

Got it! Packaging this into a structure that's ready for direct import into an Azure DevOps Git repository, resembling a standard multi-module Terraform project.

I'll provide:

1. The complete folder structure.
2. All .tf files within their respective modules and the root.
3. Example .tfvars files.
4. The azure-pipelines.yml file.
5. A README.md for setup instructions.

This structure allows you to directly clone/import this into your Azure DevOps repository.

Project Structure (Ready for Azure DevOps Repo)

```
.
├── .gitignore
├── README.md
├── main.tf
├── variables.tf
├── versions.tf
├── azure-pipelines.yml
├── environments/
│   ├── dev.tfvars
│   ├── prod.tfvars
│   └── test.tfvars
└── modules/
    ├── aihub/
    │   ├── main.tf
    │   ├── variables.tf
    │   └── outputs.tf
    ├── apim/
    │   ├── main.tf
    │   ├── variables.tf
    │   └── outputs.tf
    ├── cost_management/
    │   ├── main.tf
    │   ├── variables.tf
    │   └── outputs.tf
    └── monitoring/
        ├── main.tf
        ├── variables.tf
        └── outputs.tf
```

File Contents

.gitignore

```
# Terraform
```

```
.terraform/
*.tfplan
*.tfstate
*.tfstate.backup
.terraform.lock.hcl
terraform.tfstate.d/

# Azure Pipelines
azure-pipelines.yml
```

README.md

Azure AI Hub Landing Zone Accelerator

This repository contains Terraform Infrastructure as Code (IaC) to deploy a production-ready Azure AI Hub Landing Zone, integrated with API Management, comprehensive monitoring, and cost management solutions. It's designed for multi-environment deployments (Dev, Test, Prod) using Azure Pipelines.

Table of Contents

1. [Overview] (#overview)
2. [Prerequisites] (#prerequisites)
3. [Repository Structure] (#repository-structure)
4. [Configuration] (#configuration)
 - * [Terraform Variables] (#terraform-variables)
 - * [Environment-Specific Configuration] (#environment-specific-configuration)
5. [Azure DevOps Pipeline Setup] (#azure-devops-pipeline-setup)
 - * [Service Connection] (#service-connection)
 - * [Terraform State Backend] (#terraform-state-backend)
 - * [Variable Group] (#variable-group)
 - * [Environments (Azure DevOps)] (#environments-azure-devops)
6. [Deployment Steps] (#deployment-steps)
7. [Self-Service for Application Teams (Separate)] (#self-service-for-application-teams-separate)
8. [Cost Management & Monitoring] (#cost-management--monitoring)
9. [Customizing API Management Policies] (#customizing-api-management-policies)
10. [Troubleshooting] (#troubleshooting)

1. Overview

This IaC solution provisions the following Azure resources:

- * **Azure AI Hub:** The central hub for AI model management.

- * **Azure API Management (APIM):** To expose AI models via a secure, managed API gateway with custom policies, products, and VNet integration.
- * **Monitoring Stack:**
 - * Azure Log Analytics Workspace
 - * Azure Application Insights
 - * Azure Event Hub (for logging/telemetry)
 - * Azure Function App (to process events, e.g., push to Power BI)
- * **Cost Management:** Azure Budgets to monitor spending.
- * **Networking Integration:** Leverages existing Virtual Networks and Subnets for secure, private connectivity (APIM, AI Hub Private Endpoints).

Key features include:

- * **Modular Design:** Reusable Terraform modules for each component.
- * **Environment Flexibility:** Easily deploy to different environments (Dev, Test, Prod) using `tfvars`.
- * **Conditional Provisioning:** Ability to enable/disable certain resources (e.g., APIM, monitoring).
- * **Existing Resource Injection:** Option to use existing Log Analytics Workspaces or Application Insights instances.
- * **Azure DevOps Pipelines:** Automated CI/CD for Infrastructure deployment.

2. Prerequisites

- * An Azure Subscription.
- * An Azure DevOps Organization and Project.
- * Permissions to create/manage Azure resources (Contributor/Owner role on target subscription).
- * Permissions to create/manage Azure DevOps pipelines, service connections, and variable groups.
- * Existing Azure Virtual Network (VNet) and at least two subnets (one for APIM, one for Private Endpoints for AI Hub/other services).
 - * The APIM subnet must have enough available IP addresses and meet APIM VNet integration requirements (e.g., no delegated subnets if using older APIM VNet types, though newer V2 might support delegation).
 - * The Private Endpoint subnet should be dedicated for Private Endpoints and not contain other resources.
- * (Optional but Recommended) An Azure Storage Account and Container for Terraform remote state.

3. Repository Structure

. |— .gitignore # Git ignore file |— README.md # This README file |— main.tf # Root

Terraform configuration |— variables.tf # Root level variables |— versions.tf # Terraform version constraints and Azure provider |— azure-pipelines.yml # Azure DevOps Pipeline definition |— environments/ # Environment-specific variable files | |— dev.tfvars | |— prod.tfvars | |— test.tfvars |— modules/ # Reusable Terraform modules |— aihub/ # Azure AI Hub module | |— main.tf | |— variables.tf | |— outputs.tf |— apim/ # Azure API Management module | |— main.tf | |— variables.tf | |— outputs.tf |— cost_management/ # Azure Cost Management module | |— main.tf | |— variables.tf | |— outputs.tf |— monitoring/ # Azure Monitoring module |— main.tf |— variables.tf |— outputs.tf

4. Configuration

Terraform Variables (`variables.tf`)

The `variables.tf` file at the root level defines global variables that can be overridden by environment-specific `.tfvars` files. Key variables to review and potentially customize:

- * `environment`: (dev, test, prod)
- * `location`: Azure region
- * `resource_group_name_prefix`: Prefix for resource group names.
- * `tags`: Global tags applied to all resources.
- * `existing_vnet_name`, `existing_vnet_resource_group_name`: Details of your pre-existing VNet.
- * `existing_apim_subnet_name`, `existing_aihub_subnet_name`: Names of subnets within your VNet.
- * **Feature Flags:**
 - * `enable_apim`: Set to `false` to skip APIM deployment.
 - * `enable_cost_management`: Set to `false` to skip cost management setup.
 - * `enable_monitoring`: Set to `false` to skip monitoring components.
- * **Existing Resource Injection:**
 - * `use_existing_log_analytics_workspace`: Set to `true` to use an existing LAW.
 - * `existing_log_analytics_workspace_id`: Provide the Resource ID if `use_existing_log_analytics_workspace` is `true`.
 - * `use_existing_app_insights`: Set to `true` to use an existing App Insights.
 - * `existing_app_insights_id`: Provide the Resource ID if `use_existing_app_insights` is `true`.
- * `apim_sku_name`: Choose appropriate SKU (e.g., `Developer_1`, `Standard_V2`, `Premium`).
- * `apim_publisher_email`, `apim_publisher_name`: Contact details for APIM.
- * `aihub_name`: Name for your Azure AI Hub.
- * `cost_management_budget_amount`,
`cost_management_budget_time_grain`,
`cost_management_budget_notification_emails`: Budget details.

```
* `event_hub_namespace_sku`, `event_hub_name`, `function_app_name`,  
`function_app_runtime`: Monitoring component details.
```

```
### Environment-Specific Configuration (`environments/*.tfvars`)
```

