

Integração OCI Resource Manager e GitHub

inLab – Hands-on

Click or tap to enter a date, Version 1.0
Copyright © 2022 , Oracle and/or its affiliates
Public

Objetivo

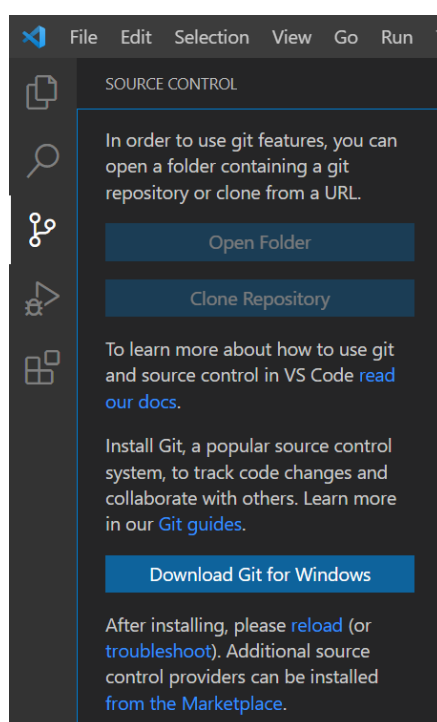
Nesse LAB iremos aprender as etapas necessárias para criar e distribuir código Terraform reutilizável de forma simples e prática, seguindo as seguintes etapas:

1. Configurar o OCI Resource Manager para utilizar o GitHub como um provedor de configuração
2. Criar nosso script terraform, publicar no GitHub e utilizar a integração para o deploy
3. Utilizar “Schema Documents” para padronizar a entrada de variáveis em nosso script Terraform
4. Utilizar o “GitHub Actions” para gerar uma nova release distribuível do nosso script
5. Possibilitar o deploy em 1 clique do nosso script em qualquer Tenancy do OCI

Pré-Requisitos

Para completar com sucesso as tarefas aqui listadas você precisa de:

1. Conta “**Always Free**” no OCI, caso ainda não tenha, clique [aqui](#) para criar.
2. Conta gratuita no GitHub, clique [aqui](#) caso ainda não tenha
3. O software **VSCODE**, ou outro software de sua preferência, para edição e publicação de nossos scripts no GitHub (Clique [aqui](#) para baixar o VSCODE)
4. Com o **VSCODE** instalado, abra-o e clique em “**Controle de Código-Fonte**” nos ícones à esquerda, faça o download e instale o **Git for Windows**. Completada a instalação, clique em “**Recarregar**”



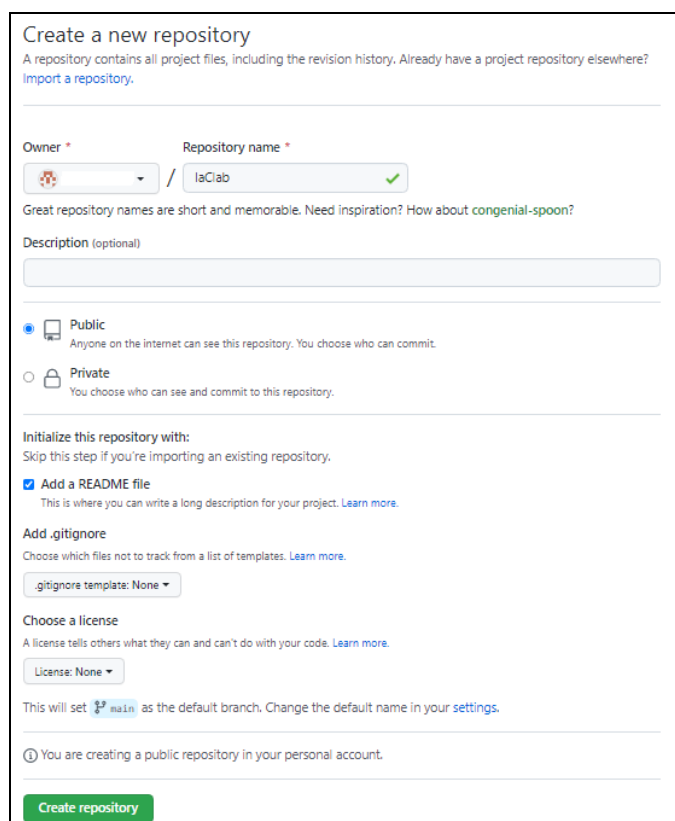
Criando nosso projeto no GitHub e acessando pelo VSCODE

Nessa etapa iremos fazer o setup do nosso ambiente, criando um repositório novo no GitHub para usarmos no nosso projeto e configurando o VSCODE para acessar esse repositório.

1. No seu navegador favorito, acesse o [GitHub](#), e faça login com as suas credenciais.
2. Na página inicial vamos criar um repositório para o nosso LAB, para isso clique em **“Create Repository”**

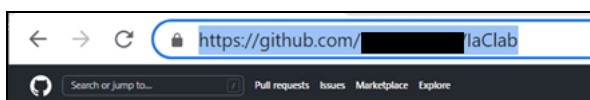
Create repository

3. Dê um nome ao seu novo repositório, selecione a opção **“Add a README file”** e clique em **“Create Repository”**

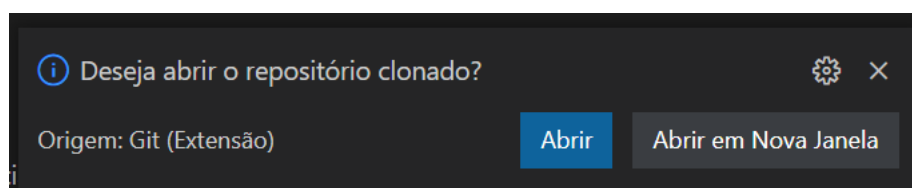


The screenshot shows the GitHub 'Create a new repository' page. It includes fields for 'Owner' (selected as 'laClab') and 'Repository name' (with a green checkmark). Below these is a 'Description (optional)' text area. The 'Public' radio button is selected under the 'visibility' section. In the 'Initialize this repository with:' section, the 'Add a README file' checkbox is checked. The 'Add .gitignore' section shows a dropdown set to 'None'. The 'Choose a license' section also shows a dropdown set to 'None'. At the bottom, a green 'Create repository' button is visible.

