Suggestions for Standard Resource Structure

What Copilot does:
When you start writing a `resource`, `variable`, `provider`, or `output` block, Copilot detects the pattern and automatically suggests a complete structure based on best practices or official documentation.

Benefits:

- Saves time writing resources from scratch
- Reduces syntax errors
- Acts as a learning aid for beginners

Example:

If you write:

```
resource "aws_instance" "web" {
```

Copilot might suggest:

```
  ami           = "ami-0abcdef1234567890"
  instance_type = "t2.micro"
  tags = {
    Name = "WebServer"
  }
}
```

More Descriptive and Consistent Variable Names

What Copilot does:
It recommends variable, resource, and output names based on project context and naming best practices. This improves readability and maintainability.

Benefits:

- Promotes team-wide consistency
- Avoids generic or confusing names
- Improves documentation of code intent

Example:

Instead of writing:

```
variable "var1" {
  type = string
}
```

Copilot may suggest:

```
variable "vpc_cidr_block" {
  description = "CIDR block for the VPC"
  type        = string
}
```

Quick Generation of Reusable Module Blocks

What Copilot does:
When you start writing a `module` block, Copilot suggests commonly used parameters. For well-known modules (like those from the Terraform Registry), Copilot often “knows” the expected structure.

Benefits:

- Speeds up integration of complex modules
- Minimizes mistakes with incorrect inputs
- Encourages reusable infrastructure-as-code

Example:

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  name   = "my-vpc"
  cidr   = "10.0.0.0/16"
  azs    = ["us-west-1a", "us-west-1b"]
  ...
}
```

Copilot can auto-complete this based on the module source.

Automatic Generation of Helpful Comments

What Copilot does:
It detects your code and suggests natural language comments explaining what the block does.

Benefits:

- Enhances in-code documentation
- Helps collaborators understand and maintain infrastructure
- Great for multi-team or onboarding scenarios

Example:

```
resource "aws_security_group" "web_sg" {
  name        = "web-sg"
  description = "Allow web traffic"
}
```

These comments can be automatically suggested by Copilot.

Simplifying Repetitive Patterns and Refactoring Suggestions

What Copilot does:
It detects code repetition and may suggest more efficient ways to handle it, such as using `locals`, `count`, `for_each`, or extracting values into variables.

Benefits:

- Reduces code duplication
- Encourages clean architecture
- Improves scalability and reusability

Before Example:

```
tags = {
  Environment = "dev"
  Project     = "myapp"
}
```

Copilot may suggest defining this in `locals`:

```
locals {
  common_tags = {
    Environment = "dev"
    Project     = "myapp"
  }
}
```

And then reusing it:

```
tags = local.common_tags
```