Achieving IaC and documentation via terraform provider

CNTUG meetup #65
2025/01/17

Jeff Chen

About Me

Jeff Chen

- Currently work as Devops, Site Reliability Engineer, aka 水管工
- OSS lover
- My LinkedIn

Outline

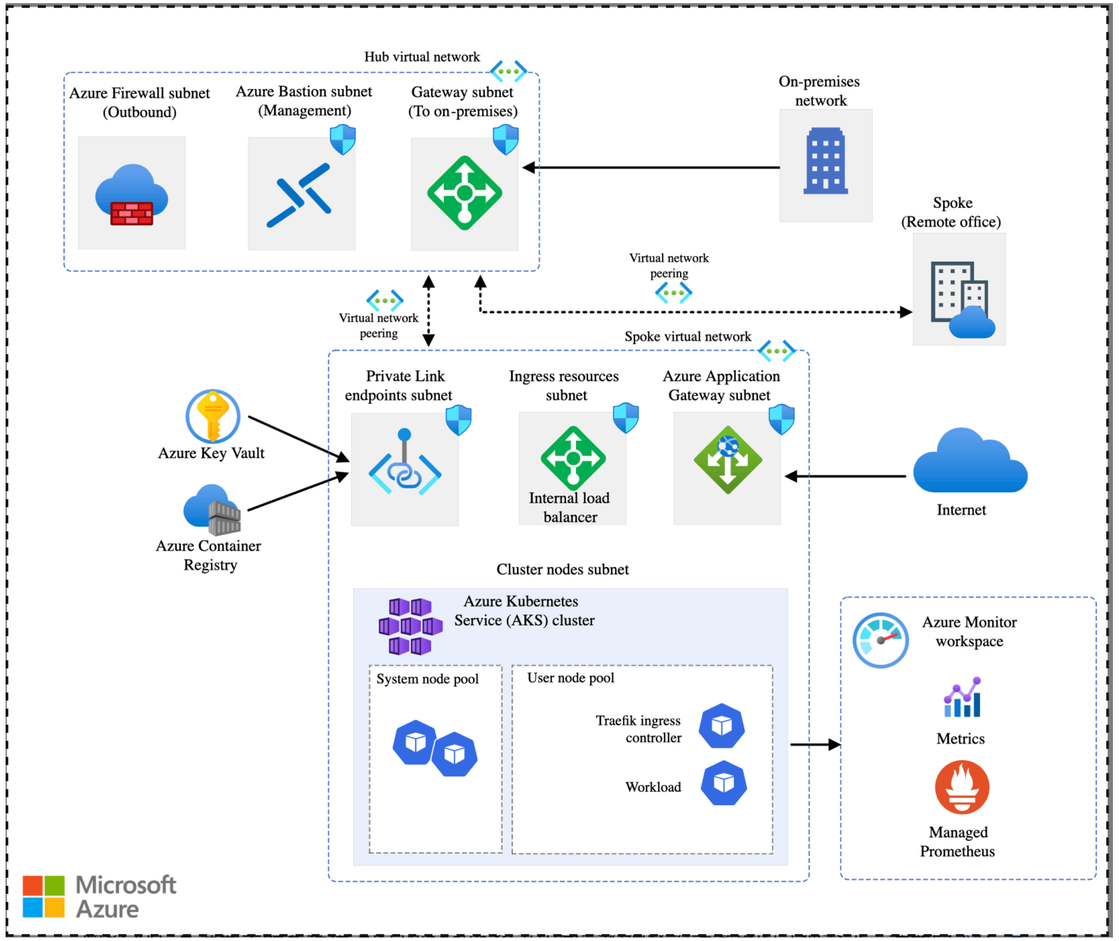
- Background story
- Motivation
- Evaluation
- Terraform provider development
- Demo
- Q&A

Background story

Internet of Things based application

- Kubernetes
  - Application development
- Cloud infrastructure
  - VM, Network, security

