How to: Restore DB from Snapshot

Overview

A database instance can be created to restore from snapshot. These snapshots can be in the form of automated or manual snapshots. Automated snapshots are taken by AWS during the backup window to backup the entire DB instance.

Restoring the database instance can be done through AWS Console, CLI, terraform code, or a combination of those.

After restoring a database from snapshot, ensure that further deployments of Airflow through the CICD pipeline will continue to reference the newly restored database.

Restore from Snapshot

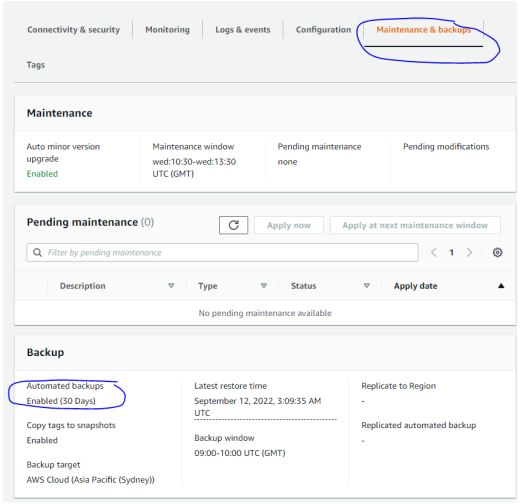
At a high-level, the steps to restore DB from snapshot involves:

- Creating a new database instance from a snapshot
- Updating the /ato/sdpaap/<landing_zone>/<environment>/airflow/config/sql_alchemy_conn SSM Parameter with the newly restored RDS endpoint value
- Restarting airflow-webserver and airflow-scheduler
- Ensuring the terraform state file is updated with the newly restored RDS details in order for subsequent Airflow CI pipeline deployments to use the correct RDS endpoint

Backup Window

Backup Retention Period

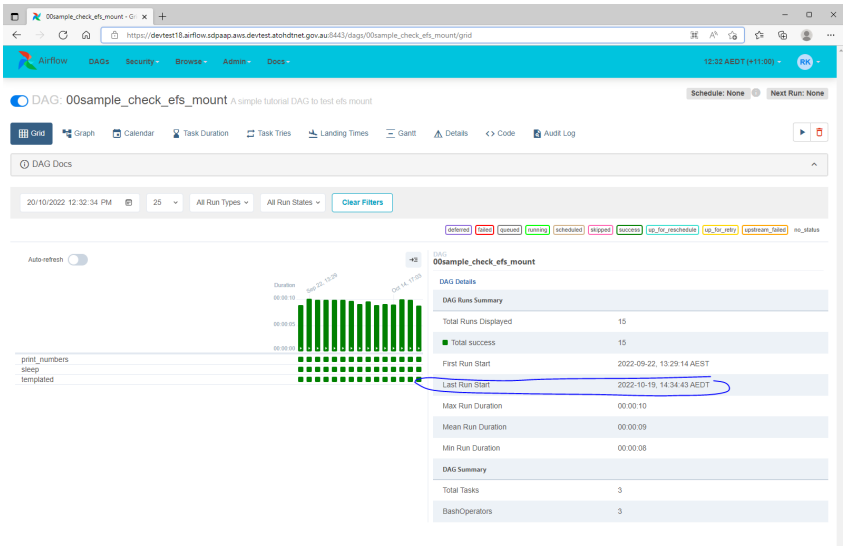
This is the number of days for which automatic backups are kept. Currently set to 30 days.



