Master Your Terraform Deployments with Azure DevOps

Jim Counts

jim.counts@solliance.net

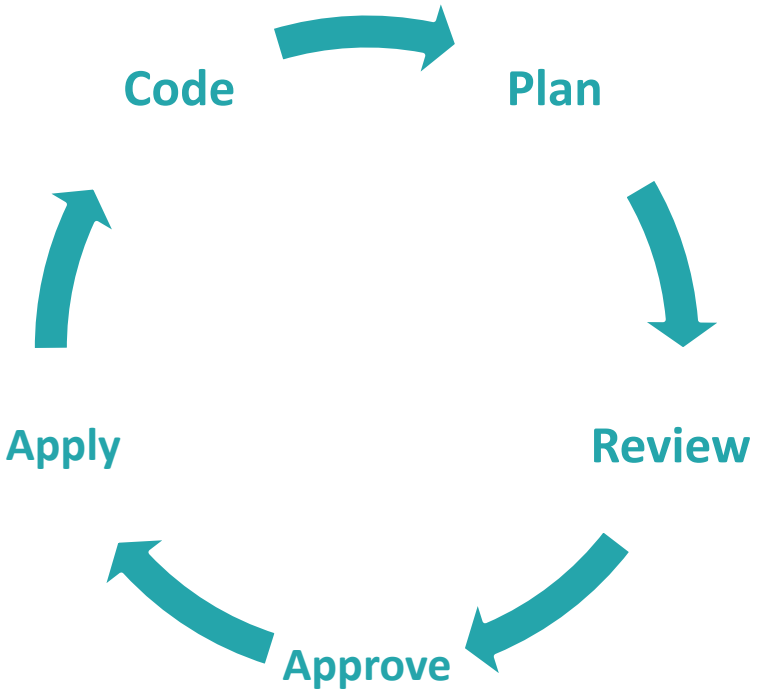
Introduction

- **Who am I?**
  - Cloud/DevOps Architect at Solliance
  - Background in C# programming
  - Messing around with CI/CD pipelines for ~10 years
- **What to cover**
  - How Terraform works
  - How to adapt Terraform to a deployment pipeline
  - How to get out of trouble

DEV intersection

Terraform 101

- Terraform is a tool to manage infrastructure as code (IaC)
- IaC can be as simple as a collection of scripts
- Terraform includes a sophisticated workflow engine



DEV intersection

Terraform 101

- Terraform uses Hashicorp Configuration Language (HCL)
- A bit of a cross between YAML and JSON

main.tf

```
44
45  # Create Network Security Group and rule
46  resource "azurerm_network_security_group" "nsg" {
47    name                = "myTFNSG"
48    location            = local.resource_location
49    resource_group_name = azurerm_resource_group.rg.name
50
51    security_rule {
52      name                       = "SSH"
53      priority                   = 1001
54      direction                  = "Inbound"
55      access                     = "Allow"
56      protocol                   = "Tcp"
57      source_port_range          = "*"
58      destination_port_range     = "22"
59      source_address_prefix      = "*"
60      destination_address_prefix = "*"
61    }
62  }
63
```

