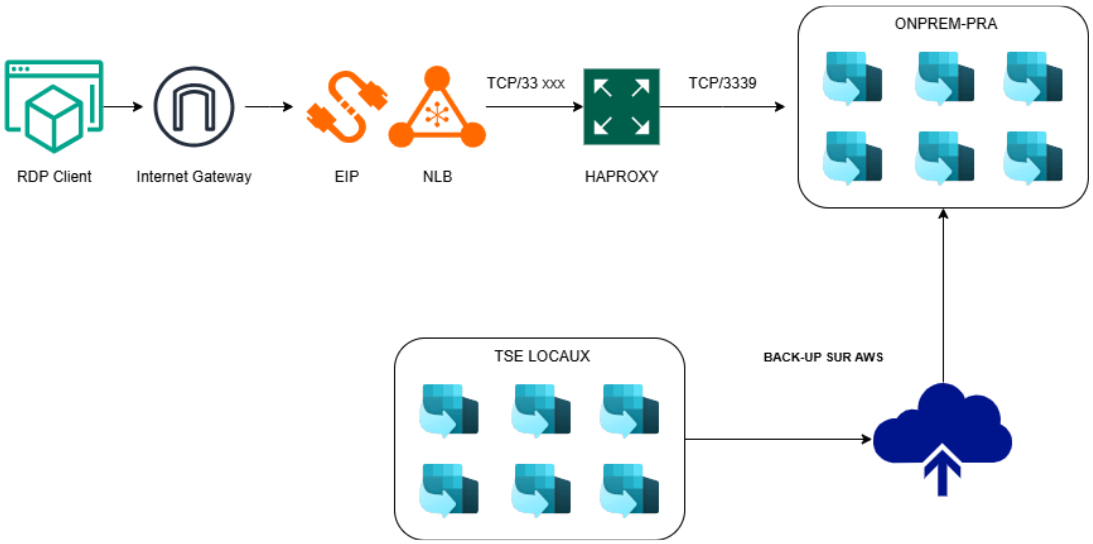
Rapport anonymisé pour Github



1. Contexte et objectifs

Afin de garantir la continuité de service en cas de panne majeure sur les serveur TSE de nos clients, j'ai été chargé de concevoir et déployer un **Plan de Reprise d'Activité (PRA)** sur AWS.

Ce projet visait à :

- Mettre en place une infrastructure de secours cloud (onprem-pra),
- Reproduire les conditions de production via un serveur TSE de test (srvtse005),
- Valider la faisabilité technique d'une bascule via un **drill (test simulé)**,
- Automatiser l'ensemble du déploiement via **Terraform**.

2. Déroulement du projet

2.1. Création des Security Groups

La première étape a consisté à créer les **Security Groups** suivants :

- onprem-pra : groupe principal du PRA sur AWS
- SRVTSExxx : groupe spécifique pour le serveur de test TSE

```
accounts > production > security-groups > ! security-groups.yaml > ...
149   security-group:
581   ########## PRA-ONPREM-SG ##########
582     onprem_pra_sg:
583       description: Security Group for onprem_pra_sg
584       exists: false
585     SRVTSE[redacted]_sg:
586       description: Security Group for SRVTSE[redacted]_sg
587       exists: false
```

👉 **Pourquoi commencer par les Security Groups ?**
Les security groups sont des **prérequis réseau** pour tout ce qu'on déploie ensuite (instances EC2, NLB, etc.).
Ils définissent **qui peut accéder à quoi** (ports, protocoles, IP autorisées).
Terraform a besoin de ces ressources pour pouvoir attacher les règles au fur et à mesure du déploiement.

2.2. Création des EIP pour onprem_pra

🎯 Objectif
Assigner des **IP publiques fixes (Elastic IP)** aux différentes zones de disponibilité (AZ) dans lesquelles le **NLB "onprem-pra"** est déployé.
Cela garantit :

- Une **adresse IP stable** même en cas de recréation du NLB,
- La **redondance** sur plusieurs AZ (3 EIP = 3 AZ),
- Une **résilience réseau optimale** dans un scénario de Plan de Reprise d'Activité.

Elément	Rôle
aws_eip	Réserve une IP publique fixe dans AWS
domain = "vpc"	Attache l'EIP à un composant dans un VPC (ici NLB)

Elément	Rôle
Redondance	3 EIP = 3 AZ pour tolérance aux pannes

> ⓘ **Etape 1 pour EIP**
>
> → Ici, on demande à AWS de **réserver des IP publiques fixes** qui nous appartiennent, avant de les assigner à quoi que ce soit.