Pre-restore Steps

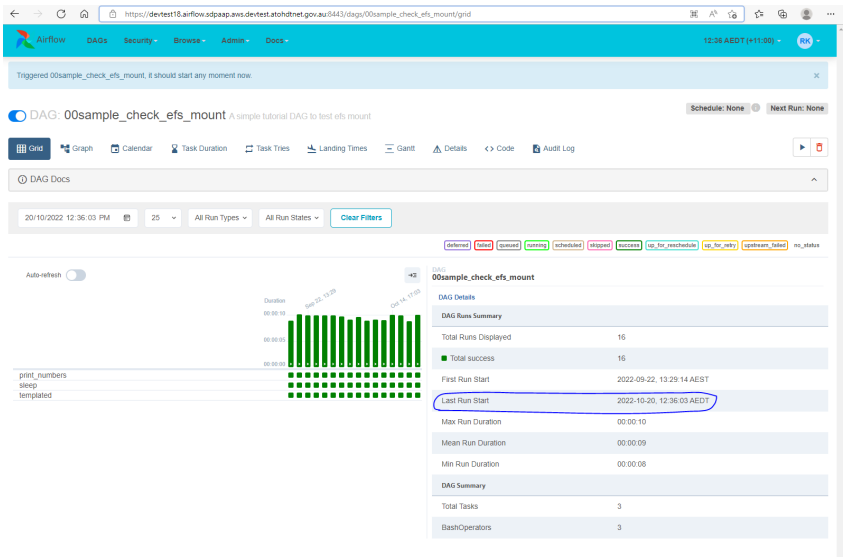
1. To demonstrate that the restore works, we'll run a sample DAG capture its current run state

    - The DAG was last run on 2022-10-19, 14:34:43 AEDT
    - Current datetime is 2022-10-20 12:30:00 AEDT

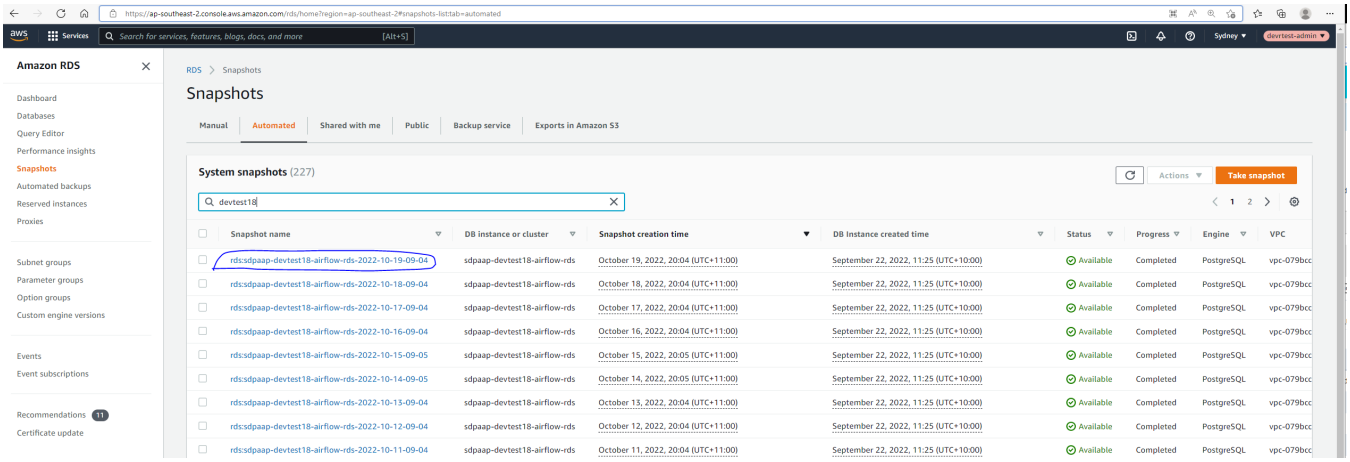


2. Run the sample DAG.

    - Now, the DAG's last run is 2022-10-20, 12:36:03 AEDT



3. As a test, the database will be restored to the latest automated snapshot taken in the last 24 hours in the next section.



4. The expected result of the restore from snapshot will be to not have the last DAG run in step 2, and that the last DAG run is as per Step 1 as there was no other DAG run post the snapshot creation time.

