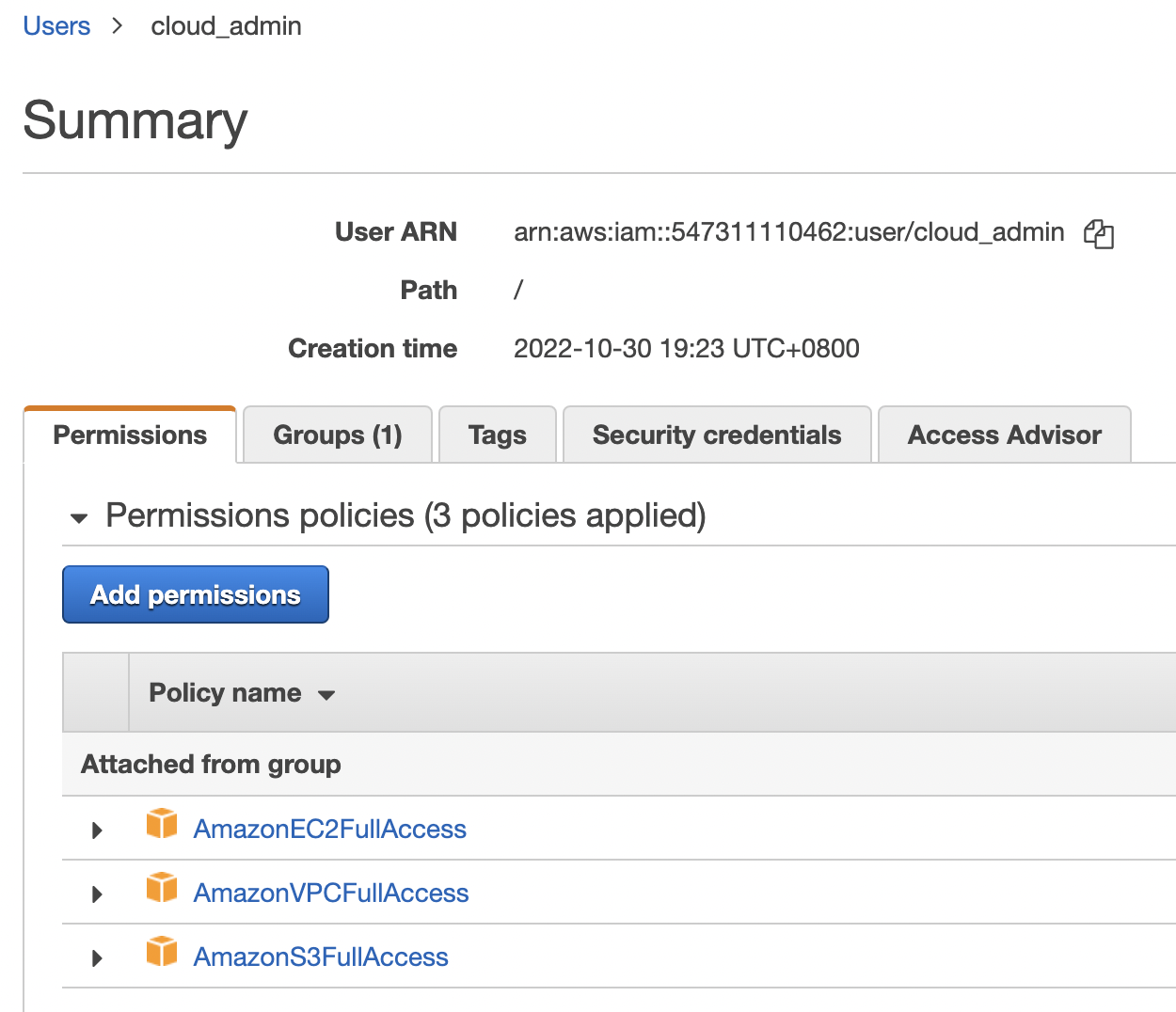
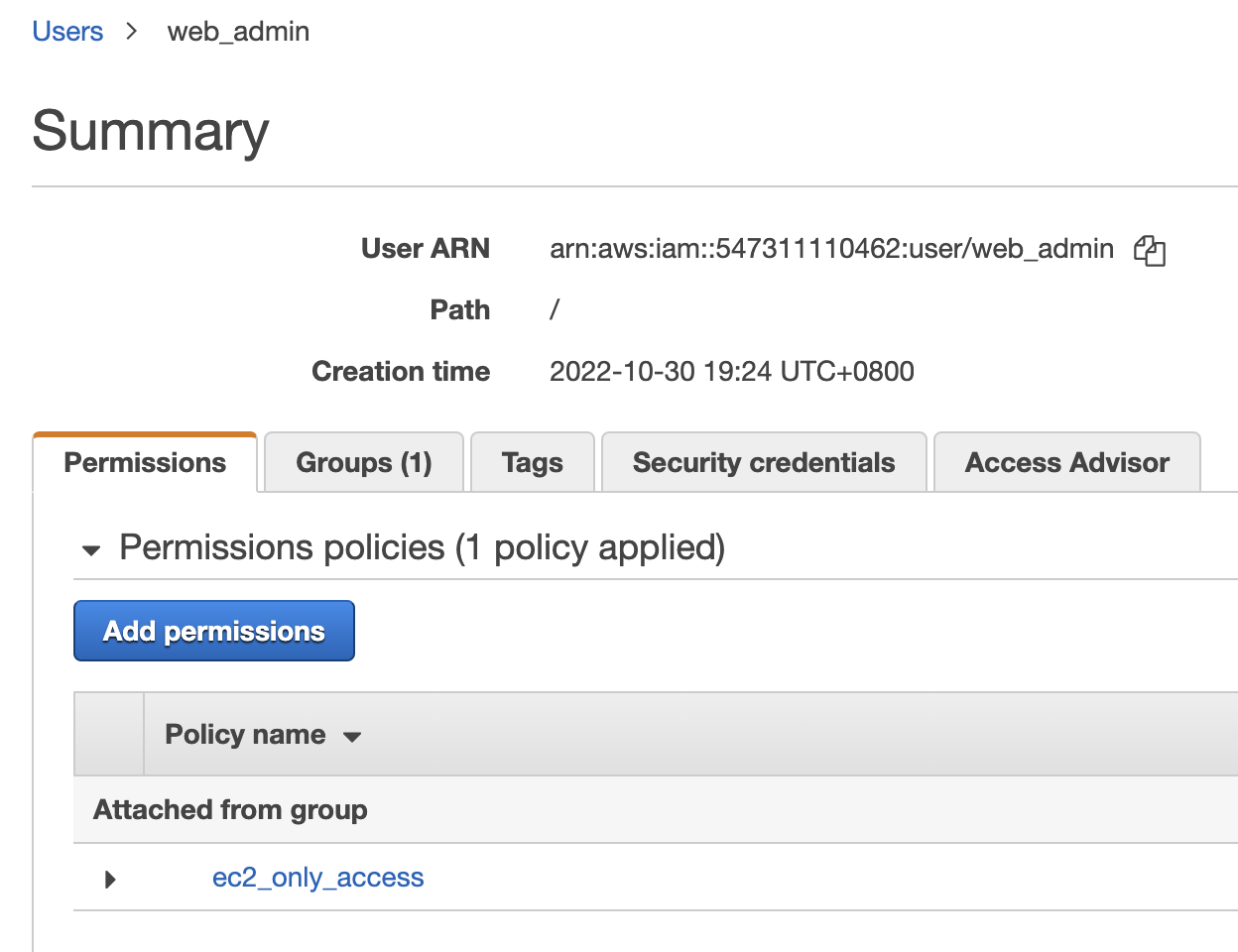
**Step 1 – hands-on**

