Prashant Lakhera
Mar 3, 2019 · 3 min read · Listen

100 Days of DevOps — Day 21- MySQL RDS Database Creation using Terraform

Welcome to Day 21 of 100 Days of DevOps, Let continue our journey with terraform and today we are going to create MySql database using terraform

What Is Amazon Relational Database Service (Amazon RDS)?

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

Step1: *Create a DB subnet group*

- *In order to create a new MySql database we first need to create a subnet group and assign at least two subnets to it.*

```
resource "aws_db_subnet_group" "rds-private-subnet" {
  name = "rds-private-subnet-group"
  subnet_ids = [*(var.rds_subnet1),*(var.rds_subnet2)*]
}
```

rds_db_subnet.tf hosted with ❤ by GitHub — view raw

Step2: *Create a Security Group to allow mysql port 3306*

```
resource "aws_security_group" "rds-sg" {
  name   = "my-rds-sg"
  vpc_id = *(var.vpc_id)*

}

# Ingress Security Port 3306
resource "aws_security_group_rule" "mysql_inbound_access" {
  from_port       = 3306
  protocol        = *tcp*
  security_group_id = *(aws_security_group.rds-sg.id)*
  to_port         = 3306
  type            = *ingress*
  cidr_blocks     = *["0.0.0.0/0"]*
}
```

mysql_rds_port.tf hosted with ❤ by GitHub — view raw

Step3: *Next step is to create MySQL resource*

```
resource "aws_db_instance" "my_test_mysql" {
  allocated_storage    = 20
  storage_type         = *gp2*
  engine               = *mysql*
  engine_version       = *5.7*
  instance_class       = *(var.db_instance)*
  name                 = *mydrktestmysql*
  username             = *admin*
  password             = *admin123*
  parameter_group_name = *default.mysql5.7*
  db_subnet_group_name = *(aws_db_subnet_group.rds-private-subnet.name)*
  vpc_security_group_ids = *[*(aws_security_group.rds-sg.id)*]*
  allow_major_version_upgrade = true
  auto_minor_version_upgrade  = true
  backup_retention_period     = 35
  backup_window               = *22:00-23:00*
  maintenance_window          = *Tat:00:00-Sat:03:00*
  multi_az                    = true
  skip_final_snapshot         = true
}
```

rds_mysql.tf hosted with ❤ by GitHub — view raw

```
* allocated_storage: This is the amount in GB
* storage_type: Type of storage we want to allocate(options available
*standard (magnetic), "gp2" (general purpose SSD), or "io1"
(provisioned IOPS SSD).
* engine: The database engine(for supported values check
https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_CreateDB
Instance.html) eg: Oracle, Amazon Aurora,Postgres
* engine_version: engine version to use
* instance_class: instance type for rds instance
* name: The name of the database to create when the DB instance is
created
* username: Username for the master DB user.
* password: Password for the master DB user
* db_subnet_group_name: DB instance will be created in the VPC
associated with the DB subnet group. If unspecified, will be created
in the default
* vpc_security_group_ids: List of VPC security groups to associate.
* allow_major_version_upgrade: Indicates that major version upgrades
are allowed. Changing this parameter does not result in an outage and
the change is asynchronously applied as soon as possible.
* auto_minor_version_upgrade:Indicates that minor engine upgrades will
be applied automatically to the DB instance during the maintenance
window. Default is true
* backup_retention_period: The days to retain backups for. Must be
between 0 and 35. When creating a Read Replica the value must be
greater than 0
* backup_window: The daily time range (in UTC) during which automated
backups are created if they are enabled. Must not overlap with
maintenance_window
* maintenance_window: The window to perform maintenance in. Syntax:
"ddd:hh24:mi-ddd:hh24:mi".
* multi_az: Specifies if the RDS instance is multi-AZ
* skip_final_snapshot: Determines whether a final DB snapshot is
created before the DB instance is deleted. If true is specified, no
DBSnapshot is created. If false is specified, a DB snapshot is created
before the DB instance is deleted, using the value from
final_snapshot_identifier. Default is false
```

