

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](#)



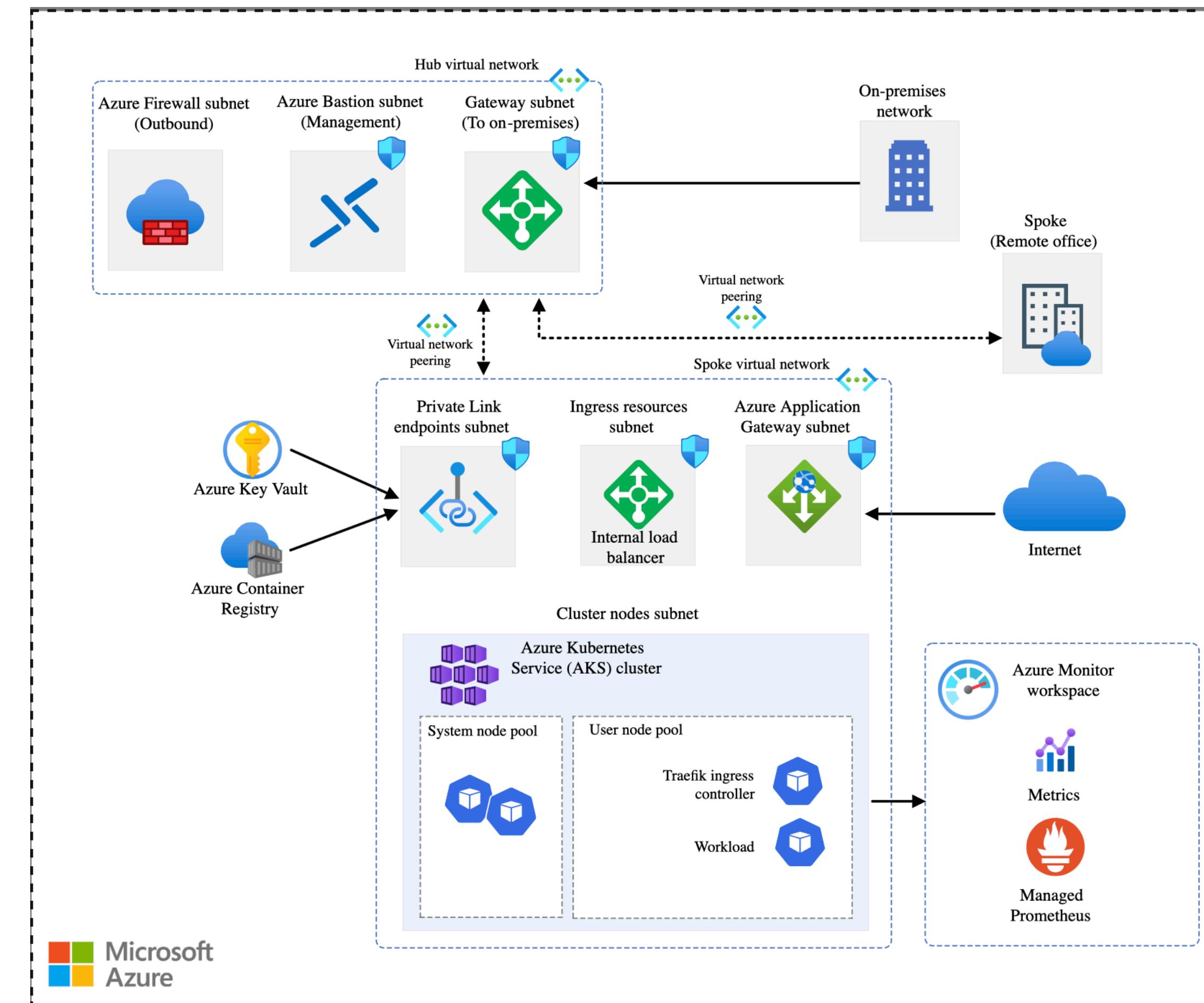
Outline

- 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

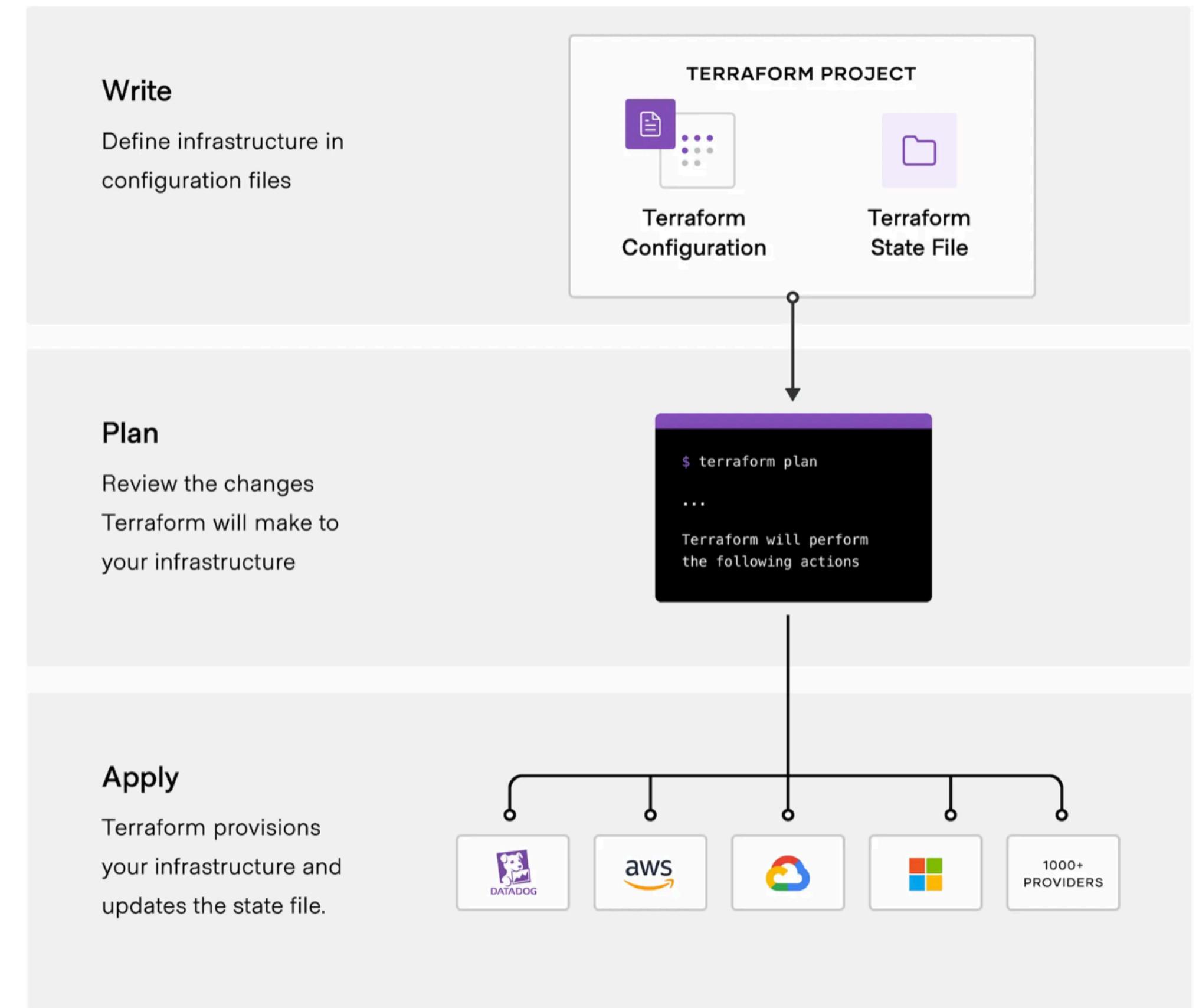
Infrastructure as code is needed

- 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

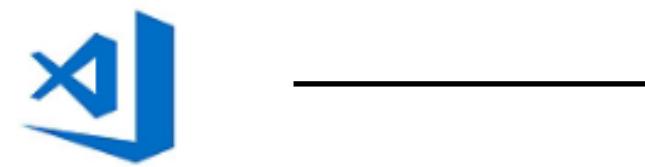
- Cloud Provider: Azure, AWS, GCP
- Write HCL
- Plan
- Apply
- Ref: [doc 1](#), [doc 2](#)