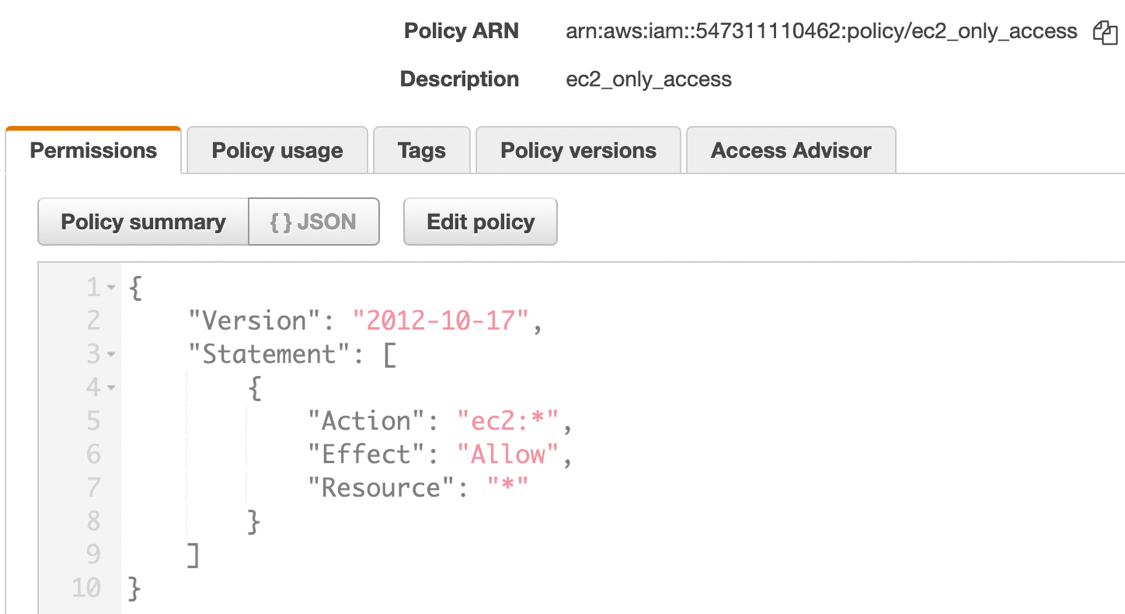
Setup the application and give us the following screenshots:

1. Setup **IAM** users for some roles: cloud admin, web application admin and dev/ops role.
   1. The cloud admin just can access and operate the infrastructure.

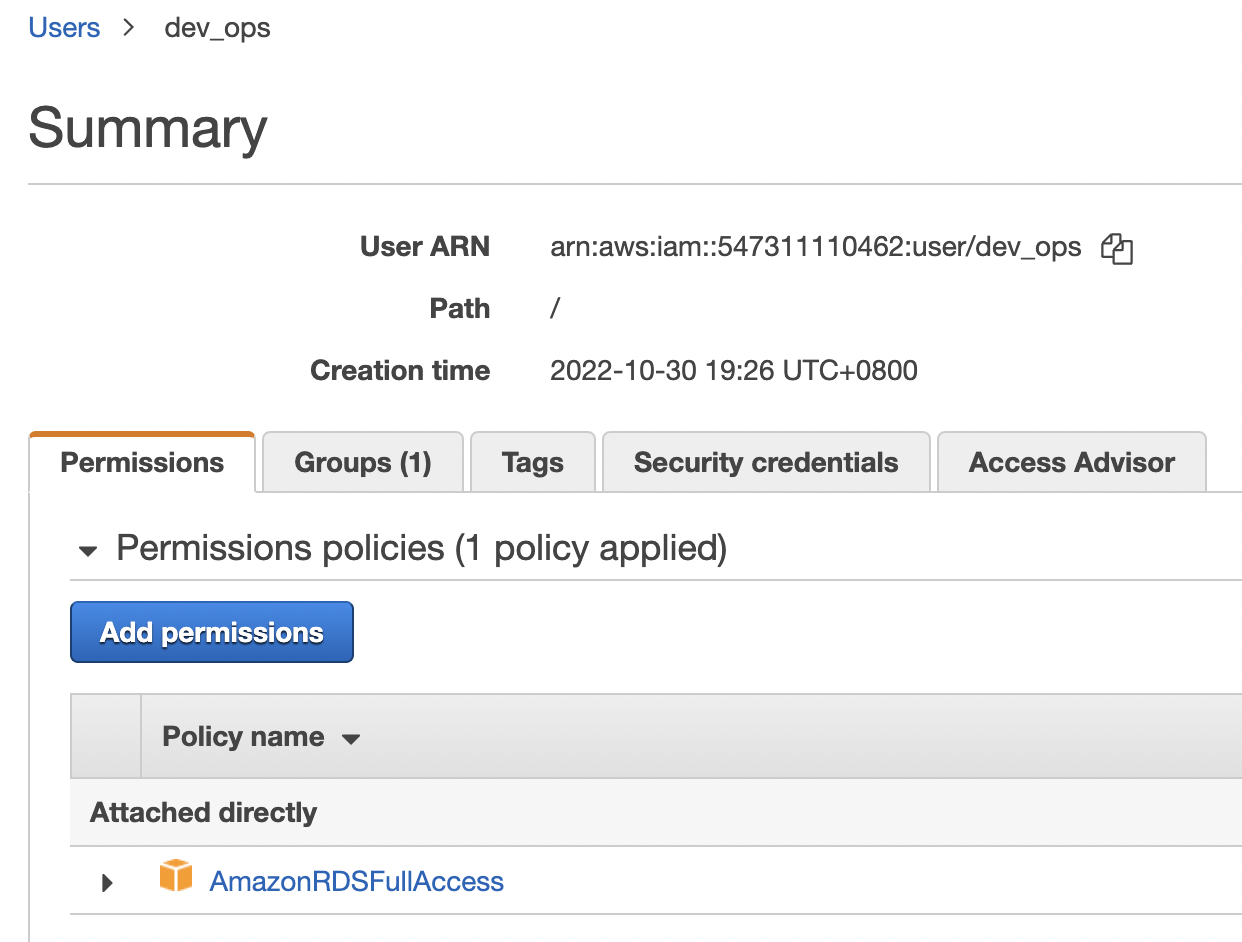


* 1. Web application admin can only access and update the web html files.



