

Terraform Tips and Tricks

Nelson Calero

Pythian Principal Consultant

October, 2022





Nelson Calero

Pythian Principal Consultant

- 20+ years Oracle database experience
- Oracle ACE Director
- Montevideo, Uruguay

LinkedIn: <http://www.linkedin.com/in/ncalero>

Twitter: @ncalerouy



Uruguay



Solutions designed to power the value of your data.

Solution that maximize
the entire data state

Data & Analytics	CloudOps	Applications
Data & Analytics Consulting	Cloud Migrations	Oracle EBS
Database Services	Cloud Management – InfraOps / FinOps	SAP
Cloud Data Platform Services	Google Cloud	Collaboration & Workspace
Data Insights & Advanced Analytics	DevOps – CI/CD	Application Development, Migration & Refactoring

Wherever your data and
applications reside



Public Cloud



Hybrid Cloud



On Premises

No matter what platform
you choose



25

Years in Business

420+

Experts across every Data Domain & Technology

400+

Global Customers



500+ technical experts helping peers globally

The **Oracle ACE Program** recognizes and rewards community members for their technical and community contributions to the Oracle community

3 membership tiers



For more details on Oracle ACE Program:
ace.oracle.com



Nominate
yourself or someone you know:

ace.oracle.com/nominate



Agenda

- Terraform basics (1 slide)
- Terraform Idiosyncrasy
- Cloud provider Idiosyncrasy



Terraform basics

- Infrastructure as Code, declarative config files (HCL)
- Simple install - standalone binary file
- Few operations - plan/apply/destroy
- Providers - implement how to deal with your resources. 1000+
- State file - mapping of existing resources to TF configuration
- Simple structure
 - Reads code from current directory
 - Split code in files for readability: main.tf, variables.tf, outputs.tf
- Advanced configurations - workspaces, modules, remote state files
- Changes in versions - Features, state file format

Terraform basics

- Infrastructure as Code, declarative config files (HCL)
- Simple install - standalone binary file
- Few operations - plan/apply/destroy
- Providers - implement how to deal with your resources.
- State file - mapping of existing resources to TF configuration
- Simple structure
 - Reads code from current directory
 - Split code in files for readability: main.tf, variables.tf, outputs.tf
- Advanced configurations - workspaces, modules, remote state
- Changes in versions - Features, state file format
- <https://www.terraform.io/>
- Lucas' blog:

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 3.27"
    }
  }
  required_version = ">= 0.14.9"
}

provider "aws" {
  profile = "default"
  region = "us-west-1"
}

resource "aws_instance" "app_server" {
  ami           = "ami-830c94e3"
  instance_type = "t2.micro"

  tags = {
    Name = "AppServerInstance"
  }
}
```

Terraform idiosyncrasy

- 1) State file relevance
- 2) Concurrent work on same resources
- 3) Reducing scope of operations
- 4) Resource parameters validation
- 5) Re-using code
- 6) Available features = TF + provider

Terraform idiosyncrasy - 1) state file

Demo 1

Code taken from public Oracle repo (few changes for simplicity):

<https://github.com/oracle/terraform-provider-oci/tree/master/examples/database/adb>

Terraform idiosyncrasy - state file

```
$ terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...
```

```
The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.
```

```
data.aws_security_group.sgpr: Refreshing state...
```

```
-----  
An execution plan has been generated and is shown below.
```

```
Resource actions are indicated with the following symbols:
```

```
  + create
```

```
Terraform will perform the following actions:
```

```
  # aws_db_instance.master will be created
```

```
  + resource "aws_db_instance" "master" {
```

```
    + address                               = (known after apply)
```

```
    + allocated_storage                     = 500
```

```
    + apply_immediately                     = (known after apply)
```

```
  ...
```

```
    + identifier                            = "nelson-pg14"
```

```
  ...
```

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

Terraform idiosyncrasy - state file

```
$ terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...  
The refreshed state will be used to calculate this plan, but will not be  
persisted to local or remote state storage.
```