- Ref: architecture doc



Background story

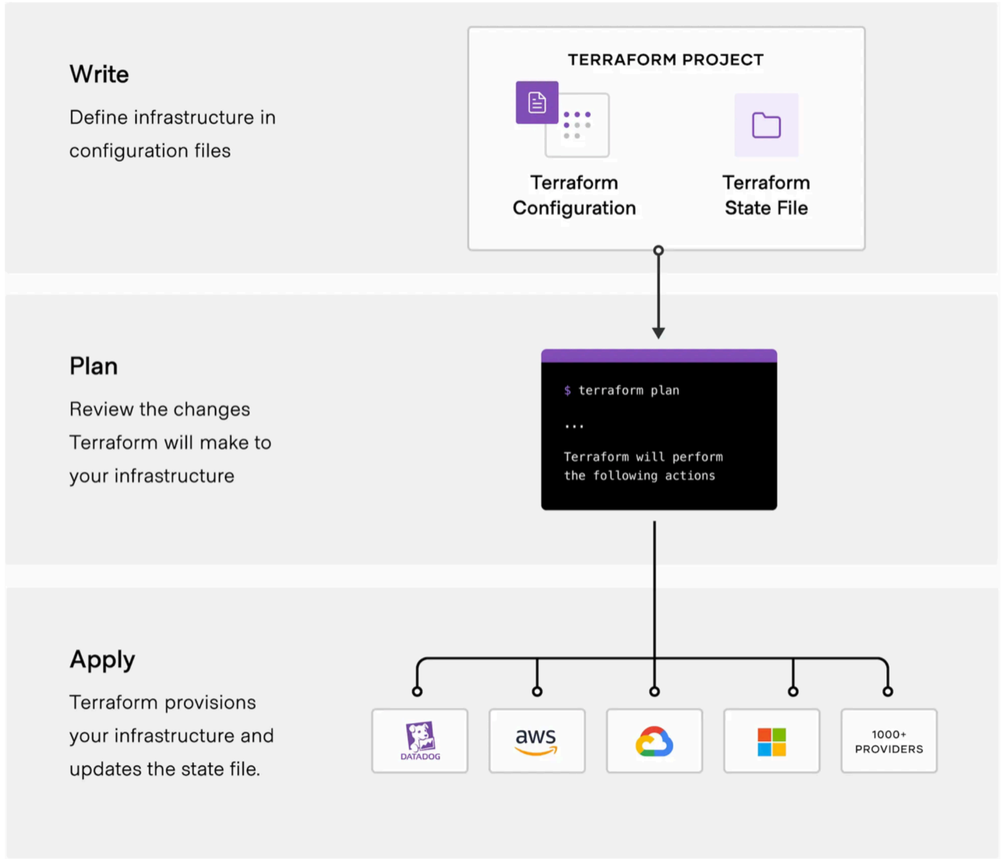
Infrastructure as code is needed

- Business requirements => public DNS / IP change
- Infrastructure enhancement => private DNS, network private endpoint
- Security enhancement => TLS version, cipher suites
  - Infrastructure as Code concept is needed
- Terraform HCL code is not readable for all people.
  - Need a just in time documentation for application, infra & pm team.

Motivation

How terraform works

- Cloud Provider: Azure, AWS, GCP
- Write HCL
- Plan
- Apply



- Ref: doc 1, doc 2

Motivation

Infrastructure as code

Change Requirement

write terraform code

```
resource "azurerm_virtual_machine" "jumpbox" {
  name                  = "jumpbox"
  location              = var.location
  resource_group_name   = azurerm_resource_group.vmss.name
  network_interface_ids = [azurerm_network_interface.jumpbox.id]
  vm_size               = "Standard_DS1_v2"

  storage_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "16.04-LTS"
    version   = "latest"
  }

  storage_os_disk {
    name              = "jumpbox-osdisk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "jumpbox"
    admin_username = var.admin_user
    admin_password = local.admin_password
  }

  os_profile_linux_config {
    disable_password_authentication = false
  }

  tags = var.tags
}
```

