Terraform

CI/CD

Presented by Sean P. Kane

https://techlabs.sh/

Release 2024-02-16-1510

Instructor

Sean P. Kane

@spkane

techlabs

https://techlabs.sh

Follow Along Guide

Textual Slides

O'Reilly Online Sandbox VM

https://learning.oreilly.com/interactive-lab/devops-tools-sandbox/9781098126469

NOTE: *VM sessions will expire after 60 minutes!*

Prerequisites (1 of 3)

NOTE: You *MUST* have open access to Github if you want to participate in the hands-on portion of class from your local computer system.

Prerequisites (2 of 3)

- A recent computer and OS
  - Recent/Stable Linux, macOS, or Windows 10+
  - Reliable and fast internet connectivity
- Hashicorp Terraform

Prerequisites (2 of 2)

- A graphical web browser
- A text editor
- A software package manager
- `git` client
- General comfort with the command line will be helpful.
- [optional] `tar`, `wget`, `curl`, `jq`, `ssh` client
  - `curl.exe` for Windows - https://curl.se/windows/

A Note for Powershell Users

Terminal commands reflect the Unix bash shell. PowerShell users will need to adjust the commands.

- Unix Shell Variables
  - `export MY_VAR='test'`
  - `echo ${MY_VAR}`
- PowerShell Variables
  - `$env:my_var = "test"`
  - `Get-ChildItem Env:my_var`
  - `Remove-Item Env:\my_var`

Translation Key

`\` - Unix Shell Line Continuation
`` ` `` - Powershell Line Continuation (sort of)

`${MY_VAR}` - Is generally a placeholder in the slides.

A Note About Proxies & VPNs

Proxies can interfere with some activities if they are not configured correctly and VPNs can increase audio and video streaming issues in class.

- Configure `HTTP_PROXY`, `HTTPS_PROXY`, and your web browser.
- Terraform
- Docker
- Docker-Compose

Instructor Environment

- **Operating System**: macOS (v14.X.X+)
- **Terminal**: iTerm2 (Build 3.X.X+) - https://www.iterm2.com/
- **Shell Prompt Theme**: Starship - https://starship.rs/
- **Shell Prompt Font**: Fira Code - https://github.com/tonsky/FiraCode
- **Text Editor**: Visual Studio Code (v1.X.X+) - https://code.visualstudio.com/
- `export BUILDKIT_COLORS=run=green:warning=yellow:error=red:cancel=cyan`

Code Setup

- If you missed the first class.

```
$ cd ${HOME}
$ mkdir class
$ cd ${HOME}/class
$ git clone https://github.com/spkane/todo-for-terraform
$ cd todo-for-terraform
```

Spin up the Environment

```
$ cd ${HOME}/class
$ cd todo-for-terraform/terraform-infrastructure-aws
$ . ~/bin/ns1_personal
$ terraform apply
$ . ./bin/ip_vars.sh
$ cd ..
```

- Note: Students **CAN NOT** run `terraform` in the `terraform-infrastructure-aws` directory. This is for the instructor only.

Define GitHub Token in Environment

- Remember that GitHub token from last week? This is where we need to use it again.

```
$ export TF_VAR_github_token=${GITHUB_TOKEN}
```

- I am going to run `$ . ~/bin/gh_personal` to set this up locally.

Copy and Apply the Code

- `cd $HOME/class/todo-for-terraform`
- `mkdir -p tf-code`
- `cp -a terraform-tests-3 tf-code`
- `cd ./tf-code/terraform-tests-3`
- Review the Terraform code

Initialize and Apply

- `terraform init`
- `terraform apply`

The GitHub Repo

- Repo URL: `terraform output -raw repo_url`
- Open it up in your web browser

Terraform Cloud Account

- Navigate your web browser to:
  - https://app.terraform.io/session

Terraform Cloud Account Creation

- Fill out:
  - `Username`
  - `Email`
  - `Password`
  - Agree to `Terms of Use`
  - Acknowledge `Privacy Policy`
- Click `Create account`
- Click confirmation link in email.