```
data.aws_security_group.sgpr: Refreshing state...
```

```
-----  
An execution plan has been generated and is shown below.  
Resource actions are indicated with the following symbols:  
+ create  
Terraform will perform the following actions:  
  
$ aws rds describe-db-instances --region=us-west-1 --db-instance-identifier nelson-pg14  
--query  
"*[].[DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,EngineVersion:EngineVersion]" --output table
```

```
Terraform
```

```
# aws_rds_db_instance
```

```
+ resource "aws_rds_db_instance"
```

```
+ {
```

```
+   db_instance_class = "db.m5.large"
```

```
+   db_instance_identifier = "nelson-pg14"
```

```
+   db_instance_profile_name = "aws-elasticmapreduce-profile"
```

```
+   db_subnet_group_name = "aws-elasticmapreduce-subnet-group"
```

```
+   engine = "postgres"
```

```
+   engine_version = "12.7" -->
```

```
+   instance_class = "db.m5.large"
```

```
+   name = "nelson-pg14" -->
```

```
-----  
| DescribeDBInstances |  
+-----+-----+-----+  
| DBClusterIdentifier | DBInstanceIdentifier | EngineVersion |  
+-----+-----+-----+  
| None | nelson-ora19 | 19.0.0.0.ru-2021-07.rur-2021-07.r1 |  
+-----+-----+-----+
```

```
+ identifier = "nelson-pg14"
```

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


Terraform idiosyncrasy - state file

```
$ terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.

data.aws_security_group.sgpr: Refreshing state...
aws_db_instance.master: Refreshing state... [id=nelson-ora19]
```

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
+ create
~ update in-place
- destroy

Terraform will perform the following actions:
```

```
# aws_db_instance.master: Refreshing state... [id=nelson-ora19]
+ resource "aws_db_instance" "master" {
  + identifier = "db-ora19"
  + instance_class = "db.m4.xlarge"
  + instance_profile_name = "aws-ec2-profile"
  + kms_key_id = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
  + multi_az = true
  + parameter_group_name = "default:oracle-ee"
  + performance_insights_enabled = true
  + storage_type = "gp2"
  + storage_encrypted = true
  + storage_size = 100
  + vpc_security_group_ids = ["sg-12345678"]
}
```

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

```
$ terraform import aws_db_instance.master
```

```
aws_db_instance.master
Prepared aws_db_instance.master
aws_db_instance.master
```

```
Import successful!
The resources that were imported are listed above with the
IDs they were imported with. You must perform this operation
for each resource. Please perform this operation only once.
```

```
$ terraform plan
```

```
$ terraform apply
```

```
$ terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.
```

```
data.aws_security_group.sgpr: Refreshing state...
aws_db_instance.master: Refreshing state... [id=nelson-ora19]
```

```
No changes. Infrastructure is up-to-date.
```

```
This means that Terraform did not detect any differences between your configuration and real physical resources that exist. As a result, no actions need to be performed.
```

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


Terraform idiosyncrasy - state file

- Default state file is *terraform.tfstate* in current directory
- Critical data, requires good backup policy
- *terraform import* is not always able to recreate all attributes (ex: passwords, MySQL grants)
- Two key complementary features:
 - remote state file
<https://www.terraform.io/language/state/remote>
 - OCI discovery
https://registry.terraform.io/providers/oracle/oci/latest/docs/guides/resource_discovery

Terraform idiosyncrasy - state file / remote

- Remote state file - <https://www.terraform.io/language/state/remote>

```
remote_state {
  backend "s3" {
    bucket     = "terraform-states"
    key        = "networking/terraform.tfstate"
    region     = "us-phoenix-1"
    endpoint   = "https://acme.compat.objectstorage.us-phoenix-1.oraclecloud.com"
    shared_credentials_file = "../terraform-states_bucket_credentials"
    skip_region_validation   = true
    skip_credentials_validation = true
    skip_metadata_api_check  = true
    force_path_style         = true
  }
}
```