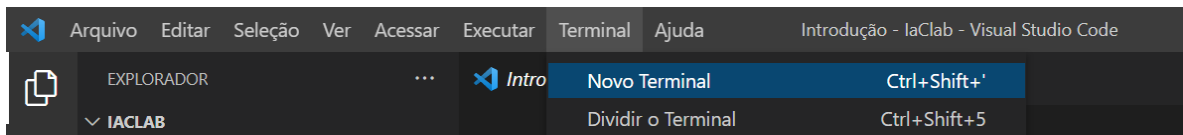
4. Com o repositório criado, copiamos a URL que aparece no Browser e voltamos ao VSCODE



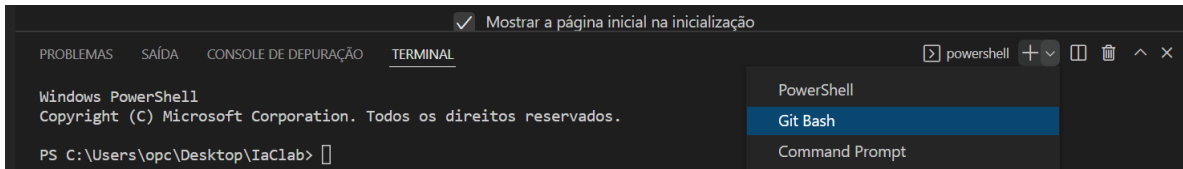
5. No VSCODE selecione novamente **“Controle de Código-Fonte”** nos ícones à esquerda, clique no botão **“Clone Repository”** e cole a URL que copiamos no passo anterior.
6. Selecione a pasta local em seu computador que os arquivos do projeto serão armazenados
7. Na mensagem que aparecerá no canto inferior direito da tela, clique em **“Abrir”**



8. Clicar no menu **Terminal** do VSCODE, e escolher novo terminal



9. Na janela do terminal, alterar para **GitBash**

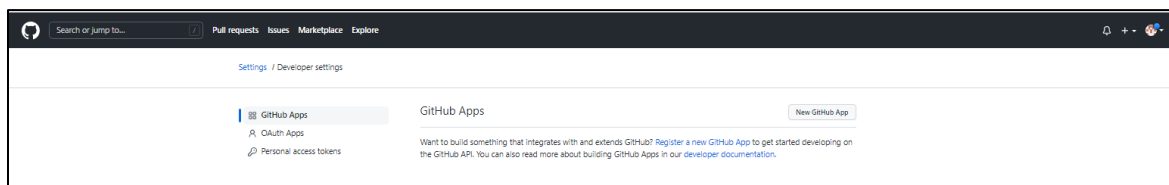


10. Com isso o nosso setup está pronto e o próximo passo é configurar o Resource Manager do OCI (Configuration Source Providers) para usar o GitHub como origem do código

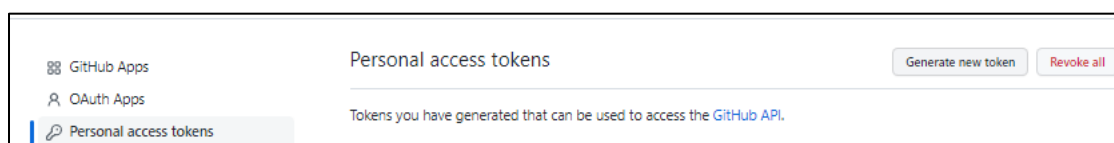
Configurando um “Configuration Source Provider” no OCI

A primeira tarefa é configurar a autenticação do OCI ao GitHub, e para isso vamos precisar gerar um Access Token no GitHub.

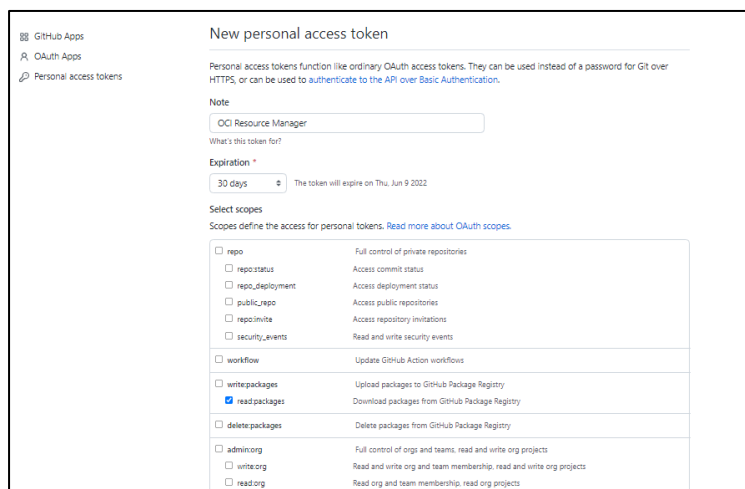
1. Na página do GitHub clicar no ícone de seu perfil, Settings, e na última opção do menu à esquerda clicar em “**Developer Settings**”, em seguida clicar em “**Personal access tokens**”.



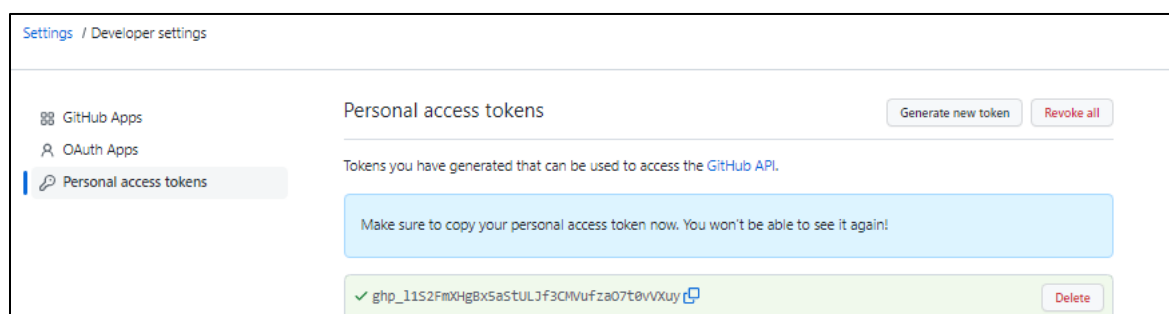
2. Na tela que abrirá clique no botão “**Generate new token**”



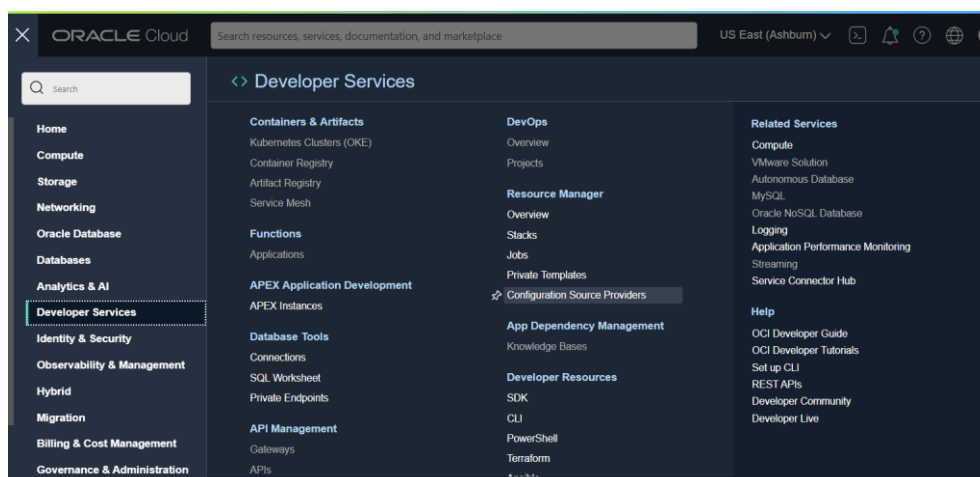
3. Nesta tela preenchemos somente o campo “**Note**” com um nome para o token que estamos criando e selecionamos a opção “**read:packages**” essa é a única permissão necessária para a nossa integração funcionar. Clique em “**Generate Token**” no final da página.