Login to Terraform Cloud GUI

- Navigate your web browser to:
  - https://app.terraform.io/app/getting-started

Login to Terraform Cloud CLI

- `terraform login`

Login Error

- Got an error?
  - `Error: Credentials for app.terraform.io are manually configured`
- This usually means you already have a block like this in `~/.terraformrc` or one of the files under `~/.terraform.d`.

```
credentials "app.terraform.io" {
token = "REDACTED"
}
```

Create Terraform Cloud Token

- Click `Create API token` button in the UI.
- Copy token via the blue clipboard icon on the right.
  - Paste the token into the `terraform login` prompt and hit enter.
- If all goes well you should see a `Welcome to Terraform Cloud!` message.
- Click the `Done` button in the UI.

Terraform Cloud Organization

- Navigate your web browser to:
  - https://app.terraform.io/app/organizations/new

Create a New Organization

- Fill out form
  - **Organization name**
    - Must be unique
  - **Email address**
    - Must be valid
- This should redirect you to:
  - https://app.terraform.io/app/${ORG_NAME}/workspaces

Create a Workspace

- Click **Create a workspace**
- Click **CLI-driven workflow**
- Fill out form
  - **Name**: `terraform-class`
  - **Project**: `Default Project`
- Click **Create workspace**

Terraform Cloud Repo

```
$ cd ..
# inside directory "todo-for-terraform/tf-code"
$ git clone https://github.com/spkane/terraform-cloud-example.git
$ cd terraform-cloud-example
```

Configure Backend

- edit `backend.tf` with correct ORG and WORKSPACE name.

```
cloud {
  organization = "{{ORGANIZATION_NAME}}"

  workspaces {
    name = "{{WORKSPACE_NAME}}"
  }
}
```

Configure the Provider

- Open `provider.tf`
  - Configure the todo provider
    - Change `host = "127.0.0.1"` to `host = "todo-api.techlabs.sh"`

Prep Environment

- If you have `${TF_PLUGIN_CACHE_DIR}` set
  - RUN
    - `unset TF_PLUGIN_CACHE_DIR`
  - OR
    - `terraform providers lock -platform=linux_amd64`

Initialize Terraform Cloud

- Run `terraform init`
- Run `terraform apply`
  - **NOTE**: It is important to understand, that by default this Terraform run is being run remotely and there is no access to local system or network resources.
  - If we were running the `todo` service locally, this would not work as currently written.

Apply Output

```
$ terraform apply

Running apply in the remote backend. Output will stream here. Pressing Ctrl-C
will cancel the remote apply if it's still pending. If the apply started it
will stop streaming the logs, but will not stop the apply running remotely.

Preparing the remote apply…

To view this run in a browser, visit:
https://app.terraform.io/app/${ORG}/${WORKSPACE}/runs/run-${JOB_ID}
…
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
```

Explore the UI

- Open the run link from the `terraform apply` output.
  - https://app.terraform.io/app/${ORG}/${WORKSPACE}/runs/run-${JOB_ID}
- Overview
- States
- Variables
- Settings

Commit to CI/CD Repo

```
$ cd ..
$ git clone https://github.com/${GH_USER}/testing-terraform-ci-cd
$ mv terraform-cloud-example/*.tf testing-terraform-ci-cd/
$ mv terraform-cloud-example/.terraform.lock.hcl testing-terraform-ci-cd/
$ mv terraform-cloud-example/.gitignore testing-terraform-ci-cd/
$ rm -rf ./terraform-cloud-example
$ cd testing-terraform-ci-cd/
$ git add .
$ git commit -m "Initial commit"
$ git push origin HEAD
```

Setup Version Control (1 of 2)

- **NOTE**: Popup blockers will cause you issues here.
- Web UI: Settings - Version Control
- Click `Connect to version control`
- Select `Version control workflow`
- Select `GitHub` - `GitHub.com`
- Login or click `Continue` in the GitHub popup

Setup Version Control (2 of 2)

- **NOTE**: Popup blockers will cause you issues here.
- Click `Authorize Terraform Cloud`
- Select the `Organization` that you want to install it to.
- Pick `Only select repositories`
- Choose `testing-terraform-ci-cd`
- Click `Install`