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
}
```



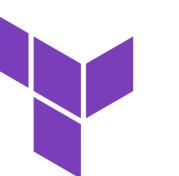
Git



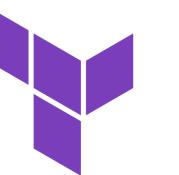
CI / CD pipeline



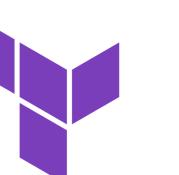
terraform test



terraform plan



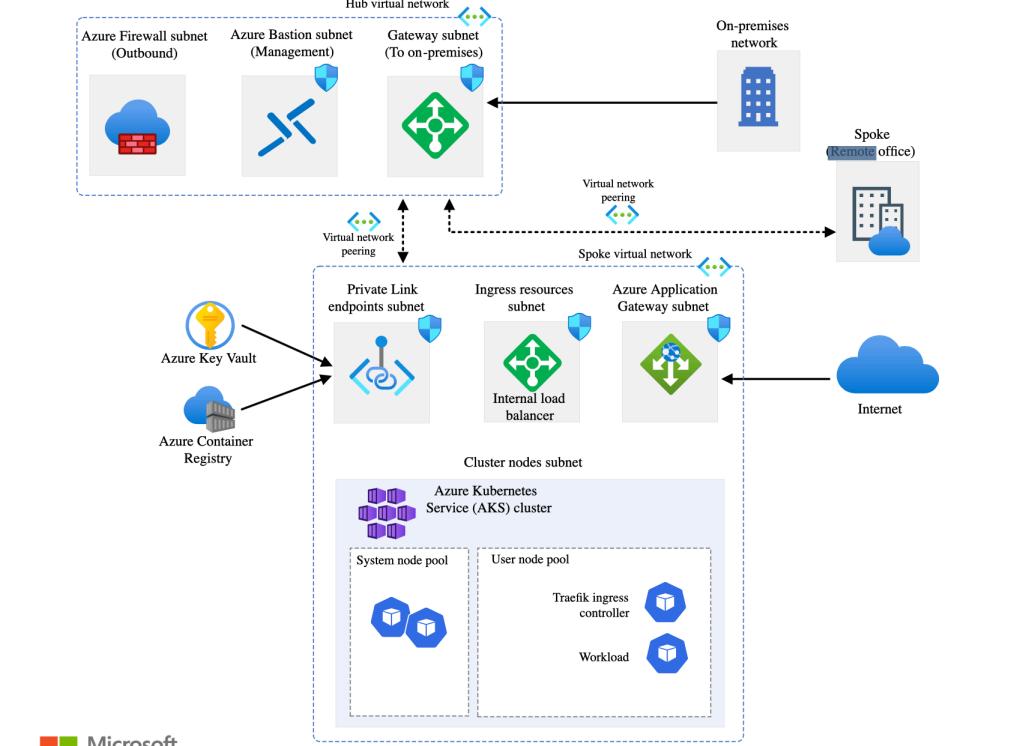
terraform apply



Integration test



generate document



Microsoft Azure

/ terraform demo project / Overview / Wiki / wiki / app1

app1

CHIENFU.CHEN 陳建甫 2024年12月29日

Azure

- subscription
- resource group rg-demo-project-demo-sea
- location sea

AKS

- login info