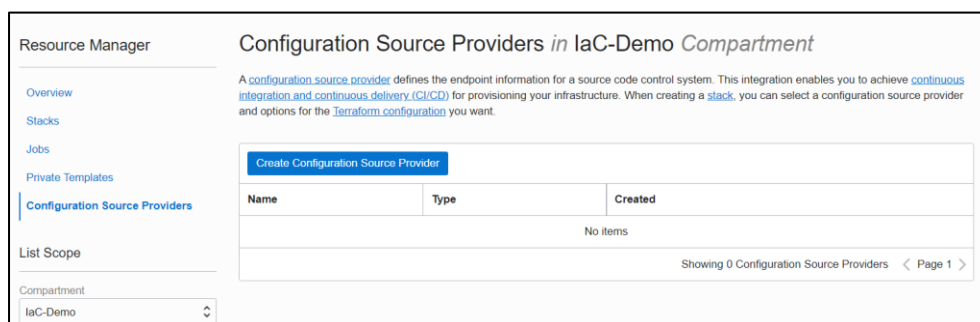
4. Copie o token gerado para usarmos a seguir.



5. Agora acessamos a console do OCI e selecionamos “**Developer Services**” no menu, e em seguida “**Configuration Source Providers**” na seção “**Resource Manager**”.



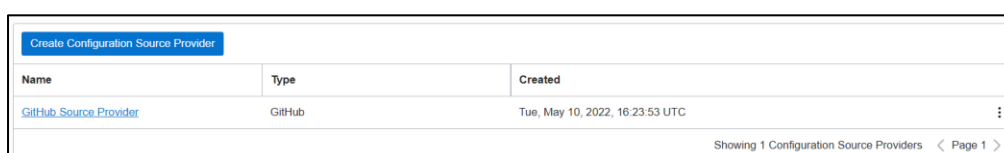
6. Confirme o **Compartment** onde o conector será criado e clique em “**Create Configuration Source Provider**”



7. Na tela que se abre à esquerda, preencher conforme o exemplo abaixo, inserir o “**Personal Access Token**” que criamos no GitHub e clicar em “**Create**”

The screenshot shows the 'Create Configuration Source Provider' form. The form has the following fields: 'Name' (filled with 'GitHub Source Provider'), 'Description' (Optional, empty), 'Compartment' (dropdown menu showing 'IaC-Demo'), 'Type' (dropdown menu showing 'GitHub'), 'Server URL' (filled with 'https://github.com'), and 'Personal Access Token' (password field). There are 'Create' and 'Cancel' buttons at the bottom. A note at the bottom states: 'The token must be granted the read_api scope. Learn more'.

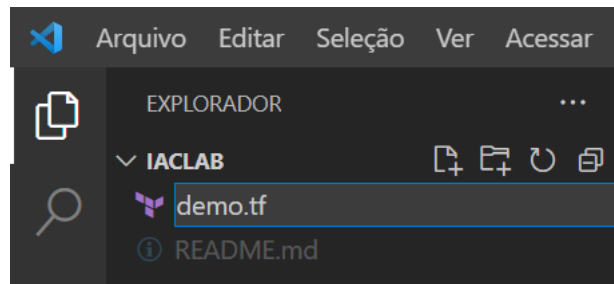
8. Confirme que o conector está criado e com isso nossa integração está finalizada.



Criando o nosso primeiro script Terraform no GitHub

Agora vamos começar a desenvolver nosso script, para esse LAB faremos um script simples que cria um Bucket no serviço de Object Storage no OCI.

1. Vamos voltar ao VSCODE, e acessamos o projeto que criamos anteriormente e criamos um novo arquivo clicando no ícone ao lado do nome de nosso projeto, e damos o nome de **"demo.tf"**.



2. Vamos usar o seguinte script em nosso arquivo **"demo.tf"**, lembrando de trocar o **Compartment ID** e a **região** pelos dados dos seus Tenancies.

```
provider "oci" {  
  region = var.region  
}  
  
variable "compartment_ocid" {  
  default = "insira o OCID do Compartment onde será criado o bucket"  
}  
  
variable "region" {  
  default = "insira a região onde o seu Tenancy está criado ex: us-ashburn-1"  
}  
  
variable "bucket_name" {  
  default = "resource_manager_demo_bucket"  
}  
  
data "oci_objectstorage_namespace" "namespace" {  
  compartment_id = var.compartment_ocid  
}  
  
resource "oci_objectstorage_bucket" "create_bucket" {  
  compartment_id = var.compartment_ocid  
  name = var.bucket_name  
  namespace = data.oci_objectstorage_namespace.namespace.namespace  
  access_type = "NoPublicAccess"  
}  
  
output "new_bucket" {  
  value = oci_objectstorage_bucket.create_bucket  
}
```

3. Agora salvamos o nosso script usando “**ctrl+s**” ou através do menu “**Arquivo**” e vamos carregá-lo no GitHub.
4. Ainda no VSCODE vamos usar a janela do terminal para executar os seguintes comandos, substituindo o conteúdo em vermelho pelos seus próprios dados.

```
opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git add demo.tf

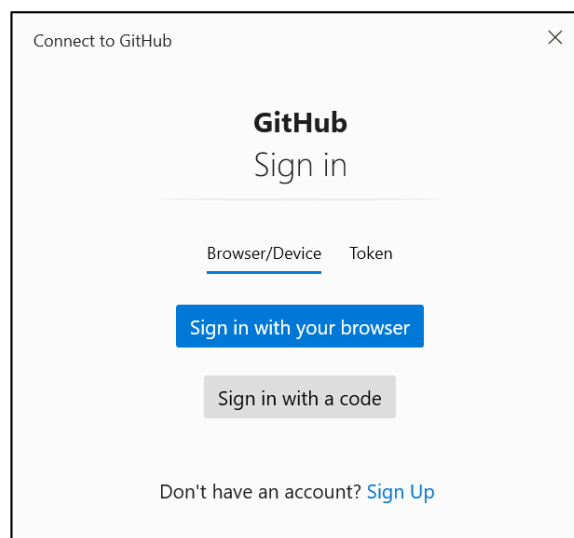
opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git config --global user.email "seuemail@seudominio.com"

opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git config --global user.name "Nome Sobrenome"

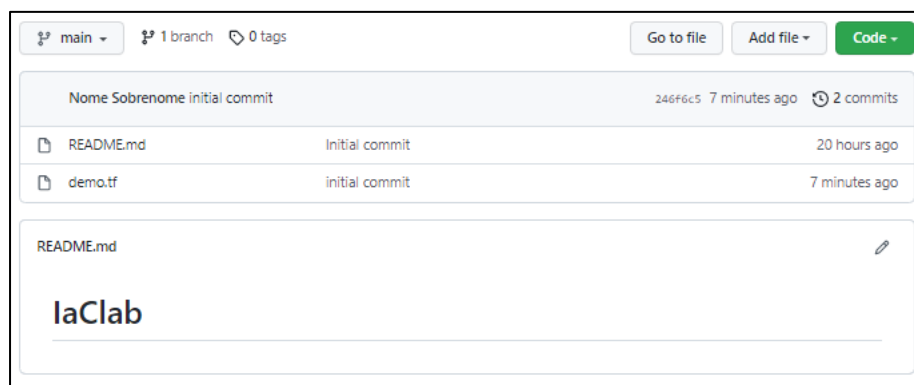
opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git commit -m "initial commit"
[main 246f6c5] initial commit
1 file changed, 29 insertions(+)
create mode 100644 demo.tf

opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 578 bytes | 578.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mygitaccount/laClab
c29cb80..246f6c5 main -> main
branch 'main' set up to track 'origin/main'.
```