Select Repository

- Select `testing-terraform-ci-cd`
- Leave `Terraform Working Directory` empty
- Select `Auto apply`
- Leave `Automatic Run Triggering` set to `Always trigger runs`
- Leave `VCS branch` empty
- Leave `Automatic speculative plans` checked
- Leave `Include submodules on clone` unchecked
- Click `Update VCS settings`

First VCS Run

- Navigate to:
  - https://app.terraform.io/app/${ORG}/${WORKSPACE}/runs

New Terraform Pull Request

```
$ git checkout -b feature/class-repo
```

Add new GH Repo

- in `main.tf` (*combine broken lines*)

```
module "simple_github_repo" {
  source  = "github.com/spkane/terraform-github-repository?
ref=8b72dcbac2c4287672a22435e36bbc27a869db5c"

  name            = "testing-terraform-modules-via-ci-cd"
  description     = "Testing GitHub repo automation via Terraform CI/CD.
This repo should probably be deleted."
  visibility      = "private"
  auto_init       = true
  default_branch = "main"
  has_issues      = "true"
}
```

Add Required Provider

- in `main.tf`

```
terraform {
  required_providers {
    todo = {
…
    }
    github = {
      source  = "integrations/github"
      version = "~> 5.0"
    }
  }
}
```

Configure the GitHub Provider

- in `provider.tf`

```
provider "github" {
}
```

- Since it is better to avoid using Terraform variables for sensitive data, let's give that a try.
- https://registry.terraform.io/providers/integrations/github/latest/docs#oauth--personal-access-token

Add Repo Outputs

- Create `outputs.tf` with the following:

```
output "repo_url" {
  value = module.simple_github_repo.github_repository.html_url
}
```

Commit and Push PR

```
$ git add .
$ git commit -m "add new repo"
$ git push origin HEAD
…
remote:
Create a pull request for 'feature/class-repo' on GitHub by visiting:
https://github.com/${GH_USER}/testing-terraform-ci-cd/pull/new/feature/class-repo
…
```

- Navigate to the above URL.

Create Pull Request

- Optionally, add `comment`
- Click `Create pull request`
- Terraform GitHub App should run speculative plan.
  - Once check passes, click `Show all checks`
  - Then click `Details`
- Examine the resulting plan in the Terraform Cloud UI.

Merge Pull Request

- Assuming all is good, let's merge the PR!
- Navigate to
  - https://github.com/${GH_USER}/testing-terraform-ci-cd/pull/1
- Click `Merge pull request`
- Click `Confirm merge`
- Click `Delete branch`

Merge Check Status

- Watch the status of the check on the `main` branch.
  - https://github.com/${GH_USER}/testing-terraform-ci-cd/commits/main

Examine Results

- **Spoiler**
  - The apply failed. *Why?*
- Navigate to:
  - https://app.terraform.io/app/${ORG}/${WORKSPACE}/runs
- Examine the `Diagnostics`

Results

- The provider is not fully configured on the remote system.
  - Terraform Cloud does not have access to our Github token.
- Let's fix this!

Adding an Environment Variable

- We **could** add this via the UI by navigating to:
  - https://app.terraform.io/app/${ORG}/${WORKSPACE}/variables
    - Click `+ Add variable`
    - Select `Environment variable`
    - Set `Key` to `GITHUB_TOKEN`
    - Set `Value` to your GitHub Token (*not hidden when typed*)
    - Check `Sensitive`
    - Set `Variable Description` to `The token for the GH provider`
    - Don't click `Add variable` (*There is a better option*)

The Terraform Cloud Provider

- But there is also a Terraform Cloud/Enterprise provider that we can use:
  - https://registry.terraform.io/providers/hashicorp/tfe
- Ideally, we would actually use this for setting up everything in Terraform Cloud, but as an introduction to Terraform Cloud it is useful to explore things via the UI.

Variable Setup via the TFE Provider

```
$ cd ../../terraform-infrastructure-tfcloud/
```

- Examine the HCL
- This should just work, **IF**
  - You have run `terraform login` and the token is still valid.
  - You have the variable TF_VAR_github_token set correctly in your local environment.
  - Both of which should be true.