Ref: <https://docs.oracle.com/en-us/iaas/Content/API/SDKDocs/terraformUsingObjectStore.htm>

Terraform idiosyncrasy - state file / OCI discovery

Demo 2

OCI discovery docs: https://registry.terraform.io/providers/oracle/oci/latest/docs/guides/resource_discovery

Terraform idiosyncrasy - state file / OCI discovery

- *terraform-provider-oci* gets installed by terraform:

```
$ ls -lrt .terraform/providers/registry.terraform.io/hashicorp/oci/4.77.0/linux_amd64/
total 129320
-rw-r--r-- 1 calero calero      96384 May 31 17:31 CHANGELOG.md
-rw-r--r-- 1 calero calero      16725 May 31 17:31 LICENSE.md
-rw-r--r-- 1 calero calero       3071 May 31 17:31 README.md
-rwxr-xr-x 1 calero calero 132177920 May 31 17:31 terraform-provider-oci_v4.77.0
```

- Discover databases and create the state file:

```
$ mkdir discover
$ .terraform/providers/registry.terraform.io/hashicorp/oci/4.77.0/linux_amd64/terraform-provider-oci_v4.77.0
-command=export -compartment_id=$COMPID -services database -generate_state -output_path=../discover/
...
$ ls -lrt ../discover2
total 44
-rw-r--r-- 1 calero calero 10109 May 31 18:46 terraform.tfstate.tmp.backup
-rw-r--r-- 1 calero calero 22086 May 31 18:46 terraform.tfstate
-rw-r--r-- 1 calero calero  6085 May 31 18:46 database.tf
-rw-r--r-- 1 calero calero   38 May 31 18:46 provider.tf
-rw-r--r-- 1 calero calero  171 May 31 18:46 vars.tf
$
```

Terraform idiosyncrasy - state file / OCI discovery

- *terraform-provider-oci* gets installed by terraform:

```
$ ls -lrt .terraform/providers/registry.terraform.io/hashicorp/oci/4.77.0/linux_amd64/
total 129320
-rw-r--r-- 1 calero calero      96384 May 31 17:31 CHANGELOG.md
-rw-r--r-- 1 calero calero      16725 May 31 17:31 LICENSE.md
-rw-r--r-- 1 calero calero       3071 May 31 17:31 README.md
-rwxr-xr-x 1 calero calero 132177920 May 31 17:31 terraform-provider-oci_v4.77.0
```

- Discover databases and create the state file:

```
$ mkdir discover
$ .terraform-provider-oci
command=discover
...
$ ls -lrt
total 44
-rw-r--r--
-rw-r--r--
-rw-r--r--
-rw-r--r--
-rw-r--r--
$ cd discover
$ terraform plan
oci_database_autonomous_database.export_example_autonomous_data_warehouse: Refreshing state...
[id=ocid1.autonomousdatabase.oc1.iad.anuwcljst77gzoiaaewxote6h4i73oamzloxdueabnpwprspzngjoce52xq]
oci_database_autonomous_database.export_example_autonomous_database: Refreshing state...
[id=ocid1.autonomousdatabase.oc1.iad.anuwcljtt77gzoiamz3gg5cfsbdqgcoj7lrobis75qp7lwubm7d77bpblqgq]
No changes. Your infrastructure matches the configuration.
Terraform has compared your real infrastructure against your configuration and found no differences,
so no changes are needed.
$
```

Terraform idiosyncrasy - 2) concurrent work

```
$ terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...  
The refreshed state will be used to calculate this plan, but will not be  
persisted to local or remote state storage.
```

```
data.aws_security_group.sgpr: Refreshing state...  
aws_rds_cluster.master: Refreshing state... [id=nelson-auroramysql56]  
aws_rds_cluster_instance.cluster_instances[0]: Refreshing state... [id=nelson-auroramysql56-0]  
aws_rds_cluster_instance.cluster_instances[1]: Refreshing state... [id=nelson-auroramysql56-1]  
mysql_user.monitor: Refreshing state... [id=monitor@%]
```

```
Error: Could not connect to server: dial tcp 127.0.0.1:3306: connect: connection refused
```

