DevOps Lab Assignment

#### (Provide your code snippets. You may answer the questions with screenshots if applicable.)

In this lab, we're going to deploy some of the services we’ve used in class through “Infrastructure as Code” tools. To start, you’ll deploy using Terraform on your local machine. Then, you’ll take the Terraform logic that you wrote and put it in a GitHub repo and configure a GitHub Action to deploy automatically anytime a code change is pushed to the repo. At the end, you’ll provide a simple architecture diagram of what you deployed.

Here’s what we want to deploy:

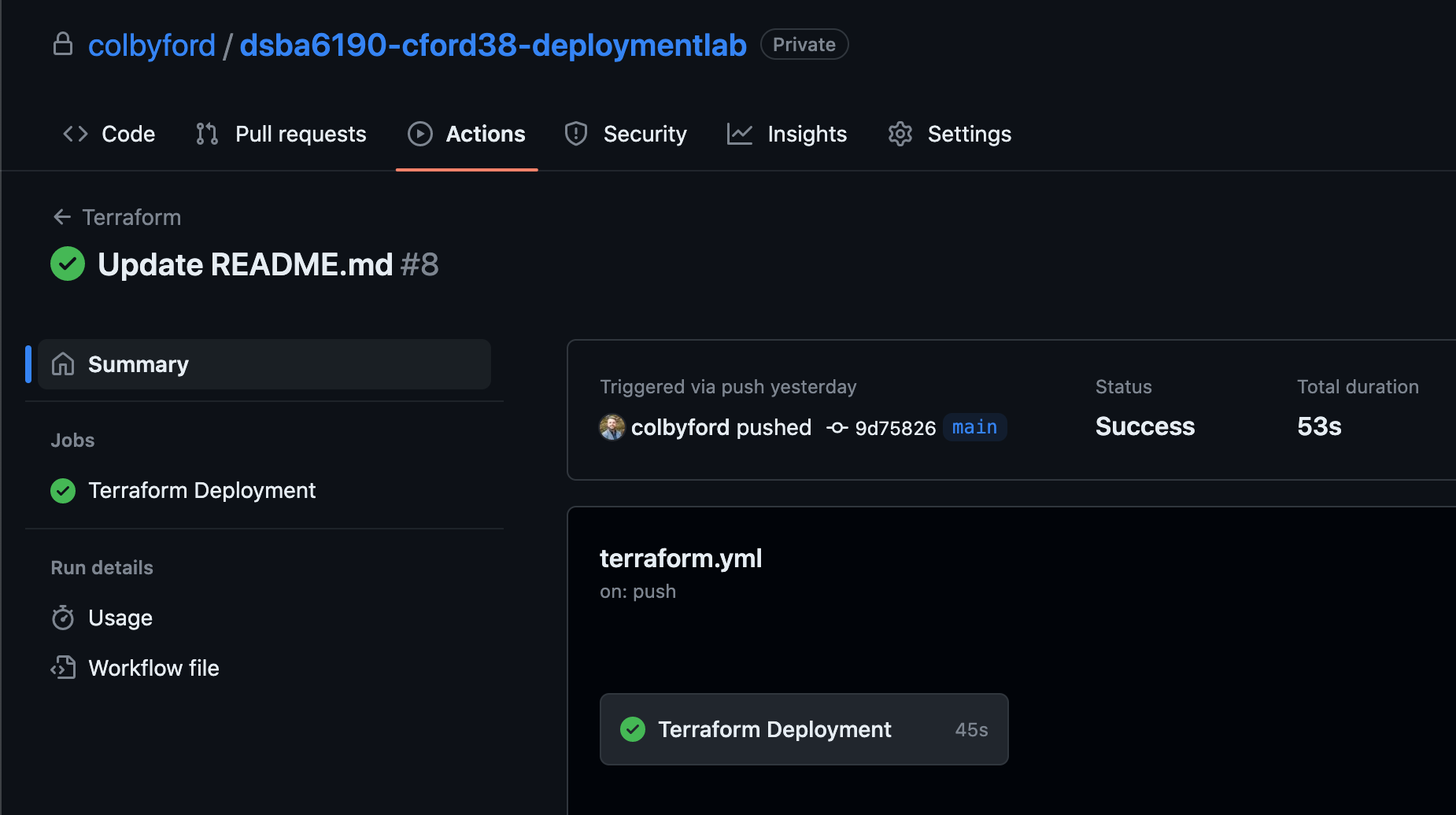
|  |  |  |
| --- | --- | --- |
| **Service** | **Example Name** | **Other Settings** |
| Resource Group | rg-dsba6190-cford38-dev-eastus-001 | None |
| Virtual Network (and subnet) | vnet-dsba6190-cford38-dev-eastus-001 | * Address Space: 10.0.0.0/16 * Subnet Prefix: 10.0.2.0/24 * Appropiate service endpoints   + e.g., Microsoft.Sql, Microsoft.Storage |
| Storage Account | stodsba6190cford38dev001 | * Turn on the hierarchical namespace * LRS Replication * Standard account tier * Deploy inside the virtual network |
| Azure SQL Server and Database | sql-dsba6190-cford38-dev-001  db-dsba6190-cford38-dev-001 | * Database SKU: Basic * Deploy with a virtual network rule |