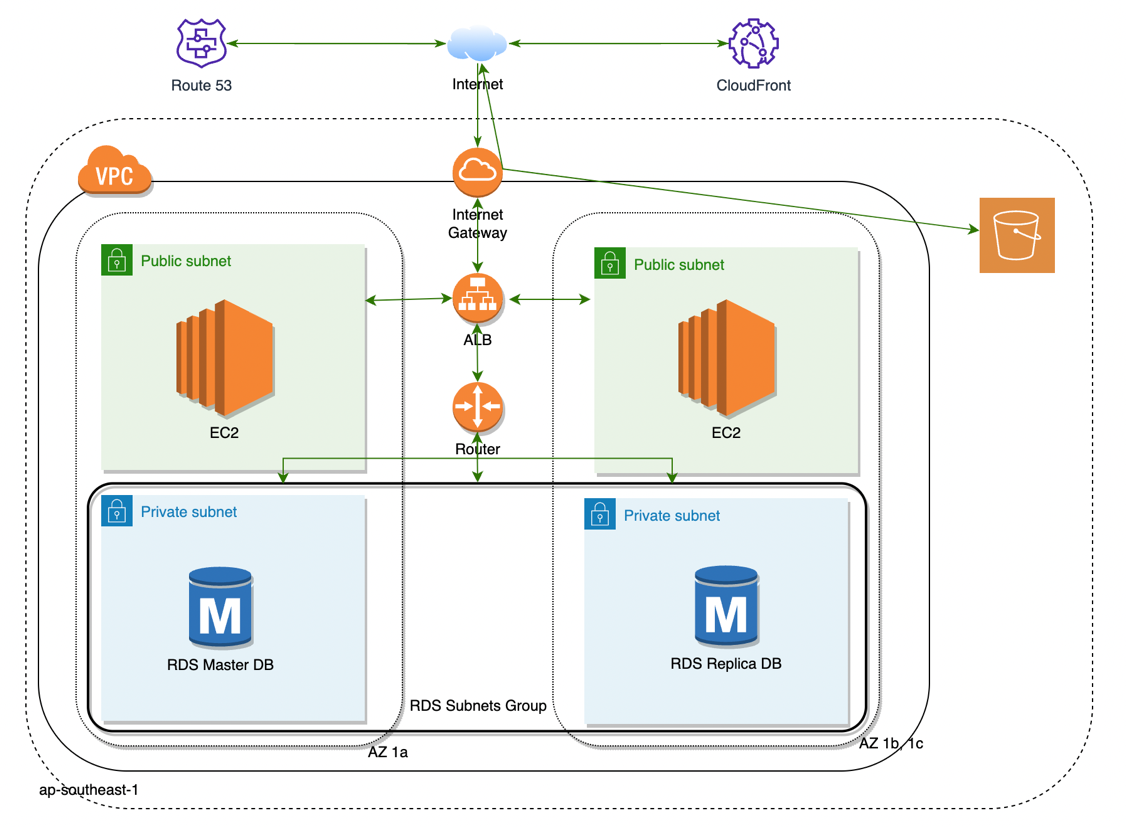


* 1. Dev/ops role can access and control the RDS database.



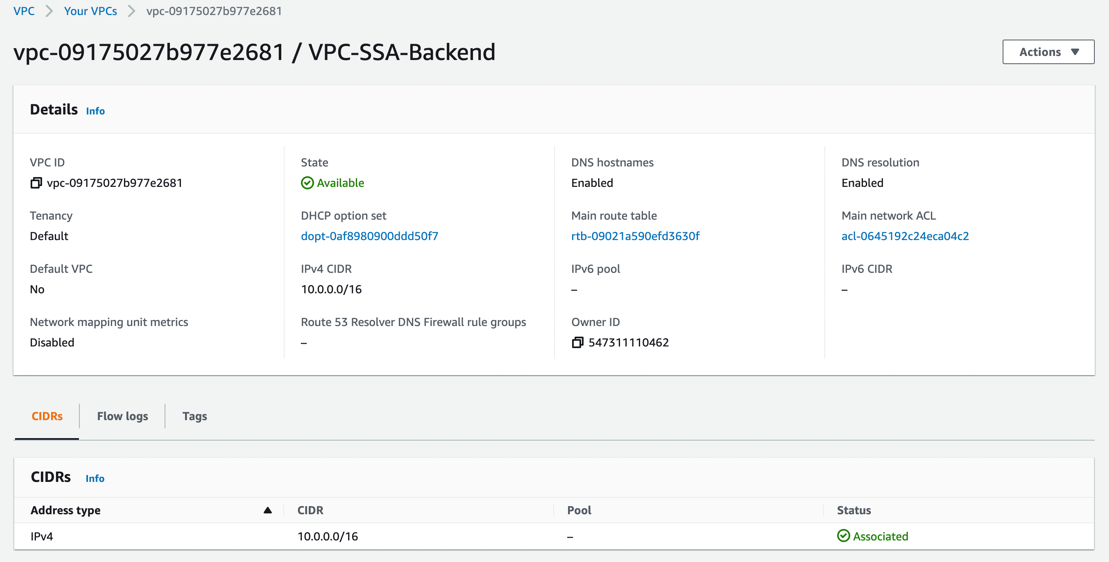
1. Save and send the screenshots:

The VPC and its security related screenshots;

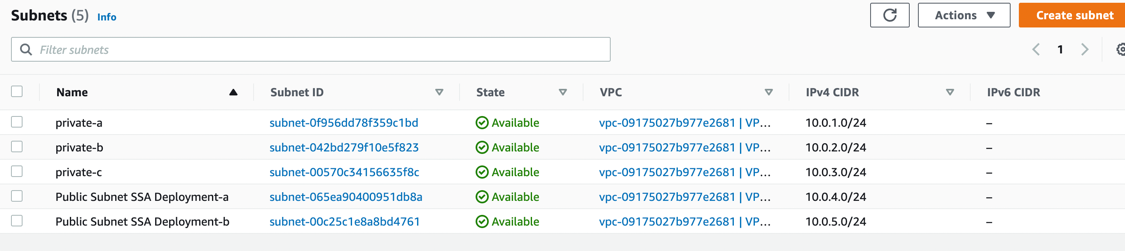


**VPC** and **Security group** info;

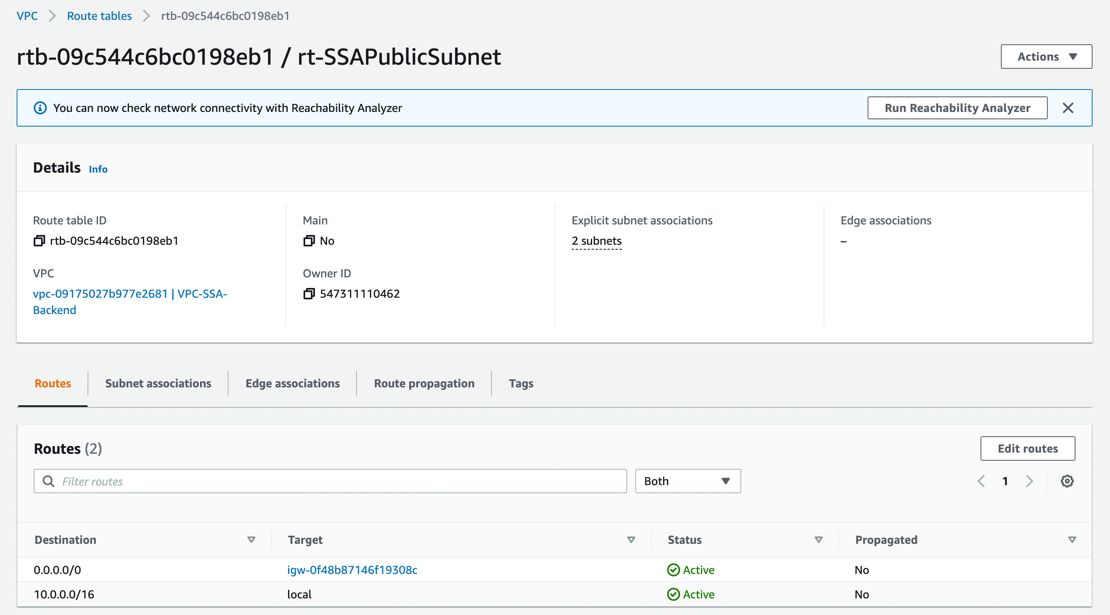
VPC:



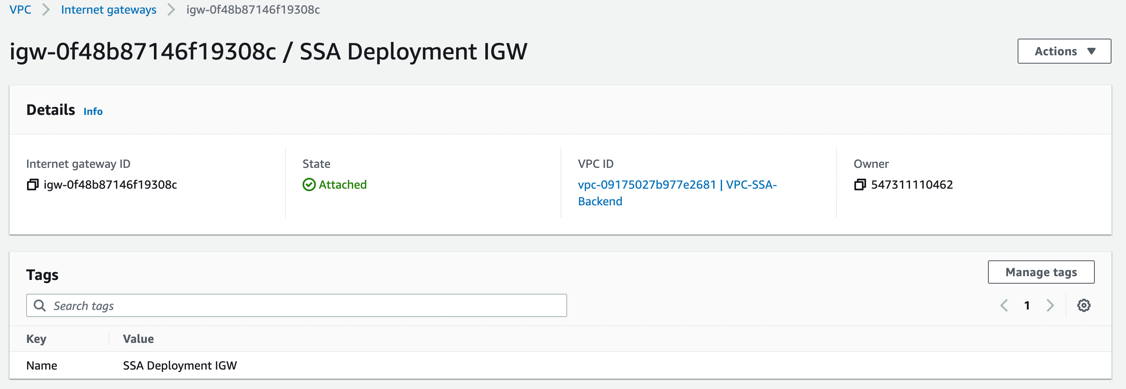
Subnets:



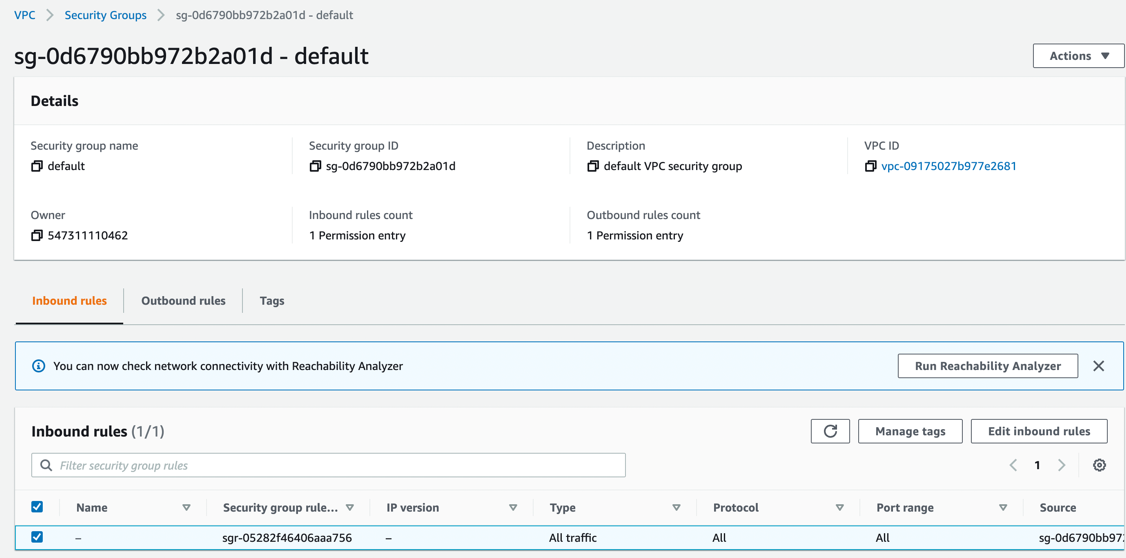
Route table:



Internet gateway:

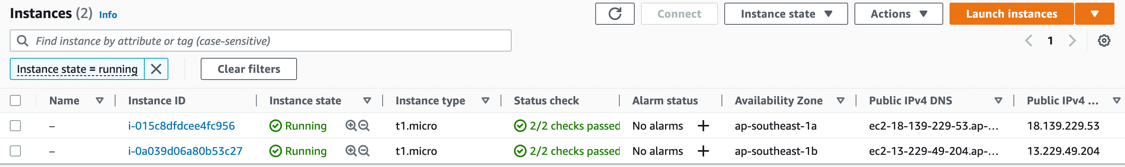


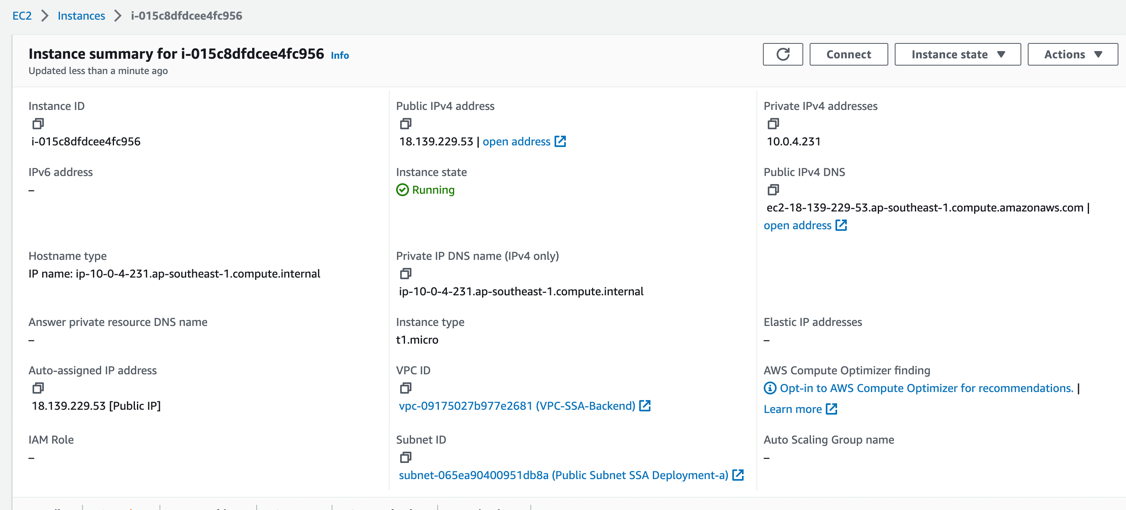
VPC Security group:

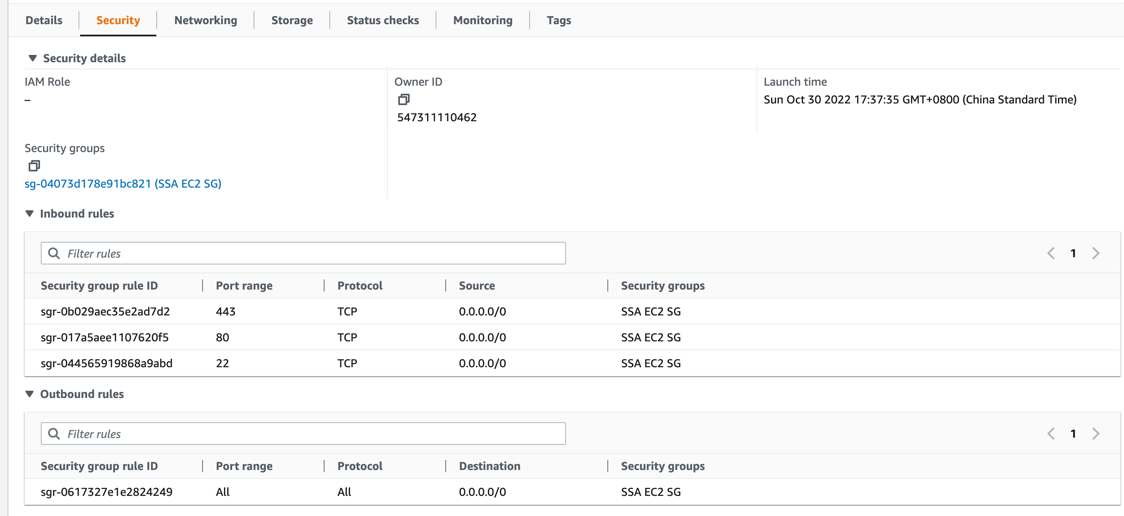


All ALB and **EC2** detail info;