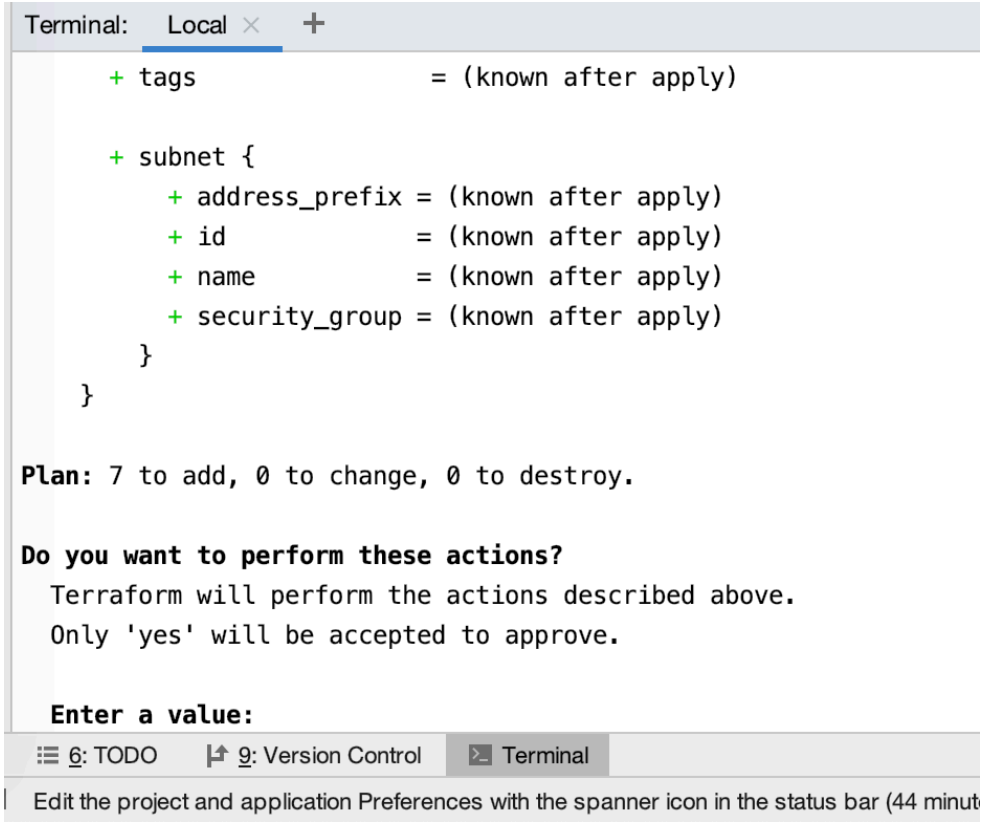
DEV intersection

Terraform Workflow

- What is the current state of the environment?
- What does this code say the environment should be?
- Here is my plan to get the environment into the desired state
- Should I apply it?
- Ok, I'm Applying it
- Save the new current state

DEV intersection

Terraform Safety is All About The Plan



- If the plan is safe, the deployment is safe
- Regarding safety, Terraform has no intelligence
- Terraform asks for approval before making changes
- Your responsibility to review the plan and provide approval

DEV intersection

Infrastructure to Deploy

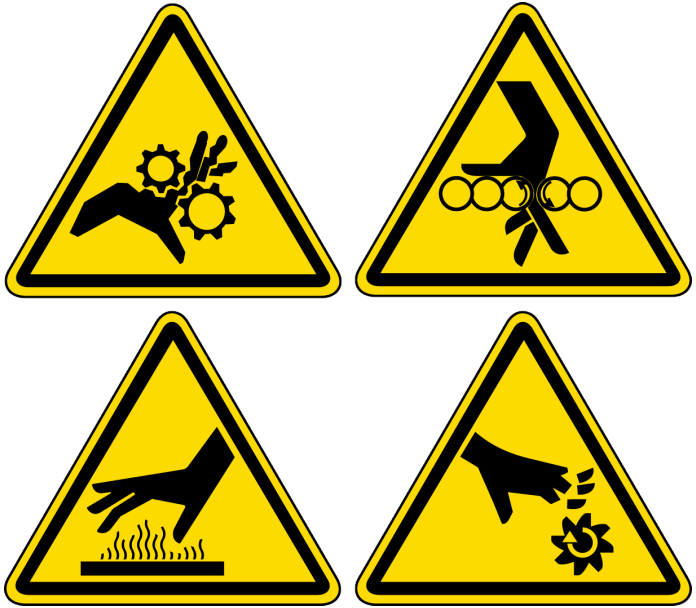
- A virtual machine
  - An Azure resource group
  - A virtual network
  - A subnet
  - A public IP
  - A network security group that allows SSH
  - A network interface

Demo: Terraform In Action

DEV intersection

Terraform Pipelines

- **Make Terraform non-interactive**
- **Without Sacrificing Safety**
- **Ways to make pipelines safe**
  - Feature Flags
  - Unit Tests
  - Static analysis
  - Very few options like this for IaC