Create our Variable

- Replace `${TFCLOUD_ORG_NAME}` in `terraform apply` command.

```
$ terraform init
$ terraform apply -var="org_name=${TFCLOUD_ORG_NAME}"
```

Re-attempt Broken Apply (1 of 2)

- Navigate to:
  - https://app.terraform.io/app/${ORG}/${WORKSPACE}/runs

Re-attempt Broken Apply (2 of 2)

- Open most recent `errored` run
- Click `+New Run`
- Fill out `Reason for starting run` with `Adding GH token to TFC`
- Leave `Choose run type` set to `Plan and apply (standard)`
- Click `Start run`
- Examine results

Out-of-Band Changes

- Navigate to:
  - https://github.com/${GH_USER}/testing-terraform-modules-via-ci-cd
- Click the gear on the upper right, next to `About`.
- Change the `description`, however you want.
- Click `Save changes`

Local Plan

```
$ cd ../tf-code/testing-terraform-ci-cd/
$ git checkout main
$ git pull
$ terraform init
$ terraform plan
```

Local Plan Output

```
Running plan in Terraform Cloud.
…
Note: Objects have changed outside of Terraform
…
  # module.simple_github_repo.github_repository.main will be updated in-place
  ~ resource "github_repository" "main" {
      ~ description                 = "Testing GitHub repo automation via
Terraform CI/CD. This repo should probably be deleted whenever I get a
moment." -> "Testing GitHub repo automation via Terraform CI/CD. This
repo should probably be deleted."
        id                          = "testing-terraform-modules-via-ci-cd"
        name                        = "testing-terraform-modules-via-ci-cd"
        # (33 unchanged attributes hidden)
    }

Plan: 0 to add, 1 to change, 0 to destroy.
```

Local Apply

```
$ terraform apply
```

- **NOTE**: This doesn't work when we have a Terraform Cloud workspace that is configured to use a version control system (VCS).

Apply Changes via the UI

- Navigate to:
  - https://app.terraform.io/app/${ORG}/${WORKSPACE}/
- Click `+New Run`
- Fill out `Reason for starting run` with `Fixing out-of-band change`
- Click `Start run`
- Verify the `Description` change was applied.

Confirm Description Fix

- Navigate to:
  - https://github.com/${GH_USER}/testing-terraform-modules-via-ci-cd
- Verify that the `Description` has been reverted to it's intended state.

VCS Destroy in Terraform Cloud

- Navigate to:
  - https://app.terraform.io/app/${ORG}/${WORKSPACE}/settings/delete
- Click `Queue destroy plan`
- In `Enter the workspace name to confirm:` type `terraform-class`
- Click `Queue destroy plan`
- Verify that `terraform destroy` was successful.
- **NOTE**: You should do this now, BEFORE we tear-down the AWS infrastructure in a few more slides.

Delete Workspace

- Navigate to:
  - https://app.terraform.io/app/${ORG}/${WORKSPACE}/settings/delete
- Click `Delete from Terraform Cloud`
- In `Enter the name to confirm:` type `terraform-class`
- Click `Delete workspace`
- We don't need to delete our `GITHUB_TOKEN` variable because this remove everything in the workspace.

Delete the CI/CD Repo

```
$ cd ../terraform-tests-3/
$ terraform destroy
$ cd ../..
```

Tear Down the Environment

```
$ cd terraform-infrastructure-aws
$ terraform destroy
$ cd ..
```

- Note: Students **CAN NOT** run `terraform` in the `terraform-infrastructure-aws` directory. This is for the instructor only.

Additional CI/CD Options

- Atlantis (*open source*)
  - https://www.runatlantis.io/
- Spacelift (*multi-vendor support*)
  - https://spacelift.io/
- Doppler (*secrets management*)
  - https://www.doppler.com/
- and more…

What We Have Learned

- Creating a Terraform Cloud account
- Using `terraform login`
- Using Terraform Cloud via the UI/CLI
- Using GitOps with Terraform Cloud
- Using the Terraform Cloud Provider
- and more...

Additional Reading

Terraform: Up & Running
Terraform Documentation

Additional Learning Resources

https://learning.oreilly.com/

Student Survey

Please take a moment to fill out the class survey linked to from the bottom of the ON24 audience screen.

O’Reilly and I value your comments about the class.

Thank you!

Any Questions?

Sean P. Kane

Hands-on technical training and engineering

https://techlabs.sh/