Github Link for Complete Code

100days0ofdevops/100days0ofdevops
Contribute to 100days0ofdevops/100days0ofdevops development by creating an account on GitHub.
github.com

- *One of the clear issues I see in the above code is that we are passing the password in the plain text inside the terraform code*
- *Now to encrypt that password we can use KMS*

Step1: *First Create KMS Keys*

```
resource "aws_kms_key" "rds-key" {
  description = "Key to encrypt rds password"
  tags {
    Name = "my-rds-kms-key"
  }
}

resource "aws_kms_alias" "rds-kms-alias" {
  target_key_id = *(aws_kms_key.rds-key.id)*
  name = *"alias/rds-kms-key"*
}
```

kms_keys.tf hosted with ❤ by GitHub — view raw

Step2: *Now use that key to encrypt a secret on a command line*

```
aws kms encrypt --key-id <kms key id> --plaintext admin123 --output
text --query CiphertextBlob
```

Step3: *Now the encoded string we got pass it as a payload in your terraform code*

```
data "aws_kms_secret" "rds-secret" {
  *secret*" {
    name = "master_password"
    payload = "payload value here"
  }
}

resource "aws_db_instance" "my_test_mysql" {
  allocated_storage    = 20
  storage_type         = *gp2*
  engine               = *mysql*
  engine_version       = *5.7*
  instance_class       = *(var.db_instance)*
  name                 = *mydrktestmysql*
  username             = *admin*
  password             = *(data.aws_kms_secret.rds-secret.master_password)*
  parameter_group_name = *default.mysql5.7*
  db_subnet_group_name = *(aws_db_subnet_group.rds-private-subnet.name)*
  vpc_security_group_ids = *[*(aws_security_group.rds-sg.id)*]*
  allow_major_version_upgrade = true
  auto_minor_version_upgrade  = true
  backup_retention_period     = 35
  backup_window               = *22:00-23:00*
  maintenance_window          = *Tat:00:00-Sat:03:00*
  multi_az                    = true
  skip_final_snapshot         = true
}
```

kms_rds_password.tf hosted with ❤ by GitHub — view raw

- *Now why I didn't put this solution in first place and the reason for that, because of the below-mentioned error and I want to present a working solution*

```
$ 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.

~ aws_kms_key.rds-key: Refreshing state... (ID: 9731d86-5899-k380-ae72-c27c517ec918)
data.aws_kms_secret.rds: Refreshing state...
data.aws_availability_zones.available: Refreshing state...
aws_kms_alias.rds-kms-alias: Refreshing state... (ID: alias/rds-kms-key)

Error: Error refreshing state: 1 error(s) occurred:

* data.aws_kms_secret.rds: 1 error(s) occurred:

* data.aws_kms_secret.rds: data.aws_kms_secret.rds: This data source has been replaced
```

terraform_plan.txt hosted with ❤ by GitHub — view raw

- *There is already a bug opened for this issue*

data-source/aws_kms_secret: Soft remove data source type with removal message by bflad · Pull...
References: #5144 #5195 The aws_kms_secret data source uses dynamic attribute functionality which is not supported in...
github.com

Looking forward from you guys to join this journey and spend a minimum an hour every day for the next 100 days on DevOps work and post your progress using any of the below medium.

- *Twitter: @100daysofdevops OR @lakhera2015*
- *Facebook: https://www.facebook.com/groups/795382630808645/*
- *Medium: https://medium.com/@devopslearning*
- *Slack: https://devops-myworld.slack.com/messages/CF41EFG49/*
- *GitHub Link: https://github.com/100daysofdevops*

Reference

100 Days of DevOps — Day 0
D-day is just one day away and finally, this is a continuation of the post(I posted a month earlier)
medium.com