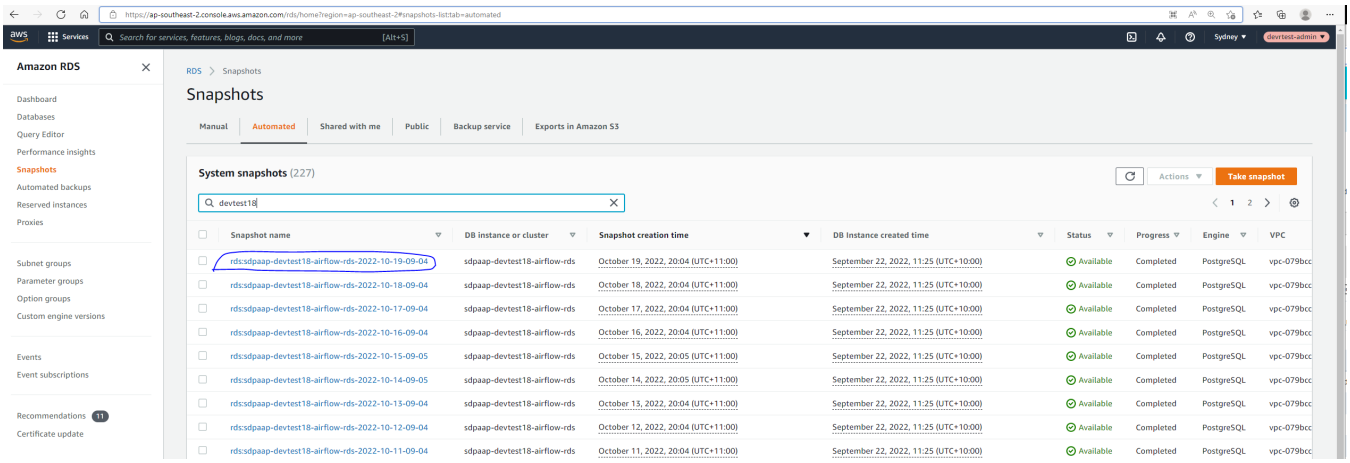
Restore from Snapshot Steps

1. Decide on the RDS DB instance prefix name to be used for restore

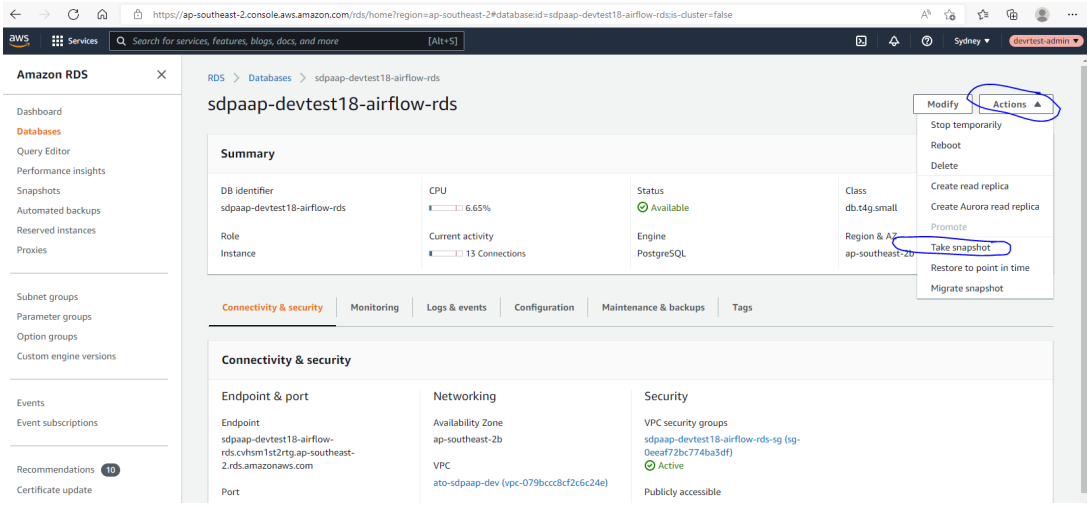
- The default prefix for RDS is "sdpaap".
- Since restore from snapshot requires a new DB instance to be created and we do not want to delete the existing DB instance, we need to specify a new prefix for the RDS name.
- Example new prefix = "sdpaap-restore"