Each `.tfvars` file in the `environments/` directory (e.g., `dev.tfvars`, `prod.tfvars`) overrides the default values in `variables.tf` and provides environment-specific settings. ****Customize these files**** to match your environment details, existing network setup, and desired resource configurations.

```
**Example `dev.tfvars`:**
```

```
````terraform  
environment = "dev"
location = "East US"
apim_sku_name = "Developer_1"
apim_publisher_email = "dev-admin@example.com"
apim_publisher_name = "Dev Team"
aihub_name = "dev-aihub"
cost_management_budget_amount = 200
cost_management_budget_notification_emails =
["devops-alerts@example.com"]
function_app_name = "dev-aihub-func"
existing_vnet_name = "dev-vnet"
existing_vnet_resource_group_name = "dev-network-rg"
existing_apim_subnet_name = "apim-subnet"
existing_aihub_subnet_name = "private-endpoint-subnet"
tags = {
 Environment = "Dev"
 Project = "AIHub"
}
use_existing_log_analytics_workspace = false # Example: Provision new
in dev
enable_cost_management = true
enable_monitoring = true
enable_apim = true
```

#### **Example prod.tfvars:**

```
environment = "prod"
location = "West Europe"
apim_sku_name = "Premium" # Use Premium for
production
apim_publisher_email = "prod-admin@example.com"
apim_publisher_name = "Operations Team"
aihub_name = "prod-aihub"
cost_management_budget_amount = 5000
cost_management_budget_notification_emails =
["prod-alerts@example.com", "finance-team@example.com"]
```

```

function_app_name = "prod-aihub-func"
existing_vnet_name = "prod-vnet"
existing_vnet_resource_group_name = "prod-network-rg"
existing_apim_subnet_name = "apim-subnet"
existing_aihub_subnet_name = "private-endpoint-subnet"
tags = {
 Environment = "Prod"
 Project = "AIHub"
 Criticality = "High"
}
use_existing_log_analytics_workspace = true # Example: Use existing in
prod
existing_log_analytics_workspace_id =
"/subscriptions/YOUR_PROD_SUB_ID/resourceGroups/PROD_CORE_MONITORING_R
G/providers/Microsoft.OperationalInsights/workspaces/prod-central-log-
analytics"
use_existing_app_insights = true
existing_app_insights_id =
"/subscriptions/YOUR_PROD_SUB_ID/resourceGroups/PROD_CORE_MONITORING_R
G/providers/Microsoft.Insights/components/prod-central-appinsights"
enable_cost_management = true
enable_monitoring = true
enable_apim = true

```

## 5. Azure DevOps Pipeline Setup

### Service Connection

1. In your Azure DevOps project, navigate to **Project settings** -> **Service connections**.
2. Create a new **Azure Resource Manager** service connection.
3. Choose **Service principal (automatic)** or **Service principal (manual)**.
  - **Automatic:** Simplest. Azure DevOps creates the SPN.
  - **Manual:** You create the SPN in Azure Portal/CLI and provide details. This gives you more control.
4. Grant the Service Principal **Contributor** role (or higher for initial setup like role assignments) on the Azure subscription(s) where you plan to deploy the AI Hub and its resources. For production, consider a more granular custom role after initial setup.

### Terraform State Backend

**Crucial for collaborative and pipeline-driven deployments.**

1. **Manually create an Azure Storage Account and Container:**
  - Create a dedicated Resource Group (e.g., tfstate-rg).
  - Create a Storage Account within that RG (e.g., tfstateaccount0123). Ensure the name is globally unique.
  - Create a Container within the Storage Account (e.g., tfstate).

2. **Add backend.tf:** Create a file named backend.tf at the root of your Terraform project (alongside main.tf, variables.tf, etc.).

```
backend.tf
terraform {
 backend "azurerm" {
 resource_group_name = "tfstate-rg" # Replace with
your TF state RG name
 storage_account_name = "tfstateaccount0123" # Replace with
your TF state SA name
 container_name = "tfstate" # Replace with
your TF state container name
 key = "aihub-accelerator.tfstate" # Base key
for the state file
 }
}
```

**Important:** The key will be dynamically extended by the pipeline (key=\$(TFSTATE\_KEY)-dev.tfstate).

## Variable Group

1. In your Azure DevOps project, navigate to **Pipelines -> Library**.
2. Create a new Variable Group (e.g., TerraformBackendVariables).
3. Add the following variables:
  - AZURE\_SERVICE\_CONNECTION: The **name** of the Azure Service Connection you created (e.g., AzureRM-AIHub).
  - TFSTATE\_RG: The name of the resource group holding your Terraform state storage account (e.g., tfstate-rg).
  - TFSTATE\_SA: The name of your Terraform state storage account (e.g., tfstateaccount0123).
  - TFSTATE\_CONTAINER: The name of the container in your storage account (e.g., tfstate).
  - TFSTATE\_KEY: A base key for your state files (e.g., aihub-accelerator).
4. Link this variable group to your pipeline. The azure-pipelines.yml already includes variables: - group: TerraformBackendVariables.

## Environments (Azure DevOps)

1. In your Azure DevOps project, navigate to **Pipelines -> Environments**.
2. Create new environments named Dev and Prod (and Test if you have one).
3. For Prod (and potentially Test), add **Approvals and checks** to ensure human approval before deployment. This is crucial for production deployments.

## 6. Deployment Steps

1. **Clone this repository** into your Azure DevOps Git repository.
2. **Customize:**
  - Update environments/\*.tfvars with your specific Azure details, VNet/subnet names,

- and desired resource configurations.
  - (Optional) Modify the modules/ to add more resources or refine existing ones as per your specific needs (e.g., Private Link configurations for AI Hub, advanced APIM policies).
  - Ensure the backend.tf file (created locally, then committed) points to your actual Azure Storage Account for state.
- 3. **Setup Azure DevOps:** Follow the steps in section 5 for Service Connection, Terraform State Backend, and Variable Group.
- 4. **Run Pipeline:** The azure-pipelines.yml is configured to trigger on changes to the terraform directory in main branch or PRs. You can also manually queue a build.
  - The pipeline will first perform terraform plan for Dev and Prod, generating plan artifacts.
  - Upon successful planning, it proceeds to the apply stage, which will wait for approvals for Prod environment.
  - Review the plan and approve the deployment to proceed.

## 7. Self-Service for Application Teams (Separate)

As requested, the self-service onboarding for application teams is *separate* from this core infrastructure IaC. It would typically involve:

- **Process:** An application team requests access to the AI Hub/models via a portal or ticketing system.
- **Automation:** An Azure Function, Logic App, or another specialized pipeline would:
  - Create an Azure API Management User.
  - Create an Azure API Management Subscription, linking the user to a specific APIM Product (e.g., "OpenAI Access Product").
  - Generate and securely provide the subscription keys to the application team.
  - Implement an approval workflow (e.g., Power Automate, Azure DevOps Approvals).
- **Terraform for Onboarding (Optional):** You could have a *separate, lighter Terraform project* specifically for managing APIM users and subscriptions. This project would be invoked by the self-service automation.

## 8. Cost Management & Monitoring

- **Cost Management:** The cost\_management module provisions Azure Budgets. You can extend this to include:
  - More granular budgets (e.g., per resource group).
  - Cost export to Azure Storage for detailed analysis in Power BI.
- **Monitoring:**
  - Log Analytics and Application Insights gather logs and metrics.
  - The Event Hub is designed as a central ingestion point for custom application logs or specific Azure service diagnostics.
  - The Azure Function App serves as an event processor. You would deploy custom code to this Function App (e.g., written in C#, Python, Node.js) that reads from the Event Hub, transforms the data, and pushes it to a Power BI streaming dataset or a Data Lake for further analysis.
  - **Power BI:** Direct Power BI dataset/dashboard creation is not handled by Terraform.



This requires Power BI APIs or manual setup.

## 9. Customizing API Management Policies

The `modules/apim/main.tf` includes a basic `xml_content` block for the `openai_proxy_api`. This is where you define your APIM policies.

- **Best Practices:**

- **Named Values:** Use `azurerm_api_management_named_value` to externalize dynamic values (like API keys, tenant IDs, URLs) from your policy XML. For sensitive data like API keys, always integrate with Azure Key Vault. The APIM module includes a basic setup for APIM's managed identity to access Key Vault.
- **External Policy Files:** For complex policies, consider storing them in separate `.xml` files and using Terraform's `file()` function to load them:

```
Example in modules/apim/main.tf
...
xml_content =
file("${path.module}/policies/openai-inbound.xml")
...
```

- **Common Policies:**

- **Authentication/Authorization:** JWT validation, client certificate validation.
- **Rate Limiting/Throttling:** Control API usage.
- **IP Filtering:** Restrict access to specific IP ranges.
- **Caching:** Improve performance.
- **Request/Response Transformation:** Modify headers, body, etc.
- **Cross-Origin Resource Sharing (CORS):** Allow web applications to make cross-domain requests.

## 10. Troubleshooting

- **Terraform Init Errors:** Check your `backend.tf` configuration and ensure the Storage Account and Container exist and are accessible by your Service Principal.
- **Permissions Issues:** Ensure your Azure DevOps Service Connection's Service Principal has the necessary roles/permissions on the target subscription and resource groups.
- **VNet Integration:** APIM VNet integration can be complex. Verify subnet delegation (if applicable for your APIM SKU/VNet type), NSG rules, and sufficient IP address space.
- **Plan/Apply Failures:** Review the Terraform error messages carefully. They usually point to the specific resource and reason for failure.

This README.md should be placed at the root of your repository, next to `main.tf` and `azure-pipelines.yml`.

### Complete Code Modules and Root Files

Now, the actual code for each file as a direct copy-paste.

#### **main.tf (Root)**

```

main.tf
resource "azurerm_resource_group" "aihub_rg" {
 name =
"${var.resource_group_name_prefix}-${var.environment}-aihub"
 location = var.location
 tags = var.tags
}

Data sources for existing network components
data "azurerm_virtual_network" "existing_vnet" {
 name = var.existing_vnet_name
 resource_group_name = var.existing_vnet_resource_group_name
}

data "azurerm_subnet" "apim_subnet" {
 name = var.existing_apim_subnet_name
 virtual_network_name =
data.azurerm_virtual_network.existing_vnet.name
 resource_group_name =
data.azurerm_virtual_network.existing_vnet.resource_group_name
}

data "azurerm_subnet" "aihub_private_endpoint_subnet" {
 name = var.existing_aihub_subnet_name
 virtual_network_name =
data.azurerm_virtual_network.existing_vnet.name
 resource_group_name =
data.azurerm_virtual_network.existing_vnet.resource_group_name
}

#
Module Calls
#

AI Hub Module
module "aihub" {
 source = "../modules/aihub"

 name = var.aihub_name
 resource_group_name = azurerm_resource_group.aihub_rg.name
 location = var.location
 tags = var.tags

 # Private endpoint integration (if AI Hub supports Private Link)
 # Uncomment and configure if `network_isolation_mode` in AI Hub is
 "Enabled"
 vnet_id =
data.azurerm_virtual_network.existing_vnet.id

```

```

 private_endpoint_subnet_id =
data.azurerm_subnet.aihub_private_endpoint_subnet.id
}

API Management Module
module "apim" {
 count = var.enable_apim ? 1 : 0

 source = "./modules/apim"

 name = "${var.aihub_name}-apim-${var.environment}"
 resource_group_name = azurerm_resource_group.aihub_rg.name
 location = var.location
 sku_name = var.apim_sku_name
 publisher_email = var.apim_publisher_email
 publisher_name = var.apim_publisher_name
 tags = var.tags

 # VNet integration for Internal/External mode
 vnet_type = "External" # Adjust to "Internal" if full private access
is needed for APIM
 subnet_id = data.azurerm_subnet.apim_subnet.id

 # Pass AI Hub endpoint for API configuration
 aihub_endpoint = module.aihub.aihub_endpoint
}

Monitoring Module
module "monitoring" {
 count = var.enable_monitoring ? 1 : 0

 source = "./modules/monitoring"

 resource_group_name = azurerm_resource_group.aihub_rg.name
 location = var.location
 environment = var.environment
 tags = var.tags

 use_existing_log_analytics_workspace =
var.use_existing_log_analytics_workspace
 existing_log_analytics_workspace_id =
var.existing_log_analytics_workspace_id

 use_existing_app_insights = var.use_existing_app_insights
 existing_app_insights_id = var.existing_app_insights_id

 event_hub_namespace_sku = var.event_hub_namespace_sku
 event_hub_name = var.event_hub_name

```

```

function_app_name = var.function_app_name
function_app_runtime = var.function_app_runtime
power_bi_workspace_id = var.power_bi_workspace_id
power_bi_dataset_name = var.power_bi_dataset_name

Optionally, pass APIM ID if you want to configure APIM diagnostics
to send to Log Analytics
apim_id = try(module.apim[0].apim_id, null)
}

Cost Management Module
module "cost_management" {
 count = var.enable_cost_management ? 1 : 0

 source = "../modules/cost_management"

 resource_group_name = azurerm_resource_group.aihub_rg.name
 subscription_id = data.azurerm_subscription.current.id
 budget_amount = var.cost_management_budget_amount
 time_grain = var.cost_management_budget_time_grain
 notification_emails = var.cost_management_budget_notification_emails
 tags = var.tags
}

Data source for current subscription (used by cost management)
data "azurerm_subscription" "current" {}

```

## **variables.tf (Root)**

```

variables.tf
Global Variables

variable "environment" {
 description = "The deployment environment (e.g., 'dev', 'test',
'prod')."
 type = string
}

variable "location" {
 description = "The Azure region where resources will be deployed."
 type = string
}

variable "resource_group_name_prefix" {
 description = "Prefix for resource group names. A common suffix will
be added."
 type = string
}

```

```

 default = "rg"
}

variable "tags" {
 description = "A map of tags to apply to all resources."
 type = map(string)
 default = {}
}