Terraform idiosyncrasy - concurrent work

```
$ terraform plan
```

```
Ref $ terraform plan
The ...
per Error locking state: Error acquiring the state lock: ConditionalCheckFailedException: The conditional
dat request failed
aws Lock Info:
aws ID: f6f04b96-de6d-754a-7bbe-8d0504f75e79
aws Path: my-tf-state-286052569778/test/us-west-1/services/my-db/aurora-mysql/terraform.tfstate
mys Operation: OperationTypePlan
Who: ncalero@tst-rds01
Version: 0.12.31
Err Created: 2021-12-28 13:45:41.113713713 +0000 UTC
Info:
$
```

Terraform idiosyncrasy - concurrent work

```
$ terraform plan
```

```
Ref $ terraform plan  
The ...  
per Error locking state: Error acquiring the state lock: ConditionalCheckFailedException: The conditional  
dat request failed  
aws Lock Info:  
aws ID: f6f04b96-de6d-754a-7bbe-8d0504f75e79  
aws Path:  
mys Operati $ terraform force-unlock f6f04b96-de6d-754a-7bbe-8d0504f75e79  
Who: Do you really want to force-unlock?  
Version Terraform will remove the lock on the remote state.  
Created This will allow local Terraform commands to modify this state, even though it  
Info: may be still be in use. Only 'yes' will be accepted to confirm.
```

```
Err
```

```
$
```

```
Enter a value: yes
```

```
Terraform state has been successfully unlocked!
```

```
The state has been unlocked, and Terraform commands should now be able to  
obtain a new lock on the remote state.
```

```
$
```


Terraform idiosyncrasy - 3) reduce scope

```
$ terraform plan
```

```
data.oci_database_autonomous_db_versionstest_adb_versions: Reading...
oci_database_autonomous_database.autonomous_data_warehouse: Refreshing state...
[id=ocid1.autonomousdatabase.oc1.iad.anuwcljst77gzoiaaewxote6h4i73oamzoloxdueabnpwprspzngjoce52xq]
oci_database_autonomous_database.autonomous_database: Refreshing state...
[id=ocid1.autonomousdatabase.oc1.iad.anuwcljtt77gzoiamz3gg5cfsbdqgcoj7lrobis75qp7lwubm7d77bpblqgq]
data.oci_database_autonomous_databases.autonomous_databases: Reading...
```

```
...
```

```
Plan: 0 to e
```

```
$ terraform state list
```

```
data.http.ip
data.oci_database_autonomous_db_versions.test_adb_versions
oci_database_autonomous_database.autonomous_data_warehouse
oci_database_autonomous_database.autonomous_database
random_string.adb_admin_password
```

```
$ terraform plan -target oci_database_autonomous_database.autonomous_database
```

```
data.oci_database_autonomous_db_versions.test_adb_versions: Reading...
oci_database_autonomous_database.autonomous_database: Refreshing state...
[id=ocid1.autonomousdatabase.oc1.iad.anuwcljtt77gzoiamz3gg5cfsbdqgcoj7lrobis75qp7lwubm7d77bpblqgq]
```

```
No changes. Your infrastructure matches the configuration.
```

