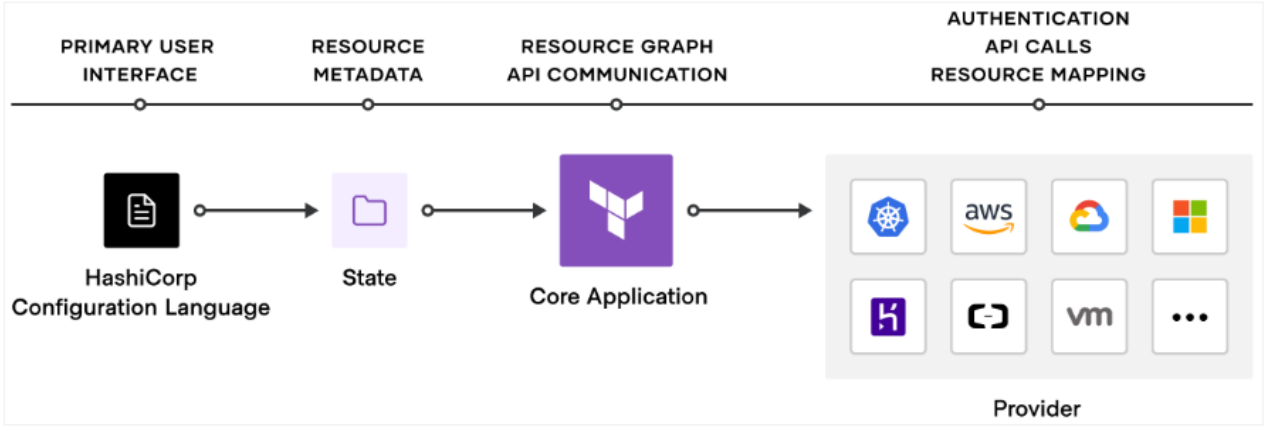
Terraform Troubleshooting

Terraform Debugging

- [Official Docs: Debugging Terraform]
  - Logging
  - Crash Log Interpretation

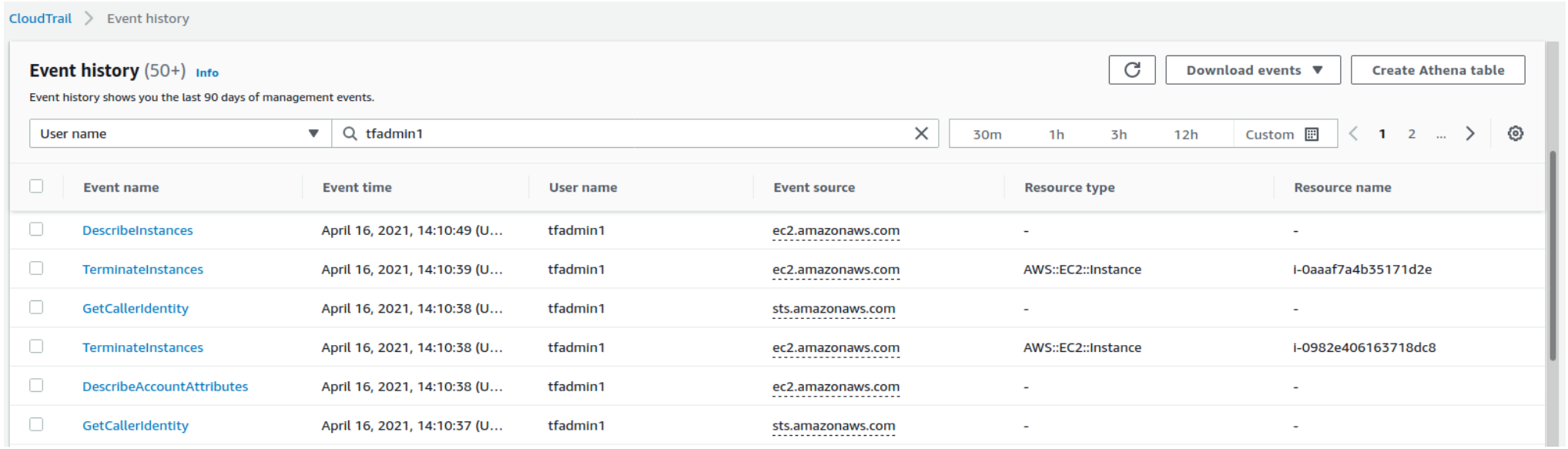


Contents

- AWS CloudTrail (or equivalent)
- Terraform Logging
- Pinning versions of Terraform, Providers and Modules – recommended practices
- Using Terraform Console
- Using Proxies
- Performance Tuning

