GNOME and Buildstream, two (three?) years later
Abderrahim Kitouni

>> Hello. Abderrahim, I just uploaded your presentation. I can hear you well. Also, if you want to share your camera during the talk. We'll be starting in five minutes.

>> Sorry, but I don't have a webcam.

>> No problem. I can hear you fine.

>> Do you have any slides to share? Abderrahim, do you have any slides to share?

>> I'm seeing these weird texts.

>> Yeah. I see them too. Strange. You have time to export them again?

>> I actually have no idea what's going on.

>> Okay. Do you want to go ahead with this? They have no resolution or something like that.

>> Okay. This looks better.

>> So welcome, Abderrahim. Yes, let's start. Thank you.

>> Thank you.

>> So today I'm going to talk a bit about GNOME and BuildStream. BuildStream is the tool we used to build GNOME. You probably have seen the GNOME OS talk on Wednesday. We used it also to build Flatpak and other things. So what is BuildStream? BuildStream is a Free Software tool for integrating software stacks. It takes inspiration, lessons and use-cases from various projects including Bazel, OBS and Reproducible Builds, Yocto, Baserock.

So the idea that one can have is have multiple standards. We have a new standard and the other standards that are still being used. So the next step is how we go back to using only one. So this is a follow up to Michael Catanzaro's talk in 2018. By searching I found all the talks even from Emmanuele Bassi and Tristan as well. There have been two previous talks on BuildStream, Sam Thursfield has made some of these talks.

I'm going to talk about what we did and what we want to do with BuildStream. The first thing I can say we achieved is replacing GNOME continuous. What is GNOME continuous? The is the continuous integration and delivery platform of GNOME. What this means is we continuously build every part of GNOME. I think people are more familiar with it now that we use GitLab.

So what continuous does is that it builds every module that changes, commit the result to OSTree, build a VM image based on that and test the resulting VM image. At least that's what it did when it was working. So what we have now with BuildStream is that we have effectively replaced

continuous. As continuous was shut a few months ago, we are better than the last days of continuous. In the last days there was a problem and now that we can do that with BuildStream we have effectively replaced continuous. It is not without disadvantages, it's slower than continuous mainly because continuous used OSTree. But BuildStream uses something else and we have to compare things and it's a bit slow which we hope it will be resolved.

Continuous is used to build 886 and both 32 and 64 bit. But we don't build for every change. At minimum we build nightly but sometimes every time there is changes to the BuildStream files we build. And this is the perhaps something good is that it's easier to use if you want to build locally. Now with continuous we build things locally, but it wasn't easy so that's why we believe nobody fixed it when it was broken.

As of today, the images can be upgraded using OSTree. So perhaps less than hour you should be able to upgrade if you downloaded VM image two days ago with using OSTree. So the missing part is that we don't test the generated images. We have no way to know for sure. But we need some help. One of them is OpenQA. So I said you can try it. This is the GNOME OS that was talked about on Wednesday. We have two variants available. One is called user for "ordinary users." And devel for development.

The VM image, when you download it, the root was OSTree admin switch and it's this command. So the next part is replacing Flatpak-builder. Flatpak-builder was used multiple ways. This is the first thing we did with BuildStream. So it's easier to have all your dependencies without having to have them on the whole system. So what does BuildStream provide? You have rudimentary Flatpak plugin. This isn't bad. But for apps it's easier to have something that's closer to Flatpak. It's easy to write.

The second thing is we need one f-b for each Flatpak builder. You need also a different file. I think the most important one is BuildStream is not integrated with the Flatpak builder. It has a few advantages. The main one is workspaces. I can open a workspace in BuildStream in the directory and I change things there and I have them being applied. This is similar to what Flatpak does. It lets you modify one module locally and build the resulting Flatpak on time.

In short, BuildStream is great for building Flatpak. It is great for developing apps like we can have workspaces that helps us. At the same time the app and one of its dependencies, but it's not so great for building apps -- I mean, for writing the mainstream metadata in the first place. We need better Flatpak plugin otherwise it isn't easy to use. At least from a graphical use interface. I can argue that it's easier than Flatpak builder on the command line with workspaces and everything. But it's not easy. Then we can start moving up. We may want to move this up because we built them anyway. So it's perhaps better to have everything in the same space. Rather than having Flatpak manifest for every app and having another definition for this.