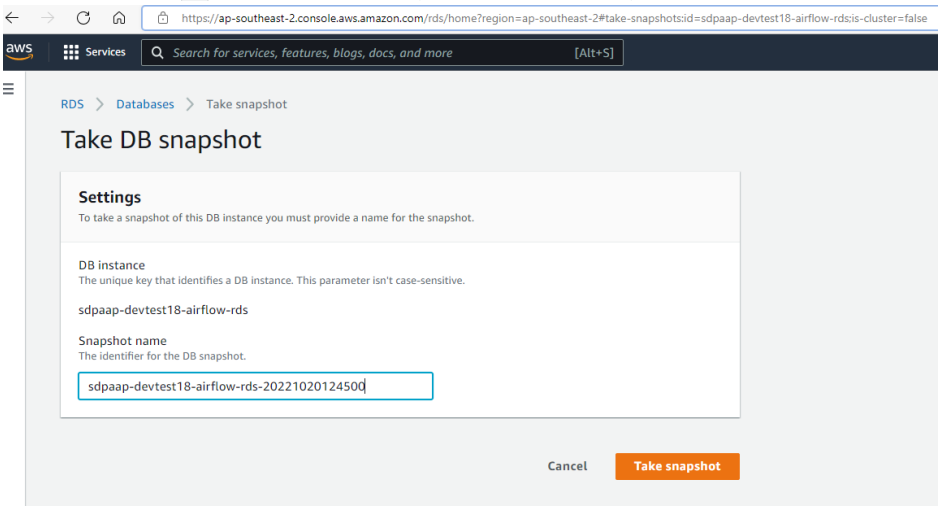
2. Identify the snapshot to restore the DB from

The latest automated snapshot will be used as an example for this restoration



3. Take a manual snapshot of current RDS instance via console





4. Get the SSM Parameter Value of sql_alchemy_conn

- Retrieve the current RDS endpoint used by Airflow in AWS console or on the jumphost by running the CLI command below. This will be used at a later step to verify that the database connection endpoint has been changed in Airflow.

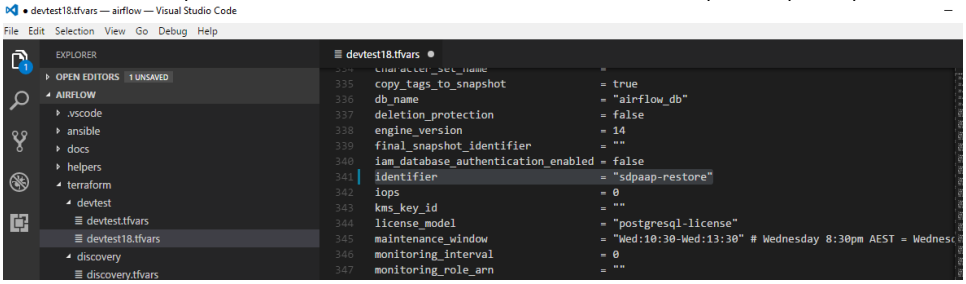
```
aws ssm get-parameter --name /ato/sdpaap/<landing_zone>/<environment>/airflow/config/sql_alchemy_conn --with-decryption | jq '.Parameter.Value'

E.g.: aws ssm get-parameter --name /ato/sdpaap/devtest/devtest18/airflow/config/sql_alchemy_conn --with-decryption | jq '.Parameter.Value'
```