Existing Network Details
variable "existing_vnet_name" {
 description = "Name of the existing VNet."
 type = string
}

variable "existing_vnet_resource_group_name" {
 description = "Resource Group name of the existing VNet."
 type = string
}

variable "existing_apim_subnet_name" {
 description = "Name of the existing subnet for APIM."
 type = string
}

variable "existing_aihub_subnet_name" {
 description = "Name of the existing subnet for AI Hub private
endpoints."
 type = string
}

Feature Flags for conditional deployment/inclusion
variable "enable_apim" {
 description = "Whether to deploy Azure API Management."
 type = bool
 default = true
}

variable "enable_cost_management" {
 description = "Whether to deploy cost management resources."
 type = bool
 default = true
}

variable "enable_monitoring" {
 description = "Whether to deploy monitoring resources (Log
Analytics, App Insights, Event Hub, Functions)."
 type = bool

```

```

 default = true
}

Variables for injecting existing resources
variable "use_existing_log_analytics_workspace" {
 description = "Set to true to use an existing Log Analytics
Workspace."
 type = bool
 default = false
}

variable "existing_log_analytics_workspace_id" {
 description = "The resource ID of the existing Log Analytics
Workspace to use."
 type = string
 default = null
}

variable "use_existing_app_insights" {
 description = "Set to true to use an existing Application Insights
resource."
 type = bool
 default = false
}

variable "existing_app_insights_id" {
 description = "The resource ID of the existing Application Insights
resource to use."
 type = string
 default = null
}

APIM specific variables
variable "apim_sku_name" {
 description = "SKU for API Management (e.g., 'Developer_1',
'Standard_V2')."
 type = string
 default = "Developer_1" # Use Standard_V2 or Premium for
production
}

variable "apim_publisher_email" {
 description = "Publisher email for API Management."
 type = string
}

variable "apim_publisher_name" {
 description = "Publisher name for API Management."
}

```

```

 type = string
}

AI Hub specific variables
variable "aihub_name" {
 description = "Name for the Azure AI Hub."
 type = string
}

Cost Management specific variables
variable "cost_management_budget_amount" {
 description = "The amount for the budget alert (e.g., 1000 for $1000)."
 type = number
 default = 500
}

variable "cost_management_budget_time_grain" {
 description = "The time grain for the budget (e.g., 'Monthly', 'Quarterly', 'Annually')."
 type = string
 default = "Monthly"
}

variable "cost_management_budget_notification_emails" {
 description = "A list of email addresses to receive budget alerts."
 type = list(string)
 default = []
}

Monitoring specific variables
variable "event_hub_namespace_sku" {
 description = "SKU for Event Hub Namespace (e.g., 'Standard', 'Basic')."
 type = string
 default = "Standard"
}

variable "event_hub_name" {
 description = "Name of the Event Hub."
 type = string
 default = "aihublogs"
}

variable "function_app_name" {
 description = "Name for the Azure Function App."
 type = string
}

```

```

variable "function_app_runtime" {
 description = "Runtime for the Function App (e.g., 'node', 'dotnet', 'python')."
 type = string
 default = "dotnet" # Example for .NET 8 Isolated
}

variable "power_bi_workspace_id" {
 description = "The ID of the Power BI workspace for integration (optional)."
 type = string
 default = null
}

variable "power_bi_dataset_name" {
 description = "The name of the Power BI dataset for integration (optional)."
 type = string
 default = null
}

```

### **versions.tf (Root)**

```

versions.tf
terraform {
 required_providers {
 azurerm = {
 source = "hashicorp/azurerm"
 version = "~> 3.0" # Use a suitable version
 }
 }
 required_version = ">= 1.0"
}

provider "azurerm" {
 features {}
}

```

### **azure-pipelines.yml (Root)**

```

azure-pipelines.yml
trigger:
 branches:
 include:
 - main

```



```

paths:
 include:
 - '**/*' # Trigger on any change to the repo

pr:
 branches:
 include:
 - main
 paths:
 include:
 - '**/*'

variables:
 - group: TerraformBackendVariables # Link to an Azure DevOps
Variable Group for service connection and backend details
 - name: terraformWorkingDirectory
 value: '$(Build.SourcesDirectory)' # The root of your repo is
where main.tf resides

stages:
 - stage: TerraformPlan
 displayName: 'Terraform Plan'
 jobs:
 - job: PlanDev
 displayName: 'Plan Dev Environment'
 pool:
 vmImage: 'ubuntu-latest'
 steps:
 - checkout: self
 - task: AzureCLI@2
 displayName: 'Azure Login'
 inputs:
 azureSubscription: $(AZURE_SERVICE_CONNECTION) # Your
service connection name
 scriptType: 'bash'
 scriptLocation: 'inlineScript'
 inlineScript: |
 az account show
 - script: |
 terraform init
-backend-config="resource_group_name=$(TFSTATE_RG) "
-backend-config="storage_account_name=$(TFSTATE_SA) "
-backend-config="container_name=$(TFSTATE_CONTAINER) "
-backend-config="key=$(TFSTATE_KEY)-dev.tfstate"
 terraform plan -var-file=environments/dev.tfvars
-out=tfplan-dev
 workingDirectory: '$(terraformWorkingDirectory)'
 displayName: 'Terraform Init and Plan (Dev)'

```

```

- publish: '$(terraformWorkingDirectory)/tfplan-dev'
 artifact: 'tfplan-dev'
 displayName: 'Publish tfplan-dev artifact'

- job: PlanProd
 displayName: 'Plan Prod Environment'
 pool:
 vmImage: 'ubuntu-latest'
 steps:
 - checkout: self
 - task: AzureCLI@2
 displayName: 'Azure Login'
 inputs:
 azureSubscription: $(AZURE_SERVICE_CONNECTION)
 scriptType: 'bash'
 scriptLocation: 'inlineScript'
 inlineScript: |
 az account show
 - script: |
 terraform init
-backend-config="resource_group_name=$(TFSTATE_RG) "
-backend-config="storage_account_name=$(TFSTATE_SA) "
-backend-config="container_name=$(TFSTATE_CONTAINER) "
-backend-config="key=$(TFSTATE_KEY)-prod.tfstate"
 terraform plan -var-file=environments/prod.tfvars
-out=tfplan-prod
 workingDirectory: '$(terraformWorkingDirectory)'
 displayName: 'Terraform Init and Plan (Prod)'
 - publish: '$(terraformWorkingDirectory)/tfplan-prod'
 artifact: 'tfplan-prod'
 displayName: 'Publish tfplan-prod artifact'

- stage: TerraformApply
 displayName: 'Terraform Apply'
 dependsOn: TerraformPlan
 condition: succeeded()
 jobs:
 - deployment: DeployDev
 displayName: 'Deploy Dev Environment'
 environment: 'Dev' # Azure DevOps Environment for approvals
 pool:
 vmImage: 'ubuntu-latest'
 strategy:
 runOnce:
 preDeploy:
 steps:
 - download: current
 artifact: tfplan-dev

```

