Standard Module Structure

The standard module structure is a file and directory layout we recommend for reusable modules distributed in separate repositories. Terraform tooling is built to understand the standard module structure and use that structure to generate documentation, index modules for the module registry, and more.

The standard module structure expects the layout documented below. The list may appear long, but everything is optional except for the root module. Most modules don't need to do any extra work to follow the standard structure.

- **Root module**. This is the **only required element** for the standard module structure. Terraform files must exist in the root directory of the repository. This should be the primary entrypoint for the module and is expected to be opinionated. For the Consul module the root module sets up a complete Consul cluster. It makes a lot of assumptions however, and we expect that advanced users will use specific *nested modules* to more carefully control what they want.

- **README**. The root module and any nested modules should have README files. This file should be named `README` or `README.md`. The latter will be treated as markdown. There should be a description of the module and what it should be used for. If you want to include an example for how this module can be used in combination with other resources, put it in an examples directory like this. Consider including a visual diagram depicting the infrastructure resources the module may create and their relationship.

  The README doesn't need to document inputs or outputs of the module because tooling will automatically generate this. If you are linking to a file or embedding an image contained in the repository itself, use a commit-specific absolute URL so the link won't point to the wrong version of a resource in the future.

- **LICENSE**. The license under which this module is available. If you are publishing a module publicly, many organizations will not adopt a module unless a clear license is present. We recommend always having a license file, even if it is not an open source license.

- **`main.tf`**, **`variables.tf`**, **`outputs.tf`**. These are the recommended filenames for a minimal module, even if they're empty. `main.tf` should be the primary entrypoint. For a simple module, this may be where all the resources are created. For a complex module, resource creation may be split into multiple files but any nested module calls should be in the main file. `variables.tf` and `outputs.tf` should contain the declarations for variables and outputs, respectively.

- **Variables and outputs should have descriptions.** All variables and outputs should have one or two sentence descriptions that explain their purpose. This is used for documentation. See the documentation for variable configuration and output configuration for more details.

- **Nested modules**. Nested modules should exist under the `modules/` subdirectory. Any nested module with a `README.md` is considered usable by an external user. If a README doesn't exist, it is considered for internal use only. These are purely advisory; Terraform will not actively deny usage of internal modules. Nested modules should be used to split complex behavior into multiple small modules that advanced users can carefully pick and choose. For example, the Consul module has a nested module for creating the Cluster that is separate from the module to setup necessary IAM policies. This allows a user to bring in their own IAM policy choices.

  If the root module includes calls to nested modules, they should use relative paths like `./modules/consul-cluster` so that Terraform will consider them to be part of the same repository or package, rather than downloading them again separately.

  If a repository or package contains multiple nested modules, they should ideally be composable by the caller, rather than calling directly to each other and creating a deeply-nested tree of modules.

- **Examples**. Examples of using the module should exist under the `examples/` subdirectory at the root of the repository. Each example may have a README to explain the goal and usage of the example. Examples for submodules should also be placed in the root `examples/` directory.

  Because examples will often be copied into other repositories for customization, any `module` blocks should have their `source` set to the address an external caller would use, not to a relative path.

A minimal recommended module following the standard structure is shown below. While the root module is the only required element, we recommend the structure below as the minimum:

```
$ tree minimal-module/
.
├── README.md
├── main.tf
├── variables.tf
├── outputs.tf
```

A complete example of a module following the standard structure is shown below. This example includes all optional elements and is therefore the most complex a module can become:

```
$ tree complete-module/
.
├── README.md
├── main.tf
├── variables.tf
├── outputs.tf
├── ...
├── modules/
│   ├── nestedA/
│   │   ├── README.md
│   │   ├── variables.tf
│   │   ├── main.tf
│   │   ├── outputs.tf
│   ├── nestedB/
│   ├── .../
├── examples/
│   ├── exampleA/
│   │   ├── main.tf
│   ├── exampleB/
│   ├── .../
```