Go SDK Vanity Import Path

Improved user import experience

Robert Burke <rebo@google.com>

https://s.apache.org/go-beam-vanity-import

April 2018

Proposal

Use *beam.apache.org/sdks/go* as a vanity import path prefix for Apache Beam Go SDK packages. Eg:

```
import "beam.apache.org/sdks/go/pkg/beam"
```

Instead of the current path of:

```
import "github.com/apache/beam/sdks/go/pkg/beam"
```

Summary

Go permits specifying custom vanity import paths. This offers a few benefits:

- Clear Apache Beam branding of the Go SDK packages,
    - Currently this is infixed in the github path.
    - Compare to java's import prefix: org.apache.beam.sdk.
- Decouples using the Go SDK from the code repo residing on GitHub.
    - This allows infrastructure migrations to occur without breaking users, such as what happened to Thrift's when Thrift migrated to Gitbox.
    - This includes any repo splitting we do in the future.
- Permits improved naming for subsequent backwards incompatible major versions, while maintaining the package name suffix.
    - Eg. *beam.apache.org/sdks/go/pkg/v2/beam*

Required Actions

Before the Go SDK reaches v1, and there are expectations of backwards compatibility:

Step 1 - We add a htaccess redirects to the beam.apache.org site to handle requests to subpaths at beam.apache.org/sdks/go, that return a static file with the required metadata.

Step 2 - Commit a PR for all the Go SDK code imports in the Beam repo to use the new paths. This promotes the new vanity import path by example.

Suggested Actions

After the required actions are complete, users will still be able to access the code via the old import paths, which could become confusing over time.

Step 3 - (Optional) Restrict imports to the canonical path.

This would enforce consistency between all Beam Go SDK users at the go tool level. Not strictly necessary, but convenient for avoiding confusion.

Implementation

The absolute minimal case is that all import paths, that when used as URLs return a page that contains the `go-import` meta tag in the head section.

```
<!DOCTYPE html>
<html>
<head>
<meta name="go-import" content="beam.apache.org/sdks/go” git
https://github.com/apache/beam/sdks/go">
</head>
<body>
<!-- Add a helpful page of explanatory text, and links to the go doc, the code,
the programming model, etc here. -->
</body>
</html>
```

This is sufficient for the vanity import path to work. See the Remote Import Path documentation for more details. This can be done with a single static page, that ideally sub paths of beam.apache.org/sdks/go are served. In other words, a simple htaccess redirect to a static page would be sufficient, for the minimum viable implementation. Such as:

```
# Redirect all go package path URLs to the static content page for go.
RedirectMatch 301 /sdks/go/(.*) https://beam.apache.org/sdks/go/go-import.html
[L,R=301]
```

With that added to our site’s htaccess file, as well as the appropriate static page at the described location, covers the importing case sufficiently to move to step 2.

If we wish to go a bit further, in the ideal case we also redirect to the Go doc page for the package at godoc.org (example). This can also be done server side.

The key is that the go tools use a query parameter `?go-get=1` to distinguish itself. We can redirect humans and tools separately increasing user friendliness. The complete alternative:

```
# Redirect all go tool requests to the static content page for go packages.
RewriteCond %{QUERY_STRING} go-get=1
RedirectMatch 301 /sdks/go/.* https://beam.apache.org/sdks/go/go-import.html

# For all other queries, they're probably users, redirect them to the godoc.
# Original: https://beam.apache.org/sdks/go/pkg/beam/arbitrary/subpackage
# Desired:
https://godoc.org/beam.apache.org/sdks/go/pkg/beam/arbitrary/subpackage
RedirectMatch 301 (/sdks/go/.*) https://godoc.org/beam.apache.org/$1
```

Risks And Other Questions

What about our other go tools, eg the container bootloader and the example code?
This approach covers all go code in the repo, including the container bootloader (at beam.apache.org/sdks/go/container) and examples, such as wordcount (at beam.apache.org/sdks/go/examples/wordcount)

How does a Go SDK developer set up their $GOPATH properly with this proposal?
A Go SDK developer would need to set up a *beam.apache.org* folder in their $GOPATH, and git clone.

```
$ cd $GOPATH
$ mkdir beam.apache.org
$ git clone https://github.com/apache/beam.git beam.apache.org
```

This permits development for all languages from the cloned repo, and permits the go tools to find the development copies of the go code where it expects them to be.

Why keep the "sdks/go/pkg" segments?
This is partly answered by the Use a different URL alternative, but the reasons are:

- Git is opinionated about how repos are cloned, and trying to put different sub packages into different roots would be difficult.
- Go is opinionated about where to find packages.
- The proposal is one that satisfies both tools, while also not splitting where folks can find/get code in the repository.
    - Eg. Today, to run a Go SDK wordcount example one would need to *go get github.com/apache/beam/sdks/go/examples/wordcount* then *go run github.com/apache/beam/sdks/go/examples/wordcount*
    - This proposal makes that slightly better *go get beam.apache.org/sdks/go/examples/wordcount* then *go run beam.apache.org/sdks/go/examples/wordcount*
- Removing them would make developing examples difficult since one wouldn't be able to depend on code in progress for the example.
    - The naive approach would end up accidentally depending on the code at head, rather than the code in the repo.

