How Does an API/Provider Get Out of Labs?

As described in detail in Creating a New API or Provider, the right place for a new API/provider to start is in labs. There we can discuss all the implementation details and start testing while the API/provider is under development. Once it is in a good shape, it can be promoted to the main repo.

In general, the minimal criteria for an API/provider to be considered "in good shape" are:

1. It is being maintained
2. It has an abstraction
3. The code is up-to-date w.r.t. jclouds coding guidelines

1. It is being maintained

This usually means that we need to be able to run live tests against the provider and someone takes care of it. One prerequisite for this that jclouds has access to a free testing account that can be used.

2. It has an abstraction

This is the perhaps the most critical of the three points. We need one of the jclouds abstractions (BlobStore, ComputeService, LoadBalancer) to be implemented in order promote the provider, since jclouds is ultimately all about the abstractions and common interfaces.

That means:

- The interface itself has to be implemented, like in other providers.
- The <MyProviderAbstraction>LiveTest (e.g. AzureComputeServiceLiveTest) must be created and should extend the Base<Abstraction>LiveTest. See e.g. the DigitalOceanComputeServiceLiveTest for an example.
- For compute abstractions, the <MyProvider>TemplateBuilderLiveTest must be created and should extend the BaseTemplateBuilderLiveTest. See e.g. the HPCloudComputeTemplateBuilderLiveTest for an example.

These test classes are the "abstraction contract". Once the live tests are passing, we know the abstraction is properly implemented and working as expected.

3. The code is up-to-date

This is especially relevant if your initial implementation relies on an SDK or other library. We completely understand that this can be more convenient for in initial implementation, but for your API/Provider to be promoted, the implementation really needs use the existing jclouds HTTP layer.

"Why do I need to you the jclouds HTTP layer?"

We ask you to use the jclouds HTTP layer not just because "we do it this way:" what we've learned is that using the jclouds HTTP layer as a common base for all APIs and providers is a key part of making jclouds easier to use and promoting portability. Here are some examples of what the HTTP layer provides:

- A driver mechanism that allows users to choose their preferred HTTP driver. Currently we support the Java default HttpUrlConnection, the Apache HTTP Client, and OkHttp. The latter is Android friendly and supports PATCH verbs and other stuff that is not supported in the default JRE implementation, so we need to make sure that users still have this choice when using an API/provider.
- A generic way of handling failure at the HTTP layer, with a generic fallback mechanism that allows every provider to configure failure behavior. This fallback delegation is generic so all APIs and providers benefit from it. API and provider writers can also provide custom fallbacks if necessary.
- A generic retry policy that allows users to configure how and when failed request are retried. There are several implementations, but users can configure a more convenient one, if needed.
- A generic way to configure request timeouts "per API method". Users can configure the timeout for each call, if they need to.

There are additional points, but we hope these main ones help illustrate why new APIs and providers are required to use the jclouds HTTP layer in order to be promoted.

"Can I start out using an existing SDK or library?"

If you're planning to base your API or provider on an existing SDK or library, we can usually live with that in labs. As described above, it will need to be modified to use the jclouds HTTP layer to be promoted, though.

How best to proceed depends a bit on how much experimentation you expect to be doing in labs. If you want to get user feedback quickly, you can merge the provider into labs "as-is" and consider switching to the jclouds HTTP layer later. If you're looking to progress through labs quickly and are confident that your API or provider is pretty complete, it probably makes sense to tackle the HTTP layer straight away.

Note that we are trying to use the "labs" repo just for development, and providers should have a limited TTL there. We try to avoid partially developed provi ders from staying there forever, so please feel free to reach out to us if you have any questions! Together, we usually can come up with a a plan to get your API or provider promoted quite quickly.

Happy contributing!