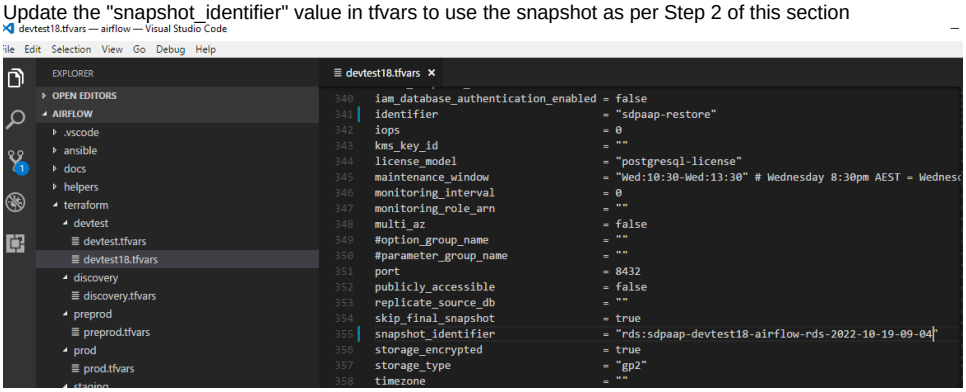
- Note down the value of sql_alchemy_conn SSM Param

5. Update the value of RDS identifier prefix and snapshot identifier in airflow code's tfvars

- Create a new feature or release branch in airflow repo
- In VSCode or IDE, update the "identifier" value in tfvars to use the RDS restore prefix as per Step 1 of this section



- Update the "snapshot_identifier" value in tfvars to use the snapshot as per Step 2 of this section



- Save your changes. Commit and push your changes to remote

6. Clone the airflow repo and dev-utils repo onto the jumphost

- Back to the jumphost as your "a" account
- Ensure your own ecdsa ssh keys are setup on the jumphost and the public ecdsa key is uploaded to your Gitlab profile
- Clone the repos

```
# in your home directory on the jumpbox
mkdir ~/git
cd ~/git
git clone git@gitlab.sdpaap.aws.prod.atohnet.gov.au:sdpaap/platform/airflow.git
git clone git@gitlab.sdpaap.aws.prod.atohnet.gov.au:sdpaap/ops-utils/dev-utils.git
```

7. Checkout your airflow feature or release branch on the jumphost

- Still on the jumphost, checkout your feature branch and ensure you have the tfvars for your environment

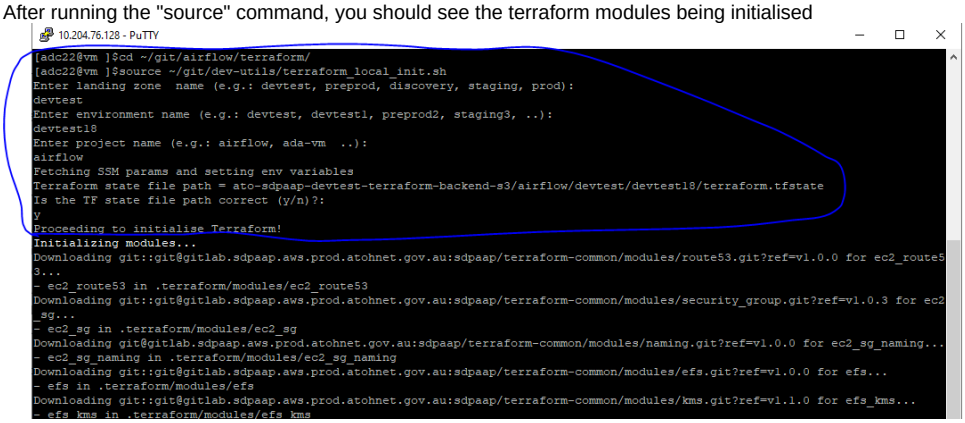
```
cd ~/git/airflow
git checkout <feature or release branch>
ls -l terraform/<landing_zone>/<environment>.tfvars
```