```
$ az logout$ az login --tenant 19f25823-17ff-421f-ad4e-8fed035aedd
$ az account set --subscription $SUBSCRIPTION
$ az aks get-credentials --resource-group rg-demo-project-demo-sea --n
$ kubectl convert-kubeconfig -l azurecli
```

VMS

- label |key|value|
|---|---|
| dnodepool | worker | * spec
- number

terrafarm-demo-project.wiki

Enter page title

document

wiki

app2

app1

Evaluation

Terraform Azureddevops provider Resource Wiki

azureddevops

The screenshot shows the Azure DevOps Documentation interface. On the left, there's a sidebar with a search bar and a list of provider names. The 'azureddevops_wiki' provider is highlighted with a blue dot. The main content area has a title 'azureddevops_wiki' and a brief description: 'Manages Wikis within Azure DevOps project.' Below this is a section titled 'Example Usage' containing Terraform code examples.

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

resource "azureddevops_git_repository" "example" {
    project_id = azureddevops_project.example.id
    name      = "Example Repository"
    initialization {
        init_type = "Clean"
    }
}

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

The screenshot shows the source code for the 'ResourceWiki' provider. It includes imports at the top, followed by a function definition for 'ResourceWiki'. The function takes parameters for Create, Read, Update, Delete, and Timeouts, along with an Importer and Schema. The Schema is defined as a map of string schemas, each with fields for name, type, and other properties like required and validation functions.

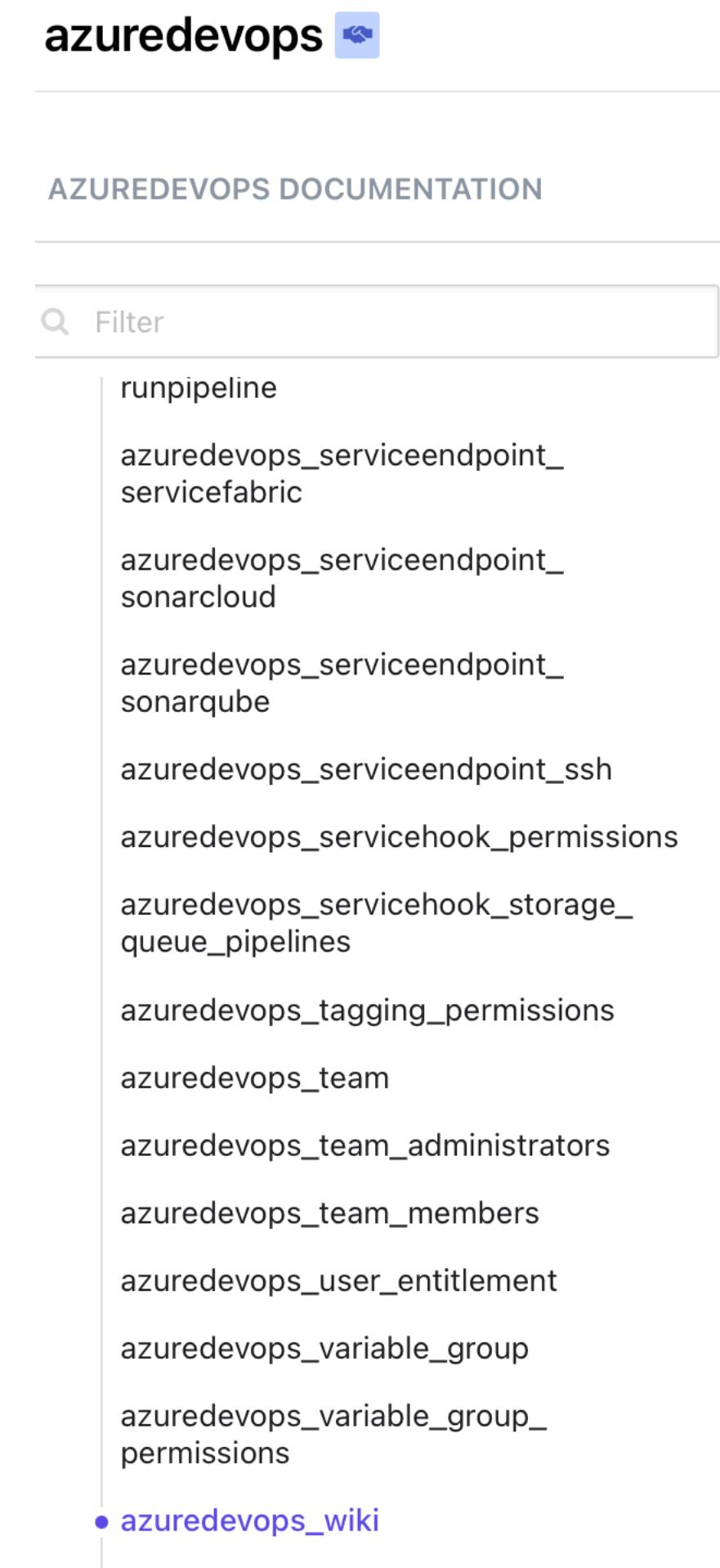
```
azureddevops > internal > service > wiki > resource_wiki.go > ResourceWiki
1 package wiki
2
3 import (
4     "fmt"
5     "time"
6
7     "github.com/google/uuid"
8     "github.com/hashicorp/terraform-plugin-sdk/v2/helper/schema"
9     "github.com/microsoft/terraform-provider-azureddevops/internal/client"
10    "github.com/microsoft/terraform-provider-azureddevops/internal/utils/converter"
11    "github.com/microsoft/terraform-provider-azureddevops/internal/validation"
12    "github.com/microsoft/terraform-provider-azureddevops/v7/git"
13    "github.com/microsoft/terraform-provider-azureddevops/v7/wiki"
14 )
15
16 func ResourceWiki() *schema.Resource {
17     return &schema.Resource{
18         Create: resourceWikiCreate,
19         Read:   resourceWikiRead,
20         Update: resourceWikiUpdate,
21         Delete: resourceWikiDelete,
22         Timeouts: &schema.ResourceTimeout{ ...
23     },
24     Importer: &schema.ResourceImporter{ ...
25 },
26     Schema: map[string]*schema.Schema{
27         "name": {
28             Type: schema.TypeString,
29             Required: true,
30         },
31         "type": {
32             Type: schema.TypeString,
33             Required: true,
34             ValidateFunc: validation.StringInSlice([]string{
35                 string(wiki.WikiTypeValues.ProjectWiki),
36                 string(wiki.WikiTypeValues.CodeWiki),
37                 false},
38             ),
39         },
40         "project_id": {
41             Type: schema.TypeString,
42             Optional: true,
43             Compute: true,
44         },
45         "mapped_path": {
46             Type: schema.TypeString,
47             Optional: true,
48             Compute: true,
49         },
50         "repository_id": {
51             Type: schema.TypeString,
52             Optional: true,
53             Compute: true,
54         },
55         "version": {
56             Type: schema.TypeString,
57             Optional: true,
58             Compute: true,
59         },
60         "remote_url": {
61             Type: schema.TypeString,
62             Compute: true,
63         },
64         "url": {
65             Type: schema.TypeString,
66             Compute: true,
67         },
68     },
69 },
70 },
71 },
72 },
73 }
74
75 > func resourceWikiCreate(d *schema.ResourceData, m interface{}) error { ...
76 }
77
78 > func resourceWikiRead(d *schema.ResourceData, m interface{}) error { ...
79 }
80
81 > func resourceWikiUpdate(d *schema.ResourceData, m interface{}) error { ...
82 }
83
84 > func resourceWikiDelete(d *schema.ResourceData, m interface{}) error {
85     clients := m..(*client.AggregatedClient)
86 }
```

- <https://registry.terraform.io/providers/microsoft/azureddevops/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.



azureddevops_wiki

Manages Wikis within Azure DevOps project.

Example Usage