```

- task: AzureCLI@2
 displayName: 'Azure Login'
 inputs:
 azureSubscription: $(AZURE_SERVICE_CONNECTION)
 scriptType: 'bash'
 scriptLocation: 'inlineScript'
 inlineScript: |
 az account show
 deploy:
 steps:
 - script: |
 terraform init
-backend-config="resource_group_name=$(TFSTATE_RG) "
-backend-config="storage_account_name=$(TFSTATE_SA) "
-backend-config="container_name=$(TFSTATE_CONTAINER) "
-backend-config="key=$(TFSTATE_KEY)-dev.tfstate"
 terraform apply tfplan-dev
 workingDirectory: '$(terraformWorkingDirectory)'
 displayName: 'Terraform Apply (Dev)'

- deployment: DeployProd
 displayName: 'Deploy Prod Environment'
 environment: 'Prod' # Azure DevOps Environment for approvals
 pool:
 vmImage: 'ubuntu-latest'
 strategy:
 runOnce:
 preDeploy:
 steps:
 - download: current
 artifact: tfplan-prod
 - task: AzureCLI@2
 displayName: 'Azure Login'
 inputs:
 azureSubscription: $(AZURE_SERVICE_CONNECTION)
 scriptType: 'bash'
 scriptLocation: 'inlineScript'
 inlineScript: |
 az account show
 deploy:
 steps:
 - script: |
 terraform init
-backend-config="resource_group_name=$(TFSTATE_RG) "
-backend-config="storage_account_name=$(TFSTATE_SA) "
-backend-config="container_name=$(TFSTATE_CONTAINER) "
-backend-config="key=$(TFSTATE_KEY)-prod.tfstate"
 terraform apply tfplan-prod

```

```
workingDirectory: '${terraformWorkingDirectory}'
displayName: 'Terraform Apply (Prod)'
```

## **environments/dev.tfvars**

```
environments/dev.tfvars
environment = "dev"
location = "East US" # Choose your desired Dev
region
apim_sku_name = "Developer_1"
apim_publisher_email = "dev-admin@example.com"
apim_publisher_name = "Dev Team"
aihub_name = "dev-aihub"
cost_management_budget_amount = 200
cost_management_budget_notification_emails =
["devops-alerts@example.com"]
function_app_name = "dev-aihub-func" # Must be globally
unique for storage/function name
existing_vnet_name = "dev-vnet" # Replace with your
existing Dev VNet name
existing_vnet_resource_group_name = "dev-network-rg" # Replace with RG
of your existing Dev VNet
existing_apim_subnet_name = "apim-subnet" # Replace with your
APIM subnet name
existing_aihub_subnet_name = "private-endpoint-subnet" # Replace
with your private endpoint subnet name
tags = {
 Environment = "Dev"
 Project = "AIHub"
 CostCenter = "DevOps"
}
use_existing_log_analytics_workspace = false # Set to true to use an
existing LAW
existing_log_analytics_workspace_id =
"/subscriptions/xxx/resourceGroups/yyy/providers/Microsoft.Operational
Insights/workspaces/my-existing-la" # Uncomment and provide ID if
using existing
use_existing_app_insights = false # Set to true to use an existing App
Insights
existing_app_insights_id =
"/subscriptions/xxx/resourceGroups/yyy/providers/Microsoft.Insights/co
mponents/my-existing-appinsights" # Uncomment and provide ID if using
existing

Feature flags - set to false to exclude a module from deployment
enable_apim = true
enable_cost_management = true
```

```
enable_monitoring = true
```

```
Optional Power BI integration variables
```

```
power_bi_workspace_id = "YOUR_POWER_BI_WORKSPACE_ID"
```

```
power_bi_dataset_name = "AIHubTelemetry"
```

## **environments/prod.tfvars**

```
environments/prod.tfvars
```

```
environment = "prod"
```

```
location = "West Europe" # Choose your desired
```

```
Prod region
```

```
apim_sku_name = "Premium" # Production SKU for APIM
```

```
apim_publisher_email = "prod-admin@example.com"
```

```
apim_publisher_name = "Operations Team"
```

```
aihub_name = "prod-aihub"
```

```
cost_management_budget_amount = 5000
```

```
cost_management_budget_notification_emails =
```

```
["prod-alerts@example.com", "finance-team@example.com"]
```

```
function_app_name = "prod-aihub-func" # Must be
```

```
globally unique for storage/function name
```

```
existing_vnet_name = "prod-vnet" # Replace with your
```

```
existing Prod VNet name
```

```
existing_vnet_resource_group_name = "prod-network-rg" # Replace with
```

```
RG of your existing Prod VNet
```

```
existing_apim_subnet_name = "apim-subnet" # Replace with your
```

```
APIM subnet name
```

```
existing_aihub_subnet_name = "private-endpoint-subnet" # Replace
```

```
with your private endpoint subnet name
```

```
tags = {
```

```
 Environment = "Prod"
```

```
 Project = "AIHub"
```

```
 CostCenter = "CoreAI"
```

```
 Criticality = "High"
```

```
}
```

```
use_existing_log_analytics_workspace = true # Typically use existing
central monitoring in Prod
```

```
existing_log_analytics_workspace_id =
```

```
"/subscriptions/YOUR_PROD_SUB_ID/resourceGroups/PROD_CORE_MONITORING_R
G/providers/Microsoft.OperationalInsights/workspaces/prod-central-log-
analytics" # **IMPORTANT: Replace with your actual Prod LAW ID**
```

```
use_existing_app_insights = true
```

```
existing_app_insights_id =
```

```
"/subscriptions/YOUR_PROD_SUB_ID/resourceGroups/PROD_CORE_MONITORING_R
G/providers/Microsoft.Insights/components/prod-central-appinsights" #
IMPORTANT: Replace with your actual Prod App Insights ID
```

```

Feature flags - set to false to exclude a module from deployment
enable_apim = true
enable_cost_management = true
enable_monitoring = true

Optional Power BI integration variables
power_bi_workspace_id = "YOUR_POWER_BI_WORKSPACE_ID"
power_bi_dataset_name = "AIHubTelemetryProd"

```

### **environments/test.tfvars (Example, customize as needed)**

```

environments/test.tfvars
environment = "test"
location = "East US 2" # Choose your desired
Test region
apim_sku_name = "Standard_V2" # Often a step up
from Developer for Test
apim_publisher_email = "test-admin@example.com"
apim_publisher_name = "Test Team"
aihub_name = "test-aihub"
cost_management_budget_amount = 500
cost_management_budget_notification_emails =
["test-alerts@example.com"]
function_app_name = "test-aihub-func"
existing_vnet_name = "test-vnet"
existing_vnet_resource_group_name = "test-network-rg"
existing_apim_subnet_name = "apim-subnet"
existing_aihub_subnet_name = "private-endpoint-subnet"
tags = {
 Environment = "Test"
 Project = "AIHub"
}
use_existing_log_analytics_workspace = false
use_existing_app_insights = false

enable_apim = true
enable_cost_management = true
enable_monitoring = true

```

### **modules/aihub/main.tf**

```

modules/aihub/main.tf
resource "azurerm_ai_hub" "main" {
 name = var.name
 resource_group_name = var.resource_group_name
 location = var.location
}

```