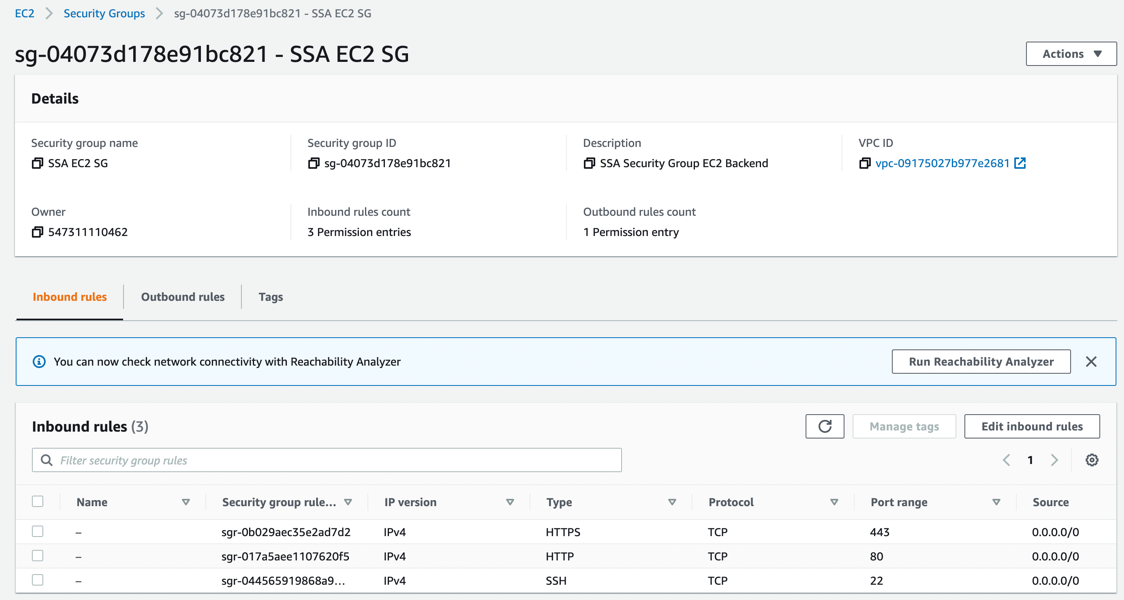
EC2:



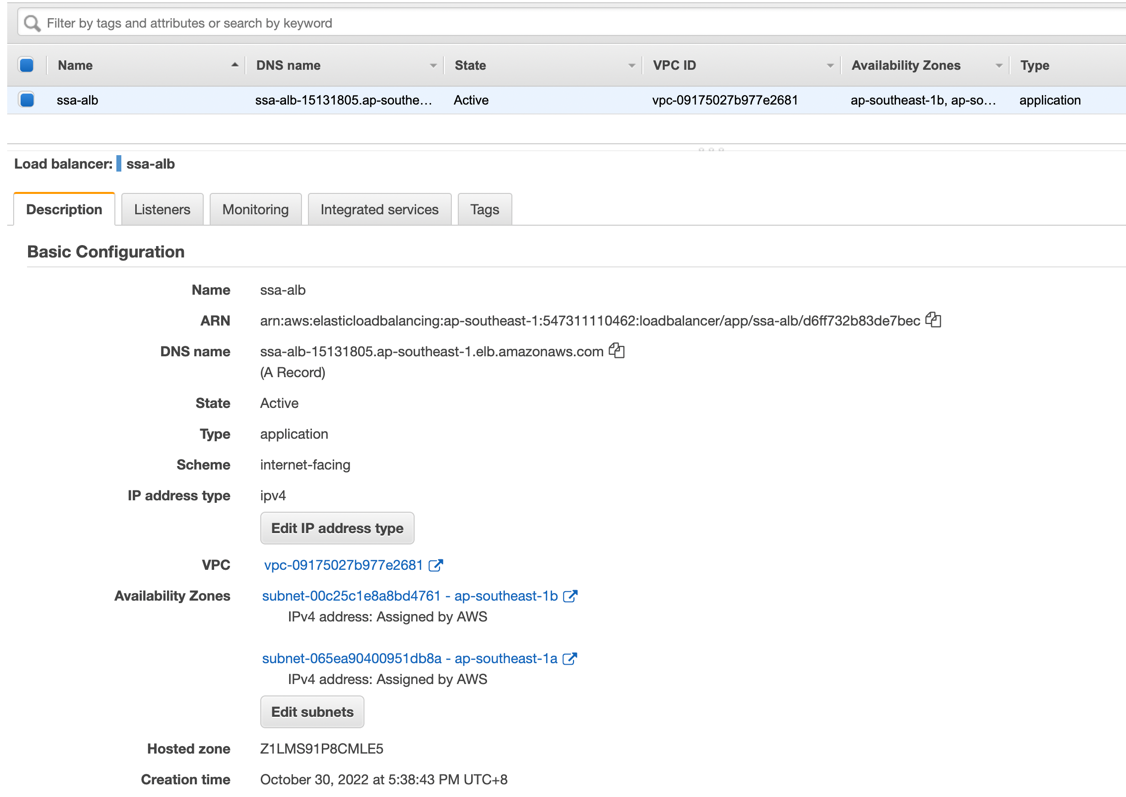


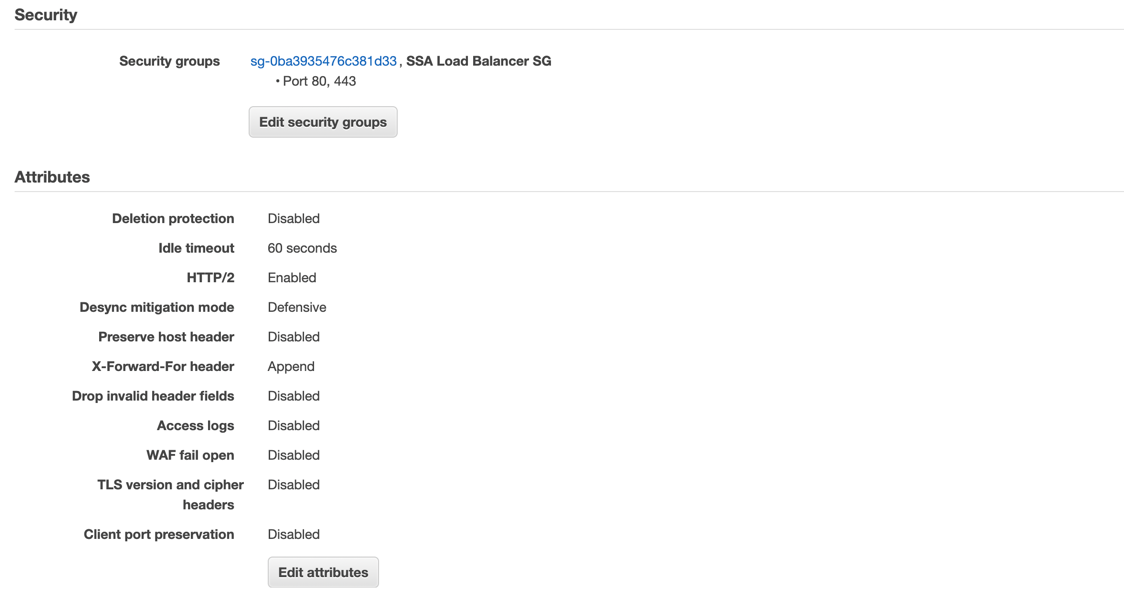


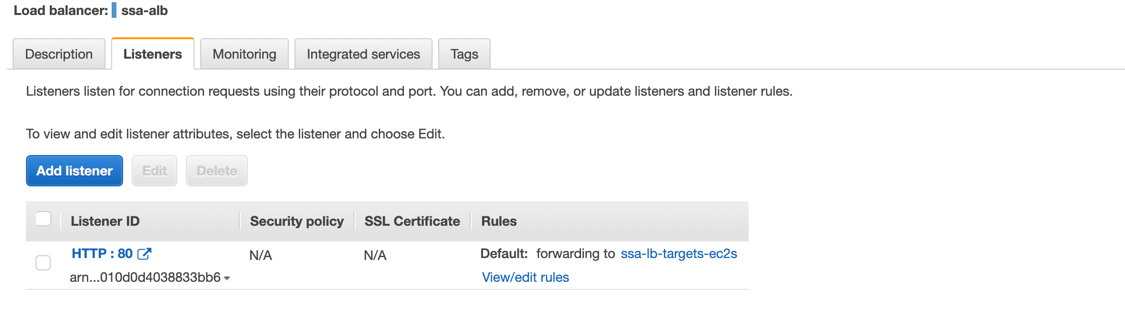
Security Group:



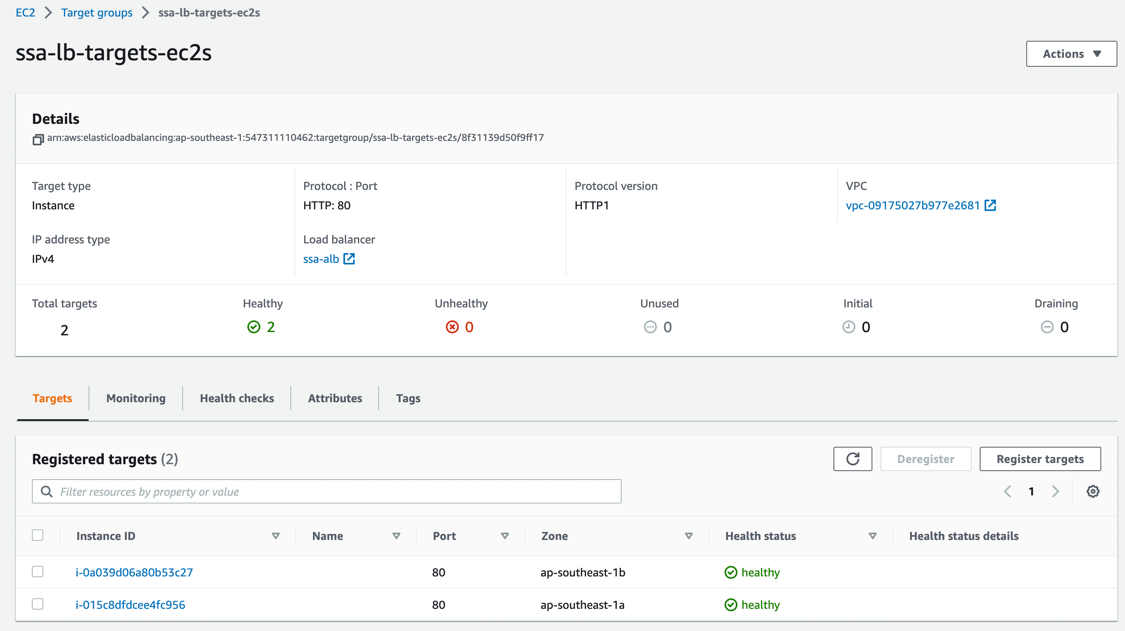
ALB:







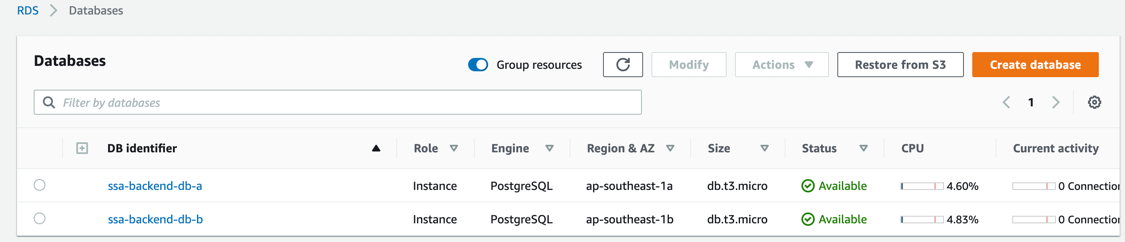
ALB Target Groups:



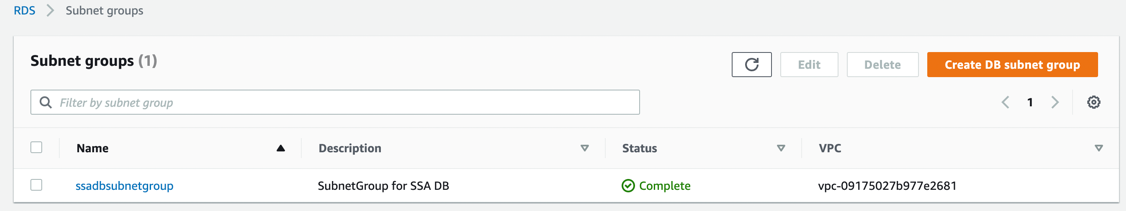
ALB Security Group:

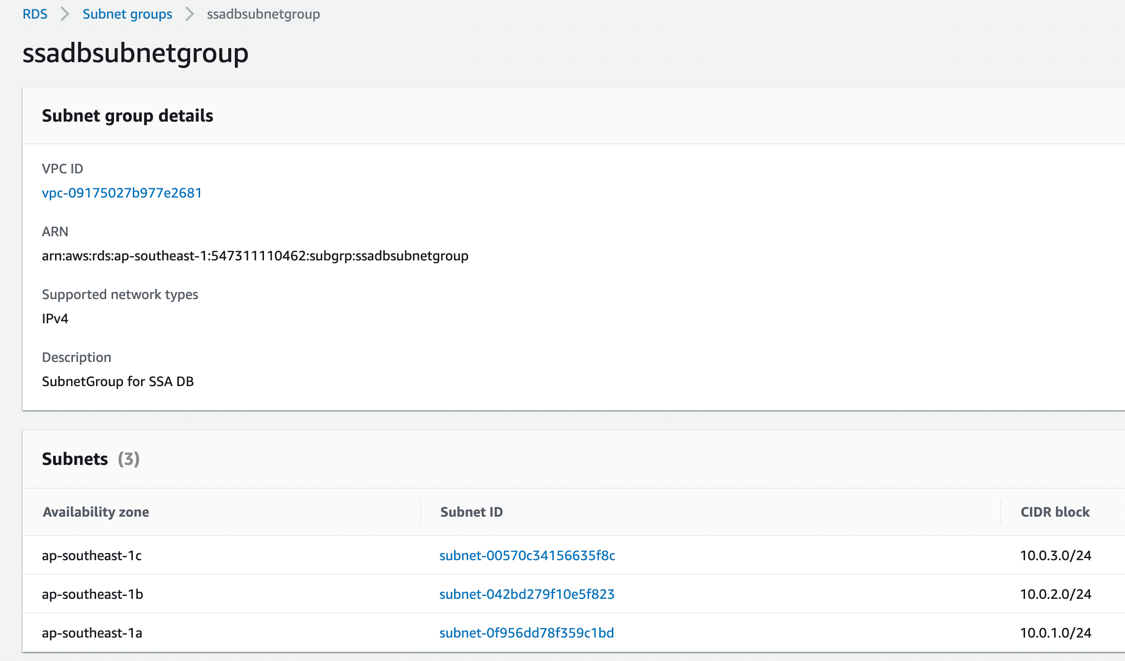


AWS **RDS** detail configure;