2.3. Mise en place de l'architecture principale avec Terraform

Dans le fichier main.tf, j'ai défini les ressources clés suivantes :

Elément	Type	Rôle dans l'infra
provider "aws"	Bloc	Déclare la région et les tags
data "aws *"	Données	Récupère les ressources AWS existantes
Sécurity Groups globaux	Réseau	Appliquer les bonnes règles de sécurité
IAM Instance Profile	Sécurité	Donner les bon droits aux EC2
VPC / Subnets *	Réseau	Placer les instances dans les bonnes zones
Route53 Zone	DNS	Ajouter des enregistrements DNS
HAProxy Instances	Compute	Permet de connecter NLB + SRV TSE

- **VPC** : C'est le réseau privé virtuel dans AWS. Toutes les ressources sont déployées à l'intérieur de ce VPC
- **AWS Subnet** Identifier précisément les subnets à utiliser dans chaque AZ, selon leur rôle
  - private-az1/2/3 → subnets privés,
  - public-az1/2/3 → subnets publics,
  - publics → tous les subnets publics du VPC principal.

  📌 Ces blocs permettent à Terraform de **ne rien recréer**, mais de réutiliser **l'existant** pour attacher les ressources (EC2, NLB, etc.).

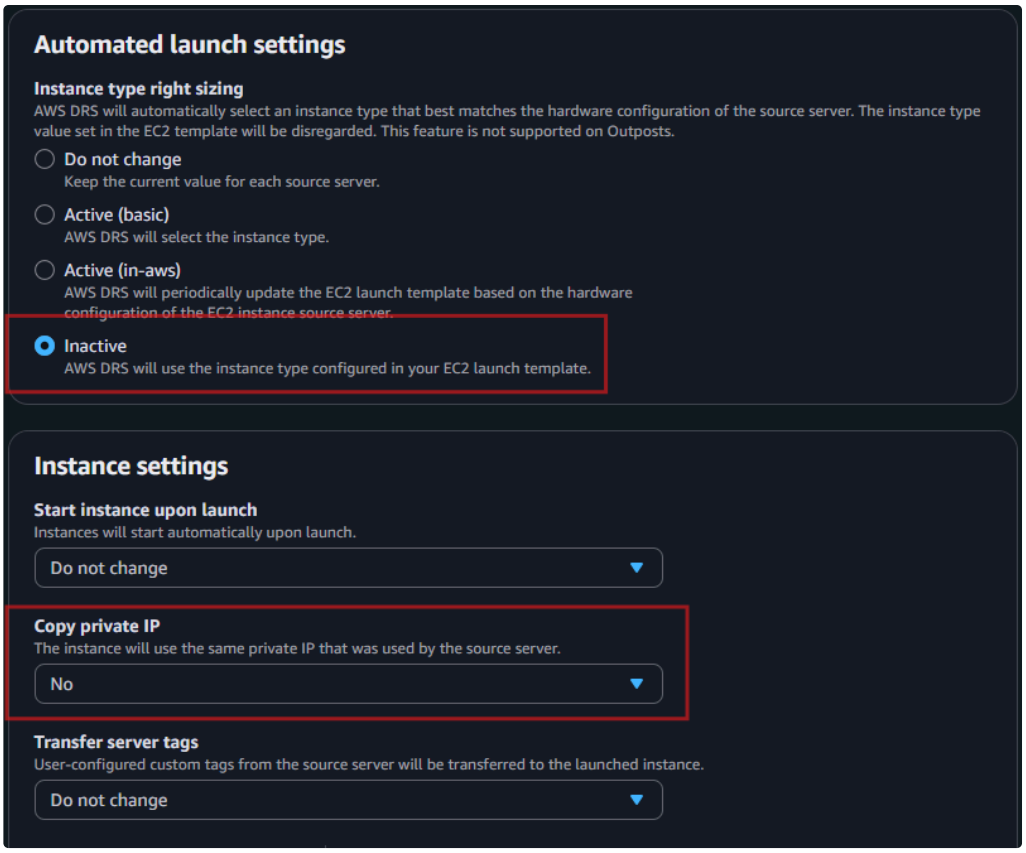
2.3. Configuration de la Launch Template

💡 **Objectif** : faciliter la réplication d'instances TSE à la demande.
Création d'un **modèle de lancement EC2** contenant :