Git

CI / CD pipeline

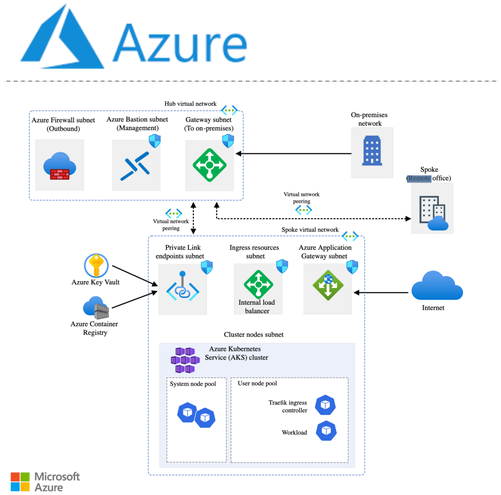
terraform test

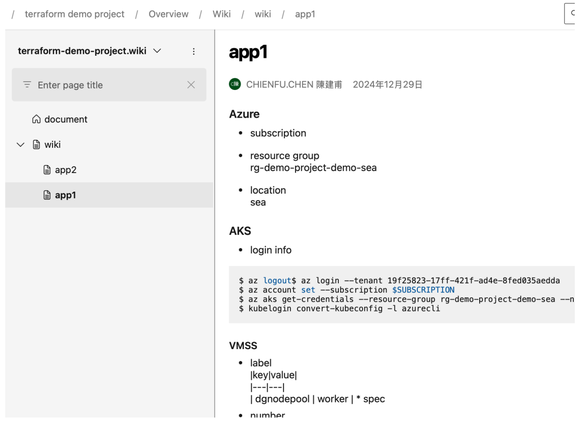
terraform plan

terraform apply

Integration test

generate document





Evaluation

Terraform Azuredevops provider Resource Wiki

azuredevops

AZUREDEVOPS DOCUMENTATION

Filter

- runpipeline
- azuredevops_serviceendpoint_servicefabric
- azuredevops_serviceendpoint_sonarcloud
- azuredevops_serviceendpoint_sonarqube
- azuredevops_serviceendpoint_ssh
- azuredevops_servicehook_permissions
- azuredevops_servicehook_storage_queue_pipelines
- azuredevops_tagging_permissions
- azuredevops_team
- azuredevops_team_administrators
- azuredevops_team_members
- azuredevops_user_entitlement
- azuredevops_variable_group
- azuredevops_variable_group_permissions
- azuredevops_wiki

azuredevops_wiki

Manages Wikis within Azure DevOps project.

Example Usage

```
resource "azuredevops_project" "example" {
  name        = "Example Project"
  description = "Managed by Terraform"
}

resource "azuredevops_git_repository" "example" {
  project_id = azuredevops_project.example.id
  name       = "Example Repository"
  initialization {
    init_type = "Clean"
  }
}

resource "azuredevops_wiki" "example" {
  name       = "Example project wiki "
  project_id = azuredevops_project.example.id
  type       = "projectWiki"
}
```

azuredevops > internal > service > wiki > resource_wiki.go > ResourceWiki

```
package wiki

import (
    "fmt"
    "time"

    "github.com/google/uuid"
    "github.com/hashicorp/terraform-plugin-sdk/v2/helper/schema"
    "github.com/hashicorp/terraform-plugin-sdk/v2/helper/validation"
    "github.com/microsoft/azure-devops-go-api/azuredevops/v7/git"
    "github.com/microsoft/azure-devops-go-api/azuredevops/v7/wiki"
    "github.com/microsoft/terraform-provider-azuredevops/azuredevops/internal/client"
    "github.com/microsoft/terraform-provider-azuredevops/azuredevops/internal/utils/converter"
)

func ResourceWiki() *schema.Resource {
    return &schema.Resource{
        Create: resourceWikiCreate,
        Read:   resourceWikiRead,
        Update: resourceWikiUpdate,
        Delete: resourceWikiDelete,
        Timeouts: &schema.ResourceTimeout{…
        },
        Importer: &schema.ResourceImporter{…
        },
        Schema: map[string]*schema.Schema{
            "name": {
                Type:     schema.TypeString,
                Required: true,
            },
            "type": {
                Type:     schema.TypeString,
                Required: true,
                ValidateFunc: validation.StringInSlice([]string{
                    string(wiki.WikiTypeValues.ProjectWiki),
                    string(wiki.WikiTypeValues.CodeWiki)},
                    false),
            },
            "project_id": {
                Type:     schema.TypeString,
                Optional: true,
            },
            "mapped_path": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "repository_id": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "version": {
```