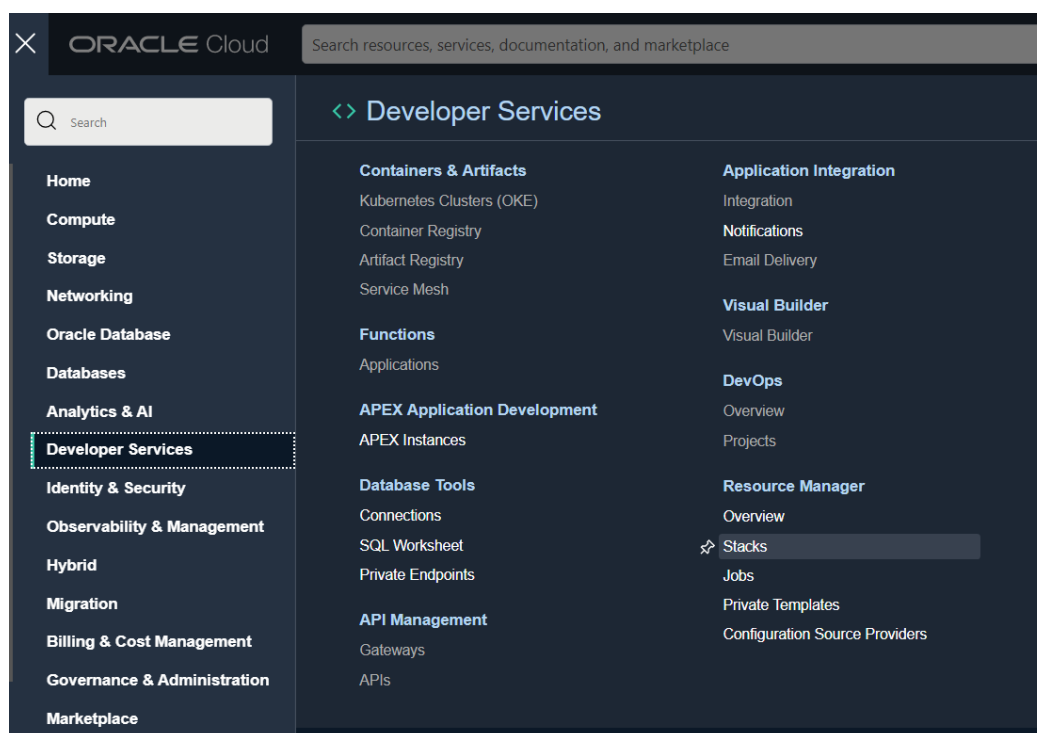
5. Quando for executado o comando “**git push**”, abrirá a tela de autenticação do GitHub em uma nova janela, basta clicar em “**Sign in with Your browser**” e após o login o comando seguirá com sucesso.



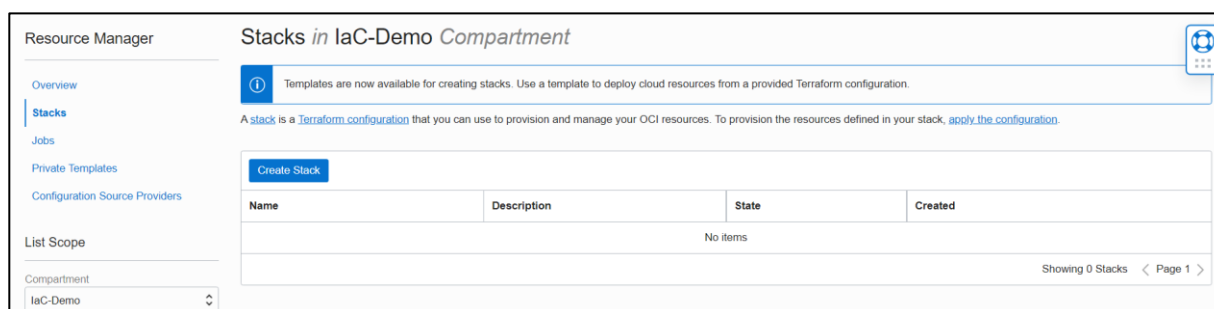
6. Acessando o nosso projeto pelo portal do GitHub você pode conferir que o nosso arquivo “**demo.tf**” foi incluído em nosso projeto.



7. Agora o nosso próximo passo é criar um “**Stack**” no OCI Resource Manager para fazer o deploy de nosso bucket através do script que temos no GitHub. Através do menu de serviços, selecionamos “**Developer Services**” e em seguida clicamos em “**Stacks**” na seção Resource Manager.



8. Confirmar se o “**Compartment**” correto está selecionado e clicar em “**Create Stack**”.



9. Na página de criação do “**Stack**” selecionamos como origem a opção “**Source Code Control System**” e em “**Stack Configuration**” selecionamos o “**Configuration Source Provider**” que criamos anteriormente, e o repositório que estamos usando em nosso projeto.

A [stack](#) is a [Terraform configuration](#) that you can use to provision and manage your OCI resources. To provision the resources defined in your stack, [apply the configuration](#).

Choose the origin of the Terraform configuration. The Terraform configuration outlines the cloud resources to provision for this stack. [Learn more](#)

☐ My Configuration
Upload Terraform configuration files.

☐ Template
Select an Oracle-provided template or private template.

☒ **Source Code Control System**
Select a Terraform configuration from GitHub or GitLab.

☐ Existing Compartment
Create a stack that captures resources from the selected compartment (resource discovery).

Stack Configuration ⓘ

Configuration Source Provider in **laC-Demo** ([Change Compartment](#))

GitHub Source Provider

Repository
laClab

Branch
main

Name *Optional*
Stack-20220510140608

Description *Optional*

Create in compartment
laC-Demo
paulosartori (root)/laC-Demo

Terraform version
1.0.x

ⓘ 0.11.x is no longer supported. [What Terraform versions are supported by Resource Manager?](#)

Tags

10. Após o preenchimento clicamos no botão “**Next**” e na tela de configuração de variáveis manteremos tudo no padrão, e clicamos em “**Next**” novamente. Na tela de “**Review**” clicamos no botão “**Create**”.
11. Nosso “**Stack**” está criado e podemos testá-lo através dos botões “**Plan**” e “**Apply**”.