```
resource "azureddevops_project" "example" {  
    name      = "Example Project"  
    description = "Managed by Terraform"  
}  
  
resource "azureddevops_git_repository" "example" {  
    project_id = azureddevops_project.example.id  
    name      = "Example Repository"  
    initialization {  
        init_type = "Clean"  
    }  
}  
  
resource "azureddevops_wiki" "example" {  
    name      = "Example project wiki "  
    project_id = azureddevops_project.example.id  
    type      = "projectWiki"  
}
```

Evaluation

What can I do?

- There was no wiki page resource at the moment
- Submit an issue to upstream: [Wiki page](#)

The screenshot shows a screenshot of a Terraform demo project's wiki page. The URL in the address bar is `/ terraform demo project / Overview / Wiki / wiki / app1`. The page title is `terraform-demo-project.wiki`. On the left, there is a sidebar with the following navigation:

- document
- wiki
 - app2
 - app1**

The main content area is titled **app1**. It displays the following information:

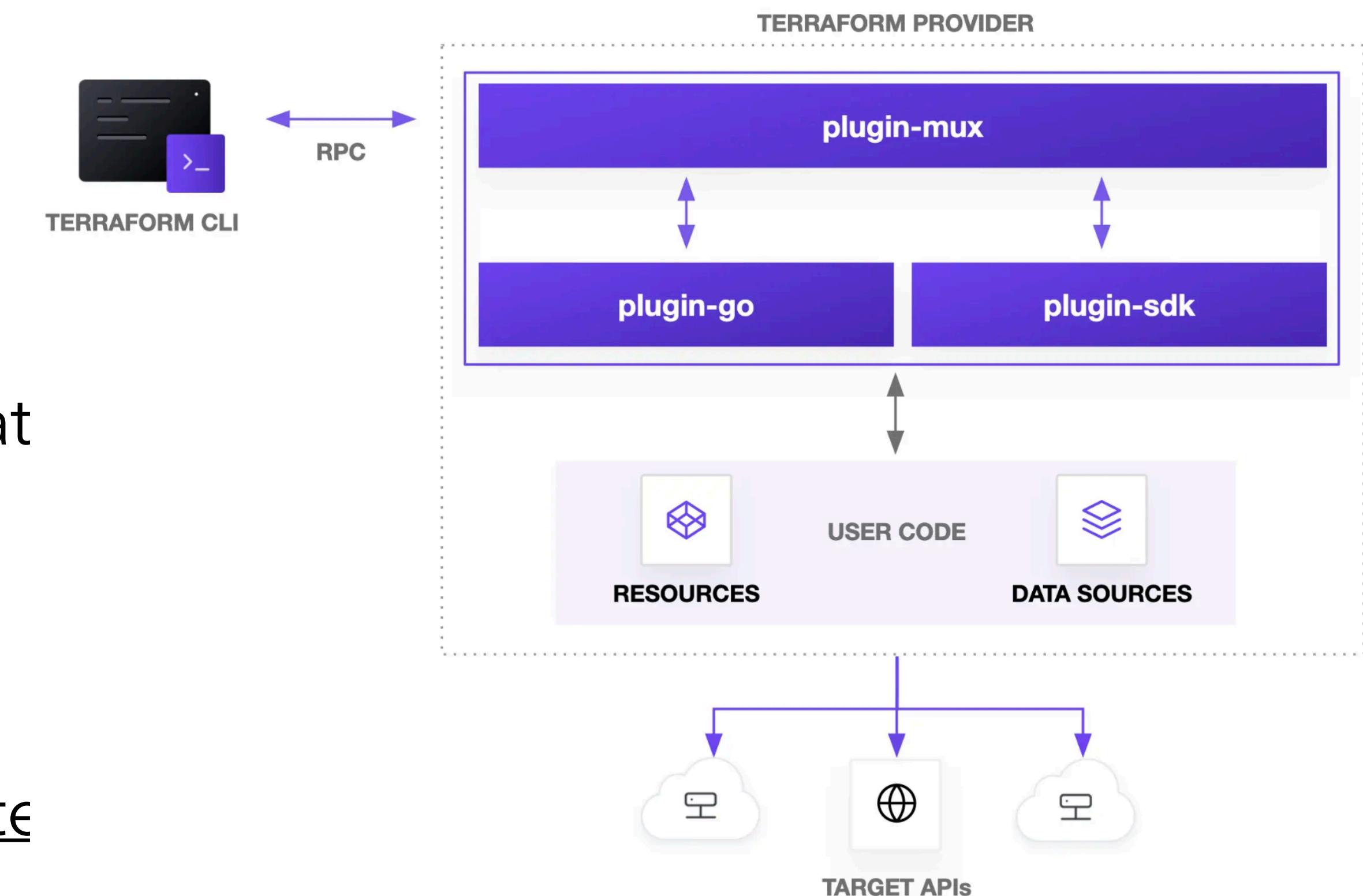
- CHIENFU.CHEN 陳建甫 2024年12月29日**
- Azure**
 - subscription
 - resource group `rg-demo-project-demo-sea`
 - location sea
- AKS**
 - login info
- VMSS**
 - label
 - |key|/value|
 - |---|---|
 - | dgnodepool | worker | * spec
 - number