For other apps it depends. Flatpak builder is easier to use and especially if there aren't many dependencies. You can have a single file with your two or three dependencies and your app. But for complex apps is where you can have multiple dependencies and you want to modify your policy and procedure and one of your dependencies at the same time.

So this is perhaps the most difficult one. It's replacing JHBuild. One of the reasons is that Flatpak builder had to replace JHBuild to a point. This is mainly building core system components.

That's what Sam's talk yesterday was about. So in theory BuildStream is a great replacement for JHBuild. If it's built here it should build everywhere. It doesn't need the system dependencies where you can, for example, use it in the system. In practice with sandbox, BuildStream has sandbox. It can use system dependencies. So the sandbox needs to be configured for different projects. The system component would perhaps be more confined as if it is running in a separate session even though there is already a session. It could be that we would easily -- I don't know if I should discuss this very briefly, but I'd like to do this. The end result is that sandbox needs to be configured differently for different projects. We need to have the release for the requirement of the project, and we have to test it.

And the third thing is that every project needs to have all its dependencies specifically stated. We have had this problem where some apps don't have icons or don't have fonts because they don't depend on things like the default icon theme or default fonts. This is the result of being in the sandbox. But perhaps we can do something to make it easier to have all this, all of the common dependencies.

And yes, I wanted to have a live demo here. I'm not sure if it will work. But if you have any questions.

>> Let me check the Etherpad. Do you want me to read them for you?

>> Yes, you can do that.

>> Okay. Great. So first current question are current run times built with BuildStream?

>> No, the run time is the first one we used on BuildStream. KDE not yet. They have some work in progress. Probably the next version of the run time we'll use BuildStream. But the current ones don't.

>> The second question is BuildStream easy to use?

>> Yes and no. It isn't that hard to use but I'll try to give a little demo if I can share my screen.

>> The third question, where can you find the VM image?

>> I placed a link. So here is the last version. It should work. You can download it from there. You compress it. There is some progress you can have it to import it.

>> The next question. What do you need to know for each one to get started and to test it?

>> This is a good question. I think the best way is for each to maintain and try to see how it fit in the development style. What do you need as run time dependencies? And what do you need system dependencies? Not dependencies in a sense that you're linking against them but whether you want them to be inside the sandbox in the system. For example, an app you would want it to talk to things for the whole system. Like tracker, for example. You may want to have it isolated from your system. It is an open question. Can OS be used as a data OS? I say yes and no. It depends. It doesn't have security fixes. I think this is important. We tried to have them but mainly for the Flatpak. They're not always up to date and they may have security problems. So it's not really

recommended. If you're a non-developer you can use it perhaps to draw a boot, to be able to follow the development of GNOME. So at least the same day you can have it, you can see the difference. This is also important for designers. Those are the questions?

I'll try to see if I can share my screen.

>> Great. Let me know if you need any help. We have 20 more minutes. I just made you presenter again. Great. I can see your screen now. I think you're on mute.

I still can't hear you.

>> Sorry. My mic was muted.

>> I can hear you well now.

>> So what we can do is go to the chat. If you all just join on the chat and try to ask questions. But if I want to work on a -- say -- I don't know, GTK, GTK, for example. I will code bst track-deps all sdk/gtk.bst. You can see it will try to work in the GTK and all its dependencies. You can do this. This will have something that is -- how can I say it? It's tested to compile at least. This is getting the latest of everything. Then I try to build sdk/gtk.bst. You can see that everything that has been built in this --

>> Sorry.

>> So you don't have to build anything. If you're like me and have a (audio breaks) sometimes it's better to build it. One thing that is also good in BuildStream is you can choose the no-strict mode on the command line or by adding it in the configuration file like I have it. It will not necessarily -- I forgot what I -- so with the no-strict it will not rebuild everything when something changes. So it doesn't need to rebuild everything. This can be a good thing as it speeds up the compilation a lot, but it can have down sides. If something changes you will need to rebuild everything.

>> Oh, I think we lost Abderrahim. Let's wait a little bit and see if Abderrahim reconnects.

[Waiting for Abderrahim to reconnect]

>> Sorry, I got disconnected. I think I'll stop here. I'll see if there are any questions. Is there any BuildStream feature? Yes. There are as you probably know after using BuildStream 1.8 there are things that are better on BuildStream 2. Right now most of the important feature for BuildStream 2 from me personally I have a request to revert it to 1.X. that's not the most important thing to have right now.

I think that's all. Thank you everyone for listening.

>> Thank you for joining us and thank you for the talk.

[Break]