

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](#)



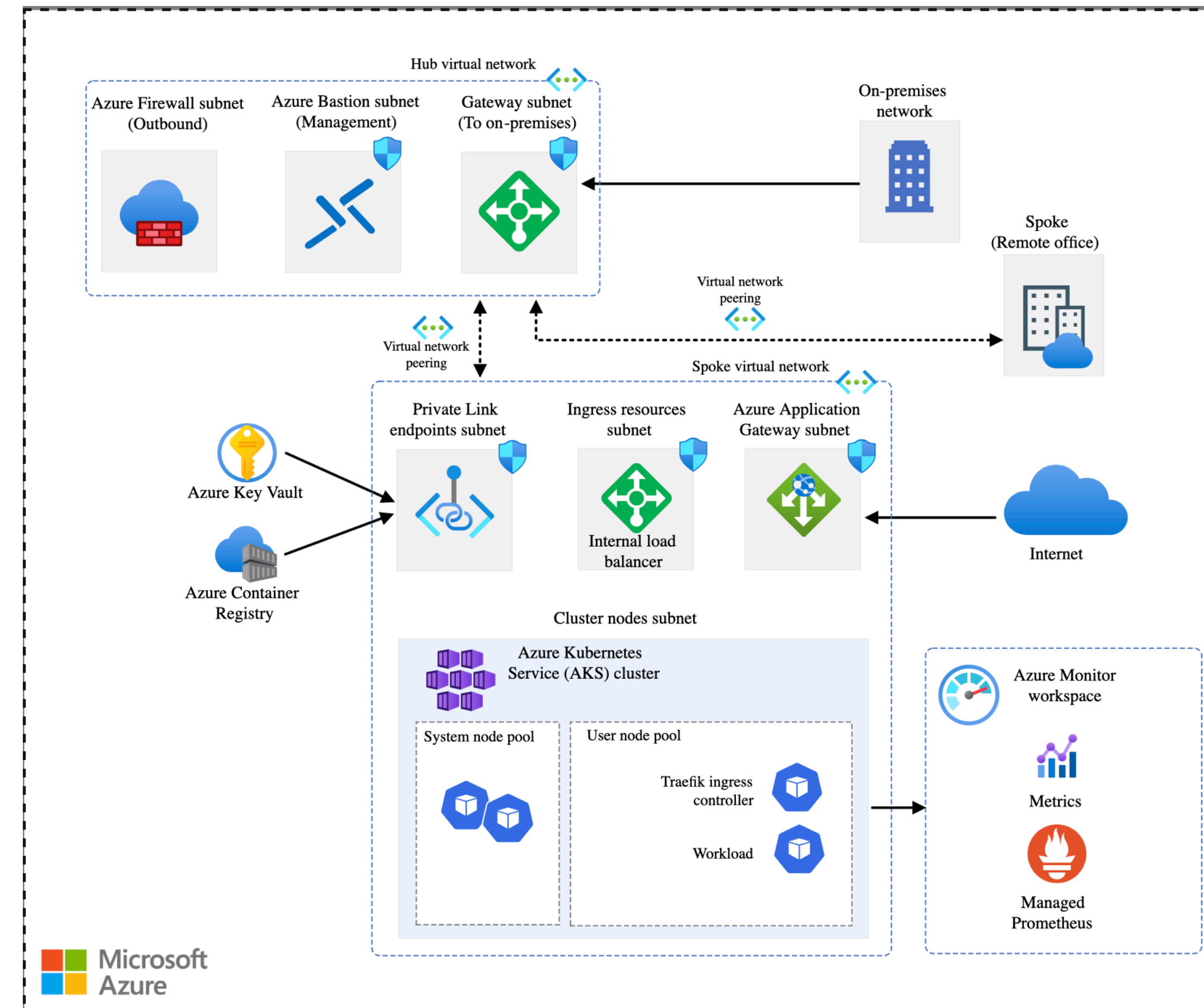
Outline

- 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