What if the repo splits?
In the event that the Beam repo splits into its per-language components,

we can clone the repo's contents to its new location w/o the sdks/go segment, and update go-import meta tag. Eg if the go repo is *apache/beam-go* the tag becomes

```
<meta name="go-import" content="beam.apache.org/sdks/go” git
https://github.com/apache/beam-go">
```

And users remain unbroken without needing to update their paths. A similar transform would be require for cloning the repo for development, to maintain the developer experience as well.

Won't this conflict with the documentation moving to beam.apache.org/sdks/<language>/?

In terms of breaking things:

- Rendered content in the existing documentation folder
    - Doesn't currently conflict since sdks/go/index.html doesn't presently exist since no Getting Started page for Go, yet.
    - This can be worked around when it does.
    - Other language specific docs are in their own dedicated <language>-<topic> paths, which the redirect doesn't apply to.
- Go imports won't break, since we can specifically redirect requests from the go tool which always have the query parameter ?go-get=1
    - We don't need to redirect to a static index.html, it can be anything. (The proposal has been changed to redirect to a `go-import.html` instead.)
- The existing documentation/sdks/<language> structure seems to largely redirect to the language docs for the SDK version.
    - Go (the language) doesn't have a settled versioning story, so we don't yet need versioned docs.
    - The go packages are rooted at <repo>/sdks/go but real packages are all subdirectories of that, so we can change the godoc.org redirect to only be from under strict subdirectories of go.

Alternatives

Do Nothing

We change nothing, users can and will be able to import and use our packages as normal until a repo move or split occurs, at which point what happened to Thrift happens to users of the Beam Go SDK.

Use a different URL

“beam.apache.com/beam”

This is shorter, but becomes messier if another language has a similar mechanism for changing import paths.

It also confuses the versioning issue for subsequent major versions:

- “beam.apache.com/beam/v3”
    - Complicates the import name used in the code, requiring all users to locally rename.
- “beam.apache.com/v3/beam”
    - Better than above, but requires the CGI script to be additionally rooted at /v3/ and other versions.

Using the proposed path avoids both of these issues, by isolating the changes to a /sdks/go/ subpath on the site, and sets the pattern for other languages.

“beam.apache.com/go/beam”
This is shorter, than the proposal, and avoids the other issue, but has issues with the story for developers on the repo. It and the previous alternative doesn’t address how a dev gets started, or breaks it up, complicating development on examples and tools, which are missed by these alternatives.

The proposal (using “beam.apache.com/sdks/go”) permits an similar workflow to today, where a Go SDK developer sets up a directory in their $GOPATH, and git clones the repo to it. We could work around the split with symlinks, but symlinks complicate the set up and are inconvenient to debug.

Appendix

Background

Importing code in Go
Go packages are imported via an import path, which tells the go tools where to look for the code.

For example, at present to import the beam package and a few subpackages, a user writes:

```
import  (
        "github.com/apache/beam/sdks/go/pkg/beam"
        "github.com/apache/beam/sdks/go/pkg/beam/io/textio"
        "github.com/apache/beam/sdks/go/pkg/beam/transforms/stats"
        "github.com/apache/beam/sdks/go/pkg/beam/x/beamx"
)
```

The path specifies the location the code can be found, which is roughly URL path the code can be downloaded from. Certain exceptions and variances exist for the standard library, and common repository maintainers, like GitHub.

Vanity Import Paths

Go permits authors to not be beholden to where the code is stored for where the code lives. These are called vanity import paths.

As described in the implementation section, all that's required for the tools to be able to find the code is a set of meta tags at the URL of the path.

```
import  (
        "beam.apache.org/sdks/go/pkg/beam"
        "beam.apache.org/sdks/go/pkg/beam/io/textio"
        "beam.apache.org/sdks/go/pkg/beam/transforms/stats"
        "beam.apache.org/sdks/go/pkg/beam/x/beamx"
)
```

It effectively adds indirection, and allows where the code is stored to be an implementation detail.

The result on the user side is that when the `go get` the package with the new path, it will store the code under `$GOPATH/beam.apache.org/go/beam` rather than `$GOPATH/github.com/apache/beam/sdks/go/pkg/beam` as it does currently.

Sub packages of a vanity path point to the same root location, and the go tool handles the rest.

Canonical Import Paths

Canonical paths are a separate step after adding a vanity import path. It permits package authors to declare where the code lives canonically. This would restrict code to being imported to the canonical location. This is managed by the go tools and source code with what is called an *import comment*, after the package declaration.

```
package beam // import "beam.apache.org/sdks/go/pkg/beam"
```

The restriction doesn't apply to "vendor" directories which is how Go currently handles versioning.

Documents

Remote Import Paths
Import Path Checking
https://blog.bramp.net/post/2017/10/02/vanity-go-import-paths/

Issue to be aware of:
http://grokbase.com/t/gg/golang-nuts/14cg93ag4p/go-nuts-vanity-import-paths-and-subpackages