```
$ az logout$ az login --tenant 19f25823-17ff-421f-ad4e-8fed035aedda
$ az account set --subscription $SUBSCRIPTION
$ az aks get-credentials --resource-group rg-demo-project-demo-sea --n
$ kubelogin convert-kubeconfig -l azurecli
```

Evaluation

Azure devops Terraform provider development

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



Provider development: Study

azureddevops

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

The screenshot shows a dark-themed API documentation page for the Azure DevOps REST API. At the top left, there's a dropdown for 'Version' set to 'Azure DevOps Services REST API 7.2'. Below it is a search bar with a magnifying glass icon and the placeholder 'Filter by title'. The main content area has a sidebar on the left with navigation links: 'Tokens', 'Wiki' (expanded), 'Attachments', 'Page Moves', 'Page Stats', 'Pages' (expanded), 'Overview', 'Create Or Update' (selected), 'Delete Page', 'Delete Page By Id', 'Get Page', 'Get Page By Id', and 'Update'. To the right of the sidebar, the page title is 'Pages - Create Or Update'. Above the main content, there's a breadcrumb trail: 'Learn / Azure / Azure DevOps / Pages /'. Below the title, it says 'Reference', 'Service: Wiki', and 'API Version: 7.2-preview.1'. A description follows: 'Creates or edits a wiki page.' Underneath, there are two examples of the HTTP PUT method:

```
PUT https://dev.azure.com/{organization}/{project}/_apis/wiki/wikis/{wikiIdentifier}/pages/{pageIdentifier}?api-version=7.2-preview.1
```

With optional parameters:

```
PUT https://dev.azure.com/{organization}/{project}/_apis/wiki/wikis/{wikiIdentifier}/pages/{pageIdentifier}{{?api-version}}&{parameters}
```

Provider development: mock resource Wiki page

The image displays two side-by-side code editors and a browser window. The left editor shows the implementation of a Terraform provider resource named `ResourceWiki`. The right editor shows the schema definition for the same resource. Below them, a browser window shows the generated API documentation for the `azureddevops_wiki_page` resource.

Code Editors:

- Left Editor:** Shows the implementation of the `ResourceWiki` provider. It includes imports for `fmt` and `time`, and defines methods for Create, Read, Update, Delete, and Importer. It also defines a schema for the `ResourceWiki` type, which includes fields for `name`, `type`, `project_id`, `mapped_path`, `repository_id`, and `version`.
- Right Editor:** Shows the schema definition for the `ResourceWiki` provider. It includes imports for `schema` and `validation`. The schema defines the `ResourceWiki` type with fields for `name`, `type`, `project_id`, `mapped_path`, `repository_id`, and `version`. It also includes validation logic for the `type` field.

Browser Window:

- Title:** azureddevops
- Section:** AZUREDEVOPS DOCUMENTATION
- Resource:** azureddevops_wiki_page
- Description:** Manages Wiki pages within Azure DevOps project.
- Example Usage:**

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

resource "azureddevops_wiki" "example" {
  name      = "Example project wiki"
  project_id = azureddevops_project.example.id
  type      = "projectWiki"
}

resource "azureddevops_wiki_page" "example" {
  project_id = azureddevops_project.example.id
  wiki_id   = azureddevops_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](#)

```
Console2

c:\Program Files\GoLang\bin>go run nptest.go
panic: runtime error: invalid memory address or nil pointer dereference
[signal 0xc0000005 code=0x0 addr=0x18 pc=0x462a5d]

goroutine 1 [running]:
panic(0x4c5ba0, 0x107dc030)
    C:/Program Files/GoLang/src/runtime/panic.go:481 +0x326
time.(*Timer).Reset(0x0, 0xa, 0x0, 0x0)
    C:/Program Files/GoLang/src/time/sleep.go:84 +0x1d
main.main()
    C:/Program Files/GoLang/bin/nptest.go:13 +0x59
exit status 2

c:\Program Files\GoLang\bin>go run nptest.go
panic: time: Reset called on uninitialized Timer