AWS CloudTrail

- Use CloudTrail Event History to know, analyze, and understand how Terraform uses the AWS SDK to configure infrastructure
  - In addition to fine tuning IAM policies for security as described in other modules, CloudTrail gives us look "under the hood"
- Sample CloudTrail Event Log, filtering by the IAM user name used by terraform (e.g. "tfadmin1")



Cloud Credentials Problems

- Verify API Calls
  - AWS CloudTrail Event Registry or equivalent for other providers
- Verify what identity we are using
  - aws sts get-user-identity [–profile <someprofile>]
  - This will give us the user or role we are using
    - Then verify the IAM permissions associated with the user or role
    - If no access to view IAM info (often the case): provide the user/role info to the AWS Admin with access to IAM

Terraform Logging

- Set Environment Variables :
  - **TF_LOG:** TRACE | DEBUG | INFO | WARN | ERROR
  - It is possible to enable logs separately for Terraform Core (TF_LOG_CORE) and Terraform providers (**TF_LOG_PROVIDER**) - same values as TF_LOG
  - TF_LOG_PROVIDER=DEBUG particularly useful for seeing AWS API calls
- **TF_LOG_PATH**:
  - path to log file – note that the variable name can be confusing: value must include path and file name, not just path.
  - `export TF_LOG_PATH=/var/log/terraform.log`

```
rafa@rp3:~$ more /tmp/terraform.log
2021/04/15 13:04:56 [INFO] Terraform version: 0.14.7
2021/04/15 13:04:56 [INFO] Go runtime version: go1.15.6
2021/04/15 13:04:56 [INFO] CLI args: []string{"/usr/local/bin/terraform", "plan"}
2021/04/15 13:04:56 [DEBUG] Attempting to open CLI config file: /home/rafa/.terraformrc
2021/04/15 13:04:56 Loading CLI configuration from /home/rafa/.terraformrc
2021/04/15 13:04:56 Loading CLI configuration from /home/rafa/.terraform.d/credentials.tfrc.json
2021/04/15 13:04:56 [DEBUG] ignoring non-existing provider search directory terraform.d/plugins
2021/04/15 13:04:56 [DEBUG] ignoring non-existing provider search directory /home/rafa/.terraform.d/plugins
2021/04/15 13:04:56 [DEBUG] ignoring non-existing provider search directory /home/rafa/.local/share/terraform
2021/04/15 13:04:56 [DEBUG] ignoring non-existing provider search directory /usr/share/plasma/terraform/plug
2021/04/15 13:04:56 [DEBUG] ignoring non-existing provider search directory /usr/local/share/terraform/plugin
2021/04/15 13:04:56 [DEBUG] ignoring non-existing provider search directory /usr/share/terraform/plugins
2021/04/15 13:04:56 [DEBUG] ignoring non-existing provider search directory /var/lib/snapd/desktop/terraform/
2021/04/15 13:04:56 [INFO] CLI command args: []string{"plan"}
2021/04/15 13:04:56 [INFO] Checkpoint disabled. Not running.
2021/04/15 13:04:56 [TRACE] Meta.Backend: no config given or present on disk, so returning nil config
2021/04/15 13:04:56 [TRACE] Meta.Backend: backend has not previously been initialized in this working directo
2021/04/15 13:04:56 [DEBUG] New state was assigned lineage "edd4bf3f-4a49-ebec-17e1-a1cf5b93b532"
2021/04/15 13:04:56 [TRACE] Meta.Backend: using default local state only (no backend configuration, and no ex
2021/04/15 13:04:56 [TRACE] Meta.Backend: instantiated backend of type <nil>
2021/04/15 13:04:56 [TRACE] providercache.fillMetaCache: scanning directory .terraform/providers
2021/04/15 13:04:56 [TRACE] getproviders.SearchLocalDirectory: .terraform/providers is a symlink to .terrafor
2021/04/15 13:04:56 [TRACE] getproviders.SearchLocalDirectory: found registry.terraform.io/hashicorp/aws v3.3
2021/04/15 13:04:56 [TRACE] getproviders.SearchLocalDirectory: found registry.terraform.io/hashicorp/google v
2021/04/15 13:04:56 [TRACE] providercache.fillMetaCache: including .terraform/providers/registry.terraform.io
2021/04/15 13:04:56 [TRACE] providercache.fillMetaCache: including .terraform/providers/registry.terraform.io
2021/04/15 13:04:56 [TRACE] providercache.fillMetaCache: using cached result from previous scan of .terraform
2021/04/15 13:04:56 [DEBUG] checking for provisioner in "."
2021/04/15 13:04:56 [DEBUG] checking for provisioner in "/usr/local/bin"
2021/04/15 13:04:56 [INFO] Failed to read plugin lock file .terraform/plugins/linux_amd64/lock.json: open .te
2021/04/15 13:04:56 [TRACE] Meta.Backend: backend <nil> does not support operations, so wrapping it in a loca
2021/04/15 13:04:56 [INFO] backend/local: starting Plan operation
2021/04/15 13:04:56 [TRACE] backend/local: requesting state manager for workspace "default"
2021/04/15 13:04:56 [TRACE] backend/local: state manager for workspace "default" will:
 - read initial snapshot from terraform.tfstate
 - write new snapshots to terraform.tfstate
 - create any backup at terraform.tfstate.backup
2021/04/15 13:04:56 [TRACE] backend/local: requesting state lock for workspace "default"
2021/04/15 13:04:56 [TRACE] statemgr.Filesystem: preparing to manage state snapshots at terraform.tfstate
2021/04/15 13:04:56 [TRACE] statemgr.Filesystem: existing snapshot has lineage "27895975-b17c-4230-2fdc-2f924
2021/04/15 13:04:56 [TRACE] statemgr.Filesystem: locking terraform.tfstate using fcntl flock
2021/04/15 13:04:56 [TRACE] statemgr.Filesystem: writing lock metadata to .terraform.tfstate.lock.info
```