DEV intersection

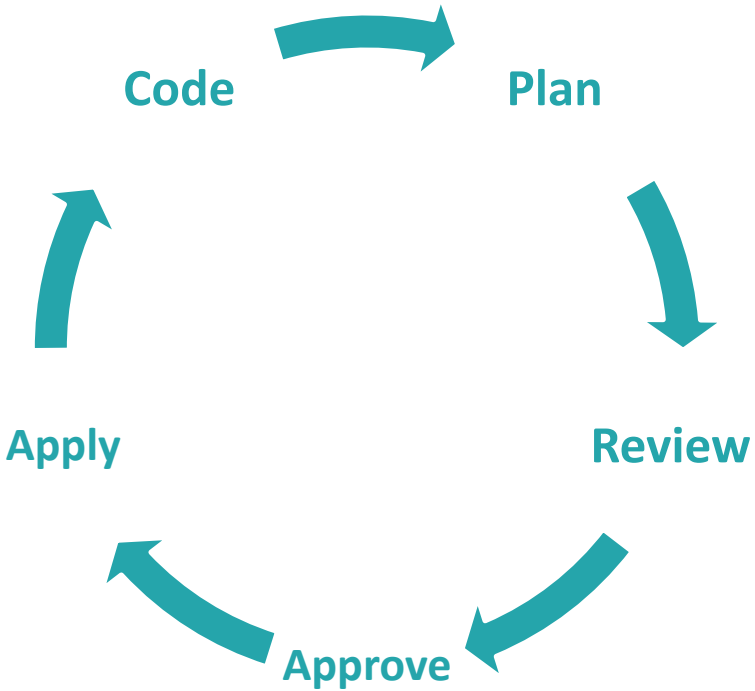
Prerequisite: Use Remote State

- **Build agents are ephemeral**
- **To avoid losing state, we must store it off the agent**
- **Extremely sensitive secrets**
- **Azure Storage Account Backend**
  - Encrypted at Rest, Role Based Access Control, Locking, Geo Replication, Soft deletes, Storage Account Firewall, Advanced Threat Protection, Logging and Monitoring, HTTPs Only Access

Demo: Setup Remote State

Terraform Pipelines Require Explicit Plans

- **Explicit plan files are the critical design feature for automation**
- **Safety—You can review an explicit plan**
- **Approval—Invoking Terraform with a plan file is the same as approving the plan**



DEV intersection

Creating a plan file

- Use the plan command
- Terraform's "plan" is just like apply…
- Except it makes no attempt to change infrastructure
- Use the "-out" parameter, to save the plan to a file

```
        + address_prefix = (known after apply)
        + id             = (known after apply)
        + name           = (known after apply)
        + security_group = (known after apply)
      }
  }

Plan: 7 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

This plan was saved to: /tmp/tfplan

To perform exactly these actions, run the following command to apply:
    terraform apply "/tmp/tfplan"

Cryptonomicon-3:terraform-getting-started-azure james$
```

6: TODO 9: Version Control Terminal

DEV intersection

Explicit Plan Files Are Approved Plans

- You must explicitly create the plan
- You must explicitly supply the plan to Terraform
- These two explicit actions add up to an approval
- Terraform skips the interactive prompt when using an explicit plan file

DEV intersection

Plan Review vs Code Review

- Code Reviews are important, but are up to interpretation
- The plan is the final word on what Terraform will try to do
- Terraform wins every* disagreement
- Actually… Azure has the final word.

Azure DevOps Build Stage

- Build Stages create artifacts
- Terraform Build Stages Produce Plan Files

- Steps
  - Download Terraform
  - Login
  - Run terraform init
  - Run terraform plan
  - Create build artifact including plan
  - Publish artifact for later use