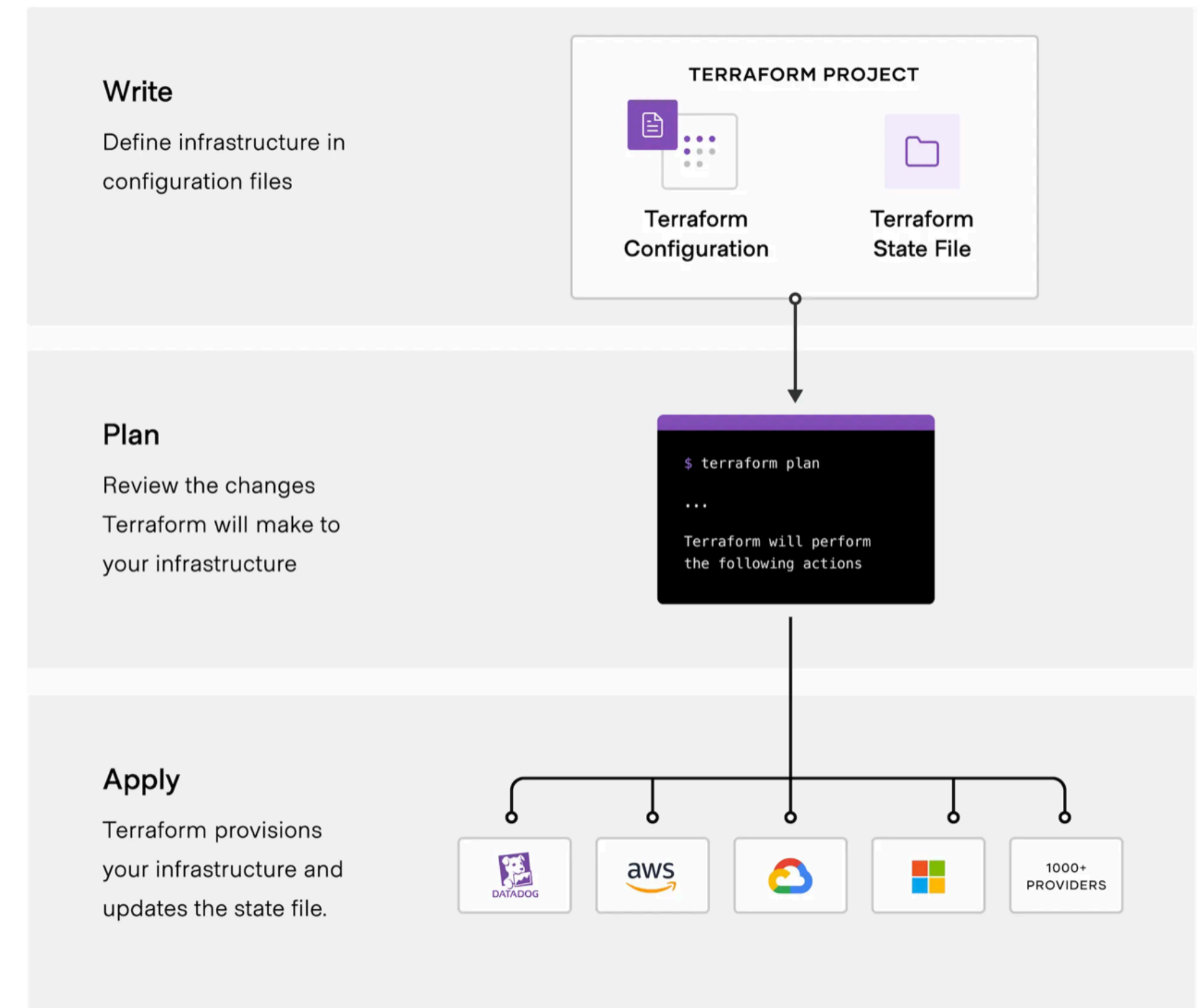
Infrastructure as code is needed

- 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.vms.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