azuredevops > internal > service > wiki > resource_wiki.go > ResourceWiki

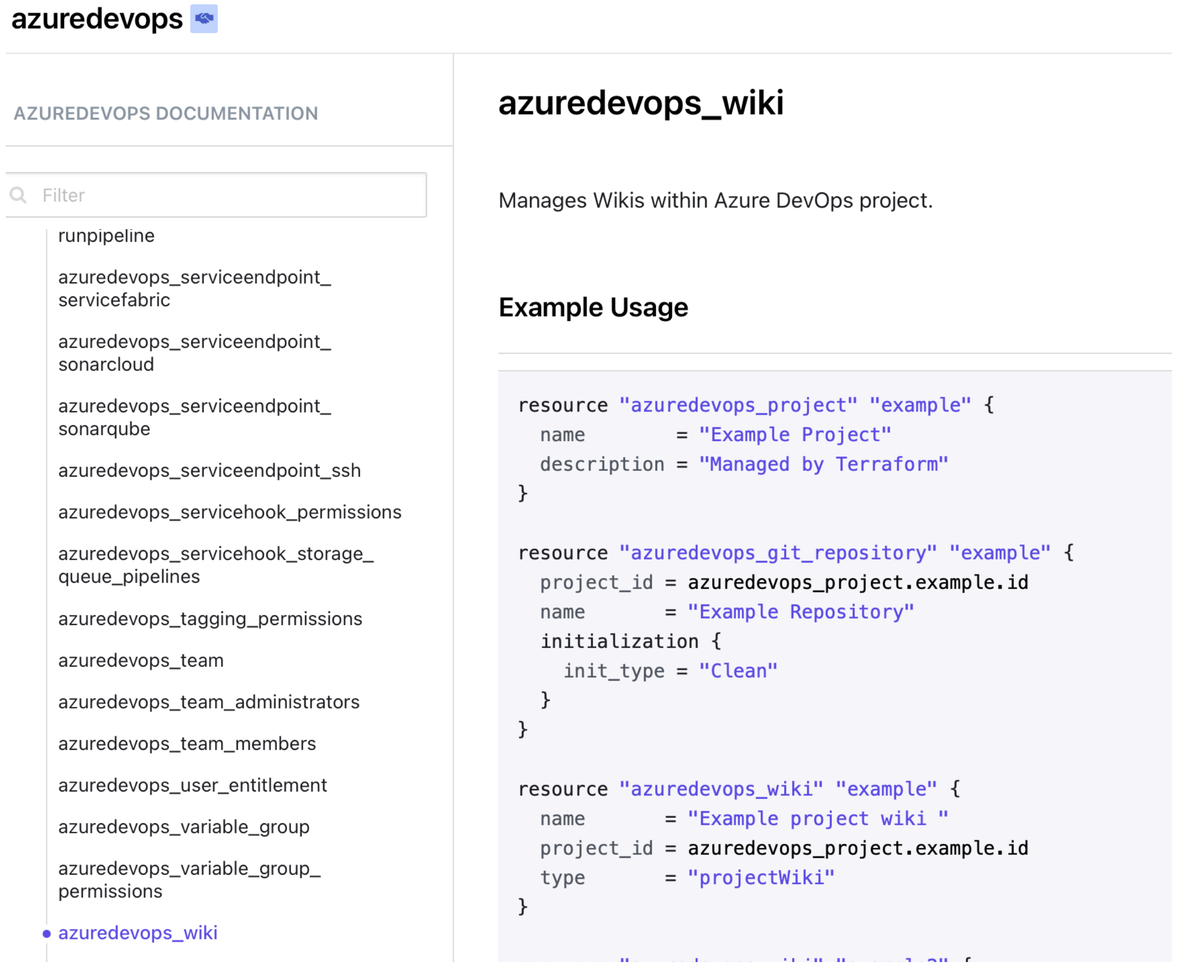
```
func ResourceWiki() *schema.Resource {
            },
            "type": {
                Type:     schema.TypeString,
                Required: true,
                ValidateFunc: validation.StringInSlice([]string{
                    string(wiki.WikiTypeValues.ProjectWiki),
                    string(wiki.WikiTypeValues.CodeWiki)},
                    false),
            },
            "project_id": {
                Type:     schema.TypeString,
                Optional: true,
            },
            "mapped_path": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "repository_id": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "version": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "remote_url": {
                Type:     schema.TypeString,
                Computed: true,
            },
            "url": {
                Type:     schema.TypeString,
                Computed: true,
            },
        },
    }
}

func resourceWikiCreate(d *schema.ResourceData, m interface{}) error {…
}

func resourceWikiRead(d *schema.ResourceData, m interface{}) error {…
}

func resourceWikiUpdate(d *schema.ResourceData, m interface{}) error {…
}

func resourceWikiDelete(d *schema.ResourceData, m interface{}) error {
    clients := m.(*client.AggregatedClient)
```

- https://registry.terraform.io/providers/microsoft/azuredevops/latest/docs/resources/wiki

