You can use this special syntax solely in `postcondition`, `connection`, and `provisioner` blocks (you'll see examples of the latter two later in this chapter) to refer to an output `ATTRIBUTE` of the surrounding resource. If you tried to use the standard `aws_autoscaling_group.example.<ATTRIBUTE>` syntax, you'd get a circular dependency error, as resources can't have references to themselves, so the self expression is a workaround added specifically for this sort of use case.

If you run `apply` on this module, Terraform will deploy the module, but after, if it turns out that the subnets the user passed in via the `subnet_ids` input variable were all in the same AZ, the `postcondition` block will show an error. This way, you'll always be warned if your ASG isn't configured for high availability.

When to use validations, preconditions, and postconditions

As you can see, `validation`, `precondition`, and `postcondition` blocks are all similar, so when should you use each one?

Use `validation` blocks for basic input sanitization

> Use `validation` blocks in all of your production-grade modules to prevent users from passing invalid variables into your modules. The goal is to catch basic input errors *before* any changes have been deployed. Although `precondition` blocks are more powerful, you should still use `validation` blocks for checking variables whenever possible, as `validation` blocks are defined with the variables they validate, which leads to a more readable and maintainable API.

Use `precondition` blocks for checking basic assumptions

> Use `precondition` blocks in all of your production-grade modules to check assumptions that must be true *before* any changes have been deployed. This includes any checks on variables you can't do with `validation` blocks (such as checks that reference multiple variables or data sources) as well as checks on resources and data sources. The