8. Setup your local terraform environment on the jumphost

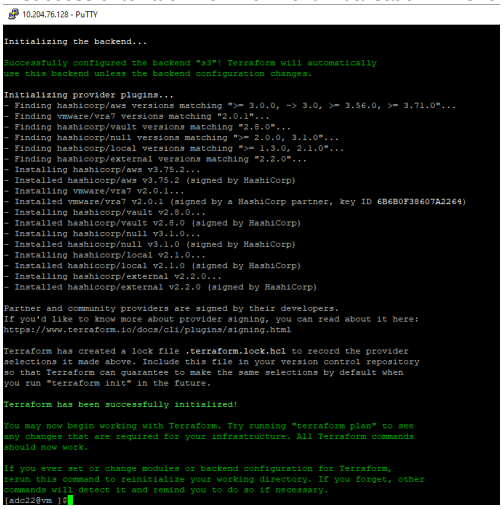
- The commands in this step are based on the instructions to setup a local terraform environment as described in the dev-utils README.
- On the jumphost

```
cd ~/git/airflow/terraform/
source ~/git/dev-utils/terraform_local_init.sh
```

- After running the "source" command, you should see the terraform modules being initialised



- A successful terraform environment initialisation will show these logs



9. Remove the db instance, db option group, db parameter group, db subnet group from tfstate file

- This is so terraform won't destroy them when you run the CICD pipeline as they are referencing the "old" database resources
- Run these commands on the jumphost where the local terrform environment was initialised as in the previous step

```
terraform state rm module.rds.aws_db_instance.this
terraform state rm module.rds.aws_db_option_group.group[0]
terraform state rm module.rds.aws_db_parameter_group.parameter_group[0]
terraform state rm module.rds.aws_db_subnet_group.generic_group[0]
```

10. Check the latest Terraform plan, which should show an RDS instance, subnet group, parameter group, and option group will be created

- Still on the jumphost with local terraform environment

```
terraform plan -var-file <landing_zone>/<environment>.tfvars

E.g.: terraform plan -var-file devtest/devtest18.tfvars
```

- A new option group, parameter group, subnet group will be created using the new prefix naming convention

```
# module.rds.aws_db_option_group.group[0] will be created
+ resource "aws_db_option_group" "group" {
    + arn                      = (known after apply)
    + engine_name              = "postgres"
    + id                       = (known after apply)
    + major_engine_version     = "14"
    + name                     = "sdpaap-restore-devtest18-airflow-rds-option-group-postgres14"
    + name_prefix              = (known after apply)
    + option_group_description = "Airflow RDS Option Group"
    + tags_all                 = (known after apply)
  }

# module.rds.aws_db_parameter_group.parameter_group[0] will be created
+ resource "aws_db_parameter_group" "parameter_group" {
    + arn         = (known after apply)
    + description = "Managed by Terraform"
    + family      = "postgres14"
    + id          = (known after apply)
    + name        = "sdpaap-restore-devtest18-airflow-rds-parameter-group-postgres14"
    + name_prefix = (known after apply)
    + tags_all    = (known after apply)
  }

# module.rds.aws_db_subnet_group.generic_group[0] will be created
+ resource "aws_db_subnet_group" "generic_group" {
    + arn         = (known after apply)
    + description = "Managed by Terraform"
    + id          = (known after apply)
    + name        = "sdpaap-restore-devtest18-airflow-rds-subnet-group"
    + name_prefix = (known after apply)
    + subnet_ids  = [
        + "subnet-032d3665eea6f2bed",
        + "subnet-09bfe26ae2ce4c40a",
        + "subnet-0b2cd0025b6172122",
      ]
    + tags        = {
        + "SDPAAP_AppID"       = "airflow"
        + "SDPAAP_AppName"     = "airflow"
        + "SDPAAP_CostCode"    = "4318"
        + "SDPAAP_Environment" = "devtest18"
        + "SDPAAP_LandingZone" = "devtest"
        + "SDPAAP_Name"        = "sdpaap-restore-devtest18-airflow-rds"
        + "SDPAAP_Owner"       = "aaptocloud@ato.gov.au"
      }
    + tags_all    = {
        + "SDPAAP_AppID"       = "airflow"
        + "SDPAAP_AppName"     = "airflow"
        + "SDPAAP_CostCode"    = "4318"
        + "SDPAAP_Environment" = "devtest18"
        + "SDPAAP_LandingZone" = "devtest"
        + "SDPAAP_Name"        = "sdpaap-restore-devtest18-airflow-rds"
        + "SDPAAP_Owner"       = "aaptocloud@ato.gov.au"
      }
  }
```

- A new RDS instance will be created using the new prefix naming convention.
  This will use the option group, parameter group, subnet group as per the naming convention.
  And will be created based from the defined snapshot_identifer.