Evaluation

Last mile HCL for documentation

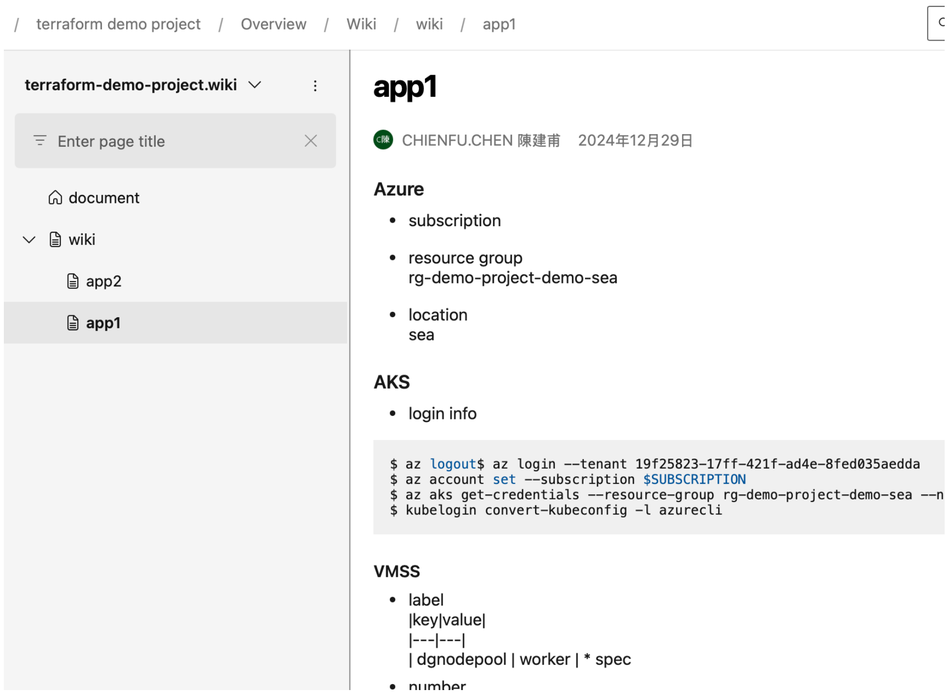
- IaC => documentation
- Azure devops terraform provider
- The wiki page dose not existed?
  - Let’s see what I can do.



Evaluation

What can I do?

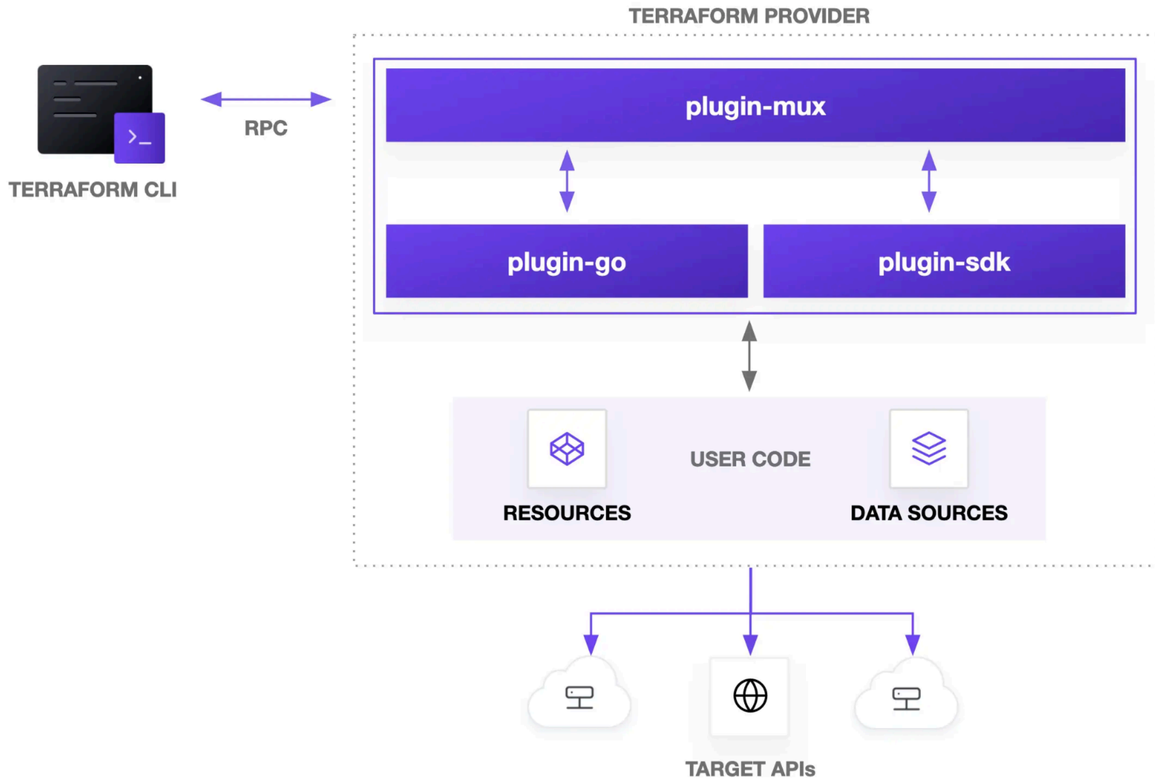
- There was no wiki page resource at the moment
  - Submit an Issue to upstream: Wiki page