A Useful Combination of Logging values to see AWS API calls

```
export TF_LOG_PROVIDER=DEBUG
export TF_LOG_PATH=/tmp/terraform.log
# Most API calls start with "Action="
tail -f /tmp/terraform.log | grep "Action="
```

```
rafa@rp3:tf_debug$ terraform refresh
aws_security_group.sec_ssh_ping: Refreshing state... [id=sg-09ba603b159d15389]
aws_instance.test_vm: Refreshing state... [id=i-0479c35264675f79d]

Outputs:

ami = "ami-0076b212fad243d9e"
instance_id = "i-0479c35264675f79d"
key_name = "tf-course"
public_ip = "3.249.248.200"
user_identity = {
  "account_id" = "044858806836"
  "arn" = "arn:aws:iam::044858806836:user/tfadmin1"
  "id" = "044858806836"
  "user_id" = "AIDAQU4OLCI2HFOYIHHAG"
}
```

grep filter does not work properly as of Terraform 1.4.x, since debug captures more call info. Kept for historical reasons - See next slide

```
rafa@rp3:tf_debug$ tail -f /tmp/terraform.log | grep "Action="
Action=GetCallerIdentity&Version=2011-06-15: timestamp=2022-05-12T10:59:01.897+02
Action=GetCallerIdentity&Version=2011-06-15: timestamp=2022-05-12T10:59:02.295+02
Action=DescribeAccountAttributes&AttributeName.1=supported-platforms&Version=2016-
Action=GetCallerIdentity&Version=2011-06-15
Action=DescribeVpcs&Filter.1.Name=isDefault&Filter.1.Value.1=true&Version=2016-11-
Action=DescribeImages&Filter.1.Name=name&Filter.1.Value.1=ubuntu%2Fimages%2Fhvm-s
Action=DescribeVpcAttribute&Attribute=enableDnsHostnames&Version=2016-11-15&VpcId=
Action=DescribeVpcAttribute&Attribute=enableDnsSupport&Version=2016-11-15&VpcId=vpc-0f68ba3ea7b781973
Action=DescribeRouteTables&Filter.1.Name=association.main&Filter.1.Value.1=true&Filter.2.Name=vpc-id&Filter.2.Value.1=vpc-0f68ba3ea7b781973&Version=2016-11-15
Action=DescribeSubnets&Filter.1.Name=vpc-id&Filter.1.Value.1=vpc-0f68ba3ea7b781973&Version=2016-11-15
Action=DescribeSecurityGroups&GroupId.1=sg-09ba603b159d15389&Version=2016-11-15
Action=DescribeInstances&InstanceId.1=i-0479c35264675f79d&Version=2016-11-15
Action=DescribeTags&Filter.1.Name=key&Filter.1.Value.1=aws%3Aec2launchtemplate%3Aid&Filter.2.Name=resource-id&Filter.2.Value.1=i-0479c35264675f79d&Version=2016-11-15
Action=DescribeVpcs&Version=2016-11-15&VpcId.1=vpc-0f68ba3ea7b781973
Action=DescribeInstanceAttribute&Attribute=instanceInitiatedShutdownBehavior&InstanceId=i-0479c35264675f79d&Version=2016-11-15
Action=DescribeVolumes&Version=2016-11-15&VolumeId.1=vol-0bdbff8486e8ed8f0
Action=DescribeInstanceAttribute&Attribute=disableApiTermination&InstanceId=i-0479c35264675f79d&Version=2016-11-15
Action=DescribeInstanceAttribute&Attribute=userData&InstanceId=i-0479c35264675f79d&Version=2016-11-15
Action=DescribeInstanceCreditSpecifications&InstanceId.1=i-0479c35264675f79d&Version=2016-11-15
```

Useful Combination of Logging values to see AWS API calls

```
export TF_LOG_PROVIDER=DEBUG
export TF_LOG_PATH=terraform.log
```