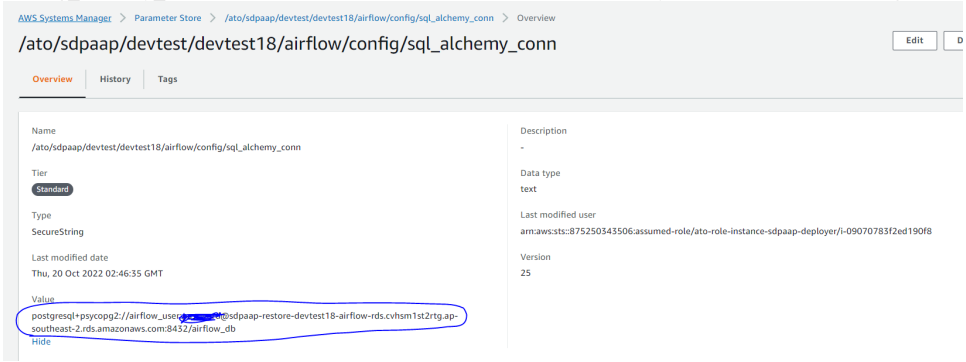
```
# module.rds.aws_db_instance.this will be created
+ resource "aws_db_instance" "this" {
    + address                               = (known after apply)
    + allocated_storage                     = 10
    + allow_major_version_upgrade           = false
    + apply_immediately                     = true
    + arn                                   = (known after apply)
    + auto_minor_version_upgrade            = true
    + availability_zone                     = (known after apply)
    + backup_retention_period               = 30
    + backup_window                         = "09:00-10:00"
    + ca_cert_identifier                    = (known after apply)
    + character_set_name                    = (known after apply)
    + copy_tags_to_snapshot                 = true
    + customer_owned_ip_enabled             = false
    + db_subnet_group_name                  = "sdpaap-restore-devtest18-airflow-rds-subnet-group"
    + delete_automated_backups              = true
    + deletion_protection                   = false
    + enabled_cloudwatch_logs_exports       = [
        + "postgresql",
      ]
    + endpoint                              = (known after apply)
    + engine                                = "postgres"
    + engine_version                        = "14"
    + engine_version_actual                 = (known after apply)
    + final_snapshot_identifier             = "sdpaap-restore-devtest18-airflow-rds-final-snapshot"
    + hosted_zone_id                        = (known after apply)
    + iam_database_authentication_enabled   = false
    + id                                    = (known after apply)
    + identifier                            = "sdpaap-restore-devtest18-airflow-rds"
    + identifier_prefix                     = (known after apply)
    + instance_class                        = "db.t4g.small"
    + iops                                  = 0
    + kms_key_id                            = "arn:aws:kms:ap-southeast-2:875250343506:key/a8450fab-c251-4704-b91b-af37eb46b3a5"
    + latest_restorable_time                = (known after apply)
    + license_model                         = "postgresql-license"
    + maintenance_window                    = "wed:10:30-wed:13:30"
    + max_allocated_storage                 = 50
    + monitoring_interval                   = 0
    + monitoring_role_arn                   = (known after apply)
    + multi_az                              = false
    + name                                  = "airflow_db"
    + nchar_character_set_name              = (known after apply)
    + option_group_name                     = "sdpaap-restore-devtest18-airflow-rds-option-group-postgres14"
    + parameter_group_name                  = "sdpaap-restore-devtest18-airflow-rds-parameter-group-postgres14"
    + password                              = (sensitive value)
    + performance_insights_enabled          = false
    + performance_insights_kms_key_id       = (known after apply)
    + performance_insights_retention_period = (known after apply)
    + port                                  = 8432
    + publicly_accessible                   = false
    + replicas                              = (known after apply)
    + replicate_source_db                   = ""
    + resource_id                           = (known after apply)
    + skip_final_snapshot                   = true
    + snapshot_identifier                   = "rds:sdpaap-devtest18-airflow-rds-2022-10-19-09-04"
    + status                                = (known after apply)
    + storage_encrypted                     = true
    + storage_type                          = "gp2"
    + tags                                  = {
        + "SDPAAP_AppID"       = "airflow"
        + "SDPAAP_AppName"     = "airflow"
        + "SDPAAP_CostCode"    = "4318"
        + "SDPAAP_Environment" = "devtest18"
        + "SDPAAP_LandingZone" = "devtest"
        + "SDPAAP_Name"        = "sdpaap-restore-devtest18-airflow-rds"
        + "SDPAAP_Owner"       = "aaptocloud@ato.gov.au"
      }
    + tags_all                              = {
        + "SDPAAP_AppID"       = "airflow"
        + "SDPAAP_AppName"     = "airflow"
        + "SDPAAP_CostCode"    = "4318"
        + "SDPAAP_Environment" = "devtest18"
        + "SDPAAP_LandingZone" = "devtest"
        + "SDPAAP_Name"        = "sdpaap-restore-devtest18-airflow-rds"
        + "SDPAAP_Owner"       = "aaptocloud@ato.gov.au"
      }
```