Resource Manager > Stacks > Stack Details

Stack-20220510140608

Plan Apply **Destroy** Edit More Actions

Stack Information Tags

Stack Information

Description:

Compartment: [REDACTED] root/laC-Demo

OCID: ...3ukylka [Show](#) [Copy](#)

Created: Tue, May 10, 2022, 17:11:30 UTC

Terraform version: 1.0.x

Configuration Source Information

Name: [GitHub Source Provider](#)

OCID: ...v36ceq [Show](#) [Copy](#)

Server: [https://github.com](#)

Repository: [https://github.com/paulosartori/laClab.git](#)

Branch: main

12. Após o término do job de “**Apply**”, confirme que o Bucket está criado, e em seguida execute o job de “**Destroy**” para eliminar o bucket e seguirmos para o próximo passo.

destroy-job-20220510141517

[Edit Job](#) [Download Terraform Configuration](#) [Download Terraform State](#) [Add Tags](#)

Job Information

Tags

OCID: ...i5mcua [Show](#) [Copy](#)

Job Type: Destroy

State: ● Succeeded

Commit Id: 246f6c55e2fb0e34e612578058e915370b3855cd

End Time: Tue, May 10, 2022, 17:16:52 UTC

Refresh State Requested: Yes

Logs

[Download Logs](#) [Show Timestamps](#)

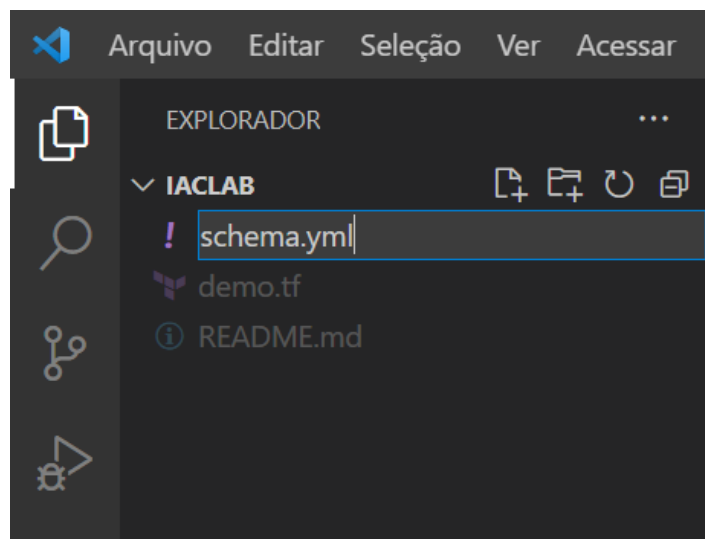
```
- "Oracle-Tags.CreatedOn" = "2022-05-10T17:14:56.228Z"
}
- etag                = "4b1b3f82-d658-4a25-a617-23dde8f2d16e"
- freeform_tags       = {}
- id                  = "n/id0cvlmgwjnu/b/resource_manager_demo_bucket"
- is_read_only        = false
- kms_key_id          = null
- metadata            = {}
- name                = "resource_manager_demo_bucket"
- namespace           = "id0cvlmgwjnu"
- object_events_enabled = false
- object_lifecycle_policy_etag = null
- replication_enabled  = false
- retention_rules      = []
- storage_tier         = "Standard"
- time_created         = "2022-05-10 17:14:56.236 +0000 UTC"
- timeouts            = null
- versioning           = "Disabled"
} -> null
oci_objectstorage_bucket.create_bucket: Destroying... [id=n/id0cvlmgwjnu/b/resource_manager_demo_bucket]
oci_objectstorage_bucket.create_bucket: Destruction complete after 2s

Destroy complete! Resources: 1 destroyed.
```

Utilizando “Schema Documents” para padronizar a entrada de variáveis em nosso script Terraform

Agora que temos nossa integração com o GitHub funcionando e nosso primeiro script criado, vamos incrementar um pouco o processo utilizando os “**Schema Documents**” que são basicamente templates que facilitam e padronizam a entrada de variáveis nos scripts Terraform utilizados no OCI. A documentação completa dos “**Schema Documents**” com sua utilização nos diferentes recursos do OCI pode ser acessadas em “[Extend Console Pages Using Schema Documents](#)”

1. Para o nosso cenário, criamos um arquivo **schema.yml**, que facilitará o preenchimento das variáveis abaixo:
 - **compartment_ocid** : vamos criar uma list drop down com todos os compartments validos no Tenancy.
 - **region**: será uma lista com todas as regiões existentes.
 - **bucket_name**: será uma entrada dinâmica para que for rodar o job e acrescentaremos validação de nome.
 - **access_type**: será uma lista com os valores válidos.
2. Vamos voltar ao VSCODE, e acessamos o projeto que criamos anteriormente e criamos um novo arquivo clicando no ícone ao lado do nome de nosso projeto, e damos o nome de “**schema.yml**”.



3. E utilizamos o código abaixo como conteúdo do arquivo criado. Para mais detalhes da estrutura do arquivo, acesse a documentação indicada anteriormente.

```
title: "Resource Manager Demo Stack"
stackDescription: "Stack para demonstrar como usar o Resource Manager no Oracle Cloud."
schemaVersion: 1.1.0
version: "20220301"
locale: "en"

variableGroups:
- title: "Informações Necessárias"
  variables:
    - "compartment_ocid"
    - "region"
    - "bucket_name"
    - "access_type"

variables:
  "compartment_ocid":
    type: oci:identity:compartment:id
    visible: true
    required: true
    title: "Compartment"
    description: "O compartment onde os recursos serão criados."

  "region":
    type: oci:identity:region:name
    visible: true
    required: true
    title: "Region"
    description: "A região onde os recursos serão criados"

  "bucket_name":
    type: string
    visible: true
    required: true
    title: "Bucket Name"
    description: "O nome do bucket que será criado."
    pattern: "^[a-zA-Z\\d-_.]+$"

  "access_type":
    type: enum
    visible: true
    required: true
    title: "Tipo de Acesso ao Bucket"
    default: "NoPublicAccess"
    enum:
      - NoPublicAccess
      - PublicRead
```

4. Também precisamos atualizar o nosso arquivo “**demo.tf**” removendo as variáveis que estavam inseridas no código:

```
provider "oci" {
  region = var.region
}

variable "compartment_ocid" {}
variable "region" {}
variable "bucket_name" {}

data "oci_objectstorage_namespace" "namespace" {
  compartment_id = var.compartment_ocid
}

resource "oci_objectstorage_bucket" "create_bucket" {
  compartment_id = var.compartment_ocid
  name = var.bucket_name
  namespace = data.oci_objectstorage_namespace.namespace.namespace
  access_type = "NoPublicAccess"
}

output "new_bucket" {
  value = oci_objectstorage_bucket.create_bucket
}
```