```

tags = var.tags

network_isolation_mode = "Disabled" # Default, or "Enabled" for
private access
If "Enabled", you would typically configure private endpoints and
DNS.
Azure AI Hub's Private Link details can be complex; ensure you
configure them correctly.
For demonstration, we'll keep private endpoint resources commented
out unless explicitly enabled.
}

Example of Private Endpoint for AI Hub (if network_isolation_mode is
"Enabled" and desired)
resource "azurerm_private_endpoint" "aihub_pe" {
count = var.enable_private_endpoint ? 1 : 0 # Variable
`enable_private_endpoint` would be needed in vars.tf
#
name = "${var.name}-pe"
location = var.location
resource_group_name = var.resource_group_name
subnet_id = var.private_endpoint_subnet_id
#
private_service_connection {
name = "${var.name}-psc"
is_manual_connection = false
private_connection_resource_id = azurerm_ai_hub.main.id
subresource_names = ["aihub"] # Common subresource
for AI Hub
}
#
private_dns_zone_group {
name = "default"
private_dns_zone_ids =
[azurerm_private_dns_zone.aihub_private_dns_zone[0].id] # Needs the
DNS Zone
}
}

Private DNS Zone for AI Hub (if private endpoint is used)
resource "azurerm_private_dns_zone" "aihub_private_dns_zone" {
count = var.enable_private_endpoint ? 1 : 0
#
name = "privatelink.api.azureml.ms" # Common DNS
zone for Azure ML/AI Hub
resource_group_name = var.resource_group_name
}

```

```

Private DNS Zone VNet Link
resource "azurerm_private_dns_zone_virtual_network_link"
"aihub_dns_link" {
count = var.enable_private_endpoint ? 1 : 0
#
name = "${var.name}-dns-link"
resource_group_name = var.resource_group_name
private_dns_zone_name =
azurerm_private_dns_zone.aihub_private_dns_zone[0].name
virtual_network_id = var.vnet_id
}

```

### modules/aihub/variables.tf

```

modules/aihub/variables.tf
variable "name" {
 description = "The name of the Azure AI Hub."
 type = string
}

variable "resource_group_name" {
 description = "The name of the resource group where the AI Hub will
be deployed."
 type = string
}

variable "location" {
 description = "The Azure region where the AI Hub will be deployed."
 type = string
}

variable "tags" {
 description = "A map of tags to apply to the AI Hub resource."
 type = map(string)
 default = {}
}

Private Endpoint variables (uncomment if you enable private
endpoints for AI Hub)
variable "vnet_id" {
 description = "The ID of the existing Virtual Network for private
endpoint."
 type = string
}

variable "private_endpoint_subnet_id" {
 description = "The ID of the existing subnet for the AI Hub private

```



```

endpoint."
 type = string
}

variable "enable_private_endpoint" {
description = "Set to true to enable private endpoint for AI Hub."
type = bool
default = false
}

```

### **modules/aihub/outputs.tf**

```

modules/aihub/outputs.tf
output "aihub_id" {
 description = "The ID of the Azure AI Hub."
 value = azurerm_ai_hub.main.id
}

output "aihub_endpoint" {
 description = "The primary endpoint of the Azure AI Hub."
 value = azurerm_ai_hub.main.primary_hub_uri
}

```

### **modules/apim/main.tf**

```

modules/apim/main.tf
resource "azurerm_api_management_service" "main" {
 name = var.name
 location = var.location
 resource_group_name = var.resource_group_name
 publisher_name = var.publisher_name
 publisher_email = var.publisher_email
 sku_name = var.sku_name
 tags = var.tags

 # VNet integration (Internal/External mode)
 virtual_network_configuration {
 subnet_id = var.subnet_id
 }
 virtual_network_type = var.vnet_type # "External" or "Internal"

 # Optional: For Premium SKU, configure availability zones if needed
 # zones = [1, 2]

 # Optional: Configure custom hostname for the portal, gateway, etc.
 # hostname_configuration {

```

```

proxy {
host_name = "api.yourdomain.com"
key_vault_id =
"/subscriptions/xyz/resourceGroups/abc/providers/Microsoft.KeyVault/va
ults/yourkv/secrets/yourcertsecret"
negotiate_client_certificate = false
}
portal {
host_name = "portal.yourdomain.com"
key_vault_id =
"/subscriptions/xyz/resourceGroups/abc/providers/Microsoft.KeyVault/va
ults/yourkv/secrets/yourcertsecret"
}
}
}

```

```

Example API for Azure OpenAI Model via AI Hub
resource "azurerm_api_management_api" "openai_proxy_api" {
 api_management_id = azurerm_api_management_service.main.id
 display_name = "OpenAI Model Proxy"
 path = "openai" # e.g.,
/openai/deployments/gpt-35-turbo/chat/completions
 protocols = ["https"]
 service_url = var.aihub_endpoint # This would be the AI Hub's
endpoint to manage models

```

```

 revision = "1"
 api_type = "http"
 description = "API to proxy requests to Azure OpenAI models
managed by AI Hub."

```

```

Custom policies for OpenAI API
xml_content = <<XML
<policies>
 <inbound>
 <base />
 <set-header name="api-key" exists-action="override">
 <value>{{openai-api-key}}</value> </set-header>
 <rate-limit-by-key calls="1000" renewal-period="3600"
counter-key="@context.Subscription.Id)" />
 </inbound>
 <backend>
 <base />
 </backend>
 <outbound>
 <base />
 </outbound>
 <on-error>

```

```

 <base />
 <return-response>
 <set-status code="500" reason="InternalServerError" />
 <set-body>{"statusCode": 500, "message": "An unexpected
error occurred."}</set-body>
 </return-response>
 </on-error>
</policies>
XML
}

```

```

Product for application teams to subscribe to
resource "azurerm_api_management_product" "openai_product" {
 api_management_name = azurerm_api_management_service.main.name
 resource_group_name =
azurerm_api_management_service.main.resource_group_name
 product_id = "openai-access" # Unique identifier for the
product
 display_name = "OpenAI Model Access"
 subscription_required = true
 approval_required = true # Requires admin approval for new
subscriptions
 published = true

```

```

 # Product-level policies (e.g., global rate limits for the product)
 xml_content = <<XML
<policies>
 <inbound>
 <base />
 <rate-limit calls="5000" renewal-period="3600" />
 </inbound>
 <backend>
 <base />
 </backend>
 <outbound>
 <base />
 </outbound>
 <on-error>
 <base />
 </on-error>
</policies>
XML
}

```

```

resource "azurerm_api_management_product_api" "link_openai_to_product"
{
 api_management_name = azurerm_api_management_service.main.name
 resource_group_name =

```

```

azurerm_api_management_service.main.resource_group_name
 product_id =
azurerm_api_management_product.openai_product.product_id
 api_name =
azurerm_api_management_api.openai_proxy_api.name
}