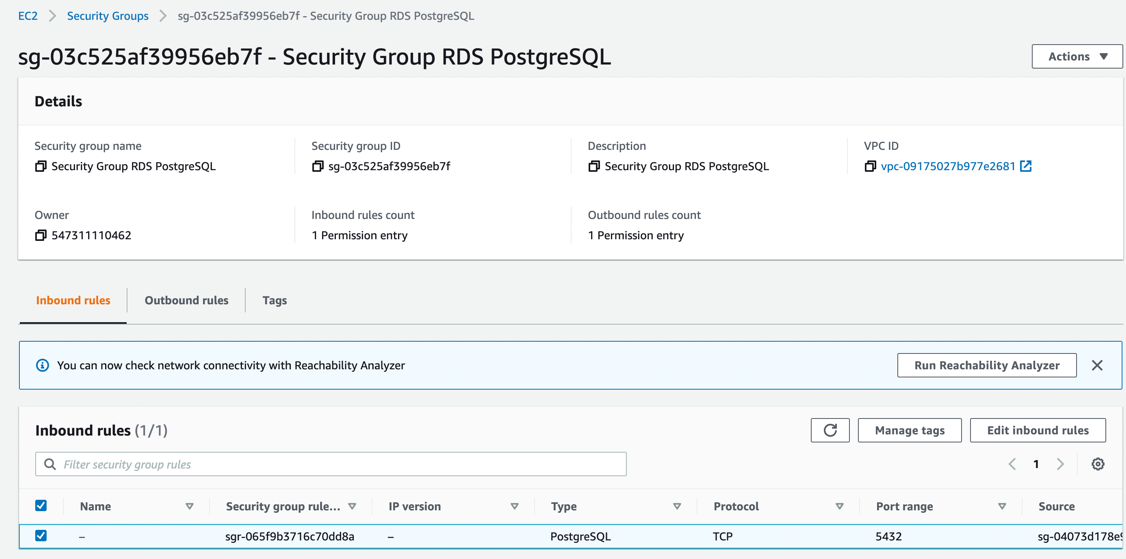


RDS Subnet Group:





RDS Security Group:



1. Protect the website backup in s3 backet

Create the a schedule cron job in each EC2 instance with the following AWS backup script: ~/aws\_backup.sh

**!**#**/**bin**/**bash

load\_date**=$(date +'%Y-%m-%d-%H')**

#backup and upload web contents to S3

**tar** czvf **/**tmp**/**web\_content\_${load\_date}**.**tgz **/**var**/**www**/**html

#aws cli to s3

aws s3 cp **/**tmp**/**web\_content\_${load\_date}**.**tgz s3**://**my-bucket-for-ssa

**if** **[** **$?** **=** 0 **];** **then**

**rm** **-**rf **/**tmp**/**web\_content\_${load\_date}**.**tgz

**fi;**

#create EC2 EBS snapshot and upload to S3 automatically

#For attached EBS volume as primary storage on EC2 instances

EC2\_INSTANCE\_ID**=$(wget -q -O - http://169.254.169.254/latest/meta-data/instance-id)**

EC2\_VOLUME\_ID**=$(aws ec2 describe-volumes --filters Name=attachment.instance-id,Values=${EC2\_INSTANCE\_ID} | grep VolumeId | head -1 | cut -d":" -f2 | tr -d '\ ",')**

**if** **[** **$?** **=** 0 **];** **then**

#aws cli to take EBS snapshot

aws ec2 create-snapshot **--**volume-id ${EC2\_VOLUME\_ID} **--**description "This is instance ${EC2\_INSTANCE\_ID} root volume snapshot"

#use copy-snapshot to copy to another region if needed

#aws ec2 copy-snapshot --source-region us-east-1 --source-snapshot-id snap-066877671789bd71b --encrypted --kms-key-id alias/my-kms-key

**fi;**

#create RDS snapshot annd upload to S3, this part can be in any one of the EC2

aws rds create-db-snapshot **--**db-instance-identifier ssa-backend-db-a **--**db-snapshot-identifier ssa\_db\_snapshot\_${load\_date}

aws rds start-export-task **--**export-task-identifier my-snapshot-export-ssa-db\_${load\_date} **--**source-arn arn**:**aws**:**rds**:**ap-southeast-1**:**547311110462**:**snapshot**:**rds**:**ssa\_db\_snapshot\_${load\_date} **\**

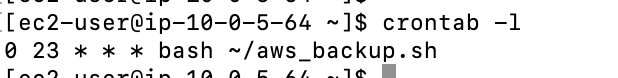
**--**s3-bucket-name my-bucket-for-ssa **--**iam-role-arn iam-role **--**kms-key-id alias**/**my-kms-key

EC2 cron job setup:

sudo service crond start  
crontab -e # insert line below and save

0 23 \* \* \* bash ~/aws\_backup.sh

Verify with command: crontabl -l



This allowed the aws\_backup.sh script to be run at 23:00 daily to perform all the required backup.

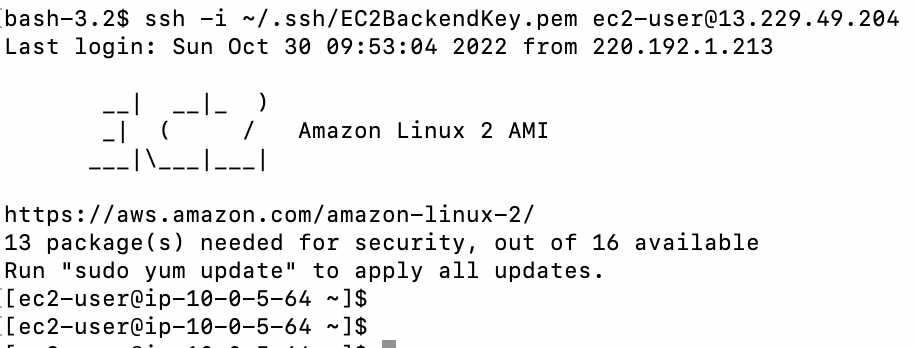
1. Based on your own option, enable the **3** necessary security service, exp **WAF**, **guard duty, security hub**… - See details in step 2 section.

Please back up the environment, we need you to show your deployment.

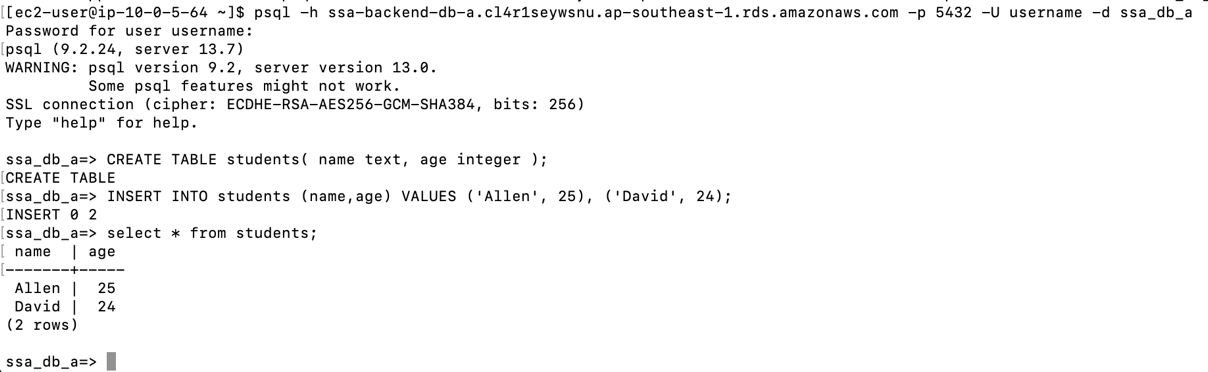
Second, you don’t need to develop a real application to connect the RDS. But you need to show the EC2 can login to the RDS, based on your ACL control.

Validation

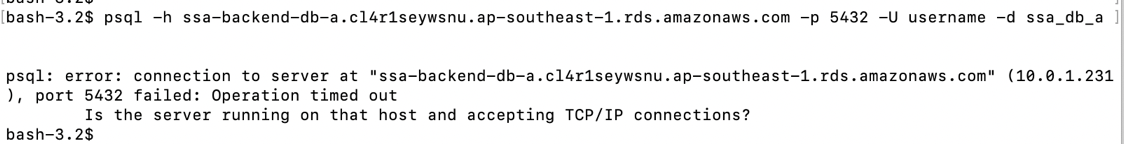
EC2 SSH connectivity



EC2 to RDS connectivity



RDS connectivity from outside of EC2



Step 2 – Solution document (no more than 6 pages totally)

1. Show your deployment topology

Using the attached ansible playbook roles to deploy the SSA application to AWS

This Ansible playbook includes roles for (including its security group):

- VPC

- EC2

- RDS (PostgreSQL)

- S3

- Httpdserver

This playbook spins up a VPC and create private and public subnets. An EC2 instance gets created in AZ a and b where Web app EC2 will be installed and placed in a public subnet for each AZ a and b. This public subnet will be accessible from the internet. Because the database must only be accessible from the EC2 instance and not from the outside world it is good practice to have it decoupled from the internet. On RDS, two postgres database instances get created for each AZ 1a (master) and 1b (replica) . They will be placed into a subnet group of three private subnets (from each AZ) allowing failover switches into another region in case of data center fail-outs for cross region AZ High Availability.