Terraform idiosyncrasy - 4) parameter validations

```
$ terraform apply
```

```
...  
aws_rds_cluster.master[0]: Creating...  
aws_rds_cluster.master[0]: Still creating... [10s elapsed]  
aws_rds_cluster.master[0]: Still creating... [20s elapsed]  
aws_rds_cluster.master[0]: Still creating... [30s elapsed]  
aws_rds_cluster.master[0]: Still creating... [40s elapsed]  
aws_rds_cluster.master[0]: Creation complete after 42s [id=my-aurora]  
aws_rds_cluster_instance.cluster_instances[1]: Creating...  
aws_rds_cluster_instance.cluster_instances[0]: Creating...
```

```
Error: error creating RDS Cluster (my-aurora) Instance: InvalidParameterCombination: RDS does not support creating a DB instance with the following combination: DBInstanceClass=db.t3.small, Engine=aurora-mysql, EngineVersion=5.7. For supported combinations of instance class and database engine version, see the documentation.
```

```
status code: 400, request id: 7c15bd63-b7cf-47a3-8a10-fae35c3e9c4
```

```
on main.tf line 7, in resource "aws_rds_cluster_instance" "cluster_instances":  
7: resource "aws_rds_cluster_instance" "cluster_instances" {
```

```
Error: error creating RDS Cluster (my-aurora) Instance: InvalidParameterCombination: RDS does not support creating a DB instance with the following combination: DBInstanceClass=db.t3.small, Engine=aurora-mysql, EngineVersion=5.7. For supported combinations of instance class and database engine version, see the documentation.
```

```
status code: 400, request id: a8cb9644-fbe2-40ee-929b-18094709cb33
```

```
on main.tf line 7, in resource "aws_rds_cluster_instance" "cluster_instances":  
7: resource "aws_rds_cluster_instance" "cluster_instances" {
```

Terraform idiosyncrasy - 4) parameter validations

- Terraform pass parameters to the provider - it runs the operation on the cloud service and it fails
- Some providers include *data sources* with *attributes* having a list of valid values for that (only if we use those when manipulating the resource)
 - Example: OCI Autonomous database
https://registry.terraform.io/providers/oracle/oci/latest/docs/data-sources/database_autonomous_db_versions
- Additionally, simple validations can be incorporated in the variable definition:

```
$ grep -A13 "autonomous_database_db_workload" variables.tf
variable "autonomous_database_db_workload" {
  default = "OLTP"
  validation {
    condition = anytrue([
      var.autonomous_database_db_workload == "OLTP",
      var.autonomous_database_db_workload == "DW",
      var.autonomous_database_db_workload == "APEX",
      var.autonomous_database_db_workload == "AJD"
    ])
    error_message = "Invalid autonomous_database_db_workload value - allowed OLTP, DW, APEX and AJD."
  }
}
```

Terraform idiosyncrasy - parameter validations

Demo 3

Terraform idiosyncrasy - 5) reusing code

- Terraform by default read the current directory only
- Can include other directories using the “*module*” directive:
<https://learn.hashicorp.com/collections/terraform/modules>

```
module "ec2_instances" {  
  source = "terraform-aws-modules/ec2-instance/aws"  
  version = "3.5.0"  
  ...  
}
```

- Third party tools to re-using code (among environments and/or deployments) - **terragrunt** example:
 - Standardize resource attributes (tags, HA, networking, security, parameters) between deployments - in same or different environments and regions
 - Support different database engines and versions
 - Validations (possible but limited to same attribute expression)
 - Modules, variables and environments - directory structure
 - Extra features: hooks, functions, etc. - check <https://terragrunt.gruntwork.io/>

Terraform idiosyncrasy - 6) “features”

- Terraform binary
 - Aurora MySQL upgrade didn't work on TF versions lower than 0.12.31 (current version: 1.1.7)
- Terraform provider
 - MySQL upgrade didn't work with AWS provider lower than 3.63.0 (several [bugs](#))
 - Resource discovery implemented in OCI (not in AWS as of today)
- Cloud provider limits/restrictions
 - Resource attributes are not validated (service restrictions) - cause runtime errors
 - allowed instance type for a RDS version (both ways)
 - IOPS parameter when using gp2 storage
 - CLI commands to the rescue (to confirm allowed values)

Cloud provider idiosyncrasy

- You can get errors today that does not repeat tomorrow, as services improves
- Don't assume all services work the same for similar operations - RTFM!
- Examples on AWS:
 - MySQL minor upgrade fails without any error - not reproducible days later when repeating the test
 - RDS “create Aurora replica” is not available in some versions after upgrades
 - Aurora space used should be checked using metrics
 - RDS parameters have **apply_mode** immediate or pending-reboot, but tricky to update running instances automatically (immediate mode)
 - Operations on RDS storage does not allow other operations until they finish - and those can take long
 - Automatic space increment - minutes
 - Change type between gp2 and IOPS - hours