Key Vault and Managed Identity for secure secrets (e.g., OpenAI API
Key)
resource "azurerm_key_vault" "apim_key_vault" {
 name = "${var.name}-kv" # Example naming
 convention
 location = var.location
 resource_group_name = var.resource_group_name
 sku_name = "standard"
 tenant_id =
data.azurerm_client_config.current.tenant_id
 soft_delete_retention_days = 7
 purge_protection_enabled = false
 tags = var.tags
}

data "azurerm_client_config" "current" {}

Use System Assigned Managed Identity for APIM for simplicity in
example
resource "azurerm_role_assignment" "apim_key_vault_reader" {
 # This role assignment grants APIM's managed identity access to Key
Vault secrets.
 # Ensure the APIM service has System Assigned Managed Identity
enabled (default for azurerm_api_management_service).
 scope = azurerm_key_vault.apim_key_vault.id
 role_definition_name = "Key Vault Secrets User" # Allows reading
secrets
 principal_id =
azurerm_api_management_service.main.identity[0].principal_id
}

Example Secret in Key Vault (replace with your actual OpenAI key)
resource "azurerm_key_vault_secret" "openai_api_key_secret" {
 name = "OpenAIAIHubKey"
 value = "YOUR_ACTUAL_OPENAI_API_KEY_HERE" # **IMPORTANT:
Replace with a secure value or inject via pipeline variables/secrets**
 key_vault_id = azurerm_key_vault.apim_key_vault.id
 content_type = "text/plain"
}

Named Value in APIM to reference Key Vault secret

```

```

resource "azurerm_api_management_named_value" "openai_api_key_nv" {
 api_management_name = azurerm_api_management_service.main.name
 resource_group_name =
azurerm_api_management_service.main.resource_group_name
 name = "openai-api-key"
 display_name = "OpenAI API Key"
 value_from_key_vault {
 secret_id = azurerm_key_vault_secret.openai_api_key_secret.id
 # identity_client_id is not needed for System Assigned Identity
 }
 secret = true
}

```

## modules/apim/variables.tf

```

modules/apim/variables.tf
variable "name" {
 description = "The name of the API Management service."
 type = string
}

variable "resource_group_name" {
 description = "The name of the resource group where APIM will be
deployed."
 type = string
}

variable "location" {
 description = "The Azure region where APIM will be deployed."
 type = string
}

variable "sku_name" {
 description = "SKU for API Management (e.g., 'Developer_1',
'Standard_V2', 'Premium')."
 type = string
}

variable "publisher_email" {
 description = "Publisher email for API Management."
 type = string
}

variable "publisher_name" {
 description = "Publisher name for API Management."
 type = string
}

```

```

variable "tags" {
 description = "A map of tags to apply to the APIM resource."
 type = map(string)
 default = {}
}

variable "vnet_type" {
 description = "The type of VNet integration for APIM ('External' or 'Internal')."
 type = string
}

variable "subnet_id" {
 description = "The ID of the subnet for APIM VNet integration."
 type = string
}

variable "aihub_endpoint" {
 description = "The endpoint of the Azure AI Hub to be used as a backend for APIM."
 type = string
}

```

### **modules/apim/outputs.tf**

```

modules/apim/outputs.tf
output "apim_id" {
 description = "The ID of the API Management service."
 value = azurerm_api_management_service.main.id
}

output "apim_gateway_url" {
 description = "The URL of the API Management gateway."
 value = azurerm_api_management_service.main.gateway_url
}

```

### **modules/cost\_management/main.tf**

```

modules/cost_management/main.tf
resource "azurerm_consumption_budget_subscription" "aihub_budget" {
 name = "${var.resource_group_name}-aihub-budget" #
 Unique budget name
 subscription_id = var.subscription_id
 amount = var.budget_amount
 time_grain = var.time_grain
}

```

```

 time_period {
 start_date = formatdate("YYYY-MM-01Z", timestamp()) # Budget
starts at the beginning of the current month
 }
 notification {
 enabled = true
 operator = "GreaterThan"
 threshold = 90 # % of budget to trigger alert (e.g., 90% of total
amount)
 contact_emails = var.notification_emails
 }
 tags = var.tags
 }
}

```

```

You could also add other types of budget alerts or exports here.
For example, to export cost data to a storage account for Power BI:
resource "azurerm_consumption_export" "monthly_cost_export" {
name = "${var.resource_group_name}-cost-export"
resource_group_name = var.resource_group_name
subscription_id = var.subscription_id
active = true
delivery_info {
destination_container_id =
azurerm_storage_container.cost_export_container.id # Requires a
storage account and container
root_folder_path = "/costdata"
}
definition {
type = "ActualCost"
timeframe = "Monthly"
dataset {
granularity = "Daily"
Optional: configuration { columns = [] }
}
}
tags = var.tags
}

```

## modules/cost\_management/variables.tf

```

modules/cost_management/variables.tf
variable "resource_group_name" {
 description = "The name of the resource group that this budget
relates to (for naming, not scope)."
 type = string
}

```

```

variable "subscription_id" {
 description = "The ID of the subscription to apply the budget to."
 type = string
}

variable "budget_amount" {
 description = "The amount for the budget alert."
 type = number
}

variable "time_grain" {
 description = "The time grain for the budget (e.g., 'Monthly',
'Quarterly', 'Annually')."
 type = string
}

variable "notification_emails" {
 description = "A list of email addresses to receive budget alerts."
 type = list(string)
}

variable "tags" {
 description = "A map of tags to apply to cost management resources."
 type = map(string)
 default = {}
}

```

### **modules/cost\_management/outputs.tf**

```

modules/cost_management/outputs.tf
output "budget_id" {
 description = "The ID of the Azure Consumption Budget."
 value =
azurerm_consumption_budget_subscription.aihub_budget.id
}

```

### **modules/monitoring/main.tf**

```

modules/monitoring/main.tf
Conditional Log Analytics Workspace creation or reference
resource "azurerm_log_analytics_workspace" "main" {
 count = var.use_existing_log_analytics_workspace ? 0 : 1

 name =
"${var.environment}-log-analytics-${var.resource_group_name}" # Unique
name for new LAW

```



```

 location = var.location
 resource_group_name = var.resource_group_name
 sku = "PerGB2018" # Consider "Consumption" or
"Dedicated" for larger scale
 retention_in_days = 30 # Adjust retention as per compliance
 tags = var.tags
}

data "azurerm_log_analytics_workspace" "existing" {
 count = var.use_existing_log_analytics_workspace ? 1 : 0
 id = var.existing_log_analytics_workspace_id
}

locals {
 log_analytics_workspace_id =
var.use_existing_log_analytics_workspace ?
data.azurerm_log_analytics_workspace.existing[0].id :
azurerm_log_analytics_workspace.main[0].id
}

Conditional Application Insights creation or reference
resource "azurerm_application_insights" "main" {
 count = var.use_existing_app_insights ? 0 : 1

 name =
"${var.environment}-appinsights-${var.resource_group_name}" # Unique
name for new App Insights
 location = var.location
 resource_group_name = var.resource_group_name
 application_type = "web" # Common type, adjust if needed
 workspace_id = local.log_analytics_workspace_id
 tags = var.tags
}

data "azurerm_application_insights" "existing" {
 count = var.use_existing_app_insights ? 1 : 0
 id = var.existing_app_insights_id
}

Event Hub Namespace and Event Hub for ingesting custom logs/metrics
resource "azurerm_eventhub_namespace" "main" {
 name = "${var.environment}-ehn"
 location = var.location
 resource_group_name = var.resource_group_name
 sku = var.event_hub_namespace_sku
 capacity = 1 # Scale up capacity as needed
 tags = var.tags
}

```

