not the case that those who create the same application twice have really done their 'homework' thoroughly?
To conclude, it should be mentioned that the **new ArchivistaVM version 2016/VII provides 100% compatibility with the already very stable base of ArchivistaVM.** Version 2016/VII can thus be used productively without any problems. The new version of ArchivistaVM is now available as both the **free Mini version** and the version for customers with a maintenance contract. With this in mind, here's to the next 10 years with ArchivistaVM.
P.S: ArchivistaVM has so far been available on every ArchivistaBox with an Intel/AMD processor, even with ArchivistaDMS (select Home button). In contrast to the previous versions of ArchivistaVM, which do not run with ARM processors,

this is no longer an issue with the current version of ArchivitsaVM. The future will show whether ArchivistaVM will also find its way into the ARM-based ArchivistaBoxes.