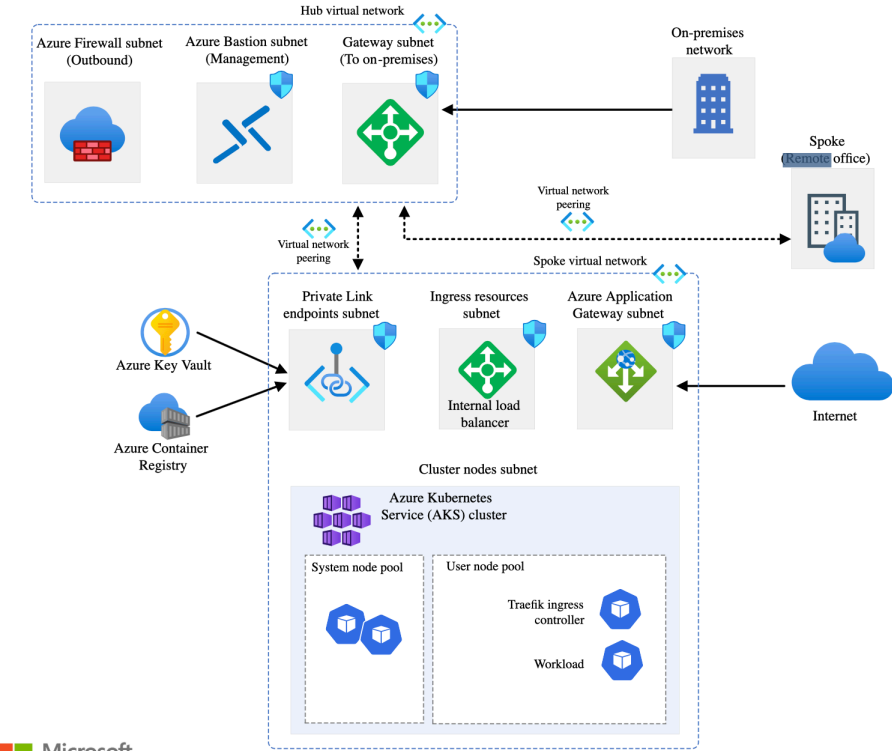
terraform apply



Integration test



generate document



Microsoft Azure

terraform demo project / Overview / Wiki / wiki / app1

terraform-demo-project.wiki

Enter page title

document

wiki

app2

app1

app1

CHIENTU.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
$ kubelogin convert-kubeconfig -l azurecli
```

VMSS

label

[key|value]

[---|---]

| dgnodepool | worker | * spec

number

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
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/hashicorp/terraform-plugin-sdk/v2/helper/validation"
10    "github.com/microsoft/azure-devops-go-api/azuredevops/v7/git"
11    "github.com/microsoft/azure-devops-go-api/azuredevops/v7/wiki"
12    "github.com/microsoft/terraform-provider-azuredevops/azuredevops/internal/client"
13    "github.com/microsoft/terraform-provider-azuredevops/azuredevops/internal/utils/converter"
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             Create: schema.DefaultTimeout(10 * time.Minute),
24             Update: schema.DefaultTimeout(10 * time.Minute),
25             Delete: schema.DefaultTimeout(10 * time.Minute),
26         },
27         Importer: &schema.ResourceImporter{
28             StateContext: resourceWikiImport,
29         },
30         Schema: map[string]*schema.Schema{
31             "name": {
32                 Type:     schema.TypeString,
33                 Required: true,
34             },
35             "type": {
36                 Type:     schema.TypeString,
37                 Required: true,
38                 ValidateFunc: validation.StringInSlice([]string{
39                     string(wiki.WikiTypeValues.ProjectWiki),
40                     string(wiki.WikiTypeValues.CodeWiki),
41                 }, false),
42             },
43             "project_id": {
44                 Type:     schema.TypeString,
45                 Optional: true,
46             },
47             "mapped_path": {
48                 Type:     schema.TypeString,
49                 Optional: true,
50                 Computed: true,
51             },
52             "repository_id": {
53                 Type:     schema.TypeString,
54                 Optional: true,
55                 Computed: true,
56             },
57             "version": {
58                 Type:     schema.TypeString,
59                 Optional: true,
60                 Computed: true,
61             },
62             "remote_url": {
63                 Type:     schema.TypeString,
64                 Computed: true,
65             },
66             "url": {
67                 Type:     schema.TypeString,
68                 Computed: true,
69             },
70         },
71     }
72 }
73
74 > func resourceWikiCreate(d *schema.ResourceData, m interface{}) error {
75     client := m.(*client.AggregatedClient)
76     projectID := d.Get("project_id").(string)
77     name := d.Get("name").(string)