11. Run the airflow CICD pipeline with your feature or release branch

- New subnet group, option group and parameter group should be created
- The RDS restore instance should be using the newly created subnet group, option group and parameter group

12. Verify that the SSM Parameter Value of sql_alchemy_conn is updated

- Run the get-parameter cli command or check in AWS console to verify.
  The sql_alchemy_conn SSM Parameter should have the value of the newly created RDS restore endpoint



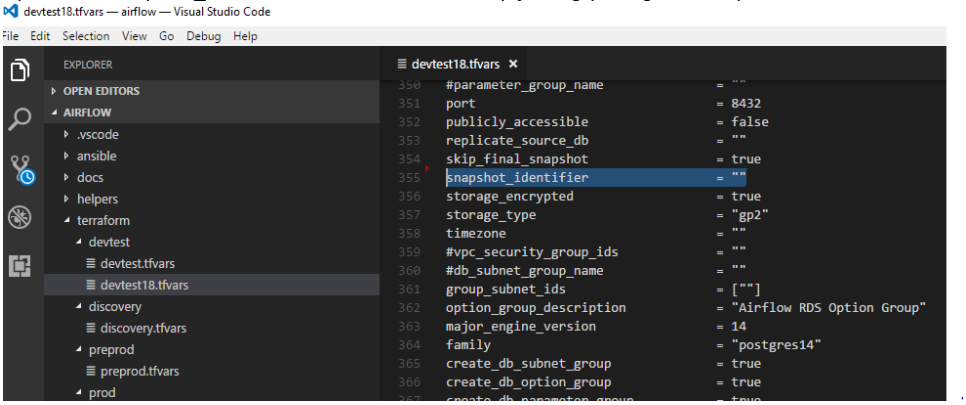
13. Login to Airflow UI and verify that the data is retored to the desired state

- For testing purposes, I have verified that the sample DAG in Airflow UI has the "Last Run Start" value as shown in Step 1 of Pre-restore Steps
- We have now successfully restored the database from a snapshot.

14. Perform post verification checks and ensure a sample DAG runs successfully

15. Remove the snapshot_identifer in tfvars

- Although keeping the snapshot_identifier as is will not impact further deployments because RDS already exists, we will change the value back to an empty string in order for an empty DB to be created and not from a snapshot in the event that a new RDS will be created.
- Update the "snapshot_identifier" value in tfvars to an empty string (its original value)



- Save your changes. Commit and push your changes to remote

16. Re-run the airflow CICD pipeline with your feature or release branch and perform post verification check

- The pipeline should run successfully with no resource and config changes
- Running a sample DAG should execute successfully