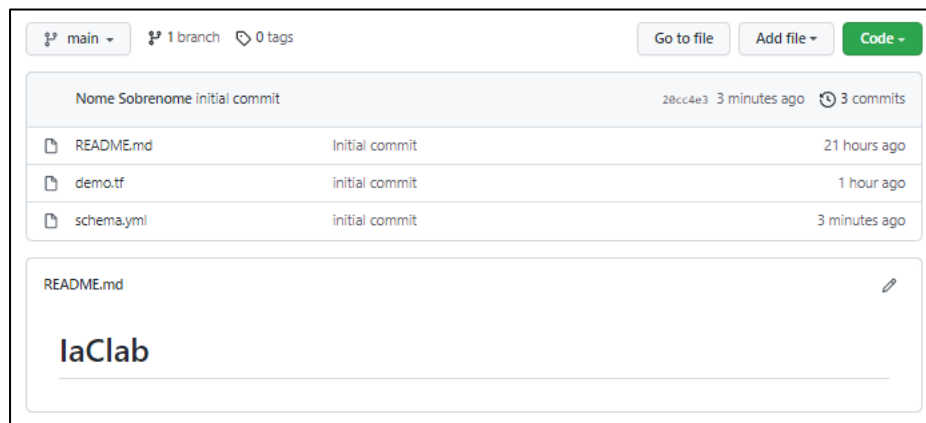
5. Feito isso, precisamos salvar os dois arquivos, o que pode ser feito de forma fácil usando “**ctrl+k+s**” ou através do menu “**Arquivo**” opção “**Salvar Tudo**”. Agora vamos carregar o arquivo de **schema** e as alterações no **demo.tf** para ao GitHub utilizando o terminal no VSCODE com os comandos abaixo:

```
opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git add schema.yml

opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git commit -a -m "initial commit"
[main 20cc4e3] initial commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 schema.yml

opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mygitaccount/laClab
 246f6c5..20cc4e3  main -> main
branch 'main' set up to track 'origin/main'.
```

6. Podemos conferir em nosso projeto do GitHub que agora temos o arquivo de schema adicionado, e em seguida voltamos na console do OCI para criar um novo Stack e testar o nosso arquivo de schema.



7. Para criar o novo stack, seguimos os mesmos passos executados anteriormente, mas note que em “**Stack Information**” já aparece a descrição que incluímos em nosso Schema Document.

A screenshot of the 'Create Stack' form in the OCI console. The form has three steps: 'Stack Information', 'Configure Variables', and 'Review'. The 'Stack Information' step is active. It shows options for 'Configuration Source Provider' (GitHub Source Provider), 'Repository' (IaClab), and 'Branch' (main). Below this is a 'Stack Configuration' section with a dropdown for 'Configuration Source Provider in IaC-Demo' set to 'GitHub Source Provider'. The 'Stack Information' section shows a 'Resource Manager Demo Stack' with a description: 'Stack para demonstrar como usar o Resource Manager no Oracle Cloud.' The 'Name' field is 'Stack-20220510150157' and the 'Description' field is 'Stack para demonstrar como usar o Resource Manager no Oracle Cloud.' The 'Create in compartment' dropdown is set to 'IaC-Demo'.

8. E clicando em “**Next**”, na tela de variáveis, já aparecem as listas que criamos para serem selecionadas:

Create Stack

1 [Stack Information](#)
 2 **[Configure Variables](#)**
 3 [Review](#)

Configure the variables for the infrastructure resources that this stack will create when you run the apply job for this execution plan.

Informações Necessárias

Compartment
 IaC-Demo
 O compartment onde os recursos serão criados.

Region
 us-ashburn-1
 A região onde os recursos serão criados.

Bucket Name
 1@
 O nome do bucket que será criado.
 Specify a value that satisfies the following regular expression: `*([a-zA-Z0-9_-])*`

Tipo de Acesso ao Bucket

NoPublicAccess
Select an option
NoPublicAccess
PublicRead

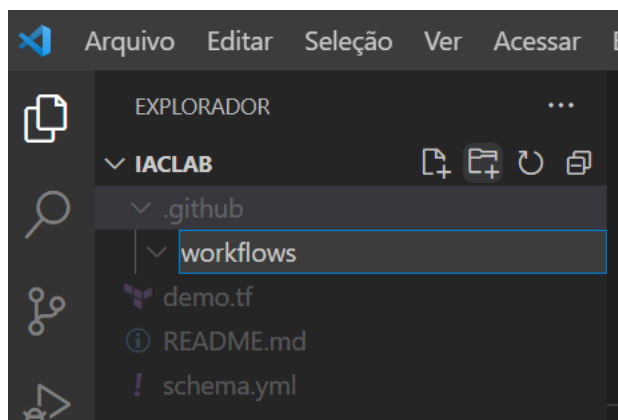
9. Após preencher com um nome válido de Bucket e selecionar as demais variáveis, clique em **“Next”** e **“Create”**.
10. Teste o stack criado utilizando os botões **“Plan”** e **“Create”** e uma vez completados com sucesso execute o job de **“Destroy”** para eliminar o Bucket.

Utilizando o “GitHub Actions” para gerar uma nova release distribuível do script

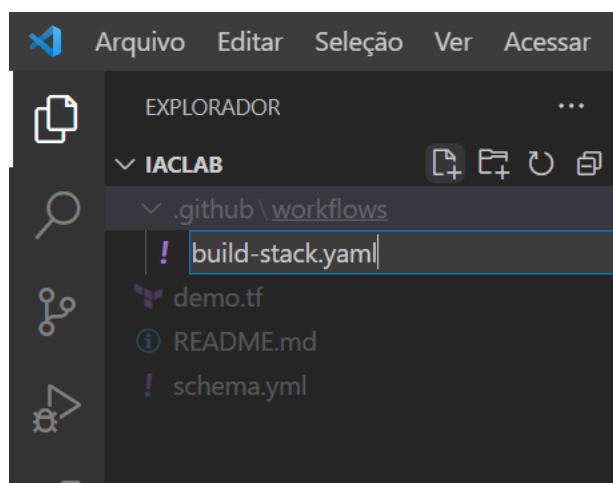
Agora que temos a versão final de nosso script, incluindo a padronização de entrada de variáveis, vamos gerar a primeira distribuição de nosso código utilizando o GitHub Actions para que todas as vezes que gerarmos uma nova versão, uma release seja disponibilizada no GitHub no formato de um arquivo .zip que pode ser utilizado por qualquer pessoa em seus respectivos Tenancies.

Para começar, vamos criar um arquivo build-stack.yaml em nosso repositório no caminho .github/workflows que executará as seguintes ações todas as vezes que subirmos uma atualização de código com uma nova TAG:

- Fazer o checkout do novo código
 - Zipar tudo que estiver no diretório raiz
 - Publicar o ZIP como um artefato no GitHub
 - Criar uma nova release no GitHub
 - Fazer o upload do artefato como uma release
1. O primeiro passo é criar as novas pastas do projeto, então vamos no VSCODE e clicamos no ícone Nova Pasta para criar a pasta “**.github**” e em seguida selecionamos a pasta “**.github**”, clicamos novamente em Nova Pasta e criamos a pasta “**workflows**”.