Terraform - parameters cheat sheet

Format code in .tf files	terraform fmt
Validate configuration	terraform validate
List state file resources	terraform state list
Inspect current state file content	terraform show
Remove a resource from state file	terraform state rm <name_in_state>
Import resource into state file	terraform import <name_in_state> <name_aws>
Remove lock from state file	terraform force-unlock <lock_id>
Operate on a single resource	terraform <op> -target <name_in_state>
Console (to test formulas/expressions)	terraform console
Debug execution	TF_LOG="TRACE" TF_LOG_PATH=file.log ; terraform <op>

Terraform console

```
[c_ncalero@myhost ~]$ terraform console
> substr("19.2",0,2)
19

> split(".", "19.2")
[
  "19",
  "2",
]

> split(".", "19.2")[0]
19

> element(split(".", "19.2"), 0)
19

> join("", ["oracle", substr("19.2", 0, 2), "small"])
oracle19small

> join("", ["oracle", split(".", "19.2")[0]])
oracle19

> exit
```

OCI CLI - useful validations

- Available database versions for DB Systems using shape VM.Standard1.1

```
$ export COMPID=$(oci iam compartment list --query "data [?name=='mycomp'].{id:id}" | jq -r '.[].id')
$ oci db version list -c $COMPID --db-system-shape "VM.Standard1.1" --output table
```

is-latest-for-major-version	supports-pdb	version
True	False	11.2.0.4
False	False	11.2.0.4.210119
False	False	11.2.0.4.210420
True	True	12.1.0.2
False	True	12.1.0.2.211019
False	True	12.1.0.2.220118
False	True	12.1.0.2.220419
True	True	12.2.0.1
False	True	12.2.0.1.211019
False	True	12.2.0.1.220118
False	True	12.2.0.1.220419
True	True	18.0.0.0
False	True	18.16.0.0
True	True	19.0.0.0
False	True	19.13.0.0
False	True	19.14.0.0

False	True	19.15.0.0
True	True	21.0.0.0
False	True	21.4.0.0
False	True	21.5.0.0
False	True	21.6.0.0

OCI CLI - useful validations

- ADBs created in compartment 'mycmp'

```
$ export COMPID=$(oci iam compartment list --query "data [?name=='mycmp'].{id:id}" | jq -r '[][.id]')

$ oci db autonomous-database list -c $COMPID --query 'data[][db_name:"db-name", db_version:"db-version", db_workload:"db-workload", display_name:"display-name", state:"lifecycle-state"]' --output table
```

db_name	db_version	db_workload	display_name	state
adbdw	19c	DW	example_autonomous_data_warehouse	AVAILABLE
adboltp	19c	OLTP	example_autonomous_database	AVAILABLE

AWS CLI - useful validations

- Available Oracle EE versions for shape db.t3.small

```
$ aws rds describe-orderable-db-instance-options --engine oracle-ee --db-instance-class db.t3.small --query "OrderableDBInstanceOptions[].{EngineVersion:EngineVersion}" --output text --region eu-west-1
```

```
12.1.0.2.v1
12.1.0.2.v10
12.1.0.2.v11
12.1.0.2.v12
12.1.0.2.v13
12.1.0.2.v14
12.1.0.2.v15
12.1.0.2.v16
12.1.0.2.v17
12.1.0.2.v18
12.1.0.2.v19
12.1.0.2.v2
12.1.0.2.v20
12.1.0.2.v21
12.1.0.2.v22
12.1.0.2.v23
12.1.0.2.v24
12.1.0.2.v25
12.1.0.2.v26
12.1.0.2.v27
12.1.0.2.v3
12.1.0.2.v4
12.1.0.2.v5
12.1.0.2.v6
12.1.0.2.v7
12.1.0.2.v8
12.1.0.2.v9
19.0.0.0.ru-2019-07.rur-2019-07.r1
19.0.0.0.ru-2019-10.rur-2019-10.r1
19.0.0.0.ru-2020-01.rur-2020-01.r1
19.0.0.0.ru-2020-04.rur-2020-04.r1
19.0.0.0.ru-2020-07.rur-2020-07.r1
19.0.0.0.ru-2020-10.rur-2020-10.r1
19.0.0.0.ru-2021-01.rur-2021-01.r1
19.0.0.0.ru-2021-01.rur-2021-01.r2
19.0.0.0.ru-2021-04.rur-2021-04.r1
19.0.0.0.ru-2021-07.rur-2021-07.r1
19.0.0.0.ru-2021-10.rur-2021-10.r1
19.0.0.0.ru-2022-01.rur-2022-01.r1
```