78     type := d.Get("type").(string)
79     mappedPath := d.Get("mapped_path").(string)
80     repositoryID := d.Get("repository_id").(string)
81     version := d.Get("version").(string)
82     remoteURL := d.Get("remote_url").(string)
83     url := d.Get("url").(string)
84     err := client.WikiClient.CreateWiki(projectID, name, type, mappedPath, repositoryID, version, remoteURL, url)
85     if err != nil {
86         return fmt.Errorf("Error creating wiki: %v", err)
87     }
88     d.SetId(fmt.Sprintf("%s-%s", projectID, name))
89     return nil
90 }
91
92 > func resourceWikiRead(d *schema.ResourceData, m interface{}) error {
93     client := m.(*client.AggregatedClient)
94     projectID, name, err := client.WikiClient.GetWiki(d.Get("project_id").(string), d.Get("name").(string))
95     if err != nil {
96         return fmt.Errorf("Error reading wiki: %v", err)
97     }
98     d.Set("name", name)
99     d.Set("type", projectID)
100    d.Set("project_id", projectID)
101    d.Set("mapped_path", name)
102    d.Set("repository_id", name)
103    d.Set("version", name)
104    d.Set("remote_url", name)
105    d.Set("url", name)
106    return nil
107 }
108
109 > func resourceWikiUpdate(d *schema.ResourceData, m interface{}) error {
110     client := m.(*client.AggregatedClient)
111     projectID := d.Get("project_id").(string)
112     name := d.Get("name").(string)
113     type := d.Get("type").(string)
114     mappedPath := d.Get("mapped_path").(string)
115     repositoryID := d.Get("repository_id").(string)
116     version := d.Get("version").(string)
117     remoteURL := d.Get("remote_url").(string)
118     url := d.Get("url").(string)
119     err := client.WikiClient.UpdateWiki(projectID, name, type, mappedPath, repositoryID, version, remoteURL, url)
120     if err != nil {
121         return fmt.Errorf("Error updating wiki: %v", err)
122     }
123     d.SetId(fmt.Sprintf("%s-%s", projectID, name))
124     return nil
125 }
126
127 > func resourceWikiDelete(d *schema.ResourceData, m interface{}) error {
128     client := m.(*client.AggregatedClient)
129     projectID := d.Get("project_id").(string)
130     name := d.Get("name").(string)
131     err := client.WikiClient.DeleteWiki(projectID, name)
132     if err != nil {
133         return fmt.Errorf("Error deleting wiki: %v", err)
134     }
135     d.SetId("")
136     return nil
137 }
```

- <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.

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"
}
```

Evaluation

What can I do?

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

terraform demo project / Overview / Wiki / wiki / app1

terraform-demo-project.wiki

Enter page title

document

wiki

app2

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-8fed035aedda$ az account set --subscription $SUBSCRIPTION$ az aks get-credentials --resource-group rg-demo-project-demo-sea --n$ kubelogin convert-kubeconfig -l azurecli
```