2. E finalmente, criamos o arquivo “**build-stack.yaml**” dentro da pasta workflows, clicando no ícone Novo Arquivo.



3. Utilizaremos o conteúdo abaixo no nosso arquivo **“build-stack.yaml”** e salvar utilizando **“ctrl+s”**

```
name: oci-terraform-build-stack
on:
  push:
    tags:
      - "v*"
jobs:
  build-stack:
    name: Build Stack
    runs-on: ubuntu-latest
    steps:

      - name: 'Checkout'
        uses: actions/checkout@v2

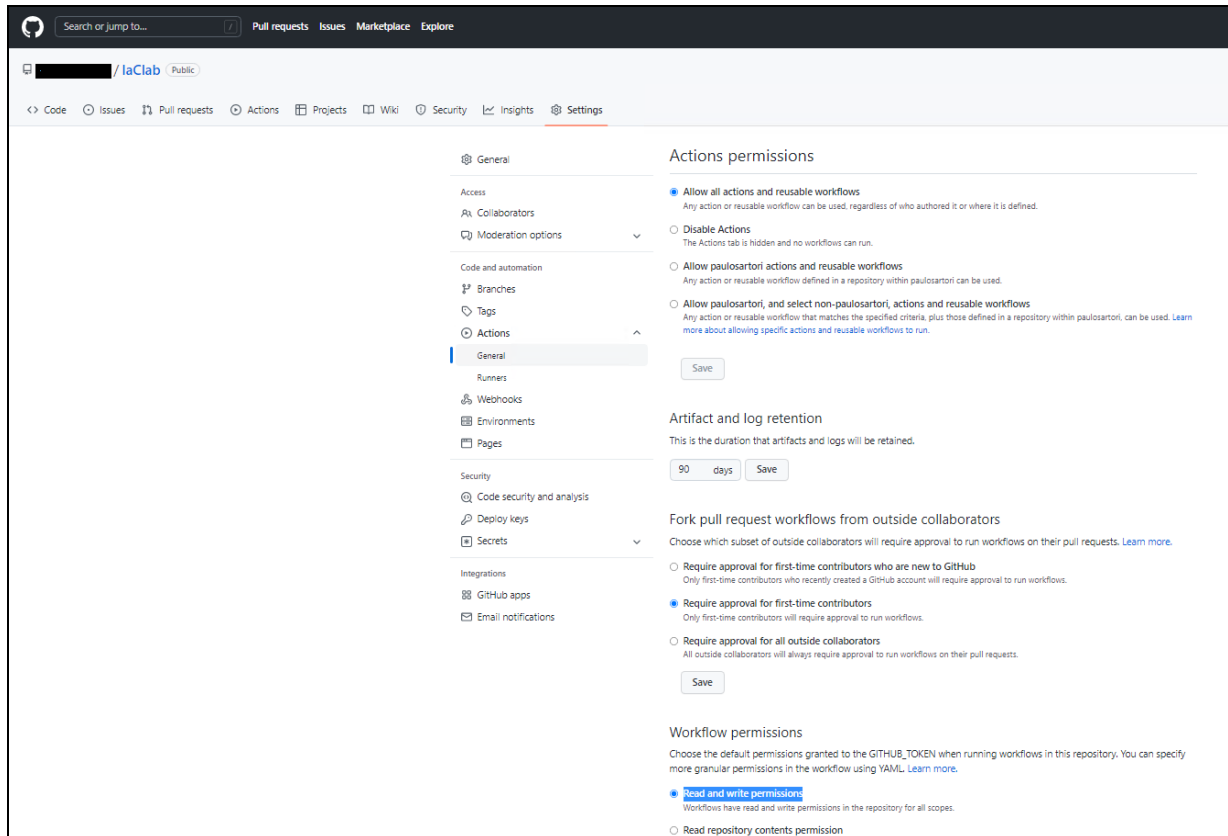
      - name: 'Build Stack'
        run: |
          zip -r stack.zip *

      - name: 'Publish Stack'
        uses: actions/upload-artifact@v2-preview
        with:
          name: 'stack'
          path: 'stack.zip'

      - name: 'Create Release'
        if: success()
        id: create_release
        uses: actions/create-release@v1
        env:
          GITHUB_TOKEN: ${secrets.GITHUB_TOKEN}
        with:
          tag_name: ${github.ref}
          release_name: Release ${github.ref}
          body: |
            Latest release
          draft: false
          prerelease: false

      - name: 'Upload Release Asset (Stack)'
        if: contains(github.ref, 'v')
        id: upload-release-asset-stack
        uses: actions/upload-release-asset@v1
        env:
          GITHUB_TOKEN: ${secrets.GITHUB_TOKEN}
        with:
          upload_url: ${steps.create_release.outputs.upload_url}
          asset_path: ./stack.zip
          asset_name: stack.zip
          asset_content_type: application/zip
```

4. Antes de subir as alterações para o GitHub, precisamos conferir as permissões do workflow em nosso repositório, para isso, na página do nosso repositório no portal do GitHub, clicamos em “**Settings**”, em seguida em “**Actions**” e “**General**”, garanta que na seção “**Workflow permissions**” a opção “**Read and write permissions**” esteja selecionada, e clique em “**Save**”



- Próximo passo é sincronizar as alterações que fizemos para o GitHub, executando os seguintes comandos:

```
opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git add .github/workflows/build-stack.yaml

opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git commit -a -m "initial commit"
[main 08f2e6c] initial commit
1 file changed, 49 insertions(+)
create mode 100644 .github/workflows/build-stack.yaml

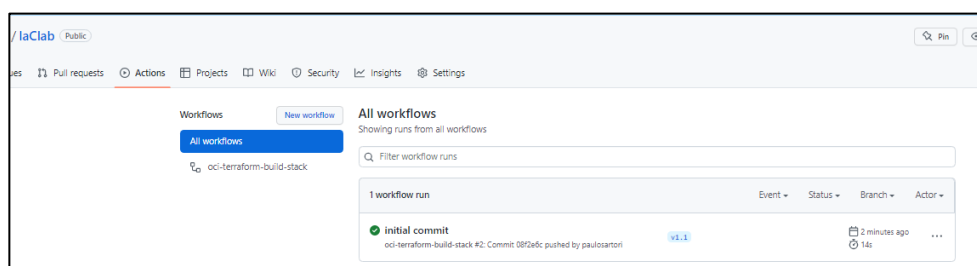
opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 901 bytes | 901.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/c/laClab
 b571f84..08f2e6c main -> main
branch 'main' set up to track 'origin/main'.
```

- E agora vamos criar a nossa primeira Release criando uma **tag** de versionamento com os seguintes comandos:

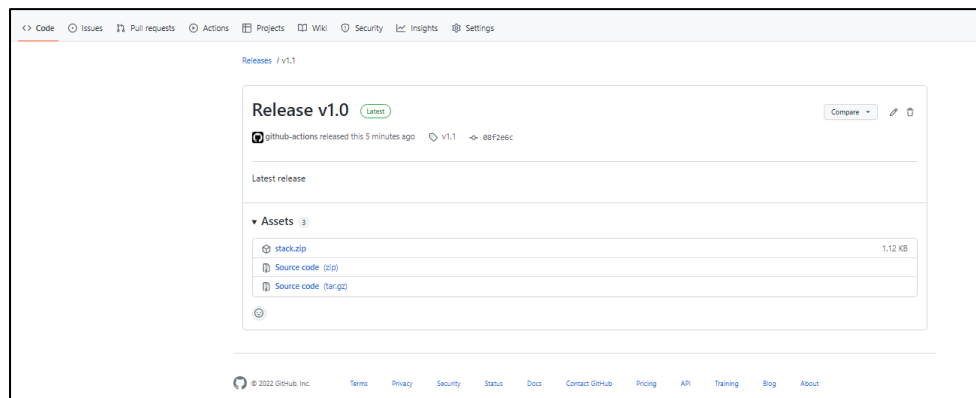
```
opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git tag -a v1.0 -m "Versão Inicial"

opc@VM-Win-Demo MINGW64 ~/Desktop/laClab (main)
$ git push origin v1.0
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 175 bytes | 175.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mygitaccount/laClab
 * [new tag]      v1.0 -> v1.0
```

- Nesse momento nosso workflow é acionado no GitHub e a execução pode ser validada no menu **Actions**.