DEV intersection

Demo: Terraform Build Stage

DEV intersection

Azure DevOps Deploy Stage

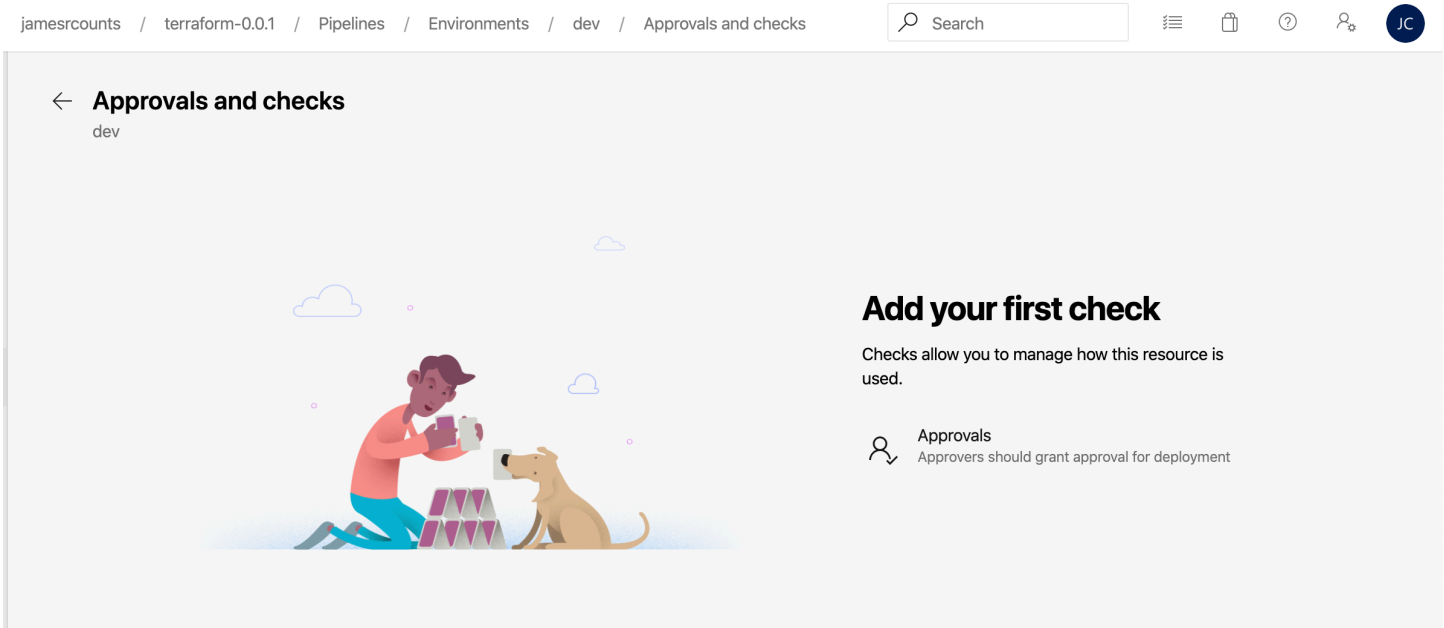
- Deploy stages consume/deploy build artifacts
- Terraform deploy stages apply explicit plan files created by Terraform build stages

- Steps
  - Download Terraform
  - Extract artifacts
  - Login
  - Run terraform apply

DEV intersection

Configure Environment to Support Approvals

- **Manual Approval Checks Force Deployment to Pause**



DEV intersection

```
Terraform Plan
------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # azurerm_network_interface.nic will be created
  + resource "azurerm_network_interface" "nic" {
      + applied_dns_servers           = (known after apply)
      + dns_servers                   = (known after apply)
      + enable_accelerated_networking = false
      + enable_ip_forwarding          = false
      + id                            = (known after apply)
      + internal_dns_name_label       = (known after apply)
      + internal_fqdn                 = (known after apply)
      + location                      = "centralus"
      + mac_address                   = (known after apply)
      + name                          = "myNIC"
      + network_security_group_id     = (known after apply)
      + private_ip_address            = (known after apply)
      + private_ip_addresses          = (known after apply)
      + resource_group_name           = "myTFResourceGroup"
      + tags                          = (known after apply)
      + virtual_machine_id            = (known after apply)

      + ip_configuration {
          + application_gateway_backend_address_pools_ids = (known af
          + application_security_group_ids                = (known af
          + load_balancer_backend_address_pools_ids       = (known af
          + load_balancer_inbound_nat_rules_ids           = (known af
```