#1: Create an IAM User with Programmatic Access for each role of user with the listed IAM policies

1. The Ansible roles VPC, RDS, EC2, and S3 need programmatic access to your AWS account. To do so generate a new IAM User with Programmatic access and attach the below policy list. You may not need the FullAccess types - feel free to limit the corresponding rights by shrinking them down to a minimum. If the user is not needed anymore delete it or just make the security key inactive.

IAM Users:

Cloud\_admin:

AmazonVPCFullAccess

AmazonEC2FullAccess

AmazonEC2FullAccess

Web\_admin:

ec2\_only\_access

DevOps:

AmazonRDSFullAccess

#2: Create A Key Pair

Within EC2 management console create a private key file and store it in a secure location. This key will allow Ansible to access the EC2 instance via SSH.

#3: Configure The Ansible Variables

You may want to configure the following variables first before running the play:

1 ./ansible.cfg:

- Specify which inventory you want to use (development, staging, or production)

- Point to the private key file generated above

2 ./inventories/\< development \>/group\_vars/all/credentials.yml:

- Specify the AWS credentials for the RDS database (username and password) and the generated access key and the corresponding secret.

3 ./inventories/\< development \>/group\_vars/all/main.yml:

- Global variables for AWS which are used roles-wide like region, database settings, and S3 bucket name

4 ./roles/\<ec2|rds|s3|ssa|vpc|httpdserver\>/defaults/main.yml:

Each role has its own configuration. Check this variable definition if they fit your needs and change them if needed.

#4: Secure Your Credentials Using Ansible-vault (optional but recommended)

The AWS credentials access key/access secret and the RDS database username and password are stored as mentioned in ./inventories/< development >/group\_vars/all/credentials.yml. It is strongly recommended to create an Ansible vault and encrypt this file instead of having the credentials in cleartext laying around.

#5: Run The Playbook With Asking For The Vaults Password\*\*

```bash

./site.yml --ask-vault-pass

```

If the whole deployment went through successfully (takes up to 10–15 minutes) open your browser and open

http://<public-dns-of-alb>:80

2. Tell us why you enable these 3-security service.

Based on list of available AWS security service, I would enable these 3 necessary security services below:

Detection: Security Hub combines information from all the services like IAM, GuardDuty, Inspector, AWS Config, CloudTrail and etc in a centralized and unified view. It collects data from all security services from multiple AWS accounts and regions, making it easier to get a complete view of your AWS security posture. Additionally, Security Hub supports collecting data from third-party security products and supporting for industry recognized security standards including the CIS, PCI DSS and other standards. Security Hub is essential to providing your security team with all the information they may need. Combine Security Hub with AWS Organizations for the simplest way to get a comprehensive security overview of all your AWS accounts.

Network and application protection: AWS Shield for DDoS protection

AWS Shield protects all your applications running on AWS from DDoS attacks. This essentially protects the perimeter of your application.

AWS Web Application Firewall (WAF) for Filter malicious web traffic and helps protect against web applications being exposed to the internet and therefore vulnerable to exploit. It can be used to monitor and protect applications and APIs built on services such as CloudFront, API Gateway, and AppSync. You can block access to your endpoints based on different criteria such as the source IP address, the request’s origin country, values in headers and bodies, and more (i.e, you can enable rate limiting, only allowing a certain number of requests per IP). It can detect and mitigate attacks like SQL injections, XSS. It comes with default rules, but your team can also customize your own settings.

Data protection: services are required to keep your data secure and ensure that data is not lost during transfer or corrupted in storage. These services typically include features for encryption, data transmission.

AWS Certificate Manager: Provision, manage, and deploy public and private SSL/TLS certificates, to Protect and secure your website and connected resource.

**Strong security enhancement solution:**

To ensure as complete security coverage as possible, I would focus on the following areas with these security service:

**Identity and access management** - Identity services enable you to define credentials, distribute and limit privileges, manage user and application roles, and identify users. These services enable you to control what resources users and applications can access and how they can manipulate them.

Services offered by AWS include:

* Identity and Access Management (IAM) - Securely manage access to services and resources
* IAM Identity Center (Formal SSO) - Manage workforce access across AWS accounts and apps
* Amazon Cognito - Identity management for your apps
* Resource Access Manager - Secure service to share AWS resources
* AWS Directory Service - Managed Microsoft Active Directory service

**Infrastructure protection** - Infrastructure protection services enable you to manage the security of configurable infrastructure, such as network connections. Most infrastructure in AWS is managed and secured for you, however, security is a shared responsibility to secure your application in AWS. You should protect networks and endpoints by traffic filtering, access controls, and IP restrictions by leveraging services below.

Services offered by AWS include:

* AWS Shield - DDoS protection
* AWS Web Application Firewall (WAF) - Filter malicious web traffic, blocking common attacks
* AWS Firewall Manager - Central management of firewall rules across apps and accounts.
* AWS Network Firewall - Deploy network security across VPCs to block common network threats

**Data protection** - services are required to keep your data secure and ensure that data is not lost during transfer or corrupted in storage, including workloads, and accounts from unauthorized access. These services typically include features for encryption, access restrictions, and key management, data duplication, and data monitoring.

Services offered by AWS include:

* Secrets Manager - Centrally manage the lifecycle of secrets
* Certificate Manager - Provision, manage, and deploy public and private SSL/TLS certificates
* CloudHSM - hardware security module manages single-tenant hardware security modules (HSMs) on AWS
* Key Management Service (KMS) - Create and control keys to encrypt or digitally sign your data
* Amazon Macie - Discover and protect your sensitive data at scale in S3

**Vulnerability and Threat detection**—services to help you identify threats and continuous monitor to track system and security events. These services include alerting, logging, and behavior analytics. Threat detection and monitoring services provide visibility into your deployment operations and help you identify suspicious events, and also help you ensure that performance standards are maintained and provide insight for optimizations.

Services offered by AWS include:

* CloudTrail - Track user activity and API usage
* CloudWatch - Observe and monitor AWS resources and applications in the cloud with real time logging
* Inspector - Automated and continual vulnerability management at scale
* GuardDuty - Protect your AWS accounts with intelligent threat detection
* AWS Config - Assess, audit, and evaluate configurations of your resources
* AWS Security Hub - Automate AWS security checks and centralize security alerts for the above services

**Compliance and data privacy**—services to help audit and enforce compliance and privacy measures. These services include automated compliance checks and encryption. Compliance and data privacy tools enable you to ensure that you are meeting compliance standards and maintain the security of sensitive information. These services require you to understand what regulations apply to your data and what those regulations require. And protect sensitive data (such as PII) from exposure.

Services offered by AWS include:

* AWS Artifact - On-demand access to AWS’ compliance reports.
* Audit Manager - Continually audit your AWS usage to simplify risk and compliance assessment
* KMS and CloudFront - Leverage field-level encryption integrates with key management

Overall, gaining visibility over your AWS environment is critical for enhancing security in AWS. By leveraging the security service above, we can build a robust and secure environment for application in the cloud with proper monitoring tool that can help you to identify and alert anomalies in the environment effectively.