

Documentation du projet CLOUD-AWS

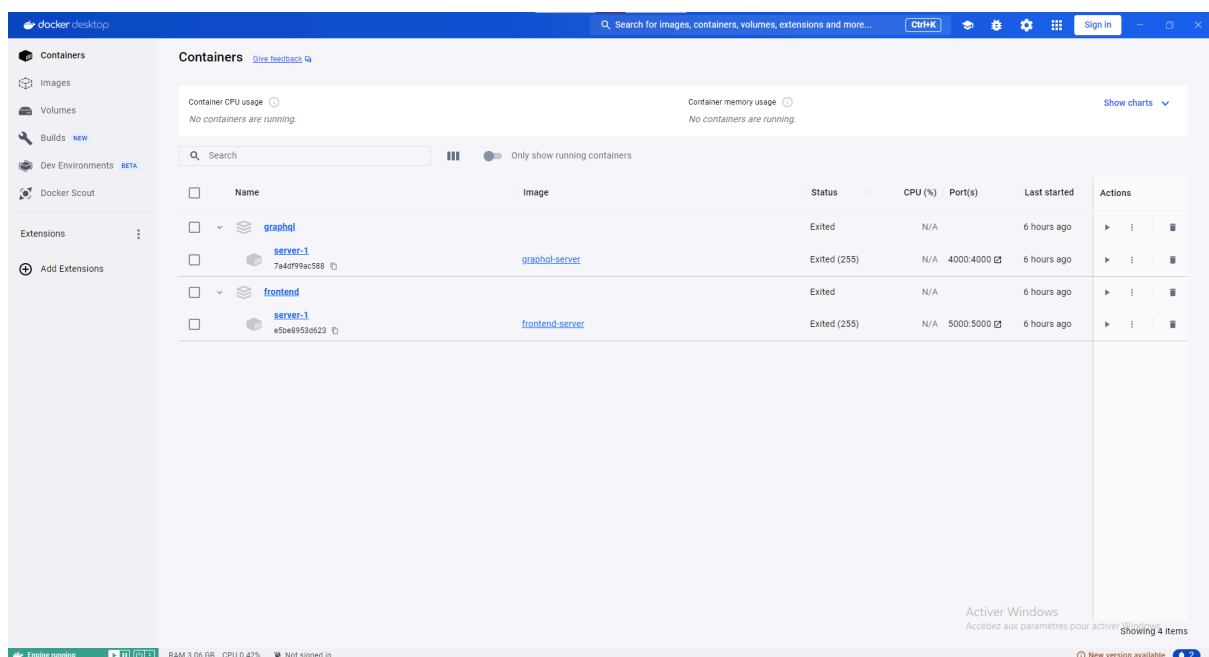
Docker : Conteneurisation

La base de ce projet provient d'un projet en GraphQL que j'ai fait cette année. Un projet GraphQL en Backend codé en TS avec une base de donnée Sqlite, et en Frontend un projet en Vue 3 (typé TS).

J'ai rencontré notamment une difficulté avec la conteneurisation de l'application GraphQL, en particulier par le fait de son nommage ts, j'ai du débbuger et essayer de faire fonctionner correctement mes Dockerfile et Docker-compose, et j'ai du rajouter des CORS origin pour la bonne communication après le déploiement.

J'ai donc mis en place deux docker-compose séparant ainsi le Frontend et le Backend en déploiement Docker.

Il suffit donc de simplement lancer la commande "docker-compose --build -d"

