it should not tightly coupled, make it flexible,

also what you created is best but increase the complexity make it bit simpler to understand and use

refine and reprint

Based on my analysis of your Terraform modules repository, here's a refined, simplified prompt for Cursor that focuses on flexibility and ease of use:

Terraform Module Optimization Prompt - Simple & Flexible Approach

Goal

Restructure these Terraform modules to be **loosely coupled**, **self-contained**, and **dead simple** to use while maintaining maximum flexibility for different infrastructure scenarios.

Core Principles

Loose Coupling Philosophy

Each module should work **independently** without assuming other modules exist. Pass only what's needed via simple variables.

Before (Tightly Coupled):

```
# Module assumes specific VNet module structure
variable "spoke_vnet_id" {}
variable "spoke_subnet_id" {}
variable "acr_id" {}
variable "kv_id" {}
```

After (Loosely Coupled):

```
# Module accepts any valid Azure resource ID
variable "subnet_id" {
  description = "Any valid subnet ID - from module output or existing resource"
  type        = string
}

variable "resource_dependencies" {
  description = "Optional IDs of resources that need access (ACR, Key Vault, etc.)"
  type        = map(string)
  default     = {}
}
```

Simplicity First

Single Responsibility - Each module does ONE thing well:

- VNet module → Creates networks and subnets, nothing else
- AKS module → Creates Kubernetes cluster, delegates networking to caller
- Private Endpoint module → Creates endpoints for ANY Azure service

Minimal Required Variables - Only make variables required if absolutely necessary:

```
# REQUIRED - Only 3-5 core parameters
variable "name" { type = string }
variable "resource_group_name" { type = string }
variable "location" { type = string }

# OPTIONAL - Everything else has smart defaults
variable "sku" {
  type    = string
  default = "Standard"
}

variable "enable_private_endpoint" {
  type    = bool
  default = false
}
```

Progressive Disclosure - Simple configs work out of the box, advanced options available when needed:

```
# Simple usage - just works
module "storage" {
  source = "./modules/Storage-Accounts"
  name   = "mystorageacct"
  resource_group_name = "my-rg"
  location = "eastus"
}

# Advanced usage - granular control when needed
module "storage_advanced" {
  source = "./modules/Storage-Accounts"
  name   = "mystorageacct"
  resource_group_name = "my-rg"
  location = "eastus"

  network_rules = {
    default_action = "Deny"
    ip_rules       = ["203.0.113.0/24"]
    bypass         = ["AzureServices"]
  }

  blob_properties = {
    versioning_enabled = true
    retention_days     = 30
  }
}
```

Specific Module Improvements

VNet Module - Keep It Simple

```
# Simple interface - no complex objects unless needed
variable "address_space" {
  type = list(string)
}

variable "subnets" {
  description = "Map of subnets - key becomes subnet name"
  type = map(object({
    address_prefix = string
    service_endpoints = optional(list(string), [])
    delegation        = optional(string, null)
  }))
}

# Output everything downstream modules need
output "vnet_id" { value = azurerm_virtual_network.this.id }
output "vnet_name" { value = azurerm_virtual_network.this.name }
output "subnet_ids" {
  value = { for k, v in azurerm_subnet.subnets : k => v.id }
}
```

AKS Module - Flexible & Independent

```
# Don't assume what resources exist - accept IDs
variable "subnet_id" {
  description = "Subnet for AKS nodes"
  type        = string
}

# Make integrations optional, not mandatory
variable "key_vault_integration" {
  description = "Enable Key Vault CSI driver"
  type = object({
    enabled = bool
    key_vault_id = optional(string)
  })
  default = { enabled = false }
}

variable "acr_integration" {
  description = "Grant AKS access to container registries"
  type = object({
    enabled       = bool
    registry_ids  = optional(list(string), [])
  })
  default = { enabled = false }
}

# Simplify node pool configuration
variable "system_node_pool" {
  type = object({
    vm_size    = optional(string, "Standard_D2s_v3")
    node_count = optional(number, 2)
    auto_scale = optional(bool, false)
    min_nodes  = optional(number, 2)
    max_nodes  = optional(number, 5)
  })
  default = {}
}

variable "user_node_pools" {
  description = "Additional node pools - only create if needed"
  type = map(object({
    vm_size    = string
    node_count = optional(number, 1)
    auto_scale = optional(bool, false)
    min_nodes  = optional(number, 1)
    max_nodes  = optional(number, 10)
  }))
  default = {}
}
```

Storage Account - Smart Defaults

```
# Minimal required input
variable "name" { type = string }
variable "resource_group_name" { type = string }
variable "location" { type = string }

# Sensible defaults for common scenarios
variable "account_tier" {
  type    = string
  default = "Standard"
}

variable "replication_type" {
  type    = string
  default = "LRS"
}

variable "enable_https_only" {
  type    = bool
  default = true
}
```

```
# Advanced options - null by default means "don't configure"
variable "network_rules" {
  type = object({
    default_action = string
    ip_rules       = optional(list(string), [])
    subnet_ids     = optional(list(string), [])
  })
  default = null
}

# Use conditional creation
resource "azurerm_storage_account_network_rules" "this" {
  count = var.network_rules != null ? 1 : 0

  storage_account_id = azurerm_storage_account.this.id
  default_action     = var.network_rules.default_action
  ip_rules           = var.network_rules.ip_rules
  virtual_network_subnet_ids = var.network_rules.subnet_ids
}
```