Evaluation

Azure devops Terraform provider development

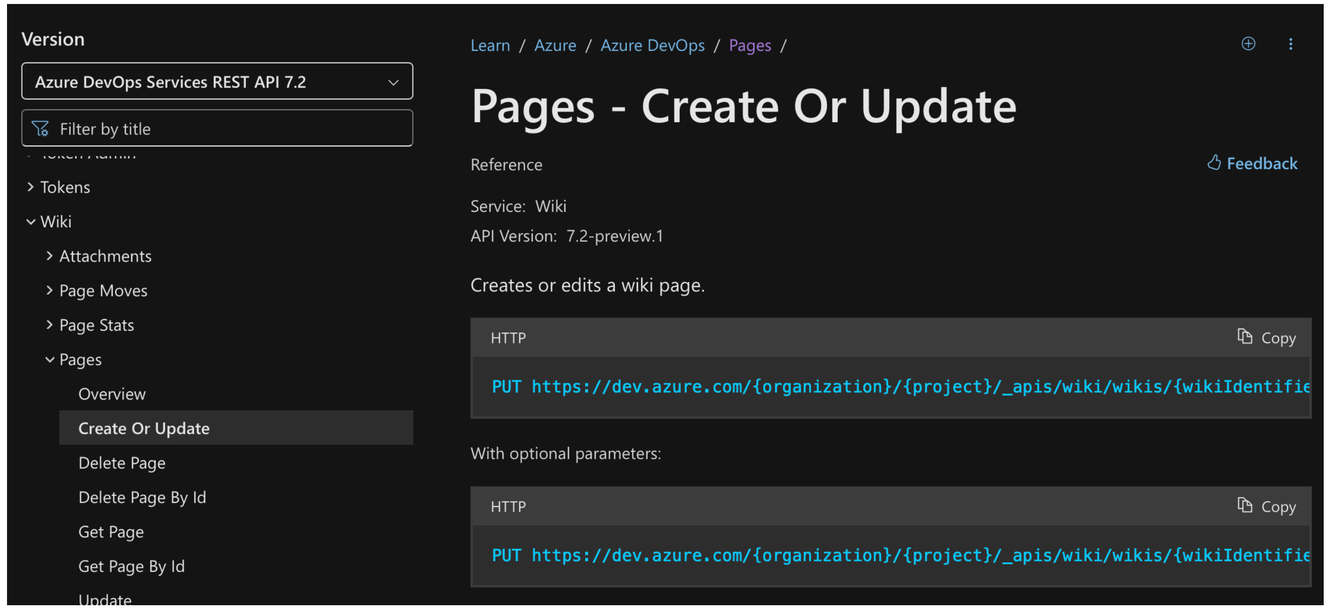
- How dose Terraform provider works?
- Provider Architecture
  - Resource
    - Schema
    - Operation: Read / Create / updat
    - Test
    - debug
    - Plugin sdk, Terraform core archite



Provider development: Study

azuredevops

- Azure resource manager terraform provider
- Azure devops terraform provider, doc
- Azure devops api doc
- Azure devops go sdk



Provider development: mock

resource Wiki page

azuredevops > internal > service > wiki > resource_wiki.go > ResourceWiki

```go
package wiki

import (
    "fmt"
    "time"

    "github.com/google/uuid"
    "github.com/hashicorp/terraform-plugin-sdk/v2/helper/schema"
    "github.com/hashicorp/terraform-plugin-sdk/v2/helper/validation"
    "github.com/microsoft/azure-devops-go-api/azuredevops/v7/git"
    "github.com/microsoft/azure-devops-go-api/azuredevops/v7/wiki"
    "github.com/microsoft/terraform-provider-azuredevops/azuredevops/internal/client"
    "github.com/microsoft/terraform-provider-azuredevops/azuredevops/internal/utils/converter"
)

func ResourceWiki() *schema.Resource {
    return &schema.Resource{
        Create: resourceWikiCreate,
        Read:   resourceWikiRead,
        Update: resourceWikiUpdate,
        Delete: resourceWikiDelete,
        Timeouts: &schema.ResourceTimeout{…
        },
        Importer: &schema.ResourceImporter{…
        },
        Schema: map[string]*schema.Schema{
            "name": {
                Type:     schema.TypeString,
                Required: true,
            },
            "type": {
                Type:     schema.TypeString,
                Required: true,
                ValidateFunc: validation.StringInSlice([]string{
                    string(wiki.WikiTypeValues.ProjectWiki),
                    string(wiki.WikiTypeValues.CodeWiki)},
                    false),
            },
            "project_id": {
                Type:     schema.TypeString,
                Optional: true,
            },
            "mapped_path": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "repository_id": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "version": {
```