Updated grep filter for recent Terraform versions that capture more info per line.
TODO: capture timestamp

```
$ tail -f terraform.log | grep -o Action.*\&
```

```
(base) rp4:lab_06_ec2_foreach$ terraform plan -refresh
data.aws_ami.amazon_linux2_kernel_5: Reading...
data.aws_vpc.def_vpc: Reading...
data.aws_caller_identity.current: Reading...
data.aws_ami.ubuntu_20_04: Reading...
data.aws_ami_ids.ubuntu_amis: Reading...
data.aws_ami.ubuntu_22_04: Reading...
data.aws_caller_identity.current: Read complete after 0s [id=975030449833]
data.aws_ami_ids.ubuntu_amis: Read complete after 0s [id=1596998940]
data.aws_ami.ubuntu_22_04: Read complete after 0s [id=ami-0f9ae27ecf629cbe3]
data.aws_ami.ubuntu_20_04: Read complete after 0s [id=ami-0e3f7dd2dc743e48a]
data.aws_ami.amazon_linux2_kernel_5: Read complete after 1s [id=ami-08fea9e08576c443b]
data.aws_vpc.def_vpc: Read complete after 1s [id=vpc-0421afec2aa04ce61]
data.aws_subnets.def_vpc_subnets: Reading...
aws_security_group.sec_web: Refreshing state... [id=sg-098f0de67c3c18fbb]
data.aws_subnets.def_vpc_subnets: Read complete after 0s [id=eu-west-1]
aws_instance.server2["department 4"]: Refreshing state... [id=i-082f7defb6cbaad70]
aws_instance.server2["dep2"]: Refreshing state... [id=i-04044ed71f14d9afb]
aws_instance.server2["dep1"]: Refreshing state... [id=i-0a13358142e5ae3fd]
aws_instance.server2["dep3"]: Refreshing state... [id=i-041566ac6387a7ea8]
```

```
Action=GetCallerIdentity&
Action=GetCallerIdentity&
Action=GetCallerIdentity&
Action=DescribeImages&Filter.1.Name=name&Filter.1.Value.1=amzn2-ami-kernel-5.10-hvm-2.0%2Ax86_64-gp2&Filter.2.Name=architecture&Filter.2.Value.1=x86_64&Filter.3.Name=virtualizati
on-type&Filter.3.Value.1=hvm&IncludeDeprecated=false&Owner.1=amazon&
Action=DescribeVpcs&Filter.1.Name=isDefault&Filter.1.Value.1=true&
Action=DescribeImages&Filter.1.Name=name&Filter.1.Value.1=ubuntu%2Fimages%2Fhvm-ssd%2Fubuntu-jammy-22.04-amd64-server%2A&Filter.2.Name=virtualization-type&Filter.2.Value.1=hvm&In
cludeDeprecated=false&Owner.1=099720109477&
Action=DescribeImages&Filter.1.Name=name&Filter.1.Value.1=ubuntu%2Fimages%2Fhvm-ssd%2Fubuntu-focal-20.04-amd64-server-%2A&IncludeDeprecated=false&Owner.1=099720109477&
Action=DescribeImages&Filter.1.Name=name&Filter.1.Value.1=ubuntu%2Fimages%2Fubuntu-%2A-%2A-amd64-server-%2A&Owner.1=099720109477&
Action=DescribeVpcAttribute&Attribute=enableDnsHostnames&Version=2016-11-15&
Action=DescribeVpcAttribute&Attribute=enableDnsSupport&Version=2016-11-15&
Action=DescribeVpcAttribute&Attribute=enableNetworkAddressUsageMetrics&Version=2016-11-15&
Action=DescribeRouteTables&Filter.1.Name=association.main&Filter.1.Value.1=true&Filter.2.Name=vpc-id&Filter.2.Value.1=vpc-0421afec2aa04ce61&
Action=DescribeSubnets&Filter.1.Name=vpc-id&Filter.1.Value.1=vpc-0421afec2aa04ce61&
Action=DescribeSecurityGroups&GroupId.1=sg-098f0de67c3c18fbb&
Action=DescribeInstances&InstanceId.1=i-041566ac6387a7ea8&
Action=DescribeInstances&InstanceId.1=i-0a13358142e5ae3fd&
Action=DescribeInstances&InstanceId.1=i-082f7defb6cbaad70&
Action=DescribeInstances&InstanceId.1=i-04044ed71f14d9afb&
Action=DescribeInstanceTypes&InstanceType.1=t3.micro&
Action=DescribeInstanceTypes&InstanceType.1=t3.micro&
```