### A couple notes:

* You’ll need to make a GitHub repository with an Action and add in Secrets.
  + Here’s my example repo: <https://github.com/colbyford/dsba6190-cford38-deploymentlab>
  + How to make Secrets: <https://docs.github.com/en/actions/security-guides/encrypted-secrets>
* Here’s is the naming convention you can follow for your resources: <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming>
* Here’s the Terraform documentation: <https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs>
  + Resource Examples: [Storage Account](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/storage_account) ([with vnet](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/mssql_database)), [Azure SQL Database](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/mssql_database) ([with vnet](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/mssql_virtual_network_rule))
* Be mindful of the type of database service and tier you choose (as there are HUGE cost differences).
* Each person should submit the lab individually. (Feel free to work together and help each other out, of course.)
* When you are done deploying the items (and collecting your screenshots for the lab), **DELETE YOUR RESOURCE GROUPS AND EVERYTHING YOU DEPLOYED**. Points will be deducted for excess costs incurred for failing to clean up after yourself.

## Create a GitHub Action and Deploy

1. In GitHub, create a new private repo called `dsba6190-<yourname>-deploymentlab`.
   1. Provide the URL to your repo:
2. Add in the Secrets for the UNCC Tenant and DSBA6190 Subscription
   * (Settings > Secrets and variables > Actions).
3. Add the instructor’s GitHub username (`colbyford`) to your Collaborators
   * (Settings > Collaborators).
4. Write your Terraform code in three files: main.tf, backend.tf, and variables.tf
   * Once you have these files, you’ll need to `terraform init` from your command line to initialize the Terraform environment around your .tf files.
   * Next, you can run `terraform fmt --recursive` to check that your .tf files are formatted correctly.
   * Run `az login` to login to your UNCC account. This is the credential that Terraform will use for local stuff.
   * Lastly, you can run `terraform plan` to have Terraform tell you what it is going to deploy.
5. Add your Terraform code to the repository. Then, commit and push those changes to GitHub.
   * The GitHub Action will run an addition Terraform command called `terraform apply` that will deploy the things the `plan` step said previously.
   * If you have issues with your GitHub Action pipeline failing, check your Secrets. Alternatively, you can run the `terraform apply` locally to see what is breaking.
6. Provide a screenshot of the successful deployment. (Replace the figure below.)



1. In Azure, check to see if your resources were deployed. Provide a screenshot of your Resource Group (replace the figure below).

Graphical user interface, text, application, email

Description automatically generated

## Show Me What You Deployed

1. Provide screenshots that show you have a Storage Account that that only accessible from the virtual network.
2. Provide screenshots that show you that your SQL server is only accessible from the virtual network.
3. How much should your resource group cost per month? ([Azure Calculator](https://azure.microsoft.com/en-us/pricing/calculator/))

## Draw an Architecture Diagram

Draw a simple architecture diagram that shows the Resource Group and all the services that were deployed into it. Be sure to include each of the service names.

* Feel free to use PowerPoint or Visio or some other tool to draw the diagram.
* Here’s the link to the Azure Icons: <https://learn.microsoft.com/en-us/azure/architecture/icons/>