Private Endpoint - Universal Connector

```
# Works with ANY Azure resource - not tied to specific services
variable "name" { type = string }
variable "resource_group_name" { type = string }
variable "location" { type = string }
variable "subnet_id" { type = string }

variable "private_connection_resource_id" {
  description = "ID of ANY Azure resource (Storage, SQL, AKS, etc.)"
  type        = string
}

variable "subresource_names" {
  description = "e.g., ['blob'] for storage, ['sqlServer'] for SQL"
  type        = list(string)
}

variable "private_dns_zone_ids" {
  description = "Optional - provide if using private DNS"
  type        = list(string)
  default     = []
}

# Simple, reusable for any service
resource "azurerm_private_endpoint" "this" {
  name                = var.name
  location            = var.location
  resource_group_name = var.resource_group_name
  subnet_id           = var.subnet_id

  private_service_connection {
    name                           = "${var.name}-connection"
    private_connection_resource_id = var.private_connection_resource_id
    subresource_names              = var.subresource_names
    is_manual_connection           = false
  }

  dynamic "private_dns_zone_group" {
    for_each = length(var.private_dns_zone_ids) > 0 ? [1] : []
    content {
      name                 = "${var.name}-dns-group"
      private_dns_zone_ids = var.private_dns_zone_ids
    }
  }
}
```

PostgreSQL Module - Configuration Patterns

```
# Core required variables
variable "name" { type = string }
variable "resource_group_name" { type = string }
variable "location" { type = string }

# Use presets for common scenarios
variable "environment_preset" {
  description = "Quick configs: dev, staging, prod"
  type        = string
  default     = "dev"
}

locals {
  preset_configs = {
    dev = {
      sku_name     = "B_Standard_B1ms"
      storage_mb   = 32768
      ha_enabled   = false
      backup_retention = 7
    }
    staging = {
      sku_name     = "GP_Standard_D2s_v3"
      storage_mb   = 65536
      ha_enabled   = false
      backup_retention = 14
    }
    prod = {
      sku_name     = "GP_Standard_D4s_v3"
      storage_mb   = 262144
      ha_enabled   = true
      backup_retention = 35
    }
  }

  config = merge(local.preset_configs[var.environment_preset], var.custom_config)
}

# Allow overrides for custom scenarios
variable "custom_config" {
  type = object({
    sku_name          = optional(string)
    storage_mb        = optional(number)
    ha_enabled        = optional(bool)
    backup_retention  = optional(number)
  })
  default = {}
}
```

Standardization Rules

Every Module Must Have

1. README.md with 2-3 simple examples (basic, intermediate, advanced)
2. versions.tf with clear provider requirements
3. outputs.tf with ALL resource attributes others might need
4. variables.tf organized: required first, then optional with defaults

Variable Naming Conventions

- Resource identifiers from other modules: `<resource>_id` (e.g., `subnet_id`, `vnet_id`)
- Feature toggles: `enable_<feature>` (e.g., `enable_backup`, `enable_monitoring`)
- Configuration blocks: `<feature>_config` (e.g., `network_config`, `backup_config`)
- Lists of items: `<item>s` (e.g., `subnets`, `node_pools`, `firewall_rules`)

Output Naming Conventions

- Resource ID: `<resource>_id`
- Resource name: `<resource>_name`

```
- Collections: `<resource>_ids` (as map), `<resource>_list` (as list)
```

- Connection info: `<resource>_connection_string` (mark sensitive)

Remove Tight Coupling

Current Issues to Fix:

- ❌ AKS module creating Private DNS zones (separate this)
- ❌ AKS module managing role assignments internally (make optional, return principal IDs)
- ❌ Modules assuming specific variable structures from other modules
- ❌ Hardcoded feature enablement (Key Vault CSI, Azure Policy)

Decoupling Strategy:

```
# Instead of module creating role assignments internally
# Export identity information and let caller decide
```

```
output "cluster_identity" {
  value = {
    principal_id = azurerm_kubernetes_cluster.this.identity[0].principal_id
    client_id    = azurerm_kubernetes_cluster.this.kubelet_identity[0].client_id
    object_id    = azurerm_kubernetes_cluster.this.kubelet_identity[0].object_id
  }
}

# Caller decides what permissions to grant
resource "azurerm_role_assignment" "aks_to_acr" {
  scope                = var.acr_id
  role_definition_name = "AcrPull"
  principal_id         = module.aks.cluster_identity.principal_id
}
```

Testing Pattern - Simple Examples

Create `examples/` folder:

```
examples/
├── minimal/
│   └── main.tf          # Absolute minimum config
├── standard/
│   └── main.tf          # Recommended production config
└── complete/
    └── main.tf          # All features enabled
```

Each example should run with just `terraform apply` - no external dependencies.

Success Criteria

After refactoring, a user should be able to:

1. **Deploy a module in 5 minutes** by copying the minimal example
2. **Understand all options in 10 minutes** by reading the README
3. **Mix and match modules freely** without fighting dependencies
4. **Upgrade one module** without breaking others
5. **Use modules with existing infrastructure** (not just new deployments)

This approach creates modules that are **Lego blocks** - simple pieces that combine in infinite ways, not a rigid framework that dictates architecture.