Another Log example – terraform destroy

terraform destroy for a configuration with an EC2 instance and associated security group
Note "polling" by the AWS Provider after TerminateInstances API call. Security group is destroyed only after Terraform verifies that instances and interfaces are destroyed

```
Action=DescribeInstanceAttribute&Attribute=userData&InstanceId=i-04044ed71f14d9afb&
Action=DescribeInstanceCreditSpecifications&InstanceId.1=i-041566ac6387a7ea8&
Action=DescribeInstanceCreditSpecifications&InstanceId.1=i-082f7defb6cbaad70&
Action=DescribeInstanceCreditSpecifications&InstanceId.1=i-0a13358142e5ae3fd&
Action=DescribeInstanceCreditSpecifications&InstanceId.1=i-04044ed71f14d9afb&
Action=GetCallerIdentity&
Action=GetCallerIdentity&
Action=GetCallerIdentity&
Action=GetCallerIdentity&
Action=ModifyInstanceAttribute&DisableApiTermination.Value=false&InstanceId=i-041566ac6387a7ea8&
Action=ModifyInstanceAttribute&DisableApiTermination.Value=false&InstanceId=i-0a13358142e5ae3fd&
Action=ModifyInstanceAttribute&DisableApiTermination.Value=false&InstanceId=i-04044ed71f14d9afb&
Action=ModifyInstanceAttribute&DisableApiTermination.Value=false&InstanceId=i-082f7defb6cbaad70&
Action=TerminateInstances&InstanceId.1=i-0a13358142e5ae3fd&
Action=TerminateInstances&InstanceId.1=i-082f7defb6cbaad70&
Action=TerminateInstances&InstanceId.1=i-04044ed71f14d9afb&
Action=TerminateInstances&InstanceId.1=i-041566ac6387a7ea8&
Action=DescribeInstances&InstanceId.1=i-0a13358142e5ae3fd&
Action=DescribeInstances&InstanceId.1=i-04044ed71f14d9afb&
Action=DescribeInstances&InstanceId.1=i-041566ac6387a7ea8&
Action=DescribeInstances&InstanceId.1=i-082f7defb6cbaad70&
Action=DescribeInstances&InstanceId.1=i-0a13358142e5ae3fd&
Action=DescribeInstances&InstanceId.1=i-04044ed71f14d9afb&
```

```
Action=DescribeInstances&InstanceId.1=i-082f7defb6cbaad70&
Action=DescribeInstances&InstanceId.1=i-082f7defb6cbaad70&
Action=DescribeNetworkInterfaces&Filter.1.Name=description&Filter.1.Value.1=AWS+Lambda+VPC+ENI%2A&Filter.2.Name=group-id&Filter.2.Value.1=sg-098f0de67c3c18fbb&
Action=DescribeNetworkInterfaces&Filter.1.Name=group-id&Filter.1.Value.1=sg-098f0de67c3c18fbb&
Action=DeleteSecurityGroup&GroupId=sg-098f0de67c3c18fbb&
Action=DescribeSecurityGroups&GroupId.1=sg-098f0de67c3c18fbb&
```

```
aws_instance.server2["dep2"]: Destroying... [id=i-04044ed71f14d9afb]
aws_instance.server2["dep1"]: Destroying... [id=i-0a13358142e5ae3fd]
aws_instance.server2["dep3"]: Destroying... [id=i-041566ac6387a7ea8]
aws_instance.server2["department 4"]: Destroying... [id=i-082f7defb6cbaad70]
aws_instance.server2["dep3"]: Still destroying... [id=i-041566ac6387a7ea8, 10s elapsed]
aws_instance.server2["dep2"]: Still destroying... [id=i-04044ed71f14d9afb, 10s elapsed]
aws_instance.server2["dep1"]: Still destroying... [id=i-0a13358142e5ae3fd, 10s elapsed]