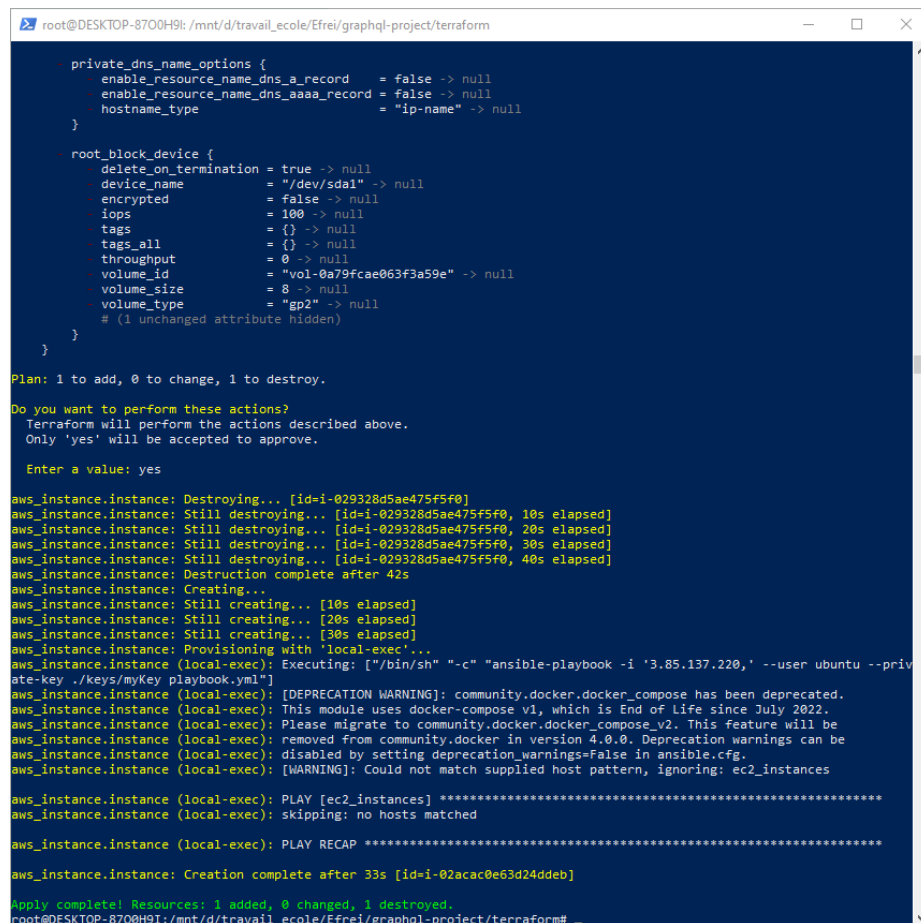


Orchestration avec Docker Swarm

Pour cette étape j'ai tenté de manipuler des Docker Swarm avec multipass, et manuellement, je n'ai pas trop compris l'intérêt global ou le but final de l'utilisation de ce procédé, ou plutôt je ne sais pas comment ce procédé se situe par rapport au CI/CD. J'ai passé cette étape malheureusement car je n'avais pas assez de temps pour me concentrer sur cela, j'ai préféré essayer d'avoir un projet déployé.

AWS EC2 : Préparation du Déploiement avec Terraform

J'ai décidé de me concentrer sur l'étape de déploiement d'instance EC2 en utilisant Terraform. J'ai essayé de comprendre en lisant la documentation et quelques "guides", il y avait malheureusement peu d'exemples utilisant ces technologies, j'ai tout de même pu mettre en place les fichiers nécessaires pour l'utilisation de Terraform et Ansible, j'ai eu tout de même beaucoup de difficulté pour la mise en place de ceux-ci tant bien par ma méconnaissance que par la difficulté de compatibilité avec Windows, j'ai passé une grande partie à déboguer et essayer de faire fonctionner Ansible surtout sur Windows. Ansible n'étant utilisable qu'avec WSL sur Windows, j'ai dû passer un moment pour le mettre en place.