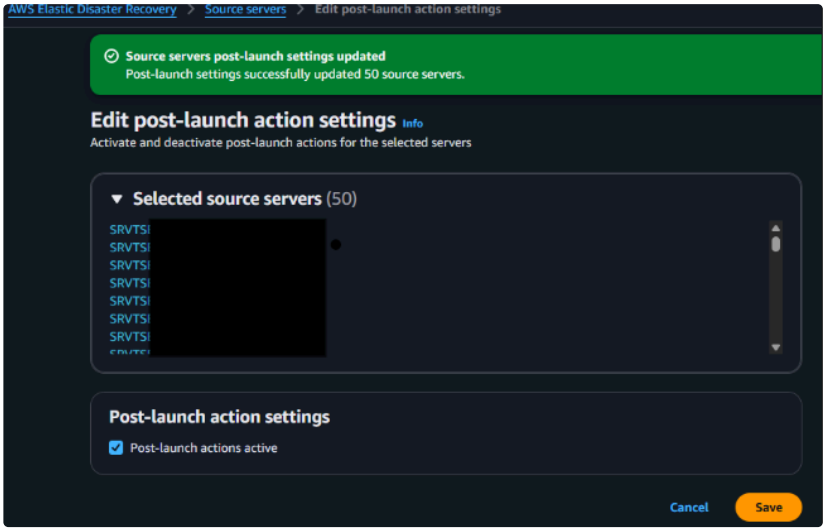
Element	Rôle
data "aws_security_group"	Réutilisation des SG existants
resource "aws_launch_template"	Prépare une machine PRA TSE prête à démarrer
block_device_mappings	Ajoute et configure 2 disques EBS
iam_instance_profile	Autorise l'instance à utiliser SSM, CloudWatch
network_interfaces	Positionne l'instance dans un subnet privé, avec les bons SG
tag_specifications	Marque les ressources pour la gestion automatique via DRS

> ⚠ **Point de vigilance**
>
> Avant de faire un apply des launch template , il est essentiel de les importer via
> tf import aws_launch_template.srvtsexxx lt-xxxxxxxxxxxx
> Afin de ne pas les recréer , ce qui engendrerait une erreur

PREREQUIS DRS LAUNCH SETTINGS : C'est 2 prérequis doivent être **ABSOLUMENT** cochés afin que notre version du code terraform soit bien gardée par défaut !



PREREQUIS DRS POST-LAUNCH SETTINGS : Afin que l'agent SSM s'installe et que l'on puisse se connecter au serveur via fleet manager



2.4. Déploiement du Network Load Balancer (NLB)

Le **Network Load Balancer** a été configuré pour permettre une **redirection des flux RDP** (et autres protocoles) vers les serveurs PRA.

Action	Rôle
Attribution de 3 EIP	Permettre une haute disponibilité 1 IP/AZ + IP public fixe pour accès stable
Création ressource nlb-onprem	Avec 3 subnet_mapping pour couverture multi-AZ
Création ressource nlb_onprem_pra_sg	Rattacher NLB au security group dédié
Enregistrement DNS Route53	Le client se connecte via remote-onprem.exemple.com:3300X
Listener NLB + ouverture de port client	Le NLB écoute sur le port 3300X pour ce client.
resource "aws_lb_target_group_attachment"	Permet de répartir les connexions entrantes entre les deux HAProxy, pour la redondance.

👉 Pourquoi un NLB ?

- Il supporte les protocoles TCP/UDP
- Il est capable de gérer de grosses charges avec une latence faible
- Il n'effectue pas de termination SSL/TLS → parfait pour des connexions RDP/Anydesk

ⓘ Etape 2 pour EIP

→ Ici, on **assigne explicitement** chaque EIP au **NLB** pour qu'il ait **une IP fixe par AZ**.

2.5. Déploiement du HAProxy

Le **HAProxy** agit comme proxy TCP entre le NLB et les instances TSE.

- Règles **ingress** et **egress** configurées pour autoriser le trafic avec les bonnes IPs et ports,
- Rattachement de srvtse005 au HAProxy pour tester la redirection de session,
- HAProxy déployé avec un SG filtrant uniquement les flux nécessaires.

2.6. Intégration en environnement de production

Dans le fichier de prod :

- Ajout du bloc data "aws_security_group" "onprem_pra_sg" pour référencer le SG PRA existant,
- Réutilisation du HAProxy de prod en l'autorisant à dialoguer avec les composants PRA.

3. Phase de test : Recovery Drill

3.1. Définition d'un recovery drill

Un *recovery drill* est un test de PRA réalisé dans des conditions contrôlées (non réelles), permettant de valider toutes les étapes de reprise.

3.2. Étapes réalisées

1. **Désactivation des services sensibles** :
   - Anydesk (télémaintenance)
   - AppY (agent de mise à jour de prod)

→ pour éviter toute interférence entre prod et PRA.

2. **Vérification de l'environnement logiciel** :
   - Lancement de l'application métier sur l'instance PRA,
   - Vérification de la base de données restaurée.
3. **Connexion via RDP** :
   - Test d'accès distant fonctionnel via le NLB et HAProxy,
   - IP statique accessible,
   - remote-onprem.exemple.com:3300X accessible.

4. Résultat du projet

✅ L'objectif principal a été atteint :

- L'environnement PRA est déployé automatiquement via Terraform,
- L'accès RDP est fonctionnel via NLB et HAProxy,
- L'application métier est disponible et la base de données accessible,
- Le test de drill a validé l'ensemble du processus de reprise.