AWS CLI - useful validations

- Supported upgrade paths for Oracle version 12.1.0.2

```
$ aws rds describe-db-engine-versions --engine oracle-ee --engine-version 12.1.0.2.v20 --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --output text
```

```
12.1.0.2.v21
12.1.0.2.v22
12.1.0.2.v23
12.1.0.2.v24
12.1.0.2.v25
12.1.0.2.v26
12.1.0.2.v27
19.0.0.0.ru-2020-04.rur-2020-04.r1
19.0.0.0.ru-2020-07.rur-2020-07.r1
19.0.0.0.ru-2020-10.rur-2020-10.r1
19.0.0.0.ru-2021-01.rur-2021-01.r1
19.0.0.0.ru-2021-01.rur-2021-01.r2
19.0.0.0.ru-2021-04.rur-2021-04.r1
19.0.0.0.ru-2021-07.rur-2021-07.r1
19.0.0.0.ru-2021-10.rur-2021-10.r1
19.0.0.0.ru-2022-01.rur-2022-01.r1
```

AWS CLI - useful validations

- Space in use by aurora Mysql instances running in region us-west-1

```
$ REGION=us-west-1; for i in $(aws cloudwatch list-metrics --region=$REGION --namespace AWS/RDS --metric-name VolumeBytesUsed --query "Metrics[].Dimensions[].[Name:Name, Value:Value]" --output table | grep DBClusterIdentifier | cut -d"|" -f3) ; do echo -n $i " " ; aws cloudwatch get-metric-statistics --region $REGION --namespace AWS/RDS --metric-name VolumeBytesUsed --dimensions Name=EngineName,Value=aurora-mysql --dimensions Name=EngineName,Value=aurora Name=DbClusterIdentifier,Value=$i --start-time $(date -u -d "1 days ago" +%F) --end-time $(date -u +%F) --statistics Maximum --period 3600 --query "Datapoints[].[Timestamp:Timestamp, Maximum:Maximum, Unit:Unit]" | jq '.[] | .Maximum' | sort -u | tail -1 ; done | sort
```

```
bookshop-db          7381361362844
sales-db             179468653458
track-db             20681679872
programs-db         35563371113
analytics-db        8650433415
master-db           17432850551
resourcing-db       6418328690
...
```

AWS CLI - useful validations

- Parameters set by user (we) in a RDS parameter group

```
$ aws rds describe-db-parameters --db-parameter-group-name mypg --region us-west-1 --query  
"sort_by(Parameters,&ParameterName)[?Source=='user'].{ParameterName:ParameterName, ApplyMethod:ApplyMethod,  
IsModifiable:IsModifiable, ParameterValue:ParameterValue}" --output table
```

ApplyMethod	IsModifiable	ParameterName	ParameterValue
immediate	True	event_scheduler	ON
immediate	True	group_concat_max_len	16384
pending-reboot	True	innodb_sort_buffer_size	8388608
immediate	True	log_bin_trust_function_creators	1
immediate	True	long_query_time	0.1
pending-reboot	True	max_allowed_packet	1073741824
immediate	True	max_connections	8000
immediate	True	slow_query_log	1

Thank You / Questions?

calero@pythian.com

@ncaleroy

<http://www.linkedin.com/in/ncalero>