```

resource "azurerm_eventhub" "aihub_logs" {
 name = var.event_hub_name
 namespace_name = azurerm_eventhub_namespace.main.name
 resource_group_name =
azurerm_eventhub_namespace.main.resource_group_name
 partition_count = 4 # Recommended 2-4 partitions for typical use
cases
 message_retention_in_days = 1 # Adjust retention as needed
}

resource "azurerm_eventhub_namespace_authorization_rule" "send_rule" {
 name = "send"
 namespace_name = azurerm_eventhub_namespace.main.name
 resource_group_name =
azurerm_eventhub_namespace.main.resource_group_name
 listen = false
 send = true
 manage = false
}

Azure Function App for processing Event Hub data and integration
(e.g., Power BI)
resource "azurerm_storage_account" "function_app_storage" {
 name = "${lower(replace(var.function_app_name,
"- ", ""))}" # Function App storage names must be lowercase
alphanumeric
 resource_group_name = var.resource_group_name
 location = var.location
 account_tier = "Standard"
 account_replication_type = "LRS" # Choose appropriate replication
tags = var.tags
}

resource "azurerm_application_insights" "function_app_insights" {
 name = "${var.function_app_name}-appinsights"
 location = var.location
 resource_group_name = var.resource_group_name
 application_type = "other" # Or "web" depending on function type
 workspace_id = local.log_analytics_workspace_id
 tags = var.tags
}

resource "azurerm_app_service_plan" "function_app_plan" {
 name = "${var.function_app_name}-plan"
 location = var.location
 resource_group_name = var.resource_group_name
 kind = "FunctionApp"

```

```

 sku {
 tier = "Consumption" # For serverless or PremiumV2 for dedicated
plan
 size = "Y1"
 }
 tags = var.tags
}

resource "azurerm_function_app" "main" {
 name = var.function_app_name
 location = var.location
 resource_group_name = var.resource_group_name
 app_service_plan_id =
azurerm_app_service_plan.function_app_plan.id
 storage_account_name =
azurerm_storage_account.function_app_storage.name
 storage_account_access_key =
azurerm_storage_account.function_app_storage.primary_access_key
 app_insights_key =
azurerm_application_insights.function_app_insights.instrumentation_key
 app_insights_connection_string =
azurerm_application_insights.function_app_insights.connection_string

 os_type = "Windows" # Or "Linux"
 https_only = true

 identity {
 type = "SystemAssigned" # Enable Managed Identity for accessing
other Azure services
 }

 site_config {
 scm_type = "None" # Or "LocalGit", "GitHub", etc.
 application_stack {
 # Configure runtime as per your function code
 dotnet_version = "8.0" # Example for .NET 8 Isolated
 use_dotnet_isolated_runtime = true
 # python_version = "3.9" # For Python functions
 # node_version = "18" # For Node.js functions
 }
 }

 app_settings = {
 FUNCTIONS_WORKER_RUNTIME = var.function_app_runtime
 EventHubConnectionString =
azurerm_eventhub_namespace.main.default_primary_connection_string
 EventHubName = azurerm_eventhub.aihub_logs.name
 # Add any other required app settings for your function logic,

```

```

e.g., Power BI connection details
 # PowerBIWorkspaceId = var.power_bi_workspace_id # If pushing
directly to Power BI
 # PowerBIDatasetName = var.power_bi_dataset_name
 }
 tags = var.tags
}

Configure Diagnostic Settings for APIM to send logs to Log Analytics
resource "azurerm_monitor_diagnostic_setting" "apim_diagnostics" {
 count = var.apim_id != null ? 1 : 0 # Only create if APIM is enabled

 name = "apim-to-loganalytics"
 target_resource_id = var.apim_id
 log_analytics_workspace_id = local.log_analytics_workspace_id

 metric {
 category = "AllMetrics"
 enabled = true
 retention_policy {
 enabled = false
 }
 }
}

log {
 category = "GatewayLogs"
 enabled = true
 retention_policy {
 enabled = false
 }
}

log {
 category = "AuditEvent"
 enabled = true
 retention_policy {
 enabled = false
 }
}

Add other relevant log categories as needed
log {
category = "TenantLogs"
enabled = true
}
}

```

## **modules/monitoring/variables.tf**

```
modules/monitoring/variables.tf
variable "resource_group_name" {
 description = "The name of the resource group for monitoring
resources."
 type = string
}

variable "location" {
 description = "The Azure region for monitoring resources."
 type = string
}

variable "environment" {
 description = "The deployment environment for naming."
 type = string
}

variable "tags" {
 description = "A map of tags to apply to monitoring resources."
 type = map(string)
 default = {}
}

variable "use_existing_log_analytics_workspace" {
 description = "Set to true to use an existing Log Analytics
Workspace."
 type = bool
}

variable "existing_log_analytics_workspace_id" {
 description = "The resource ID of the existing Log Analytics
Workspace to use."
 type = string
 default = null
}

variable "use_existing_app_insights" {
 description = "Set to true to use an existing Application Insights
resource."
 type = bool
}

variable "existing_app_insights_id" {
 description = "The resource ID of the existing Application Insights
resource to use."
 type = string
}
```

```

 default = null
}

variable "event_hub_namespace_sku" {
 description = "SKU for Event Hub Namespace (e.g., 'Standard',
'Basic')."
 type = string
}

variable "event_hub_name" {
 description = "Name of the Event Hub within the namespace."
 type = string
}

variable "function_app_name" {
 description = "Name for the Azure Function App."
 type = string
}

variable "function_app_runtime" {
 description = "Runtime for the Function App (e.g., 'node', 'dotnet',
'python')."
 type = string
}

variable "power_bi_workspace_id" {
 description = "The ID of the Power BI workspace for integration
(optional)."
 type = string
 default = null
}

variable "power_bi_dataset_name" {
 description = "The name of the Power BI dataset for integration
(optional)."
 type = string
 default = null
}

variable "apim_id" {
 description = "The resource ID of the APIM service for diagnostic
settings (optional)."
 type = string
 default = null
}

```

## modules/monitoring/outputs.tf

```
modules/monitoring/outputs.tf
output "log_analytics_workspace_id" {
 description = "The ID of the Log Analytics Workspace."
 value = local.log_analytics_workspace_id
}

output "application_insights_id" {
 description = "The ID of the Application Insights resource."
 value = var.use_existing_app_insights ?
data.azurerm_application_insights.existing[0].id :
azurerm_application_insights.main[0].id
}

output "event_hub_namespace_id" {
 description = "The ID of the Event Hub Namespace."
 value = azurerm_eventhub_namespace.main.id
}

output "event_hub_id" {
 description = "The ID of the Event Hub."
 value = azurerm_eventhub.aihub_logs.id
}

output "function_app_id" {
 description = "The ID of the Function App."
 value = azurerm_function_app.main.id
}
```

This complete set of files provides a well-organized and modular Terraform project, ready for deployment via Azure Pipelines. Remember to carefully review and customize the environments/\*.tfvars files and the backend.tf to match your Azure environment and security requirements.