aws_instance.server2["department 4"]: Still destroying... [id=i-082f7defb6cbaad70, 10s elapsed]
aws_instance.server2["dep1"]: Still destroying... [id=i-0a13358142e5ae3fd, 20s elapsed]
aws_instance.server2["dep3"]: Still destroying... [id=i-041566ac6387a7ea8, 20s elapsed]
aws_instance.server2["dep2"]: Still destroying... [id=i-04044ed71f14d9afb, 20s elapsed]
aws_instance.server2["department 4"]: Still destroying... [id=i-082f7defb6cbaad70, 20s elapsed]
aws_instance.server2["dep2"]: Still destroying... [id=i-04044ed71f14d9afb, 30s elapsed]
aws_instance.server2["dep3"]: Still destroying... [id=i-041566ac6387a7ea8, 30s elapsed]
aws_instance.server2["dep1"]: Still destroying... [id=i-0a13358142e5ae3fd, 30s elapsed]
aws_instance.server2["department 4"]: Still destroying... [id=i-082f7defb6cbaad70, 30s elapsed]
aws_instance.server2["dep1"]: Destruction complete after 31s
aws_instance.server2["department 4"]: Still destroying... [id=i-082f7defb6cbaad70, 40s elapsed]
aws_instance.server2["dep2"]: Still destroying... [id=i-04044ed71f14d9afb, 40s elapsed]
aws_instance.server2["dep3"]: Still destroying... [id=i-041566ac6387a7ea8, 40s elapsed]
aws_instance.server2["dep3"]: Still destroying... [id=i-041566ac6387a7ea8, 50s elapsed]
aws_instance.server2["department 4"]: Still destroying... [id=i-082f7defb6cbaad70, 50s elapsed]
aws_instance.server2["dep2"]: Still destroying... [id=i-04044ed71f14d9afb, 50s elapsed]
aws_instance.server2["dep2"]: Destruction complete after 51s
aws_instance.server2["dep3"]: Destruction complete after 51s
aws_instance.server2["department 4"]: Still destroying... [id=i-082f7defb6cbaad70, 1m0s elapsed]
aws_instance.server2["department 4"]: Still destroying... [id=i-082f7defb6cbaad70, 1m10s elapsed]
aws_instance.server2["department 4"]: Still destroying... [id=i-082f7defb6cbaad70, 1m20s elapsed]
                                      partment 4"]: Still destroying... [id=i-082f7defb6cbaad70, 1m30s elapsed]
                                      partment 4"]: Still destroying... [id=i-082f7defb6cbaad70, 1m40s elapsed]
                                      partment 4"]: Still destroying... [id=i-082f7defb6cbaad70, 1m50s elapsed]
                                      partment 4"]: Still destroying... [id=i-082f7defb6cbaad70, 2m0s elapsed]
                                      partment 4"]: Still destroying... [id=i-082f7defb6cbaad70, 2m10s elapsed]
                                      partment 4"]: Still destroying... [id=i-082f7defb6cbaad70, 2m20s elapsed]
aws_instance.server2["department 4"]: Still destroying... [id=i-082f7defb6cbaad70, 2m30s elapsed]
aws_instance.server2["department 4"]: Destruction complete after 2m32s
aws_security_group.sec_web: Destroying... [id=sg-098f0de67c3c18fbb]
aws_security_group.sec_web: Destruction complete after 1s
```