Reviewing the Plan

- Explicit plan is not human readable
- Terraform ”show” can produce human readable output
- Much quicker to look at the build log

DEV intersection

Demo: Terraform Deploy Stage

DEV intersection

Troubleshooting

- **Targeting**

```
terraform plan \
    -destroy \
    -target=azurerm_network_interface.nic \
    -out /tmp/tfplan
```

- **State editing (rm, mv, pull, push)**
  - `terraform state rm azurerm_network_interface.nic`
  - *Careful with pull and push!*

DEV intersection

Troubleshooting

- **Importing**

```
terraform import \
   azurerm_network_interface.nic \
   /subscriptions/…/Microsoft.Network/networkInterfaces/myNIC
```

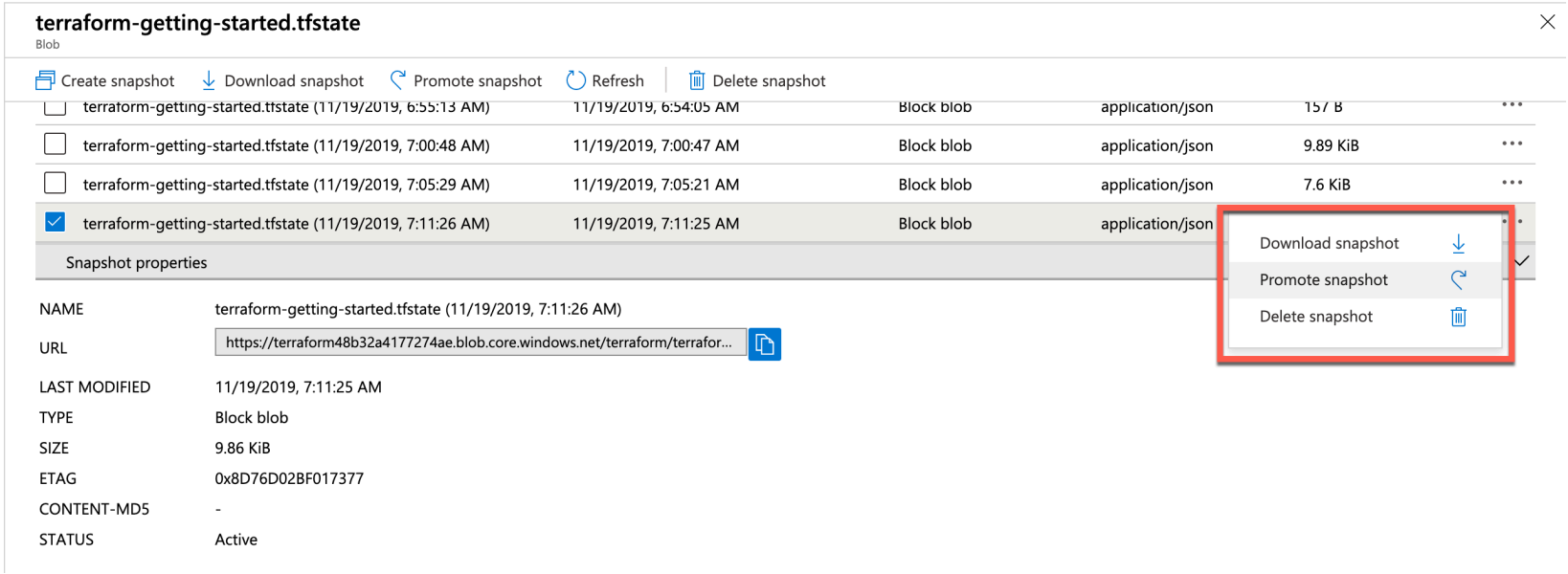
- **Infrastructure Editing**

```
az network nic delete \
   --resource-group myTFResourceGroup \
   --name myNIC
```

DEV intersection

Troubleshooting

- **State Rollback**



DEV intersection

- **Blog:** http://jamesrcounts.com/2019/10/14/azdo-safe-terraform-pipelines.html
- **Code:** https://github.com/jamesrcounts/terraform-getting-started-azure

DEV intersection

Please use EventsXD to fill out a session evaluation.

Thank you!

DEV intersection