azuredevops > internal > service > wiki > resource_wiki.go > ResourceWiki

```go
func ResourceWiki() *schema.Resource {
            },
            "type": {
                Type:     schema.TypeString,
                Required: true,
                ValidateFunc: validation.StringInSlice([]string{
                    string(wiki.WikiTypeValues.ProjectWiki),
                    string(wiki.WikiTypeValues.CodeWiki)},
                    false),
            },
            "project_id": {
                Type:     schema.TypeString,
                Optional: true,
            },
            "mapped_path": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "repository_id": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "version": {
                Type:     schema.TypeString,
                Optional: true,
                Computed: true,
            },
            "remote_url": {
                Type:     schema.TypeString,
                Computed: true,
            },
            "url": {
                Type:     schema.TypeString,
                Computed: true,
            },
        },
    }
}

func resourceWikiCreate(d *schema.ResourceData, m interface{}) error {…
}

func resourceWikiRead(d *schema.ResourceData, m interface{}) error {…
}

func resourceWikiUpdate(d *schema.ResourceData, m interface{}) error {…
}

func resourceWikiDelete(d *schema.ResourceData, m interface{}) error {
    clients := m.(*client.AggregatedClient)
```

azuredevops

Overview

AZUREDEVOPS DOCUMENTATION

Filter

- azuredevops_serviceendpoint_sonarqube
- azuredevops_serviceendpoint_ssh
- azuredevops_serviceendpoint_visualstudiomarketplace
- azuredevops_servicehook_permissions
- azuredevops_servicehook_storage_queue_pipelines
- azuredevops_tagging_permissions
- azuredevops_team
- azuredevops_team_administrators
- azuredevops_team_members
- azuredevops_user_entitlement
- azuredevops_variable_group
- azuredevops_variable_group_permissions
- azuredevops_wiki
- azuredevops_wiki_page
- azuredevops_workitem

azuredevops_wiki_page

Manages Wiki pages within Azure DevOps project.

Example Usage

Copy

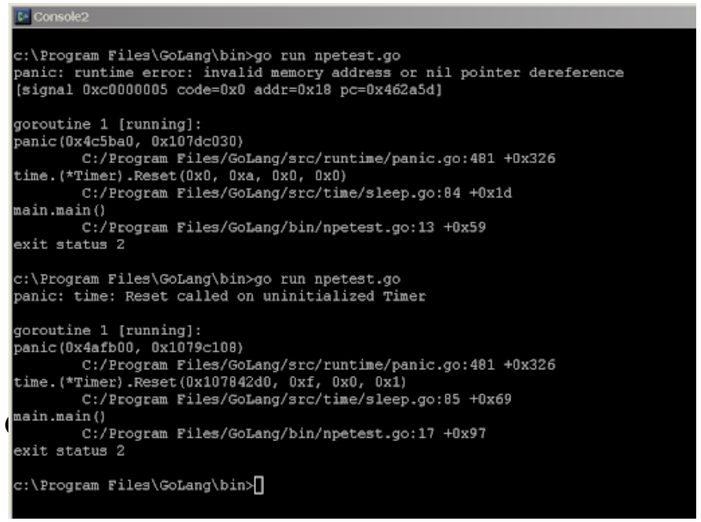
```
resource "azuredevops_project" "example" {
  name        = "Example Project"
  description = "Managed by Terraform"
}

resource "azuredevops_wiki" "example" {
  name       = "Example project wiki "
  project_id = azuredevops_project.example.id
  type       = "projectWiki"
}

resource "azuredevops_wiki_page" "example" {
  project_id = azuredevops_project.example.id
  wiki_id = azuredevops_wiki.example.id
  path = "/page"
  content = "content"
}
```

- Try to implement Terraform Resource and Document

Provider development: Debug

Why dose my code crash?