Terraform Core, Provider and Module versions

- Controlling the versions can help avoid unexpected side-effects ("preventive debugging")
- Best practices from Terraform documentation on version constraints:
  - Module Versions
    - When depending on **third-party** modules, require **specific versions** to ensure that updates only happen when convenient to you.
    - For modules **maintained within your organization**, specifying **version ranges** may be appropriate if semantic versioning is used consistently or if there is a well-defined release process that avoids unwanted updates.
  - Terraform Core and Provider Versions
    - **Reusable modules** should constrain only their minimum allowed versions of Terraform and providers, such as >= 0.12.0. This helps avoid known incompatibilities, while allowing the user of the module flexibility to upgrade to newer versions of Terraform without altering the module.
    - **Root modules** should use a ~> constraint to set both a lower and upper bound on versions for each provider they depend on.
- See also similar recommendations in CloudPosse best practices : use minimum version pinning on all providers
- See also ref to .terraform.lock.hcl file (previous chapters of the course)

```
module "first_vnet" {
  source              = "Azure/vnet/azurerm"
  version             = "2.7.0" # In production use always a
  vnet_name           = "vnet-${local.name_suffix}"
  resource_group_name = azurerm_resource_group.rg.name
```

```
version = ">= 1.2.0, < 2.0.0"
```

```
terraform {
  required_version = "~>1.2.0"

  required_providers {
    azurerm = {
      #source = "registry.terraform.io/hashicorp/azurerm"
      source  = "hashicorp/azurerm"
      version = "~>3.0"
    }
    aws = {
      source  = "hashicorp/aws"
      version = "~>4.0"
    }
    gcp = {
      source = "hashicorp/google"
      version = "~>4.0"
    }
    random = {
      source  = "hashicorp/random"
      version = "~>3.3"
    }
  }
}
```

terraform console

- terraform console – as discussed during the course
  - Note : it does lock the state file, so in principle use piping commands to terraform console

Using an HTTP/S Proxy

- Intercept all communication between TF in machine and target (e.g. AWS)
- Examples
  - Charles ($)
  - Fiddler ($)
  - Mitmproxy
- API tracing in CloudTrail and/or TF logging may be enough in most cases

Other suggestions

- Always run plan before apply
  - … and read the plan output
- Integrate plan into code review flow

Performance Tuning

- AWS may throttle API rate but this is normally not noticeable
- Consider using plugin_cache_dir option in .terraformrc
  - Saves significant disk space
  - Could avoid downloading modules/providers
- "-refresh=false" flag in terraform apply if you are sure that you can avoid checking the remote system for the latest config
- Explore increasing with flag *parallelism=n* with n > 10 (default) for apply, plan