A screenshot of a terminal window with a dark blue background and white text. The window title is 'root@DESKTOP-8700H9I: /mnt/d/travail_ecole/Efrei/graphql-project/terraform'. The terminal shows Terraform configuration for an AWS EC2 instance, followed by a plan and an apply command. The output shows the destruction of an existing instance and the creation of a new one. The new instance is provisioned with Ansible, which runs a playbook. The terminal output includes various status messages, deprecation warnings, and a final 'Apply complete!' message.

```
private_dns_name_options {
  enable_resource_name_dns_a_record = false -> null
  enable_resource_name_dns_aaaa_record = false -> null
  hostname_type = "ip-name" -> null
}

root_block_device {
  delete_on_termination = true -> null
  device_name = "/dev/sda1" -> null
  encrypted = false -> null
  iops = 100 -> null
  tags = {} -> null
  tags_all = {} -> null
  throughput = 0 -> null
  volume_id = "vol-0a79fcae063f3a59e" -> null
  volume_size = 8 -> null
  volume_type = "gp2" -> null
  # (1 unchanged attribute hidden)
}
}

Plan: 1 to add, 0 to change, 1 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.instance: Destroying... [id=i-029328d5ae475f5f0]
aws_instance.instance: Still destroying... [id=i-029328d5ae475f5f0, 10s elapsed]
aws_instance.instance: Still destroying... [id=i-029328d5ae475f5f0, 20s elapsed]
aws_instance.instance: Still destroying... [id=i-029328d5ae475f5f0, 30s elapsed]
aws_instance.instance: Still destroying... [id=i-029328d5ae475f5f0, 40s elapsed]
aws_instance.instance: Destruction complete after 42s
aws_instance.instance: Creating...
aws_instance.instance: Still creating... [10s elapsed]
aws_instance.instance: Still creating... [20s elapsed]
aws_instance.instance: Still creating... [30s elapsed]
aws_instance.instance: Provisioning with 'local-exec'...
aws_instance.instance (local-exec): Executing: ["/bin/sh" "-c" "ansible-playbook -i '3.85.137.220,' --user ubuntu --private-key ~/.keys/myKey playbook.yml"]