8. E o arquivo “stack.zip” é criado e pode ser distribuído.



A release mais recente pode ser acessada diretamente pelo link abaixo, substituindo as palavras em vermelho pelos seus respectivos dados.

[https://github.com/gituser**/**seurepositorio**/releases/latest](https://github.com/gituser/seurepositorio/releases/latest)**

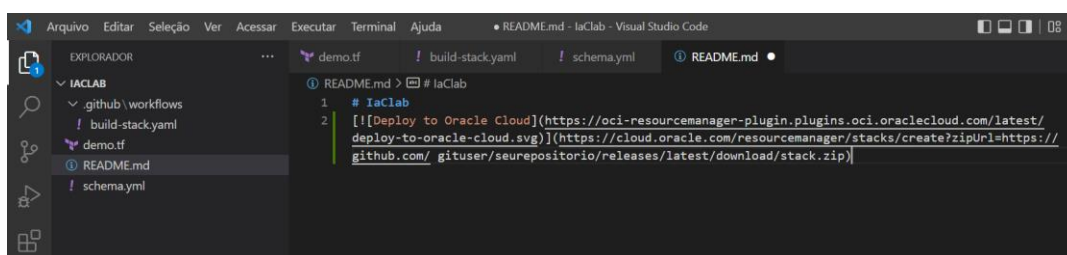
Fazendo deploy em 1 clique do nosso script em qualquer Tenancy do OCI

Para finalizar, agora que temos um arquivo ZIP que pode ser utilizado por qualquer pessoa, podemos acrescentar um botão para deploy com apenas 1 clique da nossa release do Stack direto no OCI. Isso faz com que o seu desenvolvedor não precise configurar o Resource Manager para acessar o GitHub, e pode fazer o deploy direto do repositório do GitHub ou da Intranet da sua empresa.

1. Para isso o processo é bem simples, basta acrescentar o seguinte código no nosso arquivo **README.md** usando o VSCODE para editá-lo e salvá-lo usando **ctrl+s**.

```
[![Deploy to Oracle Cloud](https://oci-resourcemanager-plugin.plugins.oci.oraclecloud.com/latest/deploy-to-oracle-cloud.svg)](https://cloud.oracle.com/resourcemanager/stacks/create?zipUrl=https://github.com/gituser/seurepositorio/releases/latest/download/stack.zip)
```

2. Lembrando de substituir as palavras em vermelho pelos seus respectivos dados.

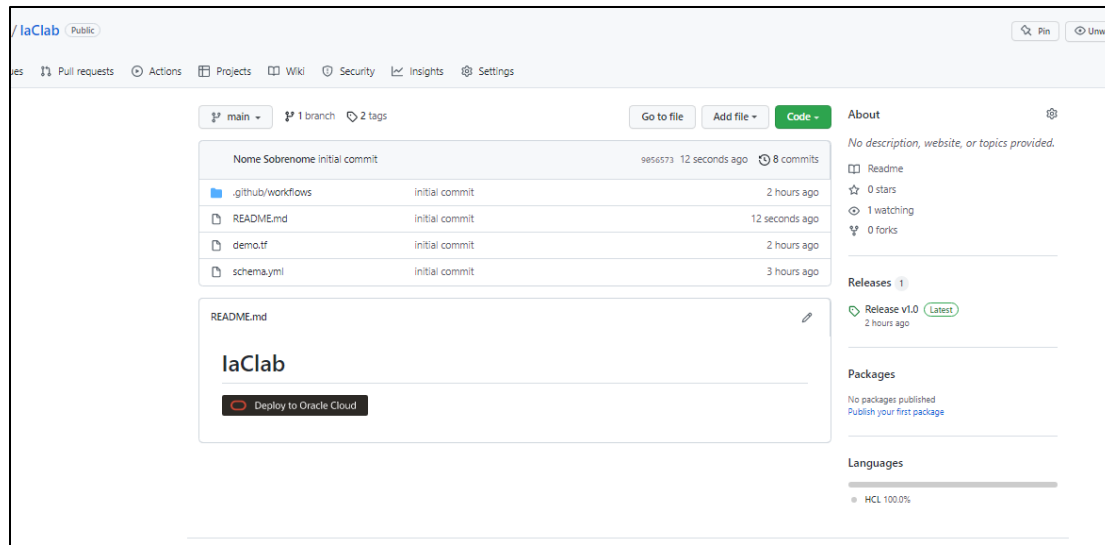


3. E em seguida subir a alteração para o GitHub com os seguintes comandos:

```
opc@VM-Win-Demo MINGW64 ~/Desktop/IaCLab (main)
$ git commit -a -m "initial commit"
[main 3b60ccb] initial commit
1 file changed, 2 insertions(+), 1 deletion(-)

opc@VM-Win-Demo MINGW64 ~/Desktop/IaCLab (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 435 bytes | 435.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/mygitaccount/IaCLab
08f2e6c..3b60ccb main -> main
branch 'main' set up to track 'origin/main'.
```

- O resultado pode ser visto no GitHub, onde em nosso repositório já aparecerá o botão de **deploy no OCI** do nosso projeto, agora qualquer pessoa que clicar no botão de deploy será levado direto para a página de deploy do Stack no OCI.



- Clicando no botão de Deploy to Oracle Cloud, será pedida a autenticação na console do OCI e em seguida aparecerá a página para criar o Stack.

1 Stack Information

2 Configure Variables

3 Review


Welcome!

You're here because you clicked a button to deploy cloud resources, using the [package](#) identified below.

Package URL: <https://github.com/paulosartori/laClab/releases/latest/download/stack.zip>

☒ I have reviewed and accept the [Oracle Terms of Use](#).

Stack Information ⓘ



Resource Manager Demo Stack
Stack para demonstrar como usar o Resource Manager no Oracle Cloud.

Working Directory
The root folder is being used as the working directory.

Name *Optional*

Description *Optional*

Create in compartment

paulosartori (root)/laC-Demo

Terraform version

ⓘ

0.11.x is no longer supported. [What Terraform versions are supported by Resource Manager?](#)

Tags
Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources.
[Learn more about tagging](#)


A partir daí é só seguir o workflow de criação.

Connect with us

Call **+1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

 blogs.oracle.com

 facebook.com/oracle

 twitter.com/oracle

Copyright © 2022, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

This device has not been authorized as required by the rules of the Federal Communications Commission. This device is not, and may not be, offered for sale

Disclaimer: If you are unsure whether your data sheet needs a disclaimer, read the revenue recognition policy. If you have further questions about your content and the disclaimer requirements, e-mail REVREC_US@oracle.com.

or lease, or sold or leased, until authorization is obtained.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0120