VMSS

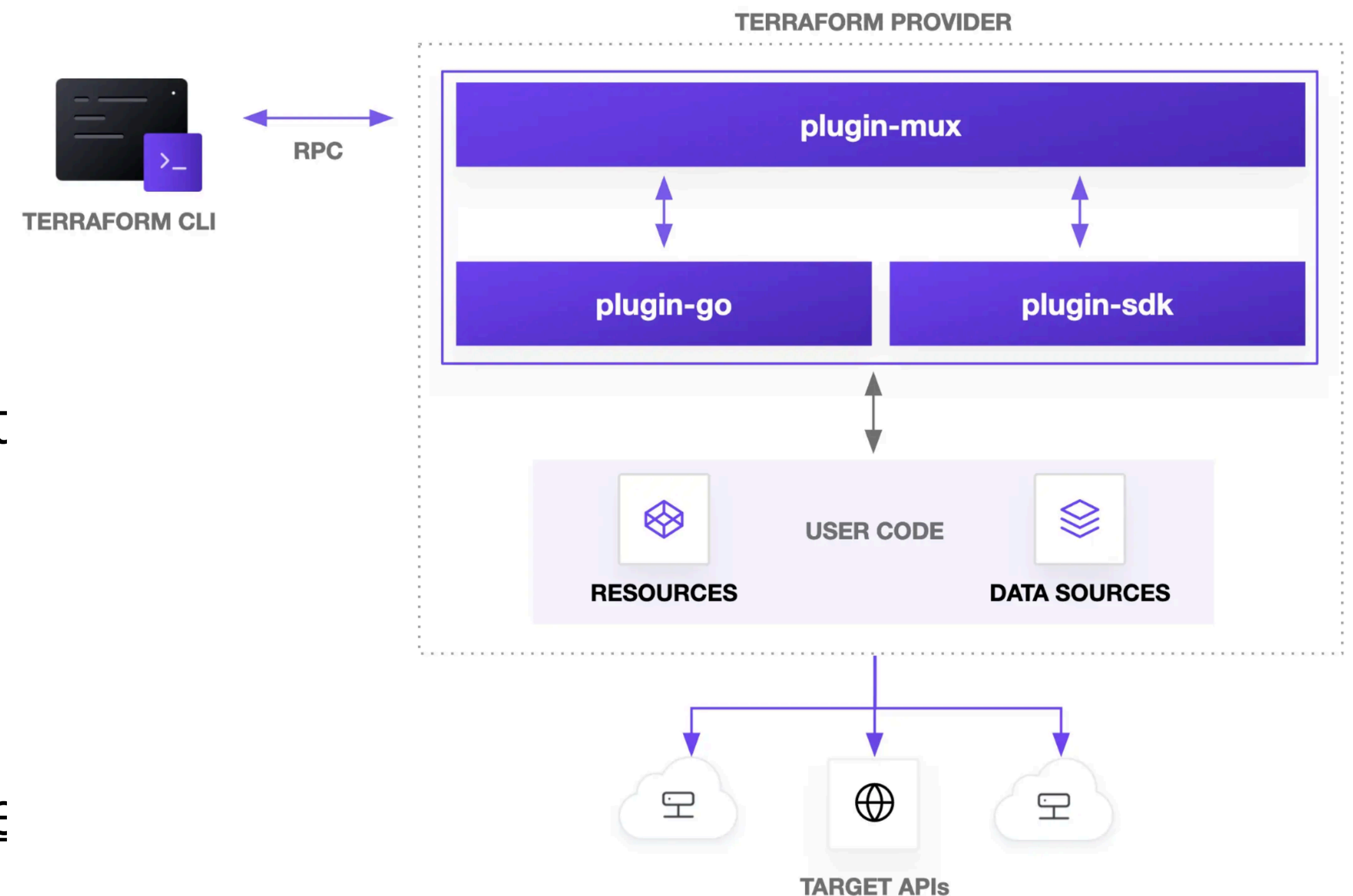
label
key	value
dgonodepool	worker

number

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

The screenshot displays the Azure DevOps REST API documentation for the 'Pages - Create Or Update' endpoint. On the left, a sidebar shows the 'Version' dropdown set to 'Azure DevOps Services REST API 7.2' and a search filter 'Filter by title'. The sidebar menu includes sections like 'Tokens', 'Wiki' (with sub-items: Attachments, Page Moves, Page Stats), and 'Pages' (with sub-items: Overview, 'Create Or Update' (highlighted), Delete Page, Delete Page By Id, Get Page, Get Page By Id, and Update). The main content area has a breadcrumb 'Learn / Azure / Azure DevOps / Pages /' and a 'Feedback' link. The title 'Pages - Create Or Update' is prominent. Below it, the 'Reference' section specifies 'Service: Wiki' and 'API Version: 7.2-preview.1'. A description states 'Creates or edits a wiki page.' Two HTTP request examples are shown, both using the 'PUT' method and the URL 'https://dev.azure.com/{organization}/{project}/_apis/wiki/wikis/{wikiIdentifie}'. Each example includes a 'Copy' button.

Provider development: mock resource Wiki page

```
azureddevops > internal > service > wiki > -o 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/hashicorp/terraform-plugin-sdk/v2/helper/validation"
10    "github.com/microsoft/azure-devops-go-api/azureddevops/v7/git"
11    "github.com/microsoft/azure-devops-go-api/azureddevops/v7/wiki"
12    "github.com/microsoft/terraform-provider-azureddevops/azureddevops/internal/client"
13    "github.com/microsoft/terraform-provider-azureddevops/azureddevops/internal/utils/converter"
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{~
27     },
28 >     Importer: &schema.ResourceImporter{~
30     },
31     Schema: map[string]*schema.Schema{
32         "name": {
33             Type:     schema.TypeString,
34             Required: true,
35         },
36         "type": {
37             Type:     schema.TypeString,
38             Required: true,
39             ValidateFunc: validation.StringInSlice([]string{
40                 string(wiki.WikiTypeValues.ProjectWiki),
41                 string(wiki.WikiTypeValues.CodeWiki)},
42                 false),
43         },
44         "project_id": {
45             Type:     schema.TypeString,