aws_instance.instance (local-exec): [DEPRECATION WARNING]: community.docker.docker_compose has been deprecated.
aws_instance.instance (local-exec): This module uses docker-compose v1, which is End of Life since July 2022.
aws_instance.instance (local-exec): Please migrate to community.docker.docker_compose_v2. This feature will be
aws_instance.instance (local-exec): removed from community.docker in version 4.0.0. Deprecation warnings can be
aws_instance.instance (local-exec): disabled by setting deprecation_warnings=False in ansible.cfg.
aws_instance.instance (local-exec): [WARNING]: Could not match supplied host pattern, ignoring: ec2_instances

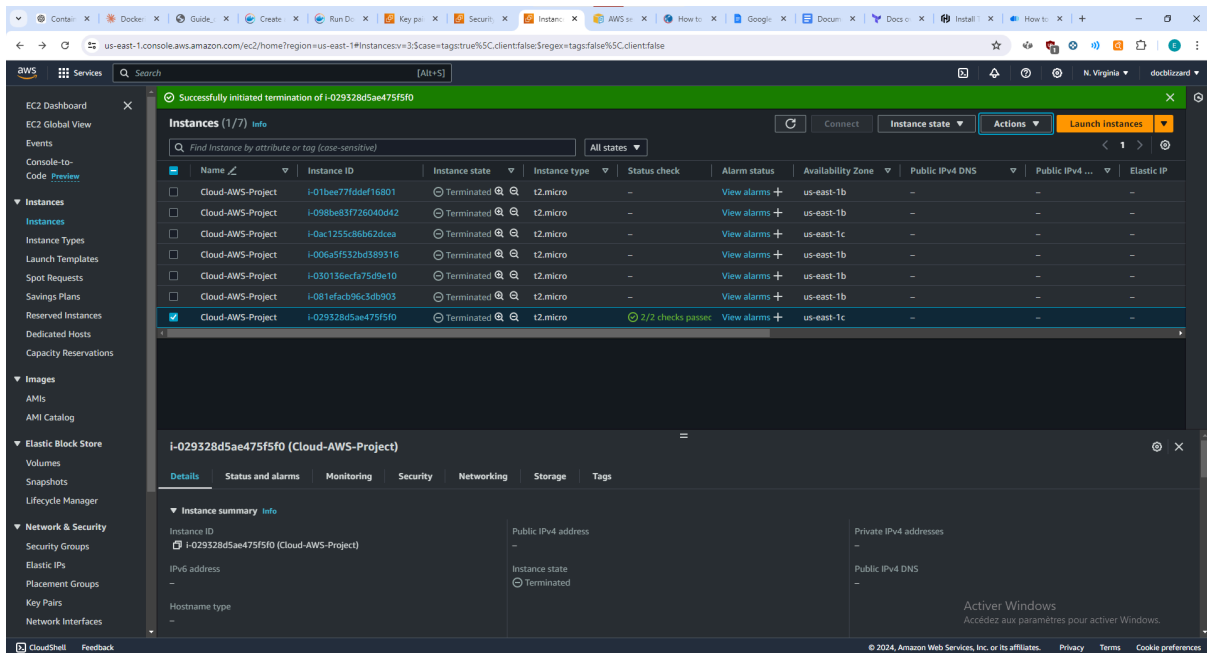
aws_instance.instance (local-exec): PLAY [ec2_instances] *****
aws_instance.instance (local-exec): skipping: no hosts matched

aws_instance.instance (local-exec): PLAY RECAP *****

aws_instance.instance: Creation complete after 33s [id=i-02acac0e63d24ddeb]

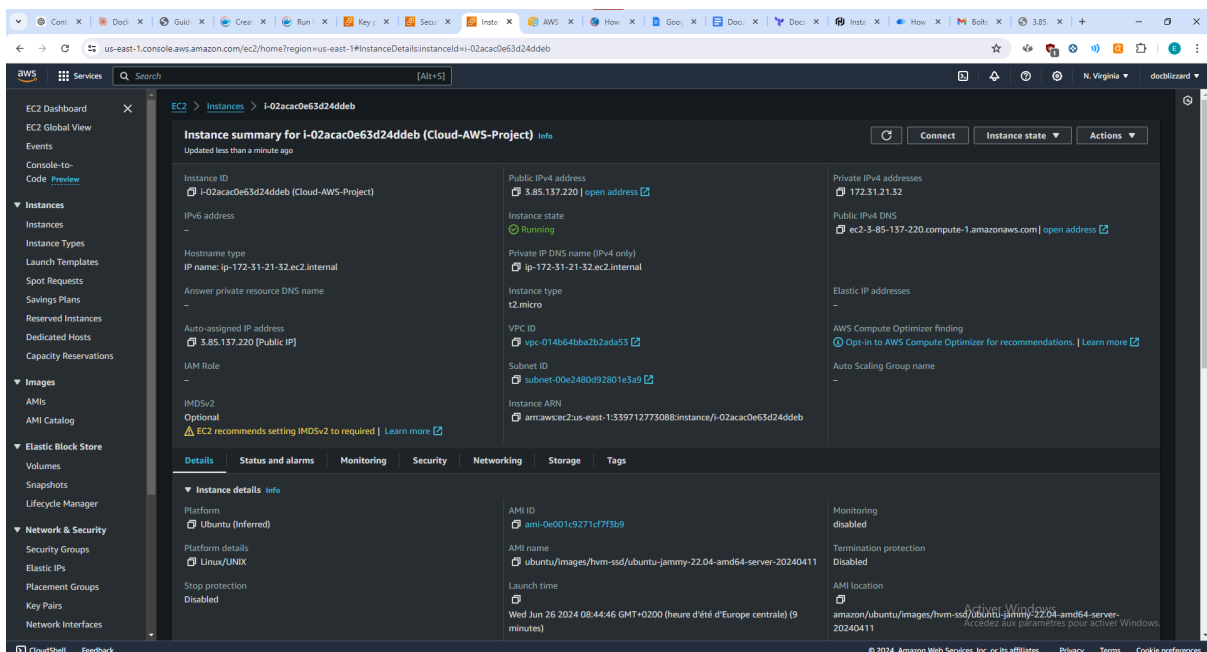
Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
root@DESKTOP-8700H9I: /mnt/d/travail_ecole/Efrei/graphql-project/terraform#
```

Utilisation de WSL pour lancer Terraform et Ansible



Instance EC2 sur AWS

J'ai pu après beaucoup de débogage lancer une instance "réussie" sur aws, malheureusement celle-ci ne fonctionne pas correctement et je n'ai pas eu assez de temps pour continuer et terminer au mois cette partie.



Je n'ai pas compris non plus comment intégrer les variables depuis le fichier .env pour pouvoir les réutiliser.

Conclusion:

Avec le nombre de projets qui sont arrivés en même temps avec peu de temps, je n'ai pas priorisé le bon et j'aurais aimé au moins avoir un déploiement en ligne, je ne pensais pas que j'aurais été bloqué sur de multiples étapes, et ce qui m'a fait perdre assez de temps pour ne pas être satisfait avec mon rendu. C'est néanmoins des technologies et des procédés que je veux absolument me pencher et maîtriser.