- HTTP 429? Null pointer?
- Keep calm and read the document
- Logging, Debug tools: doc1, doc2



Scenario 3: Implement a new resource or data source

If you need to implement a new resource or data source, you should first review a few implementations in order to u in the codebase, which generally match the way that other Terraform providers are built.

This scenario is a mix of *Scenario 1* and *Scenario 2*. However, after implementing the schema and CRUD operations your newly created resource or data source. The code for doing this is located in `provider.go`.

azuredevops > provider.go > Provider

```
10      ResourcesMap: map[string]*schema.Resource{
11          "azuredevops_build_definition":     resourceBuildDefinition(),
12          "azuredevops_project":              resourceProject(),
13          "azuredevops_serviceendpoint":      resourceServiceEndpoint(),
14          "azuredevops_azure_git_repository": resourceAzureGitRepository(),
15      },
16      DataSourcesMap: map[string]*schema.Resource{
17          "azuredevops_group": dataGroup(),
18      },
```

In order to accelerate development and ensure a common structure of all components (resource, data source), the

```
terraform apply -input=false
```

and then in another shell, get the process id for your provider. In the following, our provider has `azuredevops` in its name. Substitute a portion of your provider's name here.

```
ps | grep azuredevops
 2666 ttys001    0:00.00 grep azuredevops
 2663 ttys007    0:00.08 /Users/mhop/.terraform.d/plugins/darwin_amd64/terraform-provider-azuredevops_v0.0.1
```

We see, from above, that there are two processes with `azuredevops` in their name. One is the `grep` command. We want the other, which has process id `2663`.

Now, working quickly, edit the `launch.json` file, replacing the `1234` with `2663`:

```
{
        "version": "0.2.0",
        "configurations": [
                {
                "name": "Attach",
                "type": "go",
                "request": "attach",
                "mode": "local",
                "processId": 2663    // Have now added the correct process id.
                }
        ]
}
```

Provider development: Test design

Test framework

- Test case

Acceptance tests

The majority of tests in the provider are acceptance tests - which provisions real resources in Azure Devops and Azure. To run any acceptance tests you need to set `AZDO_ORG_SERVICE_URL`, `AZDO_PERSONAL_ACCESS_TOKEN` environment variables, some test have additional environment variables required to run. You can find out the required environment variables by running the test. Most of these variables can be set to dummy values.

The several options to run the tests are:

- Run the entire acceptance test suite

```
make testacc
```

- Run a subset using a prefix

```
make testacc TESTARGS='-run=TestAccBuildDefinitionBitbucket_Create' TESTTAGS='resource_build_definition'
```

- With VSCode Golang extension you can also run the tests using `run test`, `run package tests`, `run file tests` buttons above the test

```go
var projectName string
func basic(t *testing.T) {
    projectName := testutils.GenerateResourceName()
    tf := "azuredevops_wiki.test"
    resourceType := "azuredevops_wiki"
    resource.ParallelTest(t, resource.TestCase{
        PreCheck:     func() { testutils.PreCheck(t, nil) },
        Providers:    testutils.GetProviders(),
        CheckDestroy: checkWikiDestroyed(resourceType),
        Steps: []resource.TestStep{
            {
                Config: hclProjectWikiPageBasic(projectName),
                Check: resource.ComposeTestCheckFunc(
                    resource.TestCheckResourceAttrSet("azuredevops_wiki_page.test", "project_id"),
                    resource.TestCheckResourceAttrSet("azuredevops_wiki_page.test", "wiki_id"),
                    resource.TestCheckResourceAttrSet("azuredevops_wiki_page.test", "path"),
                    resource.TestCheckResourceAttrSet("azuredevops_wiki_page.test", "content"),
                ),
            },
            {
                ResourceName:      tf,
                ImportState:       true,
                ImportStateVerify: true,
            },
        },
    })
}
```

Provider development: Pull request

Feedback to the upstream

- Pull Request
  - Wiki
  - Wiki page

Demo

Terraform provider: azuredevops

- Let’s try azure devops example

Manages Wiki pages within Azure DevOps project.

Example Usage

Copy

```
resource "azuredevops_project" "example" {
  name        = "Example Project"
  description = "Managed by Terraform"
}

resource "azuredevops_wiki" "example" {
  name       = "Example project wiki "
  project_id = azuredevops_project.example.id
  type       = "projectWiki"
}

resource "azuredevops_wiki_page" "example" {
  project_id = azuredevops_project.example.id
  wiki_id = azuredevops_wiki.example.id
  path = "/page"
  content = "content"
}
```

QA