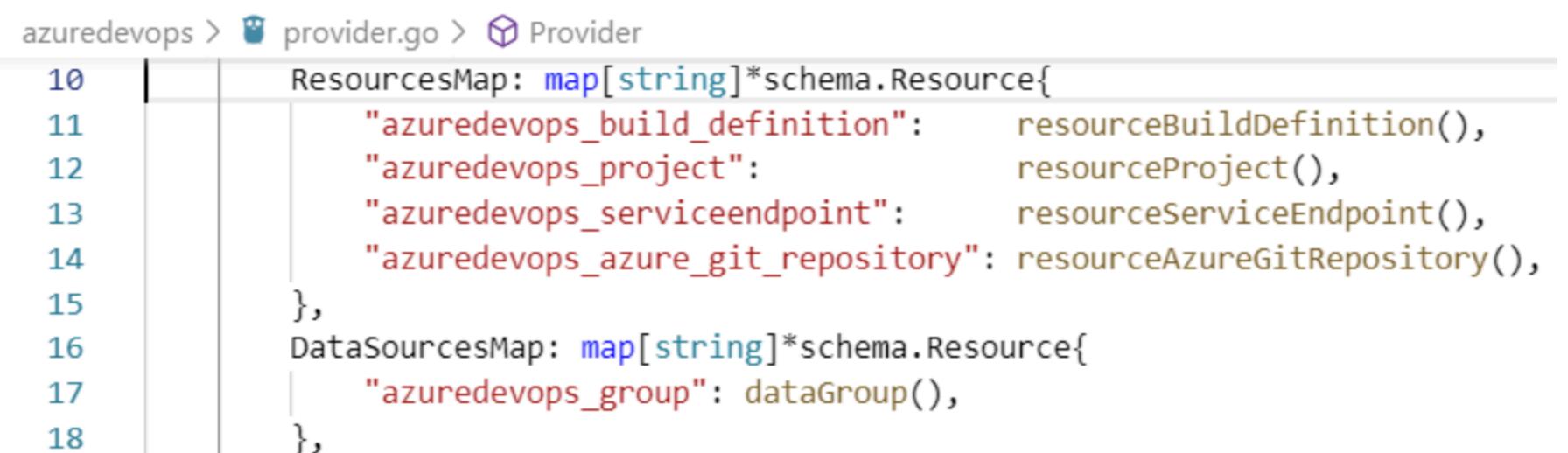
goroutine 1 [running]:
panic(0x4afb00, 0x1079c108)
    C:/Program Files/GoLang/src/runtime/panic.go:481 +0x326
time.(*Timer).Reset(0x107842d0, 0xf, 0x0, 0x1)
    C:/Program Files/GoLang/src/time/sleep.go:85 +0x69
main.main()
    C:/Program Files/GoLang/bin/nptest.go:17 +0x97
exit status 2

c:\Program Files\GoLang\bin>
```

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 understand how they are built. These are located in the codebase, which generally match the way that other Terraform providers are built.

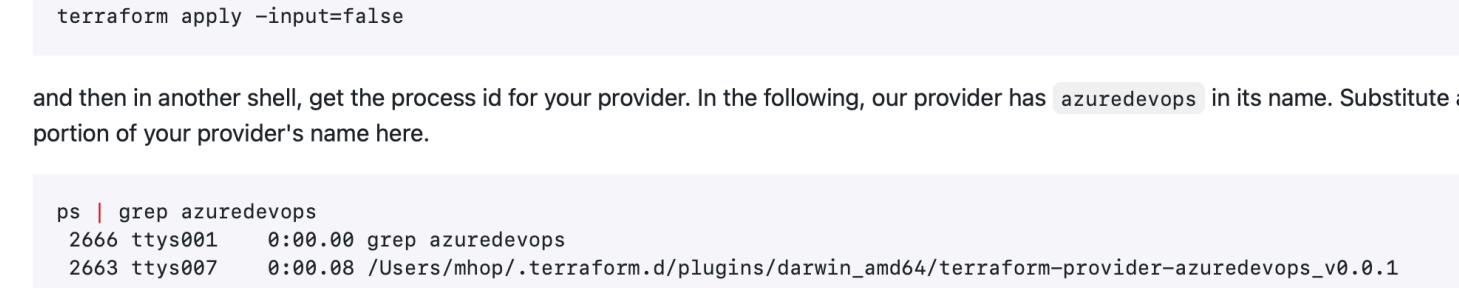
This scenario is a mix of Scenario 1 and Scenario 2. However, after implementing the schema and CRUD operations, you will have a working provider for your newly created resource or data source. The code for doing this is located in `provider.go`.