46             Optional: true,
47         },
48         "mapped_path": {
49             Type:     schema.TypeString,
50             Optional: true,
51             Computed: true,
52         },
53         "repository_id": {
54             Type:     schema.TypeString,
55             Optional: true,
56             Computed: true,
57         },
58         "version": {
59             Type:     schema.TypeString,
60             Optional: true,
61             Computed: true,
62         },
63         "remote_url": {
64             Type:     schema.TypeString,
65             Computed: true,
66         },
67         "url": {
68             Type:     schema.TypeString,
69             Computed: true,
70         },
71     },
72 },
73 }
74
75 > func resourceWikiCreate(d *schema.ResourceData, m interface{}) error {~
112 }
113
114 > func resourceWikiRead(d *schema.ResourceData, m interface{}) error {~
151 }
152
153 > func resourceWikiUpdate(d *schema.ResourceData, m interface{}) error {~
169 }
170
171 func resourceWikiDelete(d *schema.ResourceData, m interface{}) error {
172     clients := m.(*client.AggregatedClient)
173 }
```

azureddevops

AZUREDEVOPS DOCUMENTATION

Filter

azureddevops_serviceendpoint_sonarqube

azureddevops_serviceendpoint_ssh

azureddevops_serviceendpoint_visualstudiomarketplace

azureddevops_servicehook_permissions

azureddevops_servicehook_storage_queue_pipelines

azureddevops_tagging_permissions

azureddevops_team

azureddevops_team_administrators

azureddevops_team_members

azureddevops_user_entitlement

azureddevops_variable_group

azureddevops_variable_group_permissions

azureddevops_wiki

• azureddevops_wiki_page

azureddevops_workitem

azureddevops_wiki_page

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

- 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 npetest.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/npetest.go:13 +0x59
exit status 2

c:\Program Files\GoLang\bin>go run npetest.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/npetest.go:17 +0x97
exit status 2

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

```
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.
    }
  ]
}
```

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

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

```
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"  
}
```

Copy

QA