```
azuredvops > provider.go > Provider
10     ResourcesMap: map[string]*schema.Resource{
11         "azuredvops_build_definition":   resourceBuildDefinition(),
12         "azuredvops_project":           resourceProject(),
13         "azuredvops_serviceendpoint":   resourceServiceEndpoint(),
14         "azuredvops_azure_git_repository": resourceAzureGitRepository(),
15     },
16     DataSourcesMap: map[string]*schema.Resource{
17         "azuredvops_group": dataGroup(),
18     },
```

In order to accelerate development and ensure a common structure of all components (resource, data source), the provider has a standard structure. This includes a `ResourcesMap` and a `DataSourcesMap`. The `ResourcesMap` contains resources like `azuredvops_build_definition`, `azuredvops_project`, `azuredvops_serviceendpoint`, and `azuredvops_azure_git_repository`. The `DataSourcesMap` contains a single resource named `azuredvops_group`.

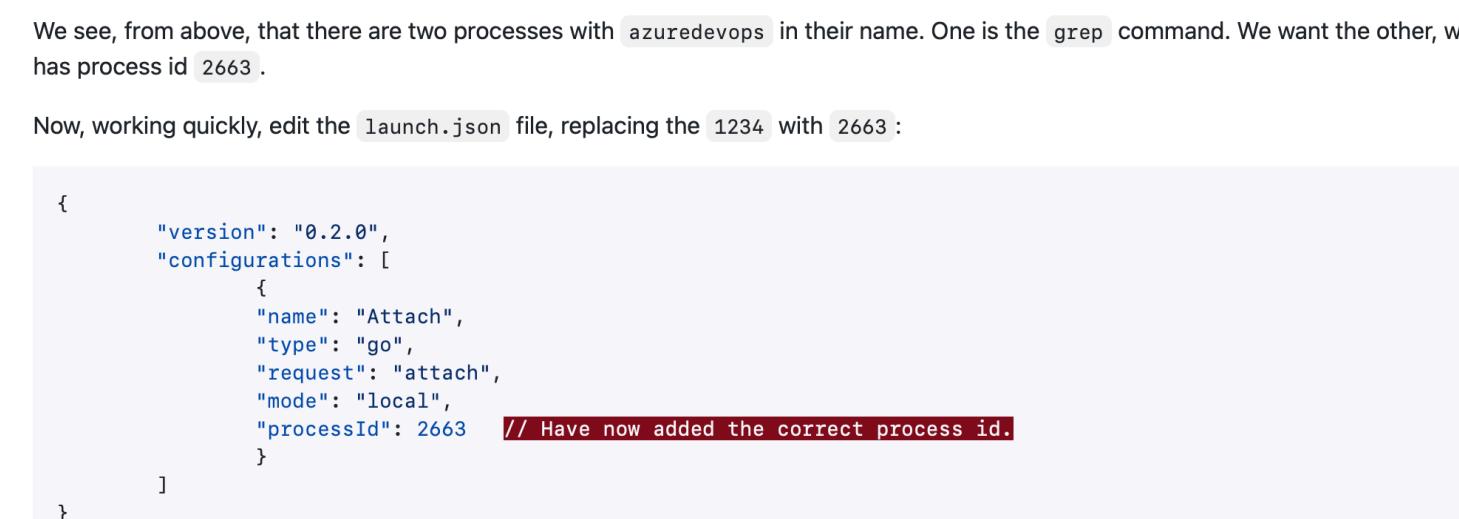
and then in another shell, get the process id for your provider. In the following, our provider has `azuredvops` in its name. Substitute a portion of your provider's name here.



```
ps | grep azuredvops
2666 ttys001  0:00.00 grep azuredvops
2663 ttys007  0:00.08 /Users/mhop/.terraform.d/plugins/darwin_amd64/terraform-provider-azuredvops_v0.0.1
```

We see, from above, that there are two processes with `azuredvops` 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.
    }
  ]
}
```

net/http.ResponseWriter(*net/http.ResponseWriter) (at line 13)

```
6 |     "sync"
7 |     "time"
8 |
9 |     "gola"
10 |     > : net/http.ResponseWriter {conn: *net/
11 |     |
12 |     func gree Hold Option key to switch to editor
13 |     fmt.Fprintf(w, "%s %v", stringutil.Reverse(
14 |     )
15 | }
```

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

```
run test func var projectName string basic(t *testing.T) {
    projectName := testutils.GenerateResourceName()
    tf := "azureddevops_wiki.test"
    resourceType := "azureddevops_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("azureddevops_wiki_page.test", "project_id"),
                    resource.TestCheckResourceAttrSet("azureddevops_wiki_page.test", "wiki_id"),
                    resource.TestCheckResourceAttrSet("azureddevops_wiki_page.test", "path"),
                    resource.TestCheckResourceAttrSet("azureddevops_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: azureddevops

- Let's try azure devops example

Manages Wiki pages within Azure DevOps project.

Example Usage

```
resource "azureddevops_project" "example" {  
    name      = "Example Project"  
    description = "Managed by Terraform"  
}  
  
resource "azureddevops_wiki" "example" {  
    name      = "Example project wiki "  
    project_id = azureddevops_project.example.id  
    type      = "projectWiki"  
}  
  
resource "azureddevops_wiki_page" "example" {  
    project_id = azureddevops_project.example.id  
    wiki_id = azureddevops_wiki.example.id  
    path = "/page"  
